FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)
UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

Grau en Enginyeria Informàtica
Enginyeria del Software

Motor de detecció de col·lisions multiplataforma amb disseny orientat a dades

Treball de Final de Grau - Modalitat B

Autor: Elena Aragón
Director i Institució: Marc Fernàndez (Empresa Interiorvista)
Ponent i Departament: Josep Casanovas (Departament d'Estadística i Investigació Operativa)

Data de la defensa: 28 d'abril del 2017

Abstracte

En Català

Aquest TFG (en modalitat B) es basa en la recerca, implementació i integració d'un motor de col·lisions dins del framework d'*Interiorvista*.

El projecte té la finalitat d'escollir quina tècnica de detecció de col·lisions i quin disseny és més adequat per a les necessitats actuals de l'empresa. Per assolir-ho s'ha fet un estudi entre diferents tècniques de detecció de col·lisions i s'ha realitzat una implementació eficient capaç de detectar totes les col·lisions possibles entre un conjunt d'objectes definits.

Un cop analitzades i seleccionades quines tècniques són més adequades s'ha realitzat la seva implementació. Aquesta part s'ha dividit en dos blocs molt diferenciats: el primer on s'han desenvolupat les tècniques fent ús d'un disseny orientat a objectes i el segon utilitzant un disseny orientat a dades. En aquest segon grup només s'han implementat les dues combinacions de tècniques més eficients. Per escollir quines combinacions eren les millors, s'ha realitzat una comparativa entre totes les opcions implementades al primer bloc emprant un mateix conjunt d'objectes.

Finalment s'ha implantat la tècnica i el disseny guanyador dins del framework.

El projecte va començar al març del 2016, tot assolint els requisits mínims del motor a l'abril del 2017.

Aquest projecte s'ha fet a l'empresa Interiorvista, ha tingut com a director Marc Fernàndez, enginyer responsable del departament de I+D dins de l'empresa, i com a ponent de la FIB el professor Josep Casanovas.

In English

This TFG (B modality) works on the basis of researching, developing and integrating a collision engine inside the *Interiorvista* framework.

This project aims to choose the best collision detection algorithm and the best architecture according to business requirements. To accomplish this objective, different collision detections methods have been investigated and implemented optimally, being able to detect every possible collision in a defined set of objects.

The development of the most fitting algorithms is only made when the research has finished. This section of the project is planned on two different parts: the first one includes the development of the algorithms working with an object-oriented architecture and the second part is focused on data-oriented architecture. This second development only makes use of the two best method combinations, which can be chosen comparing the results of the object oriented implementation with the same set of objects.

Finally the best algorithm and architecture have been integrated inside the framework.

This project started on March 2016, and the minimum requirements were reached on April 2017.

It has been developed at Interiorvista, directed by Marc Fernàndez, chief engineer of the R+D department inside the company, having as FIB lecturer professor Josep Casanovas.

En Castellano

Este TFG (en modalidad B) se basa en la investigación, implementación e integración de un motor de colisiones dentro del framework de *Interiorvista*.

El proyecto tiene la finalidad de escoger qué técnica de detección de colisiones y que diseño es más adecuado para las necesidades de la empresa. Para alcanzarlo se ha hecho un estudio entre diferentes técnicas de detección de colisiones y se ha realizado una implementación eficiente capaz de detectar todas las colisiones posibles entre un conjunto de objetos previamente definidos.

Una vez analizadas y seleccionadas qué técnicas son más adecuadas se ha llevado a cabo su implementación. Esta sección se puede dividir en dos bloques muy diferenciados: primero donde se han desarrollado las técnicas escogidas haciendo uso de un diseño orientado a objetos y segundo utilizando un diseño orientado a datos. En este segundo grupo solo se han implementado las dos combinaciones de técnicas más eficientes. Para escoger cuáles son mejores, se ha realizado una comparativa entre todas las opciones implementadas en el primer bloque utilizando un mismo conjunto de objetos.

Finalmente se ha realizado la implantación de la mejor técnica y diseño dentro del framework.

El proyecto empezó en marzo del 2016, y los requisitos mínimos del motor se alcanzaron en abril de 2017.

Este proyecto se lleva a cabo en la empresa Interiorvista, tiene como director a Marc Fernàndez, ingeniero responsable del departamento de I+D dentro de la empresa, y como ponente de la FIB al profesor Josep Casanovas.

Índex

1. Introducció

Aquesta memòria recull el procés de recerca i desenvolupament d'un motor de col·lisions.

Mitjançant els diferents apartats es descriuen tots els processos que s'han dut a terme al llarg de tot el projecte: des de la identificació del problema, l'estat de l'art i els objectius, passant per la recerca i implementació de les diferents tècniques de detecció de col·lisions, el testeig que s'ha realitzat fins a arribar al procés d'implantació del motor dins del framework.

La redacció de la memòria s'ha efectuat sobre la base de la documentació tècnica que es va anar creant en brut a mesura que es feia el projecte, i es va començar a redactar un cop el projecte ja estava força avançat.

Aquesta memòria reprodueix aquests documents de forma neta i organitzada per tal que sigui la més entenedora possible.

En aquest apartat s'exposa la contextualització del projecte i es fa una breu introducció als motors de físiques. Tot seguit es descriuen els stakeholders i s'estudia el mercat actual mitjançant l'estat de l'art.

1.1 Context

El desenvolupament de les noves tecnologies ha obligat a les empreses a replantejar-se la seva estratègia de negoci. Una d'aquestes tecnologies, Internet, té una gran influència en el dia a dia de la societat i en el seu comportament, els mitjans de comunicació social (social media), les aplicacions mòbils i la connectivitat digital estan canviant la naturalesa d'un dels negocis més antics del món, l'interiorisme[1].

En aquesta situació l'any 2002 va néixer l'empresa d'Interiorisme *Interiorvista*[2] amb la següent idea: "Com seria poder veure el teu espai ideal (la teva habitació, lavabo, armari…) abans de realitzar la compra, des del confort de casa seva amb només uns pocs clics". Des de l'any 2008 l'empresa s'ha especialitzat en la creació d'aplicacions per a la indústria de l'interiorisme. Un equip global de dissenyadors d'interiors, experts en 3D, professionals de UX i enginyers del Software es dediquen a desenvolupar

solucions innovadores que milloren l'experiència dels consumidors en línia i a les botigues dels clients.

Una part fonamental d'aquestes aplicacions consisteix en els planificadors 2D i 3D. *Interiorvista* treballa amb el motor Unity per realitzar aquests planificadors 3D. Aquesta eina és molt completa i dóna moltes facilitats per a treballar-hi, però a la vegada té les seves mancances (que ja es tractaran a l'apart *"1.4-Estat de l'art"*). És per aquest motiu que l'empresa ha decidit treballar en un framework propi que permeti recrear les aplicacions actuals prescindint del Unity.

Una de les necessitats d'aquest framework consisteix a disposar d'un motor de col·lisions multiplataforma, eficient i capaç de controlar totes les possibles col·lisions que es puguin donar en una escena estàtica creada amb uns objectes prèviament i correctament definits.

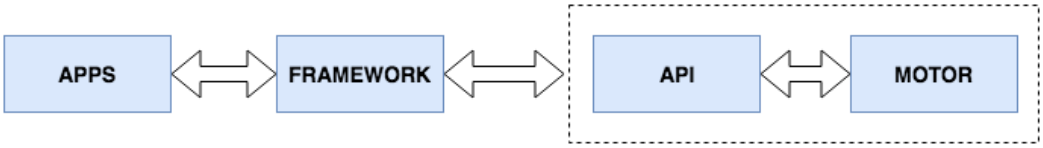


Figura 1: Diagrama de la solució

Les aplicacions faran ús d'aquest motor de col·lisions per detectar quan un usuari realitzi canvis a temps real dins del planificador. Si un usuari mou un objecte a un lloc ocupat per un altre, ha de rebre un feedback indicant-li que no pot col·locar l'element en aquell lloc.

A continuació hi ha un exemple d'una de les aplicacions de l'empresa on es pot veure l'abans i el després que un usuari realitzi el moviment d'un element. A la primera imatge l'objecte està situat en un lloc correcte i, en realitzar el moviment i l'objecte passa per sobre d'un altre element, apareix un feedback vermell indicant a l'usuari que en aquella posició no es pot col·locar l'objecte.

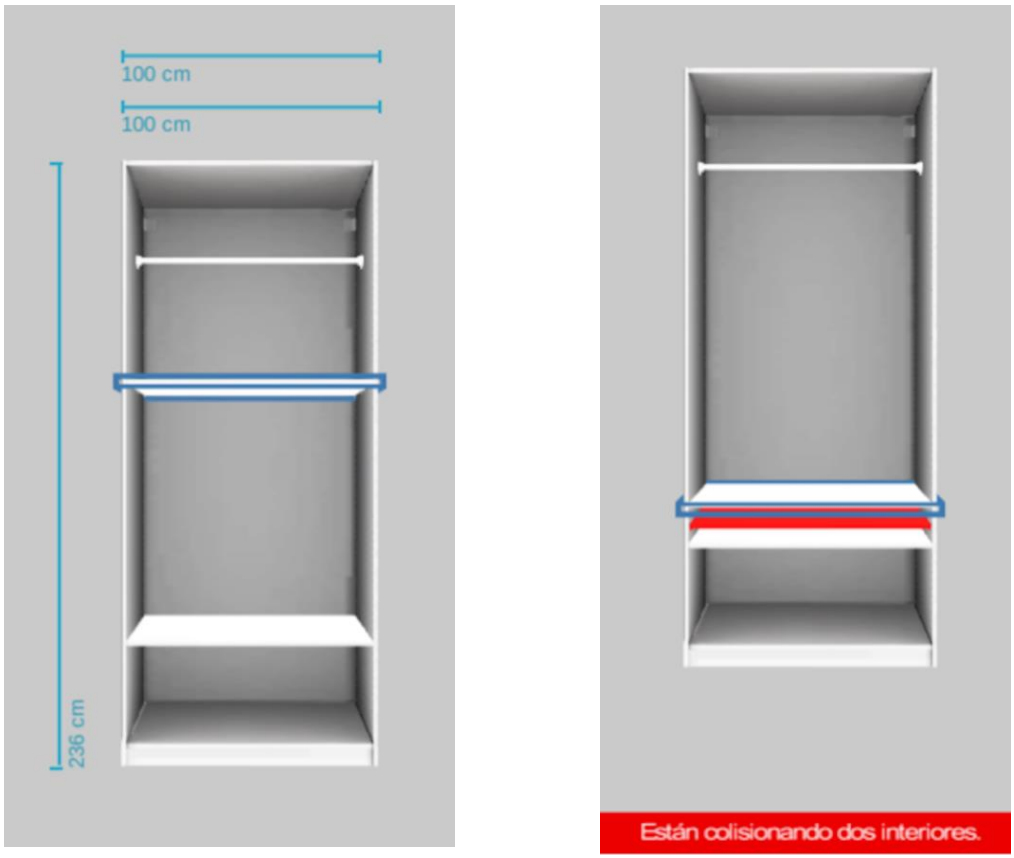


Figura 2: Exemple d'ús dins de les aplicacions de l'empresa. Esquerra: situació abans del moviment. Dreta: situació després del moviment

Així doncs aquest projecte s'encarrega de la creació del motor de col·lisions del framework d'*Interiorvista*.

1.2 El motors de simulació de físiques

Per entendre aquest projecte primer de tot cal fer una breu iniciació al món dels motors de físiques per contextualitzar l'estudi realitzat.

Un **motor de físiques**[3] (*Physics Engine*) és un software capaç de realitzar simulacions de certs sistemes físics com la dinàmica del cos rígid (incloent-hi la detecció de col·lisions), la dinàmica dels cossos tous, el moviment d'un fluid i l'elasticitat d'un cos.

Els motors de físiques es poden classificar en dues categories segons la capacitat de càlcul que requereixen:

- Simuladors de temps real
- Simuladors d'alta precisió

El *framework* que es vol implementar a l'empresa consisteix en un sistema de computació en temps real (*Real-time computer graphics*). Aquest és un subcamp dins del món dels gràfics per ordinador centrat en la producció i anàlisi d'imatges en temps real. El terme s'utilitza més freqüentment en referència als gràfics per computador interactius en 3D, típicament generats per la GPU. L'objectiu dels gràfics per computador és generar una imatge via ordinador utilitzant certs paràmetres desitjats.

Un motor de físiques està compost de diverses parts i components que queden fora de l'abast d'aquest projecte. Però sí és necessari definir que un motor de físiques té una fase de **detecció de col·lisions** (*collision detection*) i una fase posterior per calcular la **resolució de col·lisions** on es realitza la resposta física a la col·lisió (*collision resolution*). La fase de detecció de col·lisions es realitza mitjançant un component anomenat **Motor de col·lisions**.

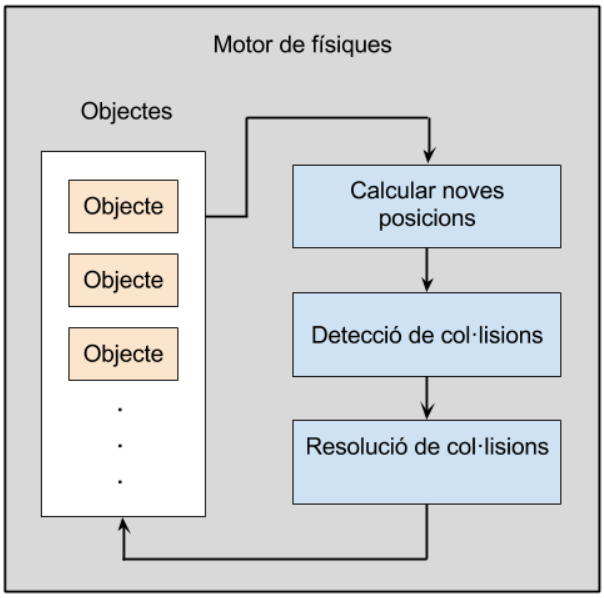


Figura 3: Diagrama d'un motor de físiques

1.3 Actors implicats (stakeholders)

A continuació es detallen tots els actors implicats en el projecte, és a dir, aquelles persones o organitzacions que tenen relació amb el projecte i que es veuran implicats sigui directa o indirectament.

Es poden diferenciar tres tipus d'usuaris afectats per aquest projecte: els usuaris de les aplicacions a les versions web, mòbil o tauleta, els treballadors de les botigues físiques dels clients i els treballadors interns de l'empresa Interiorvista.

1.3.1 Usuaris de les aplicacions per informar-se o realitzar compres online

És el grup d'usuaris format per les persones que ja han fet ús o faran ús de les aplicacions de l'empresa sigui mitjançant les versions webs o les versions per a mòbils i tauletes. Actualment les aplicacions estan disponibles a la versió web i amb el nou framework es podrà realitzar la versió mòbil i tauleta.

1.3.2 Treballadors de les botigues físiques dels clients

És el grup de treballadors format per les persones que ja han fet ús o faran ús de les aplicacions de l'empresa a les versions d'escriptori instal·lades a les botigues físiques de les companyies (on treballen els usuaris coworkers).

1.3.3 Treballadors interns a l'empresa Interiorvista

És el grup format per les persones que s'encarreguen de crear les aplicacions. Amb l'ús del nou framework hauran de recrear les aplicacions existents i crearan les noves. Formen part de diferents projectes i cadascun té uns requeriments diferents que cal tenir en compte per a la creació del nou motor.

1.4. Estat de l'art

Aquest apartat està dividit en tres subapartats. En el primer s'explica la situació actual dels motors de físiques, en el segon s'analitzen els diferents motors i finalment es fa una reflexió sobre quin és el pas a seguir.

1.4.1. Situació actual

L'evolució de les noves tecnologies ha imposat a les empreses a adaptar-se als nous temps per poder seguir sent competitives respecte a la competència.

Dins del món de l'interiorisme actual està molt estès la utilització de programes per crear les eines per configurar i reproduir virtualment entorns reals (els planificadors). Actualment existeixen múltiples motors de físiques. Per a realitzar l'estudi de l'estat de l'art s'ha escollit un subconjunt per poder exemplificar la situació actual del mercat tenint en compte el requeriment de l'empresa de poder detectar les col·lisions en temps real.

En el seu moment *Interiorvista* va escollir Unity per sobre d'altres sistemes com Unreal, el simulador de físiques BulletPhisics, ODE, PhysX o Havok Physics[4]. Es va considerar que era el que donava millors resultats per a les funcionalitats que requeria l'empresa i de cara als resultats visuals que oferia.

Unity[5] en si mateix no és un motor de col·lisions, sinó que es tracta d'un motor de videojocs multiplataforma creat per Unity Technologies l'any 2004 per David Helgason (CEO), Nicholas Francis (CCO), i Joachim Ante (CTO). Unity està disponible com a plataforma de desenvolupament per Microsoft Windows, OS X i Linux. La plataforma de desenvolupament dóna suport de compilació amb un gran nombre de plataformes (web, PC, dispositius mòbils, Smart TV, dispositius de realitat virtual, consoles). A

partir de la versió 5.4.0 no dóna suport al desenvolupament de contingut per a navegadors mitjançant el plugin web i en el seu lloc utilitza WebGL.
Un dels punts forts de Unity és que disposa d'un gran ventall de manuals d'ús, fòrums i pàgines on es pot consultar informació de tot tipus sobre el seu funcionament i programació.

Característiques:
Unity pot ser utilitzat juntament amb un gran nombre de programes (3ds Max, Maya, Blender, Cinema 4D, Adobe Photoshop...). El motor gràfic utilitza Direct3D (Windows), OpenGL (Mac i Linux) i OpenGL ES (Android i iOS). Permet treballar amb shaders gràcies al llenguatge ShaderLab (propi de Unity). També dona suport integrat per Nvidia i el motor de físiques PhysX (del que es parlarà més endavant). Disposa d'un control de versions propi per a assets i scripts.

El problema principal que existeix amb Unity recau en el fet que l'exportació a WebGL no és adequada per a les necessitats actuals de l'empresa. De cara a poder disposar de l'aplicació en format mòbil i tauleta també dóna problemes a causa del pes de l'exportació.

També considerar que Unity té altres mancances. Per exemple, si hi ha més de dos objectes que col·lisionen en un moment, el tractament de la col·lisió és poc acurat i provoca que els resultats de la col·lisió siguin erronis. Per exemple es pot acabar accedint a llocs ocupats per altres objectes o el càlcul de la resposta d'una col·lisió doni resultats inesperats.

1.4.2. Anàlisi de la competència

En aquest segon apartat es realitza una comparativa entre un subconjunt dels motors de físiques que existeixen actualment al mercat.

1.4.2.1 BulletPhysics

BulletPhysics[6] (Bullet) és un motor de físiques que simula la detecció de col·lisions, la dinàmica del cos tou i rígid. El seu principal autor, Erwin Coumans, va treballar per a

Sony Computer Entertainment R&D entre el 2003 i el 2010, per a AMD fins al 2014, i actualment treballa per a Google.

La biblioteca Bullet és lliure i de codi obert subjecte a les llicències zlib. La pàgina web de Bullet conté un fòrum sobre física per a discutir sobre l'entorn de simulació física per a jocs i animacions.

Característiques:

Bullet permet tant la simulació de cossos rígids com de cossos tous (fluids o gasos), amb detecció de col·lisions discreta i continua. Entre les formes de col·lisió que inclou es destaca l'ús de l'algoritme GJK per a polígons convexos.

Bullet pot treballar conjuntament amb altres programes com Unity, Maya, Softmage, Cinema 4D, Blender... Disposa d'optimitzacions opcionals per a PlayStation 3 Cell SPU, CUDA i OpenGL.

1.4.2.2 Open Dynamics Engine

Open Dynamics Engine[7] (ODE) és una llibreria de software lliure, de qualitat professional, dirigida a la simulació de cossos rígids articulats. El seu autor Russel Smith, amb l'ajuda de diferents col·laboradors, van iniciar el projecte l'any 2001.

En tractar-se d'una llibreria de codi obert, facilita la seva implementació interna, i a la vegada disposa d'una completa guia d'usuari que a més d'incloure instruccions sobre el seu funcionament i instal·lació, també conté nocions teòriques i pràctiques sobre els conceptes que utilitza el seu motor.

Característiques:

ODE està enfocat a la simulació d'estructures articulades tant en temps real com interactiva, les quals consisteixen en cossos rígids de diferents formes connectats entre si mitjançant connexions de diferents tipologies. Aquest motor prioritza la velocitat en contra de la precisió física, la qual cosa el fa menys fiable respecte a altres motors del mercat. L'avantatge principal que té, és la gran capacitat de personalització de l'entorn (col·lisions).

1.4.2.3 PhysX

La Unitat de Processament de Física (*Physics Processing Unit*, PPU) **PhysX**[8] és un xip i una eina de desenvolupament dissenyat per realitzar càlculs físics complexos. Conegut anteriorment com a SDK de NovodeX, va ser originalment dissenyada per AGEIA però quan Nvidia va adquirir AGEIA el va integrar en els seus propis xips. L'any 2005 Sony va firmar un acord amb AGEIA per fer ús del SDK NovodeX a la PlayStation3, i el seu ús s'ha popularitzat des de llavors.

PhysX en tractar-se d'un motor privat no facilita la seva implementació interna, tot i que proporciona gratuïtament eines que permeten construir exemples per a veure les seves funcionalitats.

Característiques:
PhysX inclou un sistema de físiques per a cossos rígids i tous i permet simular objectes amb un alt grau de realisme. Disposa d'un gran ventall de mecanismes per aplicar conceptes físics (fricció, acceleració, moviment lineal i rotació...). Finalment destacar que PhysX està optimitzada per treballar amb multithreading.

1.4.2.4 Havok Game Dynamics SDK

Havok Game Dynamics SDK[9] (Havok) és un motor físic de simulació dinàmica utilitzat en videojocs per recrear les interaccions entre objectes i personatges. Permet detectar col·lisions, gravetat, massa i velocitat en temps real arribant a recrear ambientes realistes i naturals.

El motor Havok va ser comprat per Intel per poder fer la competència al seu principal rival, PhysX. Havok en ser un motor privat i no facilita la seva implementació interna, tot i que proporciona gratuïtament eines que permeten construir exemples per entendre les seves funcionalitats. Disposa d'una extensa documentació d'ajuda i grans eines de creació de continguts, depuració i personalització, característiques que li han permès convertir-se en referència dins del món dels videojocs.

Característiques:
Destacar que Havok a les seves últimes versions utilitza únicament la GPU per a la realització dels càlculs, deixant lliure a la CPU. Havok fa ús de la llibreria de Direct3D i OpenGL.

Es tracta d'un motor multithreading i multi plataforma. Ofereix un gran rendiment respecte al temps de detecció de col·lisions i simulacions reals de solucions físiques. També dona una gran rapidesa i robustesa, l'ús de tècniques innovadores pròpies de l'empresa permet que disposin d'unes físiques molt precises.

1.4.3. Conclusions

En general, els motors de físiques que s'han consultat es tracten en la majoria d'eines privades a les quals no se'ls pot realitzar modificacions i en el cas de BulletPhysics, Open Dynamics Engine o PhysX, que es tracten de programes de codi lliure, són massa complexos per a les necessitats actuals de l'empresa.

Per aquest motiu s'ha considerat millor no utilitzar un producte extern perquè cap dels estudiats s'ajusta a les necessitats de l'empresa i també perquè d'aquesta manera es podrà tenir un millor control un cop desenvolupat.

S'ha de reconèixer però, que hi ha productes atractius com BulletPhysics o Unreal, dels que es poden extreure idees. En realitzar l'estudi s'ha vist que Unreal fa ús de la tècnica Signed Distance Field per treballar l'apartat de l'Ambient Occlusion[10] i que BulletPhysics utilitza l'algoritme de distància de Gilbert-Johnson-Keerthi dins del seu motor de col·lisions.

En realitzar una cerca sobre tècniques de col·lisions s'ha trobat que existeixen un gran nombre de tècniques. Donat que el projecte està acotat temporalment, s'ha decidit implementar-ne només tres. Les dues que s'han extret dels motors actuals, **Signed Distance Field** i l'**algoritme de distància de Gilbert-Johnson-Keerthi**, i la tècnica **Separating Axis Theorem.**[11].

2. Abast del projecte

En aquest apartat es tracta amb més profunditat la problemàtica que es vol resoldre i es defineixen els passos a seguir per assolir tots els objectius.

2.1 Formulació del problema

El problema que vol resoldre aquest projecte és eliminar els punts negatius que s'han trobat amb l'ús de la plataforma Unity. Després d'haver explicat el context de l'empresa i l'àmbit del projecte, es pot afirmar que Unity:

- En el moment d'iniciar el projecte oferia una exportació a WebGL que estava en fase beta. Inicialment Unity no estava enfocat per a treballar per a web.
- Baix rendiment per a Internet Explorer.
- Cap control de la memòria en web.
- No permet actuar a temps per solucionar la pèrdua de context.
- S'està obligat a treballar amb una gestió de recursos definida.
- El temps de compilació és molt llarg.
- El temps de càrrega és lent.
- El sistema de detecció de col·lisions que disposa no s'adapta a les necessitats de l'empresa.
- Es tracta d'una eina multiús, capaç de moltes funcionalitats. Però per les necessitats de l'empresa no dóna els resultats més adients.

A partir dels punts anteriors es pot afirmar que la quantitat i complexitat dels canvis impossibiliten seguir utilitzant la plataforma Unity.

2.2. Objectius del projecte

Per solucionar la problemàtica esmentada anteriorment, aquest projecte té com a objectiu crear un motor de col·lisions multiplataforma que permeti adaptar les aplicacions actuals de l'empresa a un entorn que exclogui el Unity. Per fer-ho haurà d'assolir els següents objectius:

- Prescindir de Unity.
- Poder treballar amb qualsevol de les següents plataformes: Android, iOS i web.

- Poder treballar amb els navegadors Chrome, Firefox i Internet Explorer.
- Disposar d'un control total pel que fa a memòria, recursos i entrada de dades.
- Poder treballar amb escenes 2D i 3D.
- Permetre qualsevol tipus d'alineació dels objectes (alineats als eixos o orientats).
- Poder treballar amb objectes convexos.
- Permetre orientació, translació i escalat dels objectes.
- Poder detectar totes les col·lisions que es puguin donar entre tots els objectes en el menor temps possible.
- Poder detectar totes les col·lisions que es puguin donar entre dos conjunts d'objectes en el menor temps possible.
- Escollir quina combinació de tècniques i quin disseny és millor.

2.3 Abast

Com s'ha descrit a la introducció, el projecte s'ha centrat en la creació del motor de col·lisions del framework d'Interiorvista.

Aquest projecte va començar amb la definició de quins objectius, funcionalitats i requisits tècnics havia de complir el producte resultant.

Es va fer un estudi de les tres tècniques esmentades a l'apartat "*1.4-Estat de l'Art*", **Separating Axis Theorem**, **Signed Distance Field** i l'**algoritme de distància de Gilbert-Johnson-Keerthi**.

La primera fase de la implementació seguia un **disseny orientat a objectes** (DOO). Es va definir que inicialment es faria ús d'ambients 2D i posteriorment es passaria a una implementació en 3D. Per a realitzar el càlcul de col·lisions es va definir que es treballaria amb un **col·lisionador de cubs** (les **mínimum bounding box (MBB)**[1] dels objectes tindrien forma de cub). També es va acordar que es treballaria amb **axis aligned bounding boxes (AABB)**[1] i en una segona fase es realitzarien els càlculs utilitzant **oriented bounding boxes (OBB)**[1] per tal de complir tots els requisits dels diferents stakeholders. Això repercutia en què els càlculs necessaris per detectar les col·lisions serien més complexos.

[1] A l'apartat *"7.1-Els motors de col·lisions"* hi ha una explicació més detallada.

També es va acordar que s'estudiaria com afecta la implementació d'una estructura de nivell com la que s'ha explicat prèviament. Mitjançant un sistema de **col·lisions d'esferes** (les **mínimum bounding box**[1] dels objectes tinguessin forma d'esfera) s'esperava reduir els costos dels càlculs, ja que es tracta de càlculs més eficients donat que no hi ha costats de polígons a calcular. A més consisteix en una solució independent a l'orientació dels eixos de coordenades.
Finalment s'esperava que en aplicar inicialment aquesta tècnica es reduiria el nombre d'objectes a comprovar amb les altres tècniques.

Un cop fetes les implementacions en DOO per poder seleccionar les dues opcions més eficients i fiables, es realitzaria una comparativa amb una bateria de tests unitaris entre les diferents combinacions de tècniques. Amb les guanyadores s'implementaria una segona fase seguint un **disseny orientat a dades** (DOD).

Amb les dues implementacions seria necessari fer un testeig per comprovar quina implementació és més eficient.

Un cop escollida la solució més adequada (la combinació de tècniques i el disseny) el projecte finalitzaria amb la implantació del motor dins del framework i es formaria als usuaris que en un futur faran ús del motor.

2.4 Possibles obstacles

Detectar els possibles obstacles d'aquest projecte abans de començar farà augmentar les possibilitats del seu èxit. Aquests són els obstacles que s'han detectat:

- Falta de disponibilitat d'algun component de l'equip implicat en el projecte, ja sigui per necessitat puntual d'un projecte amb més prioritat, per malaltia o per compromís personal.
- Falta d'experiència en alguna de les eines utilitzades per part de l'equip del projecte.
- Per poder complir amb les últimes fases del projecte (implantació i formació) serà necessari que el framework estigui en un estat funcional.

Tenint en compte aquests possibles obstacles, s'ha realitzat la planificació inicial assignant temps addicional per la formació de l'equip.

3. Metodologia i rigor

El projecte s'ha dut a terme l'ús d'una metodologia àgil semblant a les estudiades a la universitat que han constat de reunions de seguiment setmanals. S'ha optat per aquesta metodologia, ja que és la que s'utilitza a l'empresa per gestionar la resta de projectes interns, i dóna bons resultats, permet portar un seguiment continu de tot el procés, permet actuar en consideració en cas que es produeixin desviacions i permet validar les diferents tasques en el moment en què es van finalitzant.

3.1. Mètodes de treball

Per realitzar aquest projecte s'ha seguit la metodologia de treball Scrum, que es basa a treballar mitjançant iteracions, també anomenades sprints. Cada sprint tenia unes tasques definides a completar durant un temps assignat. Aquestes iteracions han permès obtenir feedback regular de l'estat del projecte i definir millor quines funcionalitats eren prioritàries. A l'apartat "*4.2-Recursos*" es descriuen els actors que formen part i els seus rols.

Durant l'etapa de definició de requisits s'han realitzat reunions per definir les funcionalitats bàsiques del motor actual que es volien mantenir en el nou. També es van debatre les característiques i es van proposar canvis i/o noves funcionalitats. Addicionalment, tal com s'explica al context, es van analitzar quines mesures calia prendre perquè el motor pogués servir per recrear les aplicacions actuals.

Les iteracions han tingut lloc durant la fase anomenada Sprints dins de la planificació. A cada iteració s'han definit una sèrie de tasques amb l'objectiu d'implementar una funcionalitat concreta a partir de l'especificació i la implementació de les fases anteriors. Al final de cada iteració s'ha comprovat el seu correcte funcionament i s'ha donat per tancada la iteració.

3.2. Eines de seguiment

Per monitoritzar l'evolució dels sprints del projecte s'han fet ús de les tècniques proposades per Scrum, principalment les reunions regulars. S'han dut a terme reunions setmanals que permeten gestionar els cicles curts del projecte, és a dir, avaluar els resultats obtinguts durant el període anterior i definir que s'espera del

següent. D'aquesta manera s'han corregit i resolt els dubtes que van anar apareixent durant el desenvolupament.

La gestió del projecte s'ha realitzat amb la plataforma Trello. També s'ha fet ús d'un repositori a Bitbucket per emmagatzemar el codi font del projecte i per realitzar la documentació s'ha fet ús de Google Drive.

3.3. Mètode de validació

Tal com marca Scrum, els sprints han estat formats per tasques que han estat validades al final de cada cicle durant les reunions setmanals. La validació està estretament relacionada amb la definició de la tasca, però per assegurar que la seva realització no provocava erros en els resultats de la resta d'implementacions es va crear una bateria de tests unitaris que s'executava en finalitzar cada iteració.

Si algun dels tests fallava, es buscava el motiu de l'error, ja que podria ser un error en el codi o una mancança de la tècnica que s'estigués implementant. Si era un error en el codi, es resolia i es tornava a executar la bateria de test fins a assegurar que tots els tests s'assolien correctament.

Aquesta bateria de test es va anar complementant a mesura que el projecte evolucionava, afegint nous tests segons les noves implementacions.

Una vegada es van implementar les diferents tècniques en el disseny orientat a objectes es van escollir les dues millors opcions i es va prosseguir amb les versions orientades a dades per poder realitzar una.

Amb la solució final, tècnica i disseny es va procedir amb la seva implantació dins del framework. Així doncs s'han realitzat provés més realistes que han permès verificar el correcte funcionament de la solució escollida.

A partir d'aquest moment, i havent completat totes les tasques dels diferents cicles del projecte, aquest s'ha validat i analitzat amb èxit.

4. Planificació inicial

L'ús de metodologies àgils implica que els requeriments puguin canviar durant el desenvolupament del projecte. Aquestes metodologies són flexibles i adaptables a aquests canvis, ja que poden provocar desviacions. Permeten gestionar millor aquests canvis sense retallades a l'abast o endarreriments insalvables. Es duran a terme reunions setmanals per part de l'equip involucrat en l'etapa en la qual es trobi el projecte per analitzar-ne el progrés.

Tenint en compte els requisits i l'abast descrits en els apartats anteriors es va preveure que el projecte tingués una durada estimada de 10 mesos, començant el març i acabant a finals de desembre del 2016.

La dedicació a aquest projecte ha estat de 20 hores setmanals, 4 hores per cada dia laborable. Durant la planificació s'han tingut en compte les vacances.

Aquest projecte s'ha dividit en 6 etapes: definició de requisits, gestió del projecte, disseny, sprints iteratius, implantació i feina futura.

La fase d'sprints és la part més complexa de la planificació i s'ha dividit en 10 fites, algunes amb més importància que altres, diferenciades al diagrama de Gantt de la *Figura 4* amb colors (verd – risc lleu, taronja – risc mitjà, vermell – risc elevat). Cadascuna de les 10 fites està dividida en subtasques, on cadascuna s'ha considerat un sprint del projecte.

Cada sprint s'ha compost per cinc fases: reunió de seguiment, anàlisis, disseny, implementació i testeig. A la imatge següent es pot veure el workflow de les cinc fases i els recursos assignats a cadascuna:

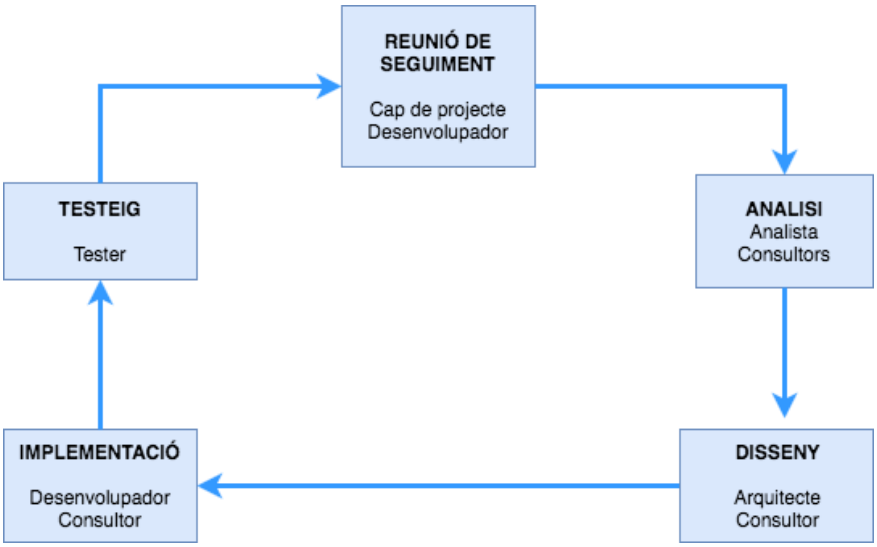


Figura 4: Workflow de les cinc fases d'una subtasca (sprint)

- **Reunió de seguiment:** es defineix quin objectiu es vol assolir en aquesta tasca.
- **Anàlisi:** es realitza un procés de formació i anàlisi per a poder dur a terme la tasca.
- **Disseny:** tenint en compte les consideracions anteriors es modifica el disseny actual per poder adaptar-lo a la nova tasca de manera que se segueixin complint els requisits de fer un disseny escalable, modulable i orientat a objectes u orientat dades segons la fase en la qual s'estigui del procés.
- **Implementació:** consisteix en la implementació de la tasca tenint en compte l'anàlisi i el disseny realitzats.
- **Testeig:** s'analitza el correcte funcionament de l'sprint. En cas que es trobi algun error el tester ho notifica al desenvolupador perquè ho arregli. Un cop resolt es repeteix el testeig fins a assolir el resultat esperat.

4.1 Diagrama de Gantt (simplificat)

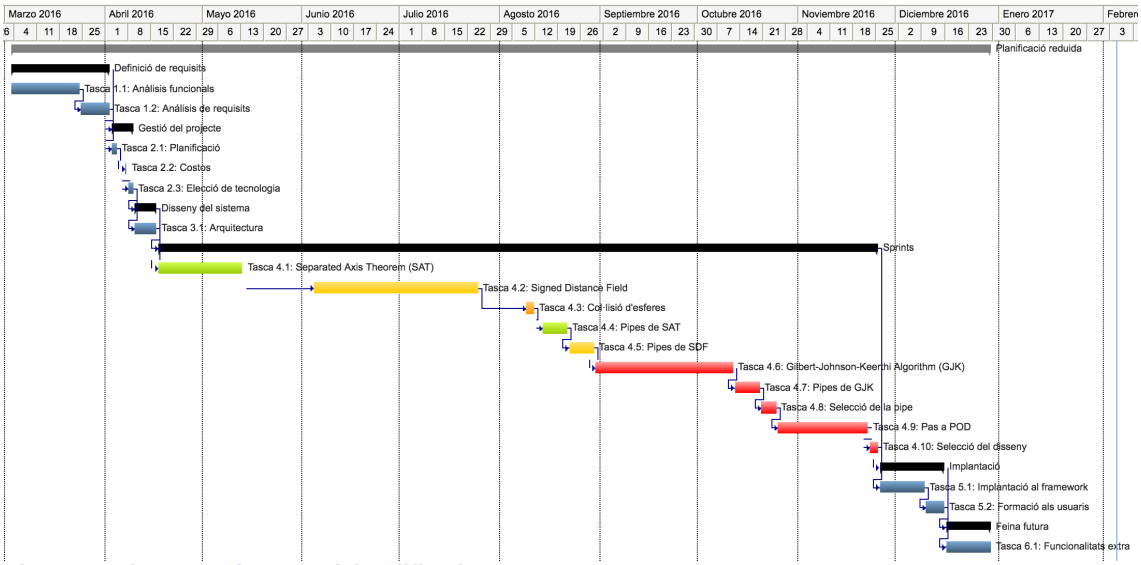


Figura 5: Diagrama de Gantt (simplificat)

A *l'Annex 1* està inclosa una versió del diagrama de Gantt (simplificat) més ampliada.

El diagrama de Gantt de la planificació inicial s'ha dut a terme a l'assignatura de GEP (un cop el projecte ja estava a mig procés) així doncs ja conté les desviacions que s'havien produït fins a aquell moment a causa de l'entrada d'algun projecte més prioritari per part de l'empresa.

A l'apartat "*4.3-Descripció de les tasques"* s'han descrit detalladament cadascuna de les etapes explicant quines tasques les formen, la seva durada i els recursos assignats per realitzar-les.

4.2 Recursos

En aquest projecte es poden diferenciar dos tipus de recursos, els humans i els materials.

4.2.1 Recursos humans

A continuació es detallen el nom, la responsabilitat i la funció dels recursos humans del projecte.

4.2.1.1 Cap de projecte

Forma part de l'equip de gestió i de l'equip de suport. Com a equip de gestió és responsable de realitzar les planificacions i pressupostos, supervisar i dirigir les diferents tasques. Com a equip de suport el cap de projecte ha de donar consells a la resta de recursos de l'equip executor perquè puguin assolir les seves tasques.

4.2.1.2 Analista

Forma part de l'equip executor i de l'equip de suport. Com a equip de gestió és responsable d'analitzar la problemàtica que es vol resoldre i definir els requisits funcionals i no funcionals. També realitza l'estudi de les diferents tècniques i defineix el seu funcionament. Com a equip de suport l'analista ha de donar consells a la resta de recursos de l'equip executor perquè puguin assolir les seves tasques.

4.2.1.3 Arquitecte

Forma part de l'equip executor i de l'equip de suport. Com a equip de gestió és responsable de realitzar l'estudi entre els dos tipus de dissenys, i dissenyar les millors arquitectures per assolir els requisits del projecte i de les diferents tècniques. S'encarrega de realitzar dos dissenys orientats a objectes (el genèric i l'específic per l'algoritme GJK) i el disseny orientat a dades per l'algoritme GJK. Com a equip de suport l'arquitecte ha de donar consells a la resta de recursos de l'equip executor perquè puguin assolir les seves tasques.

4.2.1.4 Desenvolupador

Forma part de l'equip executor. És responsable d'implementar les diferents tècniques tenint en compte l'anàlisi i el disseny. Un cop escollida la millor tècnica i el millor disseny realitza la implantació del motor dins del framework.

4.2.1.5 Tester

Forma part de l'equip executor. És responsable de testejar les diferents tècniques implementades i assegurar el seu correcte funcionament. Implementa una bateria de tests unitaris per comprovar-ho.

4.2.1.6 Consultors

Formen part de l'equip executor però la seva principal tasca és donar suport a la resta de recursos per resoldre els diferents dubtes que puguin aparèixer durant el projecte. Està format per un grup reduït de treballadors d'Interiorvista amb coneixements multidisciplinaris que han aconsellat i/o ajudat a resoldre dubtes per tal de poder assolir el projecte de la millor manera. Dins d'aquest equip s'inclouen la resta de recursos en el seu paper d'equip de suport.

4.2.1.7 Usuaris

És el grup format pels diferents treballadors d'Interiorvista que han de fer ús del motor. Han rebut una formació per entendre com funciona el motor per si en un futur sigui necessari realitzar canvis.

Els diferents rols del projecte relacionats amb els recursos humans anteriors es troben representats en el diagrama següent:

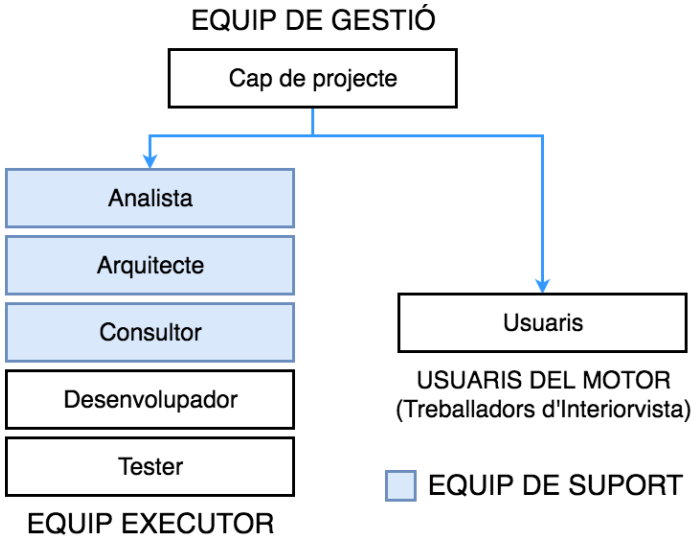


Figura 6: Diagrama de recursos humans del projecte

4.2.1.8 Els meus rols dins del projecte

El projecte s'ha dut a terme seguint les diferents fases de la gestió d'un projecte de software. Per aquest motiu s'han definit els diferents rols explicats anteriorment que són els responsables de les diferents fases.

Ara bé, durant tot el procés he dut a terme les tasques dels diferents recursos explicats, exceptuant els casos del consultor i dels usuaris. Amb l'ajuda de diferents companys de l'empresa com a consultors he dut a terme les diferents etapes explicades en aquesta memòria.

He actuat com a cap de projecte per realitzar les planificacions i els pressupostos.
He actuat com a analista per definir la problemàtica, l'estat de l'art, els objectius, els requisits i també he realitzat l'estudi i la formació en les diferents matèries necessàries per al projecte (exposades als apartats de teoria).
He actuat com a arquitecte per a definir els tres dissenys necessaris.
He actuat com a programador per implementar les diferents tècniques i la construcció de les 16 pipes resultants. També he realitzat les dues comparatives que han servit per escollir quines pipes eren més adequades i el tipus de disseny. Finalment amb l'ajuda dels companys que han implementat el framework he dut a terme la implantació del motor.
He actuat com a tester en la definició de la bateria de tests unitaris i en la correcció del codi en cas que en algun test s'obtingués un resultat erroni.

4.2.2 Recursos materials

A continuació es detallen el nom i la funció dels recursos de maquinari i programari:

- Ordinador HP i7-4510 amb 12 GB RAM.
- Pantalla Samsung 24".
- Visual Studio: Entorn de desenvolupament per C++.
- Trello: Plataforma de gestió de projectes.
- Google Drive: Plataforma d'emmagatzemament d'arxius.
- Bitbucket: Control de versions del codi.
- Gantter: Editor de diagrames de Gantt.
- Skype: Eina de comunicació per realitzar reunions a distància.
- Mail de Google: Gestionar reunions, enviar informes i aclarir dubtes.
- Astah Professional: Disseny de diagrames.
- Google Drawings: Disseny de diagrames.
- draw.io Diagrams: Disseny de diagrames.

4.3. Descripció de les tasques

Aquest projecte consta de 6 etapes: definició de requisits, gestió del projecte, disseny del sistema, sprints iteratius, implantació i feina futura.

A la següent taula es descriuen els temps totals dedicats a cadascuna de les fases del projecte:

ETAPA	DIES	HORES	% TEMPS
Definició de requisits	22 dies	88 hores	**10.37%**
Gestió del projecte	5 dies	20 hores	**2.36%**
Disseny	5 dies	20 hores	**2.36%**
Sprints	156 dies	624 hores	**73.58%**
Implantació	14 dies	56 hores	**6.6%**
Feina Futura	10 dies	40 hores	**4.7%**
TOTAL	**212 dies**	**848 hores**	**100%**

Taula 1: Pes dels temps segons les fases del projecte

Degut que el gran pes del projecte recau en l'etapa d'sprints iteratius, que inclou les cinc fases que asseguren un bon desenvolupament, s'ha cregut convenient fer una distinció dels pesos segons els diferents recursos del projecte:

RECURS	DIES	HORES	% TEMPS
Cap de projecte	23.375 dies	93.5 hores	**12.5%**
Analista	41 dies	164 hores	**21.92%**
Arquitecte	21.4 dies	85.6 hores	**11.44%**
Desenvolupador	39.575 dies	158.3 hores	**21.16%**
Tester	44.5 dies	178 hores	**23.88%**
Consultor	17.15 dies	68.6 hores	**9.1%**
TOTAL	**187 dies**	**748 hores**	**100%**

Taula 2: Pes dels temps segons els recursos del projecte

A *l'Annex 2* s'adjunta la taula amb els temps total de dedicació de cada recurs en les diferents etapes del projecte. Amb aquests temps s'ha realitzat el càlcul dels diferents costos de les *Taules 1 i 2*.

Per assegurar que la planificació compleix segons la teoria dels projectes de software, a continuació s'ha justificat que el temps dedicat a cadascuna de les tres etapes bàsiques és l'esperat perquè el projecte funcioni adequadament:

ETAPA	% TEMPS ESPERAT	% TEMPS DEDICAT
ANALISIS	40%-50%	**47.4%**
IMPLEMENTACIÓ	15%-20%	**23.3%**
TESTEIG I IMPLANTACIÓ	30%-40%	**29.3%**

Taula 3: Pes del temps dedicat segons les fases d'un projecte de software

Donat que el paper dels consultors és transversal al llarg de tot el procés s'ha considerat oportú prescindir de les seves hores per a realitzar les justificacions. Per aquest motiu es parteix amb un còmput total de 679 hores dedicades al projecte (748h reals del projecte – 68.8h consultors).

Hores d'anàlisis: s'inclouen les hores del cap de projecte (menys les dedicades a l'etapa de futur), les de l'analista i les de l'arquitecte. Per aquest motiu les hores dedicades a l'etapa d'anàlisis són de 323.1 hores.
Justificació: 93.5 hores cap de projecte – 20 hores cap de projecte a l'etapa d'implantació + 164 hores de l'analista + 85.6 hores de l'arquitecte = 323.1 hores.

Hores d'implementació: s'inclouen les hores del desenvolupador (menys les dedicades a l'etapa d'implantació). Per aquest motiu es considera que les hores dedicades a l'etapa d'anàlisis són de 110.3 hores.
Justificació: 158.3 hores desenvolupador – 48 hores desenvolupador a l'etapa d'implantació = 110.3 hores.

Hores de testeig i implantació: s'inclouen les hores del tester i les hores dedicades a l'etapa d'implantació. Per aquest motiu es considera que les hores dedicades a l'etapa de testeig i implantació són de 246 hores.
Justificació: 178 hores del tester + 20 hores del cap de projecte a l'etapa d'implantació + 48 hores del desenvolupador a l'etapa d'implantació = 246 hores.

A continuació es descriuen les tasques que formen cada etapa, amb el temps i els recursos assignats.

4.3.1 Etapa 1: Definició de requisits

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca de *Definició de requisits*.

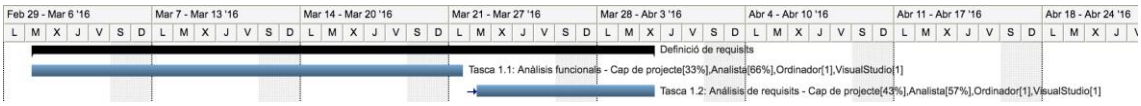


Figura 7: Diagrama de Gantt de l'etapa de definició de requisits

		Nombre	Duración	Inicio	Fin	Recursos
1		**Definició de requisits**	22d	**01/03/2016**	**30/03/2016**	
2		Tasca 1.1: Anàlisis funcionals	15d	01/03/2016	21/03/2016	Cap de projecte[33%],Analista[66%],Ordinador[1],VisualStudio[1]
3		Tasca 1.2: Anàlisis de requisits	7d	22/03/2016	30/03/2016	Cap de projecte[43%],Analista[57%],Ordinador[1],VisualStudio[1]

Figura 8: Llistat i definició de les tasques del Gantt de l'etapa de definició de requisits

Tasca 1.1 Anàlisis funcional
Descripció: Definir les funcionalitats desitjades per al motor, descriure els casos d'ús i els diagrames de seqüència.
Recursos: Cap de projecte, analista, ordinador, llicència de Visual Studio 2015.
Durada: 15 dies

Taula 4: Definició tasca 1.1 - Anàlisis funcional

Tasca 1.2 Anàlisis de requisits
Descripció: Definir els requisits tècnics necessaris per al correcte funcionament del motor.
Recursos: Cap de projecte, analista, ordinador, llicència de Visual Studio 2015.
Durada: 7 dies

Taula 5: Definició de la tasca 1.2 – Anàlisis de requisits

4.3.2 Etapa 2: Gestió del projecte

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca de *Gestió del projecte*.

	Nombre	Duración	Inicio	Fin	Recursos
1	**Gestió del projecte**	5d	**31/03/2016**	**06/04/2016**	
2	Tasca 2.1: Planificació	2d	31/03/2016	01/04/2016	Cap de projecte,Ordinador[1],Gantter[1]
3	Tasca 2.2: Pressupost	1d	04/04/2016	04/04/2016	Cap de projecte,Ordinador[1]
4	Tasca 2.3: Elecció de tecnologies	2d	05/04/2016	06/04/2016	Cap de projecte,Ordinador[1]

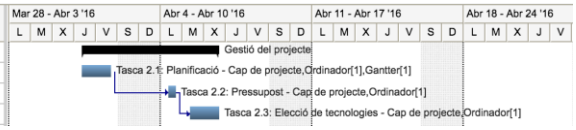


Figura 9: Diagrama de Gantt i llistat de les tasques de l'etapa de gestió del projecte

Tasca 2.1 Planificació
Descripció: Definir les tasques a realitzar durant les següents etapes del projecte per així complir amb els requisits i assolir el projecte amb èxit. També cal definir els recursos que es faran servir durant el projecte.
Recursos: Cap de projecte, ordinador, Gantter
Durada: 2 dies

Taula 6: Descripció de la tasca 2.1 – Planificació

Tasca 2.2 Pressupost
Descripció: A partir dels recursos disponibles i la seva planificació, definir el cost del projecte. Addicionalment, avaluar la sostenibilitat i la viabilitat econòmica del projecte.
Recursos: Cap de projecte, ordinador
Durada: 1 dies

Taula 7: Descripció de la tasca 2.2 – Pressupost

Tasca 2.3 Elecció de tecnologies
Descripció: Identificar els reptes tècnics que sorgiran durant el desenvolupament del projecte, per escollir les més adequades per al projecte.
Recursos: cap de projecte, ordinador
Durada: 2 dies

Taula 8: Descripció de la tasca 2.3 - Elecció de tecnologies

4.3.3 Etapa 3: Disseny del sistema

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca de *Disseny del sistema*.

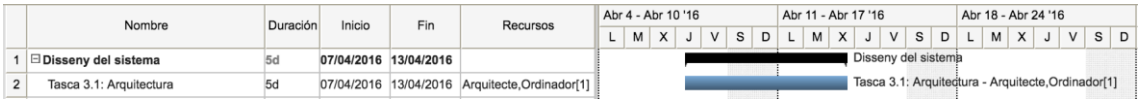


Figura 10: Diagrama de Gantt i llistat de les tasques de l'etapa de disseny del sistema

Tasca 3.1 Arquitectura
Descripció: Definir els dissenys del sistema que s'han d'utilitzar per poder assolir tots els requisits definits: disposar d'un sistema escalable i modulable. També cal realitzar una versió orientada a objectes i una orientada a dades.
Recursos: Arquitecte, ordinador
Durada: 5 dies

Taula 9: Descripció de la tasca 3.1 – Arquitectura

4.3.4 Etapa 4: Sprints iteratius

Implementar les diferents funcionalitats del motor mitjançant sprints o iteracions. Una iteració dura entre una setmana o dues i comprèn el desenvolupament de la funcionalitat i el seu testeig. És a dir, al final de cada iteració es fan proves unitàries de cada funcionalitat per assegurar el seu correcte funcionament. Aquestes proves són crítiques, ja que minimitzen els errors a la següent etapa d'implantació i fan que aquesta sigui molt més ràpida.

4.3.4.1 Tasca 4.1: Separated Axis Theorem

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 4.1 *Implementació del Separated Axis Theorem*.

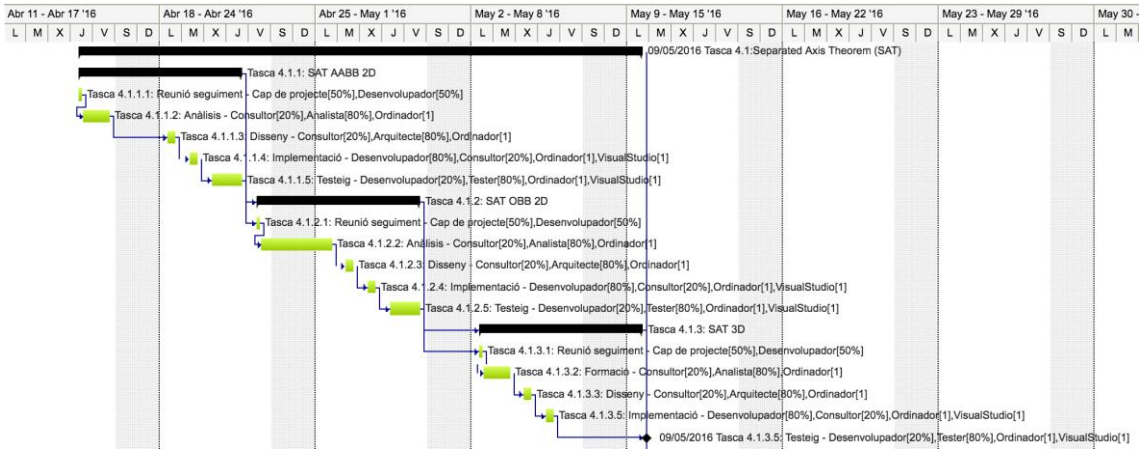


Figura 11: Diagrama de Gantt de l'etapa de Separated Axis Theorem (SAT)

Tasca 4.1:Separated Axis Theorem (SAT)	**18d**	**14/04/2016**	**09/05/2016**	
Tasca 4.1.1: SAT AABB 2D	**6d**	**14/04/2016**	**21/04/2016**	
Tasca 4.1.1.1: Reunió seguiment	0.5d	14/04/2016	14/04/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.1.1.2: Anàlisis	1.5d	14/04/2016	15/04/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.1.1.3: Disseny	1d	18/04/2016	18/04/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.1.1.4: Implementació	1d	19/04/2016	19/04/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.1.1.5: Testeig	2d	20/04/2016	21/04/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]
Tasca 4.1.2: SAT OBB 2D	**6d**	**22/04/2016**	**29/04/2016**	
Tasca 4.1.2.1: Reunió seguiment	0.5d	22/04/2016	22/04/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.1.2.2: Anàlisis	1.5d	22/04/2016	25/04/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.1.2.3: Disseny	1d	26/04/2016	26/04/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.1.2.4: Implementació	1d	27/04/2016	27/04/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.1.2.5: Testeig	2d	28/04/2016	29/04/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]
Tasca 4.1.3: SAT 3D	**6d**	**02/05/2016**	**09/05/2016**	
Tasca 4.1.3.1: Reunió seguiment	0.5d	02/05/2016	02/05/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.1.3.2: Anàlisis	1.5d	02/05/2016	03/05/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.1.3.3: Disseny	1d	04/05/2016	04/05/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.1.3.4: Implementació	1d	05/05/2016	05/05/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.1.3.5: Testeig	2d	06/05/2016	09/05/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]

Figura 12: Llistat i definició de les tasques del Gantt de l'etapa de Separated Axis Theorem (SAT)

A continuació es descriuen les subtasques que formen la tasca.

Tasca 4.1 Separated Axis Theorem
Descripció: Implementar la tècnica del Separated Axis Theorem amb un disseny orientat a objectes. S'ha dividit en tres subtasques definides a continuació.
Durada: 18 dies
SUBTASQUES:
Tasca 4.1.1 Separated Axis Theorem – AABB – 2D
Descripció: Implementar la tècnica del Separated Axis Theorem per a dos objectes

AABB en un sistema 2D i un disseny orientat a objectes.
Tasca 4.1.2 Separated Axis Theorem – OBB – 2D
Descripció: Implementar la tècnica de Separated Axis Theorem per a dos objectes OBB en un sistema 2D i un disseny orientat a objectes.
Tasca 4.1.3 Separated Axis Theorem – 3D
Descripció: Implementar la tècnica del Separated Axis Theorem per a dos objectes AABB, dos OBB i per el cas AABB i OBB en un sistema 3D i un disseny orientat a objectes.

Taula 10: Descripció de la tasca 4.1 – Separated Axis Theorem (SAT)

A la taula següent es pot observar la divisió iterativa del format d'sprints definit a l'inici del capítol, definint els recursos i les hores requerides per a les tasques 4.1.1, 4.1.2 i 4.1.3:

DESCRIPCIÓ	RECURSOS	DURADA
Reunió de seguiment	Cap de projecte, desenvolupador	0.5 dia
Anàlisi	Analista, consultor, ordinador	1.5 dies
Disseny	Arquitecte, consultor, ordinador	1 dia
Implementació	Desenvolupador, consultor, ordinador, visual Studio	1 dia
Testeig	Tester, desenvolupador, consultor, ordinador, visual Studio	2 dies
TOTAL		**6 dies**

Taula 11: Descripció de les subtasques d'una tasca (sprint)

4.3.4.2 Tasca 4.2: Signed Distance Field

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 4.2 *Implementació del Signed Distance Field*.

