TREBALL FINAL DE GRAU

“ORANGE JUICE”

ARNAU ESCRIBÀ TARDÀ
CURS 2012/2016

Index

1. Introducció 3
2. Objectius 3
2.1 Objectius Generals 3
2.2 Objectius Específics 3
3. Disseny del Joc 4
3.1 Idea del joc 4
3.2 Referències 5
3.3 Breu Game Design 7
3.3.1 Mecànica de moviment: 7
3.3.2 Motivadors: 7
3.3.3 Objectius 8
3.3.4 Modificadors 8
3.3.5 Perfils de jugador 9
3.4 Disseny 3D 9
3.4.1 Realització dels diferents elements en 3D 9
3.4.2 Modelat 9
3.4.3 Integració al *Unity3D* 15
3.4.4 Creació de animacions 15
3.5 Disseny 2D 17
3.5.1 Realització de la *UI* 17
3.5.2 Realització de petites animacions 17
4. Desenvolupament 18
4.1 Cerca dels diferents serveis de base de dades 18
4.1.1 Base de Dades relacional o no relacional. 18
4.1.2 Quina base de dades no relacional s'escull? 19
4.2 Cerca d'eines per la fàcil vinculació 20
4.3 Cerca de la teoria per la vinculació i proves. 20
4.4 Connexió del prototip amb Facebook. 21
4.5 Programació del prototip 23
4.6 Plataforma 28
5. Test d'usabilitat 30
6. Difusió 31
7. Planificació 32
8. Planificació de recursos i pressupost 33
9. Conclusions 35
1. Agraïments 36
2. Annexos 36

1. Introducció

Un videojoc casual és un videojoc dirigit a un públic de jugadors ocasionals. Aquest tipus de joc pot tenir diferents mecàniques i ser classificat en qualsevol gènere. Són distingits per les seves regles simples, i comparat amb altres jocs, no exigeixen molt compromís, fent que no sigui necessari cap esforç de recordar el funcionament després de un temps sense jugar-hi. No requereixen molt temps de dedicació o habilitats especials per jugar i , els costos de producció i distribució són baixos.

Tot i que últimament aquest tipus de joc està en creixement, ja fa temps que en disposem. Un exemple clar és el *Snake* de *Nokia*, desenvolupat per *Gremlin* a mitjans dels anys 70 i que ha sigut utilitzat com a base de molts jocs.

Cada vegada són més els usuaris que es decideixen a instal·lar aquest tipus de joc en els seus dispositius i es preveu que augmenti la quantitat donada per la millora de prestacions i emmagatzematge dels *smartphones*.

En aquest treball de final de grau s'ha realitzat un prototip d'un videojoc casual per a dispositius mòbils, connectat a una base de dades i a la xarxa social *Facebook*.
El joc tracta d'una taronja que cau per una espremedora, en el que s'ha d'anar esquivant obstacles i recollint fruita per aconseguir la major puntuació i escalar posicions en el rànquing.

2. Objectius

2.1 Objectius Generals

La creació d'un prototip de videojoc en el que s'anirà augmentant la dificultat a mesura que s'avança en el nivell. En aquest prototip l'usuari s'hi podrà registrar per mitjà d'un formulari o via *Facebook*. A més a més, el joc estarà vinculat a una base de dades, la qual cosa permetrar saber la puntuació global i crear una llista pròpia d'amics que juguen en aquest. També es vincularà amb xarxes socials com el *Facebook*. I finalment, s'adaptarà aquest prototip als diferents dispositius mòbils per poder gaudir d'una experiència més agradable.

2.2 Objectius Específics

1. Creació d'un prototip d'un videojoc en *Unity3D*.
2. Investigació sobre la connexió del *Unity3D* amb una base de dades.
3. Investigació sobre la api de *Facebook* i la incorporació en el prototip.
4. Disseny en 3D amb el programa *Cinema 4D* de l'escenari i el personatge principal.
5. Disseny en 2D del logo, la UI i altres imatges bidimensionals.
6. Adaptació pels diferents dispositius.

3. Disseny del Joc

3.1 Idea del joc

L'objectiu principal és crear un joc casual que permeti complir els objectius marcats en aquest projecte. Per poder escollir la millor opció s'ha cercat els diferents tipus de joc i classificacions. Mirant la classificació feta per *Estallo* (autor de "Videojuegos. Juicios y prejuicios", 1995) ordena els jocs de la següent manera:

- Els **jocs arcade** són aquells que es necessita un ritme ràpid de joc, temps de reacció breus i atenció focalitzada.
- Els **jocs de plataforma**, on el jugador es troba en un escenari bidimensional, i només es pot desplaçar de dreta a esquerra i de dalt a baix.
- Els **laberints**, on l'escenari principal és un laberint de grans extensions.
- Els **jocs de competicions esportives**, que com el propi nom diu es redueix a reproduir jocs de competició de diferents esports.
- Els **jocs d'acció**, que tenen un desenvolupament lineal ràpid que va variant al eliminar els enemics necessaris.
- Els **jocs de simulació,** encarregats de simular activitats i/o experiències rarament viscudes durant la vida. Permeten al jugador posar-se a la pell de situacions o tecnologies virtuals. La baixa influència del temps de reacció, la utilització de estratègies complexes i la necessitat de posar-se en situacions que exigeixen d'un coneixement específic (moltes vegades es dóna un manual d'ús), són algunes de les seves característiques principals.
- Els **jocs estratègics**, on el jugador agafar un paper específic i només coneix l'objectiu final. Habitualment els personatges venen del món de la literatura i el cine.
    - Aventures gràfiques: on l'acció succeeix per mitjà de les ordres del jugador juntament amb la possessió d'objectes que van apareixen als escenaris.
    - Rol: en aquest cas el jugador pot controlar a més de un protagonista amb característiques diferents.
    - Jocs d'estratègia militar: són molt semblants als jocs de taula d'estratègia.
- Reproduccions de **jocs de taula**, segueixen el mateix desenvolupament que els jocs de taula originals.

Tot i que Garner (1992) classifica el jocs segons la perspectiva del jugador, el camp de percepció, les capacitats interactives i l'escenari del joc, s'ha considerat millor classificació la de l'autor Estallo encara que les dos siguin vàlides.

Un cop es té la classificació dels jocs s'ha de diferenciar entre les diferents companyies productores. En primer lloc es troben les grans companyies amb centenars de treballadors. Disposen de milions d'euros en pressupost, utilitzats pel desenvolupament i el màrqueting. D'aquí surt els famosos títols AAA que es coneixen en tot el món (Jocs desenvolupats per grans empreses amb grans pressupostos). Un exemple clar és *Call of Duty*.

I després hi ha els estudis independents, que rarament sobrepassen els deu treballadors, no tenen finançament i per tant el pressupost és molt més reduït. Tot i així, el fet de no tenir un editor que posa els diners els permet ser més lliures pel que fa a la creació.

Finalment també podem classificar a gran escala dos tipus de jugadors segons el temps que dediquen a jugar i del coneixement que tenen sobre aquests. Un d'ells són els ***Casual Gamers,*** són jugadors esporàdics que dediquen poc temps a jugar i, principalment, consumeixen jocs casuals. Poden tenir perfils molt diferents. Aquests jugadors han augmentat durant aquests últims anys degut a l'adaptació i integració de videojocs en els dispositius mòbils.
L'altre tipus són els ***Hardcore Gamers.*** Es tracta de jugadors completament diferents als anteriors, dediquen molt temps a jugar a videojocs i consumeixen tota classe de jocs, tot i que preferiblement juguen més als triple A.

Per tot l'esmentat i per poder invertir més temps en el desenvolupament dels objectius i no tant en la creació d'una narrativa molt treballada, s'ha seguit un gran nombre de mecàniques, dinàmiques i un disseny gràfic més acurat.
S'ha escollit que la historia d'aquest joc tingui com a protagonista una taronja que cau per una espremedora industrial en el que el jugador ha de moure el personatge amb l'acceleròmetre del mòbil per evitar els obstacles i aconseguir diferents fruites. Aquestes últimes els donarà més punts i servirà per rebre un premi. La puntuació final és mesura amb la suma del temps que has pogut aguantar esquivant objectes i la suma de fruites recollides durant aquest procés.

Tot això fet de tal manera perquè sigui un *Casual Game*, és a dir, un joc casual en el que es juga en moments morts del dia com per exemple, trajectes, quan tens una estona esperant el teu torn, deu minuts de descans quan arribes a casa, etc.

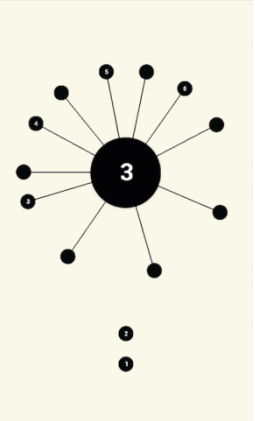
3.2 Referències

Un cop se sap la idea del joc i com ha de ser, s'ha cercat diferents jocs que serveixen per agafar alguna idea sobre aquests tipus de joc. Per fer aquesta cerca s'ha tingut en compte que el joc sigui de fàcil control amb una gràfica atractiva però poc demanada, el seu temps mínim de joc sigui molt petit i ha de ser molt addictiu. Algunes d'aquestes referències són les següents:

- Temple Run2: amb una jugabilitat fàcil i una gràfica atractiva. S'ha de portar el personatge a través de la selva i les coves, esquivant obstacles i obtenint premis i poders.

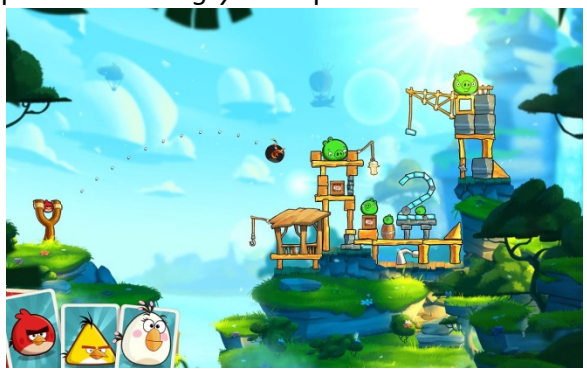
  - Google play:
    - https://play.google.com/store/apps/details?id=com.imangi.templerun2&hl=es

- Aa: amb una jugabilitat i addició molt alta. Tracta de connectar els diferents cercles amb el cercle central sense tocar cap altre cercle.



  - Google play:
    - https://play.google.com/store/apps/details?id=com.aa.generaladaptiveapps&hl=es

- Angry Birds: escollit per la seva popularitat, per la seva jugabilitat i la seva addicció. S'utilitza els exclusius poders dels *Angry birds* per acabar amb la defensa dels porcs.

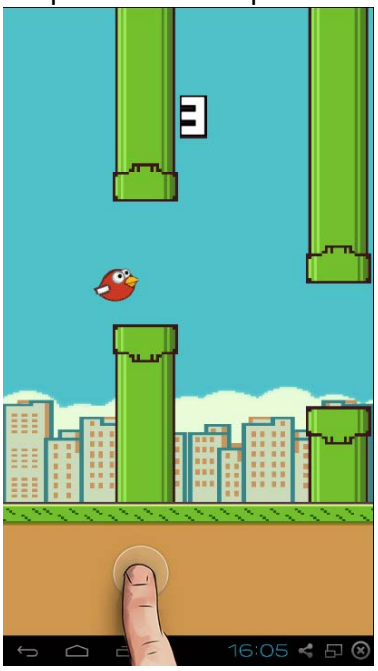


  - Google play:
    - https://play.google.com/store/apps/details?id=com.rovio.angrybirds&hl=es

- Fruit Ninja: Mític videojoc causal amb un bilió de seguidors que el recolzen. Tracta de tallar fruites amb el dit evitant obstacles, i destaca per la seva jugabilitat i gràfica.

