

DESENVOLUPAMENT D'UN VIDEOJOC

Alumne: Alejandro Álvarez Monllor
Tutor de Recerca: Pere Rius
Curs: 2n Batxillerat A
INS Ernest Lluch
Barcelona, 5 de novembre de 2013

Índex de continguts

Pròleg.....	3
1. Introducció.....	4
1.1 Motivació i hipòtesi del treball	4
1.2 Hipòtesi i objectius del treball.....	4
1.3 Estructura del treball.....	4
1.4 Metodologia per a la confecció del treball.....	5
2. Antecedents.....	6
2.1 Història dels videojocs.....	6
2.2 Tipus de videojocs.....	9
2.3 Estructura d'un videojoc.....	14
2.4 Programari de creació de videojocs.....	15
2.4.1 Blender.....	15
2.5 Programació.....	15
2.5.1 Python: Descripció del llenguatge i conceptes bàsics de programació.....	16
2.5.1.1 Assignació i variables.....	16
2.5.1.2 Expressions booleanes.....	16
2.5.1.3 Estructures condicionals.....	16
2.5.1.4 Mòduls i importació.....	17
2.6 El sistema operatiu Linux.....	18
3. Disseny del videojoc.....	19
3.1 Selecció de la temàtica del videojoc.....	19
3.2 Descripció del fil narratiu del joc.....	19
3.3 Model gràfic.....	19
3.4.1 Estructura d'escenes.....	20
3.4.2 Objectes.....	20
4. Implementació.....	22
4.1 Tecnologia utilitzada.....	22
4.2 Objectes.....	22
4.3 Navegació.....	26
4.4 Interacció.....	27
4.4.1 Interacció al rebedor.....	28
4.4.2 Interacció a la sala dels enigmes.....	29
4.4.2.1 Interacció al primer enigma.....	29
4.4.2.2 Interacció al segon enigma.....	29
4.4.2.3 Interacció al tercer engima.....	29
4.4.2.4 Interacció a la sortida.....	30
4.4.3 Interacció a la sala del laberint.....	30
4.4.4 Interacció a la sala de les plataformes.....	30
4.4.5 Interacció a la sala de la bola.....	31
4.4.5.1 Interaccions del personatge.....	31
4.4.5.2 Interaccions de la bola.....	31
4.4.6 Interacció a la sala final.....	32
4.4.7 Interacció a les “scenes de nota”.....	32
4.5 Puntuació i opertura de la porta final.....	33
5. Resultats.....	35
6. Conclusions i línies de treball futures.....	36
6.1 Conclusions generals.....	36
6.2 Línies de treball futures.....	36
6.3 Opinió personal.....	36
Glossari.....	37
Bibliografia.....	39

ANNEXOS.....	40
Annex 1: Mòduls utilitzats.....	40
1.1 mouselook.py (realitzat per Riyuzakisan).....	40
1.2 enigma1.py.....	40
1.3 enigma2.py.....	41
1.4 enigma3.py.....	42
1.5 proves_superades.py.....	43
1.6 provasuperada.py.....	43
1.7 sala_final.py.....	43
1.8 puntuacio.py.....	43
1.9 fail.py.....	44
1.10 time_ini.py i time_fin.py.....	44
1.11 final.py.....	44
Annex 2: Diari de Recerca.....	47
Fase 1: Tutorials de Blender sobre el modelatge.....	47
Fase 2: Game Engine de Blender.....	52
Pràctica 1: Moviment d'un cub sobre el pla mitjançant el teclat.....	52
Pràctica 2: Creació d'un joc simple: Laberint.....	55
Moviment i física.....	57
Dificultats (fails i temps).....	57
La meta i el canvi de nivell.....	58
Lògica de la càmera.....	59
Fase 3: Programació amb Python.....	61
Fase 4: Plantejament del videojoc (plànols).....	62

Pròleg

Si bé es cert que, en la majoria de casos, la decisió del tema a tractar per al Treball de Recerca és la part més complicada i on la gent se sol deixar més el cap, en el meu cas no ha sigut així. Tota la vida he crescut acompanyat dels videojocs, i la idea de tenir l'oportunitat de fer-ne un no feia més que excitar-me. No vull mentir, però, dient que no m'havia plantejat altres idees, en major part per la por que em feia ficar-me de cap a la piscina. Mai no havia dissenyat més que petits programes, ni havia treballat sobre una interfície dedicada a la pròpia creació de videojocs a part d'en *Game Maker*, una eina de creació de jocs en 2D bastant simple (tot s'ha de dir, també permet la creació de jocs més complexos si en saps). Vaig pensar que si pretenia fer un videojoc no aconseguiria més resultat que joc molt simple i poc engrescador com els projectes que havia fet individualment fins aleshores, em va fer por i vaig deixar córrer el tema. Però, de sobte, mig per sobre, vaig plantejar la idea a Danièle Tost, la mare d'un amic de tota la vida, i em va dir que no era tant difícil com pensava. És més, em va proposar no només de fer un videojoc, sinó de fer-lo en 3D, i aprenent a programar. Veure-la tant convençuda em va fer pensar que realment era possible, així que vaig decidir apostar per, finalment, crear el meu propi videojoc, tal com sempre havia desitjat.

1. Introducció

1.1 Motivació i hipòtesi del treball

Els videojocs són jocs amb computador o amb consoles específiques que proporcionen als usuaris una interfície gràfica a través de la qual aquests han d'aconseguir uns objectius concrets seguint unes regles determinades. Des l'aparició del primer videojoc al 1958 fins a l'actualitat, el sector dels videojocs ha crescut de forma imparable fins a ocupar en l'actualitat més d'un 50% del mercat d'oci, constituent la principal distracció de nens i adolescents.

D'altra banda, en els darrers anys, en el món del programari, ha aparegut el fenomen del programari lliure en el que els programadors publiquen de forma oberta el codi dels seus programes posant-los a disposició d'altres programadors per a que en corregeixin els errors, l'ampliïn o creïn nous programes basats en aquests. El programari lliure contrasta amb el programari propietari, el més estès actualment, en el que els codis són secrets.

La hipòtesi en què es basa aquest treball és que amb la tecnologia actual és possible dissenyar i implementar un joc 3D utilitzant programari lliure sense necessitat de tenir una formació avançada en programació. Per demostrar aquesta hipòtesi en aquest treball he après a dissenyar i implementar un joc que aporto com a prova de concepte.

1.2 Hipòtesi i objectius del treball

Els objectius específics del treball són:

1. La investigació tant de la història com del concepte de videojoc en sí, a més de l'anàlisi dels diferents tipus de jocs que han sorgit amb el temps.
2. La iniciació al món de la programació, l'explicació de la mateixa i de la seva aplicació sobre els jocs.
3. La introducció al món del programari lliure, particularment el sistema operatiu Linux.
4. La iniciació a les interfícies de disseny 3D, concretament Blender 3D i Blender Game Engine, també programari lliure.
5. El desenvolupament d'un videojoc propi com a prova de concepte.

1.3 Estructura del treball

El treball està estructurat en sis capítols. Després d'aquesta introducció, amb la voluntat d'ampliar el coneixement sobre els videojocs abans de crear-ne cap, es presentaran els antecedents. Primer, s'explicarà la història dels videojocs fent èmfasi en les creacions importants que van fer evolucionar aquest món. A continuació, es farà una descripció del concepte de videojoc i dels tipus que hi ha actualment. Finalment, s'explicarà el programari de creació de videojoc i es definirà la programació i la utilitat d'aquesta sobre l'àmbit, a més de fer un petit incís sobre el sistema operatiu Linux. En el tercer capítol s'explicarà el disseny del videojoc creat durant la part pràctica del treball, i al quart la implementació del mateix. En el capítol cinquè es presentarà un ànalisi de resultats amb *beta testers* sobre el

videojoc i, finalment, al sisè i últim capítol s'extrauran les conclusions pertinentes.

El treball consta de dos annexos, el primer conté el codi utilitzat durant el desenvolupament del videojoc i el segon mostra el que he anomenat el meu Diari de Recerca, on es veu la progressió que he realitzat a mesura que he anat aprofundint en els tutorials, i l'explicació dels mateixos. Al final d'aquest Diari de Recerca consten uns plànols del videojoc confeccionats abans de la implementació del mateix. A més, el treball consta d'un glossari amb els termes que poden resultar més difícils d'entendre o fàcils d'oblidar, i una bibliografia amb les fonts que he consultat a l'hora de redactar la part teòrica i els enllaços dels videotutorials utilitzats durant l'aprenentatge.

Finalment, al llarg de la memòria ha estat inevitable emprar anglicismes i tecnicismes que no hauria tingut gaire sentit traduir perquè és la terminologia que s'utilitza a l'àmbit del joc.

1.4 Metodologia per a la confecció del treball

En relació a la part pràctica, com abans de començar no tenia coneixements de com es feia un videojoc, vaig realitzar una sèrie de videotutorials de dificultat progressiva que he utilitzat per a tenir la base per ser capaç de realitzar l'objectiu principal. Aquests tutorials són els que estan presentats en l'annex 2.

Un cop desenvolupat el videojoc, s'ha presentat a un conjunt d'usuaris (*beta testers*) per a que el provessin, detectessin errors i n'analitzessin la usabilitat. A més, s'han recollit tots els temps i puntuacions realitzats per a poder fixar el nivell mig de dificultat.

Per altra banda, la part teòrica del treball la he realitzat mitjançant la recerca i selecció d'informació, a més d'algunes aportacions personals prèviament contrastades en els temes dels que em sentia segur per a opinar degut a la meva experiència personal.

Aquest treball ha sigut realitzat en el sistema operatiu Ubuntu 12.10, fent ús dels programes informàtics LibreOffice Writer (per a la part escrita), Blender (per a la interfície 3D i el desenvolupament global del joc), Gimp (per a l'edició de textures) i Python (com a llenguatge de programació). Tot aquest programari és lliure. Ha calgut instal·lar-lo i aprendre a utilitzar-lo, el que ha constituït una part no negligible, en hores, del temps de dedicació total del treball.

2. Antecedents

2.1 Història dels videojocs

La història dels videojocs pròpiament dita comença a l'any 1958, amb la invenció per part de William Higinbotham del que serà el primer videojoc del món: *Tennis for two*. Aquest joc funcionava en un oscil·loscopi amb dos controladors i, com el seu nom indica, era una partida de tennis vista des de dalt, en la que participaven dos jugadors. Aquesta idea va tenir molt d'èxit, però Higinbotham encara no va donar-se compte de la rendibilitat que es podia treure de l'invent, així que no el va patentar i, un any després, va desmuntar-lo i mai més se'n va saber res. No obstant, la idea de crear un programa d'entreteniment als ordinadors no va sorgir del no res. Per comprendre l'aparició de l'essència dels videojocs, cal remuntar-nos a la Segona Guerra Mundial. Alan Turing, un famós matemàtic britànic (creador del Test de Turing) treballava junt amb Claude Shannon, un estatunidenc expert en computació, per descobrir els codis que feien servir els submarins alemanys U-boot. Es varen donar compte, aleshores, de la importància de crear programes que simulessin el procés de raonament humà, el que més endavant s'anomenarà Intel·ligència Artificial (IA). Així, varen començar a fer diverses especulacions sobre el tema, i al 1949 Shannon va presentar un projecte anomenat *Programming a computer for playing chess* a una convenció de Nova York. En el seu projecte, Shannon va crear un munt de codis que es segueixen utilitzant actualment en els videojocs d'escacs. Paral·lelament, Turing també havia creat el seu propi programa d'escacs, finalitzat al 1948, però era massa potent i no va poder fer-lo servir fins al 1952, en que va provar-lo simulant els moviments de la computadora, i va aconseguir guanyar una partida. Aquest és el verdader naixement del concepte de videojoc, però aleshores encara no se li veia una utilitat real a la idea, ja que s'associava la utilització d'ordinadors a projectes científics i, d'alguna manera, més "seriosos".

A la dècada dels 50, tot i no veure's encara la utilitat real que tenien els videojocs, es van fer moltes especulacions i projectes sobre el tema. Al 1951, John Bennett va presentar en una fira britànica un computador gegant anomenat *Nimrod*, una màquina capaç de jugar a un joc d'estratègia matemàtica anomenat *Nim*. Tot i que va ser un èxit, també es va desmuntar per fer servir les peces per a projectes més importants. Més endavant, al 1952, Alexander Douglas, un alumne de la Universitat de Cambridge, va crear en un ESDAC (la primera calculadora electrònica del món) una reproducció del tres en ralla com a tesis doctoral. Però, com els projectes anteriors, no va ser comercialitzat.

Així doncs, en aquella època el concepte de videojoc era encara molt difús, però ja es podia apreciar l'aparició de les primeres idees que tocaven el tema. La situació deixava als ciutadans de l'època amb la possibilitat de jugar als escacs contra un ordinador, de jugar a tennis entre dos amics, al *Nim*... però, realment, els creadors d'aquests programes ho veien més com un projecte sense fonament, una cosa per passar l'estona, sense adonar-se'n del mercat que s'originaria arrel d'això. Una prova és la quantitat de còpies que s'han creat després de l'original *Tennis for two* de Higinbotham, o del codi del programa que permetia jugar a escacs de Turing i Shannon. Aquestes idees no varen ser explotades en la seva època, però van permetre el desenvolupament posterior de la idea per part de Wayne Witaenem, Martin Graetz i Steve Russell, els creadors de *Spacewar!*, el primer videojoc que triomfaria realment, tot i no ser patentat un altre cop. *Spacewar!* va ser creat per a PDP-1, el primer ordinador de la sèrie PDP i el factor més important en la creació de la cultura *hacker*, com a

projecte personal dels tres joves mencionats abans, quan el PDP-1 va arribar al *Massachusetts' Institute of Technology*, centre del qual eren alumnes. L'aparició d'aquest projecte al 1962 va tenir un èxit escandalós al MIT, i nombroses còpies varen ser distribuïdes per ARPAnet (l'antic internet). Tot i aquest èxit, els tres joves van decidir no patentar el seu projecte, i tampoc es van plantejar la seva comercialització, ja el hardware necessari per jugar a *Spaceware!* costava 120.000\$. De tota manera tampoc es va perdre la idea, ja que ha sigut un dels videojocs més plagiats en tota la història.

Més endavant, però, apareix la *Brown Box* de Ralph Baer, un dispositiu que permetia jugar a diversos videojocs tan sols connectant-lo a la televisió. Entre els jocs disponibles a la *Brown Box* es trobava, per exemple, el *Tennis for Two*, en una versió més moderna. Ralph Baer, qui treballava per a Sanders Associates, va presentar el seu projecte a Herbert Campman, director d'investigació i desenvolupament de l'empresa, qui li va oferir 2000 dòlars i 5 mesos per acabar el projecte. Encara que no era massa, això oficialitzava el seu treball.

A la dècada dels 70, es comença a veure que una nova força econòmica és a punt d'aparèixer. Un munt d'empreses emprenedors neixen i inverteixen en les coneudes màquines recreatives, que aconseguirien un munt d'èxit. Neix la coneguda empresa Atari, en un principi anomenada Syzygy Engineering (canvià de nom per problemes de copyright) que comercialitza *Spacewar!* en forma de recreativa, sota el nom de *Computer Space*. En aquest moment no rep gaire reconeixement, perquè el sistema de joc era molt complicat per a públic no universitari, i només aconsegueixen 250 dòlars de benefici. Dos anys després, però, Nolan Bushnell, creador d'Atari, al 1972 assisteix de públic a una demostració de la recentment elaborada *Magnavox Odyssey*, la primera videoconsola del món, desenvolupada per l'inventor de la *Brown Box*, Ralph Baer. Allà descobreix el *Ping-Pong*, una versió millorada del *Tennis for Two* que havia inclòs Ralph Baer a la seva videoconsola nova. Bushnell agafarà la idea per crear una nova recreativa, *Pong*. Aquesta nova màquina d'Atari aconsegueix, per primer cop a la història dels videojocs, un èxit escandalós: només a Estats Units es van vendre prop de 100.000 unitats, generant un benefici de més de 250 milions de dòlars anualment. En aquest moment apareix definitivament la indústria del videojoc.

Cal remarcar, realment, la importància d'Atari, no tan sols en la indústria (que la va fer néixer), sinó també en la història dels videojocs. No només varen tenir la idea principal i la varen dur a terme, sinó que ho van fer de forma impecable, creant cada cop nous títols de més qualitat. Cadascun d'aquests títols es podria considerar fàcilment com a pare d'un tipus de videojoc, doncs mentre les companyies de la competència es dedicaven a treure noves còpies de *Pong*, Atari no feia més que aportar noves idees al món, amb el que va dominar per complet el mercat durant tota la dècada dels 70. Una anècdota curiosa és el mètode de treball que utilitzava Bushnell, completament liberal. Als inicis d'Atari, quan es comercialitzava *Pong*, no donava abast amb la demanda que tenia, i va anar a l'oficina d'atur per contractar a moltíssima gent sense feina, molts dels quals eren addictes a l'heroïna i al haixix. Lluny d'importar-li la condició de drogoaddictes dels seus treballadors, Bushnell va prohibir qualsevol tipus de discriminació cap a aquests dins de les seves oficines, va difondre un missatge de germanor entre els seus treballadors i, deixant de banda l'ordre jeràrquic de l'empresa, no va establir un horari fixe de treball i va donar absoluta llibertat envers la vestimenta i l'imatge dels seus treballadors. Cal recordar que era l'època Hippie! Tot i l'estranya imatge que podria donar una empresa així, no va trigar gaire en obtindre beneficis milionaris.

