FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BARCELONA TECH

Desarrollo de juegos como soporte educativo

Adrià Meijide Fernandez

Directora: Marisa Gil Gomez

Departamento: Arquitectura de computadores

GRADO EN INGENIERÍA INFORMÁTICA

Especialidad: Computación

28 de Junio de 2016

Resumen

Los juegos de ordenador se han mostrado altamente efectivos como soporte al aprendizaje del siglo XXI, basándose en un entorno creativo y atractivo. La gamificación es el empleo de mecánicas de juego en entornos y aplicaciones no lúdicas con el fin de potenciar la motivación, la concentración, el esfuerzo, la fidelización y otros valores positivos comunes a todos los juegos. Se trata de una nueva y poderosa estrategia para influir y motivar a grupos de personas.

El objetivo de este proyecto es desarrollar un juego multiusuario que sirva como soporte a conceptos matemáticos, computacionales (paralelismo, concurrencia, distribución, definición de objetos...), basándose en un entorno creativo y atractivo. Se utilizará una metodología que permita a los propios jugadores modificar y recrear el juego.

Este proyecto se basa en la adaptación de un juego ya creado a las necesidades particulares de una práctica para una asignatura concreta. No solo se busca la práctica para la asignatura si no dar un ejemplo de que esta adaptación es posible y que se puede exportar a otras asignaturas.

Resum

Els jocs d'ordinador s'han mostrat altament efectius com a suport a l'aprenentatge del segle XXI, basant-se en un entorn creatiu i atractiu. La gamificació és un mètode d'aprenentatge a través de mecàniques de joc en entorns i aplicacions no lúdiques amb la finalitat de potenciar la motivació, la concentració, l'esforç, la fidelització i altres valors positius comuns a tots els jocs. Es tracta d'una nova i poderosa estratègia per influir i motivar grups de persones.

L'objectiu d'aquest projecte és desenvolupar un joc multiusuari que serveixin com a suport a conceptes matemàtics, computacionals (paral·lelisme, concurrència, distribució, definició d'objectes ...), basant-se en un entorn creatiu i atractiu. S'utilitzarà una metodologia que permeti als propis jugadors modificar i recrear el joc.

Aquest projecte es basa en l'adaptació d'un joc ja creat a les necessitats particulars d'una pràctica per a una assignatura concreta. No només es busca la pràctica per a l'assignatura si no donar un exemple de que aquesta adaptació és possible i que es pot exportar a altres assignatures.

Abstract

Computer games have been shown highly effective as learning support on the twenty-first century, based on a creative and attractive environment. The Gamification is the use of game mechanics in environments and not recreational applications in order to enhance motivation, concentration, effort, loyalty and other positive values common to all games. This is a powerful new strategy to influence and motivate groups of people.

The objective of this project is to develop a multi-user game that serves as support for mathematical, computational concepts (parallelism, concurrency, distribution, object definition ...) based on a creative and attractive environment. We will use a methodology that allows players themselves to modify and recreate the game.

This project is based on the adaptation of a game already created to the particular needs of a practice for a particular subject. Not only practice for the subject is sought moreover we want to give an example that this adaptation is possible and that can be exported to other subjects.

Agradecimientos

Quiero dar las gracias a mi tutora del proyecto y profesora de la universidad, Marisa Gil Gomez por ofrecer este proyecto y aceptarme como desarrollador de este. Además de ofrecerme la oportunidad que representa colaborar con el desarrollo de material educativo. Como alumno debo admitir que uno nunca se plantea el esfuerzo que conlleva realizar todo este material y que normalmente no está lo suficiente valorado. Ver el otro lado de la moneda es un buen ejercicio para percatarse de las ganas e ilusión que se ponen en la innovación para la educación.

También agradecer a toda mi familia y amigos por el apoyo recibido, por escucharme, ayudarme en lo que he podido necesitar y especialmente por distraerme cuando han visto que necesitaba despejarme del proyecto.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1. Contexto

"Este servidor va lento", "Tengo una buena conexión pero tengo lag"… Son frases que podemos oír todos los días sobre los juegos on-line. Pero, realmente, qué sabemos sobre la codificación del servidor? Qué estrategia usa para recibir toda la información y distribuir el trabajo? Qué gestión de la memoria utiliza? Es solo un thread que ejecuta secuencialmente todo el código?

Todas estas son preguntas que un alumno de ingeniería del software se puede hacer, y en la asignatura de SOAD[1] se explica y se enseña. La pregunta que nos surge es, se puede enseñar de una forma más atractiva para el alumno? Podemos, por ejemplo, mostrar a un alumno como la concurrencia es importante en un servidor a través de un juego? O lo que debemos tener en cuenta a la hora de paralelizar el servidor?

En alguna asignatura de la carrera ya se usan juegos como aprendizaje, es el caso de la asignatura de EDA[2] con su edición cuatrimestral de su juego. En los últimos años todos los alumnos hemos pasado por esta asignatura y recordamos con más o menos cariño, con más o menos frustración la competición. Esta asignatura nos ofrece un reto, una competición, para ver quién es capaz de adaptar los conocimientos de los que disponemos a un problema planteado en forma de juego.

Este proyecto quiere ofrecer una práctica a los alumnos de SOAD en la cual se les dé un juego en el que puedan analizar la gestión de los recursos, la concurrencia de ejecución y la integridad de los datos. Se quiere que el alumno se pregunte qué elementos interviene, cómo interactúan con el sistema y entre ellos. Que estas preguntas surjan durante la propia experiencia del juego. Una vez analizado, el objetivo es que desarrollen una mejora de la solución proporcionada, utilizando los conocimientos adquiridos en clase y comprobando los resultaos con las herramientas proporcionadas.

La práctica que se generara también quiere suponer un pequeño reto para los alumnos. Normalmente todos los códigos con los que se ha tenido que enfrentar/entender el alumno hasta este momento han sido escrito por los profesores de la universidad y suelen tener una estructura limpia y usar operaciones estándares. Al ser un juego ya creado y desarrollado colaborativamente con personas de todas partes del mundo nos podemos asegurar que dentro del propio código los alumnos encontraran estilos diferentes a la hora de programar.

[1] SOAD: Sistemas operativos para aplicaciones distribuidas. [1]

[2] EDA: Estructura de datos y algoritmos. [2]

Concretamente en este proyecto trataremos diferentes modelos de threads para servidores. Los tres modelos que usaremos serán: thread-per-request, thread-per-conection y thread-per-object. Más adelante se explicaran estas técnicas. También buscaremos y valoraremos la integridad y seguridad de los datos en el servidor.

Aunque se usará un juego en concreto la intención es dar los pasos, herramientas y conceptos para poder hacer lo mismo con cualquier otro.

1.2. Actores implicados

Ahora detallaremos los actores implicados en el proyecto, es decir, las persona o organizaciones que pueden interesarse por el proyecto.

Desarrollador y tester: Este proyecto tiene un único desarrollador. Él es el primer interesado en terminar este proyecto y ofrecer un buen producto. La responsabilidad del desarrollador del proyecto es adaptar el juego a las necesidades de la asignatura, comentarlo, analizarlo y de diseñar las diferentes soluciones. El desarrollador tiene que probar cada parte del proyecto y comprobar que no hay errores en el juego.

Tutora del proyecto: La tutora de este proyecto es Marisa Gil Gómez, su papel es el de supervisar que el proyecto cumpla el calendario estipulado y que se alcancen los objetivos marcados. Además, puede ayudar y guiar al desarrollador para realizar el proyecto.

Creadores del juego: Las personas que han desarrollado el juego pueden estar interesadas si se llega a la conclusión de que alguna de las soluciones es mejor que la implementada. Además de poder ofrecer a las personas que tienen servidores con el juego una mejora del rendimiento.

Alumnos: Los alumnos son el actor más importante, son los que se beneficiaran del producto final, su experiencia, la consolidación de conocimientos, su interés y que superen correctamente la práctica es lo que más ha de importar.

Profesores: Los profesores serán quien evalúe la efectividad del proyecto y quien decidirá si se aplica o no, al fin y al cabo son ellos quienes dan la materia.

1.3. Estado del arte

Desde hace unos años existe una corriente que ha estudiado el impacto de introducir el mundo de los videojuegos en la educación [3][4][5][6].

Si bien el número de juegos educativos para alumnos de primaria y la E.S.O. es grande y podemos encontrar paginas[3] que incluyen juegos para las diferentes asignaturas que se dan, conforme los conceptos son más complejos y específicos, como los que se enseñan en las carreras, encontramos cada vez menos videojuegos adaptados para las asignaturas.

Aun así, tenemos dos buenos ejemplos del uso de estos juegos aplicados a una asignatura. El primero lo encontramos en esta misma facultad, es la práctica de EDA, en la cual se pide a los alumnos que programen un usuario de un juego proporcionado (este puede ser por ejemplo el Pacman, Tron, Battle Royale, etc) [1]. El otro es en la universidad de Northern Illinois University, donde en la asignatura de métodos numéricos los alumnos deben programar el comportamiento del vehículo del juego NIU-Torcs. [7]

Actualmente en la asignatura de SOAD se está realizando una práctica donde los alumnos deben buscar un juego y mejorar el servidor un poco, sin el uso de ningún modelo en concreto. Una parte del objetivo de este proyecto es que el alumno ya tenga documentación suficiente como para que se pueda centrar más en la implementación de uno de los tres modelos que expondremos más adelante.

Debido al temario que da la asignatura ninguno de los juegos antes propuestos nos es útil para la realización de este proyecto, es por ello que necesitaríamos hacer un juego desde cero o adaptar algún juego ya existente. Dada la limitación temporal del proyecto se ha optado por adaptar un juego ya existente.