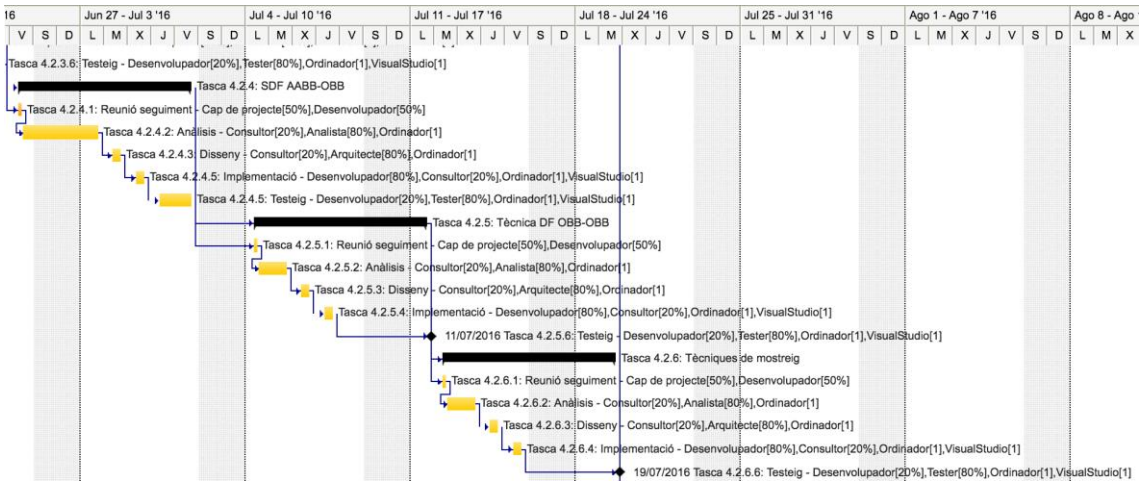


Figura 13: Diagrama de Gantt de l'etapa de Signed Distance Field (SDF)

Tasca 4.2: Signed Distance Field (SDF)	**36d**	**31/05/2016**	**19/07/2016**	
Tasca 4.2.1: SDF 1 punt - AABB	**6d**	**31/05/2016**	**07/06/2016**	
Tasca 4.2.1.1: Reunió seguiment	0.5d	31/05/2016	31/05/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.2.1.2: Anàlisis	1.5d	31/05/2016	01/06/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.2.1.3: Disseny	1d	02/06/2016	02/06/2016	Consultor[20%],Arquitecte[80%],Ordinador[1],VisualStudio[1]
Tasca 4.2.1.5: Implementació	1d	03/06/2016	03/06/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.2.1.5: Testeig	2d	06/06/2016	07/06/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]
Tasca 4.2.2: SDF AABB-AABB	**6d**	**08/06/2016**	**15/06/2016**	
Tasca 4.2.2.1: Reunió seguiment	0.5d	08/06/2016	08/06/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.2.2.2: Anàlisis	1.5d	08/06/2016	09/06/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.2.2.3: Disseny	1d	10/06/2016	10/06/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.2.2.5: Implementació	1d	13/06/2016	13/06/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.2.2.5: Testeig	2d	14/06/2016	15/06/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]
Tasca 4.2.3: SDF 1 punt - OBB	**6d**	**16/06/2016**	**23/06/2016**	
Tasca 4.2.3.1: Reunió seguiment	0.5d	16/06/2016	16/06/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.2.3.2: Anàlisis	1.5d	16/06/2016	17/06/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.2.3.4: Disseny	1d	20/06/2016	20/06/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.2.3.5: Implementació	1d	21/06/2016	21/06/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.2.3.6: Testeig	2d	22/06/2016	23/06/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]
Tasca 4.2.4: SDF AABB-OBB	**6d**	**24/06/2016**	**01/07/2016**	
Tasca 4.2.4.1: Reunió seguiment	0.5d	24/06/2016	24/06/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.2.4.2: Anàlisis	1.5d	24/06/2016	27/06/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.2.4.3: Disseny	1d	28/06/2016	28/06/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.2.4.5: Implementació	1d	29/06/2016	29/06/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.2.4.5: Testeig	2d	30/06/2016	01/07/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]
Tasca 4.2.5: Tècnica DF OBB-OBB	**6d**	**04/07/2016**	**11/07/2016**	
Tasca 4.2.5.1: Reunió seguiment	0.5d	04/07/2016	04/07/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.2.5.2: Anàlisis	1.5d	04/07/2016	05/07/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.2.5.3: Disseny	1d	06/07/2016	06/07/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.2.5.4: Implementació	1d	07/07/2016	07/07/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.2.5.6: Testeig	2d	08/07/2016	11/07/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]
Tasca 4.2.6: Tècniques de mostreig	**6d**	**12/07/2016**	**19/07/2016**	
Tasca 4.2.6.1: Reunió seguiment	0.5d	12/07/2016	12/07/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.2.6.2: Anàlisis	1.5d	12/07/2016	13/07/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.2.6.3: Disseny	1d	14/07/2016	14/07/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.2.6.4: Implementació	1d	15/07/2016	15/07/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.2.6.6: Testeig	2d	18/07/2016	19/07/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]

Figura 14: Llistat i definició de les tasques del Gantt de l'etapa de Signed Distance Field (SDF)

A continuació es descriuen les subtasques que formen la tasca.

Tasca 4.2 Signed Distance Field
Descripció: Implementar la tècnica del Signed Distance Field amb un disseny orientat a objectes. S'ha dividit en sis subtasques definides a continuació.
Durada total: 36 dies
SUBTASQUES:
Tasca 4.2.1 Signed Distance Field – 1 punt contra AABB
Descripció: Implementar la tècnica del Signed Distance Field entre un punt i un objecte en AABB en un sistema 3D i un disseny orientat a objectes.
Tasca 4.2.2 Signed Distance Field – AABB contra AABB
Descripció: Implementar la tècnica del Signed Distance Field entre dos objectes AABB en un sistema 3D i un disseny orientat a objectes.
Tasca 4.2.3 Signed Distance Field – 1 punt contra OBB
Descripció: Implementar la tècnica del Signed Distance Field entre un punt i un objecte en OBB en un sistema 3D i un disseny orientat a objectes.
Tasca 4.2.4 Signed Distance Field – AABB contra OBB
Descripció: Implementar la tècnica del Signed Distance Field entre un objecte AABB i un objecte OBB en un sistema 3D i un disseny orientat a objectes.
Tasca 4.2.5 Signed Distance Field – OBB contra OBB
Descripció: Implementar la tècnica del Signed Distance Field entre dos objectes OBB en un sistema 3D i un disseny orientat a objectes.
Tasca 4.2.6 Tècniques de mostreig
Descripció: Implementar funcions per poder mostrejar punts aleatòriament dins dels objectes 2D i 3D en un disseny orientat a objectes.

Taula 12: Descripció de la tasca 4.2 – Signed Distance Field (SDF)

A la taula següent es pot observar la divisió iterativa del format d'sprints definit a l'inici del capítol, definint els recursos i les hores requerides per a les tasques 4.2.1, 4.2.2, 4.2.3, 4.2.4, 4.2.5 i 4.2.6:

DESCRIPCIÓ	RECURSOS	DURADA
Reunió de seguiment	Cap de projecte, desenvolupador	0.5 dia
Anàlisi	Analista, consultor, ordinador	1.5 dies
Disseny	Arquitecte, consultor, ordinador	1 dia
Implementació	Desenvolupador, consultor, ordinador, visual Studio	1 dia
Testeig	Tester, desenvolupador, consultor, ordinador, visual Studio	2 dies
TOTAL		**6 dies**

Taula 13: Descripció de les subtasques de la tasca 4.2

4.3.4.3 Tasca 4.3: Col·lisió d'esferes

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 4.3 *Implementació de la col·lisió d'esferes.*

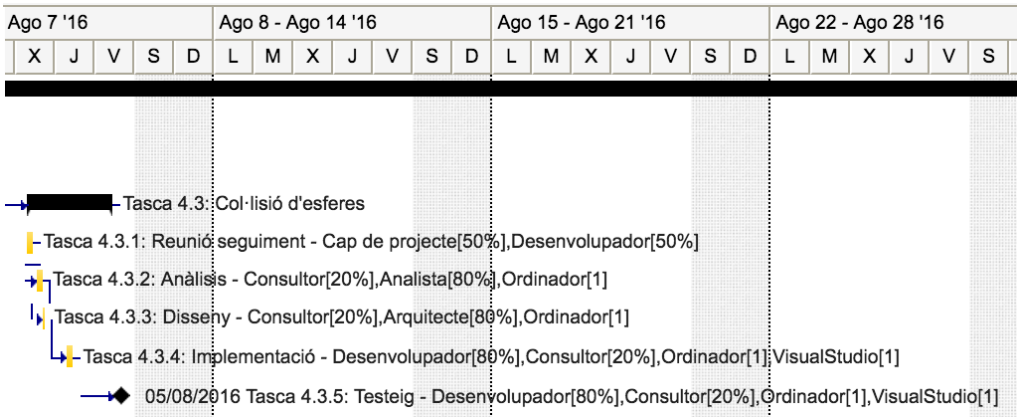


Figura 15: Diagrama de Gantt de l'etapa de col·lisió d'esferes

Tasca 4.3: Col·lisió d'esferes	3d	**03/08/2016**	**05/08/2016**	
Tasca 4.3.1: Reunió seguiment	0.5d	03/08/2016	03/08/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.3.2: Anàlisis	0.5d	03/08/2016	03/08/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.3.3: Disseny	0.5d	04/08/2016	04/08/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.3.4: Implementació	0.5d	04/08/2016	04/08/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.3.5: Testeig	1d	05/08/2016	05/08/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]

Figura 16: Llistat i definició de les tasques del Gantt de l'etapa de col·lisió d'esferes

A continuació es descriuen les subtasques que formen la tasca.

Tasca 4.3 Col·lisió d'esferes		
Descripció: Implementar la tècnica de col·lisió entre dos objectes esfèrics amb un disseny orientat a objectes. En cas que l'objecte no sigui esfèric, buscar la BoundingBox esfèrica de l'objecte i realitzar el càlcul.		
DESCRIPCIÓ	RECURSOS	DURADA
Reunió de seguiment	Cap de projecte, desenvolupador	0.25 dia
Anàlisi	Analista, consultor, ordinador	0.25 dia
Disseny	Arquitecte, consultor, ordinador	0.5 dia
Implementació	Desenvolupador, consultor, ordinador, visual Studio	0.5 dia
Testeig	Tester, desenvolupador, consultor, ordinador, visual Studio	1.5 dies
TOTAL		**3 dies**

Taula 14: Descripció de la tasca 4.3 – Col·lisió d'esferes

4.3.4.4 Tasca 4.4: Pipes del Separated Axis Theorem

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 4.4 *Implementació de les pipes del Separated Axis Theorem*.

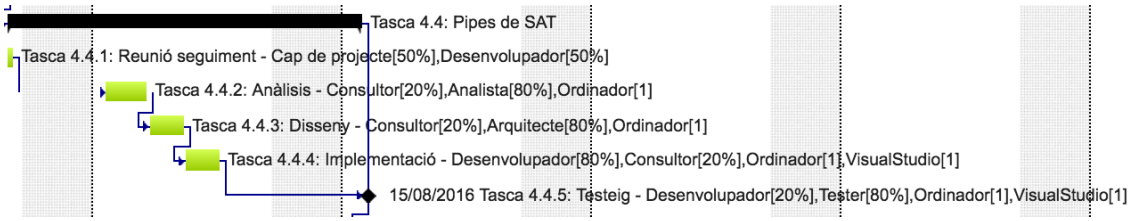


Figura 17: Diagrama de Gantt de l'etapa de pipes de Separated Axis Theorem

Tasca 4.4: Pipes de SAT	**6d**	**05/08/2016**	**15/08/2016**	
Tasca 4.4.1: Reunió seguiment	0.5d	05/08/2016	05/08/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.4.2: Anàlisis	1.5d	08/08/2016	09/08/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.4.3: Disseny	1d	09/08/2016	10/08/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.4.4: Implementació	1d	10/08/2016	11/08/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.4.5: Testeig	2d	11/08/2016	15/08/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]

Figura 18: Llistat i definició de les tasques del Gantt de l'etapa de pipes de Separated Axis Theorem

A continuació es descriuen les subtasques que formen la tasca.

Tasca 4.4 Pipes de Separated Axis Theorem		
Descripció: Implementar les pipes per a la tècnica del Separated Axis Theorem (SAT). S'han creat dues pipes, una que calcula l'execució de la tècnica de SAT i una altra que executa inicialment la tècnica de les col·lisions d'esferes i en cas de trobar col·lisió executa la tècnica de SAT.		
DESCRIPCIÓ	RECURSOS	DURADA
Reunió de seguiment	Cap de projecte, desenvolupador	0.5 dia
Anàlisi	Analista, consultor, ordinador	1.5 dies
Disseny	Arquitecte, consultor, ordinador	1 dia
Implementació	Desenvolupador, consultor, ordinador, visual Studio	1 dia
Testeig	Tester, desenvolupador, consultor, ordinador, visual Studio	2 dies
TOTAL		**6 dies**

Taula 15: Descripció de la tasca 4.4 – Pipes de Separated Axis Theorem

4.3.4.5 Tasca 4.5: Pipes de Signed Distance Field

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 4.5 *Implementació de les pipes del Signed Distance Field*.

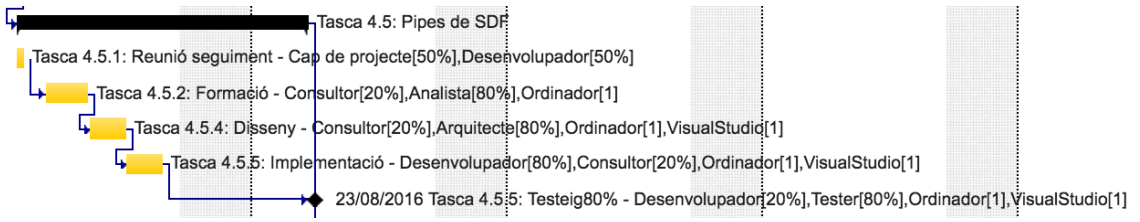


Figura 19: Diagrama de Gantt de l'etapa de pipes del Signed Distance Field

Tasca 4.5: Pipes de SDF	6d	**15/08/2016**	**23/08/2016**	
Tasca 4.5.1: Reunió seguiment	0.5d	15/08/2016	15/08/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.5.2: Formació	1.5d	16/08/2016	17/08/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.5.4: Disseny	1d	17/08/2016	18/08/2016	Consultor[20%],Arquitecte[80%],Ordinador[1],VisualStudio[1]
Tasca 4.5.5: Implementació	1d	18/08/2016	19/08/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.5.5: Testeig80%	2d	19/08/2016	23/08/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]

Figura 20: Llistat i definició de les tasques del Gantt de l'etapa pipes de Signed Distance Field

A continuació es descriuen les subtasques que formen la tasca.

Tasca 4.5 Pipes de Signed Distance Field		
Descripció: Implementar les pipes per a la tècnica del Signed Distance Field (SDF). S'han creat dotze pipes. Tres que calculen l'execució de la tècnica SDF: l'execució completa, l'execució per vèrtex i l'execució per punts de mostreig. Després s'han implementat tres pipes, com les tres explicades anteriorment, afegint un booleà que atura l'execució en el moment en què es detecta la col·lisió. Finalment s'han implementat sis pipes executant inicialment les col·lisions d'esferes.		
DESCRIPCIÓ	RECURSOS	DURADA
Reunió de seguiment	Cap de projecte, desenvolupador	0.5 dia
Anàlisi	Analista, consultor, ordinador	1.5 dies
Disseny	Arquitecte, consultor, ordinador	1 dia
Implementació	Desenvolupador, consultor, ordinador, visual Studio	1 dia
Testeig	Tester, desenvolupador, consultor, ordinador, visual Studio	2 dies
TOTAL		**6 dies**

Taula 16: Descripció de la tasca 4.5 – Pipes de Signed Distance Field

4.3.4.6 Tasca 4.6: Gilbert-Johnson-Keerthi

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 4.6 *Implementació del Gilbert-Johnson-Keerthi*.

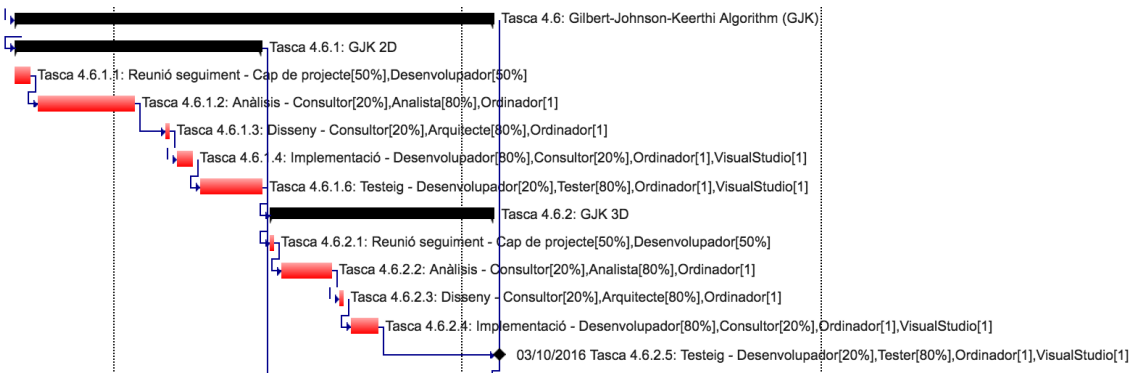


Figura 21: Diagrama de Gantt de l'etapa de Gilbert-Johnson-Keerthi

	Tasca 4.6: Gilbert-Johnson-Keerthi Algorit	**30d**	**23/08/2016**	**03/10/2016**	
	Tasca 4.6.1: GJK 2D	**16d**	**23/08/2016**	**13/09/2016**	
	Tasca 4.6.1.1: Reunió seguiment	2d	23/08/2016	24/08/2016	Cap de projecte[50%],Desenvolupador[50%]
	Tasca 4.6.1.2: Anàlisis	7d	25/08/2016	02/09/2016	Consultor[20%],Analista[80%],Ordinador[1]
	Tasca 4.6.1.3: Disseny	1d	05/09/2016	05/09/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
	Tasca 4.6.1.4: Implementació	2d	06/09/2016	07/09/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
	Tasca 4.6.1.6: Testeig	4d	08/09/2016	13/09/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]
	Tasca 4.6.2: GJK 3D	**14d**	**14/09/2016**	**03/10/2016**	
	Tasca 4.6.2.1: Reunió seguiment	1d	14/09/2016	14/09/2016	Cap de projecte[50%],Desenvolupador[50%]
	Tasca 4.6.2.2: Anàlisis	3d	15/09/2016	19/09/2016	Consultor[20%],Analista[80%],Ordinador[1]
	Tasca 4.6.2.3: Disseny	1d	20/09/2016	20/09/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
	Tasca 4.6.2.4: Implementació	3d	21/09/2016	23/09/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
	Tasca 4.6.2.5: Testeig	6d	26/09/2016	03/10/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]

Figura 22: Llistat i definició de les tasques del Gantt de l'etapa de Gilbert-Johnson-Keerthi

A continuació es descriuen les subtasques que formen la tasca.

Tasca 4.6 Gilbert-Johnson-Keerthi
Descripció: Implementar la tècnica del Gilbert-Johnson-Keerthi (GJK) amb un disseny orientat a objectes. S'ha dividit en dos tasques definides a continuació.
Durada: 30 dies
SUBTASQUES:
Tasca 4.6.1 Gilbert-Johnson-Keerthi – 2D
Descripció: Implementar la tècnica del Gilbert-Johnson-Keerthi en un sistema 2D i un disseny orientat a objectes.
Tasca 4.6.2 Gilbert-Johnson-Keerthi – 3D
Descripció: Implementar la tècnica del Separated Axis Theorem entre dos objectes OBB en un sistema 2D i un disseny orientat a objectes.

Taula 17: Descripció de la tasca 4.6 – Gilbert-Johnson-Keerthi

A la taules següents es pot observar la divisió iterativa del format d'sprints definit a l'inici del capítol, definint els recursos i les hores requerides per a les tasques 4.6.1 i 4.6.2:

DESCRIPCIÓ	RECURSOS	DURADA
Reunió de seguiment	Cap de projecte, desenvolupador	2 dies
Anàlisi	Analista, consultor, ordinador	7 dies
Disseny	Arquitecte, consultor, ordinador	1 dia
Implementació	Desenvolupador, consultor, ordinador, visual Studio	2 dies
Testeig	Tester, desenvolupador, consultor, ordinador, visual Studio	4 dies
TOTAL		**16 dies**

Taula 18: Descripció de las subtasca 4.6.1 de la tasca 4.6

DESCRIPCIÓ	RECURSOS	DURADA
Reunió de seguiment	Cap de projecte, desenvolupador	1 dia
Anàlisi	Analista, consultor, ordinador	3 dies
Disseny	Arquitecte, consultor, ordinador	1 dia
Implementació	Desenvolupador, consultor, ordinador, visual Studio	4 dies
Testeig	Tester, desenvolupador, consultor, ordinador, visual Studio	6 dies
TOTAL		**15 dies**

Taula 19: Descripció de las subtasca 4.6.2 de la tasca 4.6

4.3.4.7 Tasca 4.7: Pipes de Gilbert-Johnson-Keerthi

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 4.7 *Implementació de les pipes del Gilbert-Johnson-Keerthi.*

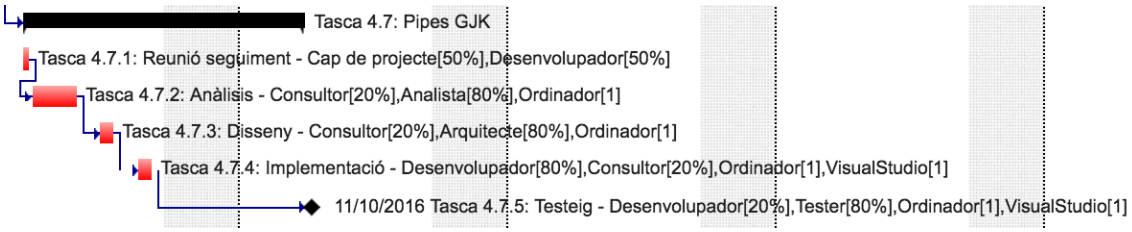


Figura 23: Diagrama de Gantt de l'etapa de pipes de Gilbert-Johnson-Keerthi

	Tasca 4.7: Pipes GJK	6d	**04/10/2016**	**11/10/2016**	
	Tasca 4.7.1: Reunió seguiment	0.5d	04/10/2016	04/10/2016	Cap de projecte[50%],Desenvolupador[50%]
	Tasca 4.7.2: Anàlisis	1.5d	04/10/2016	05/10/2016	Consultor[20%],Analista[80%],Ordinador[1]
	Tasca 4.7.3: Disseny	1d	06/10/2016	06/10/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
	Tasca 4.7.4: Implementació	1d	07/10/2016	07/10/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
	Tasca 4.7.5: Testeig	2d	10/10/2016	11/10/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]

Figura 24: Llistat i definició de les tasques del Gantt de l'etapa de pipes de Gilbert-Johnson-Keerthi

A continuació es descriuen les subtasques que formen la tasca.

Tasca 4.7 Pipes de Gilbert-Johnson-Keerthi		
Descripció: Implementar les pipes per a la tècnica del Gilbert-Johnson-Keerthi (GJK). S'han creat dues pipes, una que calcula l'execució del GJK i una altra que executa inicialment les col·lisions d'esferes.		
DESCRIPCIÓ	RECURSOS	DURADA
Reunió de seguiment	Cap de projecte, desenvolupador	0.5 dia
Anàlisi	Analista, consultor, ordinador	1.5 dies
Disseny	Arquitecte, consultor, ordinador	1 dia
Implementació	Desenvolupador, consultor, ordinador, visual Studio	1 dia
Testeig	Tester, desenvolupador, consultor, ordinador, visual Studio	2 dies
TOTAL		**6 dies**

Taula 20: Descripció de les subtasques de la tasca 4.7

4.3.4.8 Tasca 4.8: Selecció de les pipes

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 4.8 *Seleccionar les millors pipes del disseny orientat a objectes*.

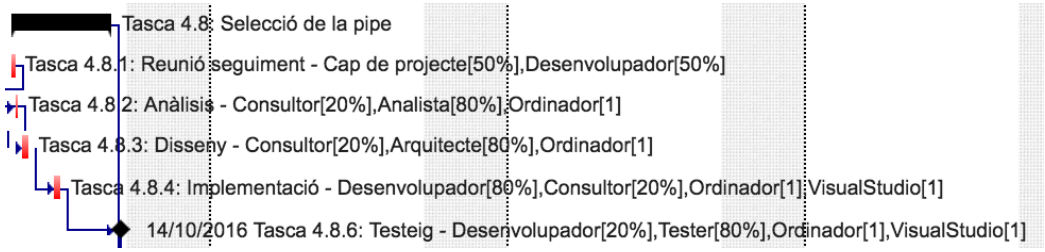


Figura 25: Diagrama de Gantt de l'etapa de selecció de les pipes

Tasca 4.8: Selecció de la pipe	3d	12/10/2016	14/10/2016	
Tasca 4.8.1: Reunió seguiment	0.25d	12/10/2016	12/10/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.8.2: Anàlisis	0.25d	12/10/2016	12/10/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.8.3: Disseny	0.5d	12/10/2016	12/10/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.8.4: Implementació	0.5d	13/10/2016	13/10/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.8.6: Testeig	1.5d	13/10/2016	14/10/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]

Figura 26: Llistat i definició de les tasques del Gantt de l'etapa de selecció de les pipes

A continuació es descriuen les subtasques que formen la tasca.

Tasca 4.8 Selecció de la pipe		
Descripció: Realitzar una comparativa entre les 16 pipes. Fer 1000 execucions de les 16 pipes amb un conjunt de 100 d'objectes i extreure les estadístiques per poder concloure quines són les dues combinacions més optimes.		
DESCRIPCIÓ	RECURSOS	DURADA
Reunió de seguiment	Cap de projecte, desenvolupador	0.25 dia
Anàlisi	Analista, consultor, ordinador	0.25 dia
Disseny	Arquitecte, consultor, ordinador	0.5 dia
Implementació	Desenvolupador, consultor, ordinador, visual Studio	0.5 dia
Testeig	Tester, desenvolupador, consultor, ordinador, visual Studio	1.5 dies
TOTAL		**3 dies**

Taula 21: Descripció de les subtasques de la tasca 4.8

4.3.4.9 Tasca 4.9: Pas al disseny orientat a dades

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 4.9 *Implementació de les dues millors pipes en un disseny orientat a dades.*

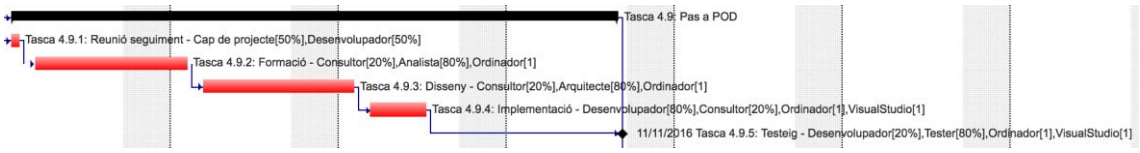


Figura 27: Diagrama de Gantt de l'etapa de pas al disseny orientat a dades

	Tasca 4.9: Pas a POD	**20d**	**17/10/2016**	**11/11/2016**	
	Tasca 4.9.1: Reunió seguiment	1d	17/10/2016	17/10/2016	Cap de projecte[50%],Desenvolupador[50%]
	Tasca 4.9.2: Formació	5d	18/10/2016	24/10/2016	Consultor[20%],Analista[80%],Ordinador[1]
	Tasca 4.9.3: Disseny	5d	25/10/2016	31/10/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
	Tasca 4.9.4: Implementació	3d	01/11/2016	03/11/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
	Tasca 4.9.5: Testeig	6d	04/11/2016	11/11/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]

Figura 28: Llistat i definició de les tasques del Gantt de l'etapa de pas al disseny orientat a dades

A continuació es descriuen les subtasques que formen la tasca.

Tasca 4.9 Pas a Disseny Orientat a Dades		
Descripció: Implementar les dues pipes escollides en un disseny orientat a dades.		
DESCRIPCIÓ	RECURSOS	DURADA
Reunió de seguiment	Cap de projecte, desenvolupador	1 dia
Anàlisi	Analista, consultor, ordinador	5 dia
Disseny	Arquitecte, consultor, ordinador	5 dia
Implementació	Desenvolupador, consultor, ordinador, visual Studio	3 dia
Testeig	Tester, desenvolupador, consultor, ordinador, visual Studio	6 dies
TOTAL		**20 dies**

Taula 22: Descripció de les subtasques de la tasca 4.9

4.3.4.10 Tasca 4.10: Selecció del disseny

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 4.10 *Seleccionar el millor disseny i la millor pipe*.

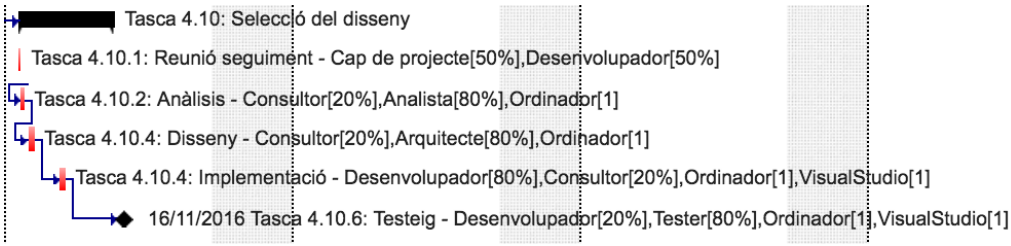


Figura 29: Diagrama de Gantt de l'etapa de selecció del disseny

Tasca 4.10: Selecció del disseny	3d	14/11/2016	16/11/2016	
Tasca 4.10.1: Reunió seguiment	0.25d	14/11/2016	14/11/2016	Cap de projecte[50%],Desenvolupador[50%]
Tasca 4.10.2: Anàlisis	0.25d	14/11/2016	14/11/2016	Consultor[20%],Analista[80%],Ordinador[1]
Tasca 4.10.4: Disseny	0.5d	14/11/2016	14/11/2016	Consultor[20%],Arquitecte[80%],Ordinador[1]
Tasca 4.10.4: Implementació	0.5d	15/11/2016	15/11/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1],VisualStudio[1]
Tasca 4.10.6: Testeig	1.5d	15/11/2016	16/11/2016	Desenvolupador[20%],Tester[80%],Ordinador[1],VisualStudio[1]

Figura 30: Llistat i definició de les tasques del Gantt de l'etapa de selecció del disseny

A continuació es descriuen les subtasques que formen la tasca.

Tasca 4.10 Selecció del disseny		
Descripció: Realitzar una comparativa entre els dos sistemes de disseny. Fer 1000 execucions de les 4 pipes amb un conjunt de 100 d'objectes i extreure les estadístiques per poder concloure quina de les solucions és la més optima.		
DESCRIPCIÓ	RECURSOS	DURADA
Reunió de seguiment	Cap de projecte, desenvolupador	0.25 dia
Anàlisi	Analista, consultor, ordinador	0.25 dia
Disseny	Arquitecte, consultor, ordinador	0.5 dia
Implementació	Desenvolupador, consultor, ordinador, visual Studio	0.5 dia
Testeig	Tester, desenvolupador, consultor, ordinador, visual Studio	1.5 dies
TOTAL		**3 dies**

Taula 23: Descripció de les subtasques de la tasca 4.10

4.3.5 Etapa 5: Implantació

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 5.1 *Implantació.*

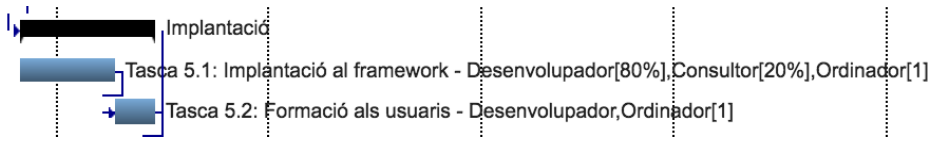


Figura 31: Diagrama de Gantt de l'etapa implantació

	Implantació	14d	17/11/2016	07/12/2016	
⚡	Tasca 5.1: Implantació al framework	10d	17/11/2016	01/12/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1]
	Tasca 5.2: Formació als usuaris	4d	01/12/2016	07/12/2016	Desenvolupador,Ordinador[1]

Figura 32: Llistat i definició de les tasques del Gantt de l'etapa implantació

A continuació es descriuen les tasques que formen la tasca.

Tasca 5.1: Implantació dins del framework
Descripció: Afegir el mòdul del motor de col·lisions amb la tècnica i el disseny més eficients dins del framework.
Recursos: Cap del projecte, desenvolupador, ordinador
Durada: 10 dies

Taula 24: Descripció de la tasca 5.1

Tasca 5.2 Formació d'usuaris
Descripció: Preparar documentació per a l'empresa explicant el funcionament del motor. Presentar el motor a l'empresa. Formar als usuaris que vagin a fer servir el framework i el motor de col·lisions. Se'ls formarà per a que puguin realitzar procediments d'actuació en el cas que sigui necessari.
Recursos: cap del projecte, desenvolupador, ordinador, usuaris
Durada: 4 dies

Taula 25: Descripció de la tasca 5.2

4.3.6 Etapa 6: Feina futura

En el següent diagrama de Gantt es pot observar la planificació definida per a la realització de la tasca 6.1 *Feina futura.*

Figura 33: Diagrama de Gantt de l'etapa de feina futura

	⊟ **Feina futura**	10d	07/12/2016	21/12/2016	
	Tasca 6.1: Funcionalitats extra	10d	07/12/2016	21/12/2016	Cap de projecte[50%],Desenvolupador[50%],Ordinador[1]

Figura 34: Llistat i definició de les tasques del Gantt de l'etapa de feina futura

Tasca 6.1 Funcionalitats extres
Descripció: Recollir les possibles millores i funcionalitat que es podrien implementar un cop es finalitzi el projecte. Definir el pla de futur del projecte.
Recursos: Cap del projecte, desenvolupador, ordinador
Durada: 10 dies

Taula 26: Descripció de la tasca 6.1

4.4 Riscos

Hi ha certs factors que poden afectar el calendari previst pel projecte. La següent taula mostra els possibles obstacles que poden aparèixer un cop es porti a terme el projecte:

RISC	GRAVETAT
R1 - No implementar la tècnica del Separated Axis Theorem en un disseny orientat a objectes	Baix
R2 - No implementar la tècnica del Signed Distance Field en un disseny orientat a objectes	Mitjà
R3 - No implementar la tècnica del Gilbert-Johnson-Keerthi en un disseny orientat a objectes	Alt
R4 - No implementar la tècnica de col·lisió d'esferes en un disseny orientat a objectes	Baix
R5 - No implementar cap tècnica en disseny orientat a objectes	Alt
R6 - No implementar les pipes del Separated Axis Theorem	Baix
R7 - No implementar les pipes del Signed Distance Field	Mitjà
R8 - No implementar les pipes del Gilbert-Johnson-Keerthi	Alt
R9 - No implementar les pipes de la tècnica del Gilbert-Johnson-Keerthi en un disseny orientat a dades	Alt
R10 - Poca experiència en motors de col·lisions i tècniques de col·lisions	Mitjà
R11 - Entrada de projectes més prioritaris	Alt
R12 – Per a la fase d'implantació es necessari que el framework estigui en un estat funcional	Alt
R13 – Per a que el framework pugui calcular les col·lisions ha de rebre la informació dels objectes correctament	Alt

Taula 27: Possibles obstacles sobre el projecte

Tots aquests riscos han estat considerats i s'han previst alternatives al pla d'acció explicats en el següent apartat.

4.5. Valoració d'alternatives i pla d'acció

En el moment de realitzar GEP el projecte es trobava a la meitat del procés (concretament a la Tasca 4.9). Ja s'havia realitzat la implementació de les tres tècniques en un disseny orientat a objectes per la qual cosa els riscos R1 al R8 no necessitaven un pla d'acció, ja que no s'havien produït.

El risc R10 es va tenir en consideració a l'hora de definir la planificació inicial. La falta d'experiència es pot minimitzar assignant temps addicional a les tasques en les quals es preveu més formació. Exemple: la inexperiència en àrees de matrius 3D o la desconeixença de les diferents tècniques de detecció de col·lisions pot suposar un

temps addicional, especialment durant la implantació. Per solucionar aquests obstacles es va assignar temps addicional a les subtasques de formació i recerca durant les diferents etapes i tasques del projecte.

L'ús de les metodologies àgils permet detectar i corregir les desviacions a les reunions del final de cada sprint. Durant el projecte ha sorgit el risc R11 (entrada de projectes més prioritaris). La falta de disponibilitat d'algun component de l'equip s'ha pogut solucionar modificant l'ordre de les tasques, avançant aquelles que no tenien dependència del component que faltava i minimitzant el temps de la desviació. Exemple: Durant tot el procés dels sprints, algun membre ha estat requerit en altres projectes amb més prioritat que aquest. En un cas així, en cas d'aturar el procés definit, s'ha pogut seguir amb el testeig d'una tasca anterior o amb la formació i recerca d'una tasca futura.

Tot i aplicar aquestes alternatives s'han produït algunes desviacions durant l'etapa d'sprints per la necessitat de treballar en altres projectes prioritaris, concretament entre la implementació de la tècnica SAT i la tècnica de SDF (Tasques 4.1 i 4.2) i entre SDF i la col·lisió d'esferes (Tasques 4.2 i 4.3). No hi ha hagut un problema greu, ja que el projecte no tenia una data de finalització concreta i s'han pogut desplaçar les posteriors tasques sense problemes. Aquesta desviació ha provocat que el projecte en comptes de finalitzar al novembre s'ajornés al gener, però per l'empresa aquesta desviació no s'ha tingut en compte, ja que les desviacions s'han produït per la necessitat de realitzar algun projecte més prioritari.

Així doncs, fins a la finalització de la Tasca 4.8 es va acabar segons el temps esperat i per tant no calia definir alternatives.

De cara al desenvolupament de les tasques que mancaven durant l'època de GEP, es va definir que es procediria de la mateixa manera explicada en cas que entrés un altre projecte. Això si, en el moment en què les desviacions provoquessin que la planificació s'allargués fins al gener, caldria aplicar altres alternatives:

- Dedicació completa al projecte: el projecte passaria a ser considerat prioritari per l'empresa i per tant es pogués dedicar una jornada completa de 8 hores.
- Afegir nous recursos: s'afegirien un o més recursos de cara a compensar el temps desviat.

Respecte a la realització de la ‘Tasca 5 – Implantació’ s’esperava que no hi hagués gaires problemes pel fet que s’ha dedicat gran temps a testejar cadascuna de les diferents tasques realitzades. El risc R12 va aparèixer, inicialment es va decidir que la implantació era un objectiu secundari del projecte per part de l’empresa i es podria donar com a finalitzat sense la seva realització. Però donat el marge de temps produït pel desplaçament de la data de lectura s’ha realitzat la implantació abans de la finalització del projecte, això si, ha implicat que la seva realització es fes fora de la jornada laboral (com s’explica a l’apartat “*13-Implantació*”).

Respecte al risc R13, les dades que rep el motor han d’estar en un format correcte, ja que durant el projecte s’ha treballat amb dades pròpies creades segons els requeriments definits. S’ha comprovant que les dades que rep el motor compleixen els requisits de dades.

Tot i la complexitat del projecte, el gran nombre de tasques a realitzar, i el desplaçament de la data de lectura finalment s’ha pogut assolir amb èxit el projecte.

5. Pressupost inicial

5.1. Identificació i estimació dels costos

Aquest projecte només té en compte els costos de creació del motor.

Tots els costos que apareixen en aquest document han estat facilitats pel cap de projecte.

RECURS	PREU PER HORA
Cap de projecte	22€/hora
Analista	15€/hora
Arquitecte	18€/hora
Desenvolupador	12.5€/hora
Consultor	15€/hora
Tester	10€/hora

Taula 28: Relació de preu per hora de cada recurs

5.1.1. Costos de creació del motor

La creació de la plataforma consta de les etapes definides anteriorment a la planificació. A continuació s'analitzen els costos associats:

5.1.1.1 Costos directes

Els costos directes s'han estimat per cada etapa del projecte i les tasques que les formen estan definides al capítol "*4-Planificació*" i al diagrama de Gantt de *l'Annex 1*. S'han tingut en compte les hores dels recursos humans. A *l'Annex 2* hi ha una taula amb els temps de dedicació de cada recurs per cadascuna de les etapes i tasques. Aquestes dades són les que s'han utilitzat per realitzar els càlculs dels costos directes de cada etapa.

La suma total dels costos directes és de **10.896,25€.**

- **Definició de requisits:** el cost d'aquesta etapa és de **1.544,00€.**

CONCEPTE	UNITATS	PREU/UNITAT	TOTAL
Cap de projecte	32h	22,00€/h	**704,00€**
Analista	56h	15,00€/h	**840,00€**

Taula 29: Costos directes de l'etapa de definició de requisits de la creació del motor.

- **Gestió del projecte:** el cost d'aquesta etapa és de **440,00€.**

CONCEPTE	UNITATS	PREU/UNITAT	TOTAL
Cap de projecte	20h	22,00€/h	**440,00€**

Taula 30: Costos directes de l'etapa de pressupost de la creació del motor.

- **Disseny del sistema:** el cost d'aquesta etapa és de **360,00€.**

CONCEPTE	UNITATS	PREU/UNITAT	TOTAL
Arquitecte	20h	18,00€/h	**360,00€**

Taula 31: Costos directes de l'etapa de disseny del sistema de la creació del motor.

- **Sprints:** el cost d'aquesta etapa és de **7142,25€.**

CONCEPTE	UNITATS	PREU/UNITAT	TOTAL
Cap de projecte	21.5h	22,00€/h	**473,00€**
Analista	108h	15,00€/h	**1.620,00€**
Arquitecte	65.5h	18,00€/h	**1.179,00€**
Desenvolupador	95.1h	12,50€/h	**1.188,75€**
Consultor	60.1h	15,00€/h	**901,50€**
Tester	178h	10,00€/h	**1.780,00€**

Taula 32: Costos directes de l'etapa d'implementació de la creació del motor.

- **Implantació:** el cost d'aquesta etapa és de **720,00€.**

CONCEPTE	UNITATS	PREU/UNITAT	TOTAL
Desenvolupador	48h	12,50€/h	**600,00€**
Consultor	8h	15€/h	**120,00€**

Taula 33: Costos directes de l'etapa d'implantació de la creació del motor.

- **Feina futura:** el cost d'aquesta etapa és de **690€.**

CONCEPTE	UNITATS	PREU/UNITAT	TOTAL
Cap de projecte	20h	22,00€/h	**440,00€**
Desenvolupador	20h	12,5€/h	**250,00€**

Taula 34: Costos directes de l'etapa de feina futura de la creació del motor.

5.1.1.2 Costos indirectes

No s'ha adquirit maquinari ni programari específic per aquest projecte, ja que l'empresa ja en disposa per la seva activitat habitual. No s'han tingut en compte les

despeses de subministrament (aigua, electricitat, Internet) o lloguer perquè l'impacte que hi té aquest projecte no és quantificable i en tot cas seria significatiu respecte tota l'activitat que es duu a terme a l'empresa.

En tot cas s'han considerat les amortitzacions de maquinari i programari utilitzades pel projecte. S'ha suposat que el coeficient lineal mínim d'amortització és del 25% (fixat per l'Agència Tributaria[12]) i que el projecte té una durada aproximada de 212 dies.

La suma total dels costos indirectes és de **125,69€.**

A continuació es llista el programari utilitzat durant el projecte:

CONCEPTE	COST TOTAL	AMORTITZACIÓ
Visual Studio[13]	454,54€	**26,54€**
Trello	0€	**0,00€**
Google Drive	0€	**0,00€**
Bitbucket	600€	**0,70€**
Gantter	0€	**0,00€**
TOTAL		**27,24€**

Taula 35: Costos de l'amortització del programari utilitzat durant el projecte.

Justificació:

Gran part del programari utilitzat és gratuït. Les úniques llicències adquirides són les del Visual Studio 2015, una subscripció per la qual s'ha de pagar 37,88€ mensuals, i l'accés al repositori Bitbucket amb un cost de 600€ mensuals.

Càlculs:

Visual Studio: [454,54€ (subscripció anual) x 0,25 x 341,1 hores (temps desenvolupador i tester)] / (4 hores * 365 dies) = 26,54€

Bitbucket: [600€ (50€ mensuals per 50 usuaris) x 0,25 x 341,1 hores (temps desenvolupador i tester)] / [(4 hores * 365 dies) * (50 usuaris)] = 0,7€

A continuació es llista el maquinari utilitzat per cada un dels membres del projecte:

CONCEPTE	COST TOTAL	AMORTITZACIÓ
Cap projecte	900	**14,40€**
Arquitecte	900	**25,27€**
Analista	700	**20,26€**

Desenvolupador	700	**18,97€**
Consultor	700	**08,22€**
Tester	700	**21,33€**
TOTAL		**98,45€**

Taula 36: Costos de l'amortització del maquinari utilitzat durant el projecte.

Justificació:

El cost total dels ordinadors s'ha obtingut a partir de la quantitat per defecte que es destina a cada lloc de treball de l'empresa.

Càlcul (cap de projecte): [900€ (ordinador) x 0,25 x 93,5 hores (temps)] / (4 hores * 365 dies) = 14,40€

5.1.1.3 Contingència

La partida de contingència d'aquest pressupost serà del 15% de la suma dels costos directes i indirectes amb un total de **1.653,29€.**

(10.896,25€ + 125,69€) x 0,15 = 1653,29€

5.1.1.4 Imprevistos

S'ha reservat una part del pressupost per possibles imprevistos detectats durant la planificació com indisponiblitat d'algun membre de l'equip (I1) o inexperiència en l'ús d'alguna eina (I2).

La suma total dels imprevistos és de **728,00€.**

IMPREVIST	OCURRÈNCIA	HORES	COST ESTIMAT	EXPOSICIÓ AL RISC
I1	0,7	20	560,00€	**392,00€**
I2	0,8	15	420,00€	**336,00€**
TOTAL				**728,00€**

Taula 37: Costos dels imprevistos del projecte

El coeficient d'ocurrència és la probabilitat que es doni aquest imprevist. Per calcular el cost estimat de cada imprevist s'han multiplicat les hores que comportaria pel cost d'un desenvolupador, que en tots dos casos és qui es veuria més perjudicat.

Finalment, per calcular el cost total del projecte aplicant l'impost de l'IVA (21 %):

CONCEPTE	TOTAL
Costos directes	**10.896,25€**
Costos indirectes	**125,69€**
Contingència	**1.653,29€**
Imprevistos	**728,00€**
Total sense IVA	**13.403,23€**
TOTAL	**16.217,90€**

Taula 38: Cost total de creació del motor.

5.2 Control de gestió

El projecte ha estat pressupostat detalladament, i s'han tingut en compte totes les condicions necessàries perquè no es produeixi un desajust. La part que pot suposar un cost extra són les parts d'implementació de l'aplicació i de la implantació dins de framework.

En cas que es produís una desviació que impliques que la data de finalització fos al gener, es va considerar aplicar les alternatives descrites a l'apartat *"4.5-Valoració d'alternatives i pla d'acció".*

A continuació es descriuen els càlculs necessaris per tenir en compte aquestes desviacions.

5.2.1 Dedicació completa per part dels recursos implicats:

Es considera que per resoldre les desviacions els recursos han de disposar d'una dedicació completa en el projecte, d'aquesta manera podrien dedicar 8 hores en comptes de 4 hores. Caldria doncs afegir al pressupost inicial el cost del desviament explicat a l'apartat 5.2.3.

5.2.2 Afegir un o més recursos:

En aquest cas faria falta saber quin tipus de recurs és el necessari i dividir el temps desviat per les hores que aquest nou recurs li pot dedicar. Caldria doncs afegir al pressupost inicial el cost del desviament explicat a l'apartat 5.2.3.

5.2.3 Explicació dels càlculs de les desviacions

Donat que únicament es podrien produir desviacions a les tasques que mancaven per fer (Tasta 4.9, Tasca 4.10, Tasca 5.1, Tasca 5.2 i Tasca 6.1), s'han realitzat els càlculs tenint en compte només els recursos implicats en aquestes tasques (cap de projecte, analista, arquitecte, desenvolupador, consultor, tester).

En els dos casos definits el cost de desviament es pot calcular segons els següents criteris:

5.2.3.1 Desviament dels costos directes:

En cas que algun recurs realitzi més hores de les que s'ha planificat inicialment, es veuran afectats els costos directes. Caldrà afegir el cost de cada recurs segons la següent definició:

Cost del desviament = preu (recurs) x hores (dedicades de més el recurs)

Per exemple, suposant que hi ha hagut una desviació de 8 hores per part del desenvolupador i del tester al realitzar la tasca 4.9, caldria sumar al pressupost **180€** per la part de costos directes**.**

RECURS	CALCULS	COST PER AFEGIR
Hores extres desenvolupador	12,50€ * 8 hores	**100€**
Hores extres tester	10,00€ * 8 hores	**80€**
TOTAL		**180€**

Taula 39: Exemple del càlcul de desviació dels costos directes

Així doncs caldria afegir 180€ de desviacions als 10.555,75€ de costos directes, que quedarien amb un valor de **10.735,75€.**

5.2.3.2 Desviament en els costos indirectes

De la mateixa manera les desviacions en el costos indirectes es realitzarien mitjançant la suma de les desviacions dels costos del programari i del maquinari.

5.2.3.2.1 Desviament en el programari

En cas que el desenvolupador i/o el tester realitzin més hores que les que s'han planificat inicialment, també es veurà afectat el càlcul dels costos indirectes i caldrà afegir el cost del maquinari que facin servir segons la següent definició:

Cost del visual Studio: [454,54€ (subscripció anual) x 0,25 x (temps desenvolupador i tester)] / (4 hores * 365 dies) Cost del bitbucket: [600€ (50€ mensuals per 50 usuaris) x 0,25 x (temps desenvolupador i tester)] / [(4 hores * 365 dies) * (50 usuaris)]

Seguint amb l'exemple anterior, caldria afegir les 16 hores noves del desenvolupador i del tester a les 341,1 hores que hi havia contemplades. Així doncs els càlculs s'hauran de realitzar amb el valor de 357,10 hores de dedicació del desenvolupador i del tester i caldria sumar al pressupost **30,20€** de costos indirectes de programari.

PROGRAMARI	JUSTIFICACIÓ	COST
Visual Studio	[454,54€ (subscripció anual) x 0,25 x **357,10 hores** (temps desenvolupador i tester)] / (4 hores * 365 dies)	**27,79€**
Bitbucket	[600€ (50€ mensuals per 50 usuaris) x 0,25 x **357,10 hores** (temps desenvolupador i tester)] / [(4 hores * 365 dies) * (50 usuaris)]	**0,73€**
TOTAL		**30,20€**

Taula 40: Justificació als càlculs del desviament del programari

5.2.3.2.2 Desviament en el maquinari

En cas que s'incorpori un nou recurs o si s'incrementen les hores d'un recurs caldrà afegir el cost del maquinari que faci servir segons la següent definició:

[Preu ordinador x 0,25 x hores recurs] / 4 hores * 365 dies

Seguint amb l'exemple anterior, caldria afegir al desenvolupador les 8 hores noves a les 158,3 hores que hi havia contemplades i al tester les 8 hores noves a les 178 hores que hi havia contemplades. Així doncs els càlculs s'hauran de realitzar amb el valor de

166,3 hores totals pel desenvolupador i 186 hores pel tester i caldria sumar al pressupost **42,22€** de costos indirectes de maquinari.

MAQUINARI	JUSTIFICACIÓ	COST
Desenvolupador	[700€ x 0,25 x **166,3 hores**] / 4 hores * 365 dies	**19.93€**
Tester	[700€ x 0,25 x **186 hores**] / 4 hores * 365 dies	**22.29€**
TOTAL		**42,22€**

Taula 41: Justificació als càlculs del desviament del maquinari

Així doncs caldria afegir 72,44€ de desviacions als 125,89€ de costos indirectes, que quedarien amb un valor de **198,13€.**

5.2.3.3 Desviament en la contingència

En cas que hi es produeixin desviaments que afectin els costos directes i/o indirectes serà necessari recàlcular el valor de la contingència segons la següent definició:

(costos directes + costos indirectes) x 0,15

Seguint amb l'exemple anterior, tenint en compte que el valor dels costos directes i indirectes són 10.735,75€ i 198,13€ respectivament, la contingència seria de **1.640,08€.**

6. Sostenibilitat

A continuació es presenta l'estudi de sostenibilitat realitzat en referència al projecte. A la següent taula s'exposen els resultats de l'estudi en forma de matriu, explicats detalladament més endavant.

	PPP		VIDA ÚTIL		RISCOS	
Ambiental	Consum del disseny [0:10]	**8**	Petjada ecològica [0:20]	**16**	Riscos ambientals [-20:0]	**-5**
Econòmica	Factura [0:10]	**8**	Pla de viabilitat [0:20]	**18**	Riscos econòmics [-20:0]	**0**
Social	Impacte personal [0:10]	**10**	Impacte social [0:20]	**18**	Riscos socials [-20:0]	**-10**
Rang de sostenibilitat	[0:30]	**26**	[0:60]	**52**	[-60:0]	**-15**
TOTAL						**63**

Taula 42: Matriu de sostenibilitat

6.1. Econòmica

L'avaluació dels costos s'ha realitzat en els apartats anteriors i s'han tingut en compte els imprevistos que podien afectar la planificació.

Es considera que és un projecte viable donat els costos obtinguts, ja que amb l'existència d'un motor de col·lisions propi l'empresa no haurà d'invertir en llicències de programari extern.

Un cop especificada cada tasca i coneguda la seva importància, s'ha definit la planificació tenint en compte la seva importància i dificultat. També s'han assignat els recursos necessaris a cadascuna de les tasques per poder assolir els objectius en els terminis adients. En cas de voler realitzar aquest projecte en menys temps, es podria incorporar més recursos, però això implicaria en un increment dels costos.

No ha estat necessària cap tipus de col·laboració amb entitats externes a l'empresa, totes les tasques definides s'han dut a terme per recursos de l'empresa.

6.2. Social

Cada cop més persones fan ús de planificadors i organitzadors com a eines per fer una reforma o organitzar les seves llars. Mitjançant el nou motor de col·lisions l'empresa podrà millorar el producte final, mostrant imatges més realistes per a l'usuari. La millora del motor te impacte en tots els seus usuaris, que potencialment poden arribar a ser una proporció molt gran de la societat.

Pels motius exposats en els apartats de l'estat de l'art i formulació del problema l'empresa va considerar necessari la creació d'un nou motor de col·lisions.

L'únic col·lectiu perjudicat per la creació d'un motor de col·lisions propi per part de l'empresa és el de les empreses externes que ofereixen motors de físiques com és el cas de Unity.

6.3 Ambiental

El punt més important recau en el fet que les empreses i les persones no requereixen crear un moble físic per a fer-se una idea del que volen comprar, sinó que mitjançant els planificadors poden tenir una visió prou realista sense la necessitat de malbaratar fusta i perjudicar a la natura.

A més a més els recursos materials necessaris per dur a terme aquest projecte i, que s'han definit anteriorment, són propis de l'empresa, així doncs no s'ha fet ús de cap material manufacturat.

Tenint en compte que cada usuari ha fet ús d'un ordinador i que ha treballat les hores quantificades anteriorment podem dir que aquest projecte ha tingut un consum de 66.880kwh i 43kgCO2, tenint en compte que un ordinador té un cost de 80kwh, i un consum de 52gCO2/h i s'han definit un total de 836 hores planificades. L'única contaminació que es genera és la del consum de CO2 del maquinari.

Si l'empresa continues treballant sense fer ús del nou motor, tindria un cost més elevat, ja que el nou motor serà més eficient que el que s'utilitza actualment i per aquest motiu podrà realitzar els càlculs en menys temps. Podem dir que es disminueix la petjada ecològica.

L’empresa ja disposa dels recursos materials d’aquest projecte, ordinadors i llicències per aquest motiu no es considera que hi hagi un cost extra. Un cop finalitzat aquest projecte, els recursos es poden reutilitzar per als altres projectes.

L’empresa podrà fer ús del motor de col·lisió en la totalitat dels projectes de l’empresa actuals i futurs.

7. Teoria

7.1 Els motors de col·lisions[14]

Existeixen molts mètodes per detectar col·lisions que es diferencien per complexitat i temps de càlcul, alguns són adequats per a un tipus de geometria i altres són genèrics i poden ser utilitzats amb qualsevol tipus d'objecte.

La detecció de col·lisions és un problema molt costós en temps de càlcul, ja que segons quin algoritme s'utilitzi pot arribar-se al cas d'haver de comparar tots els vèrtexs d'un objecte amb tots el vèrtex de l'altre. Per aquest motiu és freqüent establir diversos nivells de profunditat a l'hora de detectar una col·lisió, de forma que primer s'apliquin els tests més gruixuts, que requereixen menys temps d'execució com a mitjà de filtre, i després aplicar els tests més precisos i que requereixen major quantitat de càlcul amb els objectes que no han sigut descartats en nivells anteriors.

És important destacar que els mètodes de detecció de col·lisions utilitzen únicament conceptes geomètrics i no tenen en compte les característiques dinàmiques de l'objecte com la velocitat o la massa, sinó que únicament tenen en compte la localització en l'espai (translació, rotació) i la geometria o forma (escalat).

Dins la fase de detecció de col·lisions explicada a la l'apartat *"1.2 El motors de simulació de físiques"* es poden diferenciar dues formes de càlcul, discreta i continua. En general, la detecció de col·lisions continua és més costosa i només s'utilitza en cas que sigui realment necessària. La majoria de motors fan ús de mètodes discrets, és a dir, si hi ha un intersecció entre els objectes i el motor de físiques actua en conseqüència separant-los (en cas que sigui necessari).

7.1.1 Detecció de col·lisions

En una escena hi ha un nombre determinat d'objectes i cada objecte te definides unes propietats de transformació (una posició, un escalat i una rotació) explicades en el capítol ***"7.2-Conceptes matemàtics bàsics"*.**

Un motor de col·lisions realitza un recorregut (un bucle) per parelles entre tots els objectes de l'escena i comprova si existeix intersecció entre elles.

7.1.2 Les Bounding Volume

Fer el càlcul de la col·lisió d'un objecte complex amb moltes cares, angles i/o racons pot arribar a ser molt costós. Per aquest motiu, per cada objecte és defineix un **Bounding Volume**, una "malla" que l'encapsula completament. Generalment es busca la malla amb la forma més simple possible i per això s'acostuma a fer ús de malles poligonals. Les més utilitzades per simplicitat són els cercles i els rectangles (en 2D), i les esferes i els prismes rectangulars (en 3D). Dins el rectangles i els prismes rectangulars hi ha definida una jerarquia segons les orientacions de la malla.

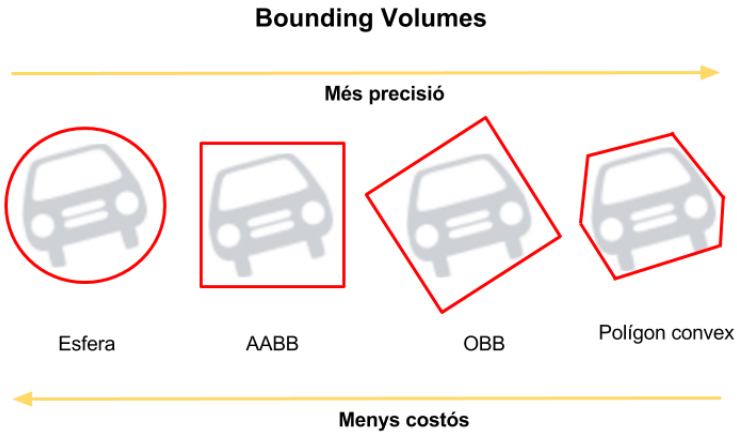


Figura 35: Bounding Volumes

Generalment es busca la malla més estreta que envolti totes les parts d'un objecte, aquesta malla més petita amb forma de prisma rectangular s'anomena la **Mínimum Bounding Box (MBB)**. En un espai 2D aquesta es l'àrea mínima i en el 3D es tractaria del volum mínim.

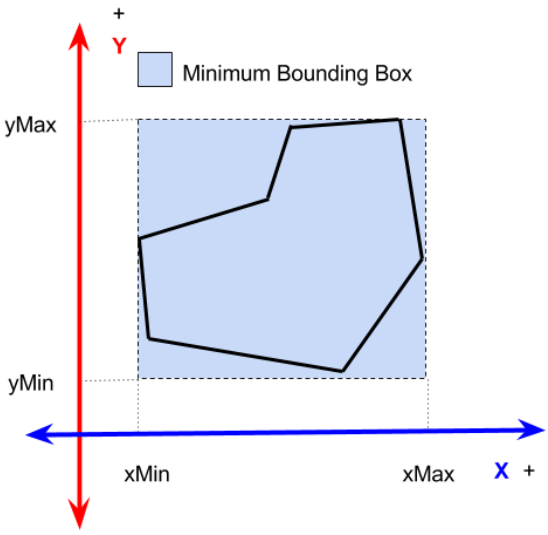


Figura 36: Mínimum Bounding Box

Segons el sistema d'eixos de l'objecte es poden diferenciar dos tipus de Bounding Box, les **Axis Aligned Bounding Box** (AABB) i les **Oriented Bounding Box** (OBB). Les AABB tenen els eixos de coordenades de l'objecte (eixos locals) alineats als eixos de coordenades de l'escena (eixos de món), en canvi les OBB tenen els eixos de coordenades de l'objecte rotats respecte els eixos de l'escena i per tant no coincideixen.

Les **Bounding Esfèriques** són la opció més barata en quant a cost de computació, però no són tant precises com les AABB. En canvi les OBB són les més costoses de calcular.