Un cop apareguda la indústria dels videojocs, no és d'estranyar que un munt d'empreses sorgissin del no-res amb noves idees tant aplicades als videojocs com al món de la informàtica en sí. A la dècada dels 80, es van fer millors increïbles en quan a velocitat de processament de dades, el que permetia crear projectes més ambiciosos. Van aparèixer definitivament els microxips, unitats de processament de mida molt més petita, el que feia guanyar espai. A més, amb l'aparició de les pantalles *raster*, es comença a treballar en l'entorn gràfic. S'ha de tenir en compte que, en origen, els dispositius de visualització eren vectorials, només es dibuixaven línies, no àrees. Les pantalles *raster* permetien per primer cop transformar imatges descrites en un format vectorial en un conjunt de píxels, el que va fer que millorés molt el realisme. Des d'aleshores, els gràfics han sigut un altre punt important a l'hora de desenvolupar un videojoc. Amb totes les millores que van sorgir en aquesta època, no és d'estranyar que s'assigni l'edat d'or dels videojocs de 1978 a 1983. Les primeres empreses, com Atari, tenien uns ingressos descomunals.

A partir d'aquest moment, la indústria dels videojocs ha seguit un progrés semblant a totes les èpoques. Per una banda, els enginyers han treballat per a millorar la velocitat de processament amb noves CPUs, per crear nous dispositius d'entrada i sortida millors... mentre que els dissenyadors de videojocs han aprofitat les millores per implementar noves idees. Cal remarcar, però, l'aparició de Nintendo en el món dels videojocs. Nintendo no serà només una de les companyies que ha portat a la llum un gran nombre de jocs considerats joies, sinó que, a més, és l'única companyia que serà capaç de desbancar al mercat estatunidenc, en poder d'Atari. Hiroshi Yamauchi, nét del creador de la companyia, va determinar dos canvis molt importants per a la història dels videojocs. En primer lloc, va dissenyar la *Game & Watch*, primera consola portàtil de la història, que va tenir un èxit descomunal venent 30 milions d'unitats en 11 anys. Això no va ser prou per Yamauchi, que va decidir inspirar-se en la consola d'Atari, la *Atari VCS*, i va crear-ne una molt superior: la *Famicom*, més coneguda com a *Nintendo Entertainment System* o *NES*. La gran demanda de títols per aquesta consola va portar a Yamauchi a tenir una idea molt innovadora per l'època: permetia a tercers desenvolupar videojocs per a Nintendo i, si els considerava dins d'un cànon de qualitat, els publicava deixant clar qui l'havia fet, però quedant-se una part dels beneficis. Això va ser vist com una gran oportunitat per tots els petits programadors, que podien explotar les seves idees. Per a Nintendo va suposar un creixement exponencial del seu catàleg de videojocs, amb clàssics com *Super Mario Bros.*, *Dragon Quest*, *Final Fantasy* o *The Legend of Zelda*. A més, gràcies a aquesta idea, varen aparèixer les empreses només dedicades a la confecció de videojocs, com SquareSoft (avui coneguda com Square Enix).

Més empreses dissenyadores tant de videoconsoles com de videojocs varen aparèixer a l'escena, com era obvi pensar. Les dues potències noves a destacar són Sega i Sony, la primera destinada a ser desbancada per la segona amb el llançament de la *PlayStation 2*. La potència dels processadors continuava creixent exponencialment i, a la dècada dels 90, es va començar a generalitzar l'ús de la xarxa per a fer jocs en línia. Més endavant, a principis del nou segle, varen aparèixer les GPUs, que són CPUs gràfiques programables. Això va comportar una millora impressionant del realisme.

Avui dia, dintre de l'àmbit dels videojocs, hem de comptar amb nous factors, com són els SmartPhones, que han donat corda als dissenyadors per a que pensin en nous projectes per a aquesta plataforma. La guerra de consoles queda repartida entre Sony, Nintendo i Microsoft, sent aquesta última la més recent incorporació. Els fabricants de hardware també lluiten, en

part, per crear el més potent, i això no fa més que ajudar al desenvolupament d'aquesta indústria.

2.2 Tipus de videojocs

A mesura que ha anat passant el temps, han aparegut diversos tipus de videojocs. Tot i que últimament el mercat del videojoc està creixent molt i estan apareixent nous tipus de videojocs, podem classificar els gèneres clàssics en la següent taula:

GÈNERE	CARACT. PRINCIPALS	EXEMPLES
<p>Aventura Distincions entre... <u>Aventura d'acció (A)</u> Videojoc d'aventura amb combats contra la IA. <u>Aventura gràfica (G)</u> Videojoc d'aventura centrat en la resolució d'enigmes, trencaclosques... Sense combat. <u>Aventura conversacional (C)</u> Videojoc d'aventura sense suport gràfic, que es corre en un terminal que reacciona amb l'escriptura, amb l'única ajuda d'un text programat que t'indica on estàs. Aquest tipus d'aventura es sol confondre amb els videojocs de Rol pel seu semblant amb D&D (Dungeons and Dragons, un famós joc de rol de taula).</p>	<p>Els jocs d'aventura es caracteritzen per plantejar una història al jugador en la que ha de prendre part interactuant amb diversos personatges. La definició real d'aquest gènere és bastant ambigua ja que s'ha fet servir per englobar molts altres, aquesta és la raó per la qual aquest gènere és en el que es classifiquen a més jocs.</p>	The Legend of Zelda (A) Uncharted (A) Tomb Raider (A) Prince of Persia (A) Assassin's Creed (A) The Professor Layton (G) Hotel Dusk: Room 215 (G) Phoenix Wright (G) Zork (C) Colossal Cave Adventure (C) Enchanter (C)

<p>Acció</p> <p>Distincions entre...</p> <p><u>Aventura d'acció (A)</u></p> <p>Videojoc d'aventura amb combats contra la IA.</p> <p><u>Hack 'N' Slash (HNS)</u></p> <p>Videojoc de pura acció (també anomenat arcade perquè eren típics a les màquines recreatives) que es centra únicament en el combat, amb combinacions d'atacs molt espectaculars i evolucions de personatges a nivell de lluita (millores d'armes, vida, poder...)</p>	<p>Els videojocs d'acció busquen despertar l'entusiasme del jugador amb multitud de combats, batalles espectaculars i gran quantitat d'atacs i combinacions. Està molt lligat amb el gènere d'aventures, ja que és complicat fer un joc de pura acció sense una trama que enganxi, tot i que s'ha fet més d'un cop amb bons resultats.</p>	<p>Legend of Zelda (A) Uncharted (A) Tomb Raider (A) Prince of Persia (A) Devil May Cry (HNS) God of War (HNS) Dante's Inferno (HNS)</p>
<p>Curses</p>	<p>Els videojocs de curses comporten una part important dels campionats de videojocs. Fiquen al jugador en el paper de pilot d'un vehicle (pot variar, ja que de vegades la cursa es dona entre personatges. Ex: Pokémon Rush) i li donen l'objectiu de guanyar curses. Aquest tipus de joc tenen la seva màxima força en el mode multijugador, ja que permet que grups d'amics competeixin entre sí, o inclús a nivell més professional en tornejos.</p>	<p>Need For Speed F-Zero Mario Kart Gran Turismo Moto GP Crash Team Racing Pokémon Rush</p>
<p>Educatius</p>	<p>Els videojocs educatius tenen com a finalitat ensenyar coneixements escolars, com poden ser llengües, matemàtiques, dibuix... Normalment es combinen amb el desenvolupament de factors motrius com els reflexos i l'agilitat mental.</p>	<p>Brain Training English Academy Alto los desastres</p>
<p>Esports</p>	<p>Videojocs destinats tant a la simulació d'activitats esportives com a la recreació de partits de l'esport sobre el que tracti el joc (com pot ser el futbol per FIFA)</p>	<p>FIFA NBA WiiFit Mario Tennis Mario Smash Football</p>

<p>Estrategia</p> <p>Distincions entre...</p> <p>Estrategia a temps real (ETR)</p> <p>Els successos es desenvolupen, com el seu nom indica, a temps real.</p> <p>Estrategia per torns (EPT)</p> <p>Els combats, ja siguin PvE (Player versus Environment) o PvP (Player versus Player) es desenvolupen per torns.</p>	<p>Un gènere que força al jugador a pensar tàctiques per aconseguir els seus objectius. Casi sempre tenen vista d'àguila i situen al jugador en el lloc de capità d'un exèrcit (en el cas de que el videojoc sigui de guerra estratègica). Si no, controla el seu jugador en un camp que es sol determinar amb caselles.</p>	<p>Warcraft (ETR) League of Legends (ETR) Starcraft (ETR) Imperivm (ETR) Age of Empires (ETR) Dofus (EPT) (també RPG) Worms (EPT) Advance Wars (EPT))</p>
<p>Lluita</p> <p>Distincions entre...</p> <p>Duel System (1v1)</p> <p>Videojocs en els que el combat és a duel 1 contra 1.</p> <p>Free for All (XvX)</p> <p>Videojocs en els que el combat és lliure i poden jugar tantes persones com controladors hi hagi a la consola (normalment 4). Tot i això, els tornejos de jocs de lluita Free for All es solen tractar com a 1v1 i juguen només dues persones (ex: Super Smash Bros.)</p>	<p>Tipus de videojocs que es centren en els combats entre dues persones, clàssics en les màquines d'arcade. Són famosos per tenir una gran quantitat de <i>combos</i> que només la elit d'aquests jocs sap dominar per posar-se per davant dels demés. Aquesta característica fa que sigui un tipus de joc extremadament competitiu que, junt amb els de carreres i esports, el transforma en un clàssic als tornejos de videojocs.</p>	<p>Mortal Kombat (1v1) Tekken (1v1) Street Fighters (1v1) Capcom VS Marvel (1v1) DC Universe VS MK (1v1) Persona Arena (1v1) Super Smash Bros. (XvX) All Stars Battle Royale(XvX)</p>
<p>Musicals</p> <p>Distincions entre...</p> <p>Jocs de ball (B)</p> <p>Jocs d'instruments (I)</p> <p>Jocs de cant (C)</p>	<p>Els videojocs musicals giren entorn de la música, i el seu objectiu és fer sentir al jugador que és capaç de, per exemple, tocar bé la guitarra, ballar bé... Aquests videojocs poques vegades ajuden a incrementar l'habilitat personal en els àmbits que tracten i solen ser només per pura diversió, tot i que hi ha gent que s'ho arriba a prendre seriosament, sobretot en jocs de cant de l'estil SingStar, on s'avalua al jugador després de cada cançó.</p>	<p>Dance Dance Revolution (B) Super Mario Dancing (B) Guitar Hero (I) DJ Hero (I) Rock Band (I) SingStar (C) Karaoke Revolution (C)</p>

<p>Plataformes</p> <p>Distincions entre...</p> <p><u>Plataformes en 1a P (1P)</u></p> <p><u>Plataformes en 3a P (3P)</u></p> <p><u>2D Scrollers (2DS)</u></p> <p>L'aventura es desenvolupa en un mapa en 2D que es va tirant cap endavant a mesura que avances.</p>	<p>Els jocs de plataformes són un tipus de joc en el que no es para de saltar mai. Es caracteritzen per precisament això, hi ha multitud d'obstacles que s'han de superar mitjançant salts, volant, i altres accions. Sempre són combinats amb altres gèneres, per tant les plataformes s'han de considerar més com una característica que com un gènere en si. Tot i així, hi ha jocs pioners en incloure les plataformes que mereixen una especial menció en el seu apartat.</p>	<p>Turok (1P) Crash Bandicoot (3P) Jak & Daxter (3P) Ratchet & Clank (3P) Super Mario 64 (3P) Spyro (3P) Super Mario (2DS) Sonic the Hedgehog (2DS) Megaman (2DS)</p>
<p>Role Play Games</p> <p>Distincions entre...</p> <p><u>RPG d'acció (RPGA)</u></p> <p>RPGs combinats amb acció</p> <p><u>RPG tàctic (RPGT)</u></p> <p>RPGs combinats amb estratègia.</p> <p><u>RPG shooter (RPGS)</u></p> <p>RPGs combinats amb shooter</p> <p><u>Massive Multiplayer Online</u></p> <p><u>Rol Play Games (MMORPG)</u></p> <p>RPGs online, on hi juguen gran quantitat de jugadors que poden lluitar entre ells, ajudar-se... tot depèn del joc.</p>	<p>Els Role Play Games (videojocs de rol) es caracteritzen bàsicament per intentar crear un vincle entre el jugador i el personatge que porta. Tot i ser moltes vegades confosos amb els videojocs d'aventura, s'ha de tenir en compte que els videojocs de rol busquen una immersió extra per part del jugador: en un RPG, el jugador ha de ser el personatge. Això ho aconsegueixen amb recursos tals com poder canviar el nom del personatge (no tots), la personalització total al gust, canvi d'armes i armadura que tenen indicat els punts de dany/defensa com al Rol clàssic de taula (per daus o amb uns números intel·ligibles per part del jugador) En definitiva, un RPG és l'aventura portada a l'extrem. Hi ha moltíssimes interaccions amb NPC (non-player characters), recerca d'informació, personalització, etc.</p>	<p>Kingdom Hearts (RPGA) Fable (RPGA) Demon's Souls (RPGA) Dark Souls (RPGA) The Elder Scrolls (RPGA) Monster Hunter (RPGA) World of Warcraft (MMORPG-A) Tera Raising (MMORPG-A) Pokémon (RPGT) Final Fantasy (RPGT) Fire Emblem (RPGT) Dofus (MMORPG-T) Metroid (RPGS) Fallout (RPGS) Mass Efect (RPGS)</p>

<p>Shooter</p> <p>Distincions entre...</p> <p><u>First-Person Shooter (FPS)</u></p> <p><u>Third-Person Shooter (TPS)</u></p> <p>“Marcianitos”</p>	<p>Els videojocs shooter són un clàssic des de l'inici dels videojocs, amb l'aparició d'Space Invaders (més conegut com “els marcianitos”). Encara que la concepció de shooter actual no té res a veure, és una primera visió del que seria un dels gèneres més explotats de la història del videojoc. En el shooter, el jugador s'obre camí a base de disparar sense parar i matar enemics. En el cas dels primers shooters, la història no estava gens treballada, ja que el que interessava era la facilitat amb la que el jugador s'enganxava a disparar sense parar.</p>	<p>Quake (FPS) Duke Nukem (FPS) Metroid (FPS) Halo (FPS) Call of Duty (FPS) Doom (FPS) Counter-Strike (FPS) Dead Island (TPS) Uncharted (TPS) MDK (TPS) Space Invaders (Marcianitos)</p>
<p>Trencaclosques</p> <p>Distincions entre...</p> <p><u>Agilitat mental (AM)</u></p> <p><u>Trencaclosques (Puzzles, P)</u></p>	<p>Els videojocs trencaclosques estan pensats per estimular el cervell del jugador. Aquest és un gènere que moltes vegades s'ha combinat en certs moments amb videojocs d'aventures o de rol per sortir de la trama mata-mata i crear reptes que requereixin enginy per part del jugador, cosa que dona una varietat molt bona al videojoc. Tot i que al principi els videojocs trencaclosques es tancaven en els models d'agilitat mental com Tetris o Bejeweled, s'ha de dir que companyies modernes com Valve han fet un gran treball en reviure aquest gènere en formes completament innovadores i molt bones, com és l'exemple de Portal, un “FPS” en el que no mates a ningú, només tens una pistola que crea portals i la fas servir durant tot el joc per resoldre trencaclosques.</p>	<p>Tetris (AM) Bejeweled (AM) Buscaminas (P) Portal (P)</p>

2.3 Estructura d'un videojoc

Els videojocs són programes orientats a esdeveniments, és a dir que estan sempre oberts, esperant esdeveniments, entrades per part del jugador o canvis en les puntuacions i els temps. El tipus d'interacció del jugador és el que determina el gènere del videojoc. L'objectiu lògic del jugador serà doncs, mitjançant les seves interaccions, tancar el programa amb una victòria.

L'estructura d'espera de senyals per part del joc és el que s'anomena *game loop* i es pot esquematitzar com segueix:

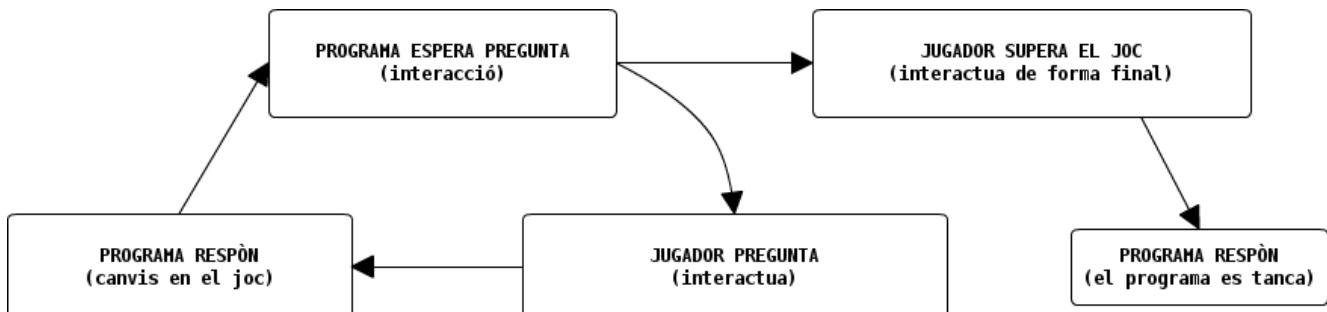


Figura 1: Estructura general del joc

Així doncs, l'objectiu del jugador serà fer una petició (interacció) al programa la resposta a la qual sigui acabar amb victòria. Aquesta és la concepció bàsica d'un videojoc, i s'ha d'entendre que aquestes parts s'han anat adornant amb el temps, per exemple, quan el jugador arriba al final d'un joc, aquest no es tanca sense més, sinó que abans se'l passa a una pantalla amb els crèdits, un vídeo final, etcètera.