  - Google Play:
    - https://play.google.com/store/apps/details?id=com.halfbrick.fruitninja&hl=es

- Flappy Bird: Per últim el joc per excel·lència i que ja no està en el mercat de *Google Play*. Donat de baixa pel seu desenvolupador durant el 2014 per l'alta addicció que provocava. Els mateixos desenvolupadors de *Android* el van fer servir de guia pels seus jocs en les versions de *Lolipop* i *Marsmallow.* Tractava de superar els obstacles en un escenari 2D mitjançant salts que es produïen quan tocaves la pantalla.



3.3 Breu Game Design

3.3.1 Mecànica de moviment:

En aquest prototip només es disposa de la mobilitat per acceleròmetre quan el mòbil està en horitzontal. Tot i així disposarà d'una programació vertical per millorar la jugabilitat. No obstant, en un futur la idea d'aquest prototip és que al finalitzar la partida si es disposa d'un bonus s'obtindrà una recompensa. Aquesta recompensa podrà donar extres en el moviment, com per exemple: saltar, visió de caixes invisibles, radar amb els pròxims obstacles, etc.

3.3.2 Motivadors:

Bàsicament els humans estan motivats per necessitats biològiques, recompenses i càstigs. Es poden dividir en dos grans blocs, els intrínsecs i els extrínsecs.

Intrínsecs*:*

Els motivadors intrínsecs són aquells que generen interès per ells mateixos. Aquest aspecte és important ja que s'activen àrees de plaer del cervell. Per tant és fonamental que un joc tingui motivadors intrínsecs ja que és l'encarregat de tenir els jugadors compromesos amb el joc.
La SDT (Self Determination Theory) identifica 3 motivadors intrínsecs:

- Competència: adquirir noves habilitats.
- Autonomia: desig de sentir-se lliure.
- Relació: desig d'estar connectats a altres.

En aquests prototip es podem trobar dos d'aquest tres motivadors. Com s'ha dit anteriorment, al finalitzar la partida, segons el teu resultat es disposarà d'un bonus on es podrà adquirir noves habilitats.

També es pot observar el motivador de la relació, ja que es pot connectar al *Facebook*, compartir resultats, competir en una taula de puntuació...

Extrínsecs*:*
La seva funció és donar suport i reforçar la motivació en períodes curts de temps. Aquests motius que impulsen a les persones a realitzar les accions són alienes a la pròpia acció. Un exemple clar és el regal d'una motocicleta per jugar a aquell joc.
En aquest projecte no s'ha utilitzat motivadors d'aquest estil, ja que són els menys efectius i de menys duració.

3.3.3 Objectius

És quelcom que s'aconsegueix mitjançant el mèrit, i suposen reptes ja que s'han d'aconseguir. Els objectius impliquen decisions i accions, això fa que actuï com estímul. Es poden expressar de manera precisa o imprecisa:

- Precisa: agafar una moneda, arribar en un temps determinat
- Imprecisa: buscar menjar, vestir-se.

Depenen d'aquest grau de precisió influirà en la motivació per realitzar-lo. Quan els jugadors ho aconsegueixen, provoca una retroalimentació per tornar a buscar nous objectius. La gran dificultat es trobar una òptima gestió entre els diferents objectius:

- Objectius grans: relacions socials, canviar el món.
- Objectius mitjans: nou poder, aconseguir una missió, superació nivell, premi...
- Objectius petits:
    - Reforçadors: adquirir habilitats, aconseguir ítems especials
    - Recol·lectores: aconseguir monedes, vides…

Així doncs, tenim com a **objectiu petits:** Poder aconseguir les fruites durant la partida.
Alguns exemples d'**objectius mitjans:** Aconseguir les tres fruites necessàries per una major recompensa.
I per finalitzar l'**objectiu més gran**: Sobreviure el major temps possible.

3.3.4 Modificadors

Els modificadors són elements que permeten la variació del ritme i la dificultat de la situació del joc.
Els introduïts en aquest prototip són els següents:

- Augment de la velocitat de la càmera a mesura que s'avança en el nivell.
- Diferents obstacles, uns maten, els altres obstrueixen.
- Posició de les fruites durant la partida.
- Canvis d'escenari.
- Canvis d'il·luminació.
- Mode aleatori.

3.3.5 Perfils de jugador

Podem classificar als jugadors depenent de la seva manera d'actuar, segons Andrzej Marczewski. En aquest joc tenim un perfil dominant, els *killers*. Són els que volen ser els millors, competir amb els seus amics o fins i tot, en "rankings" mundials. Per què aquest usuaris encara tinguin més competitivitat, també està contemplada una carrera a temps limitat entre jugadors. D'aquesta manera reforçaríem aquest perfil.

Malauradament serà més difícil trobar perfils de jugadors com ara *archievers* (són els que exprimeixen més el joc i busquen totes les maneres possibles d'acabar-lo*)*, *socializers* (necessiten compartir-ho amb més gent) o *explorers* (són els jugadors que necessiten escenaris molt grans per poder-los explorar).

3.4 Disseny 3D

3.4.1 Realització dels diferents elements en 3D

Per aquest prototip s'ha realitzat els elements 3D necessaris per poder realitzar el prototip correctament. No obstant hi ha altres models 3D pensats per si aquest prototip segueix en desenvolupament.
Per modelar tots els elements 3D s'ha fet servir el programa *Cinema 4D*, un programari ja utilitzat anteriorment i per tant que ja es té coneixença.

3.4.2 Modelat

En el procés de modelat s'ha utilitzat la tècnica més adequada per cada objecte. En la taronja per exemple, pel cos i els ulls es va partir d'una esfera, seguidament aquestes es va modificar per poder-la editar i adaptar-la al gust. En canvi, la llàgrima es va dibuixar la meitat amb l'eina llapis i seguidament es va utilitzar el modificador *Torno*, aquest modificador agafa qualsevol traçat i en fa la volta. Un exemple entenedor és si es fa una línia paral·lela a 3cm de l'eix vertical, obtindríem un cilindre de 3cm de radi.
Per les celles i les parpelles també es va utilitzar l'eina llapis però amb el modificador *Extrude*, aquest el que fa es donar profunditat. Un exemple fàcil de comprendre seria el dibuix d'un quadrat que utilitzant el modificador *Extrude* passaria a ser un cub.
Per la tija de sobre del cap es va partir d'un cilindre, també es va modificar per poder-lo editar i un cop es va tenir el diàmetre de la part superior més gran que la inferior es van arrodonir les cantonades. Finalment amb el modificador *Doblar* es va donar la curvatura desitjada.

Aquestes són les principals eines que s'ha utilitzat per modelar tots els elements 3D, no obstant s'ha utilitzat alguna *Boleana* (permet aplicar les operacions d'unió, intersecció o resta entre dos objectes) per fer els forats de les espremedores. També s'ha utilitzat el modificador *Matriz* per reproduir un element les vegades necessàries al voltant d'un element principal.

En tot els modelatges s'ha intentat reduir el nombre de polígons al màxim, d'aquesta manera el prototip estar optimitzat des del principi. Això provoca fluïdesa i una experiència més agradable. En alguns casos la reducció de polígons ha sigut menor, ja que són elements molt corbats i amb la excessiva reducció es perd la qualitat i disseny final.

Elements modelats:

- Personatge -Taronja:
  - Pes: 328 kBytes, Polígons: 5588.
    - Normal.
    - Trist.
    - Eufòria.

Figura 1 **Taronja trista**

Figura 2 **Taronja Feliç**

Figura 3 **Taronja normal**

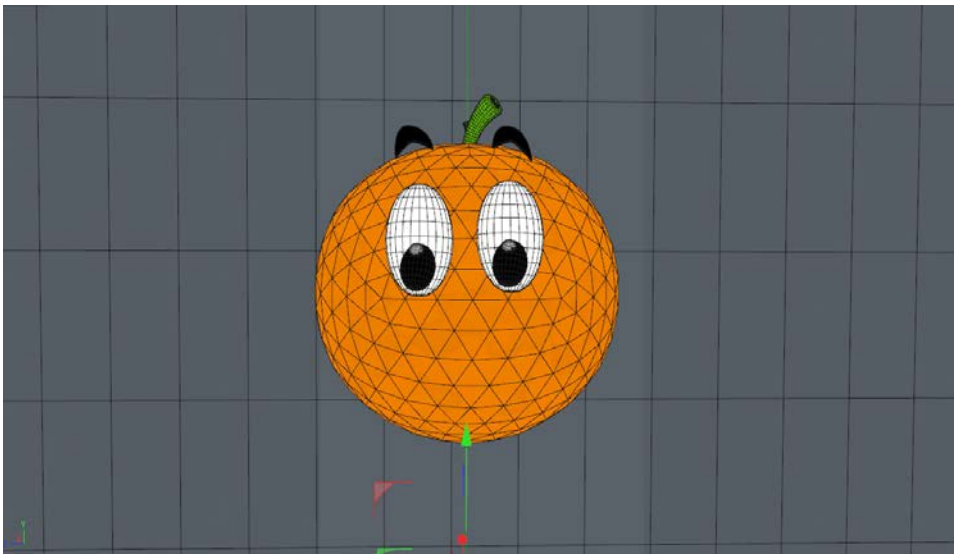


Figura 4 **Malla Taronja**

- Fruites:
  - Raïm.
    - Pes: 906 kBytes, Polígons: 11365.
  - Poma.
    - Pes: 698 kBytes, Polígons: 11836.
  - Cirera.
    - Pes: 603 kBytes, Polígons: 10293.