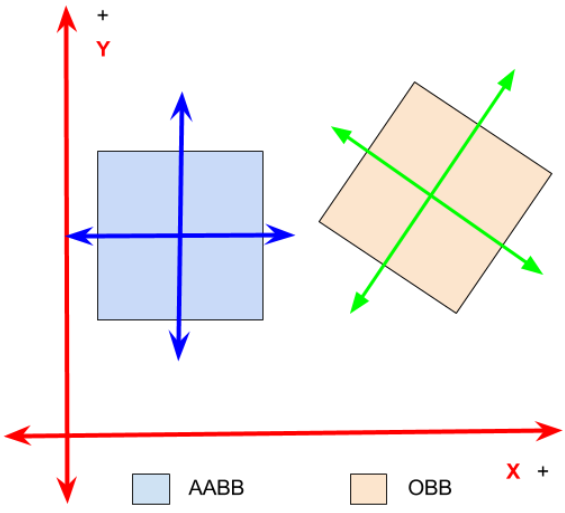


Figura 37: AABB vs OBB

7.1.3 Fases de la detecció de col·lisions

La detecció de col·lisions es pot dividir en dues fases, Broad Phase i Narrow Phase.

A la primera fase, **Broad Phase**, es determina per tot el conjunt d'objectes d'una escena quins "poden" estar col·lisionant. Aquesta part és pot implementar fent ús d'una estructura de dades específica (un **Boundary Volume Tree**[15], un **Octree**[16]...) que permet conèixer on esta col·locat cada objecte i quins té al voltant.

L'ús d'una estructura d'arbre com la del Boundary Volume Hierarchy pot implicar que la detecció de col·lisions passi a tindre un cost logarítmic en comptes d'un cost exponencial.

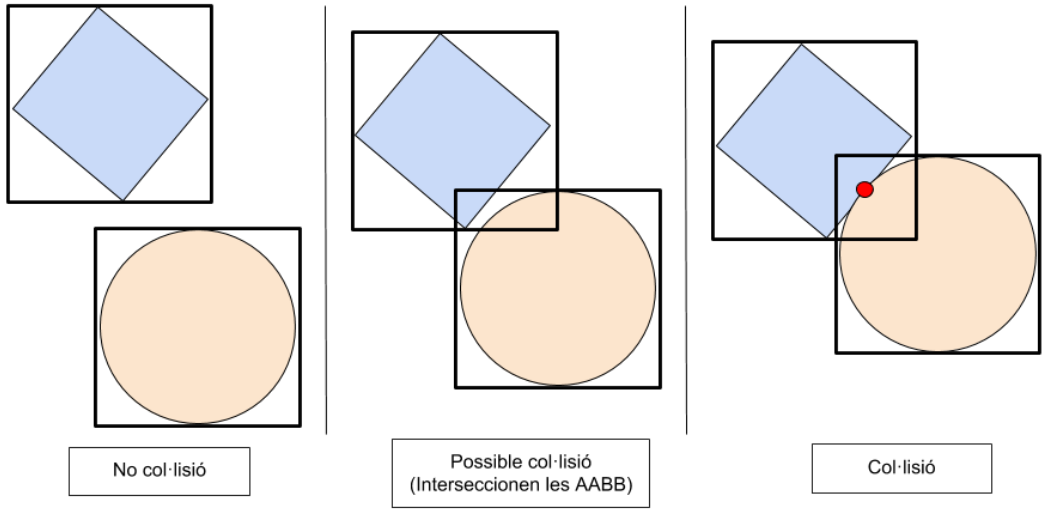


Figura 38: Classificació de tipus de col·lisions

La implementació de la Broad Phase és una optimització dins el motor de col·lisions ja que un dels requeriments del motor es que els càlculs es realitzin en un temps ràpid. El projecte requereix que la detecció i resolució de les col·lisions estigui en l'ordre de centenars de microsegons ja que l'execució total dels sistema està en l'ordre de 16-30 milisegons i resoldre les col·lisions no deuria ocupar més d'un 10% del còmput total. Fer ús d'estructures de dades que afavoreixen realitzar la criba d'objectes que no intersseccionen és un punt important dins d'un motor per evitar càlculs innecessaris.

Així doncs mitjançant les estructures de dades i les Bounding Volum dels objectes es realitza aquesta criba d'objectes.

A la segona fase, **Narrow Phase**, pels objectes definits com a “possibles objectes amb col·lisió”, es fa la comprovació mitjançant un algoritme de detecció de col·lisions per comprovar si es produeix o no la intersecció. A més a més, per calcular la resposta física de la col·lisió, és necessari obtenir la geometria de la intersecció, és a dir, un vector director que mogui els dos objectes fora de la intersecció per la distància més curta possible (profunditat de la penetració).

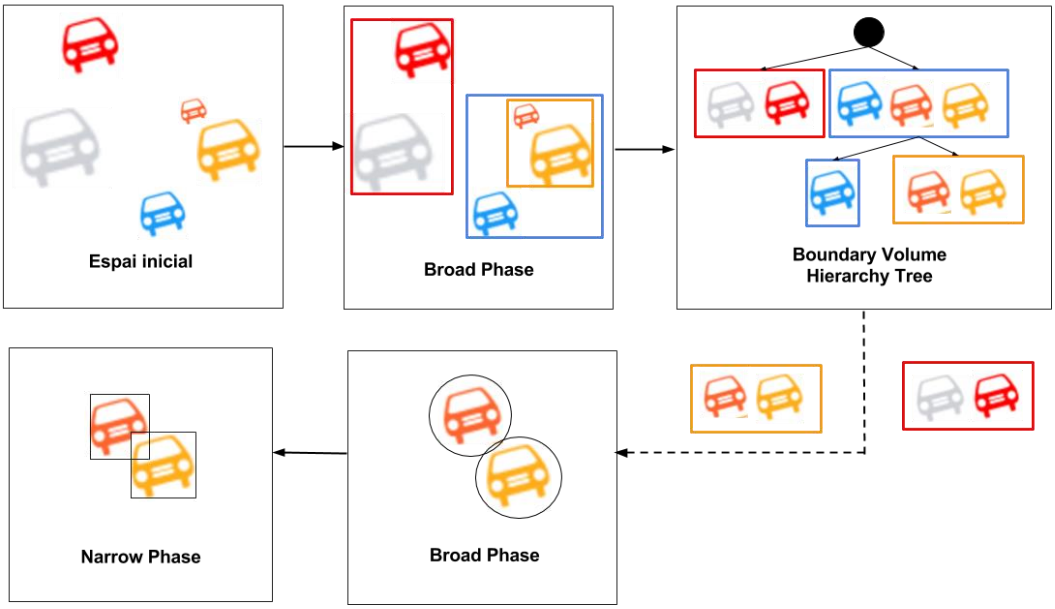


Figura 39: Exemple de les dues fases de la detecció de col·lisions

(amb l’ús d’estructures de dades)

7.1.4 Resposta a la col·lisió

Aquest punt queda fora de l’abast d’aquest projecte però és convenient fer un breu comentari per entendre l’ús final de la informació que retorna un motor de col·lisions (actualment aquest projecte només retorna si es produeix o no col·lisió).

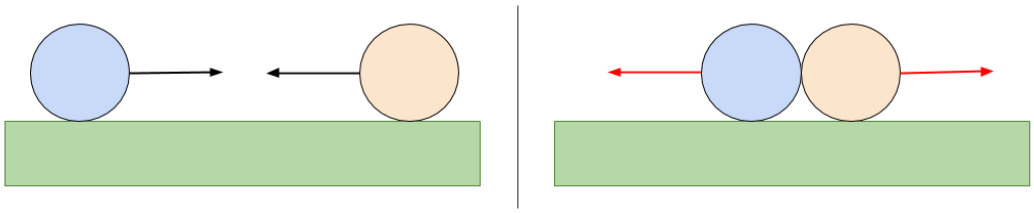


Figura 40: Exemple de la reacció a la col·lisió de dos objectes

Un cop detectada la col·lisió es disposa de la suficient informació per a que el motor de físiques actuí en conseqüència.

En un nivell molt bàsic, el motor de físiques calcula les noves posicions dels objectes que han col·lisionant per separar-los. Tot seguit, es traslladen els objectes a les noves posicions i es calcula la velocitat resultant del moviment aplicant totes les forces que poden afectar als objectes (gravetat, fricció).

7.2 Conceptes matemàtics bàsics

Per entendre el funcionament i la implementació d'alguns dels algoritmes i teoremes descrits en aquesta memòria és necessari fer una explicació de certs conceptes matemàtics bàsics emprats durant el projecte.

7.2.1. Sistema de coordenades cartesianes

Per poder representar els objectes cal definir un sistema de referència que permeti disposar d'una orientació en l'espai. Aquest és un **sistema cartesià**[17] i depenent del tipus d'espai que es faci servir, sistema cartesià d'una dimensió (1D), sistema cartesià de dos dimensions (2D) o sistema cartesià de tres dimensions (3D).

7.2.1.1 Sistema cartesià d'una dimensió (1D):

El sistema de coordenades cartesià d'una dimensió es tracta simplement d'una línia recta composta per un punt O u origen, una unitat de longitud n i una orientació (que consistiria en definir la recta dels números reals).

El punt O u origen permet dividir la línia en dos meitats, una negativa i una positiva. Per convenció s'assumeix que la meitat esquerra és la part negativa i la part de la dreta és la part positiva. D'aquesta manera es defineix que cada punt p es troba situat a una distància n del punt O, on n es qualsevol nombre real i el signe de p dependrà de en quina meitat de la recta es trobi situat.

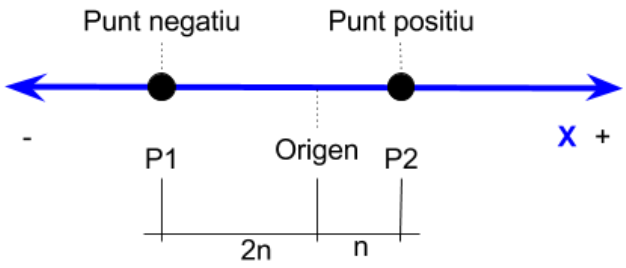


Figura 41: Sistema cartesià d'una dimensió

7.2.1.2 Sistema cartesià de dos dimensions (2D):

El sistema de coordenades cartesià de dos dimensions està compost per dues rectes reals com les descrites a l'apartat anterior, perpendiculars entre sí, que es tallen en un punt O u origen de manera que qualsevol punt P(x,y) en el pla pot ser definit per dues coordenades: una sobre l'eix horitzontal, denominat "X" o de les abscisses, i una sobre l'eix vertical, denominat "Y" o de les ordenades. Aquestes coordenades representen les distàncies ortogonals que existeixen des d'un punt als eixos cartesians.

Els eixos del sistema de coordenades cartesians divideixen el pla en quatre regions infinites que s'anomenen quadrants, cadascun compost per dos mitjos eixos. Per definició s'acostuma a definir-los del primer al quart, en números romans i en el sentit anti-horari, iniciant des del quadrant superior dret. Aquesta distinció permet identificar fàcilment els signes corresponents a les coordenades de les abscisses i les ordenades de cada quadrant.

D'aquesta manera el primer quadrant (cantonada superior dreta del pla) està compost per ordenades positives i abscisses positives, el segon (cantonada superior esquerra del pla) està compost per ordenades negatives i abscisses positives, el tercer (cantonada inferior esquerra del pla) està compost per ordenades negatives i abscisses negatives i per últim el quart (cantonada inferior dreta del pla) està compost per ordenades positives i abscisses negatives.

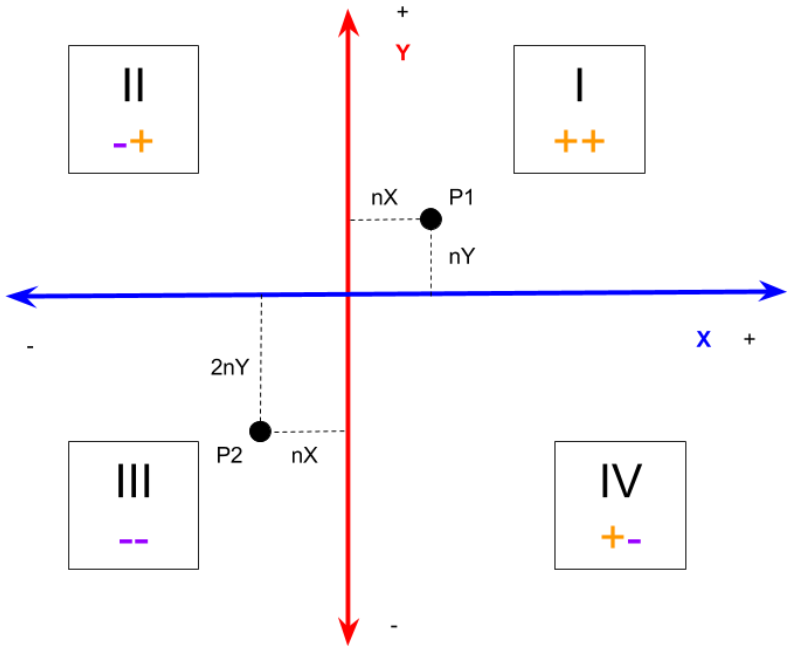


Figura 42: Sistema cartesià de dues dimensions

7.2.1.3 Sistema cartesià de tres dimensions (3D):

El sistema de coordenades cartesià de tres dimensions està compost per tres eixos perpendiculars entre si, amb una unitat de longitud comuna per a tots tres i una orientació en l'espai definida per a cadascun d'ells. El nou eix és denominat "Z". Així doncs qualsevol punt P(x,y,z) en l'espai pot ser definit per tres coordenades: una sobre l'eix X, una sobre l'eix Y i una sobre l'eix Z.

De la intersecció dels tres eixos, que es tallen igualment en un punt O u origen, es creen vuit espais definits pels signes de cadascun dels mitjos eixos que els componen.

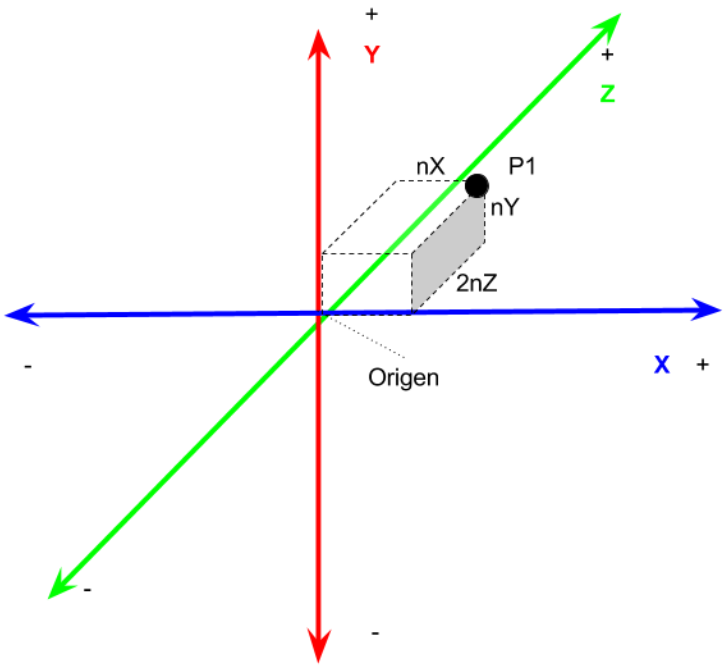


Figura 43: Sistema cartesià de tres dimensions

7.2.2.Vectors

Un **vector**[18] fixe en el pla euclidià és un segment orientat en el que es poden distingir tres característiques: un mòdul (la longitud del segment), una direcció (l'orientació de la recta) i un sentit (indicant quin és l'origen i quin l'extrem final de la recta).

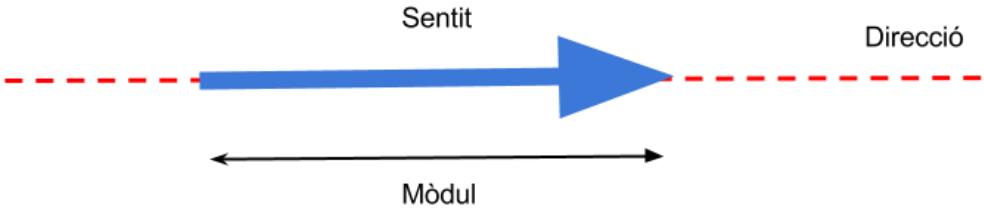


Figura 44: Components d'un vector

7.2.2.1 Propietats

7.2.2.1.1 Propietats de la suma i multiplicació d'escalars

$$\vec{a}+\vec{b}=\vec{b}+\vec{a} \qquad \vec{a}+(\vec{b}+\vec{c})=(\vec{a}+\vec{b})+\vec{c}$$

$$\vec{a}+\vec{0}=\vec{a} \qquad \vec{a}+(-\vec{a})=\vec{0}$$

$$c(\vec{a}+\vec{b}=c\vec{a}+c\vec{b} \qquad (c+d)\vec{a}=c\vec{a}+d\vec{a}$$

$$(cd)\vec{a}=c(d\vec{a}) \qquad 1\vec{a}=\vec{a}$$

7.2.2.1.2 Producte escalar

El **producte escalar (dot product)**[19] és una operació matemàtica que pren dos seqüències de números de la mateixa llargada i retorna un únic número (escalar). Algebraicament parlant consisteix en realitzar la suma entre els productes de les diferents entrades de les dues seqüències de números. Geomètricament parlant, el producte escalar que es fa ús en el projecte consisteix en obtenir l'angle de separació entre dos vectors. En cas que els dos vector estiguin normalitzats s'obté el cosinus de l'angle.

Així doncs, el producte escalar es defineix com:

$$\vec{a}=(a_1,a_2,a_3),\vec{b}=(b_1,b_2,b_3)\rightarrow\vec{a}\cdot\vec{b}=a_1b_1+a_2b_2+a_3b_3$$

i segueix les següents propietats:

$$\vec{a},\vec{b}\ són\ vectors\ i\ \vec{a}\cdot\vec{b}\ és\ un\ número$$

$$\vec{a}\cdot\vec{a}=|\vec{a}|^2$$

$$\vec{a}\cdot\vec{b}=\vec{b}\cdot\vec{a}$$

$$\vec{a}\cdot(\vec{b}+\vec{c})=\vec{a}\cdot\vec{b}+\vec{a}\cdot\vec{c}$$

$$(c\vec{a})\cdot\vec{b}=c(\vec{a}\cdot\vec{b})$$

$$\vec{0} \cdot \vec{a} = 0$$

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos\theta \text{ [2]}$$

$$\vec{a} \cdot \vec{b} = 0 \rightarrow \vec{a} = \vec{0}\ o\ \vec{b} = \vec{0}\ o\ \vec{a} \perp \vec{b}$$

7.2.2.1.3 Producte vectorial

El **producte vectorial (cross product)**[20] és una operació binaria entre dos vectors dins de l'espai de 3 dimensions. Donats dos vectors a i b (linealment independents), el producte vectorial retorna un nou vector perpendicular als dos i normal al pla que els conté (un vector ortogonal). Si dos vectors tenen la mateixa direcció o si algun dels dos te llargada zero, llavors el producte vectorial donarà zero.

Així doncs el producte vectorial es defineix com:

$$\vec{a} = (a_1, a_2, a_3), \vec{b} = (b_1, b_2, b_3) \rightarrow \vec{a}x\vec{b} = (a_2b_3 - a_3b_2, a_3b_1 - a_1b_3, a_1b_2 - a_2b_1$$

i segueix les següents propietats:

$$\vec{a}, \vec{b}\ i\ \vec{a}x\vec{b}\ són\ vectors\ de\ tres\ dimensions$$

$$|\vec{a}x\vec{b}| \perp |\vec{a}||\vec{b}| \sin\theta$$

$$\hat{\imath}x\hat{\jmath} = \hat{k}, \hat{\jmath}x\hat{k} = \hat{\imath}, \hat{k}x\hat{\imath} = \hat{\jmath}$$

$$\vec{a}x\vec{b} = |\vec{a}||\vec{b}| \sin\theta\ \hat{n}$$

$$\vec{a}x\vec{b} = 0 \Leftrightarrow \vec{a} = \vec{0}\ o\ \vec{b} = \vec{0}\ o\ \vec{a} \parallel \vec{b}$$

$$\vec{a}x\vec{b} = -\vec{b}x\vec{a}$$

$$(c\vec{a})x\vec{b} = \vec{a}x(c\vec{b}) = c(\vec{a}x\vec{b})$$

$$\vec{a}x(\vec{b}x\vec{c}) = \vec{a}x\vec{b} + \vec{a}x\vec{c}$$

$$\vec{a} \cdot (\vec{b}x\vec{c}) = (\vec{a}x\vec{b}) \cdot \vec{c}$$

$$\vec{a}x(\vec{b}x\vec{c}) = (\vec{c} \cdot \vec{a})\vec{b} - (\vec{b} \cdot \vec{a})\vec{c}$$

[2] θ és l'angle entre els vectors $\vec{a}\ i\ \vec{b}$
$|\hat{n}| = 1,\ \hat{n}\ \perp \vec{a}, \hat{n} \perp \vec{b}\ i\ \hat{n}, \vec{a}\ i\ \vec{b}$ segueixen la regla de la mà dreta.

7.2.2.2 Tipologies

7.2.2.2.1 Vector unitari

Un **vector unitari**[21] és un vector de mòdul 1.

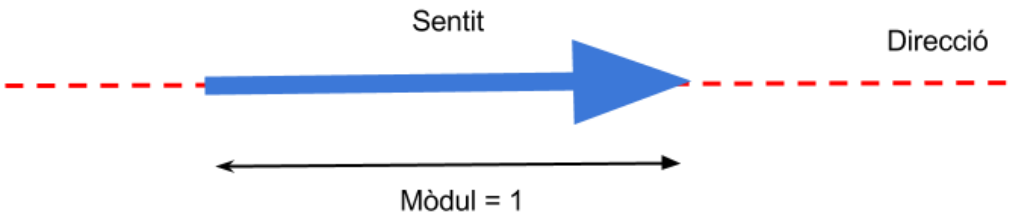


Figura 45: Components d'un vector unitari

7.2.2.2.2 Vector normal (perpendicular)

Un **vector normal** a un objecte te la propietat de ser ortogonal (perpendicular) a tots els vectors tangents a l'entitat geomètrica.

7.2.3 Matrius

Una **matriu**[22] és una estructura de dades rectangular organitzada en files i columnes.

7.2.3.1 Propietats

7.2.3.1.1 Propietats de la suma

Sent A,B i C matrius de m x n:

1. $A + B = B + A$
2. $(A + B) + C = A + (B + C)$
3. $A + 0 = A$ on 0 és la matriu-zero m x n (on tots els valors són 0).
4. $A + B = 0$ si i només si $B = -A$.

7.2.3.1.2 Propietats de la multiplicació

1. Sent A,B i C tres matrius. Si es pot realitzar els productes *AB*, (*AB*)*C*, *BC*, i *A*(*BC*), llavors (*AB*)*C* = *A* (*BC*)
2. Si α i β són nombres, i A és una matriu, llavors $\alpha(\beta A) = (\alpha\beta)A$
3. Si α és a nombre, i A i B són dues matrius el producte de les quals A x B és possible, llavors $\alpha(AB) = (\alpha A)B = A(\alpha B)$
4. Si A és una matriu n x m i 0 és la matriu-zero, llavors $A0 = 0$

7.2.3.1.3 Propietats de la suma i la multiplicació

1. Sent A,B i C tres matrius. Si es poden realitzar els seus productes, llavors $(A + B)C = AC + BC$ *i* $A(B + C) = AB + AC$
2. Si α i β són nombres, i A i B són una matriu, llavors $\alpha(A + B) = \alpha A + \alpha B$ i $(\alpha + \beta)A = \alpha A + \beta A$

7.2.3.1 Tipologies

7.2.3.1.1 Matriu identitat

La **matriu identitat**, que es denota amb el símbol I_n, és una matriu quadrada de mida n en la que tots els elements de la diagonal principal són iguals a 1 i la resta d'elements són iguals a 0.

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

S'anomena matriu identitat ja que realitzar el producte d'una matriu A x I_n = A.

7.2.3.1.2 Matriu inversa

Donada una matriu A quadrada de dimensió n, es diu que A és invertible si existeix una altra matriu B tal que el producte de matrius A x B = B x A = I_n. La matriu B, si existeix, és única i s'anomena la **matriu inversa**[23] d'A, i es denota com A^{-1}.

$$A \times A^{-1} = A^{-1} \times A = I_n$$

Només poden tenir inversa les matrius quadrades i que tinguin un determinant diferent de 0.

7.2.4 Quaternions

Els **quaternions**[24] són una generalització dels nombres complexos. De tal manera que si un nombre complex defineix dues dimensions afegint la component **i** (sent $i^2 = -1$), un quaternió defineix quatre dimensions afegint les components **i**,**j**,**k**, de manera que:

$$i^2 = j^2 = k^2 = ijk = -1$$

Un quaternió, doncs, és un nombre de la forma:

$$z = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$$

On els 4 nombres reals a, b, c i d defineixen únicament el quaternió z.

El valor absolut del quaternió z es defineix com a:

$$|z| = \sqrt{a^2 + b^2 + c^2 + d^2}$$

Els quaternions permeten una representació diferent de rotacions i orientacions en tres dimensions. Aquests s'apliquen per exemple en gràfics per ordinador.

7.2.4.1 Propietats

Addicció: $\forall q, p, r \in Q$

- pertanyença: $p + q \in Q$
- commutativitat: $p + q = q + p$
- associativitat: $(p + q) + r = p + (q + r)$
- identitat: existeix $0 = 0 + 0i + 0j + 0k \in Q$ tal que $0 + p = p + 0 = p$
- inversa: existeix $(-q) \in Q$ tal que $q + (-q) = (-q) + q = 0$
- suma: $p + q = (w_p + w_q) + (x_p + x_q)i + (y_p + y_q)j + (z_p + z_q)k$
- diferència: $p - q = (w_p - w_q) + (x_p - x_q)i + (y_p - y_q)j + (z_p - z_q)k$

Multiplicació: per $\forall q, p, r \in Q$ i $\forall n \in R$

- pertinença: $pq \in Q$
- no-commutativitat: $pq \neq qp$
- associativitat: $(pq)r = p(qr)$
- distributivitat: $p(q + r) = pq + pr$ i $(q + r)p = qp + rp$
- identitat: existeix $1 = 1i + 1j + 1k \in Q$ tal que $1p = p1 = p$
- inversa: si $q \neq 0$, llavors existeix $q^{-1} \in Q$ tal que $qq^{-1} = q^{-1}q = 1$
- producte:$pq = \lfloor w_p w_q - v_p \cdot v_q, (w_p v_q + w_q v_p + v_p x v_q \rfloor$ on $\cdot$ indica producte escalar i x indica producte vectorial
- no divisors per zero: si $pq = 0$, llavors també $p = 0$ o $q = 0$
- divisió: $\frac{p}{q} = pq^{-1}$, si $p = qr$ llavors $q = pr^{-1}$ i $r = q^{-1}p$
- escalat: $qn = nq = \lfloor nw_q, (nx_q, ny_q, nz_q) \rfloor$

Per a aquest projecte les propietats més importants a tenir en compte són:

- Un quaternió normalitzat només dona la rotació
- El mòdul d'un quaternió dona l'escalat.

7.2.4.2 Motius per fer ús de quaternions en comptes de matrius de transformacions

7.2.4.2.1 Eliminar l'Euler lock

El sistema de rotacions que es fa servir al framework i al motor es un sistema que utilitza la suspensió de Cardan[25], un mecanisme de suspensió consistent en dos cercles concèntrics els eixos dels quals formen un angle recte, que permet mantenir l'orientació d'un eix de rotació a l'espai encara que el seu suport es mogui.

Es fa servir per muntar una escena d'objectes de manera que puguin orientar els seus eixos de rotació en qualsevol direcció de l'espai.

El **Euler lock**, també conegut com a Gimbal lock[26], es la pèrdua d'un dels grau de llibertat en un mecanisme de tres dimensions com l'explicat anteriorment. Es produeix quan els eixos de dos dels tres cercles són posicionats en paral·lel, d'aquesta manera es produeix un "bloqueig" d'un dels eixos i el mecanisme queda amb dos dimensions de rotació.

La part del bloqueig és només una qüestió visual, realment no es bloqueja cap eix de rotació, únicament hi ha dos eixos que roten de la mateixa manera i produeixen l'efecte que s'ha perdut un grau de llibertat.

Amb l'ús de quaternions s'elimina aquest bloqueig i en tot moment es disposa de rotació en els tres eixos.

7.2.4.2.2 Computació més eficient

Donat un vector, aquest es pot rotar aplicant un producte amb una matriu de rotació o mitjançant un quaternió.

A l'hora de realitzar la computació, l'ús de quaternions permet un còmput més baix que no pas realitzar productes amb matrius. El fonament recau en que la combinació de molts quaternions és més estable numèricament que la combinació de moltes matrius de transformació.

Un quaternió pot assolir els mateixos resultats de transformacions que aplicant matrius de transformacions.

Si un vector v es multiplica per un quaternió q:

$$v \times q = v \text{ rotat segons el quaternió}$$

L'avantatge que tenen els quaternions envers les matrius de transformacions es que permeten encadenar productes i realitzar el càlcul molt més eficientment.

Així doncs:

$$v \times q \times q = v \text{ rotat dos cops segons el quaternions}$$

Si es realitzés un producte d'un quaternió que aplica una rotació de 45º, en realitzar el doble producte, el vector final haurà rotat 90º, donat que:

$$q_{45^o} \times q_{45^o} = q_{90^o}$$

De la mateixa manera que existeix la inversa d'una matriu també existeix la **inversa d'un quaternió** q^{-1}. La qual cosa permet realitzar les transformacions dels punts entre els espais de món i els espais locals (a l'apartat "*7.2.6-Espai de món i espai local*" hi ha una explicació més detallada).

Així doncs si el producte de q per v rota el vector i el transforma a espai de món:

$$q \times v = v \text{ en espai de món } (v_M)$$

llavors si q^{-1} és la inversa del quaternió q, llavors en fer el producte entre la inversa del quaternió i els vèrtex transformats (V_M) es desfà la transformació i es torna a tenir v en espai local:

$$q^{-1} \times v_M = \text{transforma } v_M \text{ en v } (v_L)$$

7.2.5. Representació d'objectes

Un objecte es pot representar per un conjunt de punts, anomenats vèrtex i una matriu de transformació.

7.2.5.1 Els vèrtex

Un **vèrtex**[27] és la representació d'un punt en l'espai. En geometria, és un punt comú entre dos costats consecutius d'una figura geomètrica.

Els objectes emprats en aquest projecte són considerats quadrats (2D) i cubs (3D).

Per un espai 2D, un quadrat es pot definir mitjançant 4 vèrtex, on cadascun dels vèrtex està compost per dues coordenades (x,y).

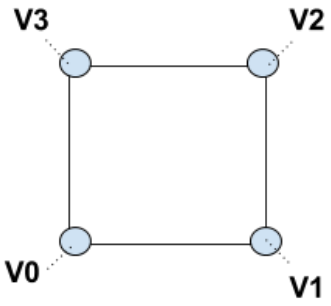


Figura 46: Els quatre vèrtex d'un quadrat

Per un espai 3D, un cub es pot definir mitjançant 8 vèrtex, on cadascun està compost per tres coordenades (x,y,z).

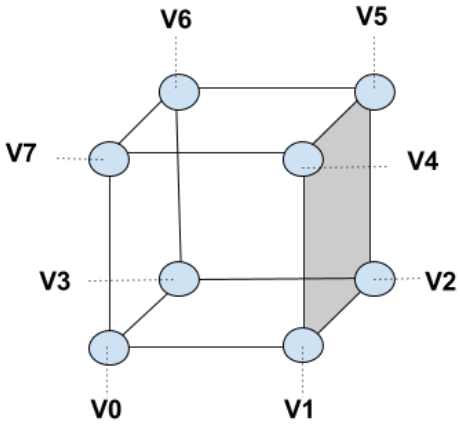


Figura 47: Els vuit vèrtex d'un cub

7.2.5.2 La matriu de transformació

Una **matriu de transformació**[28] és una matriu que conté la informació de translació, rotació i escalat d'un element.

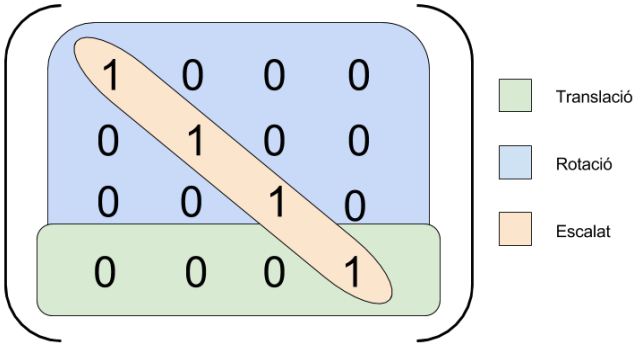
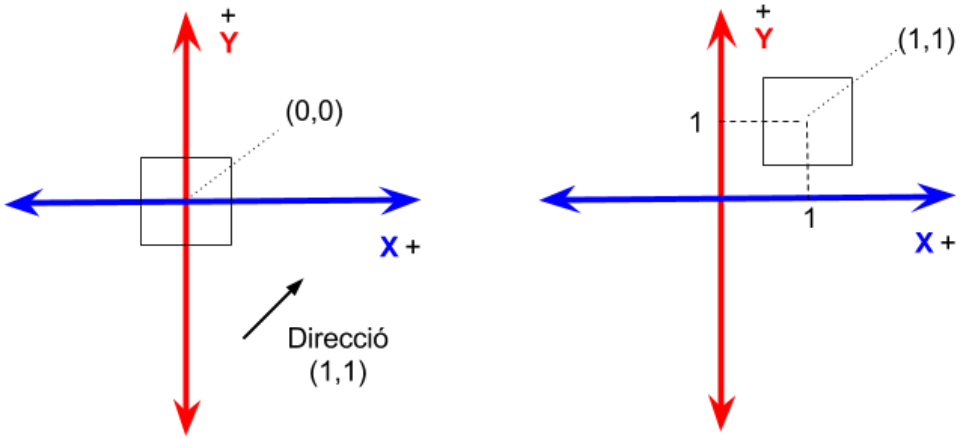


Figura 48: Repartició de la matriu de transformació

7.2.5.2.1 Translació:

Una **translació**[29] és una transformació geomètrica que mou tots els punts d'un objecte paral·lelament a una direcció donada, en una magnitud constant.



IM7: Exemple d'una translació en 2D

La **matriu de translació** és la matriu que representa una translació en l'espai euclidià.

En 2D la translació d'un punt es realitzaria mitjançant la matriu de translació de la forma següent:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix} x \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

D'aquesta forma, multiplicant el vector (x,y), que representa el punt original, per la matriu de translació dona com a resultat el punt traslladat (x',y').

En 3D la translació d'un punt es realitzaria mitjançant la matriu de translació de la forma següent:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} x \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

D'aquesta forma, multiplicant el vector (x,y,z), que representa el punt original, per la matriu de translació dona com a resultat el punt traslladat (x',y',z').

7.2.5.2.2 Rotació:

Una **rotació**[30] és una transformació en el pla o en l'espai que descriu el moviment d'un objecte al voltant d'un eix. És pot definir per tres angles d'Euler o per un angle de rotació més la direcció de l'eix de rotació.

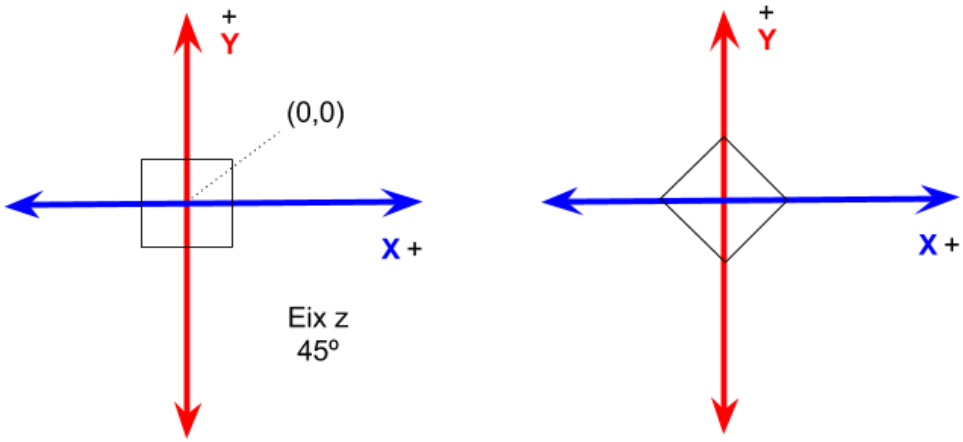


Figura 49: Exemple d'una rotació en 2D

Les rotacions al voltant de l'origen es poden calcular fàcilment emprant transformacions matricials d'una matriu anomenada **matriu de rotació,** la matriu que representa una rotació en l'espai euclidià.

En 2D la rotació d'un punt es pot realitzar respecte a l'eix X i respecte a l'eix Y. D'aquesta forma al multiplicar un punt (x,y), que representa el punt original, per la matriu de rotació dona com a resultat el punt rotat (x',y').

La rotació respecte l'eix X és realitzaria de la forma següent:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} x \begin{pmatrix} x \\ y \end{pmatrix}$$

En 3D la rotació d'un punt es pot realitzar respecte a l'eix X, respecte a l'eix Y i respecte a l'eix Z. D'aquesta forma al multiplicar un punt (x,y,z), que representa el punt original, per la matriu de rotació dona com a resultat el punt rotat (x',y',z').

Les següents matrius exemplifiquen les tres rotacions bàsiques sobre els eixos *x*,*y*, i *z* respectivament:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix} x \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$ Rotació sobre l'eix x

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{pmatrix} x \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$ Rotació sobre l'eix y

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} x \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$ Rotació sobre l'eix z

7.2.5.2.3 Escalat:

L'**escalat**[31] és una transformació lineal que incrementa o disminueix un objecte a partir d'un factor d'escala que pot ser igual a totes les direccions o específic per cada eix. Si el factor d'escala és més gran que 1, l'objecte és dilata. Si el factor d'escala és un número positiu menor a 1 l'objecte es contrau.

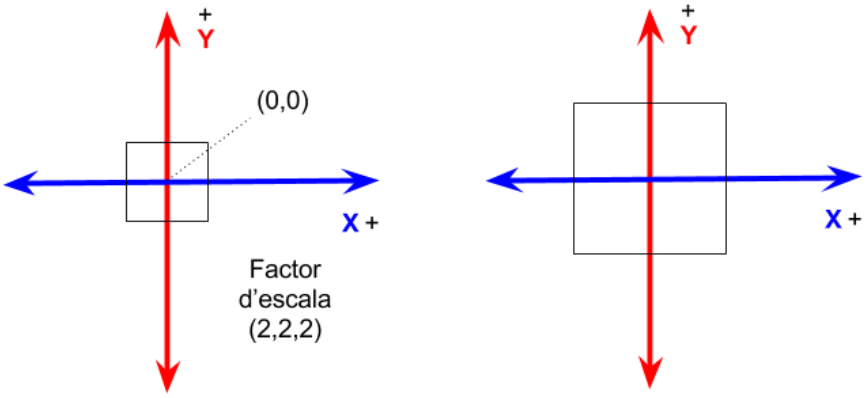


Figura 50: Explemple d'un escalat en 2D

L'escalat és pot representar mitjançant una **matriu d'escalat**, una matriu que representa l'escalat en l'espai euclidià.

En 2D l'escalat d'un punt es realitzaria mitjançant la matriu d'escalat de la forma següent:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} e_x & 0 \\ 0 & e_y \end{pmatrix} x \begin{pmatrix} x \\ y \end{pmatrix}$$

D'aquesta forma, multiplicant el vector (x,y), que representa el punt original, per la matriu de translació dona com a resultat el punt traslladat (x',y').

En 3D la translació d'un punt es realitzaria mitjançant la matriu de translació de la forma següent:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} e_x & 0 & 0 \\ 0 & e_y & 0 \\ 0 & 0 & e_z \end{pmatrix} x \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

D'aquesta forma, multiplicant el vector (x,y,z), que representa el punt original, per la matriu d'escalat dona com a resultat el punt escalat (x',y',z').

7.2.5.3 Transformació dels vectors

Mitjançant la combinació dels productes de les diferents matrius es calcula la transformació dels punts (vectors).

vectorTransformat = MTransformació x vectorOriginal

on MTransformació = MTranslació x MRotació x MEscalat

És important seguir l'ordre especificat per dur a terme els diferents productes de matrius, ja que el resultat final varia segons quin ordre es segueixi.

A continuació hi ha un exemple de com pot arribar a variar la transformació dels vectors si s'apliquen ordres diferents.

Ambdós casos parteixen d'un objecte centrat a l'origen de coordenades i alineat amb els eixos.

Cas 1: Correcte

És segueix l'ordre: escalar, rotar.

S'aplica un escalat en l'eix X de 2 unitats i una rotació en l'eix Z de 45º.

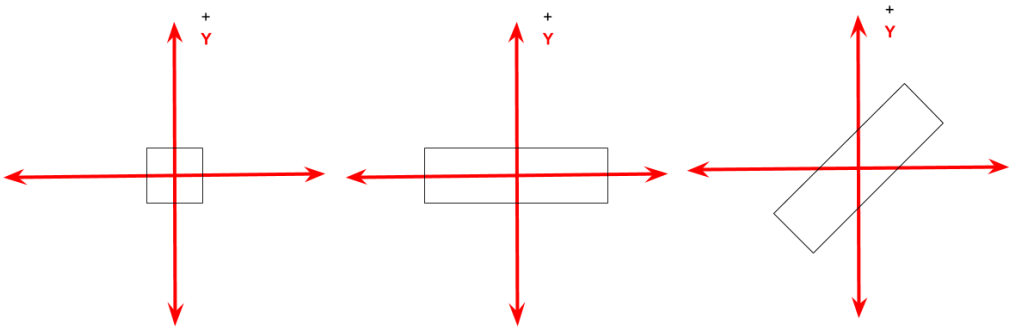


Figura 51: Exemple de transformació en correcte

Cas 2: Incorrecte

És segueix l'ordre: rotar, escalar.

S'aplica una rotació en l'eix Z de 45º i un escalat en l'eix X de 2 unitats.

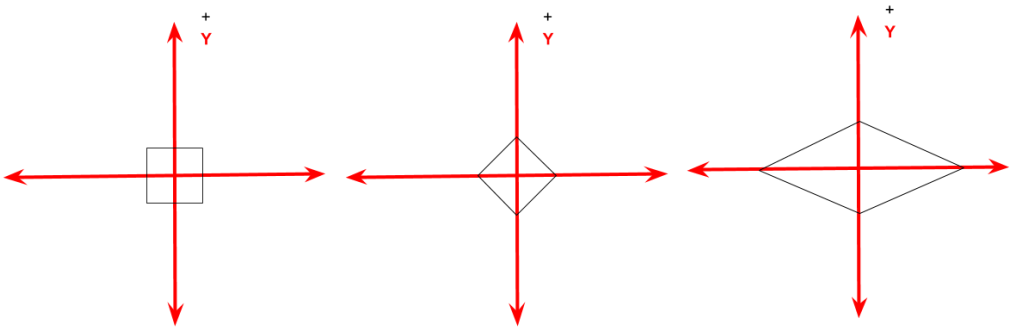


Figura 52: Exemple de transformació en incorrecte

Resultats:

La transformació final en el cas 1 i en el cas 2 són completament diferents.

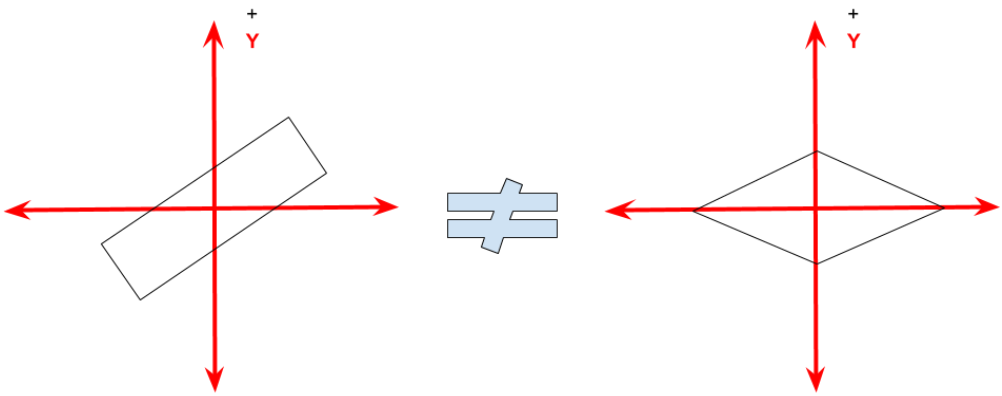


Figura 53: Comparativa dels resultats de les transformacions

7.2.6 Espai de món i espai local

Un objecte es defineix a partir d'un conjunt de vèrtex. Les coordenades d'aquests vèrtex estan definides relatives al centre de l'objecte, que si fos un vèrtex es representaria amb la coordenada (0,0) o (0,0,0). Aquests vèrtex és troben en **espai local (del model)**.

Per moure un objecte es pot fer ús de la matriu de transformació o dels quaternions definits anteriorment. L'explicació següent està feta només en el cas de les matrius per no duplicar explicacions però es valida per a l'aplicació dels quaternions.

Aplicant aquesta matriu a tots el vèrtex permetrà realitzar la transformació de l'objecte. Al fer-ho tots els vèrtex és troben en **espai de Món**.

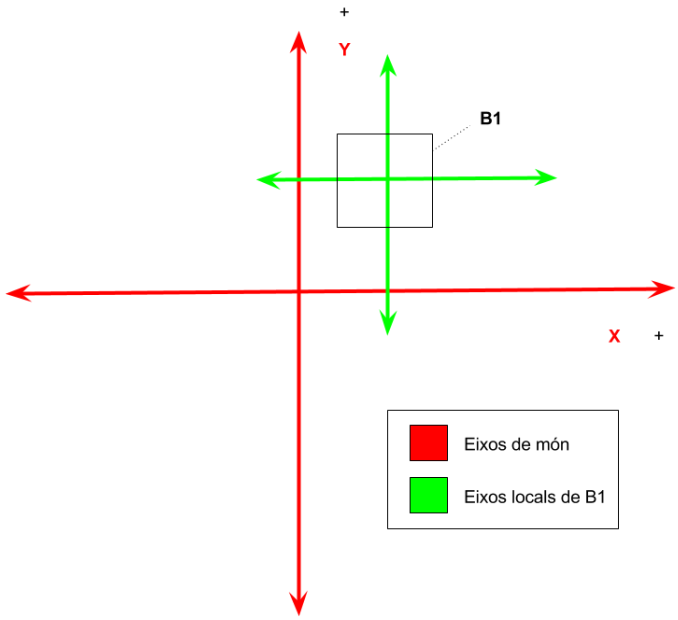
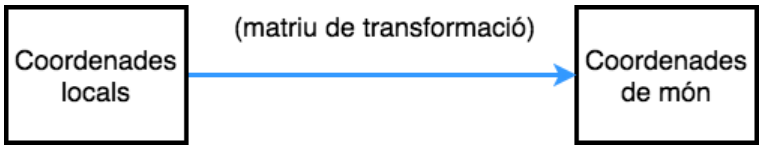


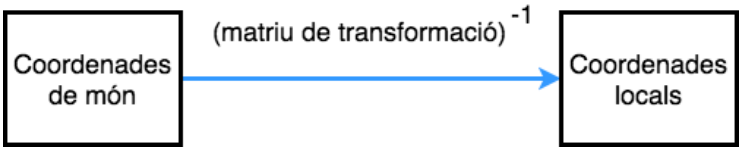
Figura 54: Exemple d'un objecte situat en espai de món respecte als seus eixos locals.

És possible passar dels vèrtex del model (locals) als vèrtex respecte World (món) multiplicant per la matriu de transformació.



Coordenades de món = MTransformació x CoordedadesLocals

I de la mateixa manera és possible passar dels vèrtex de món als vèrtex locals multiplicant per la matriu de transformació inversa.



Coordenades locals = $(MTransformació)^{-1}$ x CoordenadesMón

7.2.7 Convexitat

Una objecte és considera **convex** si, per a qualsevol línia que es traci a traves de l'objecte, aquesta només creua dues vegades. En cas que es pugui dibuixar una línia a través de l'objecte i aquesta creui més de dues vegades vol dir que aquest no és convex.

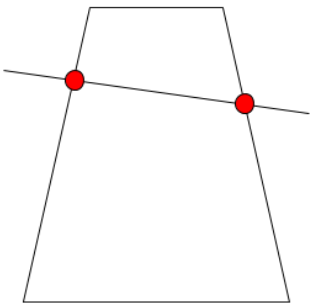


Figura 55: Objecte convex

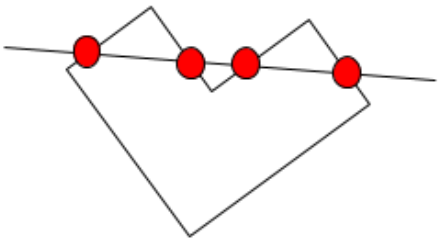


Figura 56: Objecte no convex

El primer objecte *(Figura 54)* esta considerat convex, ja que no existeix cap línia que el creui més de dos cops. El segon objecte *(Figura 55)* no és convex, ja que existeix una línia que el creua més de dues vegades.

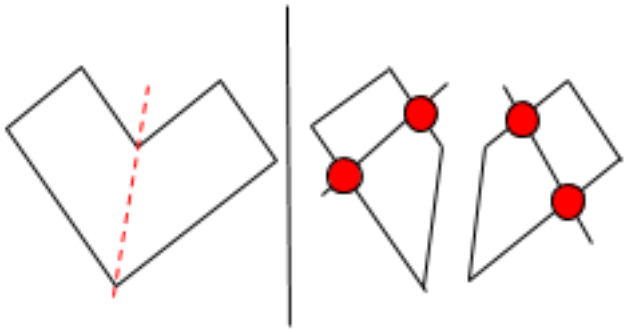


Figura 57: Descomposició en objectes convexos

Qualsevol objecte no convex es pot representar com la combinació de varis objectes convexos. A la *Figura 57* és pot veure com agafant l'objecte no convex de la *Figura 56* i aplicant-li una descomposició es pot dividir en 2 objectes convexos.

7.2.8 Projecció

La **projecció** consisteix en el procediment per a obtenir la representació en el pla (2 dimensions) dels objectes ubicats en l'espai (3 dimensions). El principi fonamental de la projecció és la correspondència entre els punts de l'objecte amb els de la imatge projectada.

Per exemple, en col·locar un objecte de tres dimensions a certa distancia sobre un pla (el terra) i en un punt per sobre de l'objecte es col·loca un focus de llum que l'il·lumini de manera que tots els rajos del focus de llum siguin paral·lels i perpendiculars a la superfície. Llavors sobre el terra es crearà la projecció en 2 dimensions de l'objecte.

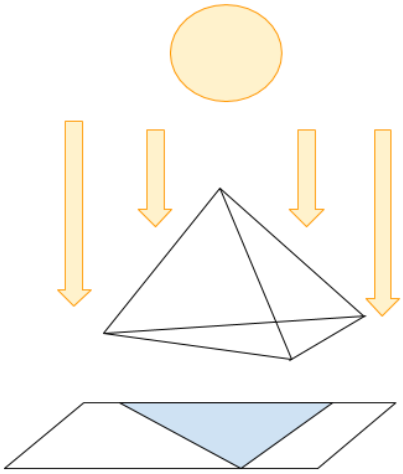


Figura 58: Exemplificació de porjecció

En aquest projecte es projecten els costats d'un objecte sobre cadascun dels eixos de coordenades.

7.3 Separating Axis Theorem

Separating Axis Theorem[32] (SAT), és un mètode per determinar si dos objectes convexos intersecconen. SAT és un algoritme genèric i ràpid que pot eliminar la necessitat de tenir codi de detecció de col·lisió per a cada tipus de parelles de formes, reduint d'aquesta manera el codi i el manteniment. L'algoritme complet de SAT no només es centra en la detecció de la col·lisió sinó que també retorna la informació de separació entre els objectes (si aquests no intersecconen), és a dir, la distància mínima de col·lisió.

7.3.1 L'algoritme en 2D

SAT afirma que: **"Si dos objectes convexos no estan col·lisionant, existeix un eix pel qual no se superposaran cap de les projeccions dels objectes."**

7.3.1.1 Detectar la no intersecció

En aquest apartat s'explica com SAT determina si dos objectes no intersecconen.

A la *Figura 59* es pot observar que els dos objectes no intersecconen. Una línia s'ha dibuixat entre ells per il·lustrar-ho.

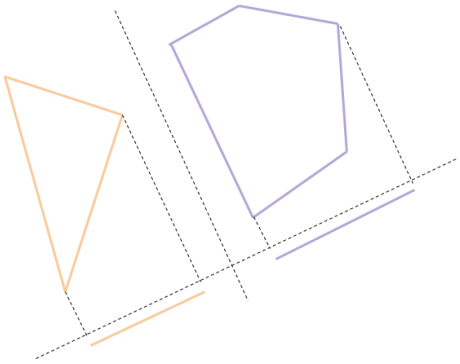


Figura 59: Dos objectes convexos separats amb les seves respectives projeccions

Si s'obté la perpendicular a la línia que separa els dos objectes (línia discontinua de la *Figura 59*) i es fa la seva projecció contra aquesta línia es pot observar que no hi ha sobreposicionen entre les projeccions. Aquella línia on les projeccions dels objectes no

es sobre posicionen s'anomena eix de separació (en anglès *"separation axis"*, SA). A la figura superior la línia discontínua és una SA i les respectives línies de colors són les projeccions dels objectes. Així doncs, aplicant l'afirmació de SAT, donat que les projeccions no se sobreposen llavors els objectes no intersecciones.

SAT necessita comprovar un gran nombre d'eixos per assegurar que hi ha intersecció (als apartats *"7.3.1.6-Quantitat d'eixos a comprovar en 2D"* i "7.3.2.1-*Quantitat d'eixos a comprovar en 3D"* es detalla la quantitat d'eixos necessaris a comprovar tenint en compte que els objectes amb els quals s'ha treballat són quadrats o cubs).

7.3.1.2 Com detectar la intersecció

En aquest apartat s'explica com SAT determina si les formes intersecciones. Si per tots els eixos, les projeccions es sobreposen, llavors es pot concloure que tots els objectes intersecciones.

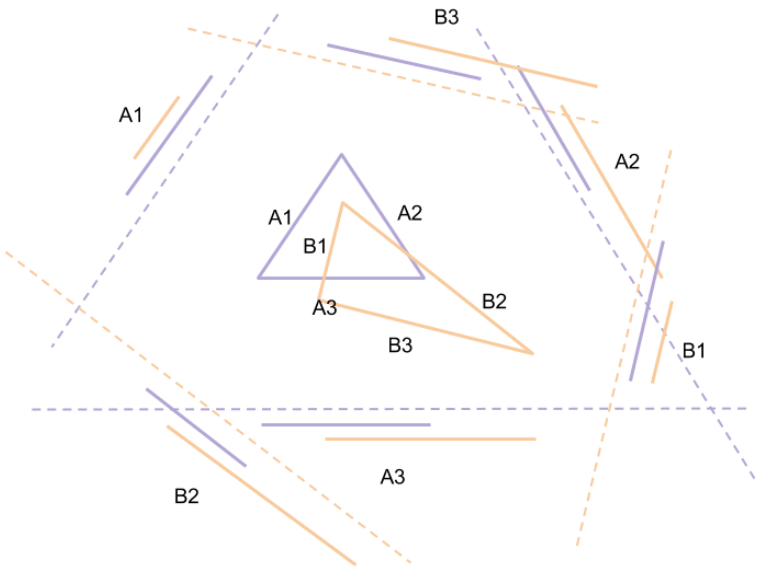


Figura 60: Dos objectes convexos interseccionant

A la *Figura 60* hi ha il·lustrats dos objectes convexos on s'han comprovat tots els seus eixos. Les projeccions sobre cadascun dels eixos es sobreposen, així doncs es pot assegurar que els objectes intersecciones.

7.3.1.3 L'obtenció dels eixos a comprovar

En aquest apartat s'explica quins son els eixos que l'algoritme necessita comprovar.

Els eixos que cal comprovar són les normals a cadascuna de les arestes dels objectes.

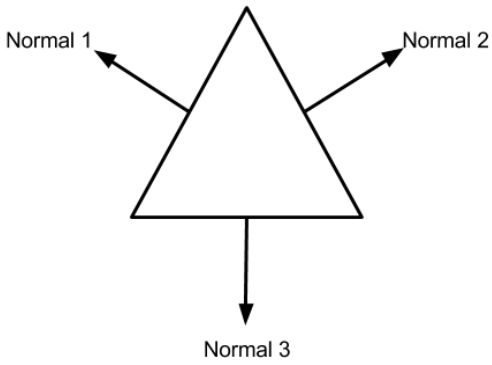


Figura 61: Vectors normals a les arestes

Els diferents vectors normals a cada aresta es poden obtenir intercanviant les coordenades (x,y) i negant una, de la manera que les normals serien (-y,x) o (y,-x). En cas que es necessiti només el resultat booleà de l'algoritme SAT (si hi ha intersecció o no) fer el càlcul del vector normal d'aquesta manera seria suficient. Però en cas que es necessiti la informació de la col·lisió (que s'explica a l'apartat de "*7.3.1.5-Calcular el Mínimum Translation Vector*") llavors aquests vectors han de ser normalitzats.

El nombre d'eixos a comprovar dependrà del tipus d'objectes amb els que es treballi. Per exemple a la *Figura 60* hi ha dos objectes triangulars, per la qual cosa caldrà comprovar 6 eixos (les normals a les arestes a1, a2, a3, b1, b2 i b3).

7.3.1.4 Projectar un objecte sobre un eix

Per projectar un polígon sobre un eix es necessari de recórrer tots els vèrtex de la forma i calcular un producte escalar amb l'eix i emmagatzemar els valors màxims i mínims.

7.3.1.5 Calcular el Mínimum Translation Vector

Fins a aquest punt l'algoritme únicament retorna si es produeix intersecció o no. A més a més, SAT pot retornar el **Mínimum Translation Vector** (MTV, el vector de translació

mínim). El MTV és el vector de magnitud mínima utilitzat per moure els objectes un cop es produeix una col·lisió. A la *Figura 60* es pot observar que l'eix b3 té la coincidència més petita. Aquest eix i la coincidència és la MTV, l'eix és converteix en el vector i la coincidència és la magnitud.

Per determinar si els objectes intersecciones s'han de recórrer tots els eixos, i al mateix temps, es pot anar fent el seguiment de la distància mínima i l'eix. En cas que és detecti intersecció es pugui retornar la MTV més petita.

7.3.1.6 Quantitat d'eixos a comprovar en 2D

Per al nostre projecte, en un espai 2D, es treballa mitjançant objectes quadrats, on el nombre d'eixos a comprovar dependrà del tipus d'orientació que tinguin els quadrats entre si.

7.3.1.6.1 Cas 1: Objecte 1 i objecte 2 tenen els eixos alineats

Ambdós objectes disposen dels seus eixos locals alineats entre ells i als eixos de món. Per aquest motiu, donat que s'està en un espai 2D caldrà **comprovar 2 eixos**: les projeccions sobre l'eix X i sobre l'eix Y de món.

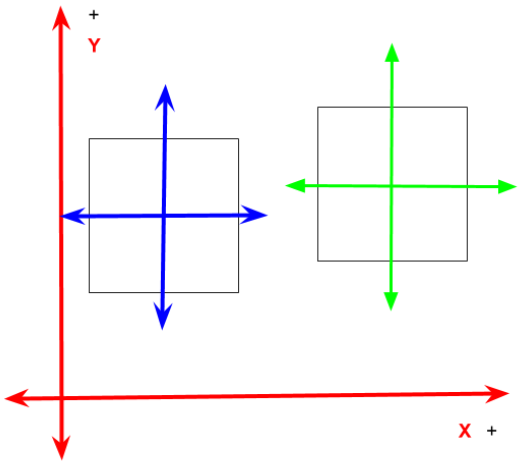


Figura 62: Representació de dos objectes AABB en 2D

7.3.1.6.2 Cas 2: Objecte 1 i objecte 2 no tenen els eixos alineats

Ambdós objectes tenen una alineació dels eixos diferents entre si i als eixos de món. En el pitjor dels casos els eixos del primer objecte tindran una alineació diferent a l'alineació del segon objecte. Per aquest motiu serà necessari **comprovar 4 eixos**: Els dos eixos del primer objecte i els dos eixos del segon objecte.