Vist des d'un punt de vista de programació, el *game loop* és:

while(user doesn't exit)	mentre(usuari no tanca)
check for user input	buscar ordres de l'usuari
run AI	activar l'IA
update scenario	actualitzar escena
resolve collisions	resoldre col·lisions
draw graphics	dibuixar els gràfics
play sounds	reproduir els sons
end while	acabar mentre

Figura 2: Estructura interna del game-loop

A cada bucle, es detecten les interaccions d'usuari, s'apliquen els mètodes d'intel·ligència artificial per a realitzar les accions desencadenades per la interacció, s'actualitza l'escena, és a dir que es mouen els personatges, es canvien les textures i colors, s'apliquen els nous quadres de les animacions, etc. A continuació, es detecten les possibles col·lisions que s'hagin produït com a conseqüència del desplaçament dels personatges. Per a evitar-les, de vegades es fan rebotar els personatges, i per tant es torna a modificar la seva posició. Aleshores ja es pot fer el procés de visualització de l'escena, també anomenat *rendering*. A la vegada s'apliquen els sons. Tots aquest procés es fa molt ràpid, i és transparent per a

l'usuari, que té en tot moment la sensació de continuïtat temporal.

2.4 Programari de creació de videojocs

En el cor dels videojocs hi ha una tecnologia bàsica que és comuna a tots: visualització (*rendering*), detecció de col·lisions, gestió i producció de so. Per això, les empreses creadores de jocs tendeixen a crear plataformes de programari en les que tots aquests elements són comuns i d'un joc a l'altre es re-aprofiten modificant només els models gràfics i la lògica dels jocs. Darrerament, s'han publicat de forma comercial o lliure diverses plataformes de creació de videojocs que ofereixen aquesta base de programació. Els sistemes més coneguts són Unity (2D /3D), Blender (3D), Game Maker (2D, plataforma) entre altres. El nucli bàsic dels jocs s'anomena generalment *motor de joc* o *Game Engine*.

2.4.1 Blender

Com he dit abans, en el desenvolupament del meu videojoc utilitzaré Blender com a programari gràfic. Aquest programa és utilitzat majoritàriament per crear models tridimensionals (modelatge geomètric) i fer animacions 3D, i té moltíssimes possibilitats en aquest àmbit. En aquest treball no es busca aprofundir en el modelatge i les animacions en Blender, però tot i així sí que he hagut d'aprendre a modelar superfícies polièdriques, per a la realització de l'entorn gràfic del videojoc.

A més de la part d'animació, Blender presenta un motor de joc (Game Engine) intern bastant ampli que ofereix un sistema de detecció de col·lisions, un simulador de lleis de la física i que disposa, a més, d'una interfície gràfica per a definir relacions lògiques entre objectes i la seva funcionalitat.

Per a programar un joc amb Blender, cal realitzar les etapes següents:

- Crear els objectes amb el modelador.
- Definir quines lleis físiques cal aplicar a cada objecte i les propietats físiques corresponents, per exemple, la gravetat del món i la massa dels objectes.
- Definir el tipus de col·lisió a detectar per a cada objecte a través de la interfície gràfica.
- Fixar el tipus d'esdeveniment que el sistema ha d'escoltar en relació a l'objecte i quina actuació cal desencadenar.
- Programar la lògica principal del joc: puntuació, inici i finalització.

La programació en Blender es realitza en el llenguatge *Python*.

2.5 Programació

Podem entendre per programació el procés de dissenyar, codificar i depurar el codi font de programes computacionals. Per fer això, és necessari el coneixement d'un llenguatge de programació. Hi ha diversos llenguatges de programació (C++, Python, QuickBasic...) i, tot i que tots presenten característiques similars, a la vegada tenen variacions entre ells. A l'hora

de programar, hem d'escollir prèviament el llenguatge en el que ho farem, i aprendre a utilitzar-lo ja que, com en una llengua parlada, els llenguatges de programació tenen una sintaxi i unes normes que s'han de complir a la perfecció (si no es fa, el resultat del programa pot ser un completament diferent a l'esperat, o simplement pot no funcionar).

Per què s'utilitza la programació als videojocs?

Si bé es cert que els programes de creació de jocs presenten una interfície des de la que es poden assignar ordres lògiques als objectes, arriba un moment en el que, si aprofundim en el desenvolupament del nostre videojoc, ens adonem que ens quedem curts, i necessitem comunicar les ordres que volem que es realitzin directament a l'ordinador. Per fer això, s'utilitzen els anomenats *scripts* o mòduls. Els *scripts* són codi dissenyat en un llenguatge de programació que contenen les ordres que es realitzaran en el nostre videojoc en aquest moment crític en el que l'única manera de fer entendre les ordres a l'ordinador sigui mitjançant la programació.

Elecció del llenguatge de programació

L'elecció del llenguatge de programació, en el cas d'aquest treball, és limitada, ja que Blender només suporta scripts de Python. Per tant, estudiare i definiré aquest llenguatge de programació, i totes les expressions de programació que apareguin estaran escrites en Python.

2.5.1 Python: Descripció del llenguatge i conceptes bàsics de programació

A l'annex 2, a la *Fase 3: Programació amb Python* s'inclou una descripció més detallada dels elements de Python que han estat necessaris per la programació del videojoc. Per altra banda, el referent als conceptes bàsics de programació en aquest llenguatge i la seva sintaxi es troba redactat en aquest apartat de la memòria.

2.5.1.1 Assignació i variables

Les variables són contenidors de dades que el programador defineix en el seu script. Per exemple, si el programador escriu “`a = 1`”, “`a`” serà una variable que contindrà un 1. Per tant, cada cop que treballem amb “`a`” estarem treballant amb 1. Això es diu assignació. Hi ha quatre tipus de variables en Python:

- `int`: contenen un nombre enter (ex: `a = 1`)
- `float`: contenen un nombre decimal (ex: `a = 2.1`)
- `string`: contenen cadenes de text. Hem de tenir en compte que Python detecta cada lletra com una unitat independent i, per tant, si definim la variable “`a`” com “`hola`”, per Python és una cadena de text formada per “`h`”, “`o`”, “`l`” i “`a`”. Aquest tractament del text permet que, si per exemple fem “`a = ho`” i “`b = la`”, si preguntem `a+b` el programa ens retorna “`holà`”.
- `boolean`: contenen expressions booleanes (ex: `a = True`)

2.5.1.2 Expressions booleanes

Les expressions booleanes són aquelles que es refereixen a una propietat lògica d'un objecte. Per exemple, “`a == False`”, “`a > 5`”, etcètera. Amb aquestes expressions es poden construir les anomenades estructures condicionals, que són vitals a l'hora de la programació.

2.5.1.3 Estructures condicionals

Les estructures condicionals són cadenes lògiques construïdes amb expressions booleans que, com el seu nom indica, treballen segons les condicions que el programador imposa en el codi. El programa detecta si les condicions s'han complert o no i, en funció d'això, es poden donar dues reaccions diferents, o tantes com condicions tingui el programa. Dins de les estructures condicionals, les tres expressions de condició són:

- **if**: S'activa si l'expressió booleana condicional es compleix.
- **else**: S'activa si no es compleix cap **if**.
- **else if (elif)**: S'activa si no es compleix cap **if**, però a diferència de l'**else** també imposa una condició. És la forma combinada de les altres dues.

Exemple 1:

```
if a < 5:  
    print("a és més petit que 5")  
else:  
    print("a és més gran o igual que 5")
```

La instrucció `print` usada en aquest exemple serveix per a que aparegui quelcom en pantalla.

Exemple 2:

```
if a < 5:  
    print("a és més petit que 5")  
elif a > 5:  
    print("a és més gran que 5")
```

2.5.1.4 Mòduls i importació

Com ja s'ha mencionat anteriorment, un mòdul és un fitxer de text que conté diverses sentències de codi. En scripts petits, un arxiu sol ésser suficient per expressar les ordres que volem, però per a ordres més complexes, s'utilitzen diversos mòduls per referir-se a ordres concretes i, un cop fetes, s'agrupen tots en l'anomenat mòdul principal que, mitjançant l'eina de la importació, obtindrà la informació necessària de cada petit mòdul sense necessitat d'haver-ho d'agrupar tot en un de sol. Això ens permet organitzar-nos millor a l'hora de programar, utilitzar mòduls prèviament creats en un mòdul propi sense haver-lo de copiar tot, etcètera.

Les tres ordres amb relació a la importació de mòduls són:

- **import**: importa un mòdul secundari al mòdul principal en la seva totalitat.
- **from**: importa la part seleccionada d'un mòdul secundari al principal.
- **imp.reload**: permet recarregar un mòdul sense reiniciar Python.

Exemple d'importació:

Imaginem que estem construint un mòdul, i volem definir una funció llarga dins del nostre mòdul. No sabem molt bé com construir-la, així que busquem per Internet el mòdul ja creat que ens donarà aquella funció. Ens el descarreguem, i el desem amb el nom de “funció.py”.

A l' hora de definir la funció, utilitzarem l'ordre *import* perquè el programa busqui el mòdul secundari on la tenim ja definida. Això seria:

```
import funció
```

Per altra banda, imaginem que no necessitem la totalitat del mòdul secundari. Acabem de recordar que, fa un temps, havíem creat un mòdul en el que ja havíem definit aquella funció, a part de moltes altres. Recordem que li havíem donat el nom de *X*, i el mòdul s'anomenava *programa.py*. Ara, volem que, dins del nostre mòdul, el principal, s'accedeixi a *programa.py*, es busqui *X* i s'importi, però ignorant la resta del mòdul secundari. Per fer això, utilitzaríem:

```
from programa import X
```

2.6 El sistema operatiu Linux

Tot i que Blender té una interfície per a *Windows*, funciona també sota *Linux*, un sistema operatiu lliure. Com un dels objectius del treball és, precisament, explorar el món del programari lliure, el joc s'ha desenvolupat en aquest sistema operatiu, motiu pel qual se li dedica un apartat explicatiu en aquest treball.

Per entendre la importància de Linux dintre del camp de la informàtica, cal saber que és el programari lliure. Normalment, els programes tenen el seu codi de programació encriptat, de manera que, en principi, ningú l'hauria de poder obtenir. El programari lliure té el codi obert, amb el que qualsevol persona hi pot fer aportacions. Linux, al ser de codi obert, està en constant creixement. Això és, sens dubte, positiu per al desenvolupament informàtic, però el creador del sistema no en treu benefici, doncs la distribució és completament gratuïta. Davant d'aquesta situació, la majoria dels programadors prefereixen mirar pel seu propi benefici en comptes de pel global, encriptant el codi, patentant-lo, i cobrant per la seva distribució.

El sistema Linux té diverses distribucions, amb diferents graus de complexitat. Centrant-nos en Ubuntu, la distribució utilitzada durant la realització d'aquest treball, podem dir que presenta una interfície gràfica prou intel·ligible, suporta arquitectures de hardware de 32 bits i 64 bits. Com a distribució de Linux, és de codi obert i tots els usuaris poden fer les seves aportacions a la pàgina web oficial: <http://community.ubuntu.com/>

3. Disseny del videojoc

3.1 Selecció de la temàtica del videojoc

Com s'ha pogut observar a la part teòrica, hi ha molts tipus de videojocs, i l'elecció d'un sí que m'ha sigut més complicada que la selecció del tema del treball. Tampoc he pogut aspirar a fer qualsevol tipus, tenint en compte que no he après a crear, per exemple, intel·ligència artificial, ja que és molt més complicat i escapa a les meves possibilitats actualment. Així que per començar, ja tinc jocs com per exemple, els de rol, descartats. En inici vaig pensar en fer un joc de trencaclosques, amb resolució d'enigmes. Ho veia fàcil de programar i vaig idear el joc entorn aquesta idea. Fent els tutorials de Blender, però, vaig fer-ne un que em va fer agafar una altra idea (*annex 2*). El tutorial consistia en crear un laberint sense parets del que no podies caure, tot i que en inici pensava que m'anava a explicar com fer un laberint clàssic, el de buscar la sortida, amb parets. Realitzar amb èxit el tutorial em va fer pensar que podia programar una prova així i, alhora, un laberint clàssic. En aquest moment vaig decidir donar-li un altre punt de vista al videojoc, decidint que consistiria en la resolució de diverses proves, diversos jocs en un de sol. Vaigaprofitar el tutorial del laberint per crear “la sala de la bola”, on el que s'ha de moure sobre el laberint ja no és el personatge, sinó una bola empentada pel personatge. A la vegada, vaig dissenyar “la sala del laberint”, que consisteix en superar un laberint clàssic, amb parets, de buscar la sortida. Junt amb aquestes dues sales tenia la idea dels enigmes. Amb tres sales creades, sentia que encara podia exprimir més el joc, i se'm va acudir la idea de fer una sala de plataformes. Això donaria riquesa al treball, demostrant l'assimilació de diversos conceptes i ensenyant l'aplicació d'aquests sobre la confecció del videojoc propi. Per assignar el videojoc a una categoria de les esmentades amb anterioritat als antecedents, el classificaria com una barreja entre trencaclosques, concretant en orientació espacial en alguns casos, i plataformes. El sistema de navegació pel personatge principal és en primera persona, i l'entorn 3D.

3.2 Descripció del fil narratiu del joc

En el videojoc, un personatge (el jugador) ha perdut la seva memòria. Desperta en una sala estranya, amb quatre proves a superar per tal d'obtenir “la llum del despertar”. A més, se l'avisa de que les seves competències seran analitzades (un petit avís al jugador de que ho ha de fer bé, que hi ha puntuació). Aquestes quatre proves li presenten diversos reptes. Per començar, a la sala dels enigmes, la seva intel·ligència es veu posada a prova. Haurà de superar tres preguntes, de dificultat progressiva. Després, apareix la sala del laberint, on serà la seva orientació espacial la que es veurà en problemes. Una mica de traça li serà necessària a l'hora d'arroseggar la bola per la sala de la bola, amb cura de que no li caigui. I s'haurà d'enfrontar amb un repte de salts a l'última sala, la de les plataformes. Només superant les quatre proves s'obrirà la sala final, on espera la llum del jugador i la seva puntuació.

3.3 Model gràfic

3.4.1 Estructura d'escenes

Cada nou escenari del joc s'anomena escena. Permeten crear entorns gràfics totalment independents entre ells. Així doncs, el videojoc consta de sis escenes principals, a saber:

1. **El rebedor:** La sala principal del videojoc, on comença i es retorna cada cop que es supera una prova. Consta de comunicacions amb totes les altres escenes, mitjançant sensors de canvi d'escena col·locats a cada sortida. Un cop acabada una prova, una animació farà sortir una barrera que impedirà el pas del jugador amb la fi de que no la repeteixi (ja que no es pot tornar al rebedor si no es supera la prova en la que has entrat, i la barrera de l'última sala està programada per comptar les vegades que superes una prova i obrir-se quan n'has superat quatre. Realitzar una mateixa prova més d'un cop comportaria l'obertura de la porta final abans d'hora).
2. **La sala dels enigmes:** Una sala senzilla, composta de quatre cilindres i connexions entre ells. Les connexions entre els cilindres estan bloquejades per portes excepte entre els dos primers, que són el d'aparició i la primera sala. Aquestes portes s'obren al superar l'enigma corresponent. La primera porta només requereix reconèixer limatge-solució amb un clic, la segona requereix de dos clics consecutius en diferents imatges-solució i la tercera tres. D'aquí que la dificultat sigui progressiva.
3. **La sala del laberint:** Un laberint en el que es posa a prova l'orientació espacial del jugador. La programació d'aquesta sala és prou simple: només consta de l'entorn, el jugador i la sortida. L'objectiu, obviament, és arribar a la sortida.
4. **La sala de les plataformes:** Consistent en saltar d'una plataforma a una altra, aquesta sala posa a prova la coordinació del jugador a l'hora de saltar. Per poder arribar amb èxit a la següent plataforma, s'ha de sincronitzar bé l'acció de córrer amb el salt.
5. **La sala de la bola:** Una bola espera davant del jugador a que aquest faci la seva aparició: l'objectiu del jugador serà arrosseggar-la mitjançant empentes sobre un pla sense que la bola o ell caiguin fins un receptacle que hi haurà al final de la sala. Fins que la bola no arribi al receptacle, aquesta prova no es podrà resoldre, ja que hi ha una barrera associada a una animació que només desapareixerà quan la bola entri a dins del receptacle.
6. **La sala final:** Un petit detall per acabar el joc amb més calma, i mostrar que no només consisteix en acabar les quatre proves. Dóna un sentiment de conclusió al jugador, que és molt necessari en un joc. De no ser així, la sensació final es veuria bastant afectada. La sala final és una sala circular, amb una relíquia al mig (representant "la llum del despertar" a obtindre pel jugador). La col·lisió del jugador amb la relíquia activa els mòduls finals de càlcul de puntuació que descriuré amb detall en l'apartat de lògica.
7. **Escena de puntuació (extra):** Depenent de la puntuació obtinguda en el joc, un cop calculada pel mòdul, el jugador es transportarà a una última escena consistent en un pla 2D, sense possibilitat de moviment ni interaccions excepte la de prémer la tecla 'ESC', que tancarà el joc. En aquest pla apareixerà la puntuació final i una última frase per conoure el joc. Tenint en compte que la puntuació varia de 0 a 10, s'hauran de fer 11 escenes a les que el jugador podrà accedir en funció de la seva puntuació (només veurà l'escena corresponent a la nota obtinguda).