Figura 5 **Cirera 3D.**

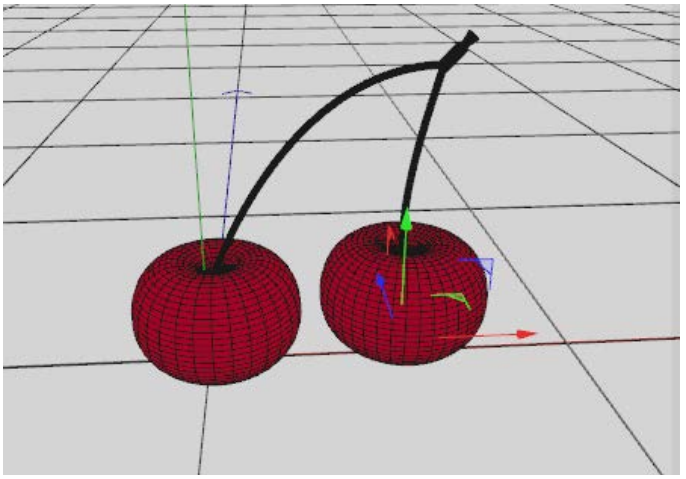


Figura 6 **Malla i modificadors utilitzats en la cirera.**

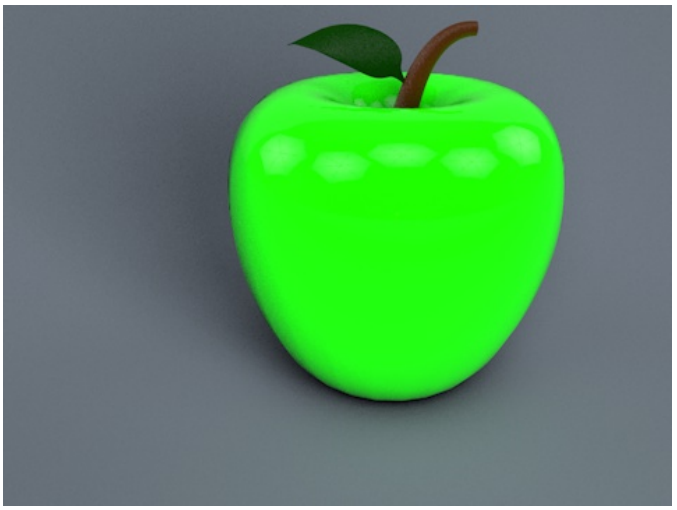


Figura 7 **Poma 3D.**

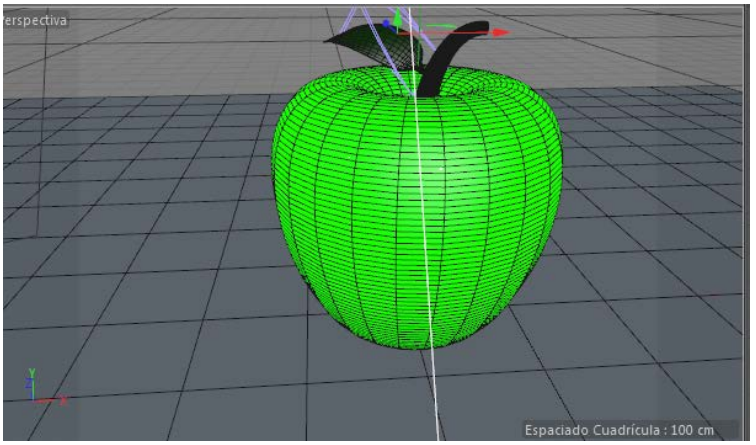


Figura 8 Malla i modificadors utilitzats en la poma.

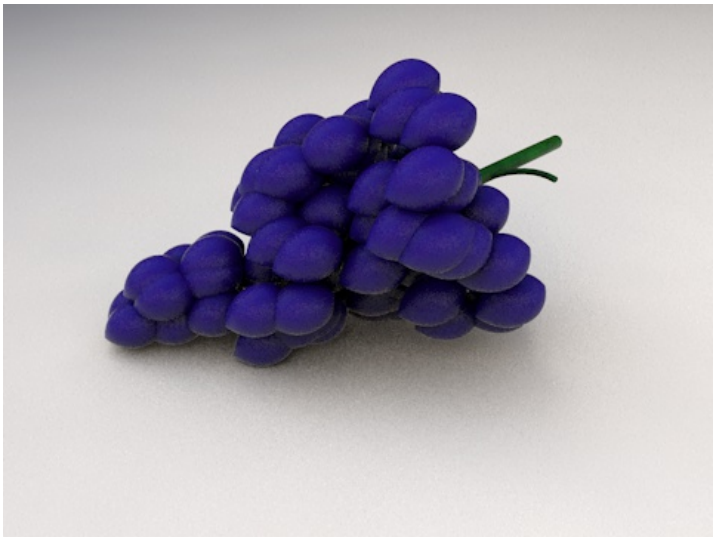


Figura 9 Raïm 3D.

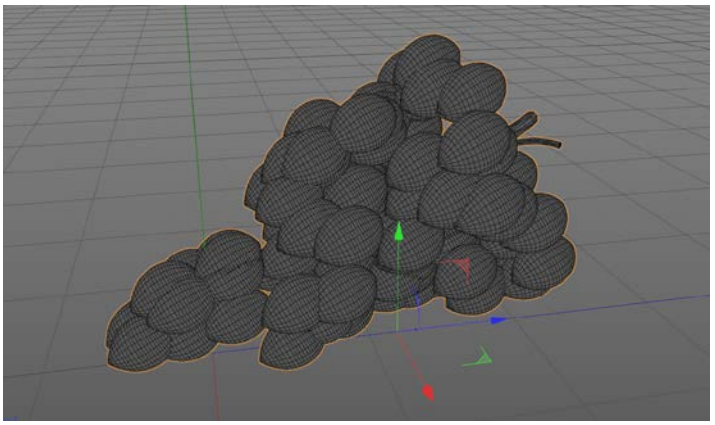


Figura 10 Malla i modificadors utilitzats en el raïm.

- Obstacles:
  - Punxes.
    - Pes: 523 kBytes, Polígons: 7196.
  - Espremedora.
    - Pes: 453 kBytes, Polígons: 7124.
  - Espremedora 2.
    - Pes: 583 kBytes, Polígons: 9676.

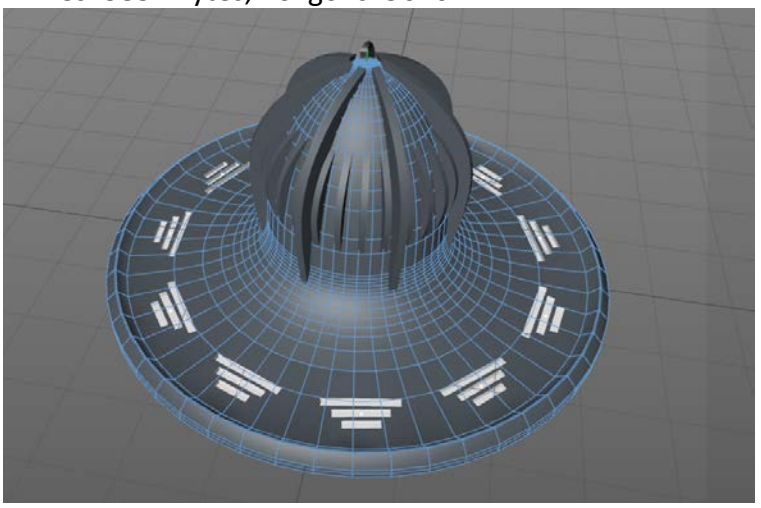


Figura 11 Malla de l'espremedora.

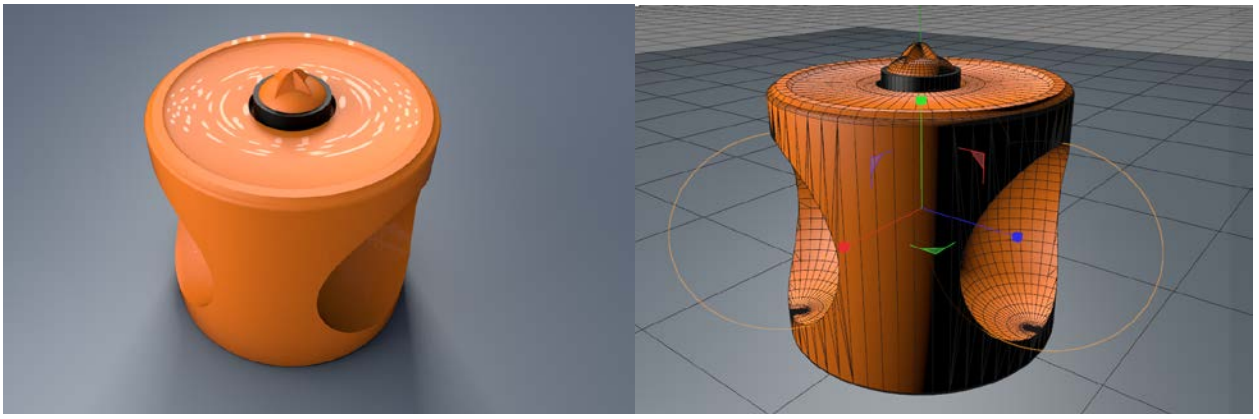


Figura 12 Escena Cinema 4D espremedora 2.

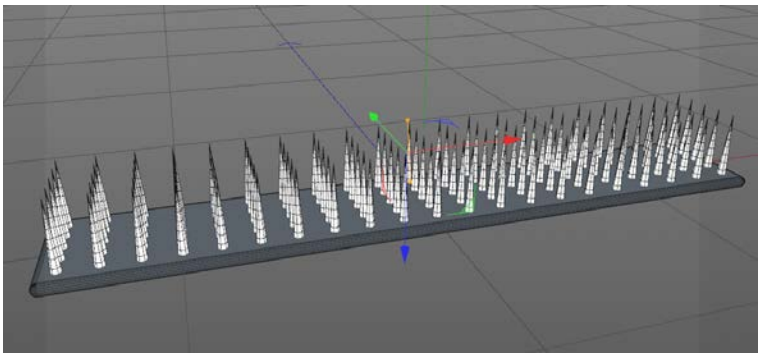


Figura 13 Malla de les punxes.

- Escenaris:
    - Laberint (figura 23).
        - Pes: 244 kBytes, Polígons: 4116.
- Tapa inicial espremedora.
        - Pes: 4227 kBytes, Polígons: 72047.