En el juego escogido para el proyecto nos podemos encontrar uno de los diferentes modelos de servidores o por el contrario que no usa ninguno de ellos. En este proyecto nos centraremos en estos tres modelos: thread-per-request, thread-per-conection y thread-per-object.

También debemos identificar qué tipo de lenguaje tiene el juego, dependiendo de este, podremos tener mayor o menor control de la memoria. En caso de que sea un lenguaje del tipo Java o C# no tendremos un control explicito ya que el garbage collector se encargara de ello. En cambio si es C, C++, etc... Tendremos un mayor control de la gestión de la memoria.

[3] http://www.mundoprimaria.com/

También nos importara la integridad y seguridad de los datos del servidor durante la partida.

A continuación se explican los tres modelos antes mencionados.

1.3.1. Thread-per-request

En este modelo cada solicitud que realiza cada uno de los clientes se asigna a un nuevo thread. Este modelo es útil para los servidores que manejan solicitudes de larga duración (por ejemplo, consultas de bases de datos) de varios clientes. Es menos útil para solicitudes de corta duración debido a la sobrecarga de crear y destruir un nuevo thread para cada solicitud. También puede consumir un gran número de los recursos del sistema operativo si muchos clientes hacen peticiones simultáneamente [8].

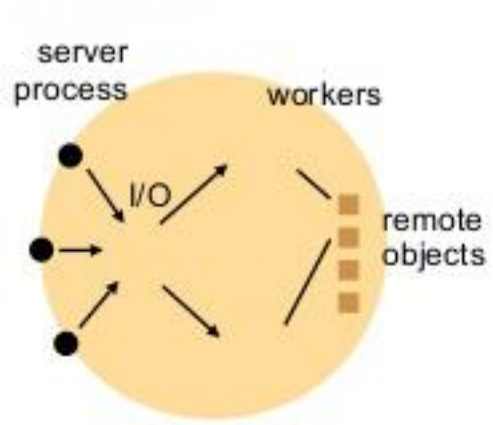


Figura 1.1Modelo de thread-per-request

1.3.2. Thread-per-conection

Este modelo es una variación del thread-per-request que amortizar el coste de crear y destruir el thread a través de múltiples peticiones. Este modelo pone cada cliente que se conecta al servidor en un thread separado para la duración de la sesión. Es útil para los servidores que llevan a cabo conexiones de larga duración con múltiples clientes. No es útil cuando los clientes que hacen solamente una única solicitud ya que sería esencialmente un modelo de thread-per-request [8].

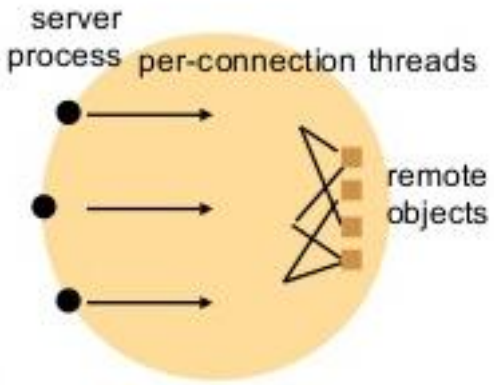


Figura 1.2Modelo thread-per-client

1.3.3. Thread-per-object

Este modelo asocia un thread para cada objeto lógico en el servidor. Es útil cuando se quiere reducir al mínimo la cantidad de trabajo requerida para el multi-thread en un servidor existente. Es menos útil si ciertos objetos reciben considerablemente más solicitudes que otros, ya que se convertirían en un cuello de botella [8].

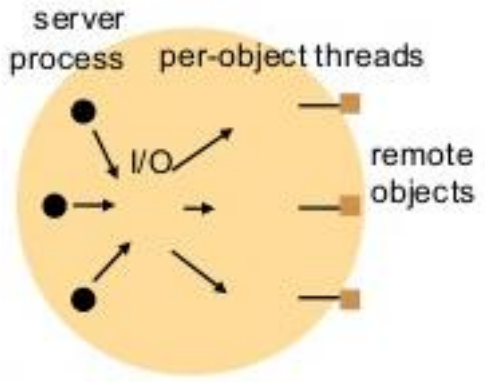


Figura 1.3Modelo thread-per-object

1.4. Organización del proyecto

La memoria del proyecto está organizada de la siguiente forma:

Capitulo 2: Se describen los objetivos del proyecto, las limitaciones a la hora de realizarlo, su alcance, la planificación temporal, la gestión económica y el análisis de sostenibilidad.

Capitulo 3: Se hace un resumen de los pasos previos, configuraciones del sistema y de los análisis realizados.

Capitulo 4: Se realiza una explicación del diseño, implementación y pruebas de los diferentes modelos.

Capitulo 5: Se explica que deberá hacer el alumno y de que material dispondrá.

2. Gestión del proyecto

2.1. Objetivos

Ahora que se ha explicado el problema que pretende solucionar este proyecto, podemos pasar a analizar los diferentes objetivos a alcanzar durante la realización de este.

Se marcan los siguientes objetivos:

Localizar, documentar y comentar: todas las zonas del código donde el cambio de modelos pueda afectar, específicamente en las zonas del código del servidor. Para entender cómo funciona el juego en si también se comentara más superficialmente el código del cliente. La finalidad de este primer objetivo es situar, guiar y familiarizar al alumno con el juego. Reduciendo el tiempo que esto supondría comparado con un juego sin esta documentación.

Analizar: la solución que usa el juego actualmente y ver si se corresponde con alguno de los que queremos implementar. También se analizara la estructura de datos, la gestión de recursos, la concurrencia de ejecución y la integridad de los datos en el servidor. En este segundo objetivo damos al alumno las herramientas para que evalúe la solución usada y pueda valorar que otras implementaciones pueden mejorar la solución existente.

Analizar, proponer, implementar y probar: diferentes soluciones. Estas cuatro acciones se llevaran a cabo si durante el análisis y el diseño de la solución esta supone una mejora para el servidor. Este tercer objetivo va más enfocado al profesor, a darle un soporte con que comparar el resultado del alumno, y al propio alumno al ofrecerle un feedback.

Proponer ejercicios a realizar por el alumno: este cuarto objetivo está dirigido tanto a los alumnos como al profesor, ofrece un ejercicio a realizar relacionado con el proyecto.

Con todo ello se quiere ofrecer al alumno una práctica en la que pueda experimentar directamente las diferentes estrategias a la hora de codificar un servidor con threads. Darle un criterio para discernir cual puede ser la estrategia que pueda interesar en cada momento y un ejemplo real.

2.2. Metodología y rigor

El proyecto esta implementado básicamente en C++ (algún archivo de configuración está escrito en LUA). Se ha usado el programa Eclipse para desarrollar y para el control de versiones se ha usado Github.

Debido al poco tiempo del que se dispone para realizar el proyecto, la mejor manera de desarrollarlo es mediante metodologías ágiles y con objetivos concretos y a corto plazo. Al ser un trabajo individual muchas de las metodologías ágiles no se pueden aplicar literalmente, ya que están destinadas a trabajos en grupo, pero hay muchos conceptos e ideas que se pueden reaprovechar. Por tanto, podemos concluir que se utilizará la metodología Scrum[9] para la realización del proyecto. Cuando la planificación sufrió de un contratiempo, la mejor opción fue mantener esta metodología ya que es idónea para adaptar-se a este tipo de contratiempos.

Mediante el uso de esta metodología de trabajo con ciclos cortos (con objetivos que se han de lograr cada semana), se garantiza una mejor visión real del estado del proyecto y de si se encuentra o no dentro del calendario marcado al inicio del proyecto.

Aunque este proyecto no tiene un cliente real, se considerará al director del proyecto como el cliente final y se mantendrán reuniones periódicas con el fin de que haya un feedback constante sobre el estado del proyecto y así se solucionar problemas más rápidamente.

Debido a la dificultad que pueden suponer los errores de programación, ya que pueden llegar a condicionar los resultados obtenidos, para intentar minimizarlo, se utilizará una metodología de desarrollo guiado por pruebas para intentar encontrar los errores lo más rápido posible y así minimizar los errores.

Después de la implementación de cada una de los diferentes modelos se realizaran unas pruebas. Primero se comprobara que el resultado continúe siendo el mismo y, a continuación, que la mejora sea lo suficientemente significativa para incorporarla.

El control de las versiones del proyecto siempre ha estado a la orden el día por si surgiera cualquier problema inesperado.

2.3. Planificación temporal

El proyecto tiene una duración aproximada de cuatro meses y medio, teniendo en cuenta que su comienzo coincide con el inicio de GEP, el 22 de febrero de 2016, y definiendo como fecha final el día de la defensa del proyecto, que es a finales de Junio de 2016.

Cabe destacar que la duración del proyecto es estimada y, debido a la metodología de trabajo escogida (Scrum), se ha visto afectada por algunas desviaciones eventuales, a las que haremos referencia más adelante.

2.3.1. Descripción de las tareas

Al utilizar una metodología ágil no se marcarán unas tareas estrictas que se vayan realizando iterativamente, sino que cada semana se definirán unos objetivos a alcanzar adaptando el proceso a posibles desviaciones que puedan surgir. Igualmente, se pueden definir algunas tareas genéricas que sí tienen un orden estricto debido a las precondiciones que existen entre ellas.

Las tares se podrían nombrar genéricamente de la siguiente manera:

1. Realización del hito inicial.
2. Análisis del juego.
3. Thread-per-request.
4. Thread-per-conection.
5. Thread-per-object.
6. Propuesta de ejercicios a realizar.
7. Hito final.