3.4.2 Objectes

Els objectes que componen les escenes es representaran com a models polièdrics, és a dir que es modelitzarà la seva superfície frontera mitjançant una malla de polígons. Per a donar realisme visual i evitar haver de subdividir excessivament la malla poligonal, s'aplicaran textures que els hi donaran realisme. La figura 3 il·lustra el concepte de model polièdric:

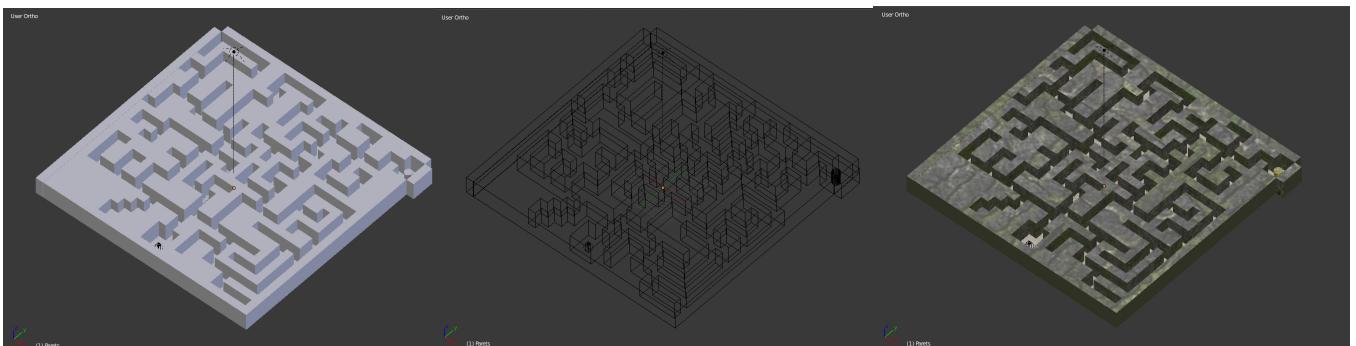


Figura 3: A l'esquerra, la sala del laberint com a sòlid polièdric. Al centre, la mateixa sala representada en filferros “wire” (malla de polígons). A la dreta, amb les textures aplicades.

4. Implementació

4.1 Tecnologia utilitzada

D'acord amb un dels objectius del treball, tot el programari utilitzat en aquest projecte és lliure. Concretament, i com s'ha dit anteriorment, s'ha utilitzat:

- el sistema operatiu linux, concretament la distribució Ubuntu 12.10
- el sistema de modelatge 3D Blender 3D
- el motor de joc Blender Game Engine
- el llenguatge de programació Python
- l'editor d'imatges Gimp per a crear i editar textures

4.2 Objectes

Per a modelitzar els objectes, s'ha seguit una metodologia en tres passos:

- modelització geomètrica de la forma externa dels objectes
- definició de materials i textures
- definició de les propietats físiques

A l'annex 2, a l'apartat *modelació, material i aplicació de les textures*, es pot veure un exemple complet de modelització d'objecte.

El paradigma de modelització bàsic utilitzat per a dissenyar la forma de l'objecte és el d'escombrat translacional: es crea un polígon 2D i s'escombra en la direcció perpendicular al pla, generant així el volum 3D i les cares laterals. La figura 4 il·lustra aquest procés:

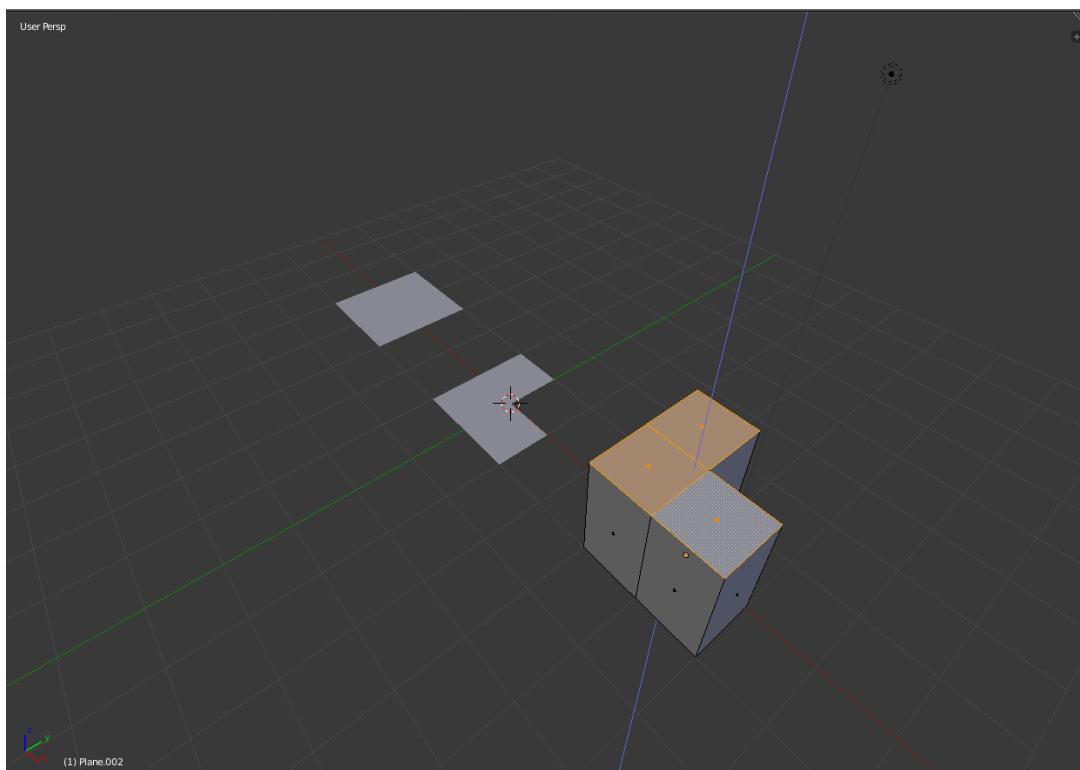


Figura 4: Modelització bàsica d'un objecte. Es crea un pla, es subdivideix i se n'eliminen les cares no desitjades, i a continuació s'escombra en la direcció perpendicular al pla per generar el volum.

Per a definir un material, cal seleccionar les opcions adients en el panell corresponent de Blender (figura 5). Podem assignar-li un color, encara que aquesta opció no té gaire sentit si més endavant volem definir-li una textura.

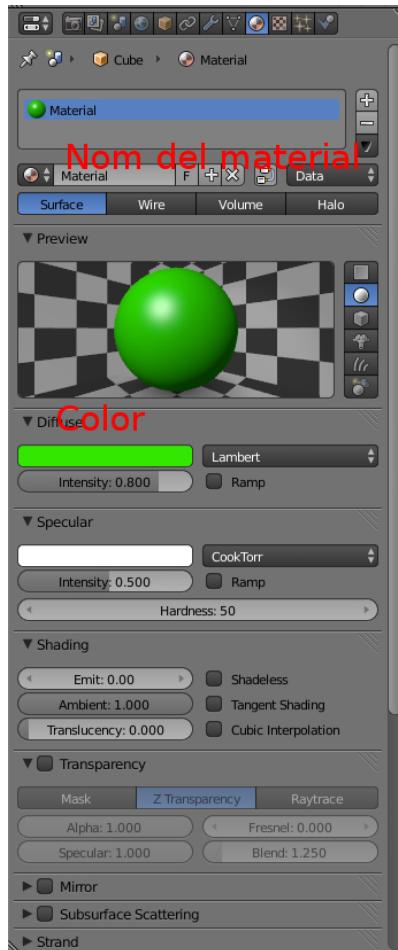


Figura 5: Panell de material de Blender

Finalment, les textures són imatges que s'apliquen a la superfície de l'objecte. Cal ser curós en la seva aplicació per a assegurar-se de que queden correctament orientades i són visibles. Aquest ha estat sense dubte una de les parts més lentes del treball. Cal tenir en compte els factors següents que han donat no pocs problemes en la realització del treball:

- les mides de les imatges han de ser potències de 2: 256, 512, 1024...
- s'han d'aplicar en modalitat *UV mapping*, és a dir desplegar-les sobre la superfície de l'objecte
- cal activar que s'apliquin a les dues cares d'un polígon si és necessari (desactivar *backface culling*)

De la mateixa manera que amb els materials, a l'hora de definir una textura cal seleccionar les opcions adients en el panell corresponent de Blender (figura 6). No obstant, cal tenir en compte que les textures depenen dels materials, el que vol dir que es defineix una textura per a un material, no per a un objecte. Això permet que un mateix objecte presenti diverses textures alhora, ja que pot estar compost de diversos materials.



Figura 6: Panell de textures de Blender. A la part de dalt es pot observar l'estructura jeràrquica que segueix l'aplicació de materials i textures. L'objecte depèn de l'escena, el material de l'objecte i la textura del material.

Finalment, dins de la interfície de Blender, es troben els objectes de la nostra creació a l'anomenat *outliner* (figura 7).

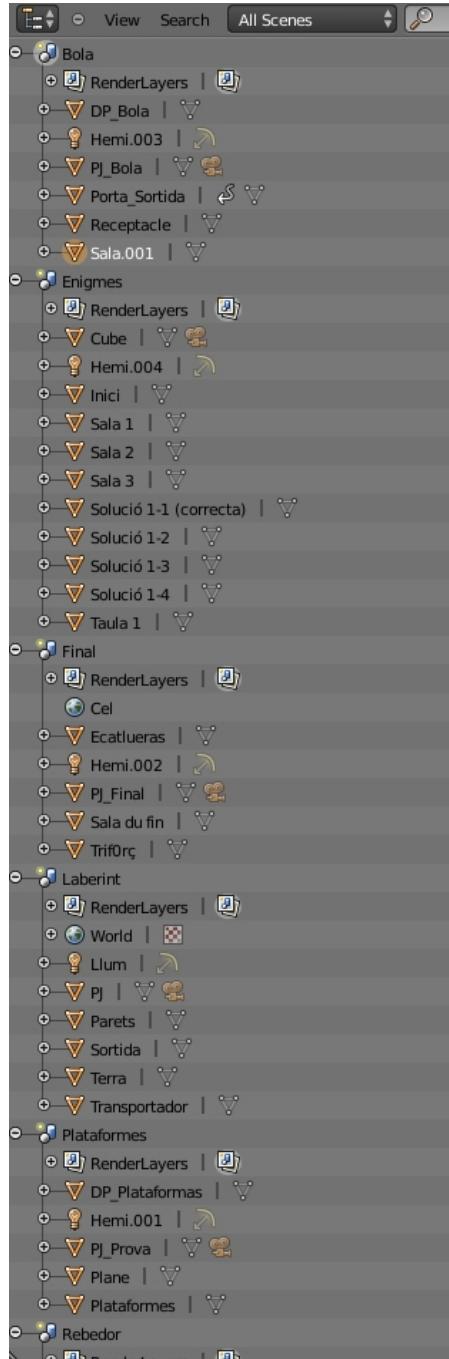


Figura 7: Llistat d'objectes del joc (outliner)

Les escenes del joc, compostes per aquests objectes s'implementen en Blender directament amb el concepte de *scene*.

4.3 Navegació

La navegació és l'acció que realitza un objecte quan es desplaça sobre l'entorn gràfic prèviament creat. Majoritàriament, la navegació és una característica pròpia del personatge principal del joc, encara que hi ha casos on s'han d'assignar unes ordres de navegació a altres objectes.

En cas de que el joc que s'està desenvolupant sigui en primera persona, o en tercera amb càmera lliure, s'ha de considerar navegació també al moviment de la càmera. En el cas del

videojoc realitzat a aquest treball, que és en primera persona, la càmera és controlada amb el ratolí mitjançant el *script mouselook* (*annex 1 – mouselook per Riyuzakisan*).

La lògica de navegació reutilitzada per al personatge de totes les *scenes* és la següent:

SENSOR (tipus – nom - sensor)			CONTROLADOR (tipus - controlador)		ACTUADOR (tipus – nom - acció)		
Keyboard	MoveFW	W	And	-	Motion (simple)	MoveFW	Y: 0.08
Keyboard	Run	Shift	And	MoveFW+Run	Motion (simple)	Run	Y: 0.1
Keyboard	MoveBW	S	And	-	Motion (simple)	MoveBW	Y: -0.04
Keyboard	LatL	A	And	-	Motion (simple)	LatL	X: -0.07
Keyboard	LatR	D	And	-	Motion (simple)	LatR	X: 0.07
Keyboard	Jump	Spacebar	And		Motion (character)	Jump	Z: 0.07
Mouse (càmera)	Mouse	Move	Python	mousemove.py (<i>annex 1</i>)	-	-	-

Així doncs el moviment del personatge és associat a les tecles WASD. En el cas de que volgués que es pogués moure amb altres tecles (sense que deixés de ser possible amb les actuals), tan sols hauria de definir els mateixos actuadors un altre cop i crear sensors amb una tecla diferent (com podrien ser les fletxes direccionals). De tota manera, no ho he fet, perquè no ho he trobat necessari. La barra espaiadora queda molt mal posicionada si el jugador s'ha de moure amb les fletxes, doncs s'ha de tenir en compte que segueix movent la càmera amb el ratolí, amb el que queden les mans massa junes i en una posició poc ergonòmica.

4.4 Interacció

Si bé la navegació és el seguit de propietats individuals que permeten a un objecte navegar per l'entorn gràfic, sent completament independent a aquest entorn (excepte en el que respecta a la física) la interacció és la propietat d'un objecte més classificable a l'acció de grup, doncs és la reacció que tenen els objectes entre ells. Podem definir doncs la interacció com la propietat d'un objecte de provocar canvis en l'entorn gràfic i la lògica del joc.

Per a associar accions a interaccions sobre objectes, Blender defineix tres conceptes: sensors, controladors i actuadors, i proporciona una interfície gràfica per a relacionar-los. La figura 8 il·lustra aquest concepte.

Un *sensor* és un tipus d'esdeveniment que el joc ha d'escoltar en relació a l'objecte, prémer una tecla, per exemple. Un *controlador* defineix les condicions addicionals que es volen imposar als sensors perquè l'acció desitjada es realitzi. Finalment, un *actuador* defineix l'acció concreta que es vol realitzar. Per defecte s'utilitza el controlador *And*, tot i que si

s'uneixen dos sensors a aquest controlador per a un sol actuador, el controlador *And* no deixarà que l'actuador s'activi si no hi ha dos polsos positius per part dels sensors associats. Per exemple, volem que amb la tecla *Shift* el nostre personatge corri, però no volem que pugui córrer si no s'està desplaçant cap endavant, ordre que tenim assignada per a la tecla *w*. Si associem el sensor *w* i el sensor *Shift* al controlador *And* per un actuador de moviment (que, obviament, tindrà una velocitat major que el desplaçament normal), aquest no funcionarà fins que no pressionem *w* i *Shift* a la vegada. Els controladors més clàssics, però, són els scripts, línies de codi que compliquen la lògica fins a punts que no arribaria el Game Engine normal.