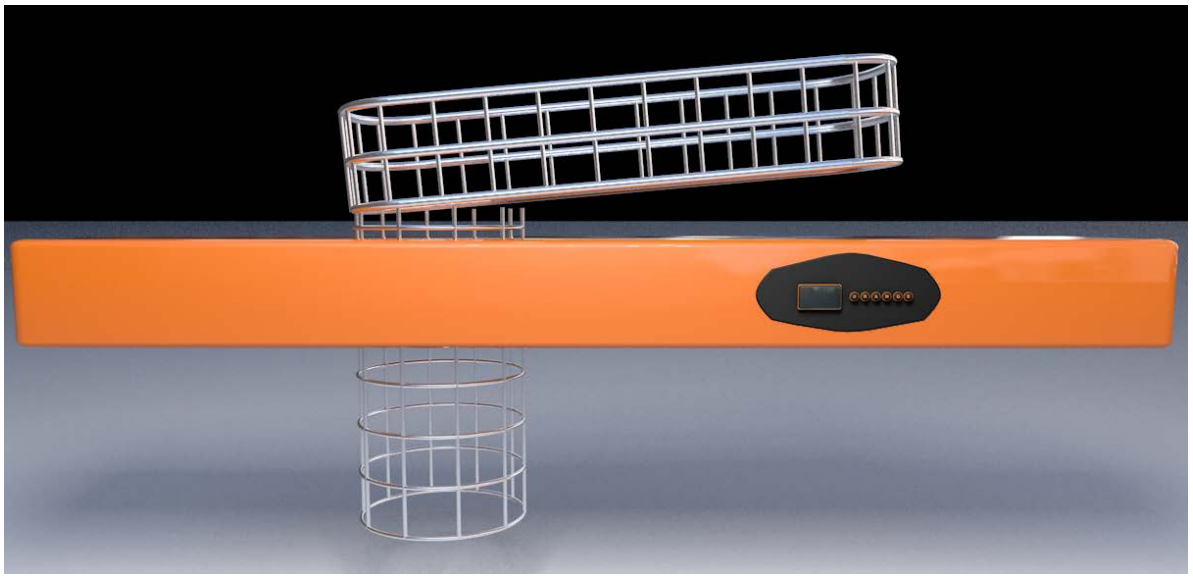


Figura 14 **Tapa inicial espremedora renderitzat.**

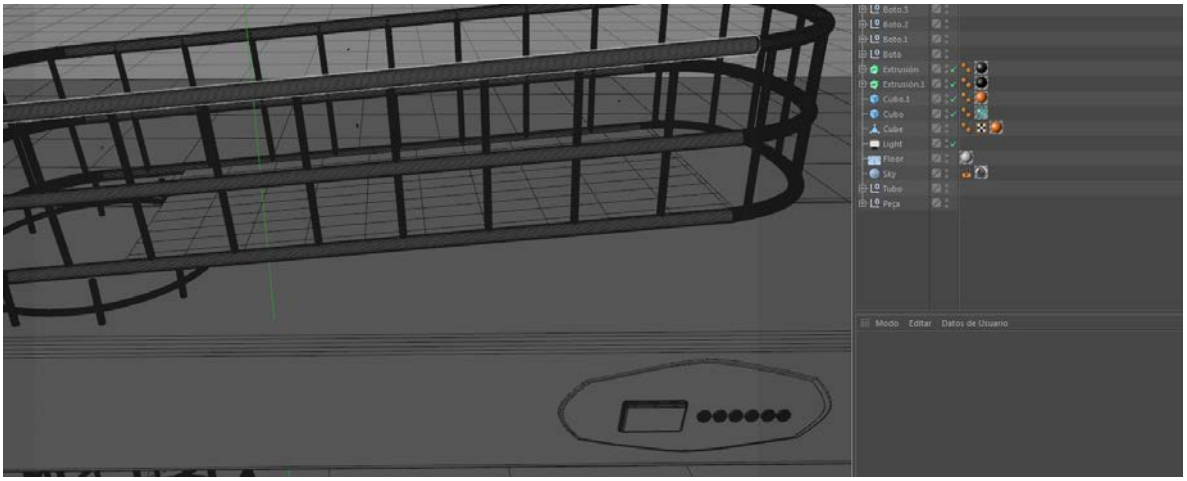


Figura 15 **Malla i modificadors utilitzats en l'espremedora inicial.**

3.4.3 Integració al *Unity3D*

Un cop fets els dissenys, el següent pas és la integració d'aquest al *Unity* i la texturització. Durant aquest procés s'han trobat diferents avantatges i inconvenients

Pel que fa als avantatges, es va veure que des del programari *Cinema 4D*, quan s'exporta en *FBX* (exportació de la peça modelada per a la incorporació a altres programes) hi ha l'opció d'integrar els materials a la peça 3D. Tot i que el resultat no és dels més satisfactoris, va fer el servei i va permetre avançar més ràpid.

També cal dir, que seria més correcte utilitzar un altra forma d'exportació per obtenir un resultat més professional, i que permetria obtenir uns colors més propers als desitjats. No obstant, no era l'objectiu d'aquest treball i, per tant, no es va profunditzar més en aquest tema.

Respecte als inconvenients, hi va haver vàries situacions en les que van sorgir problemes. Algun d'aquests van ser la utilització de diferents escales per als diferents projectes de *Cinema4D*, o la ubicació de la peça 3D i el seu pivot. La direcció de creació de les peces també va ser tot un repte, ja que els *prefabs* utilitzats, feien servir la geometria d'origen.

3.4.4 Creació de animacions

En aquest prototip s'ha utilitzat animacions, els motius d'aquestes s'han explicat en l'apartat 3.5.2. Per realitzar aquestes animacions també s'ha utilitzat el *Cinema 4D*. S'ha fet ús del mètode de *frame* a *frame*. Aquest tracta en modificar el personatge en cada cert *frame*, al final aquest conjunt de modificacions provoquen una animació.

En el cas de l'escena "Felicitats" (és l'escena que surt quan el jugador supera el seu rècord) s'ha utilitzat un modificador anomenat *Emisor*. Com bé diu la paraula és un modificador encarregat de llençar elements. Els elements que llença de baix cap dalt han sigut modelats prèviament per semblar confeti. Així doncs, entre la posició i la direcció que llença els elements modelats, fa la funció de canó de confeti.

Finalment s'ha animat el text per aconseguir una escena més dinàmica i impactant respecte la escena perdedora.

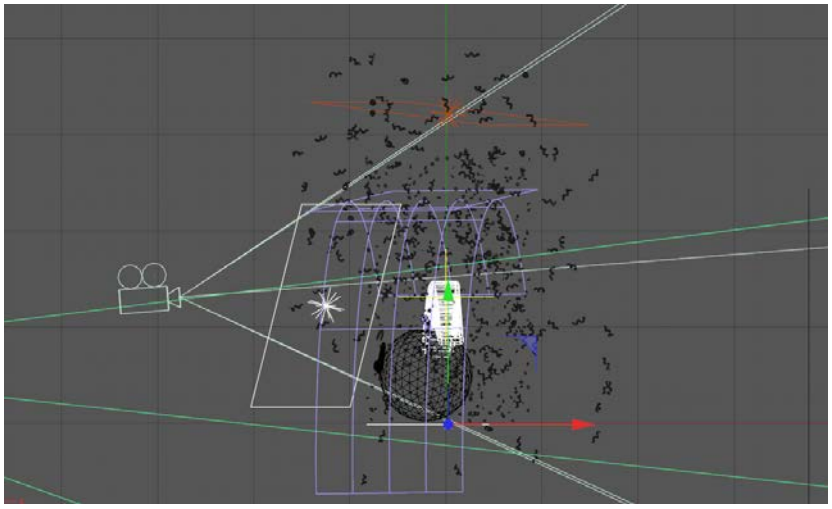


Figura 17 **Malla de la curvatura del text i malla del confeti.**

Figura 18 **Visualització dels modificadors abans del renderitzat.**

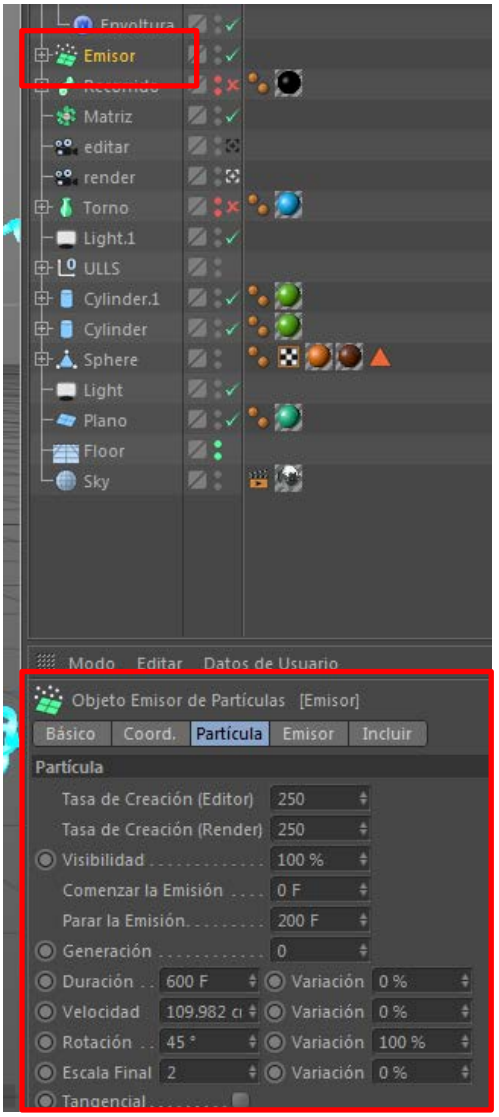


Figura 16 **Modificador *Emisor* utilitzat en el treball amb els seus valors.**

3.5 Disseny 2D

3.5.1 Realització de la *UI*

En aquest prototip només hi ha disseny en 2D en la part de la *UI*, és a dir, els diferents botons, disseny de menús, icones en l'escena de partida, etc.
Aquests dissenys han estat realitzats principalment mitjançant, *Adobe Illustrator*, i en algun cas, *Photoshop*.
Per a l'elecció dels colors, s'ha intentat buscar una tonalitat adequada al tipus de joc, al seu target, i al personatge.
Així doncs, els colors utilitzats han sigut principalment els taronges i un verd-blau saturat. Tant mateix s'ha buscat una homogeneïtat en els diferents menús del joc.

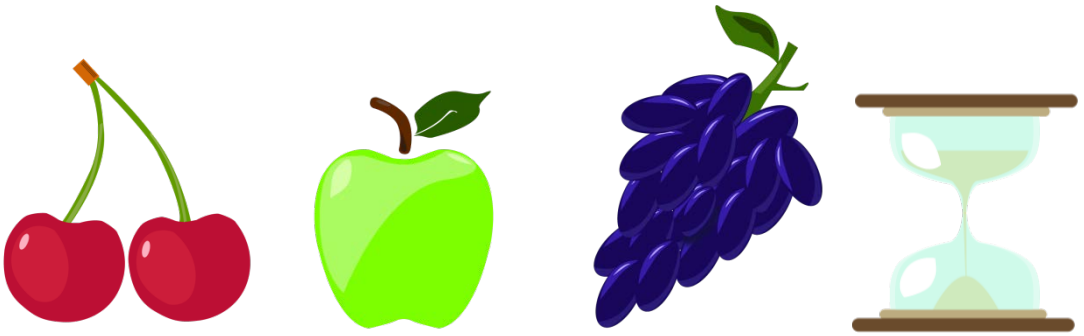


Figura 19 Disseny 2D per la *UI*.

Figura 20 Logo del prototip.

3.5.2 Realització de petites animacions

Dins de l'apartat en 2D també es troben unes petites animacions realitzades i exportades amb *Cinema4D*, i retocades en *Adobe After Effects*. Aquestes petites animacions tenen la funció de donar dinamisme i mobilitat per tal que les escenes no siguin tant avorrides i monòtones.
S'han utilitzat aquestes animacions en el menú d'inicialització i en el menú "Game over", que depenent de la puntuació, podia sortir una animació de la taronja trista o feliç.

Per duu a terme aquestes animacions, després de realitzar els últims retocs en *l'Adobe After Effects* s'ha exportat els arxius en format PNG. Un cop importat al *Unity*, s'han canviat totes les fotos en formats textures a *Sprites* (són objectes gràfics en 2D utilitzades per als personatges, objectes i altres elements del joc). A continuació s'ha agafat el primer *frame* i li hem aplicat una animació amb la resta de *frames*. D'aquesta manera podem reproduir animacions com si fos un video. Cal tenir en compte que la reproducció d'arxius multimèdia en plataformes *Android* funciona diferent, és a dir, el video es reprodueix només a pantalla completa, surten els botons del reproductor, etc.

4. Desenvolupament

4.1 Cerca dels diferents serveis de base de dades

4.1.1 Base de Dades relacional o no relacional.

Aquesta és una de les parts més importants del treball, l'elecció d'una base de dades. Primer de tot hem de saber si escollim una base de dades relacional o no relacional. Després de estar cercant diferent informació, aquí tenim un resum dels avantatges i inconvenients d'aquestes bases de dades:

- Avantatges **Relacional**
  - Està més adaptat i més conegut.
  - Degut al temps que porten al seu mercat, té millor suport i més productes.
  - Han de complir requisits de integritat, tant de dades com en compatibilitat.
- Desavantatges
  - L'atomicitat de les operacions.
  - L'escalabilitat.