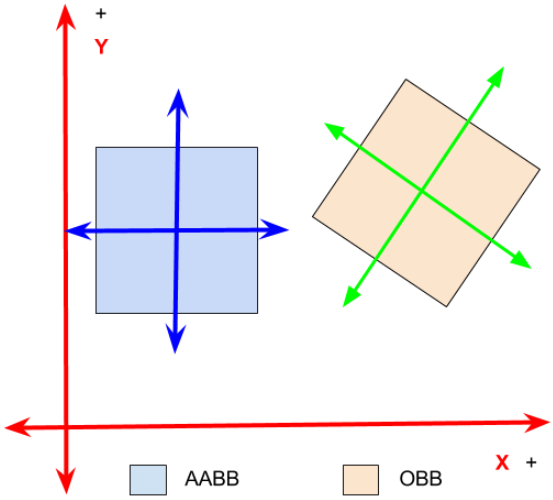


Figura 63: Representació d'un objecte AABB i un OBB en 2D

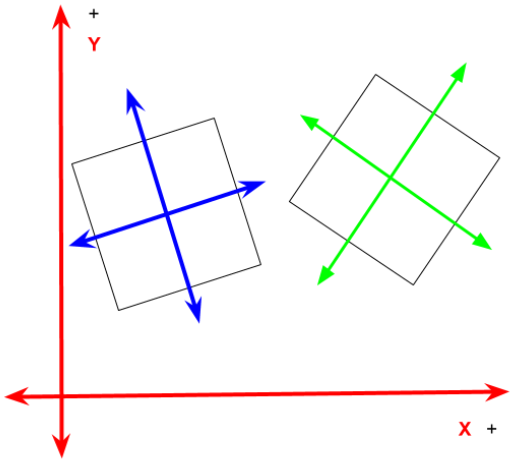


Figura 64: Representació de dos objectes OBB en 2D

7.3.2 L'algoritme en 3D

En un espai 3D es fa servir el mateix mètode, però en comptes de comprovar les normals a les arestes, serà necessari comprovar les normals a les cares dels objectes.

7.3.2.1 Quantitat d'eixos a comprovar en 3D

Per al nostre projecte, en un espai 3D, es treballa mitjançant objectes cúbics, on el nombre d'eixos a comprovar també dependrà del tipus d'orientació que tinguin cadascuna de les mínimum Bounding Box.

7.3.2.1.1 Cas 1: Objecte 1 i objecte 2 tenen els eixos alineats

Ambdós objectes disposen dels seus eixos locals alineats entre ells i als eixos de món. Per aquest motiu, donat que s'està en un espai 3D caldrà **comprovar 3 eixos**: les projeccions sobre l'eix X, sobre l'eix Y i sobre l'eix Z de món.

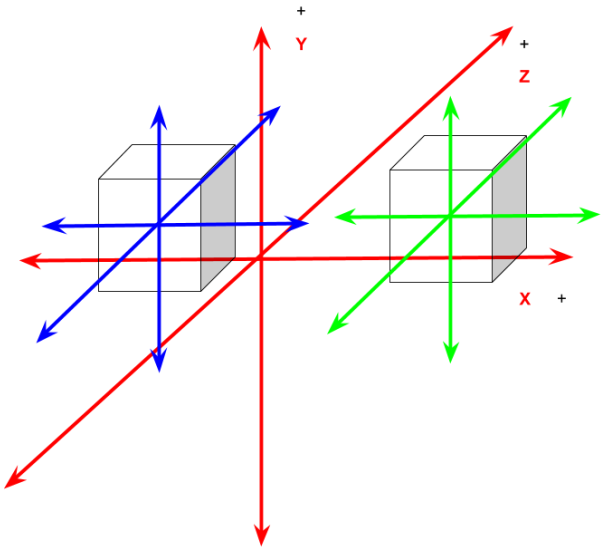


Figura 65: Representació de dos objectes AABB en 3D

7.3.2.1.2 Cas 2: Objecte 1 alineat i objecte 2 no alineat

Un primer objecte disposa d'una alineació dels seus eixos locals igual a la dels eixos de món, però l'altre objecte té una alineació diferent. Per aquest motiu, donat que es treballa en un espai 3D cal comprovar 6 eixos: les projeccions sobre els tres eixos del primer objecte i sobre els tres eixos del segon objecte.

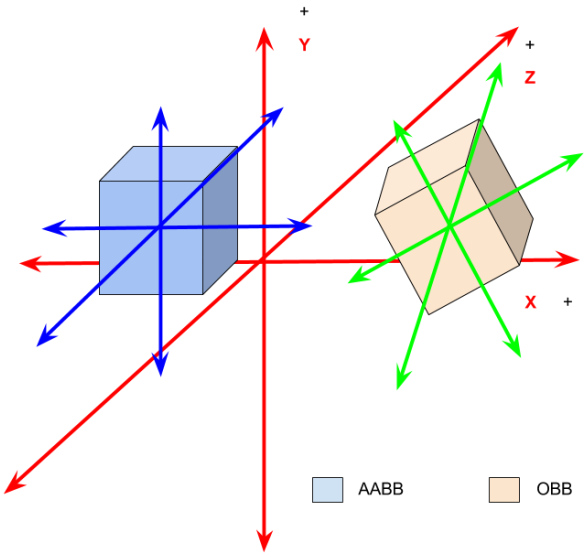


Figura 66: Representació d'un objecte AABB i un OBB en 3D

7.3.2.1.3 Cas 3: Objecte 1 i objecte 2 no tenen els eixos alineats

Ambdós objectes disposen d'una alineació dels eixos diferents entre si i als eixos de món. En el pitjor dels casos els eixos del primer objecte tindran una alineació diferent de l'alineació del segon objecte. Per aquest motiu serà necessari **comprovar 15 eixos**:

- 3 eixos a les normals de les cares del primer objecte: Ax, Ay, Az
- 3 eixos a les normals de les cares del segon objecte: Bx, By, Bz
- 9 eixos resultants de realitzar productes vectorials entre les normals dels dos objectes
    - Ax vs Bx
    - Ax vs By
    - Ax vs Bz
    - Ay vs Bx
    - Ay vs By
    - Ay vs Bz
    - Az vs Bx
    - Az vs By
    - Az vs Bz

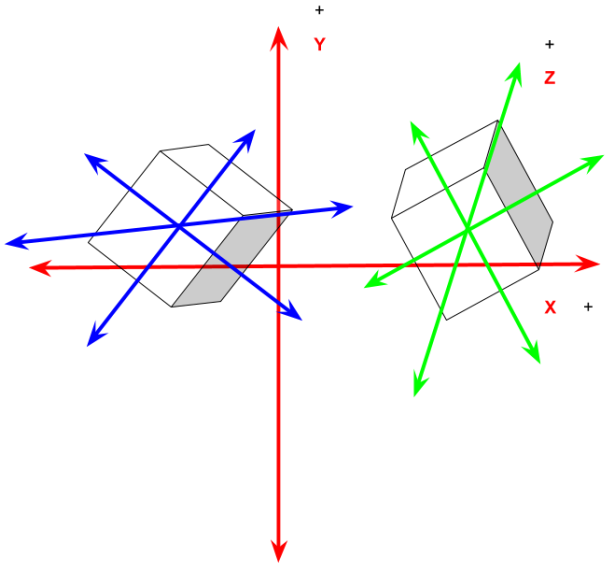


Figura 67: Representació de dos objectes OBB en 3D

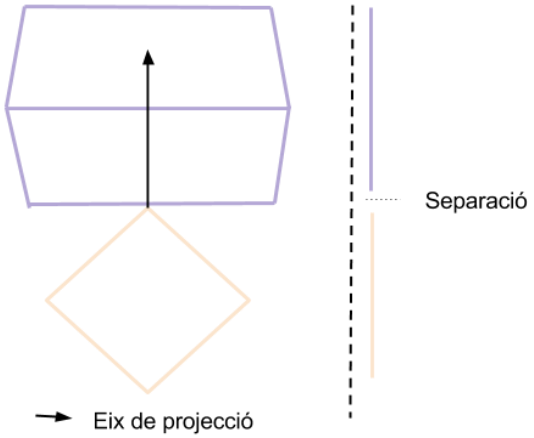


Figura 68: Exemple de la necessitat dels 15 eixos de comprovació

7.4 Signed Distance Field

Un **Signed Distance Field**[33] (SDF) es representa com una quadrícula de mostreig de la distància més propera a la superfície d'un objecte representat com un model poligonal. Generalment s'utilitza la convenció de definir amb valors negatius si s'està dins de l'objecte i amb valor positius si s'està fora de l'objecte.

SDF és molt utilitzat en el món dels gràfics per ordinador per a realitzar detecció de col·lisions fent ús de l'algoritme de Raymarching[34] i les Signed Distance Functions, però no s'ha trobat cap motor en si que utilitzi aquesta tècnica com a base.

7.4.1 Signed Distance Functions

Les **Signed Distance Functions** es basen en la idea que tot objecte ha de poder-se representar amb una funció. Cada funció rep un punt de dins de l'espai de 3 dimensions com a paràmetre, i retorna la distància mínima entre el punt i alguna superfície.

El signe del valor retornat indica si el punt està dins o fora d'aquesta superfície.

A continuació hi ha un exemple d'una esfera centrada a l'origen de coordenades. Els punts de dins l'esfera tindran una distància des de l'origen menor que el radi, els punts de l'esfera tindran la distància igual que el radi i els punts fora de l'esfera tindran distàncies majors al radi.

Així doncs una funció de Signed Distance per a una esfera centrada a l'origen i de radi 1 seria:

$$f(x, y, z) = \sqrt{x^2 + y^2 + z^2} - 1$$

Per exemple, donats els següents punts:

f(1,0,0) = 0

f(0,0,0.5) = -0.5

f(0,3,0) = 2

Es pot veure que el punt (1,0,0) es troba a la superfície de l'esfera, el punt (0,0,0.5) es troba dins de la superfície a 0.5 unitats de distància i el punt (0,3,0) es troba fora de la superfície amb el punt més proper a la superfície a 2 unitats de distància.

Quan es treballa per calcular l'ombrejat de les escenes amb **OpenGL Shading Language (GLSL)** és necessari vectoritzar les formules. Utilitzant la forma Euclidiana, la formula anterior quedaria:

$$f(\vec{p}) = \|\vec{p}\| - 1$$

Així doncs, amb GLSL es transformaria en la següent funció:

```
float sphereSDF(vec3 p) {
      return length(p) - 1.0;
}
```

7.4.2 SdBox

Per al projecte s'ha fet ús de la primitiva **Box-signed-exacta** que serveix per crear una Box mitjançant la següent funció:

```
flota sdBox( vec3 p, vec3 b ) {
      vec3 d = abs(p) - b;
      return min(max(d.x,max(d.y,d.z)),0.0) +
      length(max(d,0.0));
}
```

La funció *sdBox* rep dos paràmetres, un vector en tres dimensions (vector3) que representa la posició del Box i un vector3 que representa la mida de la mínima Bounding Box de l'objecte (Box).
Es calcula el valor de la distància restant la meitat del Box al valor absolut de la posició. En fer el valor absolut de la posició es trasllada el càlcul al quadrant positiu dels eixos de coordenades.

En realitzar la resta entre cadascuna de les components dels vectors es poden donar els següents casos:

- Que el component de la b sigui més gran que el component de la p. En aquest cas el component de la d donarà negativa.
- Que el component de la b sigui més petita que el component de la p. En aquest cas el component de la d donarà positiva.
- Que el component de la b sigui igual que la component de la p. En aquest cas el component de la d donarà zero.

Un cop calculades les 3 components de la d, si alguna és negativa, llavors com a resultat es retornarà la component amb valor mínim. En cas contrari és busca si alguna és negativa i es canvia per 0, i es retorna la longitud de la nova d.

7.4.3 Exemples pràctics

A continuació hi ha tres exemples de com es realitza el càlcul de la distància mínima en cada cas.

7.4.3.1 Exemple on la component de la distància dona negativa

p = (-3, -3, -3)
b = (4, 4, 4)
d = (-1, -1, -1)
distància mínima = min(-1,0) + lenght(0,0,0) = -1

7.4.3.2 Exemple on la component de la distància dona positiva

p = (5, 5, 5)
b = (4, 4, 4)
d = (1, 1, 1)
distància mínima = min(1,0) + lenght(1,1,1) = $\sqrt{3}$ = 1.73205080757

7.4.3.3 Exemple on la component de la distància dona igual a 0

p = (4, 0, 0)

b = (4, 4, 4)

d = (0, -4, -4)

distància mínima = min(0,0) + lenght(0,0,0) = 0

7.5 L'Algoritme de Gilbert-Johnson-Keerthi

Gilbert-Johnson-Keerthi[35] (GJK) és un algoritme iteratiu que serveix per detectar interseccions entre dos objectes convexos. L'algoritme complet de GJK no només es centra en la detecció de la col·lisió sinó que també retorna la informació de separació entre els objectes (si aquests no intersecccionen), és a dir, la distància mínima de col·lisió.

Per poder comprendre el funcionament de l'algoritme i les seves propietats és necessari entendre els conceptes en els quals està basat: Minkowski Addition, Minkowski difference, el simplex i les funcions de suport.

7.5.1 Minkowski Addition i Minkowski Difference

Els algoritmes de **Minkowski Addition** (MA) i de **Minkowski Difference** (MD)**,** són la base de l'algoritme GJK.

$$\text{MA: } A + B = \{a + b \mid a \in A, b \in B\}$$
$$\text{MD: } A - B = \{a - b \mid a \in A, b \in B\}$$

7.5.1.1 La Minkowski Addition (MA) o Minkowski Sum[36]

Geomètricament s'entén que la MA de dos conjunts de vectors A i B es forma sumant cada vector d'A a cada vector de B.

A la *Figura 69* es pot veure com es genera la MA entre el rectangle i el cercle, col·locant el centre del cercle a cadascun dels vèrtexs del rectangle.

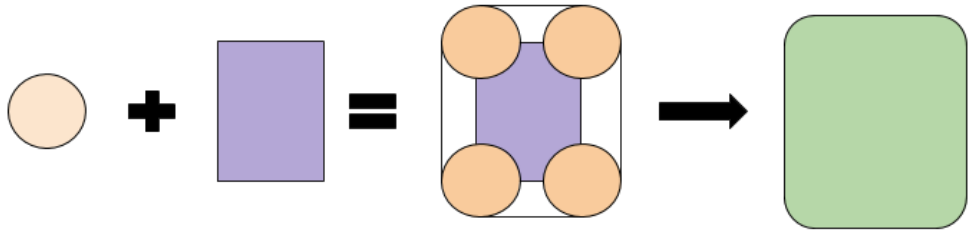


Figura 69: Exemple de visualització de la MA

A continuació hi ha un exemple de com es realitza el càlcul dels punts de la MA tenint dos objectes triangulars, A i B, amb els seus respectius conjunts de punts. Es calculen els punts de la MA sumant cada vèrtex del conjunt d'A amb cada vèrtex del conjunt de B. A la *Figura 70* es pot visualitzar els dos objectes i com queda la seva MA.

$$A = \{(1, 0), (0, 1), (0, -1)\}$$

$$B = \{(0, 0), (1, 1), (1, -1)\}$$

$$A + B = \{(1, 0), (2, 1), (2, -1), (0, 1), (1, 2), (1, 0), (0, -1), (1, 0), (1, -2)\}$$

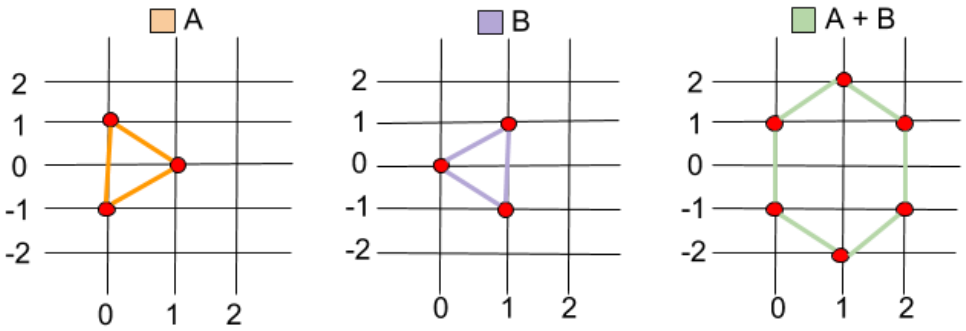


Figura 70: Exemple de càlcul de la MA de dos triangles

7.5.1.2 La Minkowski Difference (MD)

Anàlogament la **Minkowski Difference** de dos conjunts s'entén com la Minkowski Addition entre el primer conjunt contra la reflexió del segon conjunt a través de l'origen de coordenades.

Per a un objecte A, anomenem -A la reflexió d'A sobre l'origen. Així doncs la MD A-B es pot obtenir computant la MA de A i -B.

A continuació hi ha un exemple de com es realitza el càlcul dels punts de la MD tenint dos objectes triangulars A i B, amb els seus respectius conjunts de punts. Es calculen els punts de la MD restant cada vèrtex del conjunt d'A amb cada vèrtex del conjunt de B. A la *Figura 71* es pot visualitzar els dos objectes i com queda la seva MD.

$$A = \{(1, 0), (0, 1), (0, -1)\}$$

$$B = \{(0, 0), (1, 1), (1, -1)\}$$

$$A - B = \{(1, 0), (0, -1), (0, 1), (0, 1), (-1, 0), (-1, 2), (0, -1), (-1, -2), (-1, 0)\}$$

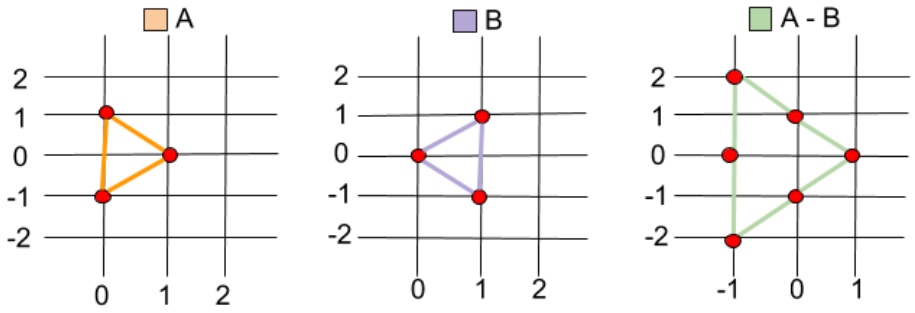


Figura 71: Exemple de càlcul de la MD de dos triangles

7.5.2 Propietats de l'algoritme

L'algoritme GJK està basat en les següents quatre propietats.

- **P1:** Donat dos objectes convexos A i B, la Minkowski Addition A+B i la Minkowski Difference A-B és també un objecte convex.
- **P2:** La Minkowski Difference de dos objectes conté l'origen sí i només sí els dos objectes intersecciones, ja que existeix un punt comú entre els dos objectes.
- **P3:** Sent MA el conjunt de punts de la Minkowski Addition de dos objectes convexos A i B. Els punts de suport de la MA són la suma dels punts de suport d'A més els punts de suport de B.
- **P4:** Sent MD el conjunt de punts de la Minkowski Difference de dos objectes convexos A i B. Els punts de suport de la MD són els punts de suport d'A en la direcció $\vec{d}$ menys els punts de suport de B en la direcció del vector director oposat (-$\vec{d}$).

Per entendre la P2 a continuació hi ha dos exemples.

En el primer els objectes no col·lisionen i per tant la MD no conté l'origen.

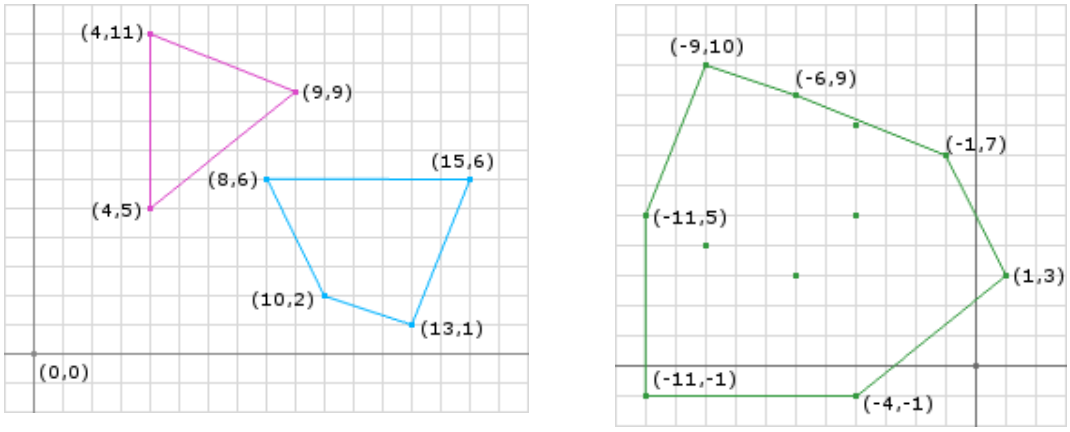


Figura 72: Exemple de càlcul en 2D de la MD de dos objectes que no interseccionen

En canvi en aquest segon exemple, com que els objectes col·lisionen la MD sí que conté l'origen.

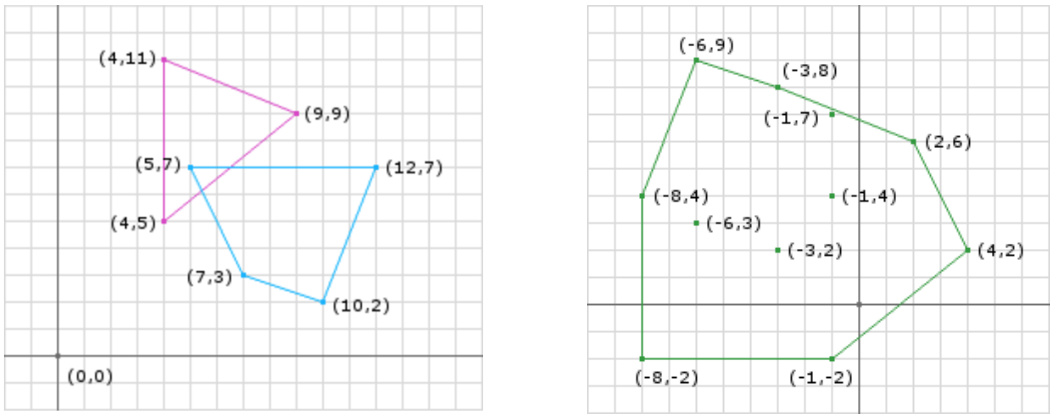


Figura 73: Exemple de càlcul en 2D de la MD de dos objectes que interseccionen

7.5.3 Simplex

GJK en comptes de treballar amb els volums opera amb els punts de la MD. Un dels punts forts de l'algoritme és que no necessita calcular tots els punts de la MD, sinó que busca omplir iterativament un **simplex** amb la quantitat de punts de la MD necessaris fins que en aquest s'inclogui l'origen o l'algoritme determini que l'origen no pot ser encapsulat.

Depenent del tipus d'espai en el qual es treballa és necessari un simplex de més o menys punts. Per un espai de k-dimensions, sent $k \geq 0$, es defineix un k-simplex com el conjunt de k+1 punts en un espai de k-dimensions.

Un 0-simplex seria un punt, un 1-simplex serien dos punts (una línia), un 2-simplex serien tres punts (un triangle), un 3-simplex serien quatre punts (un tetraedre), i així per n-dimensions es tindria un n-simplex i un conjunt de n+1 punts.

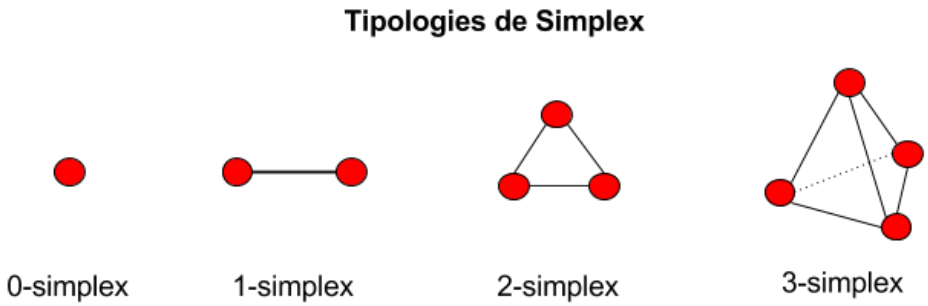


Figura 74: Tipologies de simplex

7.5.4 La funció de suport

Per construir el simplex es fa ús d'una **funció de suport**, una funció que donat els dos objectes retorna el millor punt de la MD per completar l'algoritme segons una direcció $\vec{d}$ de cerca.

Un pla P divideix l'espai en dues meitats i P dona suport a un objecte A si segueix les següents dues condicions:

- A està contingut completament en una de les dues meitats determinades per P.
- Com a mínim un dels punts fronterers d'A està contingut en P.

Els plans de suport són escollits per una direcció $\vec{d}$ específica que va variant al llarg de l'execució de l'algoritme (ja es tractarà amb més profunditat a l'apartat d'implementació del GJK). Així doncs és natural definir el **punt de suport** associat a un objecte A que mapeja un vector director $\vec{d}$ a la distància en què es troba el pla de suport de l'origen.

$$\text{Punt de suport d'A}(\vec{d}) = \max\{\, \vec{d} \cdot a : a \in A\}$$

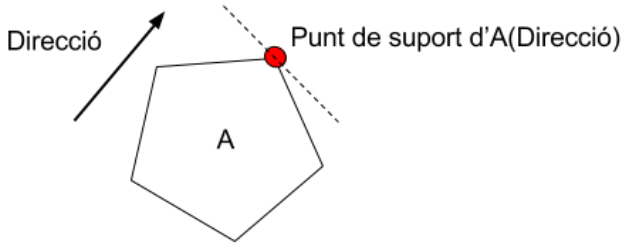


Figura 75: Exemple de punt de suport

Les funcions de suport satisfan les propietats P3 i P4:

P3 ⇒ Punts de suport de la MA($\vec{d}$) = Punt de suport de A($\vec{d}$) + Punt de suport de B($\vec{d}$)

P4 ⇒ Punts de suport de la MD($\vec{d}$) = Punt de suport de A($\vec{d}$) - Punt de suport de B(-$\vec{d}$)

Per justificar el punt 4 cal observar que:

Punt de suport de -B ($\vec{d}$) = - Punt de suport de B(-$\vec{d}$).

Les funcions de suport es defineixen com la distància dels plans de suport a l'origen, GJK necessita que es conegui el punt del pla. Així doncs el punt de suport d'un objecte A és el punt de la superfície d'A més allunyat segons el vector director.

Remarcar que els punts que genera la funció de suport i els punts amb què s'anirà omplint el simplex no són dos objectes sinó els punts de la MD dels dos objectes. La MD pot arribar a estar composta per un gran nombre de punts, calcular-los tots requeriria gran temps de processament, així doncs el càlcul que realitza la funció de suport es basa en la P4.
D'aquesta manera la construcció del simplex es realitza només amb aquells punts que tinguin més possibilitats de contenir l'origen. Com a optimització el valor retornat per la funció de suport és el punt que es troba més allunyat en la direcció $\vec{d}$. El fet d'escollir el simplex d'aquesta manera té una gran importància per a l'algoritme, ja que permet crear un simplex que contingui l'àrea màxima i d'aquesta manera s'incrementin les opcions perquè aquest finalitzi ràpidament.

7.5.5 Exemples pràctics

Per entendre millor a què es refereix la P2, a continuació hi ha un exemple en un espai 2D amb cercles.

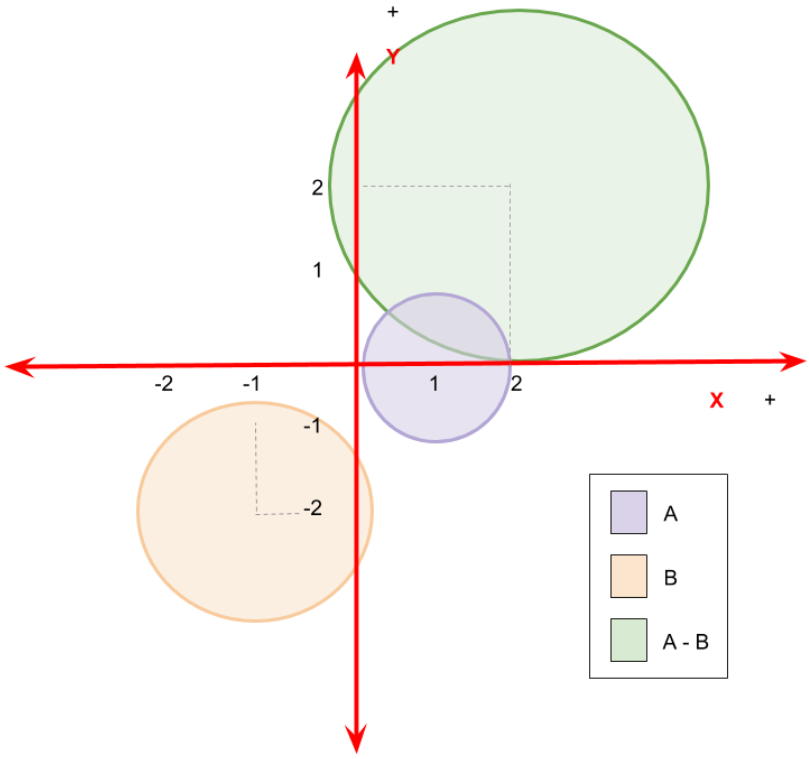


Figura 76: Exemple d'aplicació de la P2: Dos cercles que no interseccionen

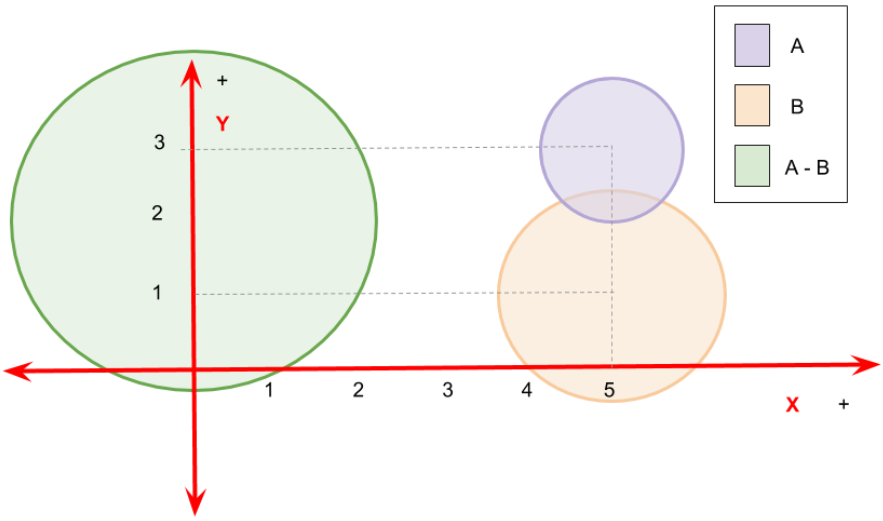


Figura 77: Exemple d'aplicació de la P2: Dos cercles que interseccionen

A la *Figura 76* A i B no interseccionen i per tant la seva MD no conté l'origen, en canvi a la *Figura 77* A i B interseccionen i la seva MD sí que conté l'origen.

En els exemples de les *Figures 72 i 73* també es pot observar si els objectes interseccionen, ja que la MD engloba l'origen, o no.

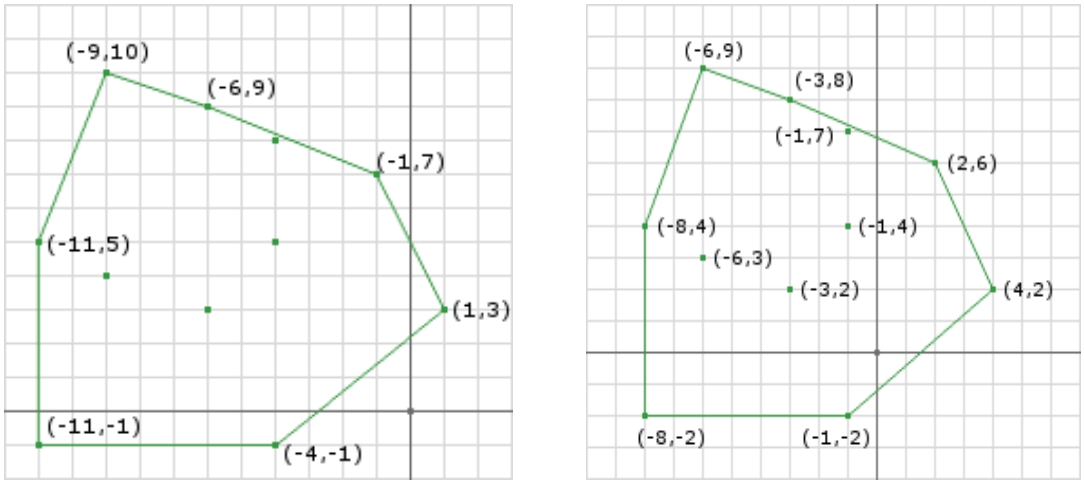


Figura 78: Comparativa dels resultats de les MD. Esquerra: no intersecció (no inclou origen). Dreta: intersecció (inclou origen)

7.6 Disseny orientat a dades

Un dels punts que és vol estudiar en aquest projecte es comprovar si hi ha millora d'eficiència si es fa servir un **disseny orientat a dades** (data-oriented design, DOD) envers un **disseny orientat a objectes**[37] (object-oriented programming, OOP).

7.6.1 Data Oriented Design

Per comprendre en què consisteix el data-oriented design (DOD) a continuació es descriuen 4 punts en els quals es basa:

- Pensar en les dades primer i en el codi segon. La jerarquia de classes no és important però els patrons d'accés a dades sí.
- Pensa com les dades són accedides en el projecte, com es transformen i que s'acaba realitzant amb elles.
- Quan hi ha un, hi ha molts. S'ha de pensar en grans conjunts de dades.
- Vigilar amb l'overhead de les funcions virtuals, punters a funcions i punters a funcions de mètodes.

Per entendre la importància del primer i del segon punt a continuació hi ha un exemple.

```
char* data = pointerToSomeData;
unsigned int  sum = 0;
for  (unsigned int  i=0; i<1000000; ++i, ++data)
{
  sum += *data;
}
```

Al codi superior es realitza la suma d'un milió de bytes. Triga 0,7 ms.

```
char* data = pointerToSomeData;
unsigned int  sum = 0;
for  (unsigned int  i=0; i<1000000; ++i, data += 16)
{
  sum += *data;
}
```

En aquest segon codi es realitza la mateixa suma, però aquesta es realitza cada 16 elements, es segueixen sumant un milió de bytes. Triga 5 ms a executar en les mateixes condicions.

Com es pot comprovar el segon codi és set vegades més lent tot i que s'estan accedint a la mateixa quantitat d'elements.

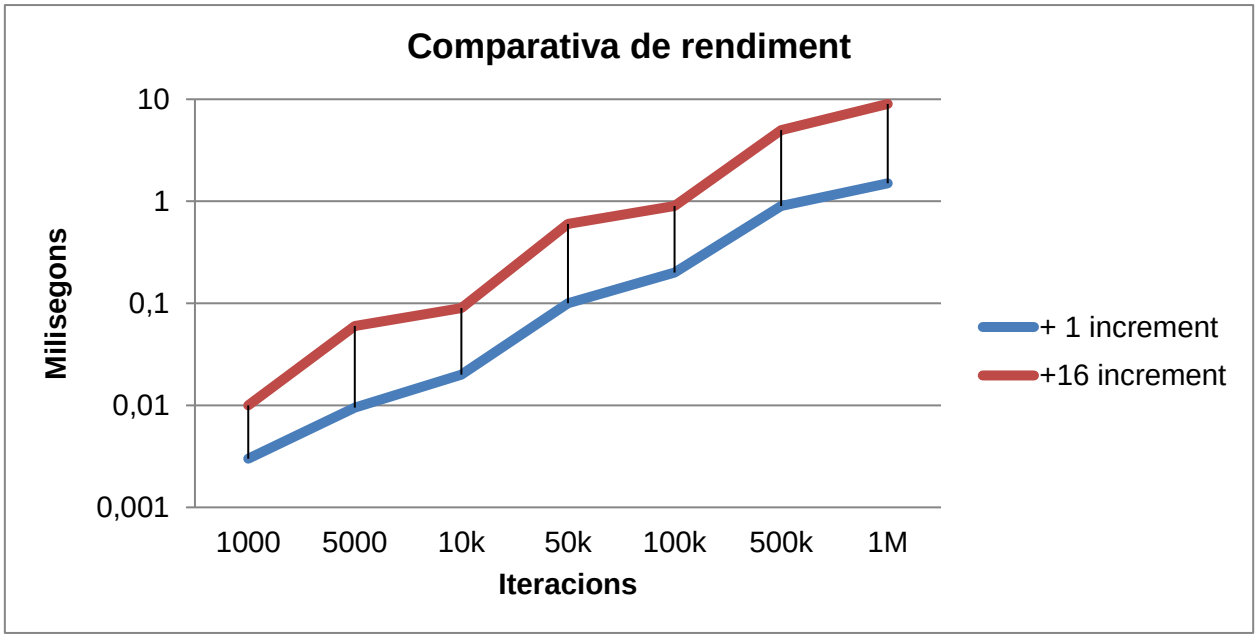


Figura 79: Comparativa dels resultats

La diferència mitjana de rendiment és de 5x i és consistent quan s'itera des de 1000 bytes fins a un milió de bytes (pot arriba a un 7x).

Per entendre les diferencies de rendiment dels dos codis és necessari pensar que està succeint en realitat. El programa indica a la CPU que executi una instrucció però la memòria de la CPU no té accés a les dades dins de la cache, així doncs espera a què

les dades es copiïn a la cache i així puguin ser accessibles per la CPU. Aquest procés es coneix com a Cache Miss. Assumint que la memòria cache L1 té una mida de línia de cache de 16 bytes, significa que la cache pot copiar 16 bytes cada cop, començant per l'adreça demanada data[i].

En el primer codi el programa intenta realitzar la suma del següent byte que ja ha sigut copiat dins de la cache L1, així doncs pot continuar calculant sense trobar un altra error de cache. Durant les 14 següents iteracions el programa es comporta de la mateixa manera, però després d'accedir a 16 bytes des del primer cache miss el bucle es trobarà en un altre fallada de cache i la CPU espera que es realitzi la copia de la següent línia de 16 bytes dins de la L1 per a seguir calculant.

En el segon codi el bucle salta 16 bytes cada cop però això no canvia la manera en la qual el hardware treballa. La cache copia 16 subsequències de bytes cada cop, ja que a cada iteració es troba amb un cache miss.

Aquestes esperes que la CPU ha de fer per a obtenir les dades són les que provoquen que el codi sigui unes 5-7 vegades més lent.

Respecte al punt 3, s'ha de considerar que en cas que per a un projecte s'hagi de definir un punt, un vector, un component, una mesh, una textura... serà necessari treballar amb molts més punts, vectors, components, meshes i textures. Per això és important escriure el codi de manera que es pugui fer front a molts d'ells a la vegada i pensar en termes de fluxos d'objectes. Transformant les classes en arrays de dades (que es tractarà més endavant).

Respecte al punt 4, és important remarcar que no només les variables són afectades per la cache, sinó que les funcions també s'emmagatzemen en memòria i per tant utilitzar l'herència i fer crides a funcions virtuals també poden desencadenar cache misses.

7.6.2 Structures of arrays vs arrays of structures

El disseny orientat a dades busca dissenyar l'arquitectura del projecte de manera que tingui sentit quan es transformin les dades. Així doncs en comptes de fer ús d'una arquitectura basada en vectors d'estructures (array of structures, AoS) utilitza un disseny basat en estructures de vectors (structures of arrays, SoA) estretament modulada i optimitzada per al consum de la CPU.

D'aquesta manera el que es vol aconseguir és assegurar que les dades a les quals es vol accedir estiguin situades contiguament en memòria i, d'aquesta manera poder accedir a un bloc de dades cada cop i reduir els cache misses.

7.6.3 Multithreading

Una altra dels avantatges que té disposar d'un disseny orientat a dades és que dóna facilitats per poder disposar d'un sistema multi-threading.

Conèixer les dades i saber en quin punt es transformen permet poder paral·lelitzar els càlculs. Es poden dividir les dades en trossos i utilitzar cada tros perquè el càlcul el realitzi un thread diferent. D'aquesta manera no es necessita cap mena de sincronització i es pot escalar fàcilment permetent gaudir dels avantatges de treballar amb N-cores.

7.6.4 Copiabilitat i mobilitat

Per poder realitzar copies de les dades facilment fent ús de la funció memcpy() o memmove() només és necessari:

- En comptes d'emmagatzemar les dades com un vector<Box*> fill, emmagatzemar un Box* pare.
- Emmagatzemar els índexs en comptes dels punters. Fer ús de uint16_t pare en comptes del punter. D'aquesta manera s'utilitza menys espai (sobretot en sistemes de 64 bits).

8. Requisits

L'enginyeria de requisits, és l'àrea de coneixement de l'enginyeria de software relativa a l'obtenció d'objectius, funcions i restriccions dels sistemes software al món real. S'encarrega de la relació d'aquests factors per precisar les especificacions del comportament del software, i la seva evolució al llarg del temps. En termes generals, la identificació de requisits és el procés de descobrir l'objectiu del sistema d'informació mitjançant la identificació dels stakeholders i les seves necessitats.[38]

Una de les principals fases del projecte, va ser l'anàlisi de requisits que calia complir amb el projecte, ja que calia descriure el comportament del sistema a desenvolupar. Es van dividir els requisits en funcionals i no funcionals.

A continuació es presenten doncs els requisits marcats per a la primera versió i la seva justificació. Pel que fa als requisits no funcionals es presenten també els criteris de satisfacció de cadascun d'ells.

8.1 Requisits funcionals

Els requisits funcionals, fan referència a la funcionalitat que ha de proporcionar el sistema a desenvolupar. Els requisits són complets i prou detallats, per això no hi ha el criteri de satisfacció.

El motor ha de funcionar sense fer ús de cap motor extern.
El motor ha de ser creat des de zero sense fer ús de cap motor o component extern. Es permetrà utilitzar de llibreries matemàtiques per facilitar els càlculs necessaris.

El motor ha de poder detectar totes les col·lisions que es donin entre tots els objectes afegits al motor.
Entre tots els objectes definits el motor ha de poder detectar totes les col·lisions que es puguin donar.

El motor ha de poder detectar totes les col·lisions que es donin entre dos conjunts d'objectes.
El motor ha de rebre els identificadors dels objectes que formen els dos conjunts i ha

de retornar quins objectes del primer conjunt col·lisionen amb quins objectes del segon conjunt.

El motor ha de poder treballar amb objectes 2D i 3D.
El motor ha de ser capaç de treballar amb objectes definits en dues i tres dimensions.

El motor ha de poder treballar amb objectes alineats i girats.
El motor ha de ser capaç de treballar amb objectes alineats als eixos de l'escena i amb objectes que disposin de la seva pròpia rotació.

El motor ha de poder treballar amb objectes convexos.
El motor únicament ha de rebre objectes convexos.

El motor ha de poder treballar amb la informació de translació, rotació i escalat.
El motor ha de poder treballar amb la informació de translació i rotació dels objectes tant si aquesta li ha arribat com a matriu de transformació o mitjançant un vector de translació i un quaternió. La informació d'escalat vindrà implícita en la definició dels vèrtexs.

8.2 Requisits no funcionals

A continuació es defineixen els requisits no funcionals del sistema, aquests són les característiques requerides del sistema, del procés de desenvolupament, del servei prestat o de qualsevol aspecte de desenvolupament. Han estat subdividits en diferents categories.

8.2.1 Requisits de rendiment

El motor ha de poder calcular totes les col·lisions que es donin entre centenars d'objectes en menys de 3-4 mil·lisegons.
Tipus de requisit: Rendiment
Descripció: Depenent del nombre d'objectes dels quals s'hagi de calcular la col·lisió el motor ha de donar una resposta com a màxim en 3-4 mil·lisegons.
Criteri de satisfacció: Es requerirà que la tècnica i el disseny escollits assoleixin el compliment d'aquest requisit.

El motor ha de poder suportar escenes amb 1000 objectes.
Tipus de requisit: Rendiment
Descripció: El motor serà utilitzat en diverses aplicacions, per aquest motiu ha de poder treballar amb escenes de com a mínim 1000 objectes.
Criteri de satisfacció: Es requerirà que el motor assoleixi el compliment d'aquest requisit. En el cas del disseny orientat a dades es realitzarà una reserva de memòria que permeti com a mínim emmagatzemar 1000 objectes (tenint en compte totes les dades que el formin).

El motor ha de ser totalment escalable.
Tipus de requisit: Escalabilitat
Descripció: El motor ha de tenir un disseny completament escalable, ja que el nombre de tècniques o de tests es pot incrementar durant el seu desenvolupament.
Criteri de satisfacció: Es realitzarà un disseny que permeti l'annexió tant de noves tècniques com de nous tests.

8.2.2 Requisits operacionals

El motor com a component del framework ha de poder ser usat en plataformes Android, iOS i web.
Tipus de requisit: Operacional
Descripció: Es pretén arribar al màxim d'usuaris possibles dins d'una viabilitat econòmica. Per aquest motiu el motor s'ha de desenvolupar de manera que es pugui fer ús en les diferents plataformes.
Criteri de satisfacció: Perquè el framework es pugui utilitzar en les diverses plataformes, el motor estarà implementat en el llenguatge c++. D'aquesta manera podrà ser compilat a les diferents plataformes.

El motor com a component del framework ha de poder funcionar en dispositius Android amb un sistema operatiu igual o superior a la versió 5
Tipus de requisit: Operacional
Descripció: Donades les necessitats tècniques del framework el motor hauria de ser compatible amb Android 5 o superior.
Criteri de satisfacció: Perquè es pugui fer us del framework des dels diversos dispositius Android, el motor estarà implementat en el llenguatge c++. D'aquesta

manera podrà ser compilat a Java mitjançant de la llibreria Android NDK[3]. Es faran proves amb els diferents dispositius Android per comprovar el correcte funcionament.

El motor com a component del framework ha de poder funcionar en dispositius iOS amb sistema operatiu igual o superior a la versió 9.
Tipus de requisit: Operacional
Descripció: Donades les necessitats tècniques del framework el motor hauria de ser compatible amb iOS 9 o superior.
Criteri de satisfacció: Perquè es pugui fer ús del framework des dels diversos dispositius iOS, el motor estarà implementat en el llenguatge c++. D'aquesta manera podrà ser compilat a Objective-C mitjançant LLVM[4]. Es faran proves amb els diferents dispositius iOS per comprovar el correcte funcionament.

El motor com a component del framework ha de poder funcionar des de diversos navegadors Firefox, Chrome i Internet Explorer.
Tipus de requisit: Accessibilitat
Descripció: El motor com a component del framework ha de poder ser utilitzat des de Firefox, Chrome, Internet Explorer, en les seves tres últimes versions a data d'abril del 2017.
Criteri de satisfacció: Perquè el framework es pugui fer ús des dels diversos navegadors, el motor estarà implementat en el llenguatge c++. D'aquesta manera podrà ser compilat a emscriptem[5] mitjançant LLVM. Es realitzaran proves amb els diferents navegadors per comprovar el correcte funcionament.

8.2.3 Requisits de dades

El motor ha de rebre la informació dels objectes en el format correcte.
Tipus de requisit: Dades
Descripció: El motor rebrà un arxiu amb la informació dels objectes dels quals es vulgui saber les col·lisions. Per cada objecte s'ha de tenir definida la seva BoundingBox (els vèrtexs que formen el cub centras a l'origen de l'objecte), la seva matriu de transformació (amb la informació de translació i rotació) o en cas contrari un

[3] Android NDK: https://developer.android.com/ndk/index.html
[4] LLVM: compilador de c++ a altres llenguatges. https://en.wikipedia.org/wiki/LLVM
[5] emscriptem: Compilador de C++ a javaScript https://github.com/kripken/emscripten

vector de translació i un quaternió amb la informació de rotació.
Criteri de satisfacció: Les dades que rebi el motor seran prèviament comprovades pel framework. En cas que les dades no compleixin la definició els resultats donaran erronis.

El motor ha de rebre quins objectes formen cadascun dels conjunts.
Tipus de requisit: Dades
Descripció: En cas de voler-se detectar les col·lisions entre dos conjunts d'objectes, el motor rebrà un arxiu amb els identificadors dels objectes que componen cadascun dels conjunts.
Criteri de satisfacció: Les dades que rebi el motor seran prèviament comprovades pel framework. En cas que les dades no compleixin la definició els resultats donaran erronis.

9 Especificació (Anàlisis funcional)

Per definir de la forma més precisa el motor a realitzar, en aquest capítol es defineix tota l'especificació necessària per implementar tots els requisits funcionals explicats en el capítol anterior.

9.1 Diagrama d'usuaris

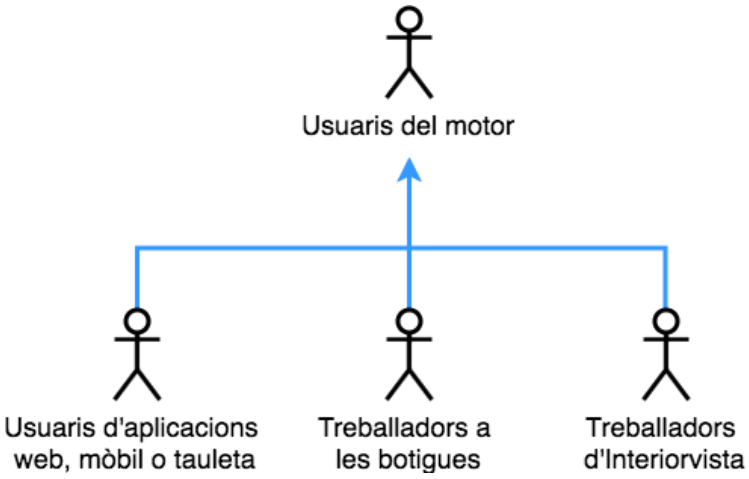


Figura 80: Diagrama d'usuaris del motor

9.2 Diagrama de casos d'ús

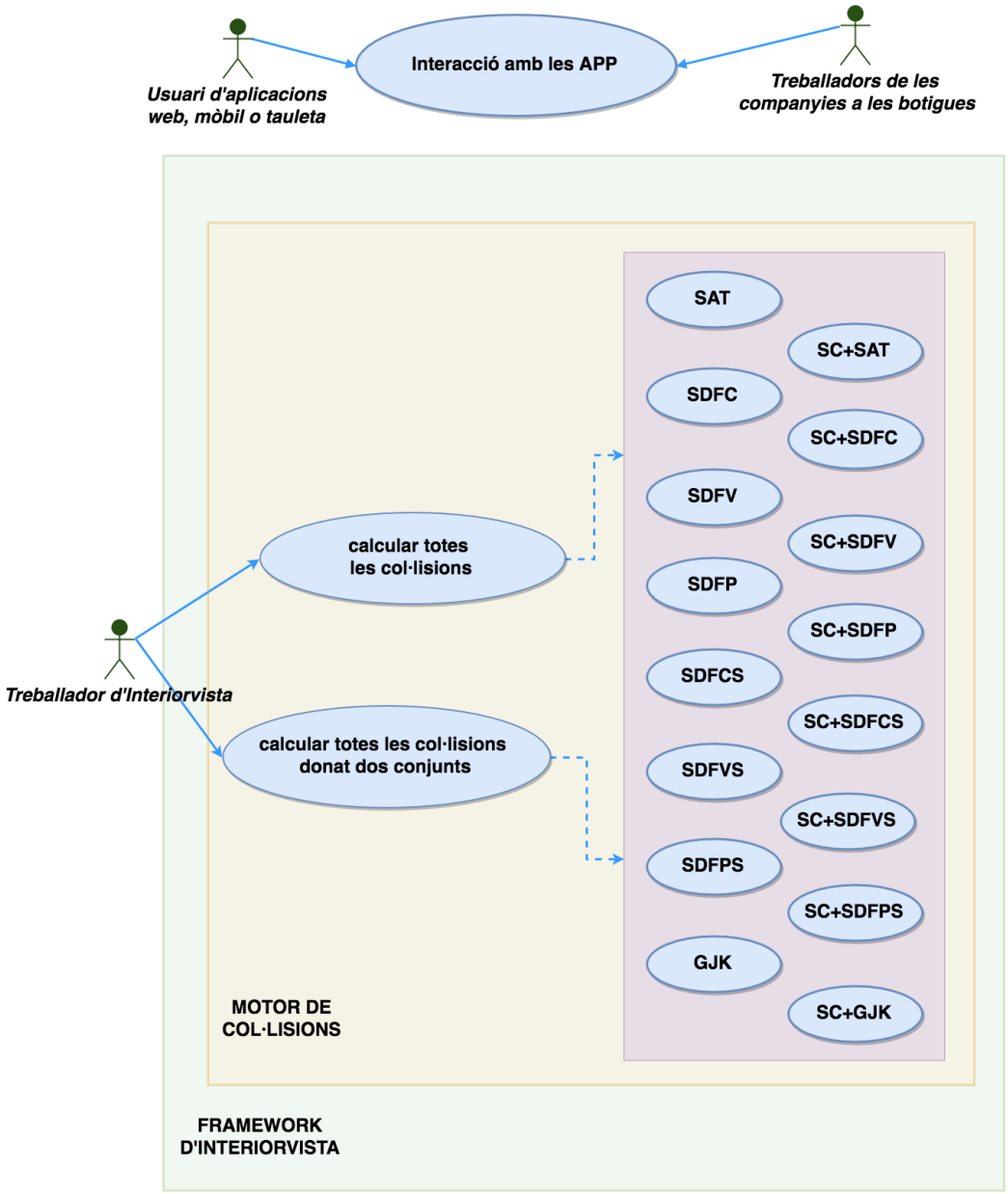


Figura 81: Diagrama de casos d'ús del motor

[6]

[6] SAT: Calcular col·lisions mitjançant la pipe **Separation Axis Theorem.**
SDFC: Calcular col·lisions mitjançant la pipe **Signed Distance Field completa**.
SDFV: Calcular col·lisions mitjançant la pipe **Signed Distance Field per vèrtex**.
SDFP: Calcular col·lisions mitjançant la pipe **Signed Distance Field per punts de mostreig**.
SDFCS: Calcular col·lisions mitjançant la pipe **Signed Distance Field completa fent ús del control de retorn ràpid.**
SDFVS: Calcular col·lisions mitjançant la pipe **Signed Distance Field vèrtex fent ús del control de retorn ràpid.**
SDFPS: Calcular col·lisions mitjançant la pipe **Signed Distance Field per punts de mostreig fent ús del control de retorn ràpid.**
GJK: Calcular col·lisions mitjançant la pipe **Gilbert-Johnson-Keerthi**.

SC+*pipe*: Calcular col·lisions mitjançant la pipe que engloba la tècnica de **col·lisió d'esferes** i la pipe.

Els usuaris d'aplicacions i els treballadors de les botigues únicament es beneficiaran del motor de col·lisions mitjançant l'ús de les aplicacions.

Els treballadors d'Interiorvista faran ús del motor directament. A continuació es llisten els casos d'ús dels treballadors d'Interiorvista i una descripció de cadascun d'ells.

9.2.1 Calcular totes les col·lisions

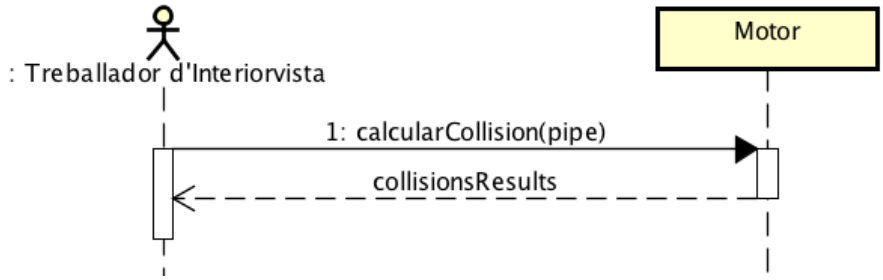


Figura 82: Diagrama del cas d'ús Calcular totes les col·lisions

Descripció:	Un treballador d'Interiorvista vol calcular totes les col·lisions possibles entre els objectes. A través d'aquest cas d'ús el treballador introduirà quina pipe vol fer servir perquè el motor calculi les col·lisions.
Especificació:	
Actor principal:	Treballador d'Interiorvista
Precondició:	El motor ha de rebre els objectes per part del Framework. Anteriorment s'han d'haver afegit els objectes al motor.
Disparador:	El treballador d'Interiorvista vol calcular totes les col·lisions entre els objectes actius del motor.
Escenari principal:	1. El treballador d'Interiorvista indica que vol calcular totes les col·lisions entre tots els objectes actius al motor i selecciona quina pipe vol utilitzar. 2. El motor realitza l'execució de la pipe per cada parella d'objectes actius i retorna un llistat de resultats, indicant per cada parella d'objectes que col·lisionen els seus identificadors i el temps que ha trigat a realitzar el càlcul. 3. S'acaba el cas d'ús.
Extensions:	2a. El motor realitza l'execució de la pipe i no detecta cap col·lisió. 2a1. El motor indica que no s'han detectat col·lisions. 2a2. Es torna al punt 3.

Taula 43: Especificació del cas d'ús Calcular totes les col·lisions

9.2.2 Calcular totes les col·lisions donat dos conjunts

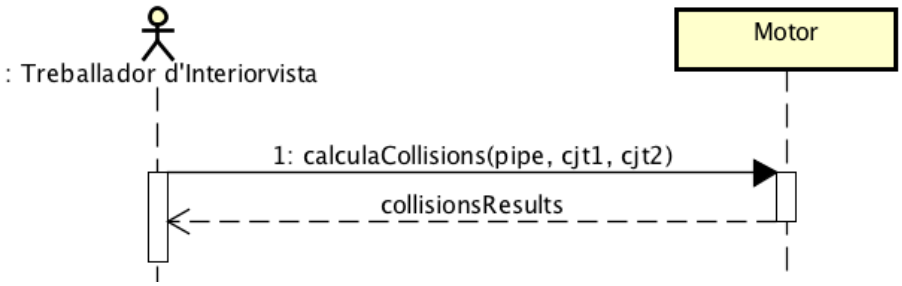


Figura 83: Diagrama del cas d'ús Calcular totes les col·lisions donat dos objectes

Descripció:	Un treballador d'Interiorvista vol calcular totes les col·lisions possibles entre dos conjunts d'objectes. A través d'aquest cas d'ús el treballador introduirà quina pipe vol fer servir perquè el motor calculi les col·lisions i quins objectes formen ambdós conjunts que es volen comprovar.
Especificació:	
Actor principal:	Treballador d'Interiorvista
Precondició:	El motor ha de rebre els objectes continguts en els conjunts per part del Framework. Anteriorment s'han d'haver afegit els objectes al motor.
Disparador:	El treballador d'Interiorvista vol calcular totes les col·lisions entre els objectes actius del motor que es trobin definits en els conjunts.
Escenari principal:	1. El treballador d'Interiorvista indica que vol calcular totes les col·lisions entre dos conjunts d'objectes. Defineix quins componen el primer conjunt i quins objectes componen el segon. A més selecciona quina pipe vol utilitzar. 2. El motor realitza l'execució de la pipe per cada parella d'objectes i retorna un llistat de resultats, indicant per cada parella d'objectes que col·lisionen els seus identificadors i el temps que ha trigat a realitzar el càlcul. 3. S'acaba el cas d'ús.
Extensions:	2a. El motor realitza l'execució de la pipe i no detecta cap col·lisió. 2a1. El motor indica que no s'han detectat col·lisions. 2a2. Es torna al punt 3.