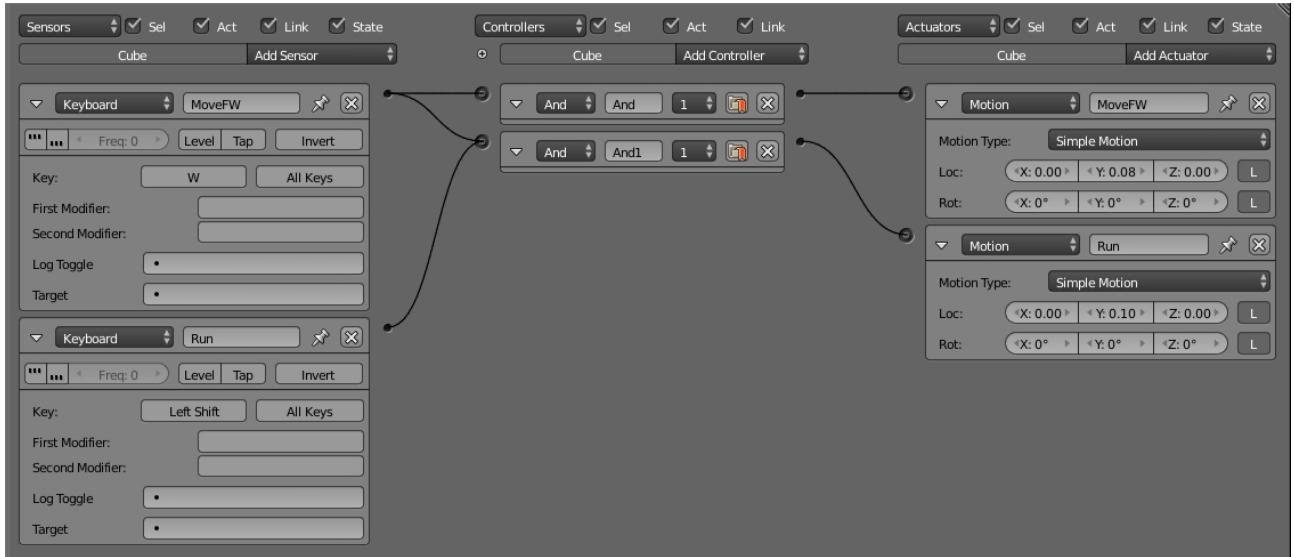


Figura 8: Exemple esmentat representat de forma gràfica

En el marc del joc, la majoria dels sensors estan associats a interaccions directes de l'usuari (tecles o moviments de ratolí). Cal observar, però, que es poden desencadenar interaccions en cadena, en la que la interacció principal del personatge sobre un objecte X resulti en una interacció de l'objecte X sobre un altre objecte Y.

Les interaccions que pot realitzar el personatge depenen de l'entorn gràfic on es trobi. Per això, definiré les possibles interaccions per a cada sala.

4.4.1 Interacció al rebedor

SENSOR			CONTROLADOR		ACTUADOR		
Touch	Enigmes	Enigmes	And	-	Scene	Enigmes	Set Scene: Enigmes
Touch	Laberint	Laberint	And	-	Scene	Laberint	Set Scene: Laberint
Touch	Bola	Bola	And	-	Scene	Bola	Set Scene: Bola
Touch	Plataformes	Plataformes	And	-	Scene	Plataformes	Set Scene: Plataformes
Touch	Final	Final	And	-	Scene	Final	Set Scene: Final

S'observa clarament que el tipus d'interaccions permeses al rebedor no són més que els canvis de sala. S'ha de tenir en compte, a més, que al principi la interacció de *Touch* amb *Final* no és possible, doncs la sala final està bloquejada fins que es superin les altres quatre. També cal pensar que, cada cop que es soluciona una prova, la interacció que porta al personatge a tornar-hi tampoc estarà permesa.

4.4.2 Interacció a la sala dels enigmes

La interacció a la sala dels enigmes és, sens dubte, la més complexa del videojoc. Requereix sensors de raig, doncs cada cop que el personatge tria una solució el programa ha de detectar si és correcta o no. A més, a partir del segon enigma, la cosa es complica, doncs si la resposta és correcta ha d'esperar a una segona resposta correcta per procedir, transformant així la resposta abans incorrecta en correcta, i la primera també en incorrecta, doncs no té sentit clicar la mateixa resposta dos cops. Es més, si es falla en el segon clic, s'ha de fer que la primera resposta torni a ser correcta. Aquest procés es veu esquematitzat als scripts corresponents, presents a l'annex 1, on es veurà explicat amb molta més claredat que en aquest apartat, només dedicat a la interacció de l'usuari.

4.4.2.1 Interacció al primer enigma

SENSOR			CONTROLADOR		ACTUADOR		
Mouse	Primer	Left Button	Python	enigma1.py	Action	Porta1	Action (play): porta1

4.4.2.2 Interacció al segon enigma

SENSOR			CONTROLADOR		ACTUADOR		
Mouse	Segon	Left Button	Python	enigma2.py	Action	Porta2	Action (play): porta2

4.4.2.3 Interacció al tercer enigma

SENSOR			CONTROLADOR		ACTUADOR		
Mouse	Tercer	Left Button	Python	enigma3.py	Action	Porta3	Action (play): porta3

4.4.2.4 Interacció a la sortida

SENSOR			CONTROLADOR		ACTUADOR		
Touch	Sortida	Sortida_Eni	And	-	Scene	Scene	Set Scene: Rebedor
					Action	Engimes	Action (play): barrera_eni
					Python	provasuperada.py	-

4.4.3 Interacció a la sala del laberint

SENSOR			CONTROLADOR		ACTUADOR		
Touch	Sortida	Sortida_Lab	And	-	Scene	Scene	Set Scene: Rebedor
					Action	Laberint	Action (play): barrera_lab
					Python	provasuperada.py (annex 1)	-

Com es pot observar, les interaccions a la sala del laberint es redueixen a una de sola: la sortida. Analitzant un laberint es veu clar que és l'única interacció possible, superar-lo. Per això resulta la prova més fàcil de dissenyar. Quan el jugador toca el material “sortida” (associat a l'objecte “sortida”, obviament), la *scene* canvia al rebedor i s'executa l'animació, prèviament creada, d'una barrera que bloquejarà el pas al jugador si vol tornar a superar la sala del laberint. A més, s'executa el mòdul *provasuperada.py*.

4.4.4 Interacció a la sala de les plataformes

SENSOR			CONTROLADOR		ACTUADOR		
Touch	Sortida	Sortida_Plat	And	-	Scene	Sortida	Set Scene: Rebedor
					Action	Plataformes	Action (play): barrera_plat
					Python	provasuperada.py	-
Touch	Fail	DP_plat	Python	fail.py (annex 1)	Scene	Restart	Restart Scene

La sala de les plataformes no té molta més complexitat que la del laberint en quant a

interaccions. Tan sols afegeix la possibilitat de “morir”, consistent en què el jugador toqui el DP_plat (Death Plane), un pla localitzat per sota del nivell de les plataformes. Així, si el jugador cau, tocarà aquest pla, activarà el mòdul *fail* i la *scene* tornarà a començar.

4.4.5 Interacció a la sala de la bola

A la sala de la bola, el jugador no és l'únic que té interaccions amb l'entorn. També la bola en té, tot i que depenen del moviment que li dona el jugador. Degut a això, en aquest apartat definiré les interaccions del personatge i de la bola.

4.4.5.1 Interaccions del personatge

SENSOR			CONTROLADOR		ACTUADOR		
Touch	Sortida	Sortida_Bola	And	-	Scene	Sortida	Set Scene: Rebedor
					Action	Plataformes	Action (play): barrera_bola
			Python	provasuperada.py	-	-	-
Touch	Fail	DP_bola	Python	fail.py	Scene	Restart	Restart Scene

Substancialment iguals que les del personatge a la sala de les plataformes, només canvia el nom dels objectes amb els que reacciona.

4.4.5.2 Interaccions de la bola

SENSOR			CONTROLADOR		ACTUADOR		
Touch	Fail	DP_bola	Python	fail.py	Scene	Restart	Restart Scene
Touch	Meta	Receptacle	And	-	Action	Meta	Action (play): porta_bola

Les interaccions de la bola consisteixen en caure i, per tant, fallar (igual que el jugador), el que desemboca en la repetició de la *scene* i un *fail* més a tenir en compte en la puntuació, i arribar a la meta, un cop el jugador l'ha arrossegat fins allà, el que obre la porta per a poder sortir del nivell i tornar al rebedor.

4.4.6 Interacció a la sala final

SENSOR			CONTROLADOR		ACTUADOR		
Touch	Final	Trif0rç	Python	temps_fin.py (annex 1)	-		
Touch	Final	Trif0rç	Python	final.py (annex 1)	Scene	0	Set Scene: 0
					Scene	1	Set Scene: 1
					Scene	2	Set Scene: 2
					Scene	3	Set Scene: 3
					Scene	4	Set Scene: 4
					Scene	5	Set Scene: 5
					Scene	6	Set Scene: 6
					Scene	7	Set Scene: 7
					Scene	8	Set Scene: 8
					Scene	9	Set Scene: 9
					Scene	10	Set Scene: 10

La sala final només presenta una possibilitat d'interacció: tocar l'objecte *Trif0rç*, el que desencadena els mòdul *temps_fin* i *final*. El primer determina el temps final, necessari pel segon, ja que l'importarà junt amb el temps inicial per obtindre el temps total empleat en completar el joc. A més, aquest mòdul determina una nota en funció dels *fails* i el temps total empleat i, depenent de la nota, envia al jugador a una *scene* amb la seva nota corresponent.

4.4.7 Interacció a les “scenes de nota”

SENSOR			CONTROLADOR		ACTUADOR		
Keyboard	Tancar	ESC	And	-	Game	Final	Quit Game

Un cop el jugador es troba a la *scene* amb la seva nota corresponent, només li queda sortir

del joc i concloure -lo. En una indicació present a la *scene*, se li indicarà que ha de prémer la tecla 'ESC' per tancar. Aquesta serà l'última interacció del jugador.

4.5 Puntuació i operatura de la porta final

El control de la puntuació en el videojoc desenvolupat es basa en un mòdul principal anomenat *final.py*. Aquest mòdul utilitza, a la vegada, tres mòduls secundaris per funcionar, a saber: *puntuació.py*, *temps_ini.py* i *temps_fin.py*. El mòdul *puntuació.py*, a la vegada, té associat el mòdul *fail.py*, que modificarà el valor de la seva variable *fails* (*el codi dels mòduls es pot trobar a l'annex 1*). A més, també importa la lògica de Blender per a poder-s'hi referir.

La puntuació obtinguda pel jugador es determina per:

- **El nombre de vegades que s'ha fallat.** Es considera fallar (totes aquestes accions sumen 1 al valor *fails* del mòdul *puntuació.py*):
 - equivocar-se en una resposta a la Sala dels Enigmes.
 - caure a la Sala de les Plataformes.
 - caure a la Sala de la Bola.
 - que caigui la bola a la Sala de la Bola.
- **El temps total empleat en la resolució del joc.** Al iniciar el joc, s'executa el mòdul *temps_ini.py*, que utilitzant la funció `time.time()` associarà un nombre de segons a la variable *timeini*. Aquests segons són els retornats per la funció `time.time()`, que retorna el nombre de segons desde l'1 de gener de 1970, el moment anomenat dins de l'informàtica com "The Epoch". Al acabar-lo, just abans de que s'executi el mòdul *final.py*, s'executarà *temps_fin.py*, que determinarà, de la mateixa manera que *temps_ini.py*, un nombre de segons per a la variable *timefin*. Aquestes dues variables seran importades pel mòdul principal *final.py*, que associarà a una nova variable, *temps*, la resta de *timefin* – *timeini*.

El funcionament del mòdul final, en sí, és prou simple. En primer lloc importa el mòdul *puntuació.py*, del que n'extreu la variable *fails* per a poder-la utilitzar. A continuació, com s'ha explicat anteriorment, importa *temps_ini.py* i *temps_fin.py*, fa la resta de temps final menys inicial i associa a la variable *temps* aquest valor. Un cop fet això, defineix la funció que donarà la nota final, una mitjana aritmètica: *puntuacionfinal(a, b)*. Aquesta mitjana està en funció de dues variables encara no definides, *a* i *b*. Aquestes variables seran una associació tant del nombre de vegades que s'ha fallat com del temps total empleat en la resolució del joc a una nota numèrica, del 0 al 10. Per fer això, s'han de definir un interval de valors.

En primer lloc, el mòdul procedeix a definir la variable *a*. Després de contrastar resultats amb els *beta testers*, he determinat un interval d'un punt menys cada cinc fallides. Això vol dir que, tenint de zero a quatre fallides, *a* serà igual a 10. A partir de cinc, serà 9, i així successivament, fins a tenir un 0.

En segon lloc, es determina la variable *b*. Utilitzant de nou els resultats obtinguts en el procés de *beta testing*, he determinat un interval d'un punt menys cada cinc minuts, excepte al principi, que han de passar deu minuts per a tenir un 9 en comptes d'un 10. A partir d'aquest moment, cada cinc minuts es perd un punt en la puntuació de temps.

Un cop determinades `a` i `b`, s'executa finalment la funció definida al principi, però s'executa associada directament a una nova variable, `c`. Al executar la funció assignant-la directament a una variable, s'aconsegueix que `c` tingui, en aquest moment, la nota final. Si el joc fós una aventura conversacional, es podria demanar directament a Python que fés un `print(c)`, amb el que la nota apareixeria a la pantalla i l'usuari ja quedaría satisfet. Però no és el cas. Es necessita, doncs, alguna manera de fer referència a una acció dins de Blender, doncs és Blender el que està determinant la visualització del jugador. És en aquest moment quan ens referim al propi controlador, havent importat prèviament el Game Engine de Blender o `bge.py` per poder-hi fer referències. Li demanem al programa que, en funció de la nota, ens passi a una *scene* concreta. Aquesta *scene* és la ja comentada amb anterioritat *scene de nota*: recordant el que era, un pla 2D amb la nota impresa sobre aquest.

A l'annex 1 es pot veure el codi d'aquest mòdul en la seva totalitat.

La porta final del rebedor s'obre quan s'han superat les quatre proves. El joc ho detecta perquè, cada cop que es supera una, s'executa el mòdul `provasuperada.py`, que suma un punt a la variable `proves` de `proves_superades.py`. Cada cop que s'accedeix a la sala del rebedor, s'executa el mòdul `sala_final.py`. Aquest importa el valor de la variable `proves` i, en cas de que sigui igual a 4 (el nombre total de proves), executa l'animació que fa obrir-se l'última porta. Aquest conte de proves superades és el motiu per el qual, un cop superada una prova, no es permet tornar a realitzar-la.

5. Resultats

El resultat material del treball ha estat el videojoc en funcionament. Com el videojoc s'ha acabat molt al final del treball, encara no s'ha pogut fer una prova sistemàtica de les seves funcionalitats amb usuaris independents. Això no obstant, el joc ha estat provat molt extensament pel seu creador, jo mateix, i amb molta freqüència pels amics i familiars. D'aquesta manera s'han pogut depurar errors, a més de fixar els cànons de puntuació, impossibles de determinar sense un previ estudi de la mitjana de temps i de fallides.

Respecte als plànols realitzats en un inici (presents a l'*annex 2*), l'únic concepte que no s'ha aconseguit assolir és el de la il·luminació per torxes.



Figura 9: Videojoc en funcionament

6. Conclusions i línies de treball futures

6.1 *Conclusions generals*

Realitzar aquest treball ha sigut beneficis en tots els sentits. En ser el desenvolupament d'un videojoc una tasca normalment realitzada per moltes més personnes, realitzar-la individualment m'ha迫çat a endinsar-me en varis disciplines alhora. Per això crec important remarcar, com a conclusió general, aquest mateix aspecte: la **pluridisciplinaritat** del treball realitzat. En aquest treball s'ha treballat dins de l'àmbit:

- **Tecnològic**, aprenent a programar i a fer servir gran quantitat de programari lliure del que abans no tenia pràcticament coneixement.
- **Físic**, treballant amb les Lleis de Newton i aplicant propietats físiques als objectes del videojoc.
- **Històric**, estudiant i redactant la història que gira entorn als videojocs.
- **Creatiu/literari**, a l'hora de desenvolupar un guió pel videojoc.
- **De dibuix tècnic**, consolidant conceptes d'orientació espacial, de vistes i treball exhaustiu amb coordenades cartesianes.
- **De dibuix artístic**, amb el disseny de nivells i textures.
- **Lingüístic**, per a la redacció de la present memòria.

6.2 *Línies de treball futures*

Havent acabat el treball, em segueixen venint noves idees a la ment que implementar al joc. Per això, com a línia de treball futura veig un gran aprofitament de les competències obtingudes durant el desenvolupament del videojoc, amb possibilitats de creació de videojocs més complexos, o la millora del realitzat en aquest treball.

6.3 *Opinió personal*

Com a opinió personal, no puc fer res més que no sigui expressar la més gran satisfacció sobre el treball realitzat. Ha sigut un treball dur, amb problemes inesperats en els pitjors moments (ja se sap com pot arribar a desesperar la informàtica), però cada cop que solucionava un d'aquests problemes no feia més que alegrar-me a més no poder i desitjar solucionar el següent. Sento que he assolit moltíssimes competències, que he aprofitat bé el treball de recerca, i ha sigut tot el contrari del que es sol dir: que és pesat, que no agrada, que només fa que destrossar-te l'estiu... Puc afirmar amb total convicció que haver-lo realitzat em servirà en un futur, ja no només dins de l'àmbit professional (que, havent contrastat els aprenentatges de programació amb un amic cursant primer d'informàtica, la carrera que vull fer, ja he vist que havia après coses que ell encara no), sinó dins del meu àmbit personal, doncs m'ha despertat de nou l'interès per crear petits jocs i... qui sap! Tal com està ara mateix l'escena dels videojocs, on estan tornant a triomfar els més simples, podria ser jo qui tingués ara una idea per un nou joc d'aquests. No és aquesta la finalitat del treball? Realitzar-nos dins de l'àmbit que ens agrada, portar les nostres capacitats al límit i desenvolupar la il·lusió de poder-ho seguir fent en un futur.

Glossari

Actuador: Dintre de la interfície gràfica de la lògica de Blender, l'actuador és l'acció en resposta al pols positiu del sensor associat.