- Avantatges **No Relacional**
  - L'escalabilitat.
  - Són més obertes i més flexibles.
  - Es poden fer canvis sense parar la base de dades.
  - Es pot executar en màquines amb pocs recursos.
  - Optimització de consultes per grans quantitats de dades.
- Desavantatges
  - Problemes de compatibilitat amb instruccions *SQL* (Structured Query Language; és un llenguatge declaratiu d'accés a base de dades relacionals que permetre especificar diferents tipus d'operacions en elles).
  - Falta d'estandardització.
  - Suport multiplataforma.
  - Tenen eines d'administració poc usables o s'accedeix per consola.

Així doncs, es recomana la utilització de base de dades **relacionals** pels camps educacionals, de desenvolupament web i empresarial.
Tanmateix, utilitzarem les taules **no relacionals** pels àmbits de *Big Data*, *Clound (XaaS)*, desenvolupament web, xarxes socials i desenvolupament mòbil.

Per tant, la base de dades escollida és una *NoSQL*, és a dir, no relacional. El següent pas és trobar quina s'escull pel treball.

4.1.2 Quina base de dades no relacional s'escull?

En primera instància es va escollir el servei *Parse*, ja que tenia coneixements previs sobre aquesta eina. A més a més, la ràpida importació al *Unity3D* (motor de videojocs multiplataforma creat per *Unity Technologies*) i la gran optimització per fer consultes mitjançant *C#* (llenguatge de programació orientat a objectes i desenvolupat per Microsoft) amb el programari *Unity3D* facilita tot el procés.

El *Parse*, propietat de *Facebook*, va deixar de donar aquest servei, fet pel qual aquesta va deixar de ser una opció factible. No obstant això, es va buscar i valorar diferents alternatives. Una era fer una instal·lació del *Parse* en un servidor extern i utilitzar el seu llenguatge, però sense el seu anàlisis i avantatges que ens aportava. L'altra alternativa era buscar un altre llenguatge.

Finalment, es va utilitzar *MongoDB*. Aquesta és una base de dades que es gestiona a partir de dades semi estructurades, és a dir, documents. Aquests són emmagatzemats en un format estàndard com pot ser amb XML, JSON o BSON. Un exemple seria el següent:

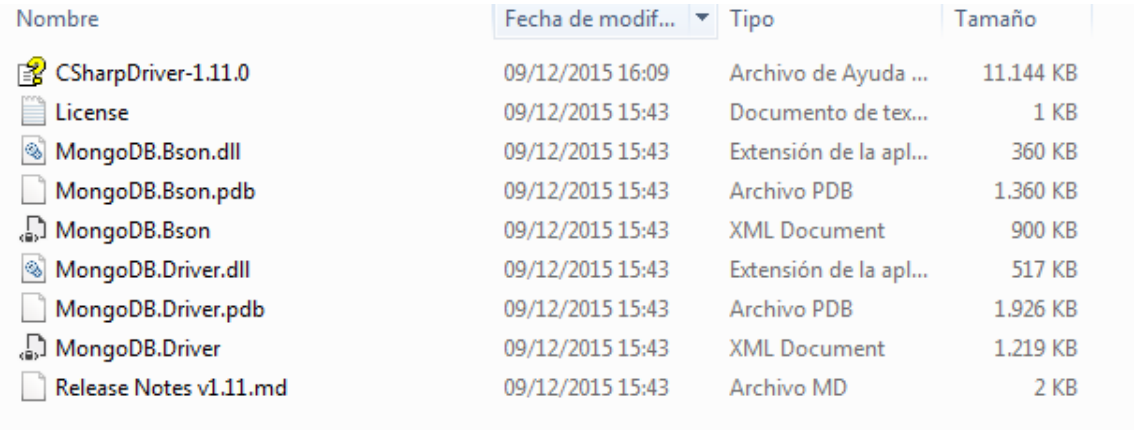
```
{
    "_id": {
        "$oid": "5756fc831a06a90e7064279e"
    },
    "name": "Josep Escribà",
    "adreça": {
        "Nacionalitat": "Åland Islands",
        "Provincia": "Lleida",
        "Poblacio": "Tàrrega"
    },
    "Puntuació": {
        "Puntuació_m": 0,
        "Puntuació_s": 950
    }
}
```

Figura 10 Exemple d'un *script* en *JSON.*

És important senyalar que *MongoDB* probablement és la base de dades NoSQL més famosa del moment, ja que en 2013 va aconseguir un finançament de més de 150 milions de dòlars. *Ebay* i *Foursquare* són algunes de les companyies que utilitzen *MongoDB*.

4.2 Cerca d'eines per la fàcil vinculació

Un cop escollida la base de dades, cal buscar la manera d'importar-ho a *Unity3D* i poder realitzar les consultes des dels scripts (arxiu de ordres) d'aquest últim.
Es va escollir la llibreria *MongoDB C-Sharp Driver* dins de les diferents llibreries que es van provar.



Nombre	Fecha de modif...	Tipo	Tamaño
CSharpDriver-1.11.0	09/12/2015 16:09	Archivo de Ayuda ...	11.144 KB
License	09/12/2015 15:43	Documento de tex...	1 KB
MongoDB.Bson.dll	09/12/2015 15:43	Extensión de la apl...	360 KB
MongoDB.Bson.pdb	09/12/2015 15:43	Archivo PDB	1.360 KB
MongoDB.Bson	09/12/2015 15:43	XML Document	900 KB
MongoDB.Driver.dll	09/12/2015 15:43	Extensión de la apl...	517 KB
MongoDB.Driver.pdb	09/12/2015 15:43	Archivo PDB	1.926 KB
MongoDB.Driver	09/12/2015 15:43	XML Document	1.219 KB
Release Notes v1.11.md	09/12/2015 15:43	Archivo MD	2 KB

Figura 22 **Llibreria *MongoDB C-Sharp Driver.***

L'última versió oficial d'aquesta llibreria no funcionava amb *Unity*. La recerca en multitud de fòrums va evidenciar un problema de compatibilitat entre aquesta llibreria i el *Unity3D*, concretament en l'apartat d'optimització del .NET.

Així doncs, va ser necessària una versió retocada, per ser exactes la 1.11, que funciona correctament en aquesta versió de *Unity*. Aquesta és la més antinga que tenen, ja que actualment van per la versió 2.2.4 creada el 2012. És per aquest motiu que gran part del codi no està optimitzat, provocant una lenta utilització d'aquest. Tot i així ens permet la seva utilització en *Unity3D*.

4.3 Cerca de la teoria per la vinculació i proves.

Per trobar molta de la informació referent a aquesta llibreria s'ha utilitzat la que la pròpia web posa a disposició en la pàgina oficial.

No obstant, un cop programat tot el codi per la creació d'un document i la mostra d'aquest en una escena amb *Unity*, no funcionava. En aquesta ocasió, la solució va ser més fàcil i ràpida. L'errada estava en què només era necessari crear un usuari i una contrasenya dins del nostre servidor per tal d'obtenir els permisos.
Un cop creat el nostre usuari, la connexió i les consultes funcionaven perfectament.

Codi de connexió amb la base de dades:

```
var connectionString = "mongodb://arnauet:47933775gpsa@ds029207.mlab.com:29207/orangejuice";
var client = new MongoClient(connectionString);
var server = client.GetServer();
var db = server.GetDatabase("orangejuice");
User = db.GetCollection<Mongo>("User");
```

Un exemple de consulta amb *MongoDB* des d'un script del *Unity3D*:

```
var rank = User.AsQueryable<Mongo>().OrderByDescending(e => e.Puntuació["Puntuació_s"]).Take(20);
Debug.Log(rank.ToArray().Length);

foreach (var c in rank)
{
    CreateFriend(c.name.ToString(), c.Puntuació["Puntuació_s"].ToString(), c.id.ToString());
}
```

Exemple de consulta per a l'obtenció d'una dada concreta:

```
var query2 = Query.EQ( "_id" , ObjectId.Parse(GameC.OrangeJ.id));
//var print = User.FindOneAs<Mongo>(query);
var print2 = User.FindOneAs<Mongo>(query2);
nom.text = print2.name;
```

Query2 és l'encarregat de buscar per mitjà de la *id* ("_id") un usuari en concret. En aquest cas és la *id* del nostre jugador que hem guardat al nostre *game controller* (script encarregat de guardar informació en el dispositiu un cop tancada l'aplicació o videojoc).

Print2 és l'encarregat de trobar aquesta consulta amb els requisits que s'ha demanat anteriorment a la *query2*.

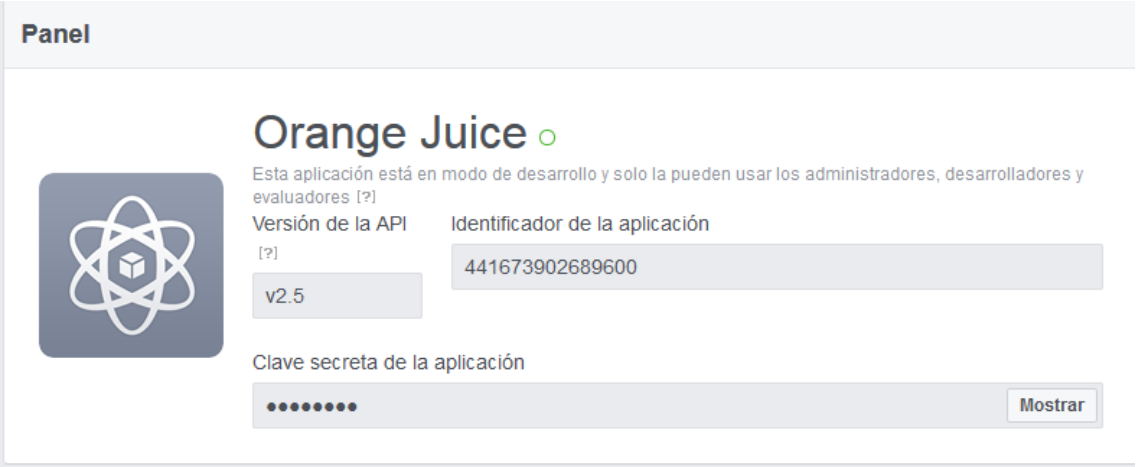
Finalment en un camp de text s'escriu la cerca del *print2* amb un senzill "print2.name". El ".name" és l'opció que es vol escollir d'aquell usuari. Si volguéssim la seva puntuació seria: print2.Puntuació["Puntuació_s"].