Taula 44: Especificació del cas d'ús Calcular totes les col·lisions donat dos objectes

9.3 Model conceptual

9.3.1 Disseny orientat a objectes genèric

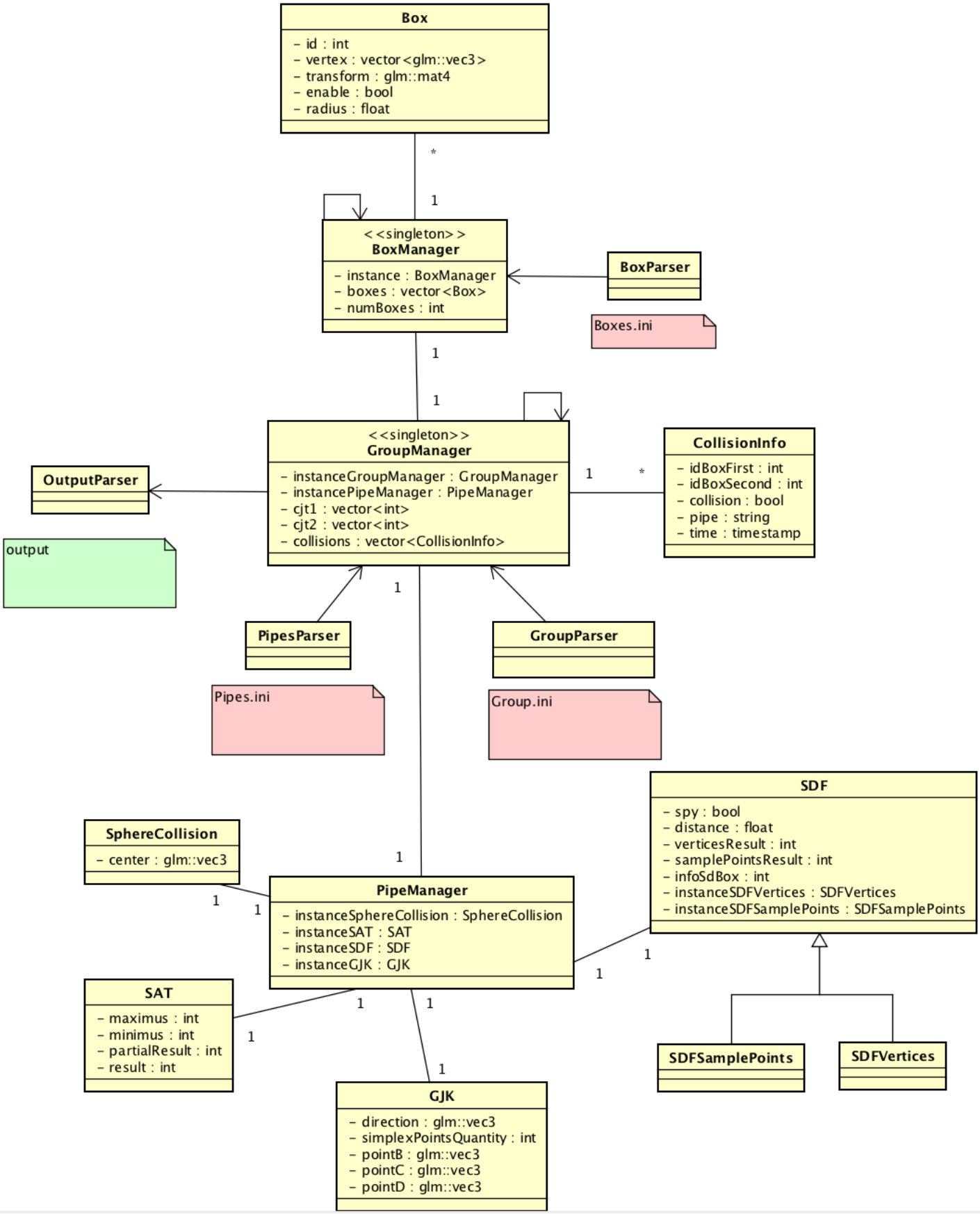


Figura 84: Model conceptual per al disseny orientat a objectes genèric

9.3.1.1 Descripció de les entitats

9.3.1.1.1 Box

Entitat que representa una Box del motor. Disposa del seu propi identificador (id). Per representar una Box s'emmagatzemen els vèrtexs originals escalats i la matriu de transformació amb la informació de rotació i translació. Una Box pot estar activa o inactiva, en cas de ser una Box inactiva no es realitzaran les comprovacions de col·lisió sobre les parelles que incloguin la Box inactiva.

ATRIBUT	DESCRIPCIÓ
id	Identificador únic d'una box
vertex	Vèrtexs originals de la box
transform	Matriu de transformació de la box
enable	Indicador d'activitat de la box: True per actiu i False per inactiu.
radius	Radi de l'esfera que conté la box.

Taula 45: Descripció dels atributs de la classe Box

9.3.1.1.2 Box Manager

Entitat encarregada de gestionar les diferents Box del motor. Disposa d'un llistat amb totes les Box inserides al motor.

ATRIBUT	DESCRIPCIÓ
instance	Instància de BoxManager
boxes	Llistat de les Box disponibles
numBoxes	Quantitat de Box disponibles

Taula 46: Descripció dels atributs de la classe Box Manager

9.3.1.1.3 Group Manager

Entitat encarregada de gestionar quina opció d'execució es vol utilitzar. Es pot escollir entre comprovar les col·lisions entre totes les Box actives al motor o especificar dos conjunts d'identificadors de Box i comprovar les col·lisions d'un conjunt contra l'altre conjunt.

ATRIBUT	DESCRIPCIÓ
instanceGroupManager	Instància de GroupManager
instancePipeManager	Instància de PipeManager
cjt1	Llistat amb els identificadors (id) de les Box que formen el conjunt 1
cjt2	Llistat amb els identificadors (id) de les Box que formen el conjunt 2
collisions	Llistat amb els resultats de les col·lisions. Per cada col·lisió comprovada retorna un CollisionInfo.

Taula 47: Descripció dels atributs de la classe Group Manager

9.3.1.1.4 Collision Info

Entitat que emmagatzema la informació de cada comprovació de col·lisió. S'identifiquen les dues Box de les que es realitza la comprovació, amb quina pipe s'executa, quin és el resultat de la comprovació i el temps d'execució.

ATRIBUT	DESCRIPCIÓ
idBoxFirst	Identificador (id) de la primera Box
idBoxSecond	Identificador (id) de la segona Box
collision	Resultat de la col·lisió: True si hi ha col·lisió, False si no hi ha col·lisió
pipe	Pipe amb la que s'ha realitzat la comprovació
time	Temps que s'ha trigat a obtenir el resultat

Taula 48: Descripció dels atributs de la classe Collision Info

9.3.1.1.5 Pipe Manager

Entitat encarregada de gestionar quina pipe es vol utilitzar. Es pot escollir entre les 16 pipes del motor.

ATRIBUT	DESCRIPCIÓ
instanciaSphereCollision	Instància de SphereCollision
instanciaSAT	Instància de SAT
instanciaSDF	Instància de SDF
instanciaGJK	Instància de GJK

Taula 49: Descripció dels atributs de la classe Pipe Manager

9.3.1.1.6 Sphere Collision

Entitat que emmagatzema el valor del centre transformat d'una box.

ATRIBUT	DESCRIPCIÓ
center	Centre transformat de la Box

Taula 50: Descripció dels atributs de la classe Sphere Collision

9.3.1.1.7 SAT

Entitat que representa la tècnica del Separated Axis Theorem.

ATRIBUT	DESCRIPCIÓ
maximus	Vèrtex màxims de la box1 i box2 respectivament
minimus	Vèrtex mínims de la box1 i box2 respectivament
partialResults	Resultats parcials de l'execució. partialResults[0]=resultats de les col·lisions en l'eix X. partialResults[1]=resultats de les col·lisions en l'eix Y. partialResults[2]=resultats de les col·lisions en l'eix Z.
result	Resultat final de l'execució

Taula 51: Descripció dels atributs de la classe SAT

9.3.1.1.8 SDF

Entitat que representa la tècnica del Signed Distance Field.

ATRIBUT	DESCRIPCIÓ
instanceSDFVertex	Instància de SDF per vèrtex
instanceSDFSamplePoints	Instància de SDF per punts de mostreig
spy	Indicador si es vol una execució de retorn ràpid. True per actiu i False per inactiu.
distance	Distància mínima entre un vèrtex d'una box i l'altre box.
verticesResult	Resultats de les col·lisions de les funcions de càlculs amb vèrtex.
samplePointsResult	Resultats de les col·lisions de les funcions de càlculs amb punts de mostreig.
infoSdBox	Informació necessària per calcular el sdBox.

Taula 52: Descripció dels atributs de la classe SDF

9.3.1.1.9 GJK

Entitat que representa la tècnica del Gilbert-Johnson-Keerthi.

ATRIBUT	DESCRIPCIÓ
instance	Instància de GJK
direction	Direcció de cerca
pointB	Segon punt del simplex
pointC	Tercer punt del simplex
pointD	Quart punt del simplex

Taula 53: Descripció dels atributs de la classe GJK

9.3.2 Disseny orientat a objectes GJK

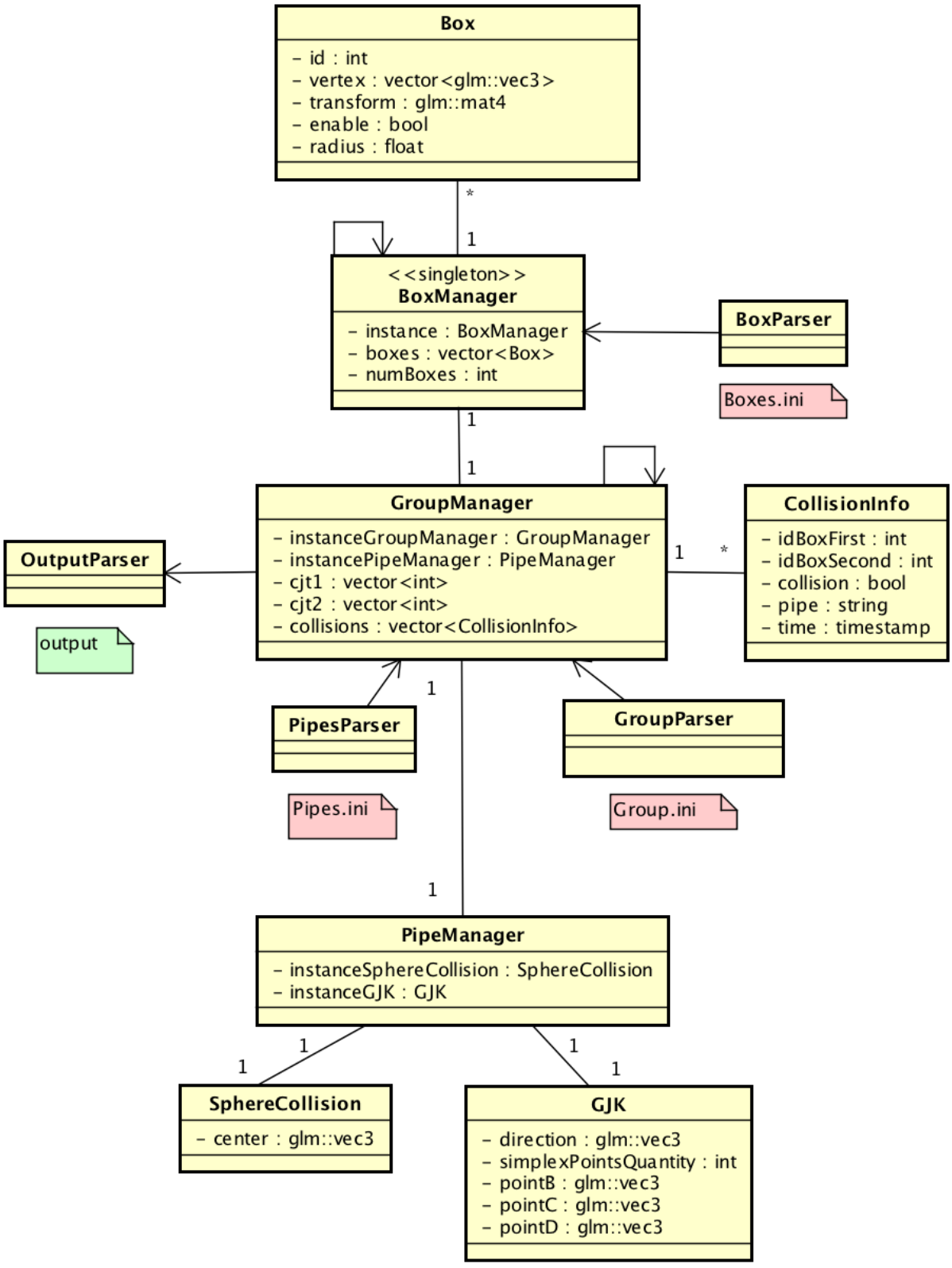


Figura 85: Model conceptual per al disseny orientat a objectes per GJK

9.3.3 Disseny orientat a dades GJK

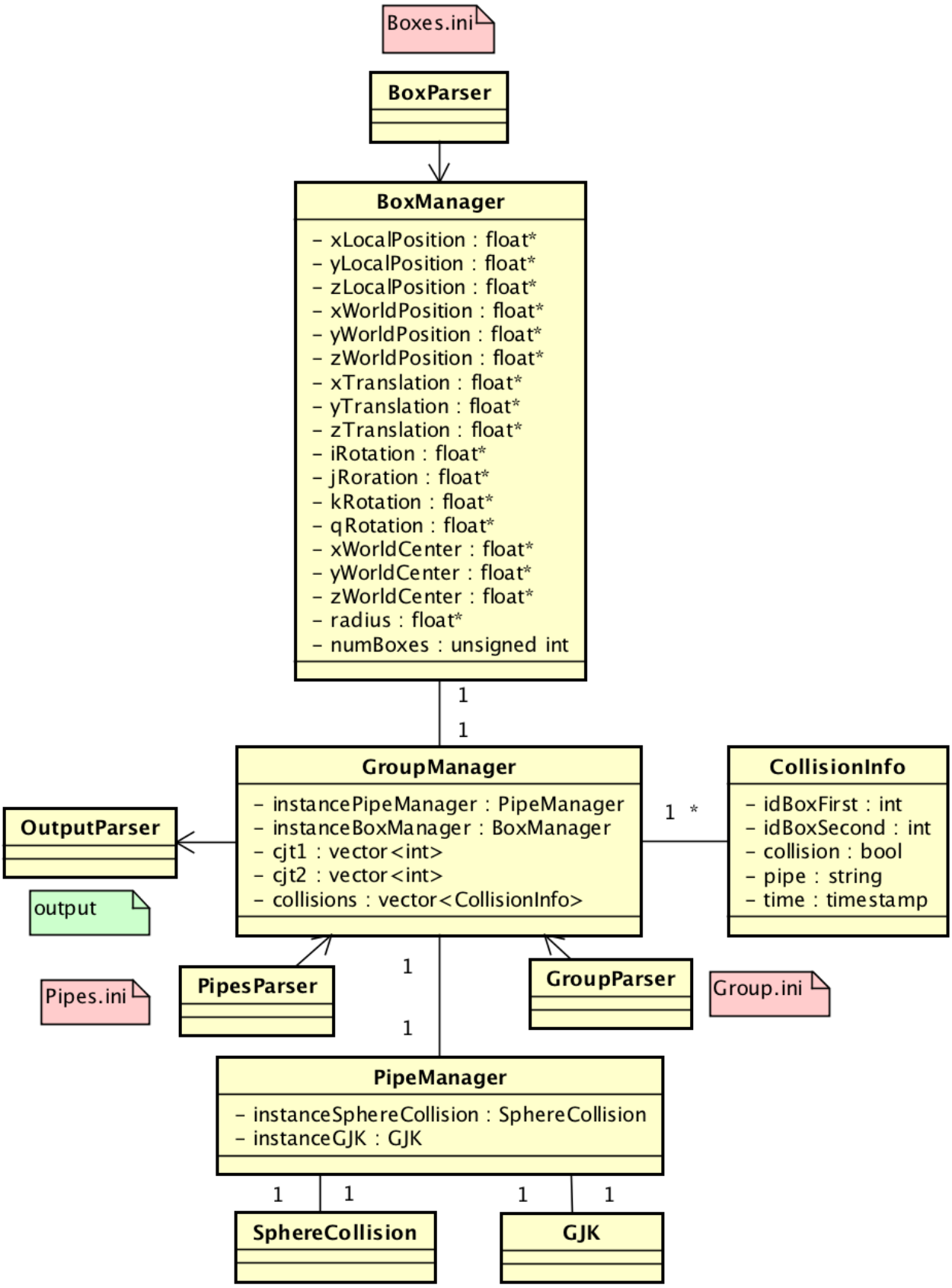


Figura 86: Model conceptual per al disseny orientat a dades per GJK

9.3.3.1 Descripció de les entitats

9.3.3.1.1 BoxManager

Entitat encarregada de realitzar la reserva de memòria, gestionar les dades de les diferents Box del motor i realitzar la destrucció dels diferents arrays de dades en acabar l'execució. Disposa d'una estructura d'arrays en memòria on s'emmagatzemen totes les dades dels objectes inserits al motor.

ATRIBUT	DESCRIPCIÓ
xLocalPosition	Array en memòria on emmagatzemar les components x dels vèrtex originals.
yLocalPosition	Array en memòria on emmagatzemar les components y dels vèrtex originals.
zLocalPosition	Array en memòria on emmagatzemar les components z dels vèrtex originals.
xWorldPosition	Array en memòria on emmagatzemar les components x dels vèrtex en món.
yWorldPosition	Array en memòria on emmagatzemar les components y dels vèrtex en món.
zWorldPosition	Array en memòria on emmagatzemar les components z dels vèrtex en món.
xTranslation	Array en memòria on emmagatzemar les components x de les translacions.
yTranslation	Array en memòria on emmagatzemar les components y de les translacions.
zTranslation	Array en memòria on emmagatzemar les components z de les translacions.
iRotation	Array en memòria on emmagatzemar les components i de les rotacions.
jRotation	Array en memòria on emmagatzemar les components j de les rotacions.
kRotation	Array en memòria on emmagatzemar les components k de les rotacions.
qRotation	Array en memòria on emmagatzemar les components q de les rotacions.
xWorldCenter	Array en memòria on emmagatzemar les components x dels centres dels objectes en món.
yWorldCenter	Array en memòria on emmagatzemar les components y dels centres dels objectes en món.
zWorldCenter	Array en memòria on emmagatzemar les components z dels centres dels objectes en món.
radius	Array en memòria on emmagatzemar els radis dels objectes.
numBoxes	Quantitat d'objectes disponibles

Taula 54: Descripció dels atributs de la classe Box Manager

10 Disseny

Per a realitzar l'estudi de les diferents tècniques de col·lisió s'han realitzat tres dissenys. Un primer disseny orientat a objectes que engloba les tres tècniques de detecció de col·lisions, un segon disseny orientat a objectes específic per a la tècnica de GJK i un tercer disseny orientat a dades per a la tècnica de GJK.

Tots tres dissenys havien de complir els requeriments de l'empresa, havien de ser adaptables i escalables, és a dir, poder permetre afegir noves tècniques o canviar l'ordre d'execució dels components. Per aquest motiu es va escollir fer un sistema dinàmic de pipes, ja que permetia la flexibilitat requerida.

Cadascuna de les tècniques consisteix en una seqüència de funcions. Cada funció rep uns valors d'entrada i produeix uns valors de sortida de manera incremental, és a dir, les dades produïdes per una funció poden ser utilitzats per una altra funció. La connexió entre la sortida d'una funció i l'entrada d'una altra s'anomena canonada (Pipe).

Durant l'apartat d'implementació es fa una explicació més detallada de les funcions i l'ordre que formen cadascuna de les pipes realitzades.

10.1 Arquitectura física

Per a realitzar l'estudi de les diferents tècniques i la implementació només ha sigut necessari fer ús d'un ordinador HP i7 amb 12 GB de RAM i una pantalla Samsung de 24" amb sistema operatiu Windows 10 que han sigut els encarregats de realitzar els càlculs.

Per a realitzar l'execució de cadascuna de les proves es rebien uns paràmetres d'entrada (input) i es produïa una sortida (output).

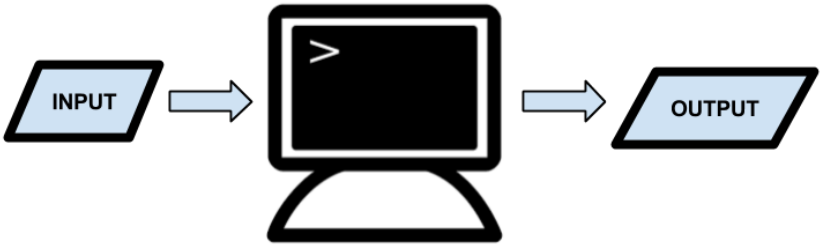


Figura 87: Arquitectura física

10.2 Arquitectura lògica

Per al disseny de l'arquitectura lògica s'ha pensat la manera d'assolir el requisit d'escalabilitat i manteniment del motor.

A continuació es descriuen els diferents components del motor per a cadascun dels dissenys.

10.2.1 Disseny orientat a objectes

Donat que el disseny orientat a objectes del GJK no deixa de ser una reducció del disseny orientat a objectes genèric s'ha aprofitat aquest apartat per realitzar l'explicació d'ambos dissenys.

10.2.1.1 Inputs i outputs

A continuació es descriuen les classes que permeten la gestió d'inputs i outputs del disseny orientat a objectes.

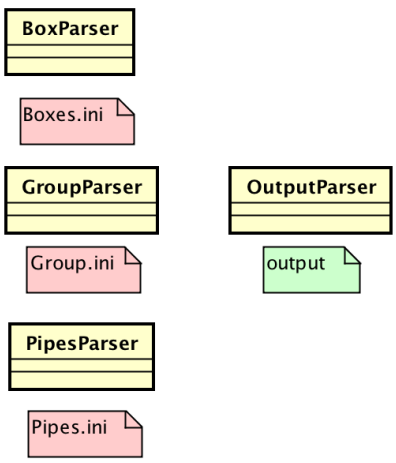


Figura 88: Inputs i outputs

S'han definit uns fitxers de configuració externs (.ini) amb la informació necessària per realitzar els càlculs. També s'ha definit un fitxer de sortida (output) amb la informació resultant de les col·lisions. Donat que l'entrada de dades i la sortida estan definides específicament per al codi actual i donat que en un futur pot arribar a modificar-se (per exemple, en comptes d'utilitzar fitxers de text s'utilitzin arxius en format json o xml) s'han creat uns Parsers encarregats d'adaptar les dades d'entrada i de sortida als valors necessaris segons correspongui.

Boxes.ini és el fitxer que conté la informació de creació dels objectes que es volen inserir al motor. Per cada objecte s'ha definit la seva posició, el seu escalat i la seva rotació.

Group.ini és el fitxer que conté la informació de quin sistema de càlcul es vol fer servir. Es pot escollir entre els dos casos d'ús definits, realitzar les col·lisions entre tots els objectes actius en el motor o definir dos conjunts d'objectes (mitjançant els seus identificadors) i realitzar el càlcul entre els objectes del primer conjunt contra els objectes del segon conjunt.

Pipes.ini és el fitxer que conté quina pipe es vol executar per a realitzar el càlcul de la detecció de col·lisions.

Output és el fitxer que conté la informació de sortida de l'execució. Per a cada parella d'objectes de la qual s'ha realitzat la comprovació de col·lisió i aquesta ha donat positiva es retorna la parella d'identificadors dels objectes, el temps que s'ha trigat a realitzar el càlcul.

10.2.1.2 Managers

A continuació es descriuen les classes Manager del disseny orientat a objectes.

Donat que un dels requisits del projecte era permetre un disseny escalable, s'han creat tres managers que són els encarregats de gestionar els diferents objectes, sistemes de càlcul i pipes. En el cas que durant el projecte s'afegissin noves tècniques, pipes i tipologies de càlcul es pogués seguir gestionant amb facilitat.

BoxManager és l'encarregat de gestionar els objectes dins del motor, permet les insercions, les modificacions i les eliminacions. A partir de la informació que rep del

BoxParser realitza les funcions de addBox, updateBox o removeBox. Disposa d'un vector de boxes amb les instàncies on s'emmagatzemen els diferents objectes.

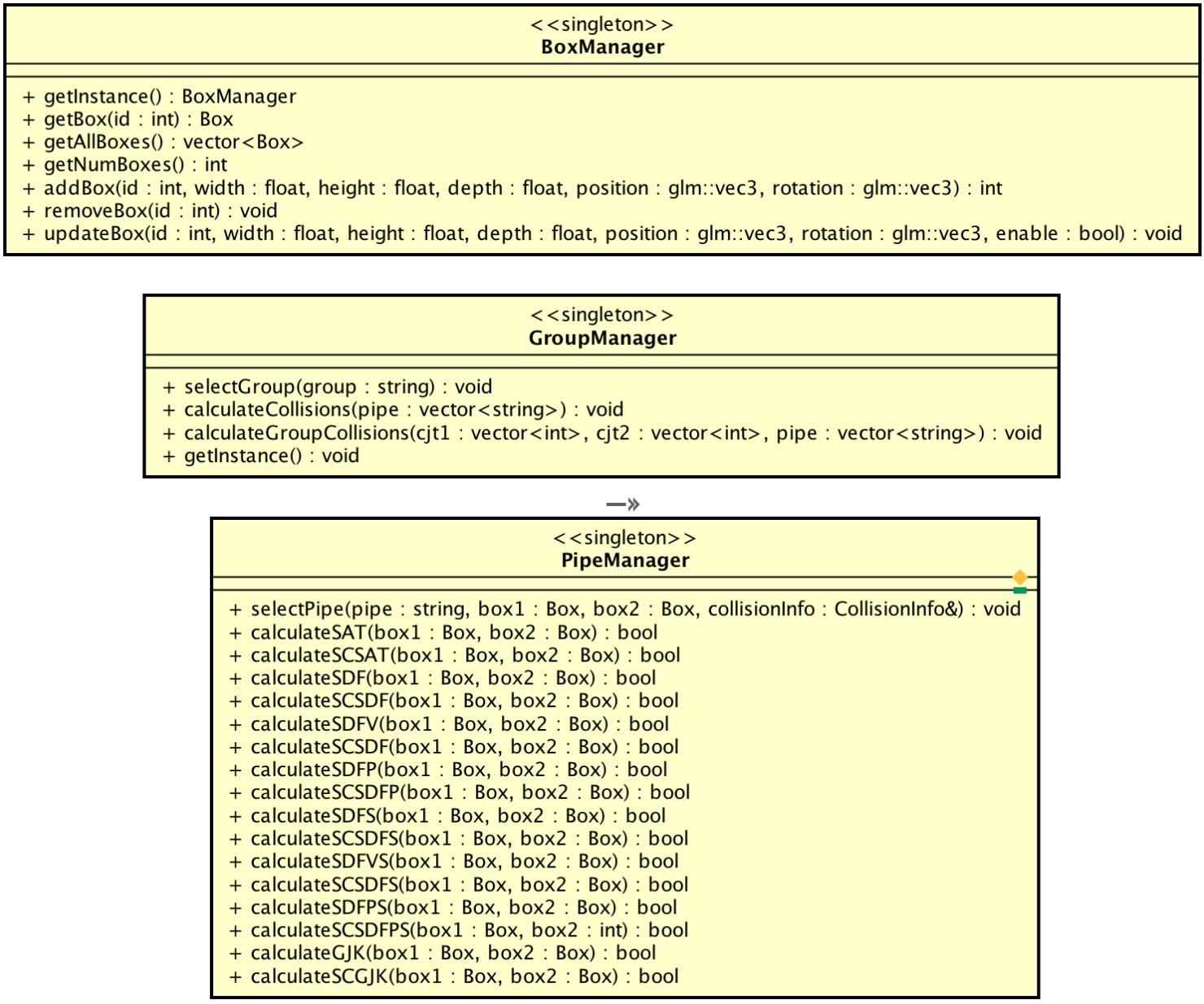


Figura 89: Managers

GroupManager és l'encarregat de gestionar quin tipus de càlcul es vol utilitzar. A partir de la informació que rep del GroupParser gestiona quin tipus de càlcul s'ha de fer. Es pot triar entre calcular totes les col·lisions que hi hagi entre tots els objectes inserits al motor o definir dos conjunts d'objectes (a partir dels seus identificadors) i calcular les col·lisions que hi ha entre els objectes del primer conjunt contra els objectes del segon conjunt.

PipeManager és l'encarregat de gestionar quina pipe s'utilitzarà per realitzar el càlcul de les col·lisions. A partir de la informació que rep del PipeParser gestiona quina pipe s'ha d'executar. Es pot triar entre les 16 pipes següents:

Pipe 1: Col·lisió d'esferes + SAT

Pipe 2: SAT

Pipe 3: Col·lisió d'esferes + SDF complet amb control de sortida

Pipe 4: SDF complet amb control de sortida

Pipe 5: Col·lisió d'esferes + SDF complet

Pipe 6: SDF complet

Pipe 7: Col·lisió d'esferes + SDF vèrtex amb control de sortida

Pipe 8: SDF per vèrtex amb control de sortida

Pipe 9: Col·lisió d'esferes + SDF per vèrtex

Pipe 10: SDF per vèrtex

Pipe 11: Col·lisió d'esferes + SDF per punts de mostreig amb control de sortida

Pipe 12: SDF per punts de mostreig amb control de sortida

Pipe 13: Col·lisió d'esferes + SDF per punts de mostreig

Pipe 14: SDF per punts de mostreig

Pipe 15: Col·lisió d'esferes + GJK

Pipe 16: GJK

10.2.1.3 Lògica

A continuació es descriuen les classes que formen la part de la lògica del disseny orientat a objectes.

Box és l'encarregada d'emmagatzemar tota la informació d'un objecte. Per cada objecte que s'insereixi al motor es crearà una instància amb les seves dades.

CollisionInfo és l'encarregada d'emmagatzemar la informació de la col·lisió que es calcula en aquell moment. Per a cada parella d'objectes es crearà una instància d'aquesta classe per emmagatzemar les dades de la comprovació de la col·lisió.

SphereCollision és l'encarregada de realitzar el càlcul de col·lisió d'esferes.

SAT és l'encarregada de calcular les col·lisions entre dos objectes fent servir la tècnica del Separated Axis Theorem. Els seus atributs són utilitzats per emmagatzemar els càlculs intermedis necessaris per assolir el veredicte de la col·lisió.

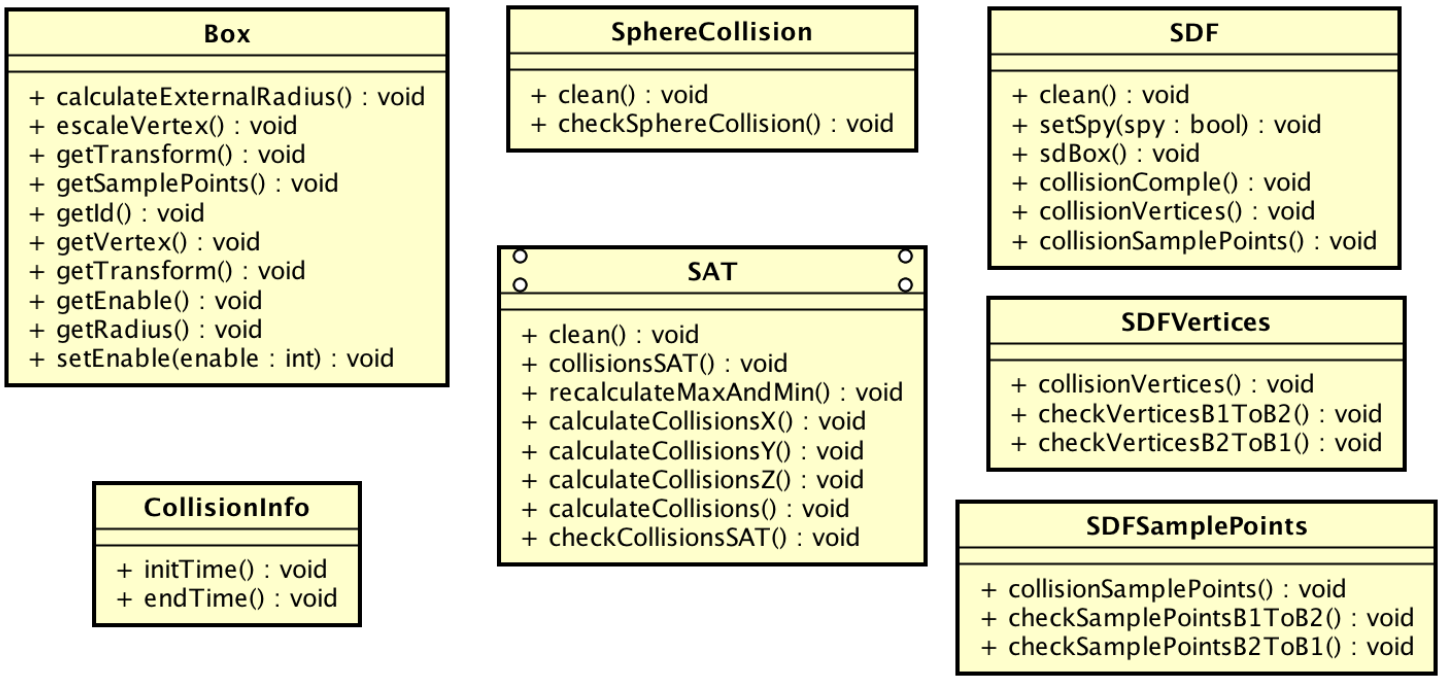


Figura 90: Lògica

SDF és l'encarregada de calcular les col·lisions entre dos objectes fent servir la tècnica del Signed Distance Field en la versió completa, és a dir, realitzant el càlcul tant per vèrtex com per punts de mostreig. Rep de la classe SDFVertex la informació del càlcul de la tècnica SDF per vèrtex i de la classe SDFSamplePoints la informació del càlcul de la tècnica SDF per punts de mostreig. Els seus atributs són utilitzats per emmagatzemar els càlculs intermedis necessaris per assolir el veredicte de la col·lisió.

SDFVertex és l'encarregada de calcular les col·lisions entre dos objectes fent servir la tècnica de Signed Distance Field en la versió per vèrtex.

SDFSamplePoints és l'encarregada de calcular les col·lisions entre dos objectes fent servir la tècnica de Signed Distance Field en la versió per punts de mostreig.

GJK és l'encarregada de calcular les col·lisions entre dos objectes fent servir la tècnica del Gilbert-Johnson-Keerthi. Els seus atributs són utilitzats per emmagatzemar els càlculs intermedis necessaris per assolir el veredicte de la col·lisió.

A *l'Annex 3* s'ha inclòs l'especificació de les funcions del GJK.

10.2.2 Disseny orientat a dades GJK

El disseny orientat a objectes i el disseny orientat a dades es poden diferenciar en els següents tres punts:

- Les dades d'entrada
- La manera en la qual s'estructura el codi
- La manera en la qual s'emmagatzemen les dades

10.2.2.1 Les dades d'entrada

Durant aquesta part del projecte s'ha aplicat un canvi important quant a les dades amb les qual es realitzarien els càlculs. Pels motius explicats a l'apartat de teoria "*7.2.4-Quaternions*", en comptes de fer ús d'una matriu de transformació, s'ha decidit fer ús d'un vector per definir la translació i un quaternió per definir la rotació. L'escalat s'ha seguit aplicant als vèrtexs i d'aquesta manera es pot seguir aplicant un escalat diferent a cada eix.

10.2.2.2 Com s'estructura el codi

En un disseny orientat a dades no hi ha objectes, ni atributs públics ni privats.

De la mateixa manera que en el disseny orientat a objectes les dades d'entrada passen pel BoxParser i arriben al BoxManager, en el disseny orientat a dades en comptes de crear-se les diferents instàncies de Box, el manager disposa d'una estructura d'arrays on emmagatzema les dades en memòria.

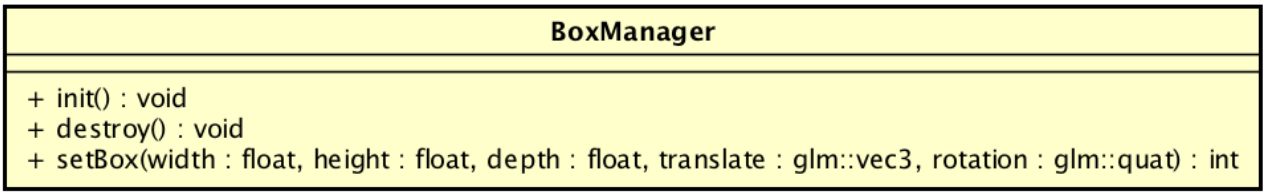


Figura 91: Funcions del BoxManager

A més de realitzar la gestió de les dades, aquest manager és l'encarregat de realitzar la reserva i la destrucció en memòria dels arrays.

Per separar els atributs públics dels privats s'han fet servir dos namespace diferents que permeten mantenir la jerarquia d'atributs. *GJK* és el namespace que inclou tota la classe i *Detail* inclou els atributs privats i l'estructura d'arrays. D'aquesta manera incloent el namespace *Detail* és possible accedir als valors de l'estructura.

10.2.2.3 Com s'emmagatzemen les dades

Donat que les instàncies de Box també han desaparegut, en el seu lloc s'ha creat una estructura d'arrays de dades on s'emmagatzemen totes les dades, tant les inicials (vèrtex originals, translació i rotació) com les dades necessàries per a realitzar les pipes del GJK, és a dir, els vèrtexs i els centres transformats en món.

En total s'han creat 18 arrays donat que els diferents atributs s'han descompost fins al mínim element. És a dir, per a un vector (x,y,z) s'han creat 3 arrays de dades, un per cadascuna de les components. I en el cas d'un quaternió (i,j,k,q) s'han creat 4 arrays.

10.2.3 Diagrames de seqüència

En aquest apartat es presenten dos diagrames de seqüència corresponents als dos casos d'ús del sistema. S'hi explica tot el procés des que la petició arriba al motor fins que s'envia la informació de les col·lisions a l'usuari.

S'han triat els casos d'ús "Calcular col·lisions" i "Calcular col·lisions per grups", ja que amb aquests diagrames de seqüència es mostren dos casos diferents. La resta de diagrames de seqüència no es reprodueixen en aquesta memòria, donat que són molt similars i no aporten més informació sobre el funcionament.

10.2.3.1 Cas d'ús: Calcular col·lisions

Aquest cas tracta d'una petició al motor. El treballador d'Interiorvista indica que vol calcular totes les col·lisions entre els objectes actius del motor i indica quina pipe vol que es faci servir per a realitzar el càlcul. El GroupManager rep la petició del treballador i s'encarrega d'obtenir els objectes i d'iterar sobre ells per crear les diferents parelles d'objectes. Per cada parella es comunica amb el PipeManager per gestionar l'execució de la pipe indicada.

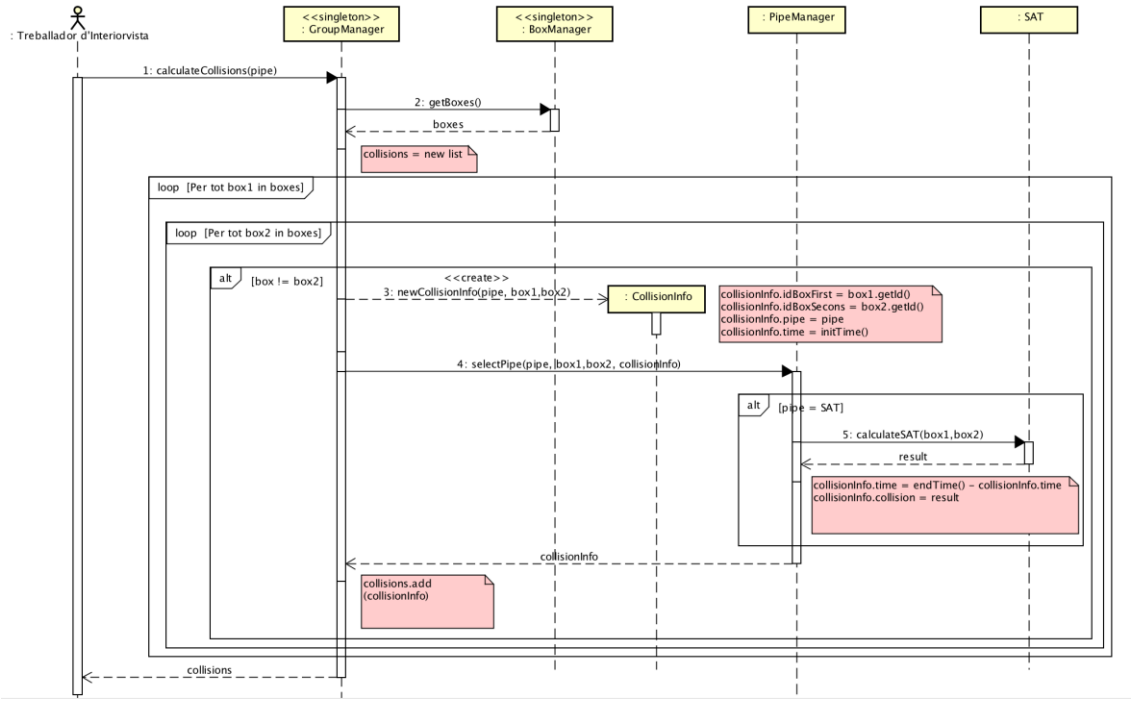


Figura 92: Diagrama de seqüència del cas d'ús Calcular col·lisions

10.2.3.1 Cas d'ús: Calcular col·lisions per grups

Aquest cas tracta d'una petició al motor. El treballador d'Interiorvista defineix entre quins objectes vol realitzar el càlcul. Indica que vol calcular totes les col·lisions entre els objectes definits en els dos conjunts i quina pipe vol que es faci servir per a realitzar el càlcul. El GroupManager rep la petició del treballador i s'encarrega d'obtenir els objectes i d'iterar sobre ells per crear les diferents parelles d'objectes. Per cada parella es comunica amb el PipeManager per gestionar l'execució de la pipe indicada.

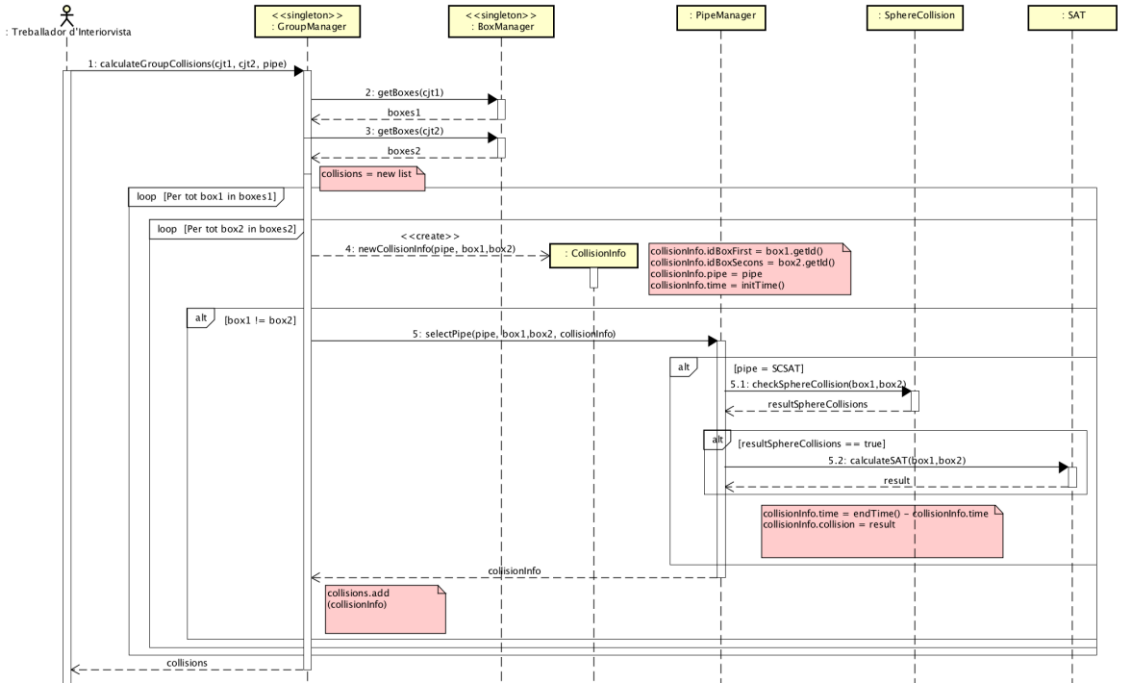


Figura 93: Diagrama de seqüència del cas d'ús Calcular col·lisions per grups

A *l'Annex 4* està inclosa una versió dels diagrames de seqüència més ampliada.

11 Implementació

11.1 Elecció de les tecnologies

L'entorn de desenvolupament del framework que vol implementar l'empresa està basat en C++. Així doncs l'elecció del llenguatge no ha estat decisió d'aquest projecte.

Respecte a les eines de desenvolupament s'ha utilitzat Visual Studio 2015 i el control de versions Bitbucket, ambdós sistemes formen part de l'estratègia global de l'empresa i s'ha seguit amb el seu ús. S'ha emprat la llibreria OpenGL Mathematics (GLM) una llibreria que proporciona classes i funcions matemàtiques en C++. S'ha escollit perquè és una eina molt completa, dóna molt bons resultats per a programes de simulació de físiques com és el nostre cas. A més a més és molt fàcil d'incorporar al projecte (es tracta d'una llibreria headerOnly) i de fer-la servir.

Com a eines de comunicació entre el director i amb els diferents consultors del projecte, s'ha emprat Skype per realitzar reunions de seguiment a distància, el Trello per organitzar les diferents tasques i sprints, el Google Drive per redactar els informes de seguiment i el mail de Google per enviar informes, gestionar reunions o aclarir dubtes.

Per dur a terme la planificació del projecte s'ha fet ús de l'eina Gantter.
Per al disseny dels diagrames s'ha fet ús del software Astah Professional.
Per al disseny de les imatges s'ha fet ús del de l'eina Google Drawings i amb draw.io Diagrams.

11.2 Representació dels objectes

S'ha definit que per representar un objecte són necessaris els seus vèrtexs i la informació de translació, rotació i escalat.

Inicialment es va decidir treballar amb els vèrtexs i la matriu de transformació. Com s'ha explicat en el capítol de teoria una matriu de transformació permet emmagatzemar la informació de translació, rotació i esclat necessaris per representar un objecte.
Cap a l'última part del projecte s'ha fet un canvi a l'estructura de dades. Com que un dels requisits d'aquest projecte consistia a obtenir els càlculs de les col·lisions de la

manera més eficient possible s'ha volgut provar si l'ús de quaternions reduïa el cost del còmput envers l'ús de matrius de transformació. S'ha decidit fer ús de quaternions per representar la rotació dels objectes, ja que el còmput de multiplicació de quaternions és més eficient que la multiplicació de matrius.
El problema que hi ha en l'ús de quaternions és que aquests no permeten poder definir un escalat diferent per cadascun dels eixos (on una matriu de transformació sí que ho permet). Per aquest motiu, ha estat necessari traspassar la informació d'escalat de l'objecte als vèrtexs d'aquest.

Així doncs en un espai 2D els vèrtexs originals dels objectes es defineixen de la següent manera:

- 4 vèrtexs (vèrtexs originals VO)
- Rectangle centrat a l'origen de coordenades (els eixos de món com els eixos locals estan alineats i centrats a l'origen).

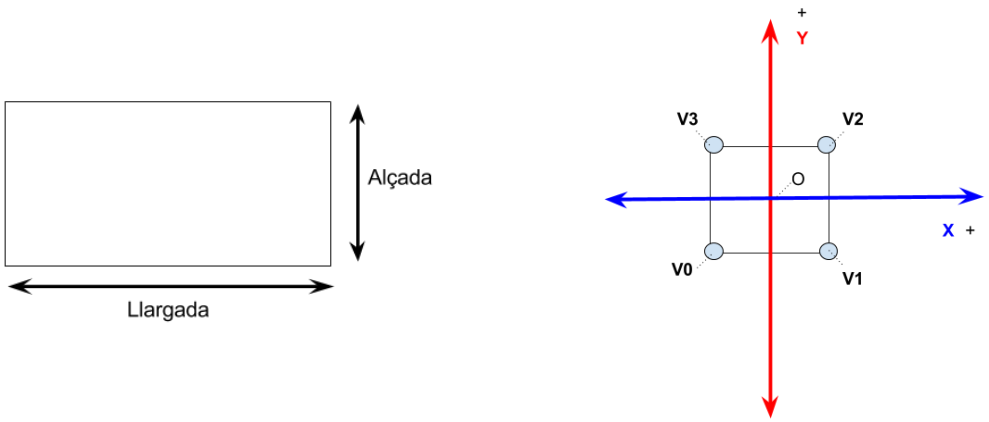


Figura 94: Definició dels objectes en 2D

V0 ⇒ (-llargada/2, -alçada/2)
V1 ⇒ (llargada/2, -alçada/2)
V2 ⇒ (llargada/2, alçada/2)
V3 ⇒ (-llargada/2, alçada/2)

De la mateixa manera en un espai 3D els vèrtexs originals dels objectes es defineixen de la següent manera:

- 8 vèrtexs (vèrtexs originals VO)
- Cub centrat a l'origen de coordenades (els eixos de món com els eixos locals estan alineats i centrats a l'origen).

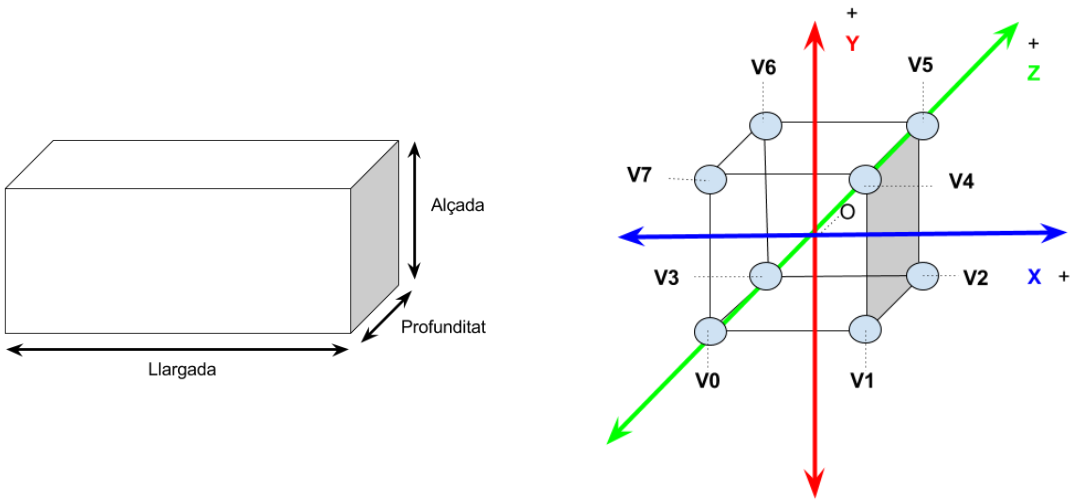


Figura 95: Definició dels objectes en 3D

V0 ⇒ (-llargada/2, -alçada/2, -profunditat/2)
V1 ⇒ (llargada/2, -alçada/2, -profunditat/2)
V2 ⇒ (llargada/2, alçada/2, -profunditat/2)
V3 ⇒ (-llargada/2, alçada/2, -profunditat/2)
V4 ⇒ (-llargada/2, -alçada/2, profunditat/2)
V5 ⇒ (llargada/2, -alçada/2, profunditat/2)
V6 ⇒ (llargada/2, alçada/2, profunditat/2)
V7 ⇒ (-llargada/2, alçada/2, profunditat/2)

En aplicar en el vèrtex la informació d'escalat, la matriu de transformació només contenia la informació de rotació i translació.

La representació final dels objectes es realitza mitjançant els vèrtexs escalats, un vector de translació i un quaternió.

11.3 Col·lisió d'esferes[39]

En aquest apartat s'explica una de les optimitzacions que s'ha aplicat en el projecte.

A l'apartat **"***7.1-Els motors de col·lisions***"** s'ha explicat que existeixen dues fases dins de la detecció de col·lisions, *Broad Phase* i la *Narrow Phase* i que la primera consistia en una optimització per agilitzar els càlculs del motor de col·lisions, ja que aplicava mètodes per filtrar els objectes que col·lisionaven.

En comptes de realitzar la comprovació amb totes les parelles d'objectes, primer de tot s'ha aplicat la tècnica de col·lisió d'esferes per eliminar aquelles parelles que no superin la comprovació, o dit d'una altra manera, només passaran a la *Narrow Phase* aquelles parelles d'objectes que hagin superat aquesta prova.

11.3.1 Les Bounding Volume esfèriques

Al mateix capítol s'ha parlat dels diferents tipus de *Bounding Volume* que existeixen. Per realitzar aquesta optimització s'ha decidit fer ús de les Bounding Volume esfèriques, ja que el càlcul per comprovar si duess esferes col·lisionen és molt més eficient que amb els altres volums.

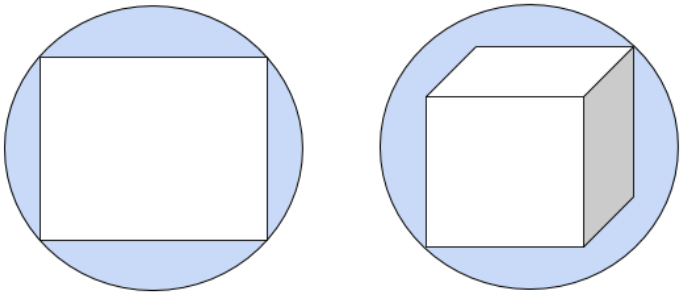


Figura 96: Bounding Volume esfèriques per 2D i 3D

11.3.2 Com es realitza la criba d'objectes

Per a realitzar la criba, per cada parella d'objectes es mira si les seves bounding volum esfèriques intersecciones o no. Es poden donar tres casos:

11.3.2.1 Cas 1: Les esferes no interseccionen

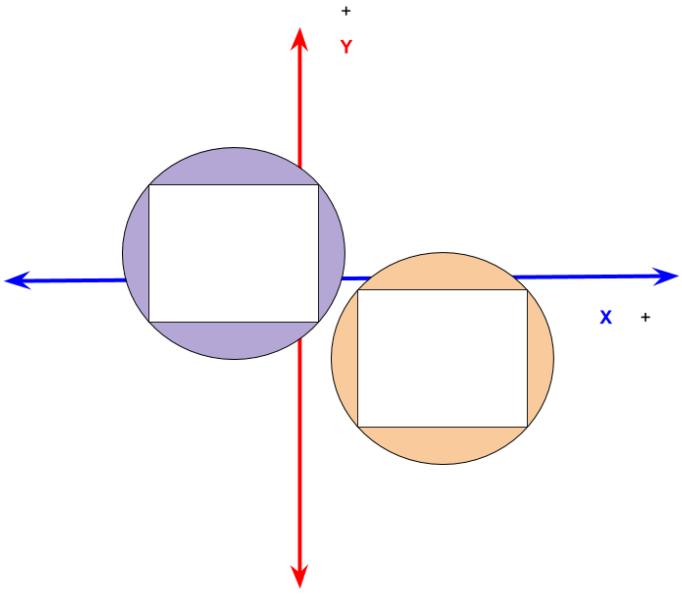


Figura 97: Exemple de no intersecció per part de les Bounding Volum esfèriques

En aquesta situació aquesta parella no passaria a la següent fase de detecció de la col·lisió, ja que és impossible que els objectes interiors intersecccionen si les Bounding Volume esfèriques no ho fan.

11.3.2.2 Cas 2: Les esferes intersecccionen però els objectes no

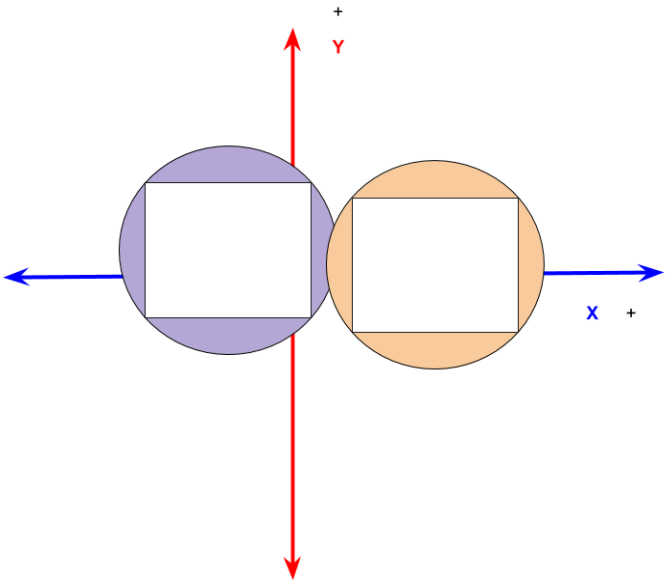


Figura 98: Exemple d'intersecció per part de les Bounding Volum esfèriques

En aquesta situació aquesta parella sí que passaria a la següent fase de detecció de la col·lisió, ja que les Bounding Volume esfèriques interseccionen i "és possible" que els objectes interiors també interseccionen. En aquest cas la *Narrow Phase* ha de detectar que no es produeix intersecció entre els objectes.

11.3.2.3 Cas 3: Les esferes i els objectes interseccionen

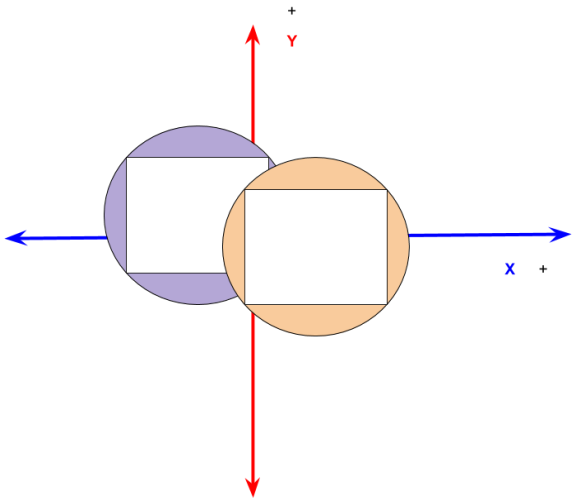


Figura 99: Exemple d'intersecció entre les bounding volums esfèriques i dels objectes interiors

En aquesta situació aquesta parella també passaria a la següent fase de detecció de la col·lisió, ja que les Bounding Volume esfèriques interseccionen i "és possible" que els objectes interiors també ho facin. En aquest cas la Narrow Phase ha de detectar que es produeix intersecció entre els objectes.

11.4 Separation Axis Therorem

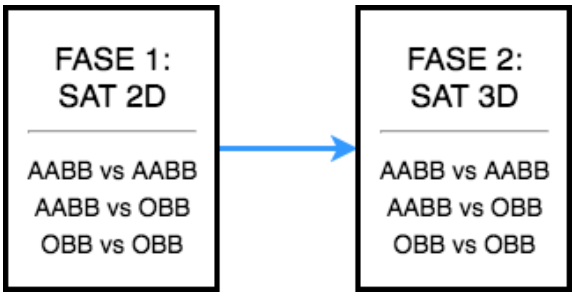
Un cop realitzat l'estudi del Separation Axis Therorem s'ha dut a terme la seva implementació.

Arribats a aquest punt s'ha decidit d'ur a terme una optimització de la tècnica, ja que en el cas de treballar amb objectes 3D i que aquests tinguin la particularitat de ser OBB s'ha vist a l'apartat de teoria que serien necessàries 15 comprovacions per parella d'objectes.
Per aquest motiu s'ha preferit adaptar la tècnica per a comprovar només 3 eixos. Un altre dels motius pels quals s'ha optat per aquesta via es per poder aprofitar la manera en què l'empresa disposa de la informació dels objectes, és a dir, a partir dels seus vèrtexs i les seves matrius de transformació.

S'ha preferit fer ús d'aquestes dades per transformar els vèrtexs dels objectes a un mateix espai i projectar-los sobre els 3 eixos de coordenades. D'aquesta manera tot i que hi ha un procés entremig de transformació de vèrtex, el nombre de comprovacions s'ha reduït a un 20%.

11.4.1 Fases de la implementació

La implementació de l'optimització de SAT es va realitzar en 2 fases:



11.4.1.1 FASE 1: Aplicació de SAT en 2D

En aquesta primera fase es va començar a treballar amb la tècnica del SAT en un espai 2D. Per comprendre el seu funcionament i com es realitzaria la implementació es van fer proves amb el cas més senzill de calcular, dos objectes AABB.

Posteriorment es va afegir la complicació de treballar amb objectes girats, OBB. Això va afectar en la implementació feta per als objectes alineats, ja que el càlcul de màxim i mínims és necessari fer-lo un cop s'han transformats els vèrtexs.

11.4.1.2 FASE 2: Aplicació de SAT 3D

Un cop realitzada la implementació per a l'espai de 2D es va procedir amb l'espai de 3D.

Es va dur a terme el mateix procediment. Una primera etapa de treball amb objectes AABB i una segona etapa amb objecte OBB.

11.4.2 Dades que necessita la tècnica

Per a poder realitzar el trasllat als eixos d'un objecte determinat és necessari disposar de cada objecte els seus vèrtexs originals (escalats) i les seves matrius de transformació (amb la informació de translació i escalat).