Arcade: Mànica recreativa, que funciona amb monedes. Amb la més recent desaparició d'aquestes, es classifica de “jocs arcade” els que segueixen, d'una manera o altra, els cànons que compartien aquestes (molts jocs de lluita, per exemple, es denominen jocs arcade).

Atari: Importantíssima empresa estatunidenca, creadora de l'arcade *Pong* i primera en obtenir beneficis milionaris mitjançant la distribució de videojocs, a més de ser creadora de grans clàssics.

Beta: Versió quasi final d'un videojoc, en la que aquest està complet, però falten per solucionar alguns errors.

Beta testers: Grup de persones que proven la versió *beta* d'un videojoc amb la finalitat de trobar el màxim d'errors possibles per a la seva posterior solució.

Combos: Combinacions d'atacs (jocs de lluita).

Controlador: Dintre de la interfície gràfica de la lògica de Blender, el controlador és la condició extra que se li dóna al sensor perquè executi l'actuador.

CPU: *Central Processing Unit* (Unitat Central de Processament). Part de l'ordinador encarregada de processar les ordres dels programes informàtics.

FPS: *First-Person Shooter*. Joc de trets en primera persona.

Game Engine: Sèrie de routines de programació que permeten el disseny, la creació i la representació d'un videojoc.

Game loop: Bucle que segueix un videojoc, fins que el jugador el supera (programa espera pregunta --> jugador pregunta --> programa respon).

GPU: *Graphics Processing Unit* (Unitat de Processament Gràfic). Processador addicional encarregat de processar els gràfics per tal de reduir la feina a fer per la CPU.

Hacker: Pirata informàtic o, més comunament, programador o tècnic especialment hàbil.

IA: Intel·ligència Artificial

Linux: Sistema operatiu basat en Unix de codi obert.

Modelatge geomètric: Acció consistent en, mitjançant les eines proporcionades pel programa on es treballa, crear formes geomètriques.

Nim: Joc d'estratègia matemàtica.

Nintendo: Importantíssima empresa japonesa, primera en aconseguir desbancar a Atari del seu èxit mundial, i creadora de molts grans clàssics.

Pantalla raster: Pantalla capaç de plasmar gràfics en format d'àrea, en comptes de vectorialment.

PDP-1: Primer computador de la sèrie PDP, en el que es va dissenyar *Spacewar!*

Pong: Primer èxit d'Atari, plagi del *Ping-Pong* de Ralph Baer (que a la vegada és plagiat de *Tennis for Two* de William Higinbotham).

Programari Iliure: Programari que pot ser usat, estudiat i modificat sense restriccions, i que pot ser copiat i redistribuït bé en una versió modificada o sense modificar sense cap restricció, o bé amb unes restriccions mínimes per garantir que els futurs destinataris també tindran aquests drets.

Python: Llenguatge de programació

Rendering: Procés de generar una imatge a partir dels càlculs de llum pertinents.

Script: Codi dissenyat en un llenguatge de programació específic, amb unes ordres que seguirà l'ordinador a l'hora de fer els càlculs.

Sensor: Dintre de la interfície gràfica de la lògica de Blender, el sensor és el canal que deixa obert l'objecte mitjançant el qual es pot interactuar amb ell, provocant una reacció de l'actuador al que està associat.

SmartPhone: Telèfon amb un sistema operatiu avançat, capaç de complementar-se amb la descàrrega d'applicacions i, excepte en els telèfons Apple, de codi obert.

Test de Turing: Test inventat pel reconegut matemàtic Alan Turing en el que un jutge es posa en un ordinador, rebent missatges d'una persona i d'un programa, sense saber qui és qui. Mitjançant preguntes realitzades als dos, ha d'endevinar qui és qui. Si el jutge falla, la Intel·ligència Artificial és considerada igual a la humana. El Test de Turing només ha sigut superat un cop a la història,

TPS: *Third-Person Shooter*. Joc de trets en tercera persona.

Ubuntu: Distribució de Linux

Windows: Sistema operatiu de codi propietari.

Bibliografia

Pàgines web consultades:

<http://www.vandal.net/noticia/38108/los-videojuegos-siguen-siendo-el-principal-mercado-del-ocio-audiovisual-en-espana/>

<http://pyspanishdoc.sourceforge.net/tut/node8.html>

<http://pycartagena.pbworks.com/w/page/37765255/Módulos>

<http://docs.python.org/dev/reference/import.html>

http://es.wikipedia.org/wiki/Historia_de_los_videojuegos

<http://www.blender.org/>

<http://es.wikipedia.org/wiki/Rasterización>

<http://es.wikipedia.org/wiki/GNU/Linux>

http://es.wikipedia.org/wiki/Núcleo_Linux

http://es.wikipedia.org/wiki/Game_engine

http://ca.wikipedia.org/wiki/Programari_lliure

<http://es.wikipedia.org/wiki/Rendering>

Tutorials:

Modelatge: http://www.youtube.com/watch?v=wcNukwr3b5w&list=PL7Z1KKHB1pIFyPS4M_LautLogISrDQ2HE ,
<http://cgcookie.com/blender/lessons/3-modeling/>

Primer Game Engine: <http://www.youtube.com/watch?v=lJRhwDmteI0>

Joc del Laberint:

http://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro/An_aMAZEing_game_engine_tutorial

ANNEXOS

Annex 1: Mòduls utilitzats

1.1 mouselook.py (realitzat per Riyuzakisan)

Aquest mòdul és l'únic utilitzat en tot el desenvolupament del videojoc que no he realitzat jo mateix. Això és degut a la gran complexitat que té, i a que el necessitava aplicar per força per a fer funcionar el meu videojoc.

El mòdul de *mouselook*, està disponible de forma gratuïta a la pàgina web de Riyuzakisan, un usuari de Python. Aquest mòdul té 788 línies de codi, motiu pel qual no l'afegiré en aquest annex (occupa 17 pàgines), però en deixo la referència:

<http://riyuzakisan.weebly.com/mousemove-script.html>

Aquest mòdul serveix per a poder moure la càmera amb el ratolí.

1.2 enigma1.py

Encarregat de fer els càlculs adients per, cada cop que es fa clic sobre un objecte al primer enigma de la sala dels enigmes, saber si s'ha encertat o no. En ser aquest el primer enigma, només s'ha de triar un objecte. Per tant, és més fàcil de programar.

Codi:

```
from bge import logic as GL
import puntuacio
action = own.actuators['portal']
cont = GL.getCurrentController()
sensor = cont.sensors['Mouse1']

if sensor.positive :
    seleccionat = sensor.hitObject
    if seleccionat.name == 'Solució_1-1a':
        action.action = 'portal'
        cont.activate(action)
    elif seleccionat.name == 'Solució_1-2':
        puntuaciofails = puntuaciofails + 1
    elif seleccionat.name == 'Solució_1-3':
        puntuaciofails = puntuaciofails + 1
    elif seleccionat.name == 'Solució_1-4':
        puntuaciofails = puntuaciofails + 1
```

Aquest mòdul fa les següents operacions:

1. Importa la lògica de Blender Game Engine, amb el que ens podrem referir, dins del codi, a objectes situats a l'entorn gràfic del Game Engine.
2. Importa el mòdul *puntuacio.py*, per poder-se referir a la variable “fails” més endavant.
3. Associa l'actuador d'acció 'portal' a la variable 'action', i els controladors a 'cont', a més del sensor de ratolí 'Mouse1' (el sensor que detecta el clic) a 'sensor'

4. Mitjançant una estructura condicional, defineix el que ha de passar quan el sensor de *mouse* sigui positiu (és a dir, quan es faci clic esquerra). Obté el nom de l'objecte amb el que ha impactat el raig que ha llançat el clic (un raig imaginari, la selecció del ratolí) i determina quina ha de ser la solució correcta. En aquest cas, 'Solució_1-1a' és la correcta, doncs executa l'acció 'porta1' (l'animació d'obrir la porta al següent enigma). Fer clic a qualsevol de les altres solucions desembocarà en l'augment del valor *fails* en 1.

1.3 *enigma2.py*

Presenta la mateixa estructura que *enigma1.py*, però més complexa. Ara caldrà determinar més d'un pas, doncs s'han de triar, per ordre, dues solucions. Això vol dir que la solució que era correcta en un principi, no ho serà després.

Quan es tracta de programar scripts d'aquesta mena, hi ha tantes formes de fer-ho com formes se li acudeixin als programadors. Jo ho he fet de la següent forma:

Codi:

```
from bge import logic as GL
import puntuacio
import correctes as a
action = own.actuators['porta2']
cont = GL.getCurrentController()
sensor = cont.sensors['Mouse2']

if sensor.positive and a == 0:
    seleccionat = sensor.hitObject
    if seleccionat.name == 'Solució_2-1a':
        a += 1
    elif seleccionat.name == 'Solució_2-2b':
        puntuacio.fails = puntuacio.fails + 1
    elif seleccionat.name == 'Solució_2-3':
        puntuacio.fails = puntuacio.fails + 1
    elif seleccionat.name == 'Solució_2-4':
        puntuacio.fails = puntuacio.fails + 1

if sensor.positive and a == 1:
    seleccionat = sensor.hitObject
    if seleccionat.name == 'Solució_2-1a':
        puntuacio.fails = puntuacio.fails + 1
        a = 0
    elif seleccionat.name == 'Solució_2-2b':
        action.action = 'porta2'
        cont.activate(action)
    elif seleccionat.name == 'Solució_2-3':
        puntuacio.fails = puntuacio.fails + 1
        a = 0
    elif seleccionat.name == 'Solució_2-4':
        puntuacio.fails = puntuacio.fails + 1
        a = 0
```

Podem observar canvis entre el primer script i el segon. El primer és en la quantitat de mòduls secundaris que importa: aquest script importa, a més de tot el que importava el primer, un mòdul anomenat *correctes.py*, i li dona el nom de *a*. Aquest mòdul, com el mòdul de puntuació, no presenta més que una variable: *correctes = 0*. Encara que pugui semblar

absurd crear un mòdul secundari per importar aquest valor, s'ha de pensar en què passaria si definíssim aquesta mateixa variable en aquest mòdul. Cada cop que fem un clic s'executa, i per tant tornaria a assignar la variable a 0. Per tant, és preferible que la importi, la modifiqui, i quan es torni a executar el mòdul importi un altre cop la variable ja modificada.

El segon canvi important és que les condicions de l'`if` han augmentat. No només val que el sensor sigui positiu, doncs si això fós així s'executarien les dues estructures condicionals a la vegada. A més, les dos estructures han de presentar condició, doncs si només la segona tingués l'extra '`and a == 1`', no s'executaría al primer clic, però al anar a endevinar la segona resposta s'executarien les dues un altre cop.

Per finalitzar, en comptes d'una estructura condicional, en tenim dues. Per `a = 0`, el resultat d'endevinar la pregunta és que el valor d'`a` augmenti en 1. Això farà que, al segon clic, el mòdul s'executi tenint en compte que la variable `a` és 1, fent funcionar així la segona estructura condicional, que té com a resultat d'endevinar la pregunta l'obertura de la següent sala, i com a resultat de fallar, a més de la ja vista amb anterioritat suma d'un punt a la variable `fails`, l'establiment de la variable `a`, de nou, a 0.

1.4 *enigma3.py*

Aquest mòdul és idèntic a *enigma2.py*, però afegeix un nou subnivell. La segona estructura condicional, que abans executava l'animació, ara també suma 1 a la variable `a`, i és la tercera afegida aquí (amb una condició de `and a == 2`) la que executa l'animació per anar al final.

Codi:

```
from bge import logic as GL
import puntuacio
from correctes import correctes as a
action = own.actuators['porta3']
cont = GL.getCurrentController()
sensor = cont.sensors['Mouse3']

if sensor.positive and a == 0:
    seleccionat = sensor.hitObject
    if seleccionat.name == 'Solució_3-1a':
        a += 1
    elif seleccionat.name == 'Solució_3-2b':
        puntuacio.fails = puntuacio.fails + 1
    elif seleccionat.name == 'Solució_3-3c':
        puntuacio.fails = puntuacio.fails + 1
    elif seleccionat.name == 'Solució_3-4':
        puntuacio.fails = puntuacio.fails + 1

if sensor.positive and a == 1:
    seleccionat = sensor.hitObject
    if seleccionat.name == 'Solució_3-1a':
        puntuacio.fails = puntuacio.fails + 1
        a = 0
    elif seleccionat.name == 'Solució_3-2b':
        a += 1
    elif seleccionat.name == 'Solució_3-3c':
        puntuacio.fails = puntuacio.fails + 1
```

```

    a = 0
elif seleccionat.name == 'Solució_2-4':
    puntuaciofails = puntuaciofails + 1
    a = 0

if sensor.positive and a == 2:
    seleccionat = sensor.hitObject
    if seleccionat.name == 'Solució_3-1a':
        puntuaciofails = puntuaciofails + 1
        a = 0
    elif seleccionat.name == 'Solució_3-2b':
        puntuaciofails = puntuaciofails + 1
        a = 0
    elif seleccionat.name == 'Solució_3-3c':
        action.action = 'porta3'
        cont.activate(action)
    elif seleccionat.name == 'Solució_2-4':
        puntuaciofails = puntuaciofails + 1
        a = 0

```

1.5 proves_superades.py

Aquest mòdul secundari determina el nombre de proves que s'han superat. És necessari, ja que, cada cop que s'inicia l'*scene* del rebedor, s'executa el mòdul *sala_final.py* (a definir més endavant), i aquest n'exporta el nombre.

Codi:

```
proves = 0
```

1.6 provasuperada.py

Aquest mòdul és el que s'executa cada cop que es supera una prova. Simplement suma 1 a la variable *proves* del mòdul *proves_superades.py*.

Codi:

```
import proves_superades
proves_superades.proves += 1
```

1.7 sala_final.py

Aquest és el mòdul mencionat al punt 1.6. Cada cop que s'inicia l'*scene del rebedor*, s'executa, i quan la variable *a* (que és una associació feta a la variable *proves*) és igual a 0, la animació encarregada de retirar la porta final és executada.

Codi:

```
import bge
from proves_superades import proves as a
action = own.actuators['porta_final']

if a == 4 :
    action.action = 'porta_final'
    cont.activate(action)
```

1.8 puntuacio.py

Aquest mòdul secundari és tant simple com necessari. Només consta d'una variable, `fails`, però aquesta té una gran importància. S'ha de determinar un valor inicial per aquesta variable, de manera que els mòduls principals puguin obtindre-la i modificar-la.

Codi:

```
fails = 0
```

1.9 fail.py

Aquest mòdul representa, dins del joc, l'acció de fallar, que puja un punt a la variable `fails` de *puntuacio.py*.

Codi:

```
import puntuacio  
puntuaciofails += 1
```

1.10 time_ini.py i time_fin.py

És absurd fer dos apartats per aquests mòduls, doncs són exactament el mateix exceptuant el nom de la variable. Importen el mòdul *time.py*, un mòdul present dins dels predeterminats de Python, i executa la funció `time.time()`, definida amb anterioritat a la memòria, apartat 4.5: Puntuació. El fet de que hagi sigut necessari realitzar dos mòduls és degut a que un és executat a l'inici del joc i l'altre al final, just abans de rebre la puntuació. Aquests dos mòduls, junt amb *puntuacio.py*, seràn importats al mòdul de puntuació final, que serà determinat a continuació.

Codi *time_ini*:

```
import time  
timeini = time.time()
```

Codi *time_fin*:

```
import time  
timefin = time.time()
```

1.11 final.py

Aquest és el mòdul que determina la puntuació final en el joc. Realitza les següents funcions:

- Importa *bge.py*, la lògica de Blender, per poder-se referir a objectes presents en el Game Engine.
- Importa el nombre final de la variable `fails` del mòdul *puntuacio.py*.
- Importa `timeini` i `timefin` dels seus mòduls corresponents.
- Fa la següent resta: `timefin - timeini`, i assigna el resultat a la variable `temps`.
- Associa l'expressió de Blender per a obtenir un controlador a la variable `cont`.
- Associa els actuadors associats al controlador “nota” (controlador que tindrà l'script) a la variable `act`.

- Defineix la funció `puntuaciofinal(a, b)`, que realitza una mitjana aritmètica entre les variables `a` i `b`.
- Assigna, mitjançant intervals, el nombre de `fails` a una nota numèrica, sota el nom de la variable `a`.
- Assigna, mitjançant intervals, el `temps` a una nota numèrica, sota el nom de la variable `b`.
- Executa la funció `puntuaciofinal(a, b)` i assigna el resultat a la variable `c`.
- En funció de la nota obtinguda, que en aquest moment es troba continguda a la variable `c`, canvia l'`scene` a l'última amb el pla 2D amb la nota impresa sobre ell.