Declaració de l'objecte per obtenir les dades:

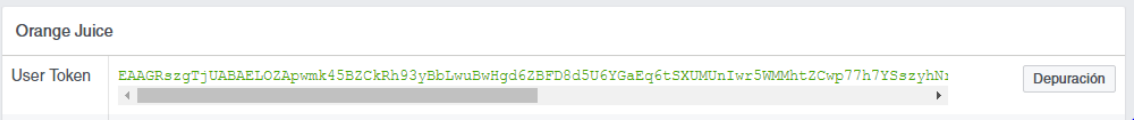
```
public class Mongo
{
    public ObjectId id { get; set; }
    public string name { get; set; }
    public BsonDocument adreça { get; set; }
    public BsonDocument Puntuació { get; set; }
    public BsonArray Pais { get; set; }
}
```

4.4 Connexió del prototip amb Facebook.

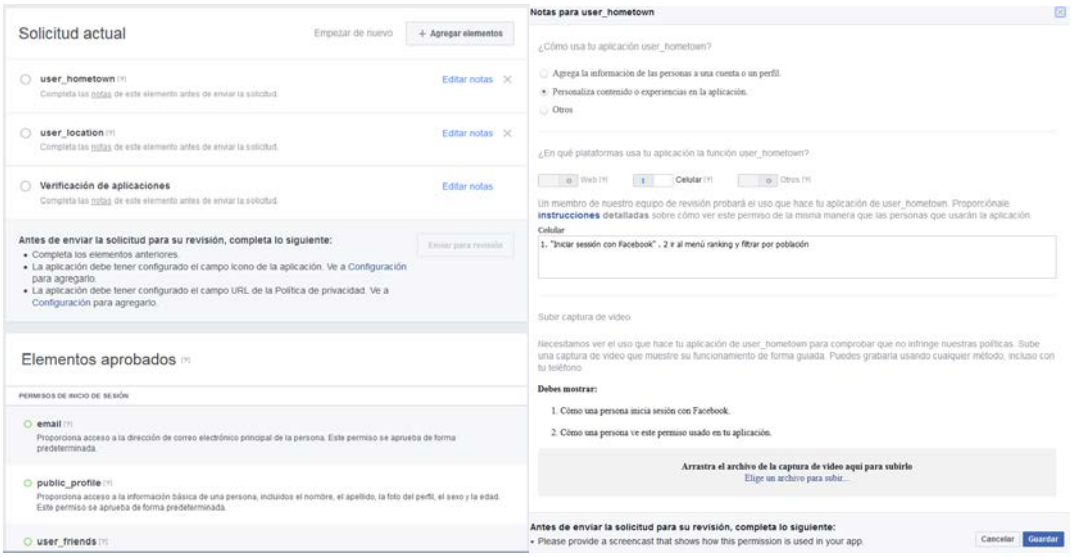
Per aquest apartat cercar la informació va ser molt fàcil ja que *Facebook* posa a la disposició de qualsevol usuari *Apis* (Application Programming Interface, és un conjunt de regles y especificacions que les aplicacions poden seguir per comunicar-se entre elles) per a la incorporació de diferents plataformes, exemples de consultes, etc.
Així que després de seguir els passos d'instal·lació que es mostra en la pàgina web, crear un nom per a l'aplicació i omplir els espais amb buits, es van fer unes proves i el funcionament va ser immediat.



No obstant, té alguns apartats difícils de comprendre. Un exemple és el "token". Es tracta d' una clau que fa la funció de permisos, i que es dóna a l'usuari quan es connecta des de la teva aplicació.



Finalment destacar que *l'api* de *Facebook* actualment està molt restringida. Això vol dir que només ens dóna tres permisos estàndards: l'email, el perfil públic i els amics. L'obtenció de qualsevol altre permís, com podria ser la ciutat, la localització, l'aniversari, etc. Pot arribar a ser molt complicat, i gairebé impossible, en les fases inicials del joc. Això és degut a què requereix omplir un extens formulari. En algunes opcions cal inclús pujar un video mostrant com una persona inicia sessió, com veu els permisos i, fins i tot, en alguns casos volen l'aplicació perquè personal de *Facebook* testegin com pot afectar en l'experiència de l'usuari, i si se'n fa un correcte ús.



4.5 Programació del prototip

El llenguatge escollit per programar en *Unity3D* ha sigut el *C#*. Per a la programació del moviment s'ha utilitzat el acceleròmetre del mòbil.

```
float speed = 500f;
 Vector3 movement = new Vector3(Input.acceleration.x * 3 , 0.0f, Input.acceleration.y);
GetComponent<Rigidbody>().AddForce(movement * speed * Time.deltaTime);
```

S'ha utilitzat el "Time.deltaTime" perquè giri a la mateixa velocitat en tots els dispositius. Aquesta força és aplicada en una esfera interior del personatge, ja que si ho apliquem a la taronja directament, acabava donant salts sol.
També s'ha programat un compte enrere que comença en tres i acaba en zero perquè el jugador es pugui preparar abans de començar. Per aquest motiu s'ha utilitzat la llibreria System.Diagnostics com es mostra a continuació:

Primer de tot es declara la llibreria:

```
using System.Diagnostics;
```

Després es crea una variable que sigui de classe "Stopwatch":

```
public Stopwatch toto;
```

Seguidament, se li diu que és un nou cronòmetre:

```
toto = new Stopwatch();
```

I per últim, s'engega o es para, segons sigui necessari:

```
toto.Start();
toto.Stop();
```

Es va haver de fer d'aquesta manera ja que d'una altra forma el cronòmetre no parava fins que sorties del joc. Aquesta llibreria no permet la utilització del *Debug.Log*. Així doncs, s'ha utilitzat el *print* per mostrar text per la consola.

També s'ha fet un sistema de puntuació que prioritzes la recollida de fruites, ja que la puntuació per cada segon sobreviscut és més baixa que la que obtenim amb les fruites.

Per guardar tota aquesta informació s'ha utilitzat un script anomenat *game controller* (script encarregat de emmagatzemar dades) que es comparteix en totes les escenes del "Unity3D" on es guarda el nom, id, puntuació màxima i última puntuació de l'usuari. Tota aquesta informació també es guardada a la base de dades del MongoDB.

Perquè funcioni en dispositius mòbils s'ha de clica l'opció de permetre que l'aplicació ho pugui guardar en la memòria interna d'aquest.

```
    public void ResetData()
    {

        if (File.Exists(Application.persistentDataPath + "/playerInfo.dat"))
            File.Delete(Application.persistentDataPath + "/playerInfo.dat");
        OrangeJ.Puntuacio = 0;
        OrangeJ.P_màxim = 0;
        OrangeJ.nom = null;
        OrangeJ.id = null;
        OrangeJ.bandera = 0;
    }
}

 [Serializable]
public class GameData {
    //ranking
    public float Puntuacio;
    public float P_màxim;
    public string nom;
    public string id;
    public int bandera = 0;

}
```

Per la creació dels escenaris s'ha utilitzat un script que genera aleatòriament els diferents fons que tenim guardats en un "array". Perquè no sigui tant aleatori, els escenaris fàcils tenen més possibilitats de sortir al principi que en el final. També s'ha utilitzat el mateix sistema per la creació de les fruites.

```
public class generador : MonoBehaviour {

    public GameObject[] Obst;
    public GameObject[] Obstfruita;
    Vector3 RandomUbic;
    Vector3 RandomUbic2;
    public float velocitat;
    public float frequencia;
    public float inici;
    public GameObject player;
    Vector3 posicio;

    // Use this for initialization
    void Start () {
        invo();
    }
    void Update() {
        posicio = player.transform.position;
        gameObject.transform.position = new Vector3(transform.position.x, transform.position.y, posicio.z-15f);
    }
    public void invo() {
        InvokeRepeating("generar", inici, frequencia);
        InvokeRepeating("generarfruita", inici, frequencia);
    }
    // Update is called once per frame
    public void generar () {
        RandomUbic = new Vector3(Random.Range(-4f, 4f),0.0f,transform.position.z) ;
    //  Debug.Log(Random.Range(1f,5f));
        GameObject meteo = Instantiate(Obst[Random.Range(0,3)], RandomUbic, Quaternion.identity) as GameObject;
        //meteo.GetComponent<Rigidbody>().AddForce(velocitat * 2f, 0f, 0f);
    }
    public void generarfruita()
    {
        RandomUbic2 = new Vector3(Random.Range(-4f, 4f), 0, transform.position.z);
        GameObject fruita = Instantiate(Obstfruita[Random.Range(0, 3)], RandomUbic2, Quaternion.identity) as GameObject;
        //meteo.GetComponent<Rigidbody>().AddForce(velocitat * 2f, 0f, 0f);
    }
```

S'ha utilitzat els *preloaders* abans de començar el joc. Aquest són una nova escena que pesa molt poc (consisteix en una imatge i una *progress bar* que calcula el que tarda a carregar l'escena principal).

Per poder realitzar aquesta escena s'ha hagut de fer servir la llibreria *UnityEngine.SceneManagement*.

```
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class Preloader : MonoBehaviour
{
    AsyncOperation async;
    ProgressBarBehaviour BarBehaviour;

    void Start() {
        StartCoroutine(Preload());
    }

    IEnumerator Preload()
    {
        print("Start Load");
        async = SceneManager.LoadSceneAsync(1);

        BarBehaviour = GetComponent<ProgressBarBehaviour>();
        while (!async.isDone)
        {
            async.allowSceneActivation = false;
            print("Loading... " + async.progress);
            BarBehaviour.Value = async.progress*100;
            print("new value: " + BarBehaviour.Value);

            if (async.progress == 0.9f) {
                BarBehaviour.Value = 100;
                async.allowSceneActivation = true;
                /*if (Input.GetKey(KeyCode.Space)) {
                    print("Space");

                }*/
            }
            yield return null;
        }
        print("Scena liaded!");
    }
```

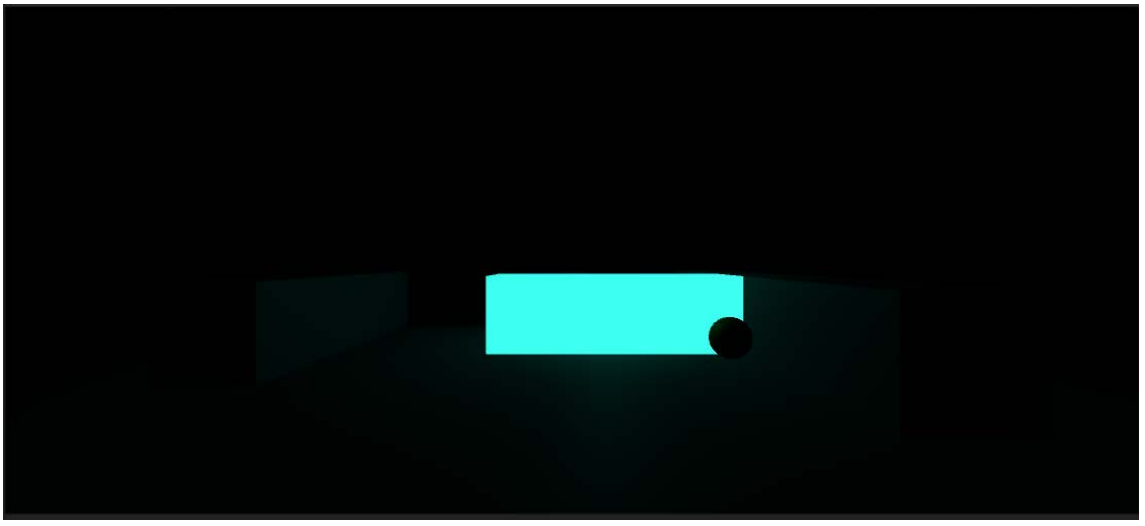
Un altre apartat que va resultar molt difícil va ser la programació de llums dinàmiques en materials de l'escena. Fins fa un temps aquesta opció no estava disponible en *Unity3D*, així que és una part nova d'aquest programari, la qual cosa ha complicat una mica la recercar d'informació sobre aquest tema.

```
public class naaa : MonoBehaviour {
    Renderer Hola;
    public Material dark;
    // Use this for initialization
    void Start () {
        Hola = GetComponent<Renderer> ();
        //Hola.material.shader = Shader.Find("Legacy Shaders/Self-Illumin/Diffuse");
        Hola.material.shader = Shader.Find("Standard");
    }

    // Update is called once per frame
    void Update () {
    }
    void OnCollisionEnter (Collision other){
        if (other.gameObject.tag == "Player") {
            DynamicGI.UpdateMaterials(Hola);
            DynamicGI.SetEmissive(Hola, new Color(0.243f, 1f, 0.945f, 0f) * 0.7f);
            Hola.material.SetColor ("_EmissionColor", new Color32 (62, 255, 241, 0));
            //Hola.sharedMaterial.SetFloat("_EmissionScaleUI", 2F);
            Debug.Log("entro autollum");

            /* Calcular color a color32
            Color color = new Color32(62, 255, 241, 0);
            Debug.Log(color);
            */
        }
    }
    void OnCollisionExit (Collision other){
        if (other.gameObject.tag == "Player") {
            DynamicGI.SetEmissive(Hola, new Color32(0, 0, 0, 0));
            Hola.material.SetColor("_EmissionColor", Color.black);
            Debug.Log("surto autollum");
            //other.GetComponent<Renderer>().material.SetFloat ("_Emission", 1f);
        }
    }

}
```