11.4.3 Procés de la tècnica

- Traslladar B2 als eixos locals de B1
- Calcular màxims i mínims
- Comprovar eixos

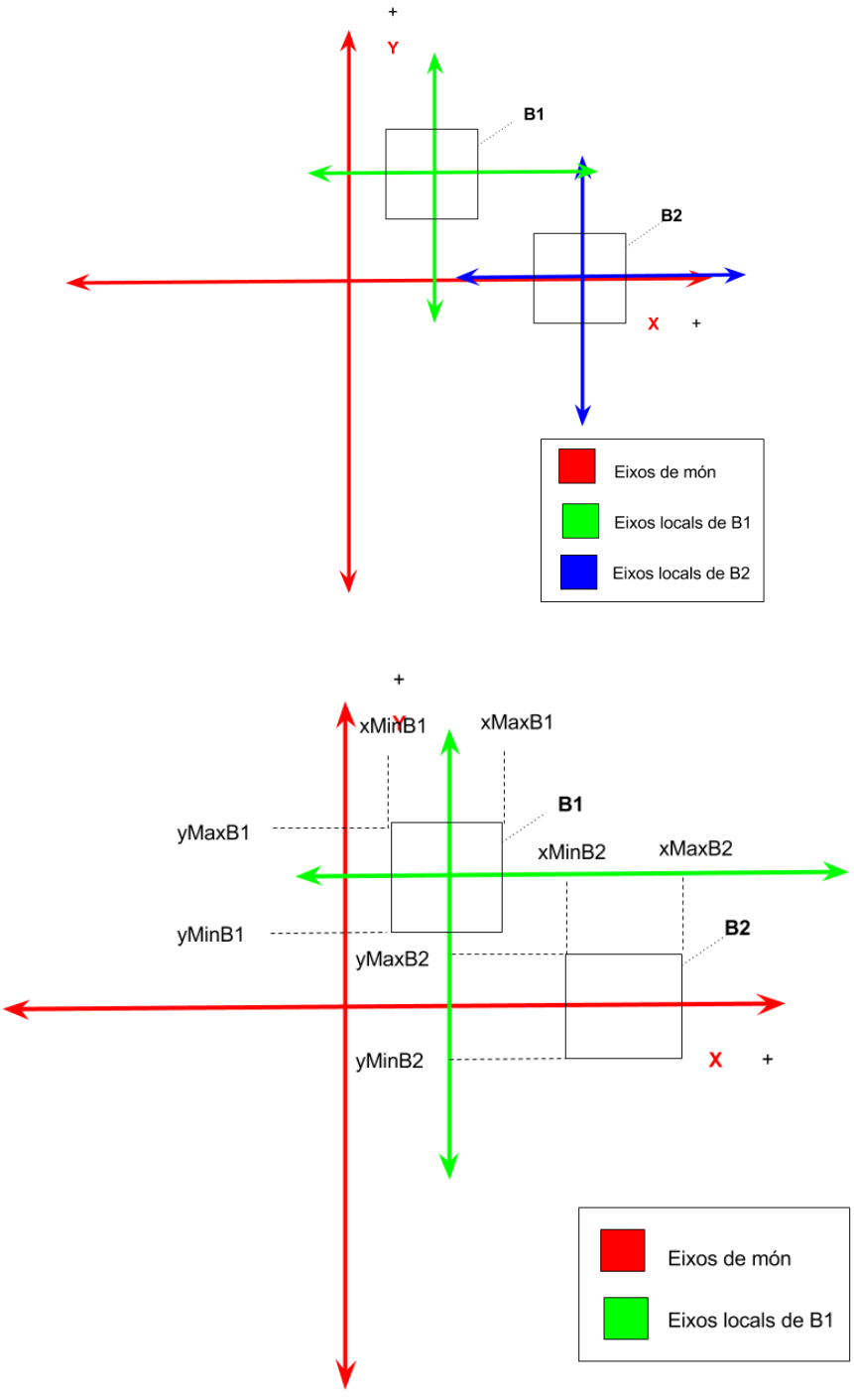


Figura 100: Exemple de càlcul AABB vs AABB 2D

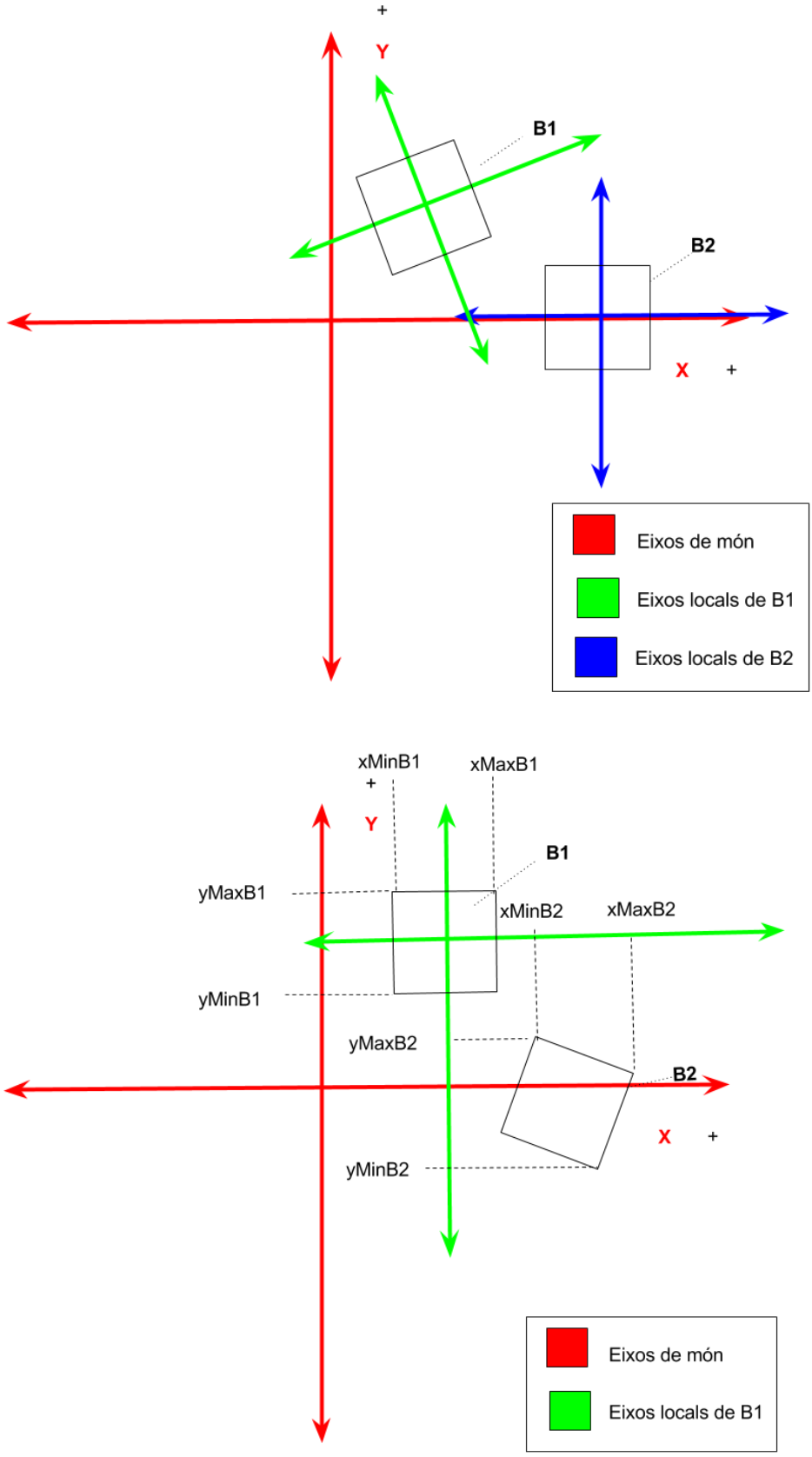


Figura 101: Exemple de càlcul AABB vs OBB 2D

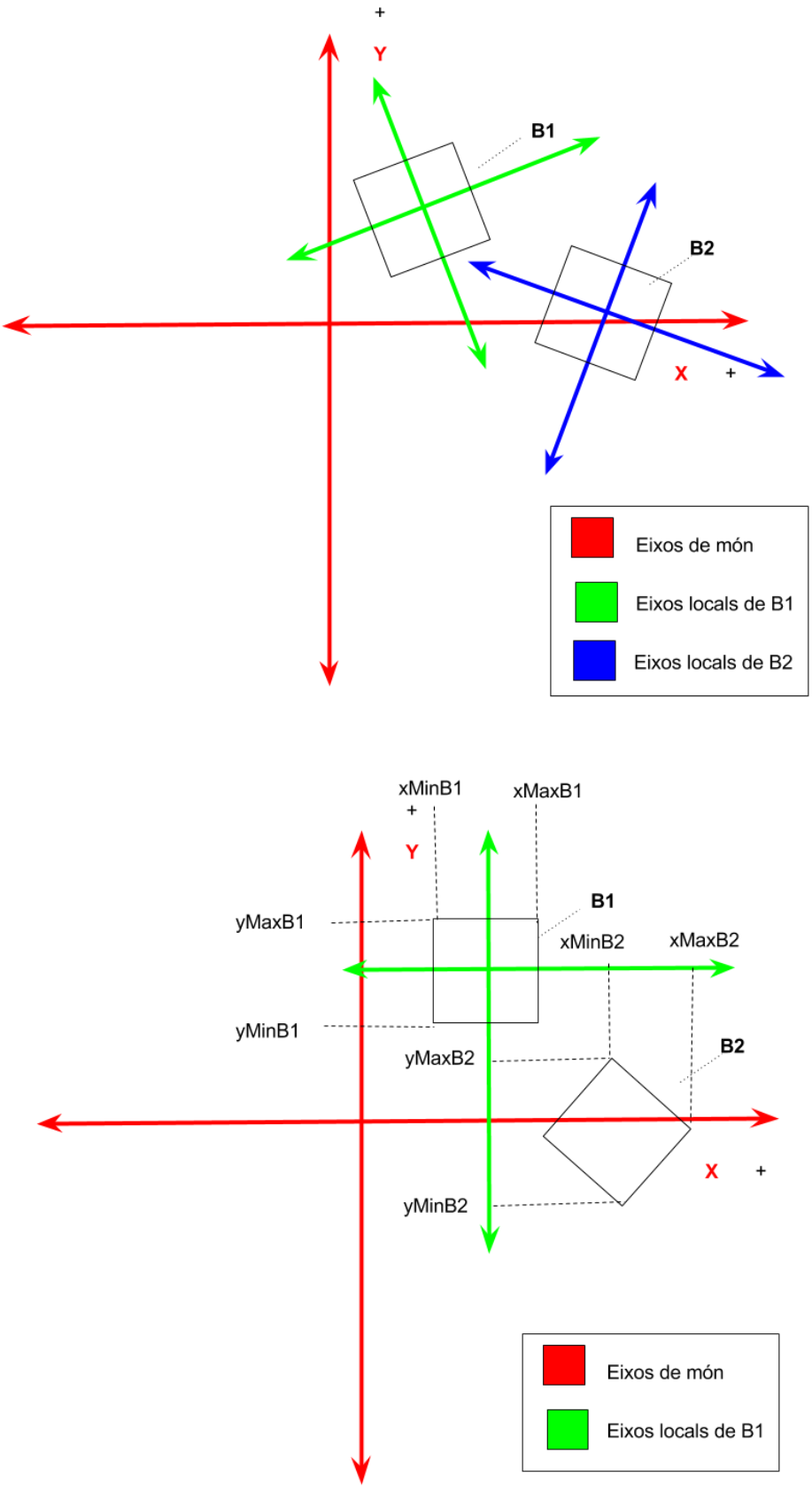


Figura 102: Exemple de càlcul OBB vs OBB 2D

11.4.3.1 Traslladar B2 als eixos locals de B1

Per poder comprovar si hi ha intersecció entre les projeccions cal que ambdós objectes estiguin situats respecte als mateixos eixos de coordenades. S'ha escollit traslladar l'objecte B2 als eixos locals de B1 però una altra opció vàlida seria traslladar l'objecte B1 als eixos locals de B2.

Per dur a terme el trasllat és necessari fer dos productes de matrius:

$$VMB2 = MTB2 \times VOB2$$
$$VTB2 = MTB1^{-1} \times VMB2$$

Primer de tot cal aplicar sobre els vèrtexs originals de l'objecte B2 (VOB2) la informació de rotació i translació continguda a la matriu de transformació de B2 (MTB2). Aquests vèrtexs nous (VMB2) estan en posició de món, és a dir, estan referenciats respecte als eixos de món.

Sobre els nous vèrtex de món de B2 (VMB2) cal aplicar les transformacions que pugui afectar l'objecte B1. Per fer-ho és necessari multiplicar els vèrtexs per la inversa de la matriu de transformació de B1 ($MTB1^{-1}$). D'aquesta manera s'obtenen els vèrtexs de B2 respecte als eixos de coordenades de B1.

11.4.3.2 Calcular màxim i mínim

Un cop estan els dos objectes respecte als mateixos eixos de coordenades, és necessari conèixer el vèrtex màxim i mínim de cada objecte. Això servirà per calcular la projecció de cadascun dels objectes respecte a cada eix de coordenades de món.

Per fer-ho cal recórrer tots el vèrtex i emmagatzemar quins són els valors màxims i mínims de cadascuna de les coordenades.

11.4.3.3 Comprovar eixos

Finalment cal comprovar per cadascun dels eixos de coordenades si existeix superposició entre els dos objectes. En cas que en un dels eixos no existeixi solapament, no hi ha col·lisió entre els objectes.

11.4.4 Aplicació del disseny de pipes

A l'apartat "*10-Disseny*" s'ha parlat que es faria ús d'un disseny dinàmic de pipes.
Per a la tècnica de SAT s'han creat 2 pipes:

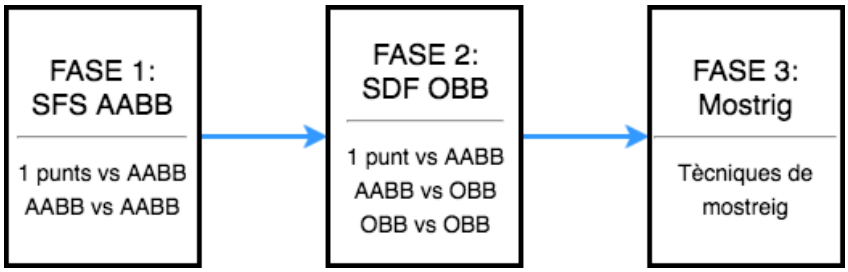
- **Pipe 1:** Col·lisió d'esferes + SAT
- **Pipe 2:** SAT

11.5 Signed Distance Field

Un cop realitzat l'estudi del Signed Distance Field s'ha dut a terme la seva implementació.

11.5.1 Fases de la implementació

La implementació de la tècnica SDF s'ha realitzat en 3 fases:



11.5.1.1 FASE 1: SDF amb AABB

En aquesta primera fase es va començar a treballar amb la tècnica del SDF. Per conèixer la tècnica es van fer unes primeres proves per veure el seu funcionament en el cas més simple: agafar un punt qualsevol en l'espai i calcular contra un objecte AABB.

Un cop es va comprendre el seu funcionament es va fer l'evolució de treballar amb els vèrtexs d'un objecte AABB contra un altre objecte AABB.

11.5.1.2 FASE 2: SDF amb OBB

En aquesta segona fase es van fer proves amb objectes OBB.

Per veure quins passos calia fer per poder treballar amb objectes OBB es va seguir el mateix procediment que a la fase anterior. Primer es van fer unes proves agafant punts qualsevols en l'espai i calculant contra un objecte OBB.

Un cop es va comprendre el seu funcionament es va fer l'evolució de treballar amb els vèrtexs d'un objecte AABB contra un altre objecte AABB i finalment es va fer el procés de treballar amb dos objectes OBB.

11.5.1.3 FASE 3: Tècniques de mostreig

En el moment de testejar la implementació de SDF per a vèrtex es va veure que hi havia casos en els quals tot i que els objectes estaven col·lisionant la tècnica no ho detectava pel fet que cap dels vèrtexs dels objectes col·lisionava amb un objecte.

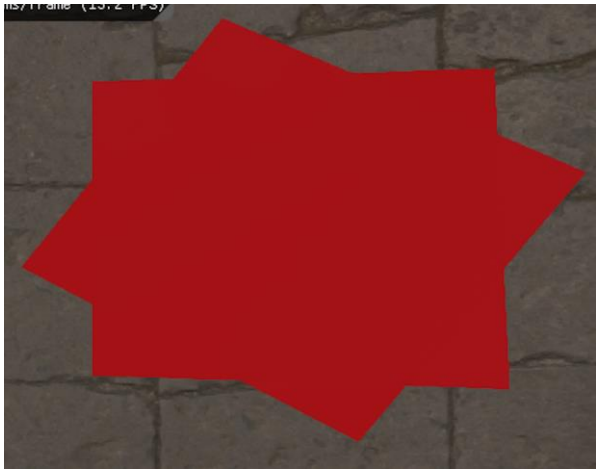


Figura 103:Exemple de la necessitat del mostreig. Cap dels vèrtex dels objectes intersecciona

Per resoldre aquests problemes es va decidir fer una segona variant de la tècnica utilitzant punts de mostreig dels diferents objectes com a punts per al càlcul.

Es van fer vàries proves variant el nombre de mostres a calcular per cada objecte, entre 500 i 5000 punts. Hi ha una relació directa entre el temps que triga l'algorisme a calcular el resultat i el nombre de punts de mostreig que es calculin, així doncs finalment es va decidir utilitzar 1000 punts donat que permetia seguir realitzant el càlcul però no es perjudicava tant en el temps d'execució.

11.5.2 Dades que necessita la tècnica

Per a poder realitzar el trasllat als eixos d'un objecte determinat és necessari disposar de cada objecte els seus vèrtexs originals (escalats) i les seves matrius de transformació (amb la informació de translació i escalat).

11.5.3 Procés de la tècnica

Inicialment es va començar a treballar amb el SDF per vèrtex, un cop es van implementar les seves pipes i es van realitzar els tests es va comprovar que aquestes pipes tenien limitacions si la zona de col·lisió no englobava cap dels vèrtexs dels objectes.

Per millorar els resultats es va fer una prova on s'executés la tècnica de SDF on en comptes d'utilitzar els vèrtexs dels objectes realitzés els càlculs a partir d'uns punts aleatoris de l'objecte. Així doncs es va acordar utilitzar 1000 punts de mostreig com a mesura òptima i que no afectes tant el temps de càlcul.

A partir d'aquestes dues tipologies de calcular el SDF es va realitzar una tercera opció: fer una solució que inclogués ambdós càlculs. És a dir, primer realitzés el càlcul per vèrtex i tot seguit ho fes pels punts de mostreig.

A continuació es descriuen els diferents processos a seguir segons el tipus de càlcul que es vol realitzar, ja sigui aplicant el sistema de vèrtex, de punts de mostreig, la versió completa de l'algorisme o aplicant el sistema de retorn ràpid.

11.5.3.1 SDF per a vèrtex

Procés que cal aplicar per a treballar amb els vèrtexs dels objectes:

- Traslladar B1 als eixos de B2
- Calcular col·lisió per vèrtex
- Traslladar B2 als eixos de B1
- Calcular col·lisió per vèrtex

11.5.3.2 SDF per a punts de mostreig

Procés que cal aplicar per a treballar amb punts de mostreig dels objectes:

- Obtenir punts de mostreig de B1
- Traslladar B1 als eixos de B2
- Calcular col·lisió per punts de mostreig
- Obtenir punts de mostreig de B2
- Traslladar B2 als eixos de B1
- Calcular col·lisió per punts de mostreig

11.5.3.3 El procés complet

El procés complet aplicant els càlculs per vèrtex i per punts de mostreig quedaria de la següent manera:

- Traslladar B1 als eixos de B2
- Calcular col·lisió per vèrtex
- Traslladar B2 als eixos de B1
- Calcular col·lisió per vèrtex
- Obtenir punts de mostra de B1
- Traslladar B1 als eixos de B2
- Calcular col·lisió per punts de mostreig
- Obtenir punts de mostra de B2
- Traslladar B2 als eixos de B1
- Calcular col·lisió per punts de mostreig

11.5.3.4 L'aplicació del sistema de retorn

Com a optimització del codi es va aplicar un sistema de retorn ràpid en cas que en alguna fase intermèdia es detectés la col·lisió.

En comptes d'executar tot l'algorisme complet, en el moment que algun dels càlculs detecti col·lisió (la distància que retorni el sdBox fos negativa) es retorna.

11.5.4 Aplicació del disseny de pipes

A l'apartat "*10-Disseny*" s'ha parlat que es faria ús d'un disseny dinàmic de pipes.

Per a la tècnica de SDF s'han creat 12 pipes:

Primer de tot s'han definit tres pipes.

- SDF aplicant vèrtexs
- SDF aplicant punts de mostreig
- SDF aplicant vèrtex i punts de mostreig

En aplicar el control de mostreig a les 3 pipes han sortit tres pipes més:

- SDF aplicant vèrtexs fent ús del control de sortida
- SDF aplicant punts de mostreig fent ús del control de sortida
- SDF aplicant vèrtex i punts de mostreig fent ús del control de sortida

En aplicar la tècnica de col·lisió d'esferes sobre les 6 pipes anteriors han sortit un total de 12 pipes.

A continuació es llisten les pipes de SDF amb el número que s'ha utilitzat per realitzar les explicacions dels resultats.

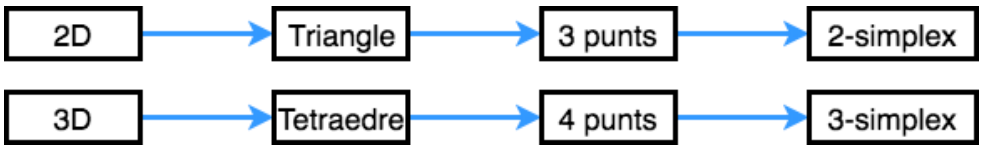
- **Pipe 3:** Col·lisió d'esferes + SDF complet amb control de sortida
- **Pipe 4:** SDF complet amb control de sortida
- **Pipe 5:** Col·lisió d'esferes + SDF complet
- **Pipe 6:** SDF complet
- **Pipe 7:** Col·lisió d'esferes + SDF per vèrtex amb control de sortida
- **Pipe 8:** SDF per vèrtex amb control de sortida
- **Pipe 9:** Col·lisió d'esferes + SDF per vèrtex
- **Pipe 10:** SDF per vèrtex
- **Pipe 11:** Col·lisió d'esferes + SDF per punts de mostreig amb control de sortida
- **Pipe 12:** SDF per punts de mostreig amb control de sortida
- **Pipe 13:** Col·lisió d'esferes + SDF per punts de mostreig
- **Pipe 14:** SDF per punts de mostreig

11.6 Gilbert-Johnson-Keerthi

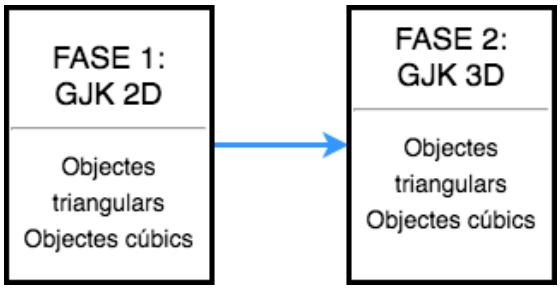
Un cop realitzat l'estudi de l'algoritme de Gilbert-Johnson-Keerthi (GJK) s'ha dut a terme la seva implementació.

Arribats a aquest punt s'ha decidit implementar només la part de l'algoritme que detecta si hi ha col·lisió, com a feina futura es realitzarà la implementació de la segona part, retornar la informació de col·lisió.

Per al cas implementat, k=2 i k=3 són les opcions corresponents per a un espai de 2 i 3 dimensions respectivament.



11.6.1 Fases d'implementació



11.6.1.1 FASE 1: Aplicació de GJK en 2D

En aquesta primera fase es va començar a treballar amb la tècnica del GJK en un espai 2D. Per comprendre el seu funcionament i com es realitzaria la implementació es van fer proves utilitzant objectes triangulars donat que el nombre de punts a calcular era menor que amb quadrats.

11.6.1.2 FASE 2: Aplicació de GJK 3D

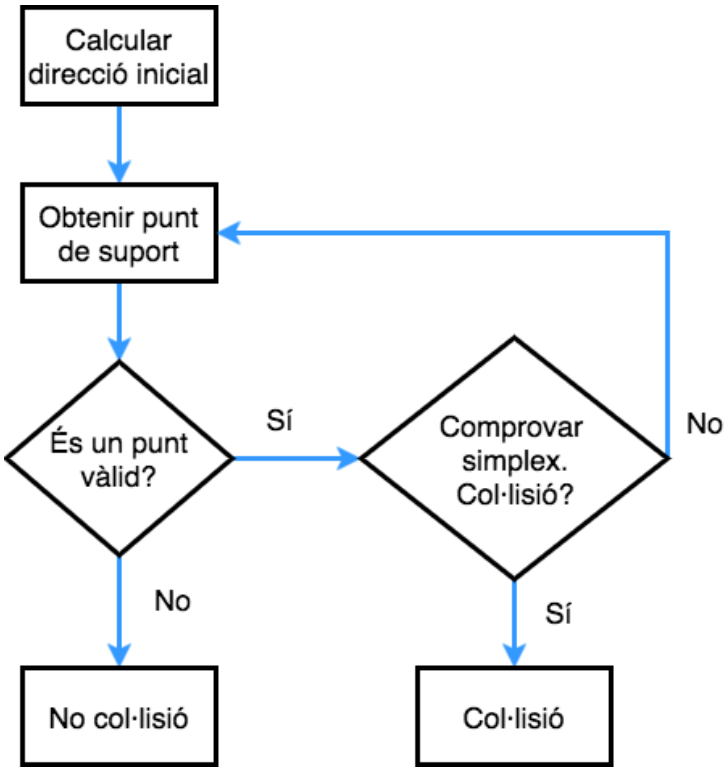
Un cop realitzada la implementació per a l'espai de 2D es va procedir amb l'espai de 3D.

Es va dur a terme el mateix procediment. Una primera etapa de treball amb objectes triangulars abans de fer el procés per a objectes cúbics.

11.6.2 Dades que necessita la tècnica

Aquesta tècnica treballa amb els vèrtexs en món dels objectes. Per a poder realitzar el càlcul dels punts en món és necessari disposar de cada objecte els seus vèrtexs originals (escalats) i les seves matrius de transformació (amb la informació de translació i escalat) o el vector translació i el quaternió amb la rotació.

11.6.3 Procés de la tècnica



11.6.3.1 Calcular direcció inicial

Inicialment es parteix amb un simplex buit. No hi ha cap punt ni direcció que pugui restringir la cerca del nou punt.

Així doncs la primera vegada cal definir una direcció de cerca. La manera més optima que s'ha cregut, consisteix a obtenir la MD dels centres dels dos objectes. En cas que els dos objectes tinguessin el mateix centre, la direcció resultant seria 0,0,0 amb la qual cosa no es podria cercar cap punt. En aquest cas s'ha optat per escollir una direcció auxiliar 1,0,0.

11.6.3.2 Obtenció dels punts de suport

A partir de la direcció determinada en aquell moment cal obtenir un punt per inserir al simplex i encapsular l'origen. Per fer-ho cal obtenir la MD en la direcció $\vec{d}$, restant el vèrtex més allunyat de l'objecte A en la direcció $\vec{d}$ i el vèrtex més allunyat en la direcció oposada -$\vec{d}$ de l'objecte B. Per obtenir el punt més llunyà en la direcció, ha sigut necessari recorre tots els vèrtexs de l'objecte i cercar quin està a més distància.

11.6.3.3 Comprovar que el punt és vàlid

Comprovar que el punt de suport obtingut ha sobrepassat l'origen. Si el producte vectorial entre el punt i el vector director $\vec{d}$ és superior a 0, llavors el punt és vàlid per afegir-lo al simplex i comprovar si podem encapsular l'origen.

En cas que el punt no sobrepassi l'origen llavors vol dir que no existeix cap vèrtex que es trobi en la direcció $\vec{d}$ que pugui ajudar a encapsular l'origen i, per tant, no es pot acabar de completar el simplex ni es pot encapsular l'origen. En aquest cas no hi ha intersecció i s'acaba l'execució.

11.6.3.4 Com cercar l'origen dins del simplex

En aquest apartat s'explica com es realitza la cerca de l'origen dins del simplex.

La zona de cerca és un espai extens, així doncs és necessari un algoritme que vagi a buscar l'origen directament. Per fer-ho cal omplir el simplex d'una manera ordenada i procurant que els punts que s'afegeixen formen un volum que inclogui l'origen. La manera en la qual s'ha treballat és l'explicada a l'apartat anterior, obtenint un punt cada vegada des de la funció encarregada d'obtenir els punts de suport i afegir-lo al conjunt de punts (simplex), es defineix que el punt nou sempre serà A i els altres punts aniran desplaçant-se segons convingui als altres punts dels simplex B,C,D.

Per cada punt que s'afegeix al simplex cal comprovar si l'origen queda encapsulat, en cas de no tenir suficients punts, caldrà comprovar quin **aspecte** (vèrtex, aresta, cara) queda més proper a l'origen per modificar la direcció de cerca i filtrar els punts del simplex per a quedar-nos només amb els punts que més s'adeqüin amb l'aspecte.

És en aquest filtratge on s'emmagatzema el punt A dins del simplex i varia la posició dels altres punts: A passa al lloc de B per deixar-li lloc al nou punt A, B passa al lloc de C... fent-ho de manera que no s'elimini cap valor.

L'algoritme finalitza quan s'ha construït un simplex triangular en 2D o un simplex amb forma de tetraedre per a 3D que inclogui l'origen o es troba que és impossible incloure'l.

A continuació hi ha exemplificat com evoluciona l'estat del simplex tenint en compte el nombre de punts que es troben al seu interior. S'explica la manera en la qual es busca quin aspecte és el que té l'origen més a prop, com es defineix la nova direcció de cerca i en quin estat queden els punts del simplex.

11.6.3.4.1 Cas 0: Cap punt al simplex

En aquest cas s'ha obtingut el primer punt però encara no s'ha afegit al simplex.

No hi ha suficients punts per crear un objecte convex que pugui englobar l'origen. Així doncs s'emmagatzema el punt A dins del punt B del simplex per deixar lloc al nou punt i s'actualitza la direcció de cerca, un vector que apunti des del punt a l'origen.

Un cop definida la direcció cal buscar un nou punt.

11.6.3.4.2 Cas 1: Un punt al simplex (1-simplex)

Amb els dos punts, A que és el nou punt trobat i B que ja està contingut al simplex, l'espai de treball queda dividit en 3 regions: A, B, AB.

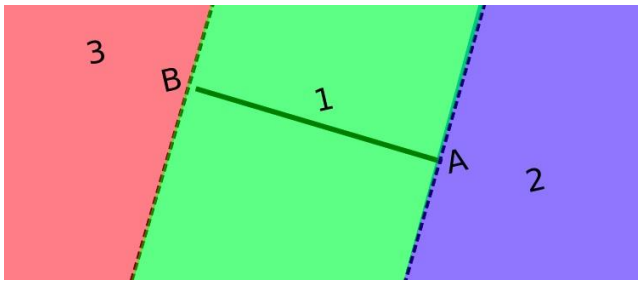


Figura 104: Regions possibles cas 1-simplex

Els plans de divisió són els plans perpendiculars al segment que passa pels dos punts A i B. La pregunta ara és quin dels aspectes de la línia té l'origen més proper, el punt A, el punt B o el vector AB. El **diagrama de Voronoi** superior ajuda a comprendre quines possibilitats existeixen *Figura 104.*

L'origen mai es trobarà a la **regió propera a B,** ja que el nou punt A s'ha cercat partint des de B (l'antic A) i en la direcció d'aquest a l'origen. La **regió propera a A** també es pot descartar, ja que el punt A està sobrepassant l'origen gràcies a la comprovació que es fa en el bucle exterior.

Per tant, l'origen segur que es troba a la regió entre els punts A i B per tant cal mantenir tots dos punts dins del simplex.

Un cop s'ha modificat el simplex amb els punts més propers a l'origen, cal actualitzar la direcció de cerca al nou punt. Així doncs cal realitzar un doble producte vectorial per obtenir una direcció perpendicular al segment però en la direcció de l'origen. Això és degut al fet que tot i saber que l'origen es troba en la regió 1 de la primera imatge, pot ser que l'origen es trobi per sobre del segment o per sota (regió 2 i 3 de la *Figura 105*).

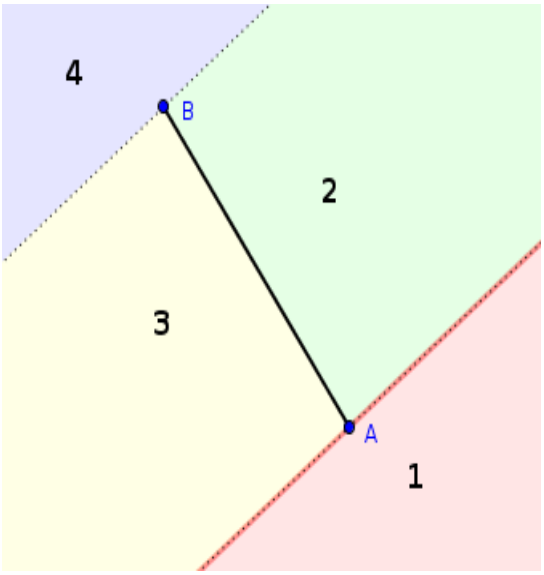


Figura 105: Regions possibles cas 1-simplex (reals)

11.6.3.4.3 Cas 2: Dos punts al símplex (2-simplex)

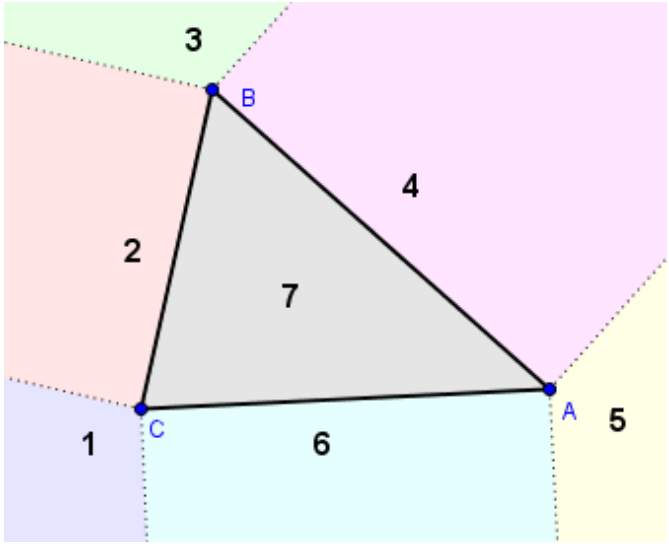


Figura 106: Regions possibles cas 2-simplex

En aquest punt s'ha obtingut un triangle afegint el nou punt A al segment BC (B i C estan emmagatzemats al simplex). Amb tres punts l'espai de treball queda dividit en 7 regions: A, B, C, AB, AC, BC, ABC. Els plans de divisió són els plans que passen pels diferents vectors AB, AC, BC.

L'origen mai es trobarà a les **regions 1, 2 i 3,** ja que a les iteracions prèvies ja s'ha determinat que l'origen s'ha de trobar entre B i C i en la direcció cap a A.

La **regió 5** també es pot descartar perquè se sap que el punt A està sobrepassant l'origen gràcies a la comprovació que es fa en el bucle exterior.

Així doncs queden tres regions per comprovar: 4, 6 i 7.

Per fer un codi més òptim les comprovacions de les regions resultants es realitza en l'ordre que permet arribar a una conclusió en el nombre menor de comprovacions, fent que el resultat d'una comprovació pugui descartar el nombre més gran de regions.

Primer de tot es comprova la **regió 4**. Si el producte escalar entre el vector perpendicular d'AB i el vector AO és positiu, llavors vol dir que l'origen es troba fora del triangle prop de l'aresta AB. Així doncs s'elimina el punt C del simplex, el punt B passa a ser C i A passa a ser B, d'aquesta manera es deixa el punt A lliure per al nou punt. I es defineix la nova direcció de cerca realitzant un doble producte vectorial per obtenir una direcció perpendicular al segment AB.

En cas que la comprovació de la regió 4 no contingui l'origen cal comprovar la **regió 6** realitzant el mateix procediment però respecte al vector AC.
Si l'origen es troba en la direcció resultant de fer el producte escalar entre el vector perpendicular d'AC i el vector AO llavors vol dir que l'origen es troba fora del triangle prop de l'aresta AC. En aquest cas cal eliminar el punt B del simplex: el punt A passa ser B i d'aquesta manera es deixa el punt A lliure per al nou punt. Finalment es defineix la nova direcció de cerca, realitzant un doble producte vectorial per obtenir una direcció perpendicular al segment AC.

En cas que la comprovació de la regió 6 no contingui l'origen vol dir que l'origen es troba a la **regió 7,** és a dir, l'origen es troba dins de la regió del triangle format pels punts ABC.

Com que s'està treballant en un espai 3D, per completar l'execució és necessari trobar un últim punt per tal de construir el tetraedre (en cas de treballar en un espai 2D ja es podria concloure que hi ha col·lisió, ja que només es necessita un simplex de 3). Cal emmagatzemar els tres punts al simplex: C passa a ser D, B passa a ser C i A passa a ser B i d'aquesta manera es deixa el punt A lliure per al nou punt.

Finalment roman definir la nova direcció de cerca. En aquest punt existeixen dues opcions, que l'origen quedi per sobre del pla del triangle o que quedi per sota. Comprovar en quin dels dos casos s'està permet fer una gran optimització en el càlcul, ja que d'aquesta manera es poden emmagatzemar els punts al símplex en l'ordre més òptim.
En cas que l'origen es trobi per sobre del pla del triangle: el punt C passa a ser D, el punt B passa a ser C i el punt A passa a ser B (d'aquesta manera es deixa el punt A lliure per al nou punt) i es defineix la nova direcció de cerca com el vector perpendicular al pla ABC.
En cas contrari cal emmagatzemar els punts al revés de manera que el triangle quedi en la posició correcta. El punt B passa a ser D i el punt A passa a ser B. Finalment es defineix la nova direcció de cerca com el vector oposat al perpendicular al pla ABC.

11.6.3.4.4 Cas 3: Tres punts al simplex (3-simplex)

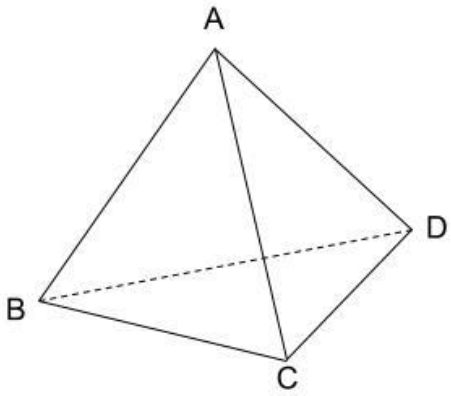


Figura 107: Exemple de tetraedre

En aquest punt s'ha obtingut un tetraedre afegint el nou punt A al triangle BCD (B, C i D estan emmagatzemats al simplex). Amb quatre punts l'espai de treball queda dividit en 14 regions: A, B, C, D, AB, AC, AD, BC, BD, CD, ABC, ABD, ACD, BCD. Els plans de divisió són els plans que passen pels diferents vectors AB, AC, AD, BC, BD, CD.

Cadascuna de les arestes (BC, BD, CD) han estat comprovades durant la construcció i se sap que l'origen no es troba en cap d'aquestes regions. Aquest fet redueix l'espai de cerca de l'origen a un prisma triangular infinit (alineat a la normal del pla format pel triangle BCD).

En realitat no es tracta d'un prisma infinit, la construcció del triangle s'ha realitzat controlant cap a quina zona es trobava l'origen en tot moment, se sap pel cas 2 regió 7 que l'origen mai es trobarà sota el triangle BCD, així doncs la meitat del prisma infinit no caldrà comprovar-lo. També se sap que el nou punt A, es troba no només per sobre del pla del triangle sinó que també queda per sobre de l'origen. Així doncs únicament cal comprovar un prisma triangular, amb base inferior el triangle BCD i per base superior el pla paral·lel al del triangle BCD però que passa pel punt A, ja que la zona per sobre del punt A no conté l'origen.

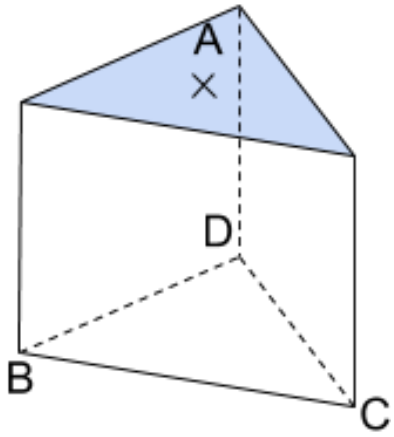


Figura 108: Prisma triangular

Així doncs queden tres regions per comprovar: 3 noves arestes, tres nous plans i el volum interior del tetraedre.

- Tres noves arestes: ab, ac, ad
- Tres nous plans: abc, acd, adb

El cas final (el que permetrà acabar l'execució de l'algoritme) es dóna quan l'origen es troba dins del tetraedre. L'única manera per trobar-lo dins és si no s'ha trobat darrere de les tres noves cares, així doncs cal començar comprovant els tres plans.

En cas que l'origen no es trobi darrere dels plans, és possible que es trobi davant d'un o dos plans. En cas que l'origen es trobi per davant d'una cara cal comprovar el pla i les seves arestes. En cas que es trobi per davant de dues cares, caldrà considerar tots dos plans i totes 3 arestes.

Primer de tot es comproven els tres plans de les noves cares. Aquestes comprovacions simplifiquen l'execució, ja que s'ha tingut cura a l'hora d'emmagatzemar els punts del triangle de forma consistent i ordenada, això permet saber en cada moment quin cantó del pla és l'interior i quin l'exterior.

Al fer aquestes comprovacions, fent els respectius productes escalars entre els diferents plans i el vector AO, ha d'aparèixer un dels següents casos:

- Opció 1: 1 cas perquè el punt es trobi dins del tetraedre.
- Opció 2: 3 casos perquè el punt es trobi davant d'una sola cara.
- Opció 3: 3 casos perquè el punt es trobi davant de dues cares.

Mentre que en total hi ha 7 casos individuals, en realitat es poden diferenciar 3 tipus de casos. Cadascun dels subcasos dins de les opcions 2 i 3 canvien en la manera en la qual estan rotats. Per aquest motiu només s'ha escollit un cas per resoldre el càlcul i si detectem que es tracta d'una versió rotada només ha sigut necessari ordenar els punts del simplex. Aquest procediment permet simplificar el codi en gran mesura.

En cas que l'origen es trobi davant d'una cara, l'algoritme es troba en la situació d'un triangle, ja que s'haurà descartat un dels punts del simplex (B, C o D). Caldrà fer les comprovacions explicades quan hi ha 2 punts al simplex (apartat *"11.6.3.4.3 Cas 2: Dos punts al símplex (2-simplex)"*) per obtenir els punts adequats i definir la nova direcció de cerca de cara a la següent iteració, amb l'excepció que no caldrà comprovar si l'origen es troba per sota del pla, ja que aquesta opció s'haurà descartat prèviament.

En cas que l'origen es trobi per davant de dues cares, primer de tot serà necessari comprovar l'aresta que comparteixen. Si l'origen es troba passada l'aresta, llavors aquest es pot trobar en qualsevol de les regions de l'altra cara o arestes, però en definitiva no deixa de ser una sola "cara". Així doncs es pot actuar com en el cas previ, quan l'origen es trobava per davant d'un sol pla. Si al fer aquesta comprovació no s'elimina la regió de la cara que s'està mirant, llavors de nou s'està en el cas que s'ha de comprovar una sola cara, però en aquest punt no caldrà comprovar l'aresta compartida.

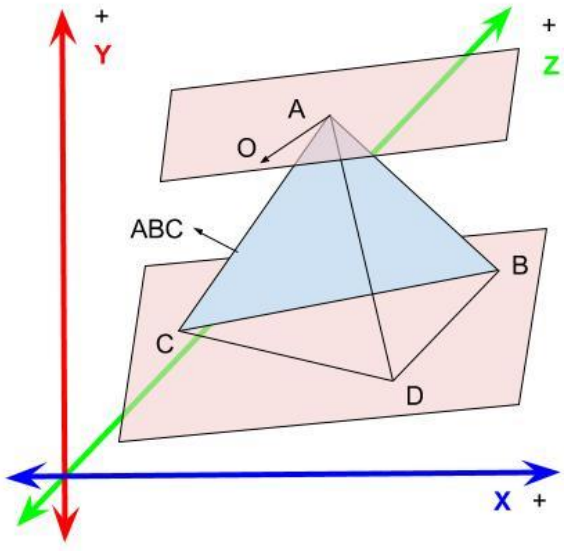


Figura 109: Exemple dels casos quan O es troba per davant del pla ABC

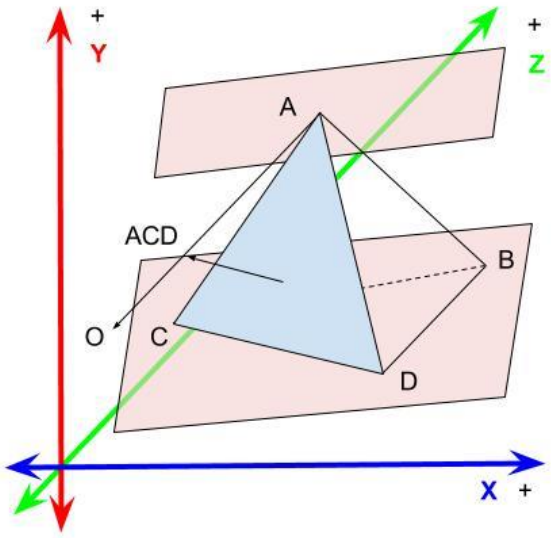


Figura 110: Exemple dels casos quan O es troba per davant del pla ACD

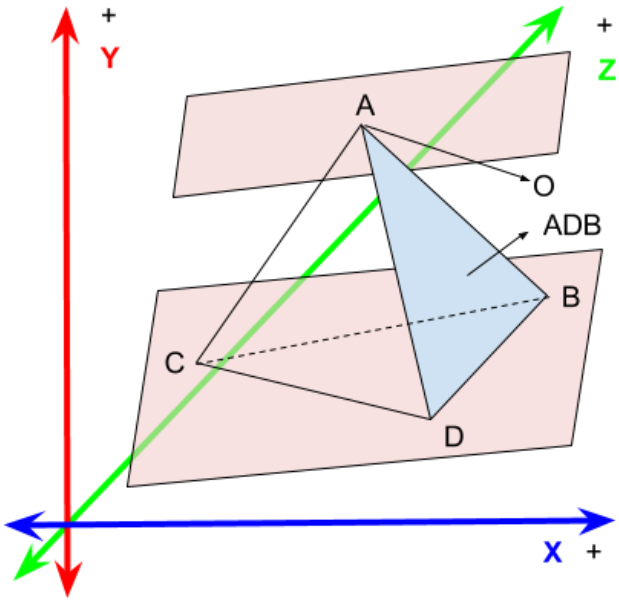


Figura 111: Exemple dels casos quan O es troba per davant del pla ADB

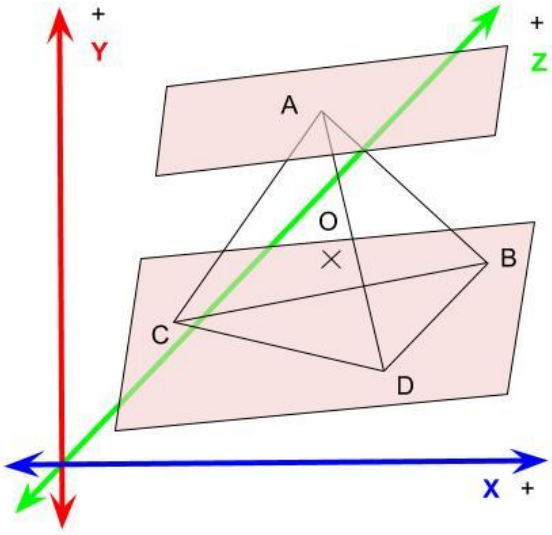


Figura 112: Exemple on O es troba dins de l'objecte

11.6.4 GJK EN 2D

Tot i que es va començar implementant la tècnica en un sistema de dues dimensions i posteriorment es va realitzar la implementació per a un sistema de tres dimensions, s'ha preferit prioritzar l'explicació per a un sistema de tres dimensions donat que és molt més completa i no deixa de ser una evolució de la tècnica en 2D.

A continuació es fa una breu explicació dels canvis realitzats per a treballar en un sistema 2D.

Tot el cas 3, quan al simplex hi han emmagatzemats 3 punts es podria eliminar completament, ja que un triangle és suficient per encapsular l'origen (com s'ha explicat al capítol de teoria *"7.5-L'Algoritme de Gilbert-Johnson-Keerthi"*). Els càlculs dels productes escalars caldria reemplaçar-los per la forma apropiada per obtenir un vector perpendicular en 2 dimensions. Finalment si l'execució arriba a la regió 7 del cas 2, quan s'han obtingut els 3 punts i s'ha comprovat que l'origen es troba al seu interior, es pot finalitzar l'execució i retornar que s'ha detectat col·lisió.

11.6.5 Aplicació del disseny de pipes

A l'apartat "*10-Disseny*" s'ha parlat que es faria ús d'un disseny dinàmic de pipes.

Per a la tècnica de GJK s'han creat 2 pipes:

- **Pipe 15:** Col·lisió d'esferes + GJK
- **Pipe 16:** GJK

11.7 Pas del GJK a un disseny orientat a dades

Un cop s'ha realitzat la implementació de les 16 pipes en el disseny orientat a objectes i es va realitzar la primera comparativa va arribar el moment de fer la comparativa amb un disseny orientat a dades. Les dues pipes més òptimes, les pipes de GJK (pipe 15 i 16) són les que s'han escollit per adaptar al nou disseny.

A partir de la teoria explicada a l'apartat *"7.6-Disseny orientat a dades"* per fer el pas de disseny s'han hagut de realitzar els següents canvis.

11.7.1 Canviar classes per namespace

Per poder seguir definint els atributs privats i públics que formaven part de les classes del disseny orientat a objectes s'han creat per cada classe un namespace que fes de nom de la classe i un namespace que englobés els atributs i funcions privades de la classe.

```
namespace GJK{
	//atributs i funcions públiques
	namespace Detail{
		//atributs i funcions privades
	}
{
```

Com a optimització en realitzar aquest canvi també es van eliminar els atributs privats i es van afegir a les funcions del GJK convertint-les en void (com a paràmetre i per referència) perquè l'execució sigues més ràpida.

11.7.2 Crear una estructura d'arrays

Per emmagatzemar les dades, donat que les instàncies de Box s'han d'eliminar, ha sigut necessari crear una estructura d'arrays per emmagatzemar en memòria les dades dels diferents objectes.

Com s'ha definit a l'apartat d'especificació *"9.3.3.1.1-BoxManager"*, aquesta estructura de dades permet emmagatzemar cadascun dels components dels objectes: identificadors, vèrtexs locals, els vectors de translació, els quaternions de rotació, els vèrtexs en món, els radis i els centres de les bounding Box en món.

Aquesta estructura de dades s'ha declarat com a **extern** dins del namespace *Detail* perquè fos accessible des de qualsevol lloc del codi però mantenint la jerarquia de classe. Així doncs per accedir a les dades de l'estructura s'ha d'haver afegit les següents declaracions dins del fitxer des d'on es vulgui accedir:

```
using namespace GJK::Detail
GJK::Values GJK::Detail::values
```

11.7.3 Reserva de memòria

La primera cosa que s'ha d'executar és la reserva de la memòria que es voldrà fer ús com a màxim, s'ha definit que el motor tindrà com a màxim 1000 objectes (però aquest valor s'ha declarat com a constant per a poder-se canviar amb facilitat en el futur.

Per eliminar pèrdues de temps d'execució produïdes per la demanada de més memòria, aquesta reserva de memòria es realitza inicialment en comptes de fer petites ampliacions dels arrays quan aquests es quedessin sense espai.

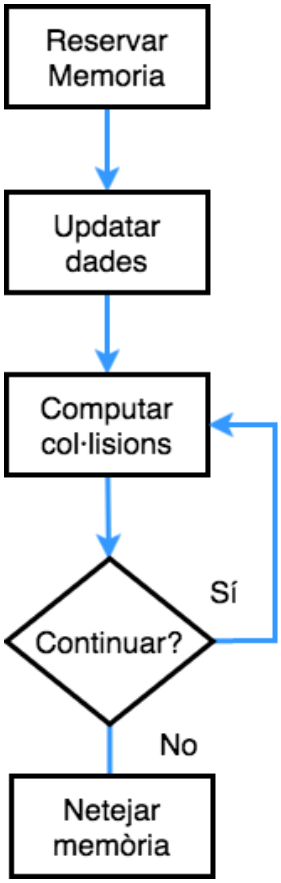
La reserva per als arrays de vèrtex es va realitzar de la següent manera donat que s'ha de tenir el compte emmagatzemar 8 vèrtexs per entrada:

```
values.xPosition = (float*)_aligned_malloc(sizeof(float) * 8 * MAX_BOX, sizeof(float) * 8);
```

i la reserva de la resta d'arrays la reserva es realitzaria com:

```
values.xTranslation = (float*)_aligned_malloc(sizeof(float) * MAX_BOX, sizeof(float));
```

El procés que es realitza en l'execució segueix el següent diagrama de flux:



11.7.4 Accés a les dades

Així doncs el *BoxManager* en comptes de crear una instància, ara ha d'emmagatzemar les dades en les posicions dels arrays corresponents. Per realitzar aquesta inserció i col·locar les dades en el lloc correcte s'ha treballat amb els identificadors dels objectes. En el moment d'inserir per primer cop un objecte, a aquest se li dóna un identificador i a partir d'aquest valor s'emmagatzemen les dades.

De la mateixa manera que es realitza la inserció, mitjançant l'identificador de l'objecte s'accedeix a la posició de l'array corresponent de la dada que es vol obtenir.

La part més complicada és mantenir la relació entre els arrays de vèrtex i la resta d'arrays donat que hi ha una relació de 8 vèrtexs per qualsevol dels altres valors. Per exemple, per emmagatzemar el vector de translació només s'ha d'emmagatzemar un

valor dins de cada array, però per emmagatzemar els vèrtexs originals s'han d'emmagatzemar 8 valors.

Per a poder treballar només amb els identificadors ha sigut necessari preparar el codi per a treballar amb aquests dos casos.

Per obtenir/emmagatzemar valor d'un sol component només cal accedir a:

```
values.xTranslation[id]
```

i en cas que es volgués obtenir/emmagatzemar valors dels vèrtexs s'ha d'executar un bucle de 8 iteracions de la següent manera:

```
for (int i = 0; i < 8; i ++){
      values.xLocalPosition[id * 8 * i] = component x de la posició
original del vèrtex i
      values.yLocalPosition[id * 8 * i] = component y de la posició
original del vèrtex i
      values.zLocalPosition[id * 8 * i] = component z de la posició
original del vèrtex i
}
```

12 Resultats

12.1 Resultats de l'estudi orientat a objectes

Un cop implementades les diferents pipes amb el disseny orientat a objectes ha sigut necessari fer un estudi per escollir les dues millors pipes per implementar-les amb el disseny orientat a dades.

Per a realitzar aquesta tria s'han utilitzat per a una banda els resultats de la bateria de tests, que ha permès detectar certes mancances de les tècniques, i per l'altre els temps d'execució de cadascuna de les pipes en un mateix entorn i entrada de dades.

Per generar els objectes que s'han utilitzat per obtenir els temps d'execució, s'han creat de manera aleatòria 100 objectes amb les següents característiques:

- Mida aleatòria entre 0 i 5 unitats
- Posició aleatòria entre -2 i 2 unitats
- Rotació aleatòria entre 0 i 360 graus

Dins del codi s'han afegit timestamps a l'entrada i a la sortida de cadascuna de les funcions que realitzen l'execució, controlant que només s'enregistri el temps de càlcul, és a dir, s'ha exclòs el temps de creació dels objectes.

Amb aquestes consideracions s'han executat les 16 pipes i s'han obtingut els següents resultats:

TÈCNICA	PIPE	TEMPS D'EXECUCIÓ (ms)
SAT	SC+SAT	143,785
	SAT	154,122
SDF	SC+SDFCS	94,753
	SDFCS	108,713
	SC+SDFC	253,556
	SDFC	280,785
	SC+SDFVS	**4,529**
	SDFVS	**4,095**
	SC+SDFV	**4,385**
	SDFV	**4,535**
	SC+SDFPS	158,417
	SDFPS	177,444
	SC+SDFP	253,097
	SDFP	270,897
GJK	SC+GJK	**4,299**
	GJK	**3,077**

Taula 55: Resultats dels temps d'execució de les 16 pipes en el disseny orientat a objectes

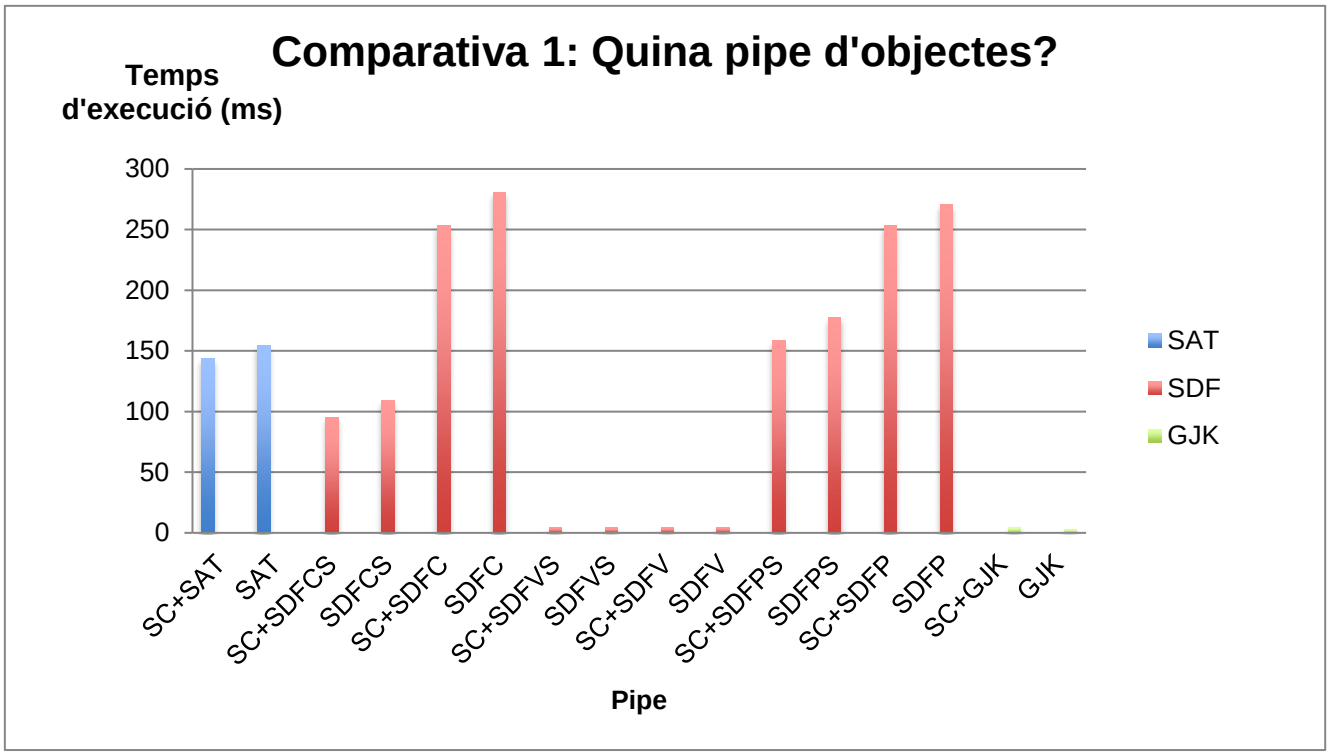


Figura 113: Il·lustració dels resultat de les pipes del disseny orientat a objectes

De les dades obtingudes es pot deduir que les millors opcions respecte a temps d'execució són les pipes del SDF per vèrtex i les pipes del GJK. La resta de pipes superen entre 30/40 vegades el temp requerit pel projecte (trigar entre 3-4 ms en executar les col·lisions amb 100 objectes).

Ara bé de les proves realitzades amb la bateria de tests s'ha comprovat que les pipes del SDF per vèrtex tenen certes limitacions. Per exemple, quan els vèrtexs dels objectes queden fora de la zona de col·lisió, és a dir, si cap dels vèrtexs es troba dins d'un altre objecte.

A continuació hi ha una imatge extreta del visualitzador del framework on es pot veure un d'aquests casos.

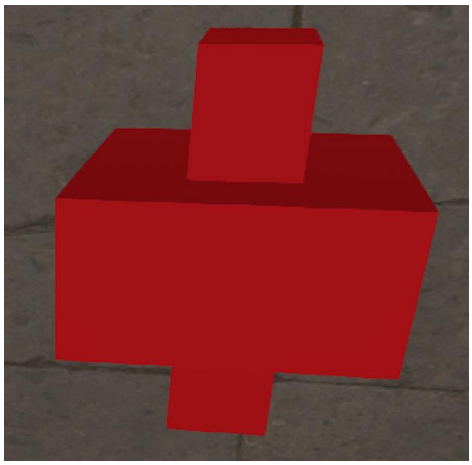


Figura 114: Exemple de situació on el SDF per vèrtex falla

A més a més, l'èxit de donar el resultat correcte depèn molt de les formes dels objectes. Si l'objecte és molt estret pot donar-se el cas que cap dels 1000 punts de mostreig es trobi dins de la zona de col·lisió. A la imatge següent es mostra un dels tests realitzats (test21) on es pot comprovar que la zona de col·lisió és la recta formada pel tall dels dos objectes. Depenent dels punts de mostreig obtinguts s'han donat casos en què les pipes de SDF per punts de mostreig no han detectat la col·lisió.

Figura 115: Exemple de situació on la zona de col·lisió és una recta

Per aquests motius finalment s'han escollit les pipes de GJK per realitzar la comparativa amb el disseny orientat a dades.

12.2 Resultats de la comparativa de disseny orientat a objectes vs disseny orientat a dades

Un cop escollides les dues millors pipes del disseny orientat a objectes s'ha dut a terme la comparativa amb el disseny orientat a dades.

Per a poder realitzar una comparativa real s'han sotmès les 4 pipes al mateix entorn de càlcul i a les mateixes dades d'entrada.

Per fer-ho primer s'han creat 100 objectes aleatoris seguint les característiques definides a l'apartat 12.1 i s'han emmagatzemat en un arxiu extern. D'aquesta manera les 4 execucions s'han realitzat llegint les dades del mateix arxiu.