Codi:

```

import bge
import puntuacio
import temps_ini
import temps_fin
temps = timefin - timeini
cont = GameLogic.getCurrentController()
act = cont.actuators["nota"]
def puntuaciófinal(a, b):
    return (a + b) / 2

if fails == 0:
    a = 10
elif fails > 0 <= 5:
    a = 9
elif fails > 5 <= 10:
    a = 8
elif fails > 10 <= 15:
    a = 7
elif fails > 15 <= 20:
    a = 6
elif fails > 20 <= 25:
    a = 5
elif fails > 25 <= 30:
    a = 4
elif fails > 30 <= 35:
    a = 3
elif fails > 35 <= 40:
    a = 2
elif fails > 40 <= 45:
    a = 1
else:
    a = 0

if temps < 600:           #10 minuts
    b = 10
elif temps > 600 <= 900:   #15 minuts (+5 minuts sempre)
    b = 9
elif temps > 900 <= 1200:
    b = 8
elif temps > 1200 <= 1500:
    b = 7
elif temps > 1500 <= 1800:
    b = 6
elif temps > 1800 <= 2100:
    b = 5

```

```
b = 5
elif temps > 2100 <= 2400:
    b = 4
elif temps > 2400 <= 2700:
    b = 3
elif temps > 2700 <= 3000:
    b = 2
elif temps > 3000 <= 3300:
    b = 1
else:
    b = 0
```

```
c = puntuaciófinal(a, b)
```

```
if c == 10
    act.scene = "10"
elif c == 9
    act.scene = "9"
elif c == 8
    act.scene = "8"
elif c == 7
    act.scene = "7"
elif c == 6
    act.scene = "6"
elif c == 5
    act.scene = "5"
elif c == 4
    act.scene = "4"
elif c == 3
    act.scene = "3"
elif c == 2
    act.scene = "2"
elif c == 1
    act.scene = "1"
elif c == 0
    act.scene = "0"
```

Annex 2: Diari de Recerca

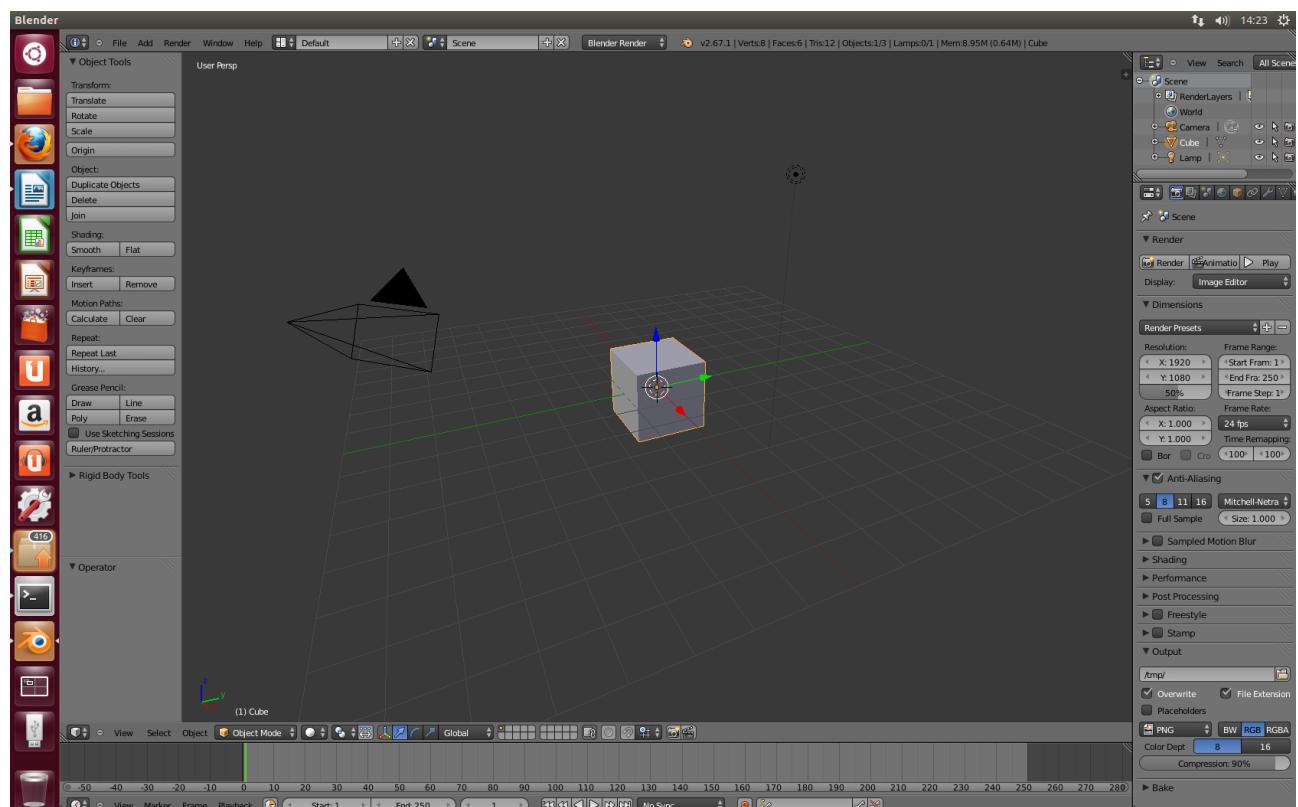
Fase 1: Tutorials de Blender sobre el modelatge (Fonts d'aprenentatge:

[http://www.youtube.com/watch?](http://www.youtube.com/watch?v=wcNukwr3b5w&list=PL7Z1KKHB1pIFyPS4M_LautLogISrDQ2HE)

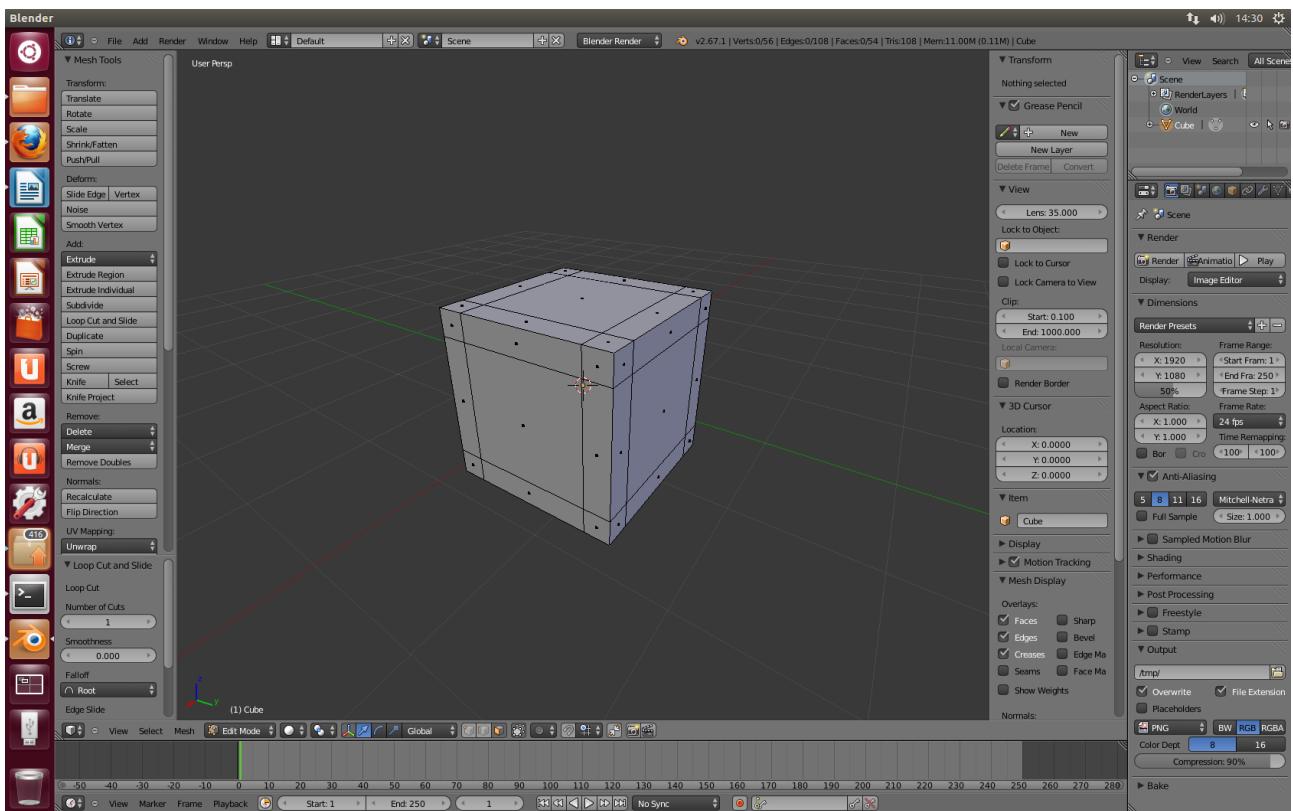
[v=wcNukwr3b5w&list=PL7Z1KKHB1pIFyPS4M_LautLogISrDQ2HE ,](http://cgcookie.com/blender/lessons/3-modeling/)

[http://cgcookie.com/blender/lessons/3-modeling/ \)](http://cgcookie.com/blender/lessons/3-modeling/)

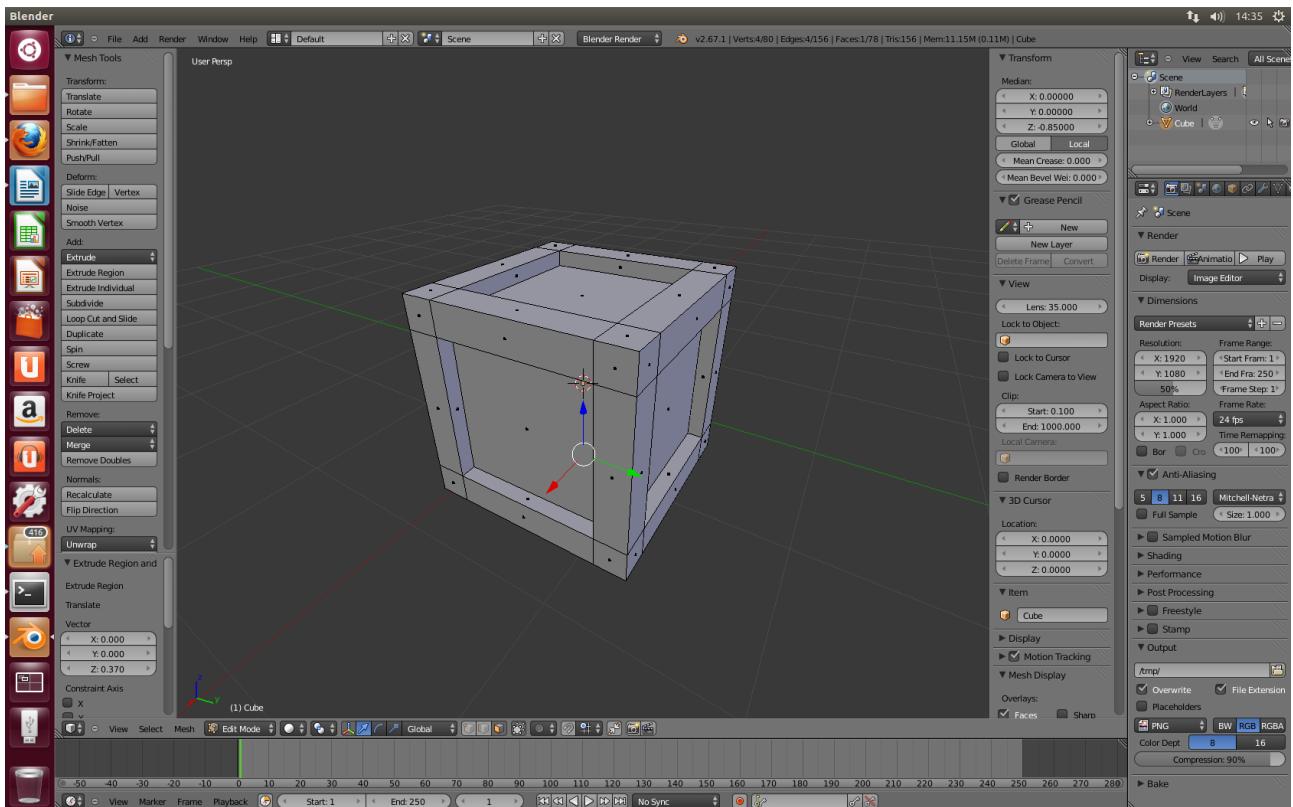
En aquesta serie de tutorials he après a crear, transportar i editar objectes en Blender. A més, he après la funció que tenen els modificadors (actuen sobre l'aparença dels objectes sense modificar la seva geometria original, fins que els apliques). He entès com s'apliquen les textures, com es tracten els objectes a mode de material per aplicar-hi a sobre la imatge que donarà sentit a l'objecte creat. Per posar en pràctica aquests coneixements nous, explicaré en aquest diari el procés seguit per crear una caixa simple en Blender.



En aquesta imatge podem veure la pantalla principal de Blender. En ella podem veure la càmera (esquerra), un cub (centre) i la llum (dreta). De moment, la càmera i la llum no m'interessen, les elimino.

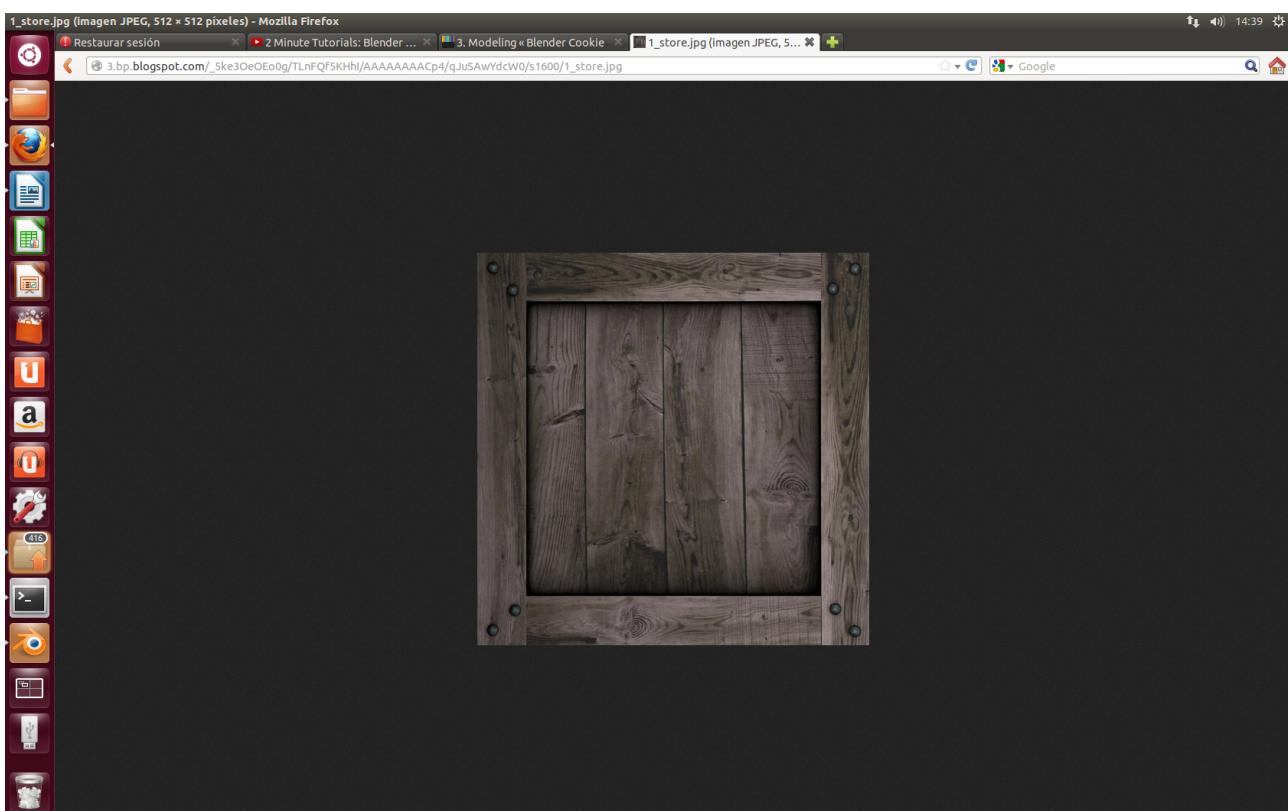


Podem veure en aquesta imatge com he realitzat uns talls al cub. Els exteriors seran les junes de la caixa, ara falta definir l'estructura “real” (les cares interiors s’han de tirar endins). Per fer això, utilitzaré les exclusions.