Malauradament, al tractar-se d'una part nova del programari de *Unity*, encara està en vies de desenvolupament la utilització d'aquest mètode en *Prefabs*, que són elements que tens a la llibreria, però no a l'escena, i els pots cridar a través d'un script. Per aquest motiu, ha sigut necessari utilitzar llums a sobre de cada element i activar-lo i desactivar-lo segons tocava cada paret.

```
ublic class ilulab : MonoBehaviour {
    Renderer Hola;
    Light llum;
    // Use this for initialization
    void Start () {
        Hola = GetComponent<Renderer> ();
        //Hola.material.shader = Shader.Find("Legacy Shaders/Self-Illumin/Diffuse");
        Hola.material.shader = Shader.Find("Standard");
    }

    // Update is called once per frame
    void Update () {

    }
    void OnCollisionEnter (Collision other){
        if (other.gameObject.tag == "Player") {
            Hola.material.SetColor("_EmissionColor", new Color32(62, 255, 241, 0));
            llum = GetComponentInChildren<Light>();
            llum.enabled = true;
            Debug.Log("entro autollum");
            /*encara no funciona com a prefab
              DynamicGI.UpdateMaterials(Hola);
              DynamicGI.UpdateEnvironment();
              DynamicGI.SetEmissive(Hola, new Color(0.243f, 1f, 0.945f, 0f) * 0.7f);
              DynamicGI.UpdateMaterials(Hola);*/
              //Hola.sharedMaterial.SetFloat("_EmissionScaleUI", 2F);


              /* Calcular color a color32
              Color color = new Color32(62, 255, 241, 0);
              Debug.Log(color);
              */
        }
    }
    void OnCollisionExit (Collision other){
        if (other.gameObject.tag == "Player") {
            Hola.material.SetColor("_EmissionColor", Color.black);
            llum.enabled = false;
            Debug.Log("surto autollum");
            //other.GetComponent<Renderer>().material.SetFloat ("_Emission", 1f);
        }
    }
```

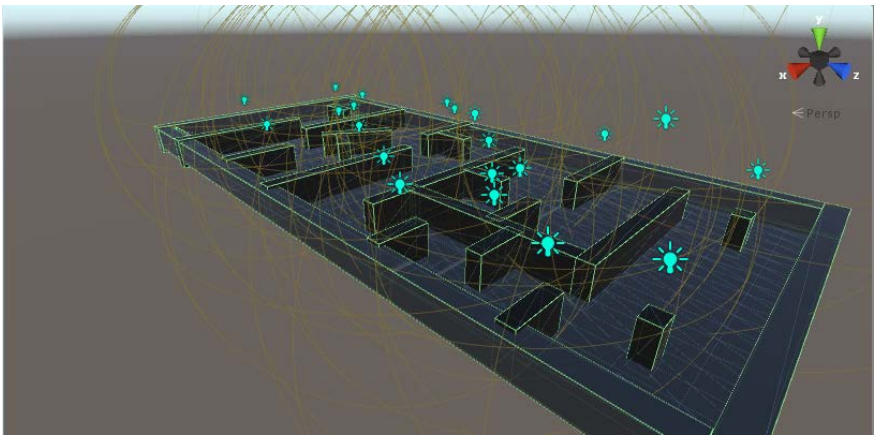


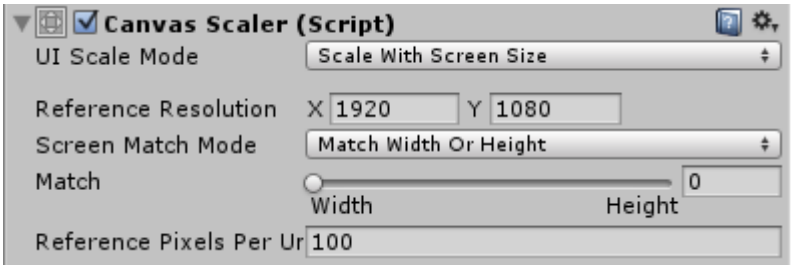
Figura 23 **Malla i il·luminació laberint.**

4.6 Plataforma

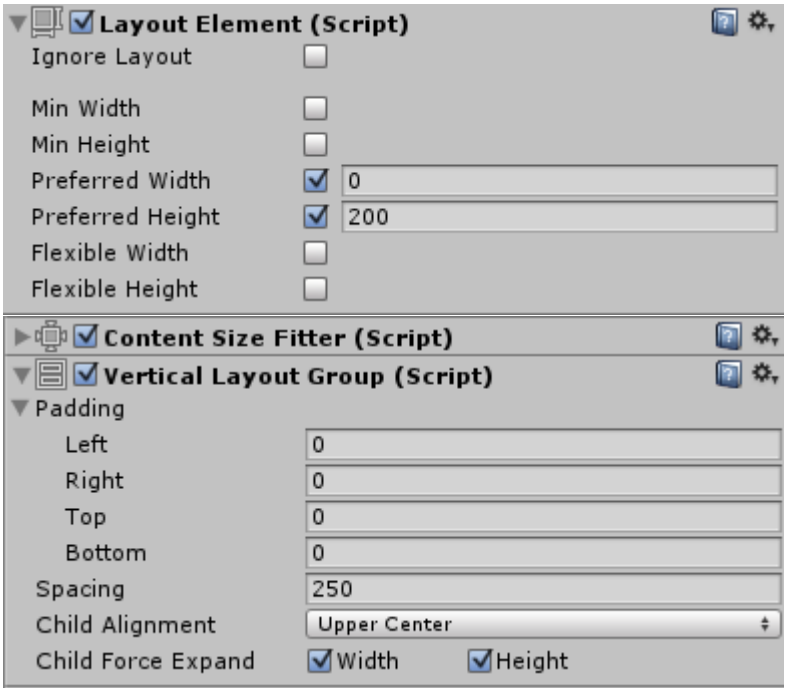
Partint de que la idea inicial era fer el prototip per a dispositius mòbils, el primer que calia saber és quin sistema operatiu es faria servir.
Es va escollir el sistema *Android* simplement per la facilitat que representa disposar d'un aparell amb el mateix sistema operatiu a l'hora de realitzar qualsevol tipus de prova. A més a més, altres punts a favor serien que és més barat, i és més fàcil penjar una aplicació en el mercat de *Android*.

No obstant, l'adaptació del dispositiu mòbil va resultar més problemàtic del que semblava a priori. Després de fer encaixar els diferents elements 2D, els botons, text... en el *canvas* (àrea on tenen que estar tots els elements del disseny d'interfície) del programa va resultar que anava tot al revés i un cop exportada *l'apk* (*Android Application Package,* és un paquet pel sistema operatiu *Android*) i oberta en el mòbil, no havia res que encaixes.

Així doncs la solució d'aquets problemes va consistir en tornar a escalar el *canvas* a una mida de pantalla, en aquest cas la del meu mòbil.

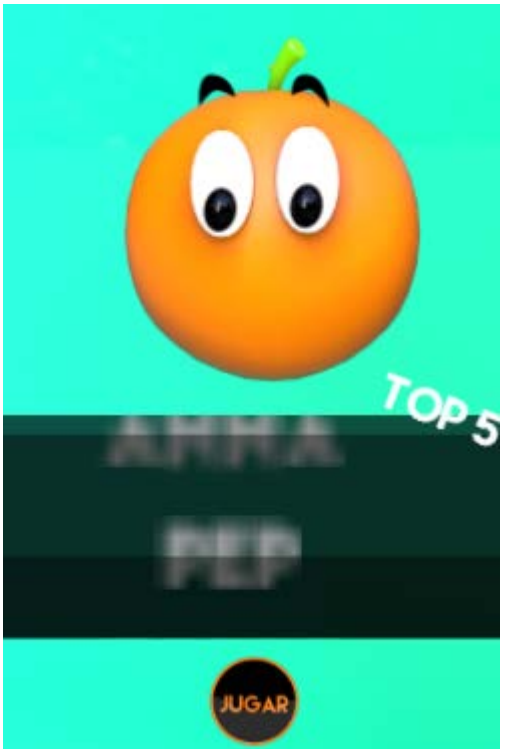


També es van introduir diferents scripts que porta incorporat el programa per forçar el seu aspecte. D'aquesta manera si la pantalla es fa gran, podem prioritzar l'amplada o l'alçada perquè es continuï veient bé.

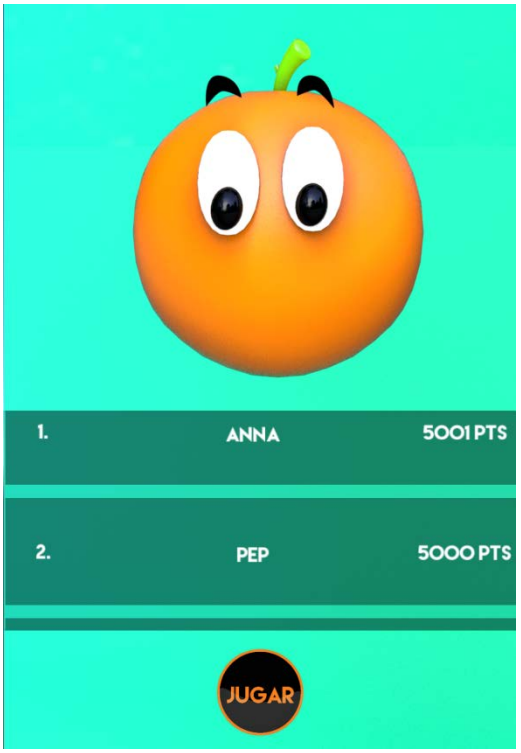


Tot i així, aquest mètode presenta alguns problemes. Un cop comences a adaptar el text, aquest es torna borrós i cada vegada s'ha de comprovar per un dipositiu mòbil. La qual cosa implica que en cada comprovació hagis de crear el joc i fer l'*apk*, passar-ho al mòbil, desinstal·lar l'anterior versió, i instal·lar la nova. A continuació es mostra un exemple:

Visualització des de l'ordenador:



Visualització des del mòbil:



5. Test d'usabilitat

Un cop realitzat un primer prototip en el què ja es podia jugar, es va realitzar un test d'usabilitat que consisteix en fer un sondeig sobre l'experiència viscuda en provar el prototip. En aquest cas el test es va aplicar a diferents joves d'entre 23 i 25 anys. El resultat va ser satisfactori ja que en general els agradava i, fins i tot, els més *killers* no es podien desenganxar fins que no superaven el record posat com a prova.
No obstant, també hi vam trobar punts negatius que han de ser modificats per millorar l'experiència de joc.