2.3.1.1. Planificación del proyecto (hito inicial)

Esta parte del proyecto es la inicial y consta de las siguientes partes:

1. Alcance del proyecto y contextualización.
2. Planificación temporal.
3. Presupuesto y sostenibilidad.
4. Presentación preliminar.
5. Pliego de condiciones.
6. Documento final y presentación oral.

No se entrara en detalle en esta fase ya que se corresponde con la autoevaluación y documentación inicial de los aspectos del proyecto. Es importante aclarar las características de todo el proyecto para su correcta gestión, y es precisamente la función de esta fase: redactar una documentación que sirva de pauta para la construcción de la práctica. No se profundizara en qué consiste cada parte ya que son compartidas por todos los alumnos de GEP. Dado que durante la realización de GEP se imponen diferentes entregas con unas fechas concretas de entrega no se ha observado ninguna desviación en tareas.

2.3.1.2. Análisis del juego

Esta tarea tiene dependencia con la primera ya que se basa en el planteamiento del proyecto. Dentro de ella nos podemos encontrar con 4 etapas consecutivas y diferentes:

Primera etapa: en la que el objetivo es que el desarrollador seleccione el juego, se familiarice con el entorno de desarrollo, con el juego y con el código del juego a la vez que va comentando el código. Esta etapa tiene una duración planeada de 6 días.

Segunda etapa: consistente en realizar un análisis de la gestión de recursos es decir del uso de la memoria. Esta etapa tiene una duración planeada de 3 días.

Tercera etapa: donde se pretende establecer en qué puntos es importante que una tarea se ejecute antes que la otra y si a primera vista podría haber una posible zona de paralelismo. Esta etapa tiene una duración planeada de 3 días.

Cuarta etapa: donde se entrara a valorar la integridad de los datos dentro del servidor en cuanto acceso por parte de los usuarios. Esta etapa tiene una duración planeada de 3 días.

Desviaciones eventuales: en esta tarea no se han dado desvíos demasiado graves más allá del incremento de un día en la primera etapa. Esto es debido a la dificultad de familiarizarse con el código, el código está escrito por personas con estilos diferentes y con pocos comentarios aclaratorios. Añadiendo así 1 día a la primera etapa.

2.3.1.3. Thread-per-request, Thread-per-conection y Thread-per-object

Estas tres tareas tienen dependencia con la segunda ya que es necesario conocer bien el programa y los resultados de los análisis nos pueden facilitar esta tarea. Entre ellas no hay ninguna dependencia ya que son tres modelos diferentes con ninguna correlación. Dentro de cada una de estas tres tareas podemos encontrar cuatro etapas:

> **Primera etapa**: que consiste en el análisis específico de las zonas que pueden ser modificadas para adaptar la solución al modelo seleccionado. Esta etapa tiene una duración planeada de 2 días.
>
> **Segunda etapa**: que consiste en el diseño, adaptación del programa al modelo que se quiere implementar, teniendo en cuenta las conclusiones de la etapa anterior. Esta etapa tiene una duración planeada de 3 días.
>
> **Tercera etapa**: que consiste en la implementación del modelo antes diseñado. Cambiando las partes que nos interesan del programa y procurando mantener el funcionamiento lo más parecido posible al programa inicial. Esta etapa tiene una duración planeada de 4 días.
>
> **Cuarta etapa**: que consiste en las pruebas que se realizaran en el servidor con los diferentes clientes. Se intentara que las pruebas sean lo más parecidas posible. En caso que el programa no funcione como debería se corregirán los errores producidos. Esta etapa tiene una duración planeada de 2 días.

Desviaciones eventuales y ejecución Thread-per-request: en esta tarea han existido dos desviaciones. Una en contra del estudiante ya que la etapa de análisis se ha alargado en 2 días debido al desconocimiento del modelo. Pero también se ha producido un desvió a favor del estudiante, debido a las características propias del juego solo se ha podido llegar a la segunda etapa ya que no suponía ningún beneficio aplicar este modelo. Es por ello que se han sumado dos días a la primera etapa u restado los días de la tercera y cuarta etapa resultando en 3 días a favor del estudiante.

Desviaciones eventuales y ejecución Thread-per-conection: en esta tarea solo han existido desviaciones en contra del interés del estudiante. Por un lado la falta de conocimientos sobre el modelo ha alargado la primera etapa en 2 días. Por otro lado la implementación se ha alargado 3 días debido a la dedicación de elegir un modo de gestión de los threads (se ha escogido OpenMP[10]), de analizarlos (Extrae[11] y Paraver[12]) y a al tiempo dedicado a buscar que versión de este era necesario usar para que se pudiera usar tanto en Windows como en Linux. Por último a la hora de realizar las pruebas se decidió usar los ordenadores de la FIB pero estos carecen de librerías necesarias para ejecutar-lo y realizar las pruebas. Por este motivo se debieron

reunir diferentes ordenadores y consumió más de un día desviándose así 3 días sobre lo planificado. Sumando las desviaciones llegamos a un total de 8 días en esta tarea.

Desviaciones eventuales y ejecución Thread-per-object: en esta tarea han existido dos desviaciones. Una en contra del estudiante ya que la etapa de análisis se ha alargado en 2 días debido al desconocimiento del modelo. Pero también se ha producido un desvió a favor del estudiante, debido a las características propias del juego solo se ha podido llegar a la segunda etapa ya que no suponía ningún beneficio aplicar este modelo. Es por ello que se han sumado dos días a la primera etapa u restado los días de la tercera y cuarta etapa resultando en 3 días a favor del estudiante.

2.3.1.4. Propuesta de ejercicios a realizar

La sexta tarea está orientada a crear la estructura de la práctica a realizar por el alumno. A organizar y dejar preparada la documentación y el programa que recibirá el alumno.

Desviaciones eventuales: en esta tarea no se han dado desvíos.

2.3.1.5. Hito final

En el hito final se presenta la memoria que incluye los análisis realizados al código inicial y todas las implementaciones de los diferentes modelos. Además, se entregará un archivo adjunto con la documentación de todo el código y el modelo aplicado, una guía para orientar al alumno en la propia implementación del juego y los ejercicios propuestos.

Desviaciones eventuales: en esta tarea han existido desvíos debido a la justificación de los desvíos y la no implementación de dos modelos. Sumando las desviaciones llegamos a un total de 2 días en esta tarea.

2.3.2. Diagrama de Gantt

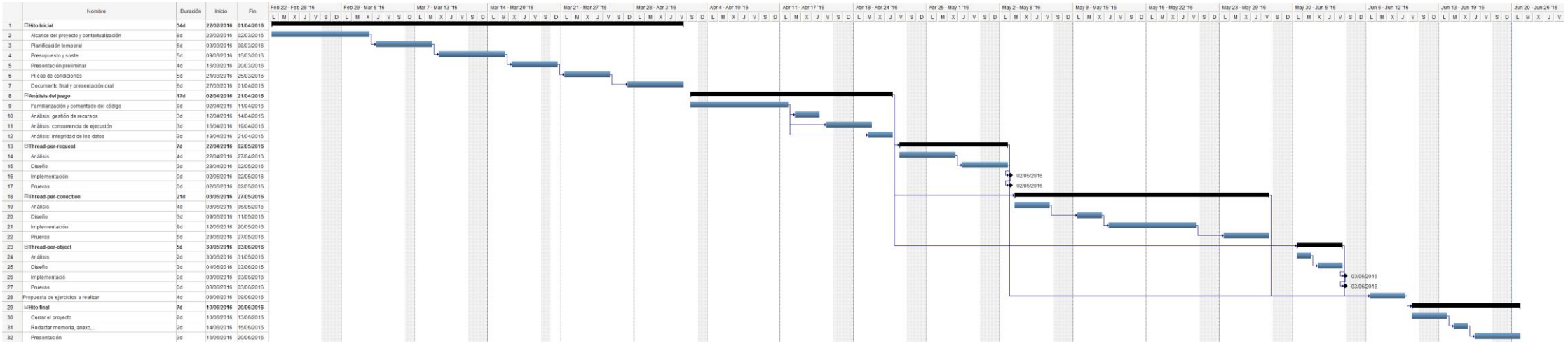


Figura 2.1. Diagrama de Gantt del proyecto

2.4. Recursos

En esta sección se explicaran los materiales que se han necesitado para desarrollar el proyecto y los cambios realizados si se ha tenido algún problema.

2.4.1. Hardware

Para realizar este proyecto se ha necesitado un ordenador con Windows y Ubuntu para el desarrollo que pudiera compilar C++ y LUA. La idea principal era usar hasta 17 ordenadores para realizar las pruebas, debido a la falta de librerías en los ordenadores de la FIB al final se opto por realizar las pruebas en la casa del desarrollador con 3 ordenadores con Windows ejecutando 4 clientes cada uno de ellos además de otro ordenador con Ubuntu ejecutando el servidor.

Los ordenadores adicionales solo son necesarios durante las etapas de pruebas y se podría reducir el número de estos pero acabaría por no reflejar la realidad ya que colapsaríamos el servicio de comunicación y la ejecución de estos. El ordenador principal es requerido en todas las etapas ya que en las pruebas se ha usado de servidor dada su mayor capacidad.

2.4.2. Software

Para realizar el proyecto se han usado como sistema operativo Windows 7 y Ubuntu 15.10. Para realizar la documentación Microsoft Office 2016 y Gantter. Github y Dropbox para mantener las versiones y los documentos. Eclipse para el desarrollo. Extrae y Paraver para generar trazas y visualizarlas. OpenMP para realizar las paralelizaciones.