Amb aquestes consideracions s'han executat les 4 pipes i s'han obtingut els següents resultats:

DISSENY	PIPE	TEMPS D'EXECUCIÓ (ms)
Orientat a objectes	SC+GJK	4,299
	GJK	3,077
Orientat a dades	SC+GJK	2,288
	GJK	1,947

Taula 56: Resultats dels temps d'execució de les 4 pipes de GJK. Comparativa entre disseny orientat a objectes i disseny orientat a dades

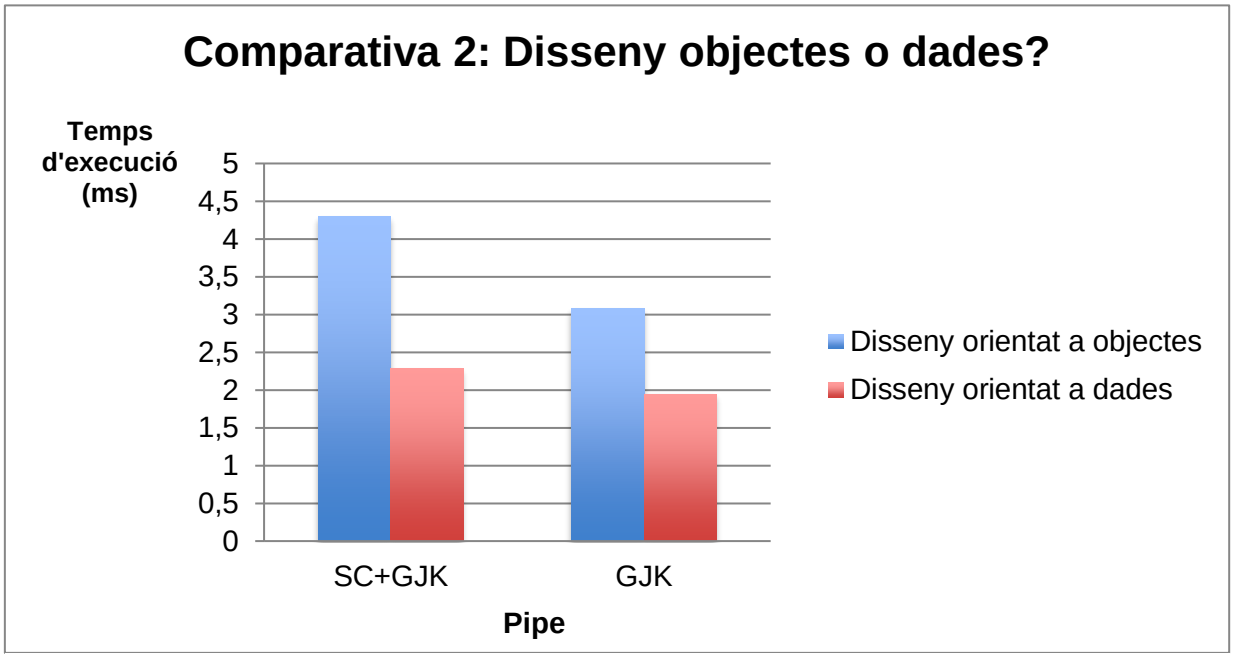


Figura 116: Resultats dels temps d'execució entre les pipes de JKG segons el disseny

De les dades obtingudes es pot deduir que les millors opcions respecte a temps d'execució són les pipes executades dins del disseny orientat a dades.

12.3 GJK amb col·lisió d'esferes o sense?

Finalment mancava per escollir quina de les dues pipes de GJK amb el disseny orientat a dades s'havia d'incloure en el framework.

En els temps obtinguts a l'apartat anterior s'ha vist que la pipe de SC+GJK triga un 7% més que la pipe que executa només la tècnica de GJK.

De les 4950 combinacions que surten d'emparellar els 100 objectes la col·lisió d'esferes ha detectat col·lisió en uns 4515 casos i d'aquests, 3760 han donat col·lisió per la pipe. És a dir, la col·lisió d'esferes ha eliminat només el 8.78% dels casos.

Així doncs la pipe de SC+GJK triga més donat l'alta quantitat d'objectes que col·lisionen, ja que cal afegir el temps de calcular la col·lisió d'esferes per a tots els casos necessaris.

S'han realitzat unes noves proves amb objectes més separats. Per fer-ho s'ha realitzat el mateix procediment que amb les proves anteriors però canviant les característiques de creació dels 100 objectes perquè es produeixin menys col·lisions, fent que la posició variï entre -4 i 4 unitats (mitja distància) i entre -5 i 5 unitats (separats).

DISSENY	PIPE	JUNTS	MITJA DISTÀNCIA	SEPARATS
Objectes	SC+GJK	4,299283	2,206691	1,783069
	GJK	3,077908	2,928293	2,313285
Dades	SC+GJK	2,288802	1,15112	0,742923
	GJK	1,947404	1,344093	1,201609

Taula 57: Resultats dels temps d'execució de les 4 pipes de GJK segon disseny i separació

Respecte a la primera prova, amb els objectes posicionats a mitja distància (posició entre -4 i 4), del total de combinacions la col·lisió d'esferes ha detectat col·lisió en uns 2051 casos i d'aquests, 1322 han donat col·lisió per la pipe. És a dir, la col·lisió d'esferes ha eliminat el 58% dels casos.
La pipe de SC+GJK ha trigat 1,15 ms a calcular les 2051 col·lisions detectades per la col·lisió d'esferes i la pipe GJK ha trigat 1,34 ms a calcular les 1322 col·lisions. De tal manera que s'ha produït una millora del 14% quant a temps d'execució.

Respecte a la segona prova, amb els objectes posicionats a mitja distància (posició entre -5 i 5), del total de combinacions la col·lisió d'esferes ha detectat col·lisió en uns 1240 casos i d'aquests, 729 han donat col·lisió per la pipe. És a dir, la col·lisió d'esferes ha eliminat el 75% dels casos.

La pipe de SC+GJK ha trigat 0,74 ms a calcular les 1240 col·lisions detectades per la col·lisió d'esferes i la pipe GJK ha trigat 1,2 ms a calcular les 729 col·lisions. De tal manera que s'ha produït una millora del 38% quant a temps d'execució.

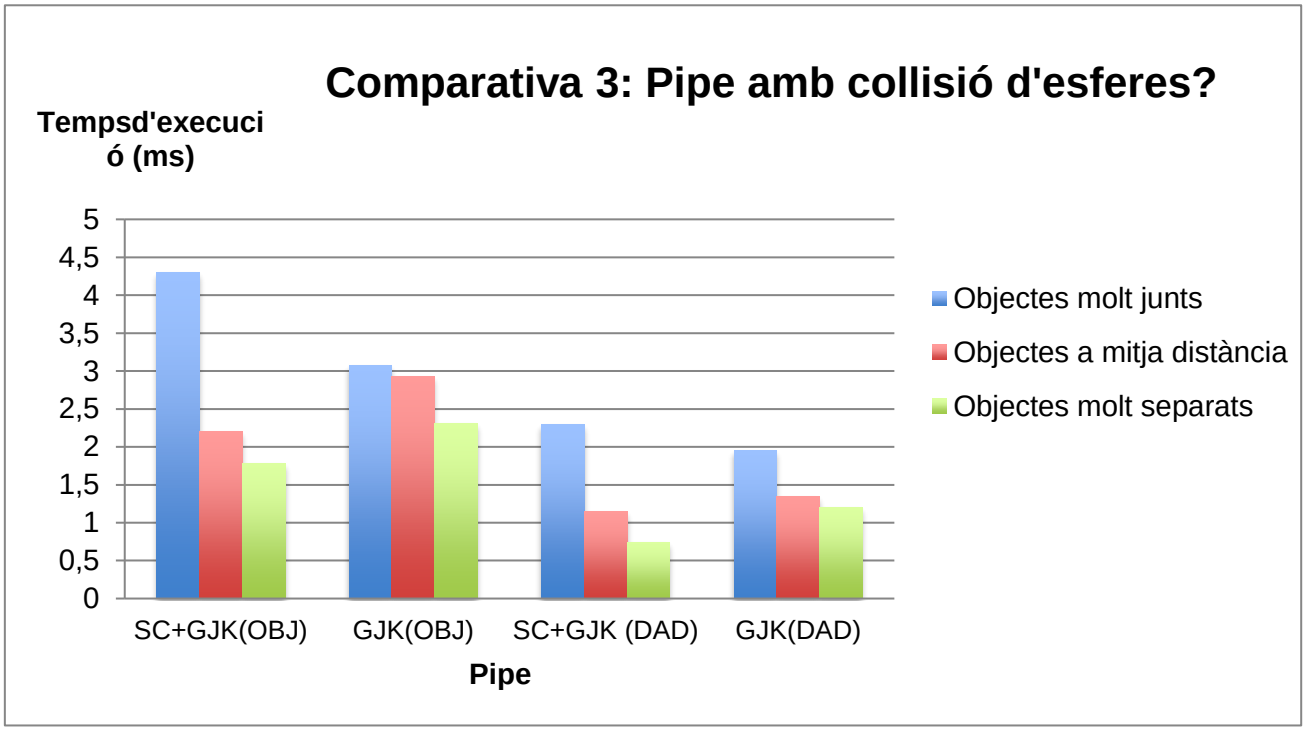


Figura 117: Il·lustració dels resultat de la comparativa entre es pipes del GJK entre disseny orientat a objectes i disseny orientat a dades

Donat que el nombre de col·lisions que s'arriben a donar a les escenes en les quals treballen les aplicacions de l'empresa és reduït s'ha decidit seguir utilitzant la col·lisió d'esferes.

Com a conclusió final s'ha escollit implementar dins del framework la pipe de SC+GJK amb el disseny orientat a dades.

13 Implantació

L'última etapa del projecte ha consistit a realitzar la implantació de la millor pipe dins del framework.

Aquesta implantació es va planificar inicialment per a realitzar-se a finals de novembre/principis de desembre però per certs motius (tractats a l'apartat *"15-Planificació final")* no es va poder fer fins al març.

Donat que per part de l'empresa el projecte es va donar per tancat (sense la implantació) i es va reprendre a finals de març com una tasca externa a la jornada laboral es van definir nous requisits.

A continuació es descriuen els nous requisits, l'estructura del framework i la implementació de la tasca d'implantació.

13.1 Nous requisits

- El motor de col·lisions ha de ser un mòdul del core del framework, ja que aquest ha de poder funcionar sense sistema de col·lisions.
- Un objecte està definit per: translació, rotació, escalat, tag i grup.
- S'han de poder afegir objectes
- S'han de poder modificar els objectes: translació, rotació, escalat, tag i grup.
- Els usuaris han de poder calcular totes les col·lisions entre els objectes amb tag *tag1* i els objectes amb tag *tag2.*
- Els usuaris han de poder calcular totes les col·lisions entre els objectes amb grup *group1* i els objectes amb grup *group2.*
- Els usuaris han de poder calcular totes les col·lisions entre un objecte *obj* i els altres objectes inserits a l'escena.
- Els usuaris han de poder calcular totes les col·lisions entre un objecte *obj* i els objectes amb tag *tag.*
- Els usuaris han de poder calcular totes les col·lisions entre un objecte *obj* i els objectes amb grup *group.*

Com a dades d'un objecte s'han afegit dos camps més, el *tag* i el *grup*. Donat que el motor només treballa amb objectes convexos és necessari que un objecte no convex

es divideixi en subobjectes convexos (com s'ha explicat a l'apartat *"7.2.7 Convexitat"*). Per poder relacionar els diferents subobjectes dins d'un mateix objecte s'hi ha inclòs el paràmetre *grup*.
Finalment s'ha afegit el paràmetre *tag* per requeriment de l'empresa per a poder realitzar les col·lisions entre els objectes que disposin de certes característiques. Aquesta definició de característiques es realitza a partir d'un número binari (un tag).

13.2 Estructura del framework

El framework està dividit en dos nivells, una capa superficial anomenada framework i una capa interna anomenada core. Dins del core es realitza tota la lògica del sistema i el framework fa una capa d'abstracció entre l'usuari i el core.

Dins del framework s'ha utilitzat el patró de components (*Component Pattern*) per estructurar el codi. *"A single entity spans multiple domains. To keep the domains isolated, the code for each is placed in its own component class. The entity is reduced to a simple container of components."*[7]

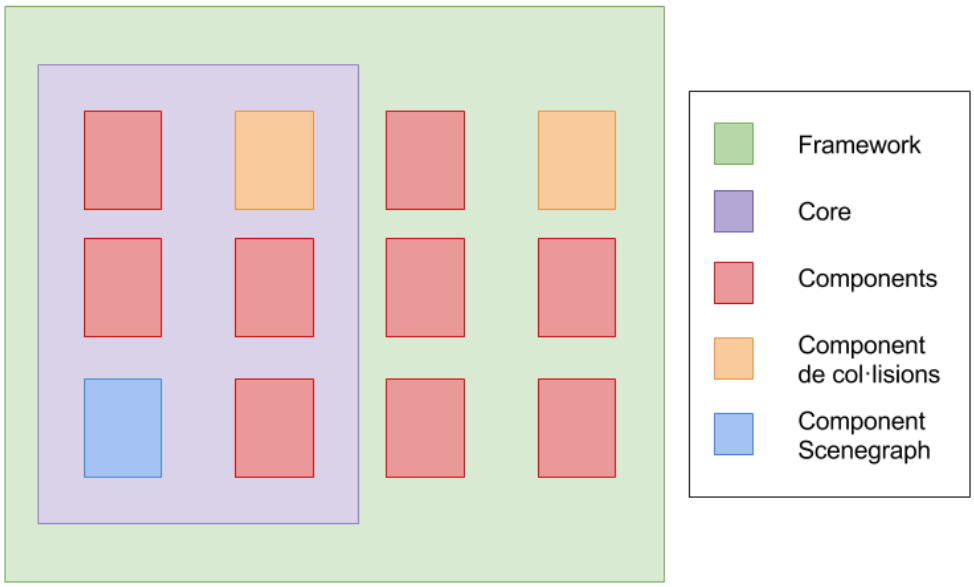


Figura 118: Exemple de l'estructura del Framework

El framework doncs està compost per un conjunt de components: Entity, Mesh, Scenegraph, Camera, Light, Material...

[7] Patró de components: http://gameprogrammingpatterns.com/component.html

En aquesta memòria no es farà una descripció detallada dels components però si una breu explicació per comprendre la tasca realitzada.

- Dins del motor tots els objectes es creen com a entitats. A cada entitat se li poden afegir els diferents components del framework.
- El component Scenegraph, és l'encarregat de gestionar la jerarquia dins del framework, emmagatzema la relació que existeix entre els diferents objectes (posicions, si hi ha alguna relació d'objectes dins d'objectes).
- Per poder visualitzar existeixen els components Camera, Light, Material i Texture que permeten visualitzar i modificar les condicions dels objectes.
- També disposa d'un component d'inputs amb el qual es poden realitzar interaccions per teclat amb els objectes.

En resum, l'usuari (treballador d'interiorvista) realitza peticions al framework i aquest les gestiona i les dirigeix al component corresponent del core.

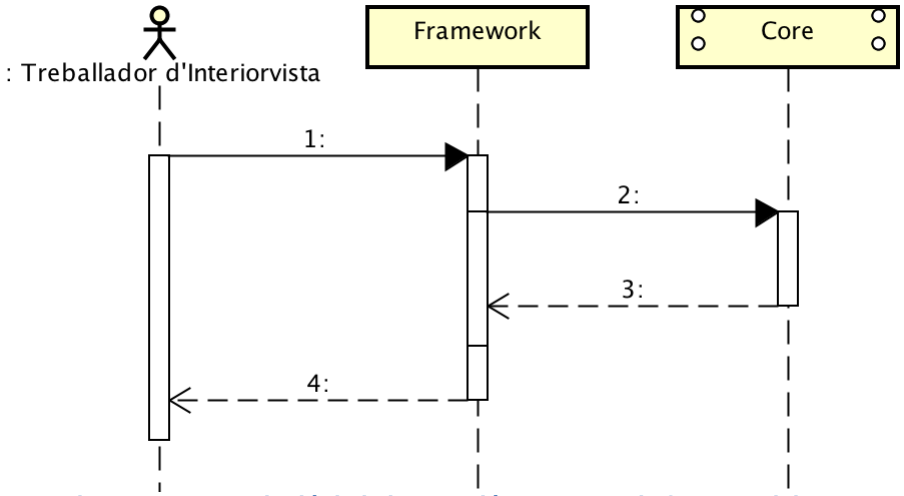


Figura 119: Descripció de la interacció entre usuari , framework i core

13.3 Implementació de la implantació

En aquest apartat es descriuen el procés d'implantació de la pipe guanyadora dins del framework de l'empresa.

13.3.1 CollisionManager

Conté tota la lògica del motor realitzada durant la primera part del projecte.

Durant aquest procés s'ha realitzat una optimització respecte a com accedeix la pipe a les dades. En comptes de treballar amb uns arrays que continguin tots els objectes de l'escena, s'han creat dos sets d'arrays específics per emmagatzemar les dades dels objectes que s'han de comprovar en aquell moment.

És a dir, si l'usuari escull comprovar les col·lisions entre els objectes de tag=2 i els objectes de tag=4 llavors en el primer set s'emmagatzemen només els objectes amb tag=2 i en el segon set s'emmagatzemaran només els objectes amb tag=4.

D'aquesta manera cada vegada que es carrega una línia dins de cache s'aprofita la localitat espacial i es carregarà sencera amb dades útils per al càlcul.

Les dades necessàries per la pipe són: les posicions en món dels vèrtexs, els centres en món i els identificadors dels objectes. Per aquest motiu s'han creat 16 nous arrays dins de l'estructura d'arrays per emmagatzemar totes les components.

A més a més, s'han afegit tres arrays més per emmagatzemar les dades dels identificadors dins del scenegraph, els tag i els grups.

Per a les 6 tipologies de càlcul s'han creat sis funcions que són les encarregades d'omplir aquest 16 arrays en funció de la definició de càlcul requerida.

A continuació és mostra l'especificació del Collisionmanager (només es llisten els atributs nous i les noves funcions):

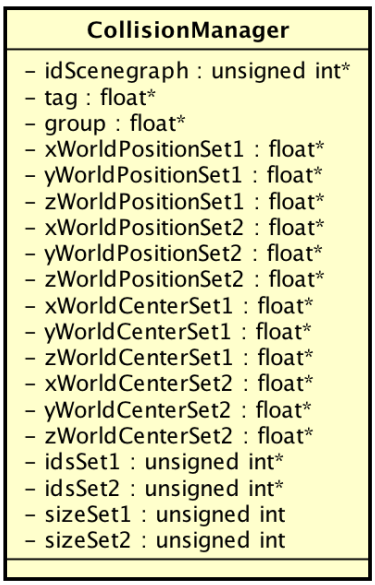


Figura 120: Identitats de l'estructura d'arrays

A *l'Annex 5* s'ha inclòs l'especificació de les funcions del collisionManager.

13.3.2 CollisionComponent

És una classe que hereta de EntityComponent.

Com s'ha dit prèviament el CollisionComponent està a la part del framework i és l'encarregat de fer de pont entre l'usuari i el core. Per aquest motiu no s'ha inclòs la seva especificació, ja que és molt semblant a la del CollisionManager.

Només remarcar que la funció setBox rep un paràmetre menys, ja que inicialment encara no es disposa de l'identificador de l'objecte dins del scenegraph i és en aquest punt en la qual es pot obtenir i enviar al manager.

13.3.3 Main

El main conté tota la definició per a poder visualitzar una escena. A part de realitzar el seteig de totes les entitats i components per visualitzar una escena també és l'encarregat de crear les entitats, definir els inputs de teclats necessaris i executar el codi en bucle.

13.3.3.1 Crear les entitats

Per crear un objecte s'han seguit els següents passos:

- Crear l'entitat
- Modificar les dades de l'entitat dins del scenegraph (setPosition, setScale, setOrientation)
- Afegir un component MeshComponent amb una mesh en forma de cub unitari.
- Afegir un component CollisionComponent.
- Emmagatzemar en memòria l'objecte seguin l'especificació del SetBox realitzada al CollisionComponent. Com que l'entitat ja disposa de les dades de transformació de l'objecte, en memòria inicialment s'emmagatzema un cub unitari AABB.
- Emmagatzemar l'identificador de l'objecte dins d'un vector que conté tots els identificadors dels objectes afegits.
- Afegir una textura blava a l'entitat.

És important emmagatzemar les dades dins de l'entitat del scenegraph, ja que per calcular els vèrtexs en món reals dins de l'escena cal transformar els vèrtexs de món de l'objecte per les dades contingudes en la matriu de transformació del scenegraph (com s'ha explicat en l'apartat anterior). Aquests vèrtexs de món de l'objecte no pateixen cap transformació donat que tota la informació està inicialitzada a l'entitat.

En finalitzar la creació de tots els objectes és necessari computar les transformacions de tots els objectes del scenegraph. És a dir, a partir de les dades que s'han afegit per cada objecte (setPosition, setScale i setOrientation) s'emmagatzemen vèrtexs locals, la translació, el quaternió i els vèrtexs de món de l'objecte dins del scenegraph.

13.3.3.2 Definir els inputs de teclat

Com a base el motor ja disposa d'unes tecles reservades per traslladar la càmera en qualsevol dels tres eixos de coordenades.

Per poder moure les entitats (s'ha programat per a poder moure una de les entitats) s'han definit 6 tecles que permetin traslladar lliurement l'entitat en qualsevol dels tres eixos de coordenades.

D'aquesta manera s'han pogut realitzar les proves per comprovar el correcte funcionament de la pipe.

Com a optimització s'ha creat un booleà *moved* que s'inicialitza a true en el moment en el qual es clica alguna de les 6 tecles. D'aquesta manera el framework comprova les col·lisions en aquell frame només quan s'ha produït un canvi a l'escena.

13.3.3.3 Executar el codi en bucle

El main disposa d'un bucle infinit que es va executant constantment. Cada execució del bucle és considerat un frame.

Dins del bucle es tornen a computar les transformacions de les dades del scenegraph per si alguna entitat ha patit algun canvi.

En el moment en el qual es clica alguna de les 6 tecles per moure l'objecte s'activa el booleà *moved* i es realitza la comprovació de les col·lisions. En cas contrari, aquesta comprovació no es realitzaria. Perquè la pipe disposi de les dades reals després d'haver-se clicat la tecla, és necessari recalcular les dades dels sets, ja que en produir-se una translació cal obtenir els nous valors.

Després cal definir quina tipologia de càlcul es vol realitzar, es pot triar entre les 6 opcions requerides:

- Els objectes de l'escena contra els objectes de l'escena
- Els objectes amb tag *tag1* contra els objectes amb tag *tag2.*
- Els objectes amb grup *group1* contra els objectes amb grup *group2.*
- L'objecte *obj* contra els altres objectes de l'escena.
- L'objecte *obj* contra els objectes amb tag *tag.*
- L'objecte *obj* contra els objectes amb grup *group.*

Un cop s'ha executat la pipe (amb la inicialització pertinent dels arrays dels dos sets que contenen les dades dels objectes a calcular) s'ha realitzat un canvi de textura en aquells objectes dels quals s'ha detectat col·lisió. Per fer-ho ha sigut necessari buscar quin era l'identificador dins del scenegraph per cadascun dels identificadors dels objectes, ja que el canvi de textura s'ha de realitzar sobre l'entitat.

Així doncs primer s'han canviat totes les textures de tots els objectes de l'escena a blau i després s'han recorregut els resultats amb els identificadors dels objectes que col·lisionen i s'han canviat les textures a vermell.

Finalment abans de finalitzar el bucle s'ha tornat a inicialitzar el booleà *moved* com a false per a preparar-lo per a la següent iteració.

14 Pla de test

Tot sistema software ha de sotmetre's a una fase de proves per tal d'assegurar el bon funcionament de les funcionalitats i components del sistema.

Les proves a les quals s'ha de sotmetre el software són molt diverses: hi ha proves automatitzades, manuals i exploratòries. Es poden definir tres modalitats de proves: unitàries, d'integració i funcionals.

- Les proves unitàries validen els components més petits del sistema. Es proven individualment les classes i es comprova que les entrades i sortides són correctes sempre.
- Les proves d'integració comproven que el sistema està totalment integrat i que per tant, pot treballar com a conjunt.
- Les proves funcionals verifiquen totes les funcionalitats amb què l'usuari interactuarà i comproven cas a cas que es produeix el resultat esperat.

Les proves permeten:

- Trobar defectes en el producte software de forma ràpida
- Verificar que el software compleix tots els requisits
- Verificar la qualitat del software
- Minimitzar el manteniment del software i disminuir els costos de suport
- Disminuir els riscos de desenvolupament posteriors en cas de ser un projecte escalable. Evitant pujar a entorns de producció versions que no compleixen els requisits

Aquestes proves, a més a més, són més productives quan són dutes a terme per un equip d'especialistes en comprovació de software. Aquests aporten una visió més àmplia i empàtica amb els usuaris i ajuden a identificar problemes que els desenvolupadors han passat per alt. Per aquest motiu durant les diferents fases s'ha tingut en compte els consells dels consultors a l'hora d'afegir proves.

14.1 Test Plan

Per tal de garantir el correcte funcionament del software, s'ha dissenyat un seguit de proves al qual s'ha sotmès el motor.

A continuació s'enumeren els criteris establerts per al bon funcionament de les proves:

- Satisfer les proves del sistema garanteix l'obtenció dels criteris desitjats tant pels usuaris com pel client
- Les proves es realitzaran sempre en les mateixes condicions de treball dins de les variables possibles.
- Els tests de prova contemplaran tots els escenaris possibles.
- La superació d'una prova no implica la fiabilitat absoluta, per tant, periòdicament es tornaran a comprovar les proves anteriorment superades.

Al llarg d'aquest pla de proves, es descriuen les proves que s'han dut a terme durant la implementació de les diferents tècniques. Les proves cobriran el testimoni funcional del sistema de tots els requisits especificats a la memòria i als models corresponents.

Els destinataris de les proves han sigut el desenvolupador i el tester que s'han encarregat de modificar tot el que fos necessari per assolir el compliment de tots els tests.

14.1.1 Identificació i justificació de les proves

Amb les proves de testeig es vol comprovar que el motor garanteix el següent:

14.1.1.1 Proves funcionals

- El motor ha de funcionar sense fer ús de cap motor extern.
- Els usuaris poden calcular totes les col·lisions entre tots els objectes afegits al motor.
- Els usuaris poden calcular totes les col·lisions entre dos conjunts d'objectes prèviament inserits al motor.
- Els usuaris han de poder treballar amb objectes 2D i 3D.
- Els usuaris han de poder treballar amb objectes alineats i girats.
- El motor han de poder treballar amb objectes convexos.
- Els usuaris han de poder definir la informació de translació, rotació i escalat dels objectes.

- Els usuaris han de poder afegir objectes al motor.
- Els usuaris han de poder modificar objectes del motor (translació, rotació, escalat, tag, grup).
- Els usuaris poden calcular totes les col·lisions entre els objectes amb tag *tag1* i els objectes amb tag *tag2.*
- Els usuaris poden calcular totes les col·lisions entre els objectes amb grup *group1* i els objectes amb grup *group2.*
- Els usuaris poden calcular totes les col·lisions entre un objecte *obj* i els altres objectes inserits al motor.
- Els usuaris poden calcular totes les col·lisions entre un objecte *obj* i els objectes amb tag *tag1.*
- Els usuaris poden calcular totes les col·lisions entre un objecte *obj* i els objectes amb grup *group1.*

14.1.1.2 Proves de rendiment

- El motor ha de poder calcular totes les col·lisions que es donin entre centenars d'objectes en menys de 3-4 mil·lisegons.
- El motor ha de poder suportar escenes amb 1000 objectes.
- El motor ha de ser totalment escalable.

14.1.1.3 Proves operacionals

- El motor com a component del framework ha de poder ser usat en plataformes Android, iOS i web.
- El motor com a component del framework ha de poder funcionar en dispositius Android amb un sistema operatiu igual o superior a 5.
- El motor com a component del framework ha de poder funcionar en dispositius iOS amb sistema operatiu igual o superior a la versió 9.
- El motor com a component del framework ha de poder funcionar des de diversos navegadors Firefox, Chrome i Internet Explorer.

14.1.1.4 Proves de dades

- El motor ha de rebre la informació dels objectes en el format correcte.
- El motor ha de rebre quins objectes formen cadascun dels conjunts.

14.1.2 Realització de les proves

A continuació es descriuen quines tècniques s'han fet servir per testejar el projecte.

14.1.2.1 Proves funcionals

Les proves unitàries de Caixa blanca és una forma per provar el bon funcionament de mòduls o operacions (i especialment operacions matemàtiques), on a partir d'unes entrades s'han d'obtenir unes sortides.

Les operacions funcionen correctament si es compleix la sortida prevista. Aquesta tècnica depèn de com està programada cada operació o mòdul.

Tècnica 1

Objectiu de la tècnica	Comprovar el correcte funcionament de totes les operacions. Garantir el correcte funcionament de les funcionalitats.
Tècnica	Proves unitàries: Caixa blanca Executar cadascun dels casos amb dades per confirmar els resultats esperats quan s'introdueixen dades vàlides.
Predicció	El tester comprova les operacions implementades a través de proves unitàries.
Eines necessàries	Ordinador amb Visual Studio amb el codi del projecte.
Criteris de satisfacció	El 100% de les proves unitàries obté la sortida esperada.
Altres consideracions	La bateria de proves ha de servir per a totes les tècniques i dissenys.

14.1.2.2 Proves de rendiment

Tècnica 1

Objectiu de la tècnica	Comprovar que el motor pot calcular totes les col·lisions que es donin entre centenars d'objectes en menys de 3-4 mil·lisegons.
Tècnica	Proves de rendiment
Predicció	El tester comprova que el motor calcula les col·lisions en menys de 3-4 ms per a 100 objectes.
Eines necessàries	Ordinador amb Visual Studio amb el codi del projecte.
Criteris de satisfacció	Es poden calcular totes les col·lisions entre 100 objectes en menys de 3-4 ms.
Altres consideracions	

Tècnica 2

Objectiu de la tècnica	El motor ha de poder suportar escenes amb 1000 objectes.
Tècnica	Proves de rendiment
Predicció	El tester comprova que el motor suporta 1000 objectes.
Eines necessàries	Ordinador amb Visual Studio amb el codi del projecte.
Criteris de satisfacció	Es poden arribar a inserir i calcular col·lisions per a 1000 objectes.
Altres consideracions	

Tècnica 3

Objectiu de la tècnica	Comprovar que el disseny sigui escalable.
Tècnica	Proves d'escalabilitat
Predicció	El tester comprova que el sistema està fet de manera que es puguin afegir nous components fàcilment.
Eines necessàries	Ordinador amb Visual Studio amb el codi del projecte.
Criteris de satisfacció	El disseny ha de ser escalable i permetre l'annexió de noves tècniques com de nous tests.
Altres consideracions	

14.1.2.3 Proves operacionals

Tècnica 1

Objectiu de la tècnica	Comprovar que el framework funciona correctament en les diferents versions d'Android, iOS i web definides.
Tècnica	Realitzar proves en les diferents versions.
Predicció	Per a realitzar les proves en Android serà necessari compilar el codi a javaScript fent ús de la llibreria Android NDK. Per a realitzar les proves en iOS serà necessari compilar el codi a Objective-C fent ús de LLVM. Per a realitzar les proves en web serà necessari compilar el codi a emscripten fent ús de LLVM.
Eines necessàries	Ordinador amb Visual Studio amb el codi del projecte. Dispositius Android i iOS pertinents.
Criteris de satisfacció	És posible interactuar amb el framework en tots els dispositius.
Altres consideracions	

14.1.2.4 Proves de dades

Tècnica 1

Objectiu de la tècnica	Comprovar l'entrada de dades.
Tècnica	Es comprovarà que les dades que rebi el motor compleixen la definició requerida.
Predicció	El tester comprova les operacions implementades a través de proves unitàries.
Eines necessàries	Ordinador amb Visual Studio amb el codi del projecte.
Criteris de satisfacció	Les dades que rebi el motor seran prèviament comprovades pel framework. En cas que les dades siguin errònies els resultats esperats donaran error.
Altres consideracions	

14.1.3 Necessitats de l'entorn

A continuació es detallen les necessitats de l'entorn per a fer les proves:

14.1.3.1 Hardware fonamental

RECURS	QUANTITAT	DESCRIPCIÓ
PC	1	HP i7-4510 12GB RAM
Pantalles	1	Samsung 24''
Dispositius Android	3	Android de diferents mides i versions de sistemes operatius.
Dispositius iOS	3	iOS de diferents mides i versions de sistemes operatius.

Taula 58: Descripció del hardware necessari per a les proves

14.1.3.2 Software fonamental

NOM	VERSIÓ
Visual Studio	2015
Chrome	57
Firefox	52
Internet Explorer	10, 11[8]

Taula 59: Descripció de software necessari per a les proves

[8] Dades extretes de http://caniuse.com/#search=webgl

14.1.3.3 Configuració de l'entorn de proves (pantalles)

MARCA	PÍXELS
Samsung	<768
Samsung	[768-992]
Samsung	[992-1200]
Samsung	>1200

Taula 60: Descripció de les pantalles necessàries per a les proves

14.1.3.4 Configuració de l'entorn de proves (dispositius mòbils)

SISTEMA OPERATIU	VERSIÓ	DIMENSIÓ
Android	5.0	7"
Android	5.1.1	8"
Android	6.0	10"
iOS	9.0	4.7"
iOS	9.1	5.5"
iOS	9.2	12.9"

Taula 61: Descripció dels dispositius Android i iOS necessari per a les proves

14.2 Test Case

Un **test case** és un document on s'especifiquen els valors d'entrada, els valors esperats de sortida i les precondicions per tal de poder executar el test. L'Estàndard IEEE 610 de 1990 defineix el test case com: *"A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement."*. [9]

14.2.1 Proves funcionals

El conjunt de proves a fer per a assegurar el correcte funcionament de les tècniques era infinit tenint en compte que es disposa de llibertat d'escalat, de rotació i de translació en els tres eixos de coordenades.

Per aquest motiu s'ha decidit fer un primer bloc de proves mitjançant una bateria de **tests unitaris**[40] que permeti poder "assegurar" el correcte funcionament de les tècniques implementades i un cop és realitzes la implantació fer un testeig més exhaustius per assegurar al 100% el seu funcionament.

[9] Definició extreta de: http://www.kaner.com/pdfs/GoodTest.pdf

14.2.1.1 Bateria de tests unitaris

Per a realitzar les primeres proves funcionals s'ha creat una bateria de tests unitaris.

Aquesta bateria de test s'ha creat a la vegada que es realitzaven les fases d'implementació de les tècniques. S'ha fet servir per a posar a prova les tècniques, alguns dels tests s'han creat per comprovar situacions límit (com el cas que la zona de col·lisió sigui un únic punt) o situacions en les quals es creia que alguna tècnica podria fallar (com és el cas del test 8).

A la vegada ha servit per a una banda per millorar el codi en cas que algun dels tests donés un resultat erroni o per trobar limitacions que tinguessin les tècniques.

TEST	ENTRADA		SORTIDA
Test1	**B1** Posició: 1.5f, 1.0f, 0.0f Mida: 0.5f, 0.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 1.0f, 1.0f, 0.0f Mida: 1.5f, 1.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test2	**B1** Posició: 1.5f, 1.0f, 0.0f Mida: 1.0f, 1.0f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 1.0f, 1.0f, 0.0f Mida: 3.0f, 3.0f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test3	**B1** Posició: 1.0f, 1.0f, 0.0f Mida: 2.0f, 3.0f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 2.0f, 1.0f, 0.0f Mida: 1.0f, 1.0f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test4	**B1** Posició: 2.0f, 3.0f, 0.0f Mida: 0.5f, 0.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 2.0f, 2.0f, 0.0f Mida: 1.5f, 1.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test5	**B1** Posició: 1.5f, 0.75f, 0.0f Mida: 1.25f, 1.75f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 0.75f, 0.75f, 0.0f Mida: 0.25f, 0.25f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test6	**B1** Posició: 0.75f, 1.75f, 0.0f Mida: 1.5f, 1.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 1.75f, 0.75f, 0.0f Mida: 1.5f, 1.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test7	**B1** Posició: -1.5f, -1.75f, 0.0f Mida: 2.0f, 1.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: -1.5f, -1.0f, 0.0f Mida: 1.0f, 1.0f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test8	**B1** Posició: -1.25f, 1.0f, 0.0f Mida: 0.75f, 0.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: -1.25f, 0.75f, 0.0f Mida: 0.25f, 1.25f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test9	**B1** Posició: -1.25f, 1.0f, 0.0f Mida: 3.0f, 1.0f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: -1.25f, 1.25f, 0.0f Mida: 1.0f, 2.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test10	**B1**	**B2**	No

	Posició: 1.0f, -1.0f, 0.0f Mida: 0.5f, 0.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Posició: 2.5f, -2.5f, 0.0f Mida: 0.5f, 0.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	col·lisió
Test11	**B1** Posició: 1.0f, 1.0f, 0.0f Mida: 2.0f, 3.0f, 0.0f Rotació: 0.0f, 0.0f, 45.0f	**B2** Posició: 2.5f, -2.5f, 0.0f Mida: 1.0f, 1.0f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	No col·lisió
Test12	**B1** Posició: 1.0f, 1.0f, 0.0f Mida: 1.5f, 1.5f, 0.0f Rotació: 0.0f, 0.0f, 45.0f	**B2** Posició: 0.5f, -0.5f, 0.0f Mida: 1.5f, 1.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test13	**B1** Posició: 1.5f, 1.0f, 0.0f Mida: 1.5f, 1.5f, 0.0f Rotació: 0.0f, 0.0f, 45.0f	**B2** Posició: 0.5f, -0.5f, 0.0f Mida: 1.5f, 1.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test14	**B1** Posició: 1.75f, -0.5f, 0.0f Mida: 1.5f, 1.5f, 0.0f Rotació: 0.0f, 0.0f, 45.0f	**B2** Posició: 0.5f, -0.5f, 0.0f Mida: 1.5f, 1.5f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test15	**B1** Posició: 2.0f, 2.25f, 1.75f Mida: 1.0f, 0.5f, 0.5f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 2.0f, 1.0f, 1.5f Mida: 2.0f, 1.0f, 1.0f Rotació: 0.0f, 0.0f, 0.0f	No col·lisió
Test16	**B1** Posició: 3.5f, 0.5f, 1.5f Mida: 1.0f, 1.0f, 2.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 3.75f, 1.0f, 0.375f Mida: 0.75f, 2.0f, 0.75f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test17	**B1** Posició: 2.5f, 1.5f, 0.0f Mida: 5.0f, 3.0f, 2.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 0.0f, 0.0f, 0.0f Mida: 4.0f, 4.0f, 2.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test18	**B1** Posició: 0.0f, 0.0f, 0.0f Mida: 2.0f, 2.0f, 2.0f Rotació: 0.0f, 0.0f, 45.0f	**B2** Posició: 1.4f, 1.4f, 0.0f Mida: 2.0f, 3.0f, 2.0f Rotació: 0.0f, 0.0f, 45.0f	Col·lisió
Test19	**B1** Posició: 0.0f, 0.0f, 0.0f Mida: 4.0f, 4.0f, 2.0f Rotació: 0.0f, 0.0f, 33.0f	**B2** Posició: 0.0f, 0.0f, 0.0f Mida: 4.0f, 4.0f, 2.0f Rotació: 0.0f, 0.0f, 45.0f	Col·lisió
Test20	**B1** Posició: 0.0f, 0.0f, 0.0f Mida: 4.0f, 4.0f, 1.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 0.0f, 0.0f, 2.0f Mida: 4.0f, 4.0f, 1.0f Rotació: 0.0f, 0.0f, 0.0f	No col·lisió
Test21	**B1** Posició: 0.0f, 0.0f, 0.0f Mida: 1.0f, 1.0f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 0.0f, 0.0f, 0.0f Mida: 1.0f, 1.0f, 0.0f Rotació: 0.0f, 90.0f, 0.0f	Col·lisió
Test22	**B1** Posició: 0.0f, 0.0f, 0.0f Mida: 1.0f, 1.0f, 0.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 0.0f, 0.0f, 0.0f Mida: 0.0f, 1.0f, 1.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test23	**B1** Posició: 3.5f, 3.5f, 3.5f Mida: 3.0f, 3.0f, 3.0f	**B2** Posició: 3.5f, 5.5f, 5.5f Mida: 3.0f, 3.0f, 3.0f	Col·lisió

	Rotació: 0.0f, 0.0f, 0.0f	Rotació: 0.0f, 0.0f, 0.0f	
Test24	**B1** Posició: 3.0f, 3.0f, 3.0f Mida: 2.0f, 2.0f, 2.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 3.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test25	**B1** Posició: 3.0f, 5.0f, 3.0f Mida: 2.0f, 2.0f, 2.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 3.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test26	**B1** Posició: 5.0f, 5.0f, 5.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 3.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test27	**B1** Posició: 6.0f, 6.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 3.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test28	**B1** Posició: 3.0f, 6.0f, 6.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 3.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test29	**B1** Posició: 3.0f, 6.0f, 6.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 2.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test30	**B1** Posició: 8.0f, 3.0f, 8.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 3.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	No col·lisió
Test31	**B1** Posició: 3.0f, 4.0f, 3.0f Mida: 0.5f, 0.5f, 0.5f Rotació: 45.0f, 0.0f, 45.0f	**B2** Posició: 3.0f, 4.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 45.0f, 0.0f, 45.0f	Col·lisió
Test32	**B1** Posició: 7.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	**B2** Posició: 3.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 0.0f, 0.0f, 0.0f	Col·lisió
Test33	**B1** Posició: 3.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 45.0f, 0.0f, 0.0f	**B2** Posició: 3.0f, 5.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 45.0f, 0.0f, 0.0f	Col·lisió
Test34	**B1** Posició: 3.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 45.0f, 0.0f, 0.0f	**B2** Posició: 7.0f, 7.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 45.0f, 0.0f, 0.0f	Col·lisió
Test35	**B1** Posició: 3.0f, 7.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 45.0f, 0.0f, 0.0f	**B2** Posició: 3.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 45.0f, 0.0f, 0.0f	Col·lisió
Test36	**B1** Posició: 7.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 45.0f, 0.0f, 0.0f	**B2** Posició: 3.0f, 3.0f, 3.0f Mida: 4.0f, 4.0f, 4.0f Rotació: 45.0f, 0.0f, 0.0f	Col·lisió

Taula 62: Bateria de tests unitaris

A *l'Annex 6* hi ha una taula amb els resultats d'aplicar la bateria de tests a les 16 pipes.

14.2.1.2 Proves amb el framework

Un cop realitzada la implantació de les pipes del GJK dins del framework s'han pogut realitzar totes aquelles proves que no es van poder fer amb la bateria de test donada la seva complexitat per aconseguir comprovar tots els casos.
A més a més s'han realitzat les proves per comprovar els nous requisits funcionals definits durant la implantació.

14.2.1.2.1 Comprovar correcte funcionament de les pipes

Bàsicament el que s'ha volgut comprovar és que la detecció de la col·lisió fos considerada al píxel. Amb l'ajuda del visualitzador s'ha posat a prova el motor en diverses situacions de rotacions, escalats i posicions. En tots els casos el motor ha assolit la seva tasca a la perfecció.

També s'ha volgut comprovar si l'aplicar rotacions la col·lisió es realitza correctament. Donat que les dades de transformació són emmagatzemades dins de l'entitat i que la tècnica treballa amb les dades emmagatzemades dins de l'estructura d'arrays es podria haver donat el cas que a la visualització l'entitat sortís girada però les dades de l'objecte dins de l'estructura no tinguessin ben aplicada les transformacions del scenegraph.

A la imatge següent es mostren els casos que es podrien haver donat respecte a la informació emmagatzemada a l'estructura:

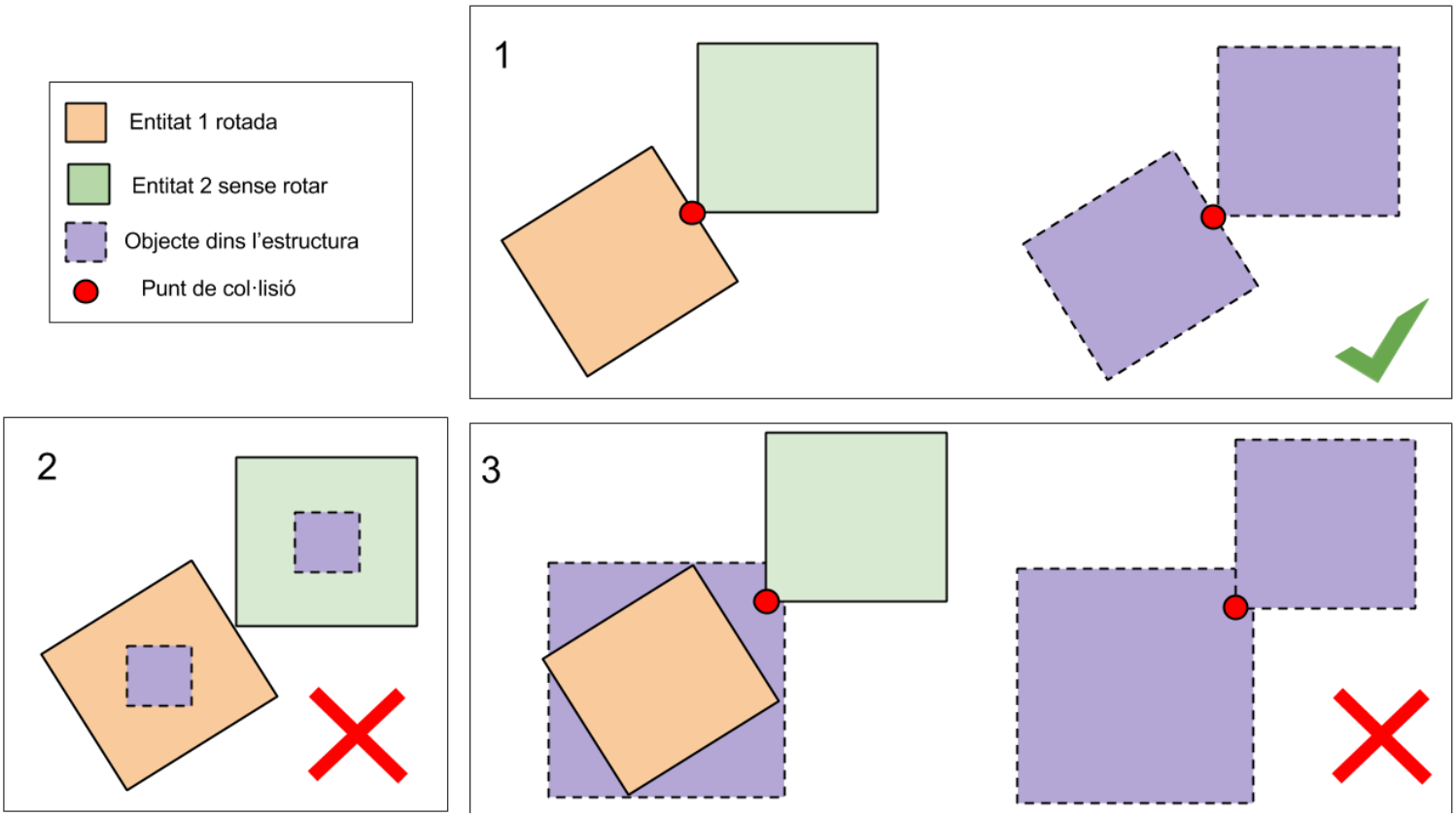


Figura 121: Exemple dels problemes que podrien haver sorgit amb la implantació

Cas 1: Entitats i estructura correctament inicialitzats:

Les entitats es visualitzen correctament (tenen la informació de les transformacions), en aquest cas l'objecte taronja està girat mentre que l'objecte verd no. Els punts de món reals (emmagatzemats a l'estructura) estan correctament calculats, a la imatge són els cubs en lila. Com que la pipe utilitza les dades de l'estructura, es calcula correctament la col·lisió.

Cas 2: Entitats correctes però estructura incorrecta:

Les entitats es visualitzen correctament (tenen la informació de les transformacions), en aquest cas l'objecte taronja està girat mentre que l'objecte verd no. Però en aquest cas als punts de l'estructura no se'ls ha aplicat la informació de transformació del scenegraph i per tant no es calcula la col·lisió, ja que entre els dos cubs morats no n'hi ha.

Cas 3: Entitats correctes però estructura parcialment incorrecte:

Les entitats es visualitzen correctament (tenen la informació de les transformacions), en aquest cas l'objecte taronja està girat mentre que l'objecte verd no. Però en aquest cas als punts de l'estructura no contenent tota la informació de transformació del

scenegraph (només han aplicat una part, en aquest cas s'aplica l'escalat però no la rotació) i per tant la col·lisió que es calcula és errònia.

S'han fet diverses proves per contemplar tots tres casos i no s'ha trobat en cap moment les situacions 2 i 3. Així doncs es considera que la implantació funciona correctament.

A continuació s'han afegit dos exemples on es pot comprovar que la detecció de la col·lisió varia segons el píxel de separació. A les dues imatges superiors (blaves) no es produeix cap contacte mentre que a les imatges inferiors (vermelles) sí que hi ha un petit contacte.

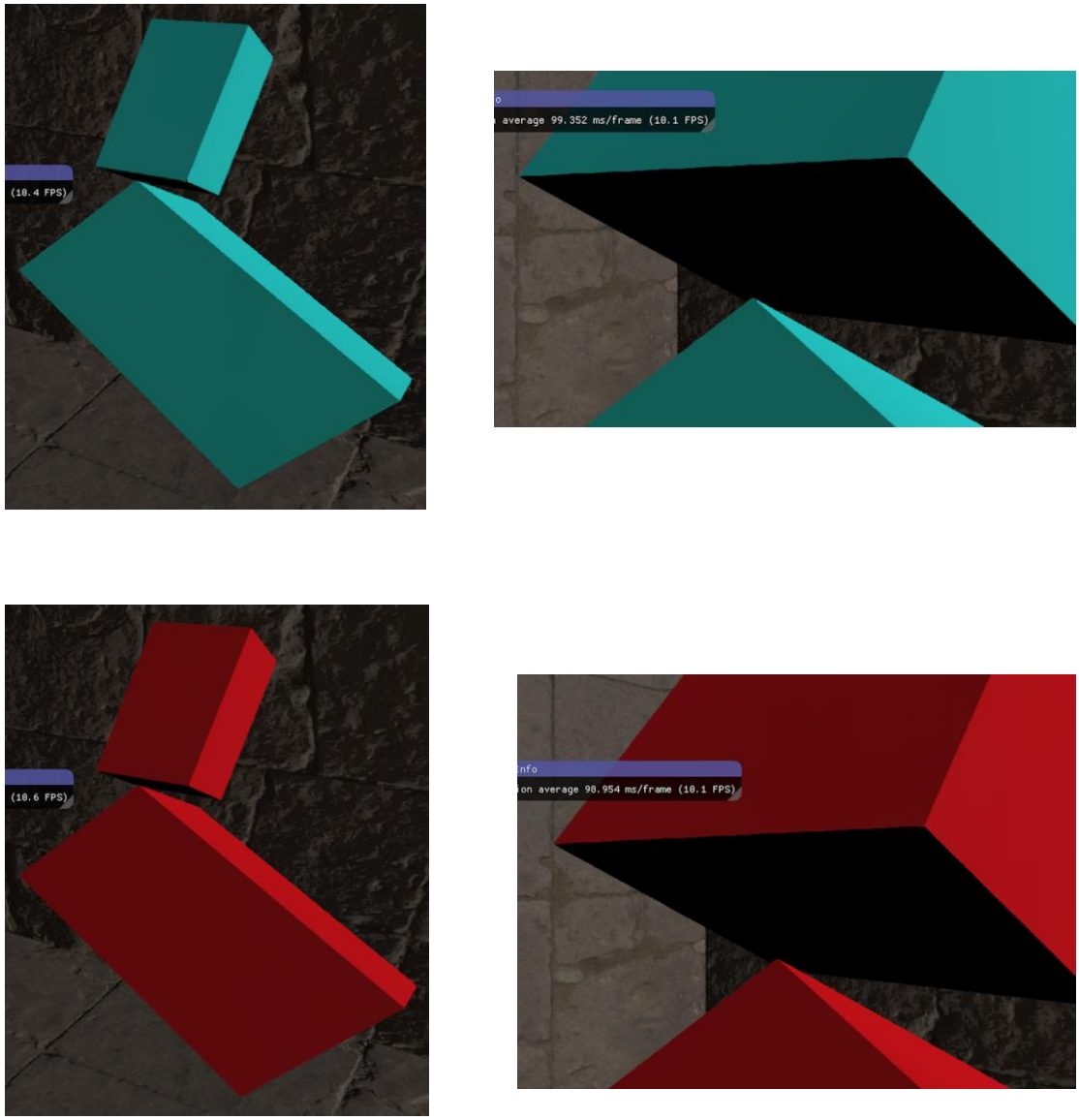


Figura 122: Exemples de deteccions dins del framework. A la part superior: No col·lisió. A la part inferior: col·lisió.

14.2.1.2.1 Comprovar requisits funcionals de la implantació

Donat que durant la realització de la implantació es van definir nous requisits ha sigut necessari realitzar les proves pertinents per comprovar el seu compliment.

A continuació es descriuen els objectes inserits per realitzar les proves:

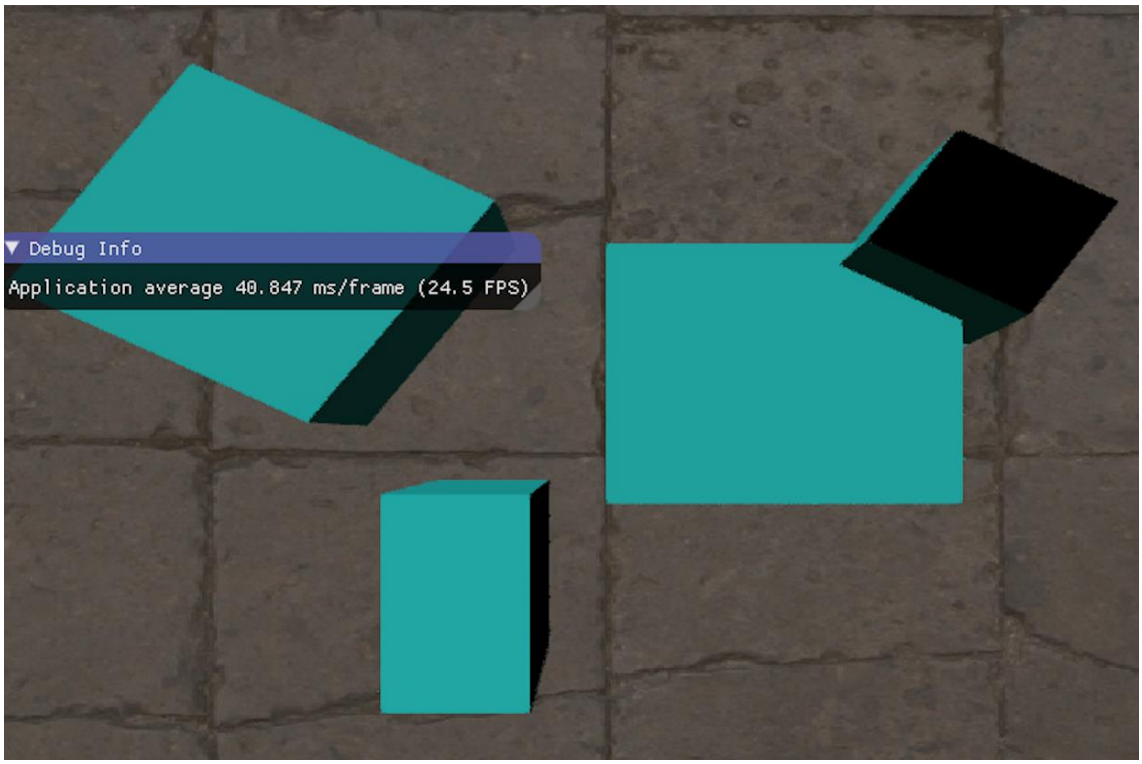


Figura 123: Il·lustració dels objectes utilitzats en les proves dins del framework

	B1	B2	B3	B4
POSICIÓ	-2.0f, 2.0f, 1.0f	1.0f, 1.0f, 1.0f	2.0f, 2.0f, 2.0f	-1.0f, -1.0f, -1.0f
ESCALAT	2.0f, 2.0f, 2.0f	2.0f, 2.0f, 2.0f	1.0f, 1.0f, 1.0f	1.0f, 2.0f, 3.0f
ROTACIÓ	0.0f, 0.0f, 45.0f	0.0f, 0.0f, 0.0f	0.0f, 30.0f, 45.0f	0.0f, 0.0f, 0.0f
TAG	3	1	3	3
GRUP	1	4	1	4

Taula 64: Entrada de dades per a les proves dins del framework

A la taula següent es descriuen per cadascun dels requisits requerits un exemple d'entrada i el llistat d'objectes que han de col·lisionar:

TEST	ENTRADA	SORTIDA
ScenaVsScena	--	B1/B2 B1/B3 B1/B4 B2/B3
TagVsTag	3, 3	B1/B3 B1/B4 B3/B4
GroupVsGroup	1, 4	B1/B2 B1/B4 B3/B2 B3/B4
ObjectVsScena	0	B1/B2 B1/B3 B1/B4
ObjectVsTag	0, 3	B1/B3 B1/B4
ObjectVsGroup	0, 4	B1/B2 B1/B4

Taula 65: Bateria de test unitaris per a les proves dins del framework

Tots els tests s'han assolit correctament.

14.2.2 Proves de rendiment

14.2.2.1 Comprovar que el temps d'execució per calcular les col·lisions amb 100 objectes no trigui més de 3-4 ms

Per realitzar aquesta comprovació s'ha realitzat l'execució de les diferents pipes amb el mateix entorn i entrada de dades.

S'han creat de manera aleatòria 100 objectes amb les següents característiques:

- Mida aleatòria entre 0 i 5 unitats
- Posició aleatòria entre -2 i 2 unitats
- Rotació aleatòria entre 0 i 360 graus

Amb aquestes proves s'ha comprovat que amb 100 objectes les dues pipes seleccionades compleixen el requisit.

A l'apartat "*12-Resultats*" es descriuen les proves i els resultats obtinguts.

14.2.2.2 Comprovar que el motor pot treballar amb 1000 objectes

Per realitzar aquesta comprovació s'han inserit 1000 objectes i s'ha comprovat que el motor pot realitzar el càlcul de totes les col·lisions.

Destacar que el temps requerit per realitzar el càlcul de totes de les col·lisions per 1000 objectes és molt superior al requeriment de 3-4 ms. Aproximadament per calcular les 500.000 combinacions la pipe SC+GJK ha trigat més de 130 ms.

Com a tasca futura s'estudiarà com millorar aquests resultats.

14.2.2.3 Comprovar que el motor és escalable

Aquest requisit s'ha assolit durant el projecte donat que el disseny realitzat tant per a l'opció orientada a objectes com per l'orientada a dades s'ha realitzat de manera que es puguin afegir noves tècniques i pipes fàcilment.

14.2.3 Proves operacionals

14.2.3.1 Comprovar que el framework funciona correctament en les diferents versions d'Android, iOS i web definides

Per realitzar aquesta comprovació és necessari compilar el codi actual realitzat en c++ als llenguatges requerits pels diferents entorns i dispositius.
Per a poder dur a terme les proves en els dispositius Android serà necessari compilar el codi a javaScript fent ús de la llibreria Android NDK. Per a fer les proves en iOS serà necessari compilar el codi a Objective-C fent ús de LLVM. Per a realitzar les proves en web serà necessari compilar el codi a emscripten fent ús de LLVM.

Donat que l'estat del framework encara és embrionari i no s'han pogut realitzar les tasques de compilació en les diferents versions aquest requisit no s'ha pogut comprovar.

Tot i així s'espera que els problemes que puguin sorgir en realitzar les compilacions no comportin molts problemes i es pugui assolir finalment aquest requisit.

14.2.4 Proves de dades

En realitzar les proves amb el framework s'ha comprovat que si les dades compleixen la definició correcta, llavors el motor assoleix correctament dels càlculs.

Actualment les proves que s'han realitzat han sigut definides a mà però en un futur serà necessari que la informació de cadascun dels objectes estigui ben creada, ja que en cas contrari el motor no podrà realitzar els càlculs correctament.

15 Planificació final

La planificació inicial ha patit canvis a causa dels obstacles ja previstos.

La part d'inexperiència ha afectat la tasca 4.9, el pas de la tècnica de GJK a un disseny orientat a dades, en un increment del temps de 60 hores. Això ha afectat la planificació de les posteriors tasques.