En Primer lloc quan arriben al menú principal es queden uns segons pensatius sense saber què fer. Una solució pot ser un "pop-up" explicant els diferents botons i la seva funcionalitat, o ressaltar i millorar la llegibilitat del botó.

Una vegada iniciada la partida hi ha tres segons de preparació del jugador en els que la taronja no es mou. Això va posar nerviosos alguns jugadors ja que esperaven començar a jugar des del primer instant. Caldria valorar si aquest temps de preparació és convenient.

L'escena on es mostra la puntuació ha de millorar en dos aspectes. Un d'aquest és el disseny ja que cal que faciliti la lectura i comprensió del jugador. Però encara més important, cal la incorporació d'un nou botó amb el títol de "Tornar-ho a provar". Durant el test es va poder observar com es feia pesat haver d'anar cada vegada al menú principal. Amb la incorporació d'aquest botó es reduirà el nombre de clics, facilitant el reintent.

6. Difusió

Com es va dir anteriorment, s'utilitzaran les xarxes socials per fer difusió del treball i poder-lo mostrar al públic. Durant el procés de creació ja s'ha fet divulgació en algunes xarxes com pot ser *Instagram* i *Facebook*. D'aquesta manera la gent ja en coneix la seva existència, pregunten pel joc i pel seu procés d'evolució.

És important fer menció que aquest és un projecte que seguirà en desenvolupament, aconseguint d'aquesta manera que tingui més seguidors i creant desig de jugar.

7. Planificació

Per poder realitzar correctament i poder entregar en el temps determinat aquest treball, s'ha utilitzat un diagrama de G*antter*. S'ha utilitzat l'eina gratuïta que incorpora el mateix *Google Drive*. En un primer moment es va planificar per a poder ser entregat durant el mes de juny. No obstant, la desconeixença d'aquest temari i les dificultats trobades durant el procés, ho han alentit. Per aquest motiu s'ha fet una nova programació pel setembre.

Gràcies aquesta programació s'ha pogut controlar el volum de feina en realització, amb el que queda per fer, i valorar si la millor opció és continuar en allò o passar a un altre apartat.

A continuació es mostra la planificació real del projecte:

	Nombre	Duración	Inicio	Fin	Predecesoras	Recursos
1	Realització de planificació	3d	19/02/2016	23/02/2016		
2	**⊞Conceptes Previs**	**22d**	**22/02/2016**	**22/03/2016**		
15	Entrega Rúbrica 1	1d	09/03/2016	09/03/2016		
16	**⊟Concretació final de les diferents parts del joc**	**9d**	**22/03/2016**	**01/04/2016**		
17	Mecàniques	5d	22/03/2016	29/03/2016	11	
18	Dinàmiques	5d	23/03/2016	29/03/2016		
19	Reforçadors	7d	24/03/2016	01/04/2016		
20	Motivadors	3d	29/03/2016	31/03/2016		
21	Creació Diagrama/estructura Base de dades	9d	22/03/2016	01/04/2016		
22	**⊟Disseny**	**111d**	**05/04/2016**	**06/09/2016**		
23	Personatge	4d	05/04/2016	08/04/2016	21FI 1d	
24	Objectes	10d	05/04/2016	18/04/2016		
25	Escenari	10d	05/04/2016	18/04/2016		
26	Menú	5d	31/08/2016	06/09/2016	40	
27	Escena/video de transició	3d	05/04/2016	07/04/2016		
28	Video inicial	1d	31/08/2016	31/08/2016	42	
29	Efectes de so	5d	09/05/2016	13/05/2016		
30	Mùsica	5d	09/05/2016	13/05/2016		
31	**⊟Programació**	**69d**	**17/05/2016**	**19/08/2016**		
32	Vinculació a Facebook	20d	20/06/2016	15/07/2016		
33	Vinculació a una base de Dades	12d	01/06/2016	16/06/2016		
34	Menus	2d	17/06/2016	20/06/2016		
35	Colisions	1d	17/05/2016	17/05/2016		
36	Escenari	14d	18/07/2016	04/08/2016		
37	Dinàmiques	1d	01/06/2016	01/06/2016		
38	Mecàniques	1d	27/06/2016	27/06/2016		
39	UI	10d	08/08/2016	19/08/2016		
40	**⊟Proves**	**7d**	**22/08/2016**	**30/08/2016**		
41	Integració	5d	22/08/2016	26/08/2016		
42	Jugabilitat	2d	29/08/2016	30/08/2016		
43	**⊟Memòria**	**26d**	**08/08/2016**	**12/09/2016**		
44	Redacta la memòria	25d	08/08/2016	09/09/2016		
45	Entrega memòria	1d	12/09/2016	12/09/2016		
46	Presentació	1d	15/02/2016	15/02/2016		

8. Planificació de recursos i pressupost

Per la realització del pressupost s'ha partit de la tècnica de base zero, és a dir, sense partir de cap altre pressupost inicial ni dades orientatives.
El primer pas a seguir, ha sigut el de calcular les diferents despeses i amortitzacions per cada mes, sabent d'aquesta manera quin era el cost real total de cada mes per realitzar una feina. Tot això ho podem observar en les següents taules:

Despeses totals per un prototip de videojoc, realitzat en 300h

Amortització				
	Preu	Anys	Total/Any	Total/mes
Ordenador	1.500 €	5	300 €	25 €
Impresora	100 €	5	20 €	2 €
Mòbil	200 €	5	40 €	3 €
Material oficina	450 €	5	90 €	8 €
				Total amortització
				38 €
Despeses variables	Preu	Hora	Total	Mes
Llum				50 €
Aigua				20 €
Treballador	15 €	300	4.500 €	2.250 €
				Total despeses variables
				2.320 €
Despeses Fix				
Llicencies				50 €
Lloguer				150 €
Internet				50 €
Assegurança				299 €
Quota autonom				267 €
				Total despeses fix
				816 €
				Despeses totals
				3.174 €

Els valors d'Assegurança i Quota d'autònom han estat extretes de *Infoautonoms 2016* i *Txerpa 2016*.

Un cop se sap el cost total de cada mes, es tracta de fer un pressupost que mínimament cobreixi les despeses, així doncs es trobaria el punt mort de l'empresa, i aquesta no tindria ni beneficis ni pèrdues. No obstant si que es podria disposar d'un salari, ja que aquest està contemplat en les despeses totals.

Finalment partint d'aquest valors, s'elabora un pressupost on s'especifica la data realitzada, la data de venciment (que acostuma a ser de trenta dies) i els diferents imports entre d'altres coses. En el meu cas, aquest seria el pressupost:

Arnau Escribà			
Carrer agricultura, 61			
Terrassa, Barcelona			
CP: 08223		Nº de presupost	1
Telf.: 637720570		Data del pressupost	7 de març del 2016
NIF: 47933775-G		Data de venciment	6 de abril del 2016

Pressupost per la realització d'un prototip d'un videojoc

Descripcó	Quantitat	Preu	Import
Maquinària	4	100 €	400 €
Arquitectura IT (servidors, hostings)	1	3 €	3 €
Desenvolupament Android	1	3.000 €	3.000 €
Disseny	1	925 €	925 €
Adaptació a tablet	1	500 €	500 €
Connexió xarxes socials	1	250 €	250 €
			Total
			5.078 €
		IVA 21% de 2.033,00€	1.066,38 €
			TOTAL + %IVA
			6.144 €

Forma de pagament:

1. 30% En la acceptació i firma del contracte
2. 30% Quan es doni l'entrega de la primera Beta d'una versió
3. 40% En l'entrega i valoració del client

9. Conclusions

Després de dissenyar aquest prototip es pot dir que he aprés moltes coses que fins ara tenia ganes de conèixer i no havia tingut la possibilitat de fer-ho. Després de descobrir la superfície del que tenia previst, m'he donat compte que hi ha més coses en les que vull investigar, com pot ser la programació de partides multijugador o la posició d'objectes amb arxius *Json* des d'una base de dades.

Respecte al prototip, he tingut parts molt més difícils del que m'esperava a priori, degut a què és el primer cop que realitzava un prototip d'aquest tipus i per aquesta plataforma. Així doncs, resumiria aquesta experiència de la següent manera:

1. La programació amb *MongoDB* a sigut molt satisfactòria tot i tenir els contratemps més grans en aquest punt.
2. La integració de *Facebook* també a sigut satisfactòria i fàcil, gràcies a la documentació que donen. Tot i que m'ha decebut les restriccions i tot el que demanen per tenir més permisos.
3. No m'ha agradat el fet que *Unity3D* no tingués actualitzat el ".NET" i que encara hagi d'utilitzar la primera versió de *MongoDB.* Juntament amb la no funcionalitat de llums dinàmiques amb *prebafs*.
4. Destacar la dificultat de poder escalar i fer encaixar el *canvas* en els dispositius mòbils. Tot i així la satisfacció de veure correctament el joc en el dispositiu compensa la feina.

Finalment crec que ha sigut un treball de final de grau molt productiu. He gaudit durant tot el procés de creació d'aquest prototip, descobrint que és molt important una bona planificació i ser constant.

1. Agraïments

Vull donar les gràcies a tots els professor que m'han ajudat en aquest treball, he tingut bastants problemes en alguns aspectes i han sigut pacients amb mi.
També donar gràcies al tutor per l'ajuda, els consells i acceptar portar-me el treball, gràcies a ell vaig descobrir el món del *Unity3D*.
Als meus pares que cada dia hem preguntaven com estava i es preocupaven pel treball encara que després de que els hi expliques es quedessin igual.
I per últim a la Núria, que ha estat tots els dies a sobre meu ajudant-me i motivant-me perquè el pogués acabar.

2. Annexos

Diferents Base de dades:
http://www.genbetadev.com/bases-de-datos/bases-de-datos-nosql-elige-la-opcion-que-mejor-se-adapte-a-tus-necesidades
Quan fer servir cada una:
https://estebanz01.wordpress.com/2013/04/04/sql-vs-nosql-cual-es-el-mejor/
Diferencia entre les dos:
http://blog.pandorafms.org/es/nosql-vs-sql-diferencias-y-cuando-elegir-cada-una/
Teoria MongoDB driver c# 1.11:
http://mongodb.github.io/mongo-csharp-driver/1.11/
Informació Facebook:
https://developers.facebook.com/docs/unity
https://developers.facebook.com/docs/unity/reference/current
Instal·lació api Facebook al Unity:
https://developers.facebook.com/docs/unity/gettingstarted
Definició casual games
http://www.elandroidelibre.com/2015/02/por-que-triunfan-los-juegos-casual-en-android.html
https://es.wikipedia.org/wiki/Videojuego_casual
Referències Videojocs casuals:
http://androidzone.org/2015/10/top-10-mejores-juegos-casuales-para-android/
Tipus de videojocs i classificacions:
www.injuve.es/sites/default/files/videojuegosintroduccion1.3.pdf