2.4.3. Humanos

Este proyecto ha involucrado a dos personas. Marisa Gil Gomez, la directora, tutora y supervisora del proyecto cuya responsabilidad ha sido aprobar y rechazar sugerencias sobre el juego y guiarme durante el proyecto. La segunda persona involucrada he sido yo, el desarrollador.

2.5. Gestión económica

Conocer el ámbito del proyecto y establecer una planificación factible para este son dos aspectos calve para su correcta gestión, pero también lo es saber determinar si el proyecto es viable a nivel económico, social y ambiental. Por este motivo, en este apartado se valoraran estos aspectos además de especificar el presupuesto necesario para llevarlo a cabo.

2.5.1. Presupuesto de recursos humanos

Cada una de las actividades, aunque sea desarrollada en la práctica por una misma persona, corresponde a un rol diferente y por lo tanto tendrá un precio de mercado diferente. Se considera que las tareas de gestión y documentación las hace un Jefe de Proyecto (a 50€/h), el diseño de la solución la realiza un Diseñador(a 35€/h), las tareas de análisis del producto las realiza un Analista (a 35€/h), y por último la implementación y las pruebas las llevará a cabo un Programador (a 30 €/h). Se repartirán las tareas tal y como se mencionan en el apartado anterior.

Rol	Horas	Precio por hora	Precio total
Jefe de proyecto	146 h	50 €/h	7.300 €
Diseñador	70 h	35 €/h	2.450 €
Analista	136 h	35 €/h	4.760 €
Programador	160 h	30 €/h	4.800 €
Total	512 h		19.310 €

Tabla 2.1. Presupuesto de recursos humanos

Desviaciones: el presupuesto inicial de recursos humanos se valoro en 18.550 € y el precio final es de 19.310 €. La diferencia supone un desvío de 760 € esto es debido al incremento de las horas jefe de proyecto y del analista pero también a la reducción por parte del programador debido a las partes no implementadas ni probadas que se planificaron en un inicio.

2.5.2. Presupuesto de hardware

Para llevar a cabo la implementación del proyecto se necesitan una serie de elementos de hardware para las tareas de documentación, implementación y pruebas. Se estimo del uso de un ordenador para la documentación y pruebas y de hasta 16 ordenadores para realizar las pruebas, finalmente y debido a las circunstancias solo se usaron 3 ordenadores de prueba durante un periodo de 5 días.

Producto	Precio	Unidades	Vida útil	Amortización
Ordenador sobremesa	900 €	1	5 años	75 €
Ordenador sobremesa	700 €	3	5 años	175 €
Total				250 €

Tabla 2.2. Presupuesto de hardware

Desviaciones: el presupuesto inicial de hardware se valoro en 1.009€ y el precio final ha sido de 250€. La diferencia supone un desvío de -759€ debido a las circunstancias que se produjeron durante las pruebas.

2.5.3. Presupuesto de software

Para realizar el proyecto ha sido necesario el uso de herramientas de software, la mayoría de ellas son gratuitas y son las que se han utilizado en diferentes asignaturas durante el Grado.

Producto	Precio	Unidades	Vida útil	Amortización
Windows 7	0 €*	1	-	0 €
Microsoft Office 2016	149 €	1	3 años	21 €
Gantter	0 €	1	-	0 €
Ubuntu 15.10	0 €	1	-	0 €
Dropbox	0 €	1	-	0 €
OpenMP	0 €	1	-	0 €
Extrae	0 €	1	-	0 €
Eclipse	0 €	1	-	0 €
Paraver	0 €	1	-	0€
Total				21 €

Tabla 2.3. Presupuesto de software

* Versión estudiante

2.5.4. Gastos indirectos

En este apartado se tienen en cuenta costes asociados a la realización del proyecto como pueden ser la energía eléctrica consumida o el papel.

Producto	Precio	Unidades	Coste aproximado
Electricidad	0,15 €/kWh	512 h	76,80 €
Papel	30 €/paquete	1	30 €
Total			106,80 €

Tabla 2.4. Presupuesto de gastos indirectos

Desviaciones: debido al incremento del tiempo a la hora de realizar la práctica se ha incrementado los gastos indirectos en 1,80 €.

2.5.5. Costes imprevistos

El único coste imprevisto que podría surgir es un fallo en un ordenador utilizado para el desarrollo del proyecto; habría que añadir el coste de reparación que aproximadamente sería de 130€, y mientras estuviera inactivo se utilizaría un segundo ordenador para no retrasar las actividades planificadas. El riesgo de que ocurra este imprevisto es de un 15%. Suponemos que no se estropearan dos ordenadores simultáneamente.

2.5.6. Presupuesto total

Finalmente juntamos todas las partes del presupuesto con tal de ver el coste total del proyecto y compararlo con el coste esperado.

Concepto	Coste aproximado	Coste final
Recursos humanos	18.550 €	19.310 €
Hardware	1009 €	250 €
Software	21 €	21 €
Costes indirectos	105 €	106,80€
Contingencia (5%)	984,25 €	-
Costes imprevistos	130 €	-
Total	**20.799,25 €**	**19687,80€**

Tabla 2.5. Coste total del proyecto

Desviaciones: podemos observar que aunque existan desviaciones estas están cubiertas con el apartado de contingencia que se había previsto. Además debido a las situaciones ya explicadas se ha ahorrado en el apartado de hardware.

Podemos concluir que el presupuesto total de la realización del proyecto es bastante elevado si tenemos en cuenta que no se espera obtener ninguna remuneración económica por los resultados obtenidos.

2.6. Sostenibilidad y compromiso social

A continuación presentamos el estudio sobre sostenibilidad que hemos hecho referente al proyecto.

2.6.1. Económica

En secciones anteriores se muestran los costes relacionados con el proyecto. Estos costes incluyen tanto los recursos materiales como los humanos para llevar a cabo el proyecto, pero no contempla el coste de futuras actualizaciones o el mantenimiento. Por lo tanto, si alguien quiere hacer una nueva versión o actualización en el futuro, debe hacer un nuevo presupuesto con sus objetivos para su versión del proyecto.

El coste de este proyecto es ciertamente viable, puesto que más de la mitad del presupuesto está destinado a pagar los gastos de la oficina y la posible contingencia del proyecto. Eso significa que el proyecto no es caro y si se replicara un proyecto más barato sólo se podría replicar trabajando más rápido que en esta ocasión. Esto implicaría menos gastos de costes indirectos debido a que trabajaríamos por un menor número de meses. Tal vez se podría ahorrar en recursos humanos simplemente por trabajar menos horas. Eso significaría un proyecto con un resultado de menor calidad de lo esperado.

La puntuación dada en este apartado de la tabla que se encuentra más adelante se basa en los recursos humanos necesarios, el tiempo invertido en el proyecto y la cantidad de trabajo realizado en este proyecto.

2.6.2. Social

Se desarrolla este proyecto para ser utilizado en un contexto académico, sólo por la FIB. El objetivo de este proyecto es el de ofrecer una herramienta, una práctica para la asignatura de SOAD. Podríamos decir que ayudaría tanto a profesores como alumnos. Ofreciendo a ambos un método diferente a las practicas convencionales y también haciéndola más realista y atractiva a los alumnos. También ofrecerá a los alumnos unas pautas de análisis que podrán aplicar a distintos juegos y/o programas.

Si se completa el objetivo del proyecto también podría tener alguna repercusión negativa ya que en los primeros cuatrimestres en los que se aplicara se debería ver el encaje real del proyecto. Teniendo que ajustarse tanto a alumnos como profesores.

Para establecer una puntuación en esta dimensión, hemos evaluado el volumen de personas involucradas y el impacto social de este proyecto puede tener. La puntuación dada en este apartado de la tabla se encuentra más adelante.

2.6.3. Ambiental

Hoy en día no cuesta pensar que la contaminación se ha abierto paso por todos los ámbitos de la vida a pesar de que el ser humano parece más comprometido con el medio ambiente. Por este motivo, durante la realización del proyecto se ha intentado reducir los recursos al mínimo para reducir su impacto. Así pues, aunque todos los elementos que podrías ser perjudiciales para el medio ambiente que incorpora el proyectos son indirectos no se puede obviar que en su proceso de obtención si contribuyen en aumentar la huella ecológica.

Este proyecto y sus predecesores pueden complementarse para dar ideas o cambiar en proyectos futuros. Con esto se quiere decir si se hiciera otro proyecto basándose en este se reduciría el tiempo de trabajo y recursos usados. La puntuación dada en este apartado de la tabla se encuentra en el siguiente apartado.

2.7. Informe de sostenibilidad

Ahora que hemos analizado los tres ámbitos, se debe asignar una puntuación a las diferentes partes en que se divide el proyecto, según el punto de vista para dar constancia del trabajo con las tres dimensiones de la sostenibilidad. Se evalúa tanto la puntuación durante la planificación, como por los resultados obtenidos y el riego futuro.

Sostenibilidad	Económica	Social	Ambiental
Planificación	8	8	6
Resultados	7	7	7
Riesgo	0	0	0
Valoración Total	43		

Tabla 2.6. Análisis de sostenibilidad

3. Preparativos, configuraciones y resultados de los análisis

Hablemos ya en este punto de los preparativos, las configuraciones y los resultados de los análisis. Este apartado pretende sentar las bases del proyecto, tanto para el alumno que está realizando este proyecto como para los que realicen la práctica.