Un cop es va escollir el disseny i tècnica que es faria ús dins del motor, va arribar el moment de dur a terme la implantació dins del framework. Per aquestes dates es van produir dues situacions pels quals el projecte no es va poder finalitzar a temps.
Per un costat el framework en el qual s'hi havia de realitzat la implantació encara estava en un estat molt inicial, i no es podria realitzar la tasca en aquell moment. Per l'altra banda, va entrar un nou projecte altament prioritari. El conjunt de les dues situacions va afectar que la fase d'implantació del motor dins del framework no es pogués dur a terme segons la planificació inicial.

Per part de l'empresa es va decidir que, tot i no haver-se produït la fase d'implantació, el projecte havia assolit els objectius principals, que consistien a escollir una tècnica i un disseny perquè el framework en fes ús per a detectar les col·lisions. Així doncs es va donar per tancat el projecte, els recursos van passar a treballar en un nou projecte i es va dir que quan el framework estigues en un estat més madur ja es faria la implantació (per part d'algun altre treballador o per part meva).

En un primer moment es va optar per seguir amb la lectura del projecte per a la convocatòria de gener però per diversos motius es va acabar ajornant la lectura fins al torn d'abril.

Donat aquest marge de temps, es va parlar amb l'empresa per saber si abans de l'abril seria possible realitzar aquesta implantació. Finalment al març es va acordar que, fora de la jornada laboral es podria realitzar aquesta implantació.

El diagrama de Gantt modificat amb la planificació definitiva s'adjunta a l'Annex 7.

16 Pressupost final

A l'apartat de *Planificació final* s'expliquen els diferents canvis que s'han produït en la planificació del projecte. L'únic canvi que ha provocat un increment del temps és la tasca 4.9, ja que la resta de tasques, tot i realitzar-se posteriorment no han patit cap més canvi.

En resum, la tasca ha patit un increment de 60 hores que ha afectat a les hores de l'analista, el desenvolupador, el consultor i el tester. L'analista ha vist incrementat el seu temps de dedicació al projecte en 13 hores, el desenvolupador en 31 hores, el consultor en 10 hores i el tester en 6 hores.

	PLANIFICACIÓ INICIAL	PLANIFICACIÓ FINAL
Costos directes	10.896,25€	11.688,75€
Costos indirectes	125,69€	244,58€
Contingència	1.653,29€	1.731,42€

Taula 66: Diferència de costos entre planificació inicial i final

16.1 Justificació dels càlculs

Segons la definició realitzada a l'apartat *5.2.3-Explicació dels càlculs de les desviacions* a continuació es realitzen els càlculs dels desviaments produïts.

16.1.1 Increment en els costos directes

JUSTIFICACIÓ	PREU PER HORA	HORES	COST
preu (analista) x hores (dedicades de més l'analista)	15€	13 hores	195,00€
preu (desenvolupador) x hores (dedicades de més el desenvolupador)	12.5€	31 hores	387,50€
preu (consultor) x hores (dedicades de més el consultor)	15€	10 hores	150,00€
preu (tester) x hores (dedicades de més el tester)	10€	6 hores	60,00€
TOTAL			**792,50€**

Taula 67: Justificació dels càlculs de l'increment dels costos directes

16.1.2 Increment en els costos indirectes

PROGRAMARI	JUSTIFICACIÓ	COST
Visual Studio	[454,54€ (subscripció anual) x 0,25 x **378,1 hores** (temps desenvolupador i tester)] / (4 hores * 365 dies)	29.42€
Bitbucket	[600€ (50€ mensuals per 50 usuaris) x 0.25 x **378.1 hores** (temps desenvolupador i tester)] / [(4 hores * 365 dies) * (50 usuaris)]	0.78€
TOTAL		**30.20€**

Taula 68: Justificació dels càlculs de l'increment dels costos indirectes de programari

MAQUINARI	JUSTIFICACIÓ	COST
Analista	[700€ x 0,25 x **177 hores**] / 4 hores * 365 dies	21,21€
Desenvolupador	[700€ x 0,25 x **189,3 hores**] / 4 hores * 365 dies	22,69€
Consultor	[700€ x 0,25 x **188 hores**] / 4 hores * 365 dies	22,50€
Tester	[700€ x 0,25 x **186 hores**] / 4 hores * 365 dies	22.29€
TOTAL		**88,69€**

Taula 69: Justificació dels càlculs de l'increment dels costos indirectes de maquinari

16.1.3 Increment en la contingència

(11.688,75€ + 244,58€) x 0,15 **= 1.789,99€**

La **desviació total** és de **911,39€** i es podrà cobrir amb part de la quantitat destinada a contingència.

CONCEPTE	TOTAL
Costos directes	**11.688,75€**
Costos indirectes	**244,58€**
Contingència	**1.789,99€**
Imprevistos	**728,00€**
Total sense IVA	**14.451,32€**
TOTAL	**17.486,09€**

Taula 70: Resum dels costos finals del projecte

17 Treball futur

Aquest projecte es pot considerar com una primera versió del motor de col·lisions dins del framework. A partir d'aquest punt es realitzaran noves iteracions on es milloraran certs aspectes.

A continuació es llisten tres punts que caldria estudiar i realitzar.

17.1 Millores de rendiment

Donat els temps d'execució obtinguts en l'execució amb 1000 objectes es creu necessari fer un estudi de mètodes i una aplicació de mètodes per millorar el rendiment. A continuació es llisten tres punts com a possibles candidats a comprovar:

- Incorporar una estructura Bounding Volum Hierarchy (BVH).
- Incorporar instruccions Single instruction, multiple data (SIMD) que permetin paral·lelisme pel que fa a dades, permeten aplicar una sola instrucció sobre un conjunt de dades.
- Paral·lelitzar les comprovacions entre les parelles d'objectes. Fent ús d'una paral·lelització en CPU (threads) o paral·lelitzar en la GPU (CUDA, OpenGL, Computer Shaders).

17.2 Noves funcionalitats

- Fer que el GJK retorni la informació de col·lisió.

Dins d'aquest projecte la implementació del GJK només retornar si els objectes col·lisionaven o no. Com a següent pas caldria fer les modificacions pertinents per a poder retornar també les dades de la col·lisió per a poder realitzar la fase de "*resolució de la col·lisió*".

18 Conclusions

18.1 Objectius del projecte

Aquest projecte va néixer amb la intenció de realitzar un estudi de diferents tècniques de detecció de col·lisions i escollir quina era la més adient per a les necessitats de l'empresa. A més a més de cara a dissenyar un motor de col·lisions eficient, s'ha pogut realitzar la comparativa entre emprar un disseny orientat a objectes respecte a un disseny orientat a dades.

Durant les fases de recerca s'ha pogut escollir la millor solució (la pipe de SC+GJK en un disseny orientat a dades). I finalment s'ha implantat dins del framework com a component de col·lisions i s'han pogut realitzar totes les proves per assegurar el seu correcte funcionament.

Així doncs arribats a aquest punt es pot dir que s'han assolit els seus objectius i requisits especificats durant l'inici del projecte. Respecte a la planificació temporal s'han patit certes desviacions com s'han explicat a l'apartat "*15-Planificació final*" endarrerint la data de lectura del projecte. A més a més s'ha produït una desviació de 60 hores que han repercutit en el pressupost final.

18.2 Valoració personal

Aquest projecte m'ha brindat la possibilitat de treballar en un entorn completament diferent del que coneixia fins ara.

Per un costat el fet de treballar en una empresa m'ha mostrat un seguit de dificultats a les quals abans no havia prestat atenció. He treballat amb un negoci (l'empresa Interiorvista) que té un gran nombre de projectes i prioritats i això ha estat un gran obstacle, ja que he hagut de fer un seguiment per tal que el projecte no caigués en l'oblit i es pogués arribar a dur a terme en un futur pròxim. Alhora he treballat amb un gran nombre de projectes diferents mentre duia el projecte, cosa que m'ha dut a aprendre a estructurar-me el temps per tal de gestionar-los tots alhora.

Al llarg de tot el projecte m'he sentit en un ambient molt càlid a l'empresa. El director i els companys que han actuat com a consultors m'han ajudat a no encallar-me en cap moment i a raonar per tal que fos jo mateixa qui trobés les solucions més enriquidores per al projecte.

Per l'altre la temàtica del projecte. Tots els coneixements adquirits durant aquest temps. Fins fa un any no sabia la diferència entre un motor de físiques o un motor de col·lisions ni coneixia cap tècnica per a detectar col·lisions. Aquest projecte m'ha permès aplicar coneixements de la carrera dins del món real la qual cosa m'ha ajudat a tenir una visió completa de tot un projecte de software.

Finalment, dedicar unes paraules al meu director Marc Fernàrndez al qual agraeixo la seva paciència durant aquest any, i al meu ponent Josep Casanoves per guiar-me i animar-me durant tot el procés. Sense ells aquest projecte no s'haguera finalitzat amb èxit.

18.3 Integració de coneixements

La realització d'aquest projecte m'ha permès poder aplicar coneixements estudiats durant la carrera. A continuació es llisten les disciplines que m'han sigut útils per al projecte i els motius:

- Enginyeria de requisits [ER, PES]: necessària durant l'etapa de definició de requisits per determinar el sistema a construir. Poder formalitzar les necessitats de les persones interessades i de l'entorn en el qual haurà de funcionar el sistema.
- Gestió de projectes de Software [GPS]: necessària durant l'etapa de gestió del projecte per realitzar la planificació, el pressupost i la gestió de recursos.
- Arquitectura del software [AS]: necessària durant l'etapa de disseny, ha ajudat a idear els diferents dissenys fent ús de patrons de disseny i eines de modelatge basades en l'estàndard UML, que faciliten seguir bones pràctiques durant l'etapa d'implementació.
- Matemàtiques [M2]: necessària durant les etapes d'anàlisis i d'implementació per poder comprendre el funcionament de les diferents tècniques.
- Interacció i Disseny d'Interfícies [IDI]: necessària durant les etapes d'anàlisi i d'implementació per entendre el funcionament de les diferents tècniques.

- Paral·lelisme [PAR]: necessària per comprendre el funcionament del disseny orientat a dades.

La meva proposta de TFG engloba la gestió d'un projecte real, que treballa amb tecnologies recents. El TFG inclou les fases de: identificació del problema, requisits i objectius, especificació i disseny de l'aplicació usant patrons de disseny, implementació, testeig i implantació. Un recull d'informació que quedarà documentada.

El projecte integra moltes de les competències tècniques definides dins de l'enginyeria del software: S'ha desenvolupat un software a mida amb un previ disseny i especificació d'aquest. Ha estat una tasca molt important la valoració de les necessitats del client i l'especificació de requisits software per a la satisfacció d'aquestes necessitats. El software ha estat dissenyat de forma eficient i tenint en compte els requisits dels usuaris i considerant aspectes socials, legals i econòmics.

El projecte ha estat detallat a partir d'un conjunt de reunions amb el client que ens han permès extreure'n requisits funcionals i tècnics i alhora han estat claus per a identificar necessitats dels usuaris. El projecte es planteja com a objectiu escollir quina tècnica i disseny és més òptim per a les necessitats del framework de l'empresa.

Arribats a aquest punt del projecte, i donat que les meves funcions s'han anat clarificant al llarg del projecte, defineixo novament les competències tècniques que he treballat al llarg del meu projecte:

- **CES1.1:** Desenvolupar, mantenir i avaluar sistemes i serveis software complexos i/o crítics. [En profunditat]

Justificació: Aquest projecte ha consistit en el desenvolupament des de zero d'un motor de col·lisions que s'ha implantat en el framework que està realitzant l'empresa actualment. Per dur a terme la implementació del motor s'ha tingut en compte altres motors existents actualment dels quals s'ha realitzat la seva avaluació a l'estat de l'art, i dels que s'ha extret idees i tècniques, per aplicar al nostre motor. Caldrà realitzar el manteniment (que ja queda fora de l'abast d'aquest projecte).

- **CES1.2:** Donar solució a problemes d'integració en funció de les estratègies, dels estàndards i de les tecnologies disponibles. [Bastant]

Justificació: Cadascuna de les tècniques implementades requeria unes dades diferents. Es va realitzar el desenvolupament de cadascuna per separat i va ser necessari realitzar una integració dels dissenys i les funcions dels tres subprojectes. A més el projecte ha finalitzat amb la integració del motor dins del framework de l'empresa.

- **CES1.3:** Identificar, avaluar i gestionar els riscos potencials associats a la construcció de software que es poguessin presentar. [En profunditat]

Justificació: A l'inici del projecte es van llistar els riscos que podrien aparèixer durant el procés i es van contemplar alternatives a cadascun d'ells. Tot i així la implantació es va acabar posposant pels motius exposats anteriorment però finalment s'ha pogut realitzar.

- **CES1.7:** Controlar la qualitat i dissenyar proves en la producció de software. [En profunditat]

Justificació: Per poder confirmar que les diferents tècniques implementades funcionen correctament s'han realitzat una bateria de tests unitaris, composta per parelles d'objectes de diferents mides, posicions i rotacions. Donat que la casuística és infinita aquesta bateria només ha comprovat una fracció de les situacions i gràcies a la implantació que finalment s'ha pogut comprovar el correcte funcionament.

- **CES1.9:** Demostrar comprensió en la gestió i govern dels sistemes software. [En profunditat]

Justificació: Per dur a terme aquest projecte s'han tingut en compte les diferents fases en la gestió d'un sistema software, definició de requisits, objectius, disseny, implementació, testeig, implantació.

- **CES2.1:** Definir i gestionar els requisits d'un sistema software. [En profunditat]

Justificació: Abans de desenvolupar les diferents parts del projecte, es van definir els seus requisits. A més a més durant la fase d'implantació es van afegir nous requisits que es van haver de definir i gestionar.

19 Índex de figures i taules

19.1 índex de figures

19.2 Índex de taules

BIBLIOGRAFIA

[1] IDO LECHNER, PSFK. 5 Ways Technology Has Impacted Interior and Home Design. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://www.psfk.com/2014/09/5-ways-technology-impacts-home-interior-design.html>

[2] Interiorvista. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017]
Disponible a Internet: <https://www.interiorvista.com/ >

[3] Wikipedia. Physics engine. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: < https://es.wikipedia.org/wiki/Physics_engine >

[4] Wikipedia. Motores de juego. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: <https://es.wikipedia.org/wiki/Anexo:Motores_de_juego >

Jaime Sánchez-Valiente Blázquez Santiago Riera Almagro Matías Salinero Delgado. MOTOR DE FÍSICA MULTIFUNCIONAL[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://eprints.sim.ucm.es/11257/1/PROYECTO_DE_SISTEMAS_INFORM%C3%81TICOS(Final).pdf >

[5] Unity. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: <https://unity3d.com/es/unity>

Wikipedia: Unity. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: <https://es.wikipedia.org/wiki/Unity_(software) >

Andrew Wesson. Quora. Why don't use unity. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://www.quora.com/Why-dont-big-game-companies-use-Unity-for-their-game-development >

[6] BulletPhysics. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: < http://bulletphysics.org/wordpress/ >

Wikipedia: Bullet. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: < https://en.wikipedia.org/wiki/Bullet_(software) >

BulletPhysics team [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: <http://bulletphysics.org/wordpress/>

Unity. BulletPhysics for Unity. [en línia].

[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://forum.unity3d.com/threads/released-bullet-physics-for-unity.408154/>

[7] ODE. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: <http://www.ode.org/>

Wiki: ODE. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: <https://www.ode-wiki.org/wiki/ >

Wikipedia: ODE. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: < https://en.wikipedia.org/wiki/Open_Dynamics_Engine >

[8] PhysX. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://developer.nvidia.com/gameworks-physx-overview >

Wikipedia: PhysX. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: < https://en.wikipedia.org/wiki/PhysX >

[9] Havok. [en línia]
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: < http://www.havok.com/physics/ >

Wikipedia: Havok. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: < https://en.wikipedia.org/wiki/Havok_(software) >

[10] Unreal Engine. Distance Field Ambient Occlusion. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: <
https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAnd
Shadows/DistanceFieldAmbientOcclusion/ >

Computer Graphics. How does the Distance Field ambient Occlusion work in the Unreal Engine[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://www.quora.com/Computer-Graphics/Computer-Graphics-How-does-the-distance-field-ambient-occlusion-work-in-the-Unreal-Engine>

[11] Kah Shiu Chong. Evatotuts+. Collision Detection Using the Separating Axis Theorem. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: <
https://gamedevelopment.tutsplus.com/tutorials/collision- detection- using-the-separating-axis-theorem--gamedev-169 >

[12] Agencia Tributaria. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.agenciatributaria.es/static_files/AEAT/DIT/Contenidos_Publicos/CAT/AYUWEB/Biblioteca_Virtual/Manuales_practicos/Sociedades/Manual_Sociedades_2015.pdf >

[13] Microsoft Store. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
< https://www.microsoftstore.com/store/mseea/es_ES/list/Visual-Studio/categoryID.66235600?gclid=COCz8euFy88CFQd2cgodtxALnw&gclsrc=ds&tduid=(4be7f7d05f61d95b492f897fc8075c4e)(190947)(10203 26)(jkp_COCz8euFy88CFQd2cgodtxALnw)() >

Microsoft Azure. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: < https://azure.microsoft.com/enus/pricing/details/visual-studio-team-services/ >

[14] Nilson Souto. Video Game Physics Tutorial - Part II: Collision Detection for Solid Objects [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet: < https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects >

Euclidean Space. 3D Theory - Collision Detection. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.euclideanspace.com/threed/animation/collisiondetect/>

David Eberly. Dynamic Collision Detection using Oriented Bounding Boxes. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
< https://www.geometrictools.com/Documentation/DynamicCollisionDetection.pdf>

Gamasutra. Crashing into the New Year: Collision Detection. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.gamasutra.com/view/feature/3429/crashing_into_the_new_year_.php>

Nvidia. Thinking Parallel, Part I: Collision Detection on the GPU. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://devblogs.nvidia.com/parallelforall/thinking-parallel-part-i-collision-detection-gpu/>

Nvidia. Thinking Parallel, Part II: Tree Traversal on the GPU. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://devblogs.nvidia.com/parallelforall/thinking-parallel-part-ii-tree-traversal-gpu/>

Tero Karas. Nvidia. Thinking Parallel, Part III: Tree Construction on the GPU. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://devblogs.nvidia.com/parallelforall/thinking-parallel-part-iii-tree-construction-gpu/>

Miguel Casillas. 3D Collision detection (C++) [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.miguelcasillas.com/?mcportfolio=collision-detection-c>

David Rousset. Tutorial series: learning how to write a 3D soft engine from scratch in C#, TypeScript or JavaScript. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/>

Andrew Kickertz. Collision detection algorithms. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://biosport.ucdavis.edu/lab-meetings/collision-detection/Kickertz%20-%202012%20-%20Collision%20detection%20algorithms.pdf>

Ming C. Lin & Stefan Gottschalk. Collision detection between geometric models: a survey. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://users.soe.ucsc.edu/~pang/161/w06/notes/cms98.pdf>

[15] Herman J. Haverkort. Introduction to bounding volume hierarchies. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
< http://www.win.tue.nl/~hermanh/stack/bvh.pdf>

Research Gate. Stack-Less LBVH Traversal for Real-Time Ray Tracing. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://www.researchgate.net/publication/215704877_Stack-Less_LBVH_Traversal_for_Real-Time_Ray_Tracing>

Real Time Rendering. Object intersection[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.realtimerendering.com/intersections.html>

Stefan Kimmerle. Real-Time Collision Detection for Dynamic Virtual Environments. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www-ljk.imag.fr/Publications/Basilic/com.lmc.publi.PUBLI_Inproceedings@117681e94b6_1860ffd/bounding_volume_hierarchies.pdf>

Hamzah Asyrani Sulaiman and Abdullah Bade. Bounding Volume Hierarchies for Collision Detection. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://cdn.intechopen.com/pdfs-wm/34468.pdf>

An Nguyen. Implicit bounding volums and bounding volume hierarchies. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
http://graphics.stanford.edu/~anguyen/papers/bvh-anguyen.pdf

[16] Euclidean Space. 3D Theory - Octtree. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.euclideanspace.com/threed/solidmodel/spatialdecomposition/octtree/index.htm>

Nvidia. Chapter 37. Octree Textures on the GPU. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter37.html>

[17] Daniel Cedeño. Sistema de Coordenadas Cartesianas.[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.excellentias.com/sistema-coordenadas-cartesiano/>

[18] Math Is Fun. Vectors [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://www.mathsisfun.com/algebra/vectors.html>

University British Columbia. Properties of Vector Operations. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.math.ubc.ca/~feldman/m226/vectorppties.pdf>

[19] Math Is Fun. Dot Product. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://www.mathsisfun.com/algebra/vectors-dot-product.html>

[20] Math Is Fun. Cross Product. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
< http://www.mathsisfun.com/algebra/vectors-cross-product.html>

[21] Math Is Fun. Unit Vector. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
< https://www.mathsisfun.com/algebra/vector-unit.html>

[22] OpenGL Tutorial. Tutorial 3 : Matrices. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/>

Math Is Fun. Matrices. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.mathsisfun.com/algebra/matrix-introduction.html>

S.O.S Math. Algebraic Properties of Matrix Operations.[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.sosmath.com/matrix/matrix2/matrix2.html>

[23] Math Is Fun. Inverse matrix[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.mathsisfun.com/algebra/matrix-inverse.html>

[24] OpenGL Tutorials. Tutorial 17 : Quaternions. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-17-quaternions/>

Iñigo Quilez. thinking with quaternions. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.iquilezles.org/www/articles/quaternions/quaternions.htm>

University of Illinois. Introduction to Quaternionial Algebr. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.ncsa.illinois.edu/People/kindr/emtc/quaternions/>

[25] Wikipedia. Suspensió de Cardan. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://ca.wikipedia.org/wiki/Suspensi%C3%B3_de_Cardan>

[26] Wikipedia Gimbal lock. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
< https://en.wikipedia.org/wiki/Gimbal_lock >

[27] Wikipedia. Vertex (geometry). [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://en.wikipedia.org/wiki/Vertex_(geometry) >

[28] Wikipedia. Transform matrix. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://en.wikipedia.org/wiki/Transformation_matrix>

[29] Wikipedia. Translation. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://en.wikipedia.org/wiki/Translation>

[30] Wikipedia. Rotació. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://ca.wikipedia.org/wiki/Rotaci%C3%B3_(matem%C3%A0tiques)#Tres_dimensions>

Wikipedia. Matriu de rotació. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://ca.wikipedia.org/wiki/Matriu_de_rotaci%C3%B3>

[31] Wikipedia. Scaling (geometry). [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://en.wikipedia.org/wiki/Scaling_(geometry) >

Wikipedia. 2D Computer graphics/Scaling [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://en.wikipedia.org/wiki/2D_computer_graphics#Scaling>

[32] Johnny Huynh. Separating Axis Theorem for Oriented Bounding Boxes.
[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.jkh.me/files/tutorials/Separating%20Axis%20Theorem%20for%20Oriented%20Bounding%20Boxes.pdf>

Dyn4j. SAT (Separating Axis Theorem). [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.dyn4j.org/2010/01/sat/>

Svenson. Separating Axis Theorem (SAT) Explanation
[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.sevenson.com.au/actionscript/sat/>

Kah Shiu. Collision Detection Using the Separating Axis Theorem. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://gamedevelopment.tutsplus.com/tutorials/collision-detection-using-the-separating-axis-theorem--gamedev-169>

Game development. How many and which axes to use for 3D OBB collision with SAT. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:

<http://gamedev.stackexchange.com/questions/44500/how-many-and-which-axes-to-use-for-3d-obb-collision-with-sat/>

[33] Iñigo Quizles. modeling with distance functions
[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://iquilezles.org/www/articles/distfunctions/distfunctions.htm>

Chris Green∗Improved Alpha-Tested Magnification for Vector Textures and Special Effects. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.valvesoftware.com/publications/2007/SIGGRAPH2007_AlphaTestedMagnification.pdf>

Daniel Wright. Dynamic Occlusion with Signed Distance Fields. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://advances.realtimerendering.com/s2015/DynamicOcclusionWithSignedDistanceFields.pdf>

Iñigo Quizles. free penumbra shadows for raymarching distance fields.
[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.iquilezles.org/www/articles/rmshadows/rmshadows.htm>

Quora. Computer Graphics: How does the distance field ambient occlusion work in the Unreal Engine?
[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://www.quora.com/Computer-Graphics/How-does-the-distance-field-ambient-occlusion-work-in-the-Unreal-Engine>

Prutsdom Jiarathanakul∗ Ray Marching Distance Fields in Real-time on WebGL. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.371.5543&rep=rep1&type=pdf>

Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, Thouis R. Jones. Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.merl.com/publications/docs/TR2000-15.pdf >

ShaderToy. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://www.shadertoy.com/view/Xds3zN>

Jamie Wong. Ray Marching and Signed Distance Functions. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://jamie-wong.com/2016/07/15/ray-marching-signed-distance-functions/>

Alan Zucconi. Volumetric Rendering: Signed Distance Functions.[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.alanzucconi.com/2016/07/01/signed-distance-functions/#part3>

Codersnotes. Signed Distance Fields. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.codersnotes.com/notes/signed-distance-fields/>

[34] Ruofei Du. Tutorial of Ray Casting, Ray Tracing and Ray Marching
[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://blog.ruofeidu.com/tutorial-of-ray-casting-ray-tracing-and-ray-marching/>

Alan Zucconi. Raymarching. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.alanzucconi.com/2016/07/01/raymarching/>

[35] Molly Rocket. Video GJK explanation [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://mollyrocket.com/849>

Wikipedia. Gilbert–Johnson–Keerthi distance algorith. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://en.wikipedia.org/wiki/Gilbert%E2%80%93Johnson%E2%80%93Keerthi_distance_algorithm>

Jacob Tyndall. 3D Convex Collision Detection, Part One: GJK. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://hacktank.net/blog/?p=93>

Phill Djonov. GJK[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://vec3.ca/gjk/>

In2GPU. GJK ALGORITHM COLLISION DETECTION 2D IN C# [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://in2gpu.com/2014/05/12/gjk-algorithm-collision-detection-2d-in-c/>

dyn4j. GJK (Gilbert–Johnson–Keerthi). [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].

Disponible a Internet:
<http://www.dyn4j.org/2010/04/gjk-gilbert-johnson-keerthi/>

Allen Chou. Game Physics: Collision Detection – CSO & Support Function [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://allenchou.net/2013/12/game-physics-collision-detection-csos-support-functions/>

Chris Pollett. GJK: Fast Convex Hull Collision Detection in 2D and 3D [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.cs.sjsu.edu/faculty/pollett/masters/Semesters/Spring12/josh/?GJK.html>

Phill Djonov. Implementing a GJK Intersection Query. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://vec3.ca/gjk/implementation/>

[36] Ron Wein. 2D Minkowski Sums [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://doc.cgal.org/Manual/3.4/doc_html/cgal_manual/Minkowski_sum_2/Chapter_main.html>

[37] Mike Acton. Data Oriented Programming. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://dataorientedprogramming.wordpress.com/tag/mike-acton/>

Mike Acton. Data oriented design and c++. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.slideshare.net/cellperformance/data-oriented-design-and-c>

Noel LLopis. Data-Oriented Design Now And In The Future. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://gamesfromwithin.com/data-oriented-design-now-and-in-the-future>

Shanee Nishry. Data-Oriented Design Matters. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://shaneenishry.com/blog/2015/03/26/data-oriented-design-matters/>

Noel LLopis. Data-Oriented Design (Or Why You Might Be Shooting Yourself in The Foot With OOP) [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://gamesfromwithin.com/data-oriented-design>

Mike Actorn. Video-CppCon 2014: Mike Acton "Data-Oriented Design and C++" [en línia].

[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://www.youtube.com/watch?v=rX0ItVEVjHc>

Sean Middleditch. Video-Data-Oriented Design [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://www.youtube.com/watch?v=16ZF9XqkfRY>

Stefan Reinalter. Stateless, layered, multi-threaded rendering – Part 1. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://blog.molecular-matters.com/2014/11/06/stateless-layered-multi-threaded-rendering-part-1/>

Stefan Reinalter. Stateless, layered, multi-threaded rendering – Part 2: Stateless API Design. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://blog.molecular-matters.com/2014/11/13/stateless-layered-multi-threaded-rendering-part-2-stateless-api-design/>

Stefan Reinalter. Stateless, layered, multi-threaded rendering – Part 3: API Design Details. [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://blog.molecular-matters.com/2014/12/16/stateless-layered-multi-threaded-rendering-part-3-api-design-details/>

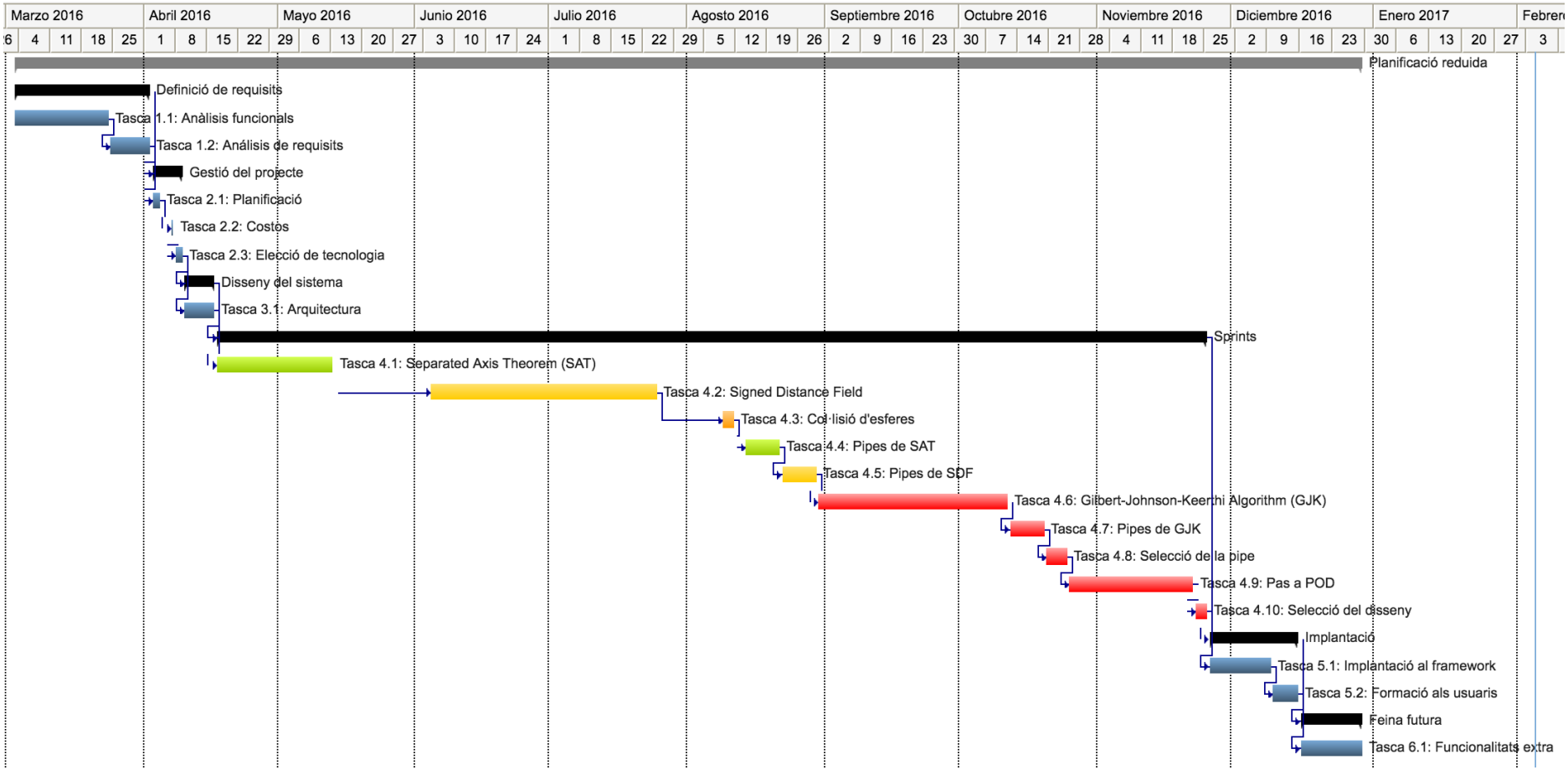
Stefan Reinalter. Stateless, layered, multi-threaded rendering – Part 4: Memory Management & Synchronization [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<https://blog.molecular-matters.com/2015/02/13/stateless-layered-multi-threaded-rendering-part-4-memory-management-synchronization/>

[38] Nuseibeh, B., Easterbrook, S., Requirements Engineering: A Roadmap. International Conference on Software Engineering
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
< http://www.cs.toronto.edu/~sme/papers/2000/ICSE2000.pdf>

[39] Miguel Casillas. Sphere to sphere collision [en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://www.miguelcasillas.com/?p=9>

[40] 180
Alex Ott. Test-driven development and unit testing with examples in C++.
[en línia].
[Accedit per darrer cop el 17 d'abril del 2017].
Disponible a Internet:
<http://alexott.net/en/cpp/CppTestingIntro.html>

Annex 1: Diagrama de Gantt inicial (simplificat)



	Nombre	Duración	Inicio	Fin	
	Planificació reduida	**212d**	**01/03/2016**	**21/12/2016**	
	Definició de requisits	**22d**	**01/03/2016**	**30/03/2016**	
	Tasca 1.1: Anàlisis funcionals	15d	01/03/2016	21/03/2016	Cap de projecte[33%],Analista[66%],Ordinador[1],Visual Studio[1]
	Tasca 1.2: Análisis de requisits	7d	22/03/2016	30/03/2016	Cap de projecte[43%],Analista[57%],Ordinador[1],Visual Studio[1]
	Gestió del projecte	**5d**	**31/03/2016**	**06/04/2016**	
	Tasca 2.1: Planificació	2d	31/03/2016	01/04/2016	Cap de projecte,Ordinador[1],Gantter[1]
	Tasca 2.2: Costos	1d	04/04/2016	04/04/2016	Cap de projecte,Ordinador[1]
	Tasca 2.3: Elecció de tecnologia	2d	05/04/2016	06/04/2016	Cap de projecte,Ordinador[1]
	Disseny del sistema	**5d**	**07/04/2016**	**13/04/2016**	
	Tasca 3.1: Arquitectura	5d	07/04/2016	13/04/2016	Arquitecte,Ordinador[1]
	Sprints	**156d**	**14/04/2016**	**17/11/2016**	
	Tasca 4.1: Separated Axis Theorem (SAT)	18d	14/04/2016	09/05/2016	Cap de projecte[4.1%],Desenvolupador[24.4%],Consultor[11.6%],Tester[26.6%],Analista[20%],Arquitecte[13.3%],Ordinador[1],Visual Studio[1]
	Tasca 4.2: Signed Distance Field	36d	31/05/2016	19/07/2016	Cap de projecte[4.1%],Desenvolupador[24.4%],Consultor[11.6%],Tester[26.6%],Analista[20%],Arquitecte[13.3%],Ordinador[1],Visual Studio[1]
	Tasca 4.3: Col·lisió d'esferes	3d	03/08/2016	05/08/2016	Cap de projecte[4.16%],Desenvolupador[27.2%],Consultor[8.33%],Tester[40.2%],Analista[6.66%],Arquitecte[13.33%],Ordinador[1],Visual Studio[1]
	Tasca 4.4: Pipes de SAT	6d	08/08/2016	15/08/2016	Cap de projecte[4.1%],Desenvolupador[24.4%],Consultor[11.6%],Tester[26.6%],Analista[20%],Arquitecte[13.3%],Ordinador[1],Visual Studio[1]
	Tasca 4.5: Pipes de SDF	6d	16/08/2016	23/08/2016	Cap de projecte[4.1%],Desenvolupador[24.4%],Consultor[11.6%],Tester[26.6%],Analista[20%],Arquitecte[13.3%],Ordinador[1],Visual Studio[1]
	Tasca 4.6: Gilbert-Johnson-Keerthi Algorithm (GJK	30d	24/08/2016	04/10/2016	Cap de projecte[5%],Desenvolupador[25%],Consultor[11.33%],Tester[26.67%],Analista[26.67%],Arquitecte[5.33%],Ordinador[1],Visual Studio[1]
	Tasca 4.7: Pipes de GJK	6d	05/10/2016	12/10/2016	Cap de projecte[4.1%],Desenvolupador[24.4%],Consultor[11.6%],Tester[26.6%],Analista[20%],Arquitecte[13.3%],Ordinador[1],Visual Studio[1]
	Tasca 4.8: Selecció de la pipe	3d	13/10/2016	17/10/2016	Cap de projecte[4.16%],Desenvolupador[27.2%],Consultor[8.33%],Tester[40.2%],Analista[6.66%],Arquitecte[13.33%],Ordinador[1],Visual Studio[1]
	Tasca 4.9: Pas a POD	20d	18/10/2016	14/11/2016	Cap de projecte[2.5%],Desenvolupador[20.5%],Consultor[13%],Tester[24%],Analista[20%],Arquitecte[20%],Ordinador[1],Visual Studio[1]
	Tasca 4.10: Selecció del disseny	3d	15/11/2016	17/11/2016	Cap de projecte[4.16%],Desenvolupador[27.2%],Consultor[8.33%],Tester[40.2%],Analista[6.66%],Arquitecte[13.33%],Ordinador[1],Visual Studio[1]
	Implantació	**14d**	**18/11/2016**	**07/12/2016**	
	Tasca 5.1: Implantació al framework	10d	18/11/2016	01/12/2016	Desenvolupador[80%],Consultor[20%],Ordinador[1]
	Tasca 5.2: Formació als usuaris	4d	02/12/2016	07/12/2016	Desenvolupador,Ordinador[1]
	Feina futura	**10d**	**08/12/2016**	**21/12/2016**	
	Tasca 6.1: Funcionalitats extra	10d	08/12/2016	21/12/2016	Cap de projecte[50%],Desenvolupador[50%],Ordinador[1]

Annex 2: Hores i dies per recurs i etapa

Recurs/Tasca (dies-hores)	DR (d)	(h)	G (d)	(h)	DIS (d)	(h)	IT1 (d)	(h)	IT2 (d)	(h)	IT3 (d)	(h)	IT4 (d)	(h)	IT5 (d)	(h)	IT6 (d)	(h)
Cap de projecte	8	**32**	5	**20**			0.75	**3**	1.5	**6**	0.125	**0.5**	0.25	**1**	0.25	**1**	1.5	**6**
Analista	14	**56**					3.6	**14.4**	7.2	**28.8**	0.2	**0.8**	1.2	**4.8**	1.2	**4.8**	8	**32**
Arquitecte					5	**20**	2.4	**9.6**	4.8	**19.2**	0.4	**1.6**	0.8	**3.2**	0.8	**3.2**	1.6	**6.4**
Desenvolupador							3.15	**12.6**	6.3	**25.2**	0.525	**2.1**	1.05	**4.2**	1.05	**4.2**	5.5	**22**
Tester							6	**24**	12	**48**	1.5	**6**	2	**8**	2	**8**	10	**40**
Consultor							2.1	**8.4**	4.2	**16.8**	0.25	**1**	0.7	**2.8**	0.7	**2.8**	3.4	**13.6**
TOTAL DIES	**22**	**88**	**5**	**20**	**5**	**20**	**18**	**72**	**36**	**144**	**3**	**12**	**6**	**24**	**6**	**24**	**30**	**120**

Recurs/Tasca (dies-hores)	IT7 (d)	(h)	IT8 (d)	(h)	IT9 (d)	(h)	IT10 (d)	(h)	IMP (d)	(h)	FUT (d)	(h)	TOTAL DIES	TOTAL HORES
Cap de projecte	0.25	**1**	0.125	**0.5**	0.5	**2**	0.125	**0.5**			5	**20**	**23.375**	**93.5**
Analista	1.2	**4.8**	0.2	**0.8**	4	**16**	0.2	**0.8**					**41**	**164**
Arquitecte	0.8	**3.2**	0.4	**1.6**	4	**16**	0.4	**1.6**					**21.4**	**85.6**
Desenvolupador	1.05	**4.2**	0.525	**2.1**	2.9	**11.6**	0.525	**2.1**	12	**48**	5	**20**	**39.575**	**158.3**
Tester	2	**8**	1.5	**6**	6	**24**	1.5	**6**					**44.5**	**178**
Consultor	0.7	**2.8**	0.25	**1**	2.6	**10.4**	0.25	**1**	2	**8**			**17.15**	**68.6**
TOTAL	**6**	**24**	**3**	**12**	**20**	**80**	**3**	**12**	**14**	**56**	**10**	**40**	**187**	**748**

DR: Definició de requisits (dies/hores) G: Gestió del projecte (dies/hores) DIS: Disseny (dies/hores) d: dies

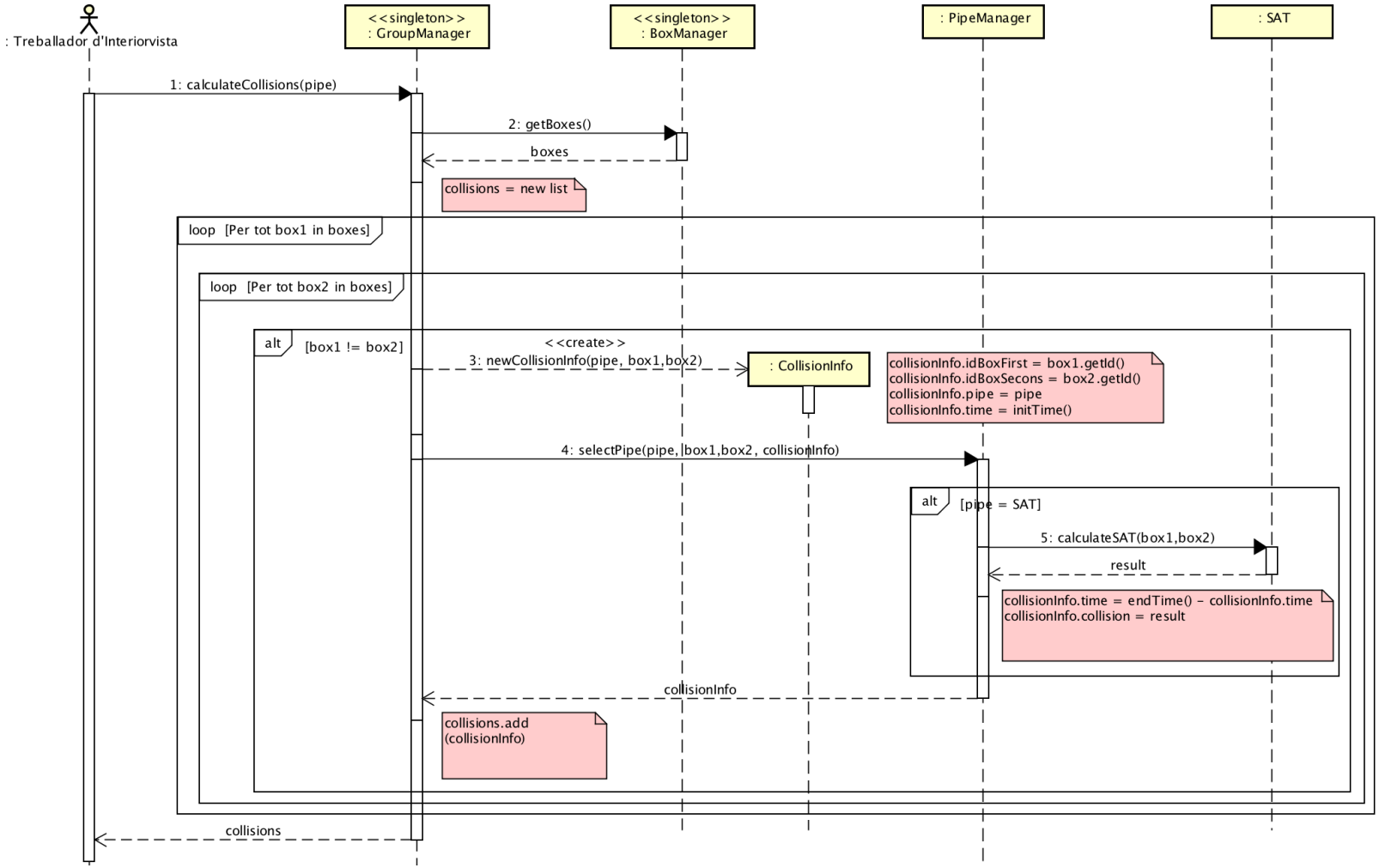
ITX: Implementació Tasca X (dies/hores) IMP: Implantació (dies/hores) FUT: Feina futura (dies/hores) h: hores

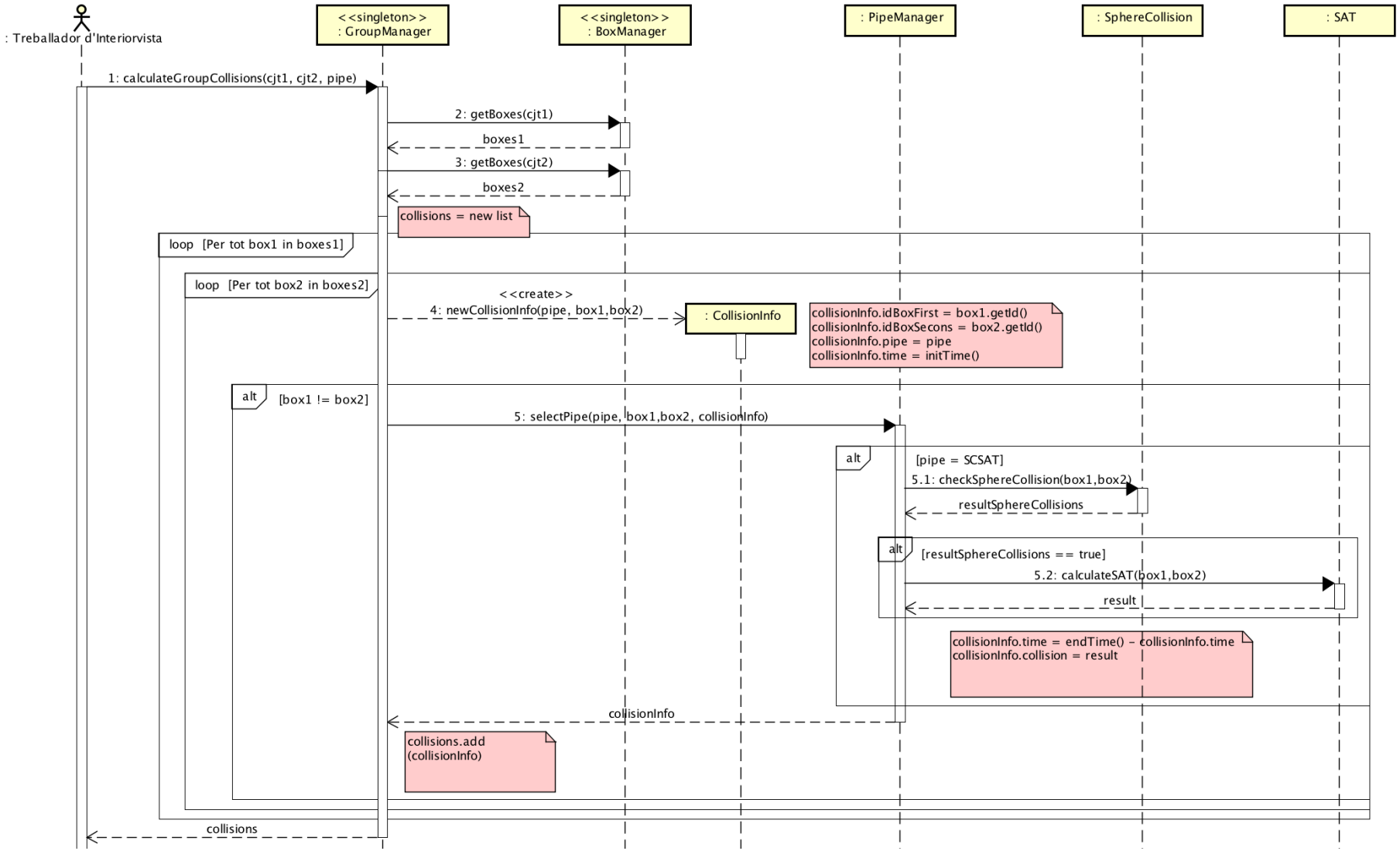
[10]Annex 3: Especificació de les funcions de GJK al disseny orientat a dades

GJK
+ clean() : void + detect(idPoint1 : int, idPoint2 : int) : bool + calcInitialDirection(idPoint1 : int, idPoint2 : int) : glm::vec3 + detect(idPoint1 : int, idPoint2 : int, direction : glm::vec3&, simplexPointsQuantity : int&) : bool + getFarthestPoint(direction : glm::vec3&, idPoint : int&) : glm::vec3 + getSupportPoint(direction : glm::vec3, idPoint1 : int, idPoint2 : int, result : glm::vec3&) : void + checkSimplex(X : X) : bool + checkZeroPointsSimplex(pointA : glm::vec3, pointB : glm::vec3&, direction : glm::vec3&, simplexPointsQuantity : int) : void + checkOnePointSimplex(pointA : glm::vec3, pointB : glm::vec3&, pointC : glm::vec3&, direction : glm::vec3&, simplexPointsQuantity : int) : void + checkTwoPointsSimplex(X : X) : void + checkThreePointsSimplex(X : X) : void + checkRegionFaceAB(planeABC : glm::vec3, vectorAB : glm::vec3, vectorAO : glm::vec3, X : X) : void + checkRegionPlaneABC(planeABC : glm::vec3, X : X) : void + checkReguionFaceAC(planeABC : glm::vec3, vectorAC : glm::vec3, vectorAO : glm::vec3, vectorAB : glm::vec3, X : X) : void + checkTwoFaces(planeABC : glm::vec3, vectorAB : glm::vec3, vectorAC : glm::vec3, vectorAD : glm::vec3, planeACD : glm::vec3, vectorAO : glm::vec3, X : X) : void

[10] X : X → pointA : glm::vec3, pointB : glm::vec3&, pointC : glm::vec3&, pointD : glm::vec3&, direction : glm::vec3&, simplexPointsQuantity : int&

Annex 4: Diagrames de seqüència dels casos d'ús "Calcular col·lisions" i "Calcular col·lisions per grups"




: Treballador d'Interiorvista
<<singleton>>
: GroupManager
<<singleton>>
: BoxManager
: PipeManager
: SphereCollision
: SAT
1: calculateGroupCollisions(cjt1, cjt2, pipe)
2: getBoxes(cjt1)
boxes1
3: getBoxes(cjt2)
boxes2
collisions = new list
loop [Per tot box1 in boxes1]
loop [Per tot box2 in boxes2]
<<create>>
4: newCollisionInfo(pipe, box1,box2)
: CollisionInfo
collisionInfo.idBoxFirst = box1.getId()
collisionInfo.idBoxSecons = box2.getId()
collisionInfo.pipe = pipe
collisionInfo.time = initTime()
alt
[box1 != box2]
5: selectPipe(pipe, box1,box2, collisionInfo)
alt
[pipe = SCSAT]
5.1: checkSphereCollision(box1,box2)
resultSphereCollisions
alt
[resultSphereCollisions == true]
5.2: calculateSAT(box1,box2)
result
collisionInfo.time = endTime() - collisionInfo.time
collisionInfo.collision = result
collisionInfo
collisions.add
(collisionInfo)
collisions

Annex 5: Especificació de les funcions de CollisionManager per a la implantació

CollisionManager
+ resetSets() : void + checkCollisionsTagVsTag(tag1 : unsigned int, tag2 : unsigned int) : vector<pair<int,int>> + checkCollisionsGroupVsGroup(group1 : unsigned int, group2 : unsigned int) : vector<pair<int,int>> + checkCollisionsObjectVsScene(idObj : unsigned int) : vector<pair<int,int>> + checkCollisionsObjectVsTag(idObj : unsigned int, tag : unsigned int) : vector<pair<int,int>> + checkCollisionsObjectVsGroup(idObj : unsigned int, group : unsigned int) : vector<pair<int,int>> + filterBoxesSceneVsScene() : void + filterBoxesTagVsTag(tag1 : unsigned int, tag2 : unsigned int) : void + filterBoxesGroupVsGroup(group1 : unsigned int, group2 : unsigned int) : void + filterBoxesObjectVsScene(idObj : unsigned int) : void + filterBoxesObjectVsTag(idObj : unsigned int, tag : unsigned int) : void + filterBoxesObjectVsGroup(idObj : unsigned int, group : unsigned int) : void + setGreen() : void + setBox(width : float, height : float, depth : float, offset : glm::vec3, rotation : glm::quat, tag : unsigned int, group : unsigned int, idBoxSceneGraph : unsigned int) : int + setTag(idBox : int, tag : unsigned int) : void + setGroup(idBox : int, group : unsigned int) : void + setSize(idBox : int, width : float, height : float, depth : float) : void + setOffset(idBox : int, offset : glm::vec3) : void + setRotation(idBox : int, rotation : glm::quat) : void + getTag(idBox : int) : void + getGroup(idBox : int) : void + getOffset(idBox : int) : void + getRotation(idBox : int) : void

Annex 6: Resultats d'aplicar la bateria de tests

TEST/PIPE	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1																
2																
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
16																
17																
18																
19																
20																
21																
22																
23																
24																
25																

26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	

Test correcte

Test incorrecte

Pipe 1: Col·lisió d'esferes + SAT

Pipe 2: SAT

Pipe 3: Col·lisió d'esferes + SDF complet amb control de sortida

Pipe 4: SDF complet amb control de sortida

Pipe 5: Col·lisió d'esferes + SDF complet

Pipe 6: SDF complet

Pipe 7: Col·lisió d'esferes + SDF per vèrtex amb control de sortida

Pipe 8: SDF vèrtex amb control de sortida

Pipe 9: Col·lisió d'esferes + SDF per vèrtex

Pipe 10: SDF vèrtex

Pipe 11: Col·lisió d'esferes + SDF per punts de mostreig amb control de sortida

Pipe 12: SDF per punts de mostreig amb control de sortida

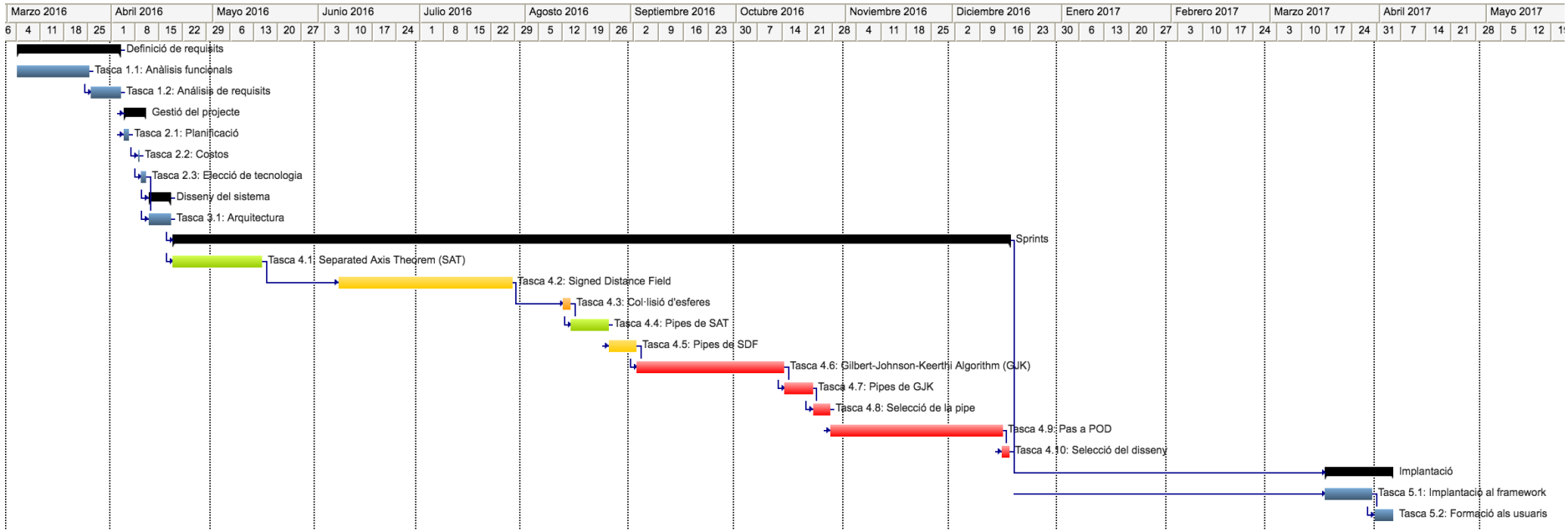
Pipe 13: Col·lisió d'esferes + SDF per punts de mostreig

Pipe 14: SDF per punts de mostreig

Pipe 15: Col·lisió d'esferes + GJK

Pipe 16: GJK

Annex 7: Diagrama de Gantt final



	Nombre	Duración	Inicio	Fin	Predecesoras	Recursos
	Definició de requisits	22d	**01/03/2016**	**30/03/2016**		
	Tasca 1.1: Anàlisis funcionals	15d	01/03/2016	21/03/2016		Cap de projecte[33%],Analista[66%],Ordinador[1],Visual Studio[1]
	Tasca 1.2: Análisis de requisits	7d	22/03/2016	30/03/2016	2	Cap de projecte[43%],Analista[57%],Ordinador[1],Visual Studio[1]
	Gestió del projecte	5d	**31/03/2016**	**06/04/2016**	**1**	
	Tasca 2.1: Planificació	2d	31/03/2016	01/04/2016	3	Cap de projecte,Ordinador[1],Gantter[1]
	Tasca 2.2: Costos	1d	04/04/2016	04/04/2016	5	Cap de projecte,Ordinador[1]
	Tasca 2.3: Elecció de tecnologia	2d	05/04/2016	06/04/2016	6	Cap de projecte,Ordinador[1]
	Disseny del sistema	5d	**07/04/2016**	**13/04/2016**	**7**	
	Tasca 3.1: Arquitectura	5d	07/04/2016	13/04/2016	7	Arquitecte,Ordinador[1]
	Sprints	171d	**14/04/2016**	**08/12/2016**	**8**	
	Tasca 4.1: Separated Axis Theorem (SAT)	18d	14/04/2016	09/05/2016	9	Cap de projecte[4.1%],Desenvolupador[17.5%],Consultor[11.6%],Tester[33.33%],Analista[20%],Arquitecte[13.3%],Ordinador[1],Visual Studio[1]
	Tasca 4.2: Signed Distance Field	36d	31/05/2016	19/07/2016	11	Cap de projecte[4.1%],Desenvolupador[17.5%],Consultor[11.6%],Tester[33.33%],Analista[20%],Arquitecte[13.3%],Ordinador[1],Visual Studio[1]
	Tasca 4.3: Col·lisió d'esferes	3d	03/08/2016	05/08/2016	12FI+10d	Cap de projecte[4.16%],Desenvolupador[17.5%],Consultor[8.33%],Tester[50%],Analista[6.66%],Arquitecte[13.33%],Ordinador[1],Visual Studio[1]
	Tasca 4.4: Pipes de SAT	6d	05/08/2016	16/08/2016	13	Cap de projecte[4.1%],Desenvolupador[17.5%],Consultor[11.6%],Tester[33.33%],Analista[20%],Arquitecte[13.3%],Ordinador[1],Visual Studio[1]
	Tasca 4.5: Pipes de SDF	6d	16/08/2016	24/08/2016	14	Cap de projecte[4.1%],Desenvolupador[17.5%],Consultor[11.6%],Tester[33.33%],Analista[20%],Arquitecte[13.3%],Ordinador[1],Visual Studio[1]
	Tasca 4.6: Gilbert-Johnson-Keerthi Algorithm (GJK)	30d	24/08/2016	05/10/2016	15	Cap de projecte[5%],Desenvolupador[18.3%],Consultor[11.33%],Tester[26.67%],Analista[33.3%],Arquitecte[5.33%],Ordinador[1],Visual Studio[1]
	Tasca 4.7: Pipes de GJK	6d	05/10/2016	13/10/2016	16	Cap de projecte[4.1%],Desenvolupador[17.5%],Consultor[11.6%],Tester[33.33%],Analista[20%],Arquitecte[13.33%],Ordinador[1],Visual Studio[1]
	Tasca 4.8: Selecció de la pipe	3d	13/10/2016	18/10/2016	17	Cap de projecte[4.16%],Desenvolupador[17.5%],Consultor[8.33%],Tester[50%],Analista[6.66%],Arquitecte[13.33%],Ordinador[1],Visual Studio[1]
	Tasca 4.9: Pas a POD	35d	18/10/2016	06/12/2016	18	Cap de projecte[2.5%],Desenvolupador[14.5%],Consultor[13%],Tester[30%],Analista[20%],Arquitecte[20%],Ordinador[1],Visual Studio[1]
	Tasca 4.10: Selecció del disseny	3d	06/12/2016	08/12/2016	19	Cap de projecte[4.16%],Desenvolupador[17.5%],Consultor[8.33%],Tester[50%],Analista[6.66%],Arquitecte[13.33%],Ordinador[1],Visual Studio[1]
	Implantació	14d	**08/03/2017**	**27/03/2017**	**10**	
	Tasca 5.1: Implantació al framework	10d	08/03/2017	21/03/2017	20	Desenvolupador[80%],Consultor[20%],Ordinador[1]
	Tasca 5.2: Formació als usuaris	4d	22/03/2017	27/03/2017	22	Desenvolupador,Ordinador[1]