3.1. Preparativos

Lo primero que se hizo en esta sección fue decidir que la opción más conveniente para este proyecto era utilizar un juego ya creado. Se tomo esta decisión porque añadiría un reto al alumno, ya que normalmente este está acostumbrado a códigos limpios, realizados por los profesores de la universidad con un estilo claro y marcado. En cambio un juego ya creado normalmente supone la intervención de diferentes programadores con estilos diferentes de programación. Otro motivo para tomar esta elección fue simplemente el tiempo disponible para realizar el proyecto ya que idear un juego con las características necesarias comportaría mucho tiempo y conocimientos de los que no disponía.

Algunos de los requisitos para el juego eran que fuera de código abierto y que tuviera un modo multijugador ya que para este proyecto se requiere que el juego tenga una estructura cliente servidor. Solo estos dos requisitos ya acotó mucho la búsqueda del juego. Los dos juegos finalmente seleccionados para probar fueron Truecraft[13] y Teeworlds[14].

Al principio se escogió Truecraft (una implementación de Minecraft beta 1.7.3) dado el éxito que tienen últimamente los juegos basados en cubos. No tardaron en surgir los problemas de mala interacción, fallos de texturas y bloqueo total por parte del juego. Es por estos motivos que se descarto usarlo.

Pasamos entonces al juego de Teeworlds (un juego en donde hasta 16 usuarios se pueden enfrentar en diferentes modos de juego), este juego tiene una base mucho mas solida, incluso esta actualmente publicado en la plataforma de Steam, y no suele dar errores.

Una vez ya decidido el juego se paso a familiarizarse y comentar el código de las zonas más importantes del servidor. Para realizar el comentado del código se empezó buscando los main de las partes principales del programa, es decir del cliente y del servidor. Una vez localizados y para asegurarnos de que las partes importantes en esta práctica no quedaran poco comentadas o sin comentar se localizaron todas las zonas del código donde hubiera interacción entre el cliente y el servidor.

Con tal de cumplir este propósito se busco en el código palabras clave que siguieran esta corriente de pensamiento. Así pues se buscaron palabras como “send”, “recovery”, “read”, “socket”, etc. Con las zonas del código que nos devolvió la búsqueda, los main y las herramientas que nos proporciona Eclipse para buscar las llamadas a las funciones se comentó el código. Estos comentarios son básicamente una descripción de las funciones que realiza, las condiciones que se deben de cumplir o aclaraciones de lo que hace una llamada o una sección del código.

Se han aprovechado los propios comentarios como una forma de guiar al alumno a las zonas que más le puedan interesar para realizar la práctica, así pues las zonas de comunicación y procesado de información recibida están más comentadas que el resto.

La parte más positiva de haber realizado este comentado de código usando estas formas de hacer o metodología es que se puede exportar fácilmente para comentar los apartados que mas interesen, por ejemplo, si deseas buscar los protocolos de comunicación puedes buscar palabras como “udp”, “tcp”, etc. En canvio si te interesa buscar como se están usando los gráficos puedes buscar palabras como “OpenGL”, “Glut”, para determinar las zonas que te interesan trabajar y comentar.

3.2. Configuraciones

Teeworlds utiliza un programa externo llamado Bam para compilarse, eso provoca que para cada plataforma utilice una configuración diferente. Es por ello que se dejara implementada una configuración diferente para Linux y para Windows y que el alumno tenga la libertad de escoger cual prefiere.

Para realizar el proyecto además se tuvo que añadir las librerías de openMP. Estas librerías son añadidas en Windows a través de Microsoft Visual Studio que incluye la versión 2.0 de openMP. En consecuencia el alumno deberá revisar la configuración actual de su ordenador y seguir las instrucciones para adaptar la compilación. En cuanto a Linux el alumno solo deberá adaptar la localización de Bam ya que el resto de librerías suelen estar incluidas en las versiones actuales de Linux, en caso contrario deberá instalar-las, se dejaran especificadas que librerías serán necesarias en la documentación para el alumno.

Además de las configuraciones anteriores se añadirá una más par Linux adaptada para el uso de la herramienta Paraver. Esta herramienta, desarrollada en el Barcelona Supercomputing Center (BSC), nos permite ver que está ocurriendo en los threads en todo momento y nos permite crear flags para ver eventos que puedan interesar al usuario. Así pues esta configuración está pensada para que el alumno pueda discernir si las mejoras que está haciendo en el código son significativas y pueda localizar errores de ejecución.

Por último se dará a los alumnos una configuración estándar del servidor, además se les incluirá los parámetros que acepta por si decidieran, por ejemplo, modificar el mapa, el modo de juego, etc.

3.3. Resultado de los análisis

3.3.1. Análisis: estructura de datos

Los datos están estructurados a lo largo de muchas clases. Todas estas están conectadas directa o indirectamente con la clase Server y con la clase Client. En este apartado solo nos centraremos en aquello que esté conectado con la clase Server y sea importante a la hora de gestionar la partida. A continuación se dará una pequeña explicación de cada una de ellas.

La clase CServer es la clase principal, se encarga de llevar la información de unas clases a otras, gestionar los mensajes que llegan, se encarga de hacer llegar la información referente a la partida a la clase Cliente y distribuir el tiempo de ejecución en las otras clases y operaciones.

La clase CGameContext es la encargada de gestionar el estado de la partida que se está llevando a cabo, mantener los estados de los jugadores y contadores (tiempo transcurrido de partida, daño, puntuación, etc), se encarga de hacer llegar la información referente a la partida a la clase Player, de enviar las nuevas informaciones generadas a los clientes y de comprobar las condiciones de victoria de la partida.

La clase CClient es básicamente una clase diseñada para almacenar información del cliente, guarda informaciones como el estado del cliente, la latencia, la última información enviada y recibida, las últimas acciones cometidas por el cliente y la información personal y de la partida de este(puntuación, nombre, clan, etc).

La clase CPlayer es la encargada de gestionar la información del jugador que manda el cliente, de gestionar la muerte y resurrección del jugador, de asignarlo a un equipo y guardar la información de la partida que le será enviada a través de CServer.

La clase CCharacter es la clase encargada de gestionar la información referente al personaje del jugador y las acciones que realice. Así pues gestiona las armas, el movimiento, los disparos, la vida, la armadura y las acciones que realice el jugador,

La clase CEventHandler se encarga de crear, almacenar y preparar para enviar todos los eventos que se puedan dar durante la partida. Estos eventos pueden ser la creación de un proyectil, la aparición de un powerUp, la resurrección de uno de los jugadores, etc.

La clase CSnapshot y sus derivadas (CSnapshotDelta, CSnapshotBuilder, etc) son las encargadas de guardar, comprimir y preparar toda la información de la partida para ser enviadas a través de la clase Server.

La clase CNetServer es la encargada de trasladar del socket al programa y del programa al socket toda la información que llega o se debe enviar. También se encarga de mantener viva la comunicación y de gestionar las conexiones entre el servidor y sus clientes.

Por último mencionar que al estar observando las clases se puede ver que no es del todo optima la gestión de la memoria, que se sobredimensiona en algunos casos en la propia definición de la clase y que algunas variables en las operaciones son creadas y destruidas constantemente cuando podrían ser simplemente creadas una vez y sobrescritas posteriormente como es el caso de las variables aData, aDeltaData y aCompData en la operación DoSnapshot de la clase CServer

3.3.2. Análisis: gestión de recursos

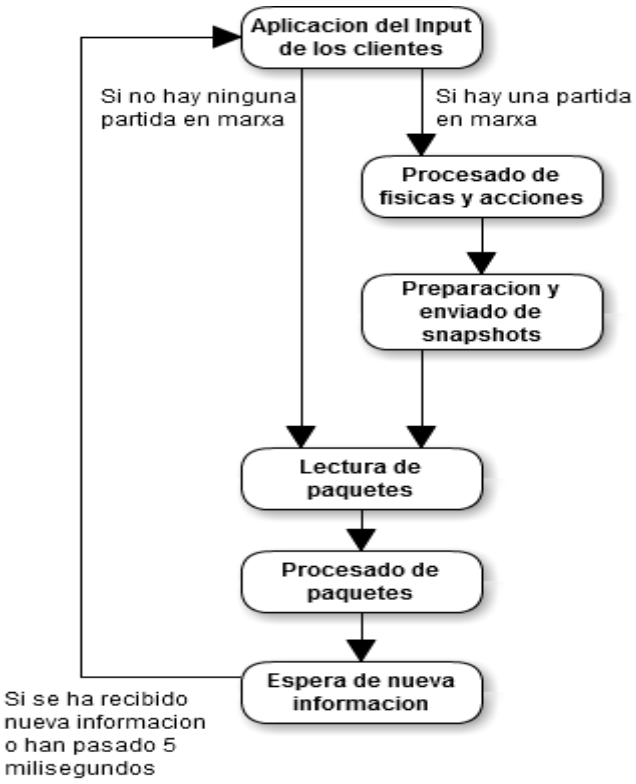
Al estar el juego escrito en C++ este no tiene un Garbage Collector que se dedique a limpiar la memoria no utilizada, por ello cuando la ejecución del programa sale del scope donde ha sido declarada una variable la elimina. Otro aspecto que debemos tener en cuenta es como se pasa la información en las llamadas ya que puede hacer crecer rápidamente la memoria si no se pasa correctamente, podríamos estar copiando grandes cantidades de memoria solo pasando un objeto cuando a lo mejor podemos usar el objeto original.

Una vez leído todo el código y observado cómo, donde se declaran las variables y especialmente como estas se pasan a las operaciones podemos concluir que el juego hace una buena gestión de la memoria. Esto es debido a que las variables usadas durante toda la ejecución, como pueden ser todos los componentes referentes a los jugadores, están declaradas globalmente, así toda operación dentro de la misma clase puede acceder a estos datos sin ser necesario que le pasen la información o los punteros a esta. Además siempre que una operación pertenece a otra clase y es necesario pasar grandes cantidades de información esta es pasada por punteros, sin necesidad de hacer copias innecesarias. Por último añadir que las nuevas variables se crean cerca de su contexto, es decir su uso, scope, es óptimo y no se puede reducir.

3.3.3. Análisis: Concurrencia de ejecución

Actualmente el servidor del juego se ejecuta totalmente en secuencial. Se ha llegado a esta conclusión después de haber buscado librerías de gestión de paralelismo y solo encontrar la librería pthread. Los únicos puntos donde se usa paralelismo son en cliente los que se encargan de la gestión de los sonidos y de la búsqueda de hosts. Eso sí, podemos empezar a ver donde puede ser interesante paralelizar el código y que partes del código deben ser secuenciales entre ellas aunque puedan contener paralelismo dentro de estas partes. Podemos observar claramente la dependencia secuencial entre las siguientes partes.

Primero se ejecuta la parte del código donde se aplica la última información enviada por los clientes, a continuación se aplican las físicas y los inputs enviados por los jugadores, después se preparan los snapshots que serán enviados a los clientes con la nueva información, seguidamente se leen primero todos los paquetes y después se comprueban y se dejan preparados para la siguiente vuelta del bucle, por último se espera a la llegada de nueva información. Podemos observar lo explicado en el esquema de a continuación:



3.1. Bucle principal del programa

Se observan tres zonas del código que pueden ser susceptibles de paralelismo dependiendo del modelo escogido. Estas zonas son las de aplicación del input, preparación y envío de snapshots y el procesamiento de los paquestes.

3.3.4. Análisis: Integridad de los datos

En cuanto la integridad de los datos, al analizar el código, se ha podido observar que los datos en ningún momento están expuestos a los usuarios.

No se pueden modificar vía paquetes ya que el formato de los paquetes es comprobado cuando entran en el servidor y se comprueba si está dentro de los parámetros permitidos. En caso contrario el mensaje se descarta.

No se pueden modificar vía mensaje ya que también estos son comprobados y en caso de no cumplir los parámetros de los mensajes estándares el mensaje es descartado.

Otra posible forma de modifica los datos seria mediante el acceso a consola que se le otorga a los administradores o a los moderadores. Esto no es posible dado que las acciones que se pueden realizar a través de esta consola son muy limitadas, corresponden con los ya programados y no se pueden modificar durante la ejecución del servidor.

4. Aplicación de los modelos

En esta sección se explican todas las decisiones tomadas durante la aplicación de los tres modelos comentados anteriormente. Se comentaran las cuatro partes en las que se encuentra dividido siempre y cuando estas se lleguen a realizar.

4.1. Thread-per request

En este modelo cada solicitud que realiza cada uno de los clientes se asigna a un nuevo thread.

4.1.1. Análisis

Entendiendo como funciona este modelo y observando cómo está estructurado y escrito el juego podemos ver que una implementación literal del modelo no es plausible ya que supondría el cambio total de su organización ya que cada una de las partes indicadas anteriormente se debe ejecutar de forma secuencial. No obstante se puede flexibilizar el modelo y aplicar-lo a pequeña escala dentro de cada una de las partes indicadas.

4.1.2. Diseño

En este paso se mira más en concreto el encaje del modelo con el juego actual. Cogiendo como referencia cada una de las partes antes mencionadas iremos una a una explicando porque el modelo encaja o deja de encajar o si el resultado sería o no conveniente.

> **Aplicación del input**: si bien es posible paralelizar el juego en este apartado siguiendo las referencias del modelo no se acaba de ajustar ya que no se puede distinguir de la solución que generaría un thread-per-conection, al final se paraleliza el código por los diferentes clientes y no por todos los inputs recibidos. Por tanto se declina implementar esta mejora ya que corresponde a la de un thread-per-conection.

Actualizar físicas e inputs: este apartado se podría paralelizar, el problema es que en cada thread se actualiza información constantemente y esta puede afectar a la de otros threads. Si no se pusiera ningún tipo de sincronización esto llevaría a errores, por ejemplo se podría estar haciendo daño a personajes que ya no se encuentran en una posición porque se ha movido, pero uno de los threads ha leído la información de la posición antes de ser actualizada. En caso de que hubiera zonas declaradas como críticas y sincronizaciones, al haber constantes modificaciones de la información, nos llevaría a una situación peor que la secuencial debido al overhead creado por estas operaciones.

Preparar y enviar snapshots: si bien es posible paralelizar el juego en este apartado siguiendo las referencias del modelo no se acaba de ajustar ya que no se puede distinguir de la solución que generaría un thread-per-conection ya que se envía la información por cada cliente. Por tanto se declina implementar esta mejora ya que corresponde a la de un thread-per-conection.

Lectura de los paquetes: este apartado no se puede realizar en paralelo básicamente porque la lectura del socket se debe realizar secuencialmente.

Procesar paquetes: en un principio si es posible realizar este apartado en paralelo ya que se podría modificar la lectura de los paquetes para que se almacenaran en un array o vector. Una vez almacenado se podría distribuir la faena en diferentes threads. El problema radica en que se podría dar la situación de procesar dos inputs del mismo cliente en paralelo. Si fueran clientes distintos no habría ningún problema. Al procesar dos inputs de un mismo cliente se pueda dar la situación de que uno de los threads lea una información antes de que el otro la modifique, pudiéndose dar situaciones tan estrambóticas como que se cree un proyectil de un arma sin que se haya visto que ese jugador ha disparado el arma o que un jugador dispare un proyectil que no se corresponde con el arma que sostiene. Para solucionar este problema en openMP deberíamos crear una región critica para que los threads se excluyan mutuamente. El problema es que no se excluye solo los threads que este gestionando el mismo cliente sino que a todos los threads. Este problema se genera en la zona de recepción de acciones de movimiento o de ataque del jugador que es la parte central durante la ejecución de una partida. Por todos estos motivos se declina la implementación de una mejora en esta zona ya que su implementación supondría obtener unos tiempos peor que secuenciales.

Esperar nueva información: este apartado no se puede realizar en paralelo ya que se trata de esperar información de los clientes. En caso de no recibir información en 5 milisegundos el programa realiza otra vuelta al bucle principal.

Llegados a este punto no tiene sentido pasar a la implementación y prueba ya que o no se aplican porque corresponden a otro modelo o porque no suponen una mejora real al proyecto.

4.2. Thread-per-conection

Este modelo es una variación del thread-per-request que amortizar el coste de crear y destruir el thread a través de múltiples peticiones. Este modelo pone cada cliente que se conecta con un servidor en un thread separado para la duración de la sesión.

4.2.1. Análisis

Como en el modelo anterior este tampoco se amolda ni es fácil de encajar en el juego tal y como esta es por eso que se flexibilizará. Debido a la necesidad de algunas partes del código a ejecutarse en secuencial no se asegura que el mismo thread siempre atienda al mismo cliente pero sí que un solo thread leerá y procesara todos sus paquete y que un solo thread actualizara su input por cada vuelta de bucle.

4.2.2. Diseño

En este paso se mira más en concreto el encaje del modelo con el juego actual. Cogiendo como referencia cada una de las partes antes mencionadas iremos una a una explicando porque el modelo encajo o deja de encajar o si el resultado sería o no conveniente.

> **Aplicación del input**: cómo podemos ver en el siguiente código en esta parte del programa se busca secuencialmente para cada cliente el estado de su jugador correspondiente al tick actual del servidor, pasando por todos los estados guardados de este.

```
//Aplicar nuevo input por todo cliente
for(int c = 0; c < MAX_CLIENTS; c++)
{
    if(m_aClients[c].m_State == CClient::STATE_EMPTY)
        continue;
    //Si es un jugador activo buscar el estado correspondiente al estado actual del servidor
    for(int i = 0; i < 200; i++)
    {
        if(m_aClients[c].m_aInputs[i].m_GameTick == Tick())
        {
            //Asignar el estado esperado para esta tick
            if(m_aClients[c].m_State == CClient::STATE_INGAME)
                GameServer()->OnClientPredictedInput(c, m_aClients[c].m_aInputs[i].m_aData);
            break;
        }
    }
}
```

Esta búsqueda afecta solo a las variables propias de cada cliente. Por este motivo este fragmento del código es perfectamente paralelizable, se trata cada uno de los diferentes clientes por separado sin que los datos de uno afecten a otro. Dado el tiempo que se puede tardar en encontrar el estado deseado y que podemos estar tratando hasta 16 clientes, ejecutando todos los clientes en paralelo nos aseguramos una reducción considerable del tiempo de ejecución. Simplemente se debe añadir encima del primer for la siguiente instrucción: #pragma omp parellel for Schedule(static, 3) //recordar que los parámetros del schedule son los que han demostrado mejores resultados en el ordenador usado durante las pruebas y que puede variar según el ordenador usado.

Actualizar físicas e inputs: en esta parte pasa exactamente lo mismo que en el punto anterior no se pude ofrecer ninguna mejora.

Preparar y enviar snapshots: si bien es posible paralelizar el juego en este apartado siguiendo las referencias del modelo, debido al uso que se hace de los snapshots anteriores para realizar el nuevo de cada uno de los usuarios nos podemos encontrar casos en los que el dato que queríamos consultar ya no existe provocando un segmentation fault. Para resolverlo necesitaríamos crear una región crítica de openMP. Su implementación nos supondría unos tiempos peores que secuenciales debido al overhead creado por estas operaciones. Por este motivo se declina su implementación.

Lectura de los paquetes: este apartado no se puede realizar en paralelo básicamente porque la lectura del socket se debe realizar secuencialmente.

Procesar paquetes: esta parte es totalmente paralelizable. Eso si precisa de ciertas modificaciones del código original. Como podemos observar en el código que hay a continuación, el servidor en cada iteración procesa uno de los paquetes pendientes. Así pues, lee el paquete y después lo procesa.

```
void CServer::PumpNetwork()
{
    CNetChunk Packet;
    //Por todo cliente actualizar conexion(reenvias informacion perdida, comprovar ultimo paquete que ha recibido,etc)
    m_NetServer.Update();

    //Mientras haya paquetes pendientes por leer, lee el paquete y realiza comprovaciones basicas
    while(m_NetServer.Recv(&Packet))
    {
        if(Packet.m_ClientID == -1)
        {
            //Infromacion del servidor master
            if(!m_Register.RegisterProcessPacket(&Packet))
            {
                if(Packet.m_DataSize == sizeof(SERVERBROWSE_GETINFO)+1 &&
                    mem_comp(Packet.m_pData, SERVERBROWSE_GETINFO, sizeof(SERVERBROWSE_GETINFO)) == 0)
                {
                    //Enviar informacion al master server
                    SendServerInfo(&Packet.m_Address, ((unsigned char *)Packet.m_pData)[sizeof(SERVERBROWSE_GETINFO)]);
                }
            }
        }
        else
        {
            //Procesa el paquete del cliente
            ProcessClientPacket(&Packet);
        }
    }
    //Elimina bans expirados
    m_ServerBan.Update();
    //Gestiona el acceso a la consola remota
    m_Econ.Update();
}
```

Como está estructurado actualmente el código no podemos paralelizarlo ya que la operación “Recv” no es paralelizable(modifica el estado de conexión del cliente, contadores del servidor y variables que se usan dentro de la misma operación). Al realizarla en paralelo se podrían producir errores en la lectura de los chunks de los paquetes. Para solucionarlo podríamos realizar estas lecturas en una región crítica, pero provocaríamos que el tiempo destinado a esta tarea fuera más grande que el actual ya que es la lectura del paquete su principal cometido.

Por estos motivos se decide hacer un cambio de estructura de esta operación. Ahora se almacenaran los paquetes en un vector de dos dimensiones donde cada posición es un vector de los paquetes de cada cliente. A continuación se ejecutaran por cada uno de los clientes todos sus paquetes. Siendo así posible realizar el procesado de paquetes de forma paralela. De esta forma no existirá ningún conflicto a la hora de procesar el input de cada cliente ya que tratara en cada iteración un paquete diferente. Eso si se deberá tener cuidado a la hora de enviar los mensajes ya que se pueden generar conflictos y errores.

Así pues realizaremos primero la lectura secuencialmente y a continuación el procesado de la información en paralelo. El código de la operación pasara a ser el siguiente:

```
void CServer::PumpNetwork()
{
    CNetChunk Packet;

    m_NetServer.Update();
    std::vector<std::vector<CNetChunk>> all_Packets(MAX_CLIENTS,std::vector<CNetChunk>(0));

    while(m_NetServer.Recv(&Packet))
    {
        if(Packet.m_ClientID == -1)
        {
            if(!m_Register.RegisterProcessPacket(&Packet))
            {
                if(Packet.m_DataSize == sizeof(SERVERBROWSE_GETINFO)+1 &&
                    mem_comp(Packet.m_pData, SERVERBROWSE_GETINFO, sizeof(SERVERBROWSE_GETINFO)) == 0)
                {
                    SendServerInfo(&Packet.m_Address, ((unsigned char *)Packet.m_pData)[sizeof(SERVERBROWSE_GETINFO)]);
                }
            }
        }
        else
        {
            all_Packets[Packet.m_ClientID].push_back(Packet);
        }
    }
    //Procesar paquetes
    #pragma omp parallel for schedule(static,3)
    for(int i=0;i<MAX_CLIENTS;++i)
    {
        for(int s=0;s<all_Packets[i].size();++s)
        {
            ProcessClientPacket(&all_Packets[i][s]);
        }
    }

    m_ServerBan.Update();
    m_Econ.Update();
}
```

Esperar nueva información: este apartado no se puede realizar en paralelo ya que se trata de esperar información de los clientes. En caso de no recibir información en 5 milisegundos el programa realiza otra vuelta al bucle principal.

4.2.3. Implementación

Durante la implementación se han seguido las estrategias antes comentadas. Una de las decisiones que se han debido tomar es la colocación de las aéreas críticas a la hora de enviar los mensajes. Estas aéreas críticas se han establecido en la zona mínima para que el fragmento a ejecutar atómicamente sea lo más pequeño posible evitando así grandes interrupciones entre los threads. Además se ha decidido que para la menor perdida del tiempo posible y dado que el servidor debería funcionar a pleno rendimiento el Schedule de las regiones paralelas tendrá por componentes static (distribución de las tareas estática, se le asigna la tarea al primer thread disponible) y 3 (el numero de iteraciones que cogerá cada thread). Se debe recordar que esta es la mejor configuración para el ordenador usado actualmente y que dependerá del ordenador donde se use el servidor.

4.2.4. Pruebas

Para comprobar la efectividad de la implementación se generaron dos trazas una de control en la que se ejecutaba el servidor secuencialmente y otra en la que se ejecutaba con 6 threads paralelos. Debido a la falta de medios en cambio de usar 16 clientes en diferentes ordenadores se usaron 12 clientes en 3 ordenadores. Identifiquemos a continuación las trazas en las diferentes secciones mejoradas y pasemos a comparar los resultados.

Empecemos por la aplicación del input a continuación:

Figura 4.2. Traza secuencial de la aplicación del input

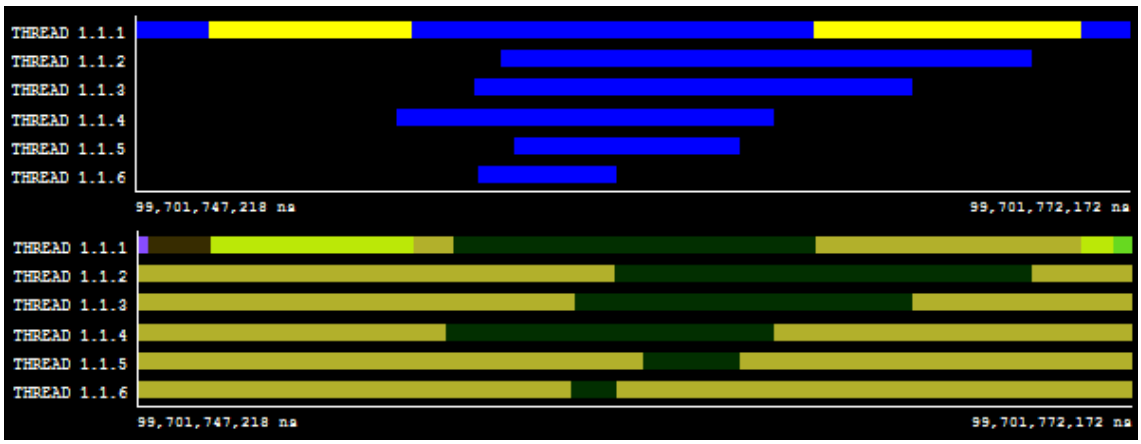


Figura 4.1. Traza paralela de la aplicación del input

Como podemos observar en las figuras anteriores existe una mejora ya que la aplicación del input pasa de tardar 52.154ns a tardar 24.954ns. Queda constatada la mejora en este apartado llegando a hacer el trabajo en la mitad del tiempo. Cabe destacar que con los suficientes threads, procesadores y la configuración más adecuada para ese caso la mejora podría ser más notable.

Pasemos ahora al procesado de los paquetes:

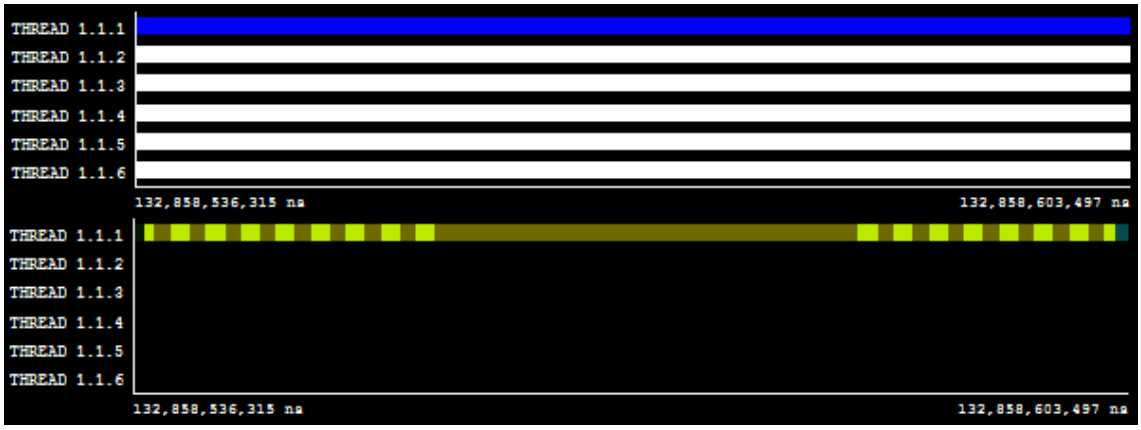


Figura 4.4. Traza secuencial del procesado de paquetes

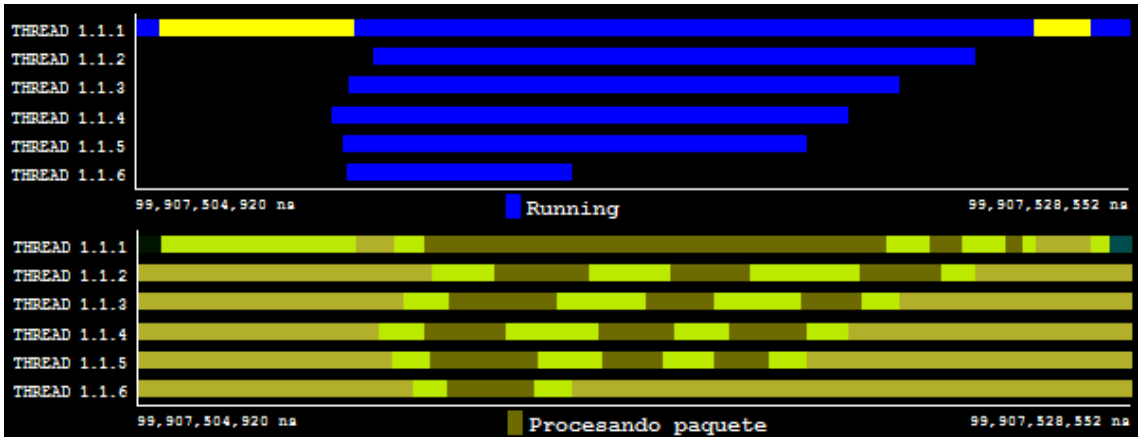


Figura 4.3. Traza paralela del procesado de paquetes

Como podemos observar en las figuras anteriores existe una mejora ya que el procesado del paquete pasa de tardar 67.182ns a tardar 23.632ns. Queda constatada la mejora en este apartado llegando a hacer el trabajo en casi un tercio del tiempo. Cabe destacar, como en el caso anterior, que con los suficientes threads, procesadores y la configuración más adecuada para ese caso la mejora podría ser más notable.

4.3. Thread-per-object

Este modelo asocia un thread para cada objeto lógico en el servidor.

4.3.1. Análisis

A diferencia de los dos anteriores este modelo no se adapta para nada al juego escogido. Al basarse en la asociación de un thread para cada objeto lógico nos encontraríamos con grandes desequilibrios ya que la mayor parte de las operaciones se ejecutan sobre el jugador y no sobre los objetos especiales encontrados por el mapa, las armas, etc. Además la comunicación con el cliente se basa en las acciones que este comete y por tanto consigo mismo y ningún otro cliente.

4.3.2. Diseño

Debido a la dificultad de comprensión y implementación (este modelo requiere estrategias de sincronización muy sofisticada) de este modelo, del remodelaje completo que se necesitaría del juego para poder adaptarlo se ha optado por descartar este modelo.

5. Practica a realizar por el alumno

Teeworlds es un juego multijugador en línea gratuito, disponible para los principales sistemas operativos. Lucha con hasta 16 jugadores en una variedad de modos de juego, incluyendo Team Deathmatch y Capture The Flag.

Después de esta pequeña introducción sobre teeworlds pasemos a explicar que debes hacer en esta práctica.

Para completar con éxito esta práctica deberás analizar el juego en estos aspectos:

- Gestión de recursos.
- Concurrencia de ejecución.
- Integridad de los datos.

Una vez analizado el juego y redactado tus conclusiones deberás diseñar e implementar una mejora en el servidor mediante uno de los métodos enseñados en la asignatura.

Se proporciona:

- El juego comentado para una rápida familiarización con él.
- Archivos de configuración para cargar el juego en Windows y Linux (Lee el archivo readme-soad para saber que archivos cambiar para usar una configuración o otra).
- La configuración necesaria para usar el juego con la herramienta Extrae(solo se puede usar en Linux).
- Un esquema de la comunicación entre cliente y servidor.

Se pide:

- Las conclusiones de los tres análisis realizados al juego.
- Que se explique cómo se ha adaptado el modelo escogido al juego y que modificaciones se han tenido que hacer.
- En qué proporción mejora la implementación a la versión secuencial del juego.

Para realizar este último apartado aprovecha las herramientas Extrae y Paraver proporcionadas. Lea el archivo readme-soad del directorio para entender como están implementados en el juego original y entender cómo usarlos.

6. Conclusiones

Después de finalizar el desarrollo de este proyecto, puedo decir que estoy satisfecho con el trabajo hecho y puedo afirmar que los objetivos marcados cuando empezó el proyecto han sido cumplidos.

Se ha cumplido el principal objetivo de crear una práctica para la asignatura de SOAD basándose en un juego. Se han desarrollado toda la documentación que puede necesitar el alumno para realizar la práctica y comentado el código que puede necesitar. Se han realizado los análisis que deberán realizar los alumnos para contrastar la información. Se ha adaptado el juego para que acepte la herramienta Extrae que permitirá al alumno visualizar que sucede en los threads del servidor. Se ha analizado y diseñado los diferentes modelos aplicados al juego y se ha implementado el que ofrecía más expectativas de mayor eficiencia, expectativas que se han cumplido.

La practica pues, siempre y cuando cuente con el beneplácito del equipo de profesores de la asignatura, está preparada para ser introducida a partir del próximo cuatrimestre. Espero que los alumnos disfruten de la realización de esta práctica y aprecien el esfuerzo y la buena voluntad puesto en este proyecto.

Por último agradecer la oportunidad que representa colaborar con el desarrollo de material educativo. Como alumno debo admitir que uno nunca se plantea el esfuerzo que conlleva realizar todo este material y que normalmente no está lo suficiente valorado. Ver el otro lado de la moneda es un buen ejercicio para percatarse de las ganas e ilusión que se ponen en la innovación para la educación.

7. Trabajo futuro

Para poder ejecutar estos programas y sus complementos será necesario que el equipo que se encarga de las imágenes de la FIB se asegure que las siguientes librerías estén presentes en OpenSuse:

- libxml2 2.5.0
- libunwind
- OpenMP
- Boost versión 1.36 o superior
- Zlib
- wxWidgets versión 2.8.0 o superior
- wxPropertyGrid versión 1.4.0 o superior
- Python
- Alza
- Gl
- Glu
- X11
- Libsdl
- Freetype

Con este mismo juego se podrían realizar otras prácticas que incluso podrían ser para otras asignaturas. Por ejemplo podríamos entrar a valorar la configuración de los sockets que se realiza y cuan adecuada es. Otra alternativa podría ser utilizar este mismo juego, programando los clientes, para realizar competiciones entre los alumnos de Algoritmia o Inteligencia Artificial y probar sus conocimientos. También se podría mirar más en profundo la gestión grafica ya que este juego usa opengl.

8. Referencias

[1] *Asignaturas del grado - Facultad de Informática de Barcelona - FIB - Universidad Politecnica de Cataluña - UPC - BarcelonaTech*. (2016). Retrieved from http://www.fib.upc.edu/es/estudiar-enginyeria-informatica/assignatures.html?assig=SOAD.

[2] *Asignaturas del grado - Facultad de Informática de Barcelona - FIB - Universidad Politecnica de Cataluña - UPC - BarcelonaTech*. (2016). Retrieved from http://www.fib.upc.edu/es/estudiar-enginyeria-informatica/assignatures.html?assig=EDA.

[3] Arias, D., Bustinza, O.F. & Djundubaev, R. (2015). Efectos de los juegos de simulación de empresas y Gamification en la actitud emprendedora en enseñanzas medias. *Revista de Educación*, 371, 133-156.

[4] Bayliss, J. D. (2007). The Effects of Games in CS1-3. *Proceedings of the Microsoft Academic Days on Game Development in Computer Science Education*, 59-63.

[5] Lander, R. N. & Callan, R. C. (2011). Casual Social Games as serius Games: *The Psycology of Gamification in Undergraduate Education and Employee Training*. Wiley.

[6]Reuss, R. L. & Gardulski, A. F. (2001). An interactive Game Approach to Learning in Historical Geology and Paleontologym *Journal of Geoscience Education*, 49(2), 120-129.

[7] Coller, B. D. & Scott M. J. (2009). Effectiveness of using a video game to teach a course in mechanical engineering. *Computers & Education*, 53, 900-912.

[8] C. Schmidt, D., & Vinoski, S. (1996). *Comparing Alternative Programming Techniques for Multi-threaded Servers*. Retrieved, from http://www.cs.wustl.edu/~schmidt/PDF/C++-report-col5.pdf.

[9] *Scrum (desarrollo de software) – Wikipedia, la enciclopedia libre*. (2016). Retrieved from https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software) .

[10] *OpenMP*. (2016). Retrieved from http://openmp.org/wp/ .

[11] *Extrae | BSC_CNS*. (2016). Retrieved from https://www.bsc.es/computer-sciences/extrae .

[12] *Paraver | BSC_CNS*. (2016). Retrieved from https://www.bsc.es/computer-sciences/performance-tools/paraver .

[13] *TrueCraft*. (2016). Retrieved from https://truecraft.io/ .

[14] *Teeworlds*. (2016). Retrieved from https://www.teeworlds.com/ .

[15] *OpenMP in visual C++*. (2016). Retrieved from https://msdn.microsoft.com/en-us/library/tt15eb9t.aspx .

[16] *OpenMP – Wikipedia la enciclopedia libre*. (2016). Retrieved from https://es.wikipedia.org/wiki/OpenMP .