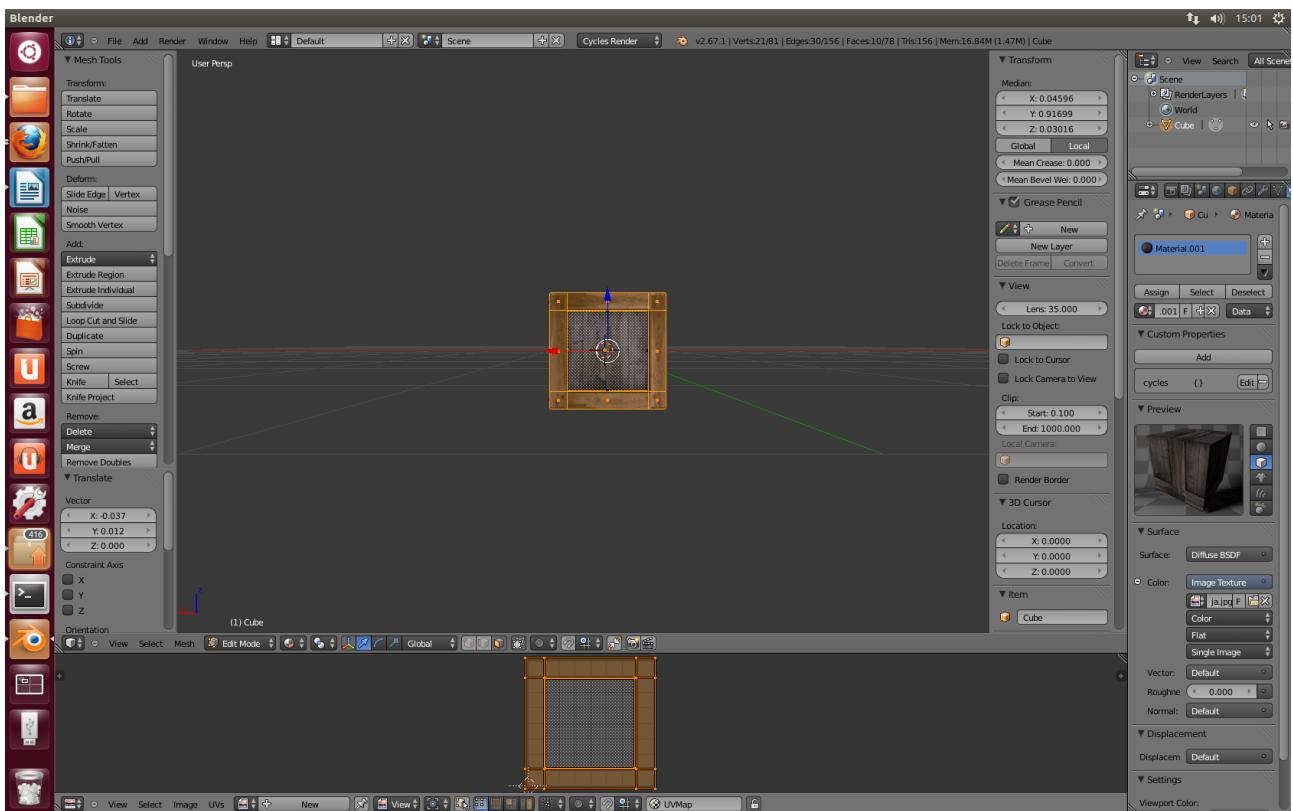


Aquí ja tenim definida l'estructura de la caixa. L'únic que em falta per fer, doncs, és obtenir una imatge que aplicar-li, per aconseguir així donar-li l'aspecte d'una caixa de veritat.

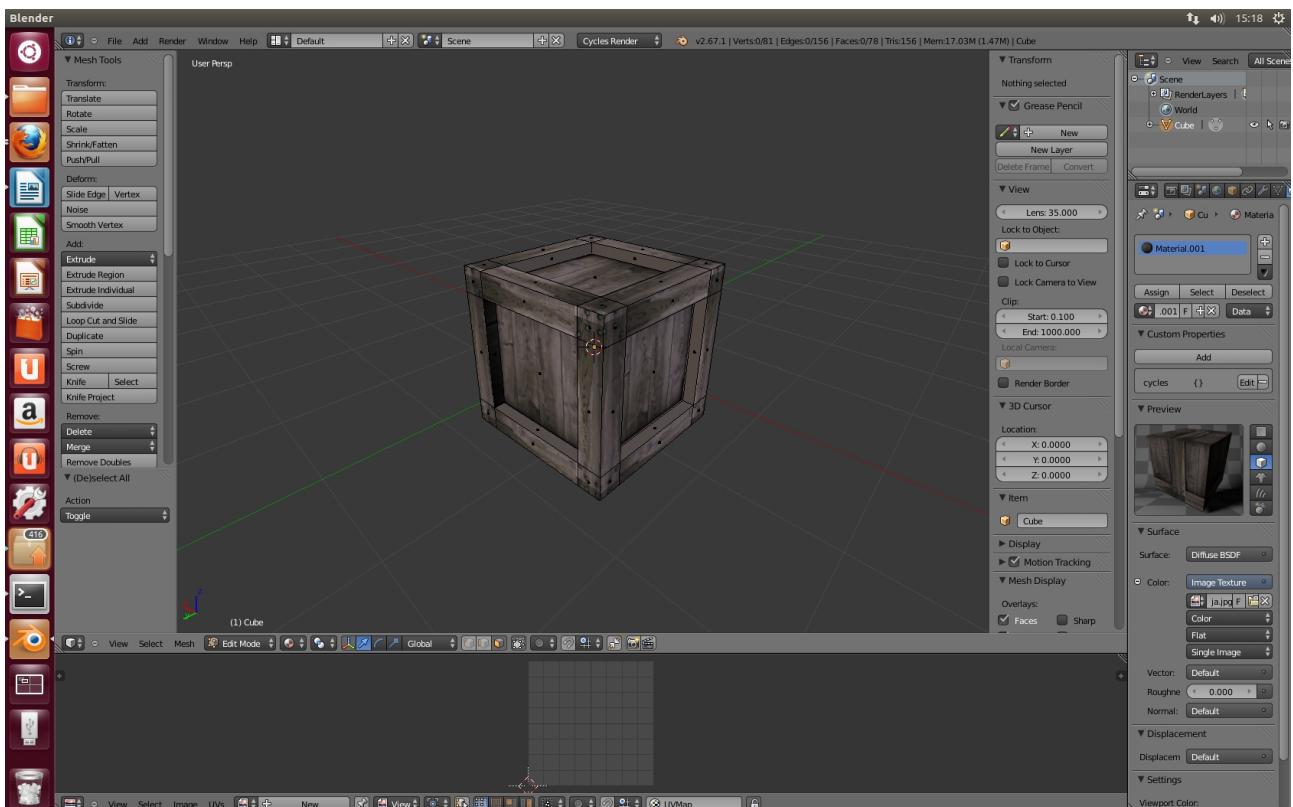
El disseny de les imatges que s'apliquen als materials realitzats és una part molt important a l'hora de crear un videojoc. Hem de tenir en compte que no només val amb crear una bona estructura, sinó que la imatge hi ha de ser acord. Per tant, és necessària una combinació eficient tant de modelatge com de disseny de la imatge per donar realisme a un objecte. Com que aquesta és una part lenta de la creació d'un objecte, per acabar la caixa d'aquesta pràctica buscarem per Internet una imatge acord amb la seva estructura. Aquesta és amb la que jo m'he quedat:



Procedeixo doncs ara a aplicar la imatge a la meva estructura.

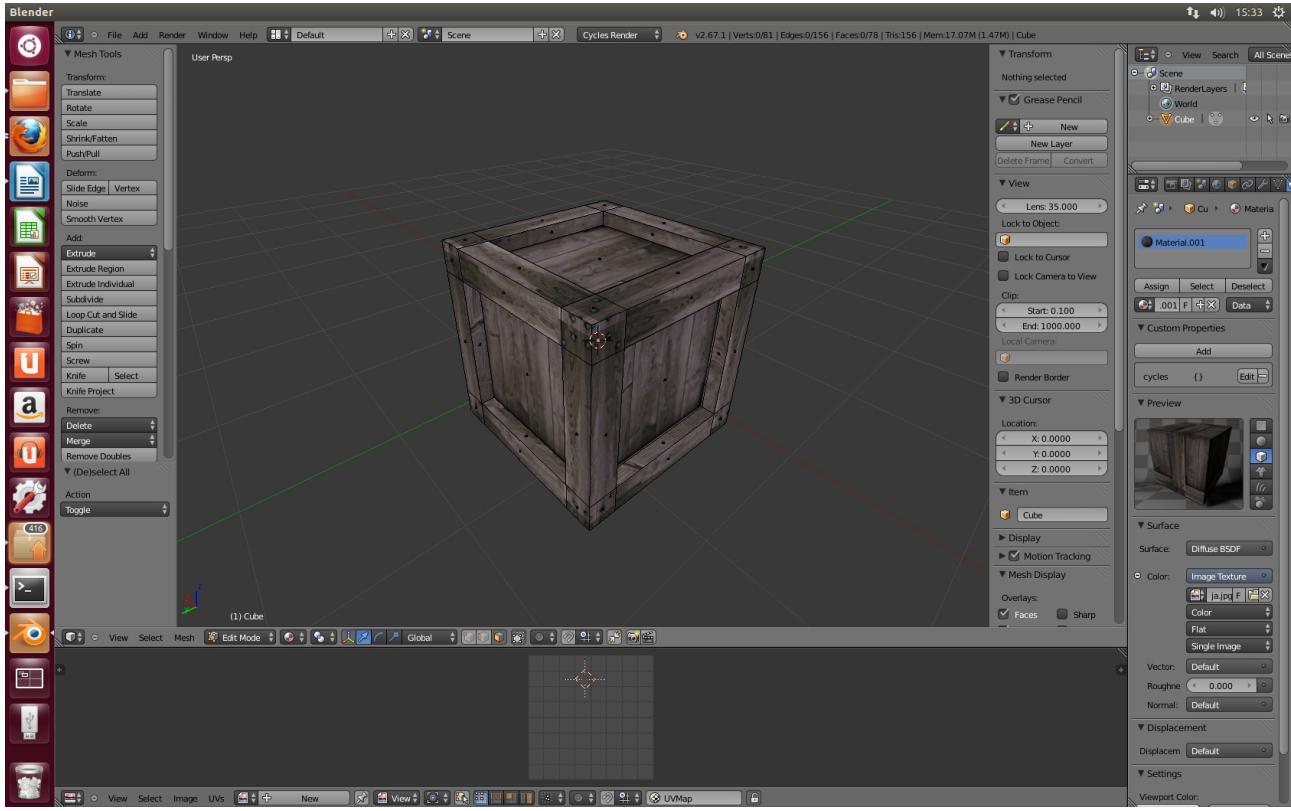


Podem veure a la pantalla com estic aplicant la imatge sobre l'estructura. Per fer-ho, la he projectat des d'un punt de vista perpendicular al pla al que l'aplico, ja que si no ho hagués fet així la imatge quedaria torta. S'ha de repetir aquest procés per a totes les cares.



La caixa està pràcticament acabada. He aplicat la imatge a totes les cares, amb cura de que quedés acord amb l'estructura que havia realitzat prèviament. Ara, l'únic que falta és

polir detalls. Si s'observa amb atenció, es por veure que les juntes de la caixa no tenen una imatge projectada per dins, són només de color marró. Com aquest tros és molt petit, no cal preocupar-se perquè la projecció quedí acord amb res, tan sols s'agafa un fragment de la imatge creïble i s'hi projecta. Procedeixo doncs a fer això.

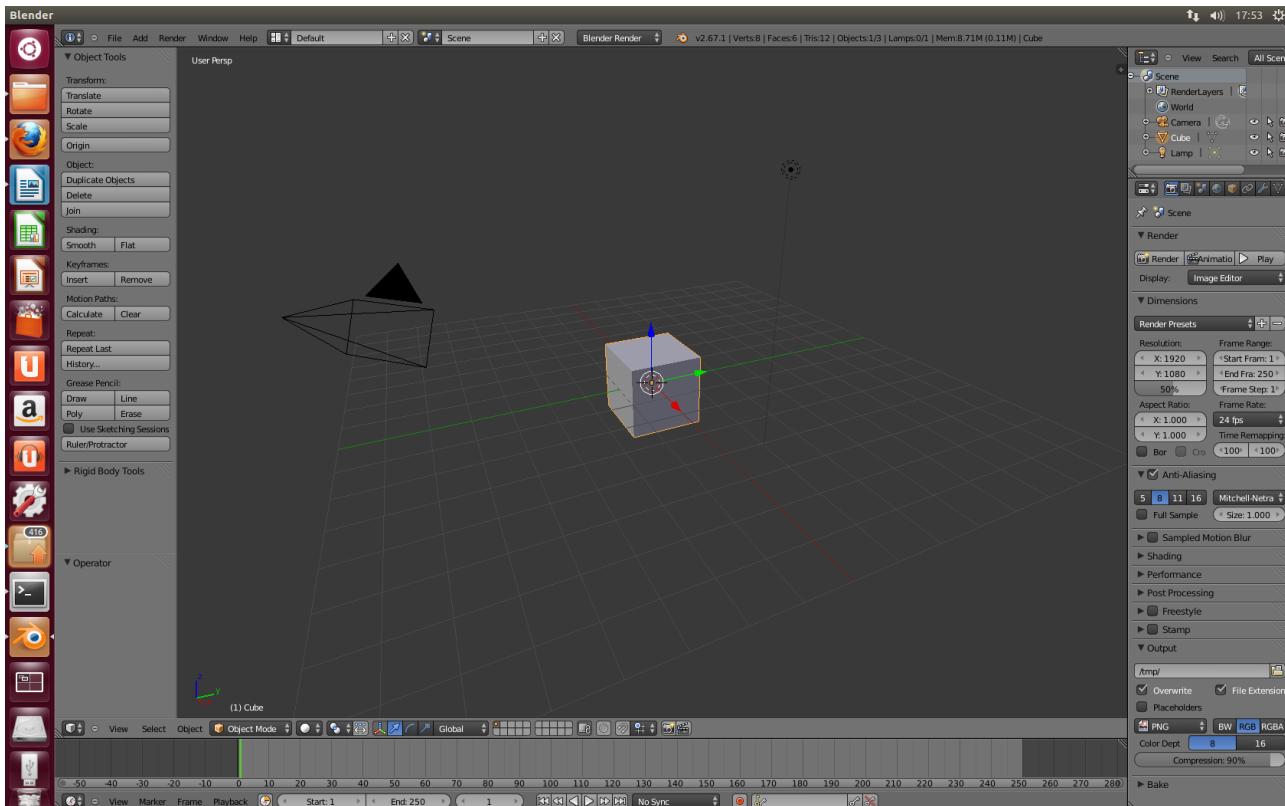


Ara sí, la imatge està totalment acabada. Aquí finalitza doncs la meva primera pràctica de Blender.

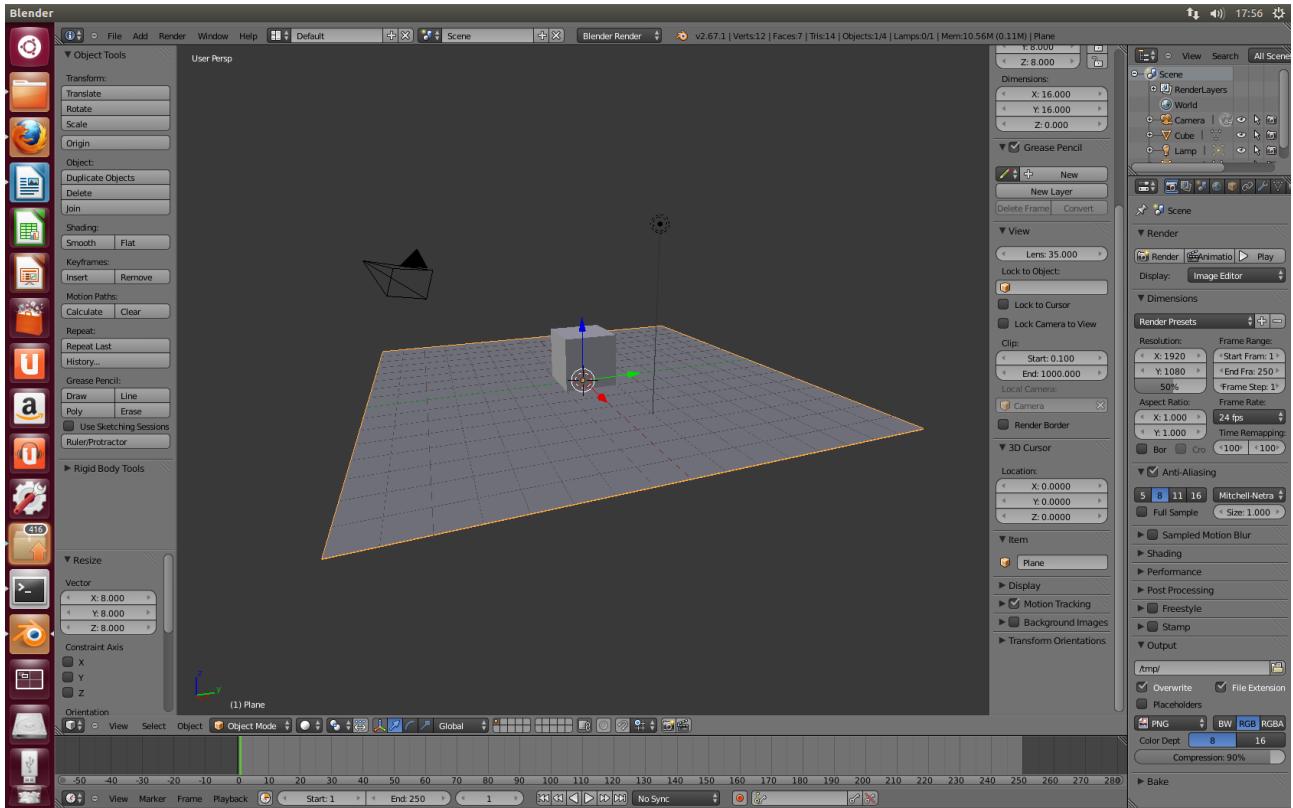
Fase 2: Game Engine de Blender.

- **Pràctica 1: Moviment d'un cub sobre el pla mitjançant el teclat** (*Font d'aprenentatge: <http://www.youtube.com/watch?v=lJRhwDmteI0>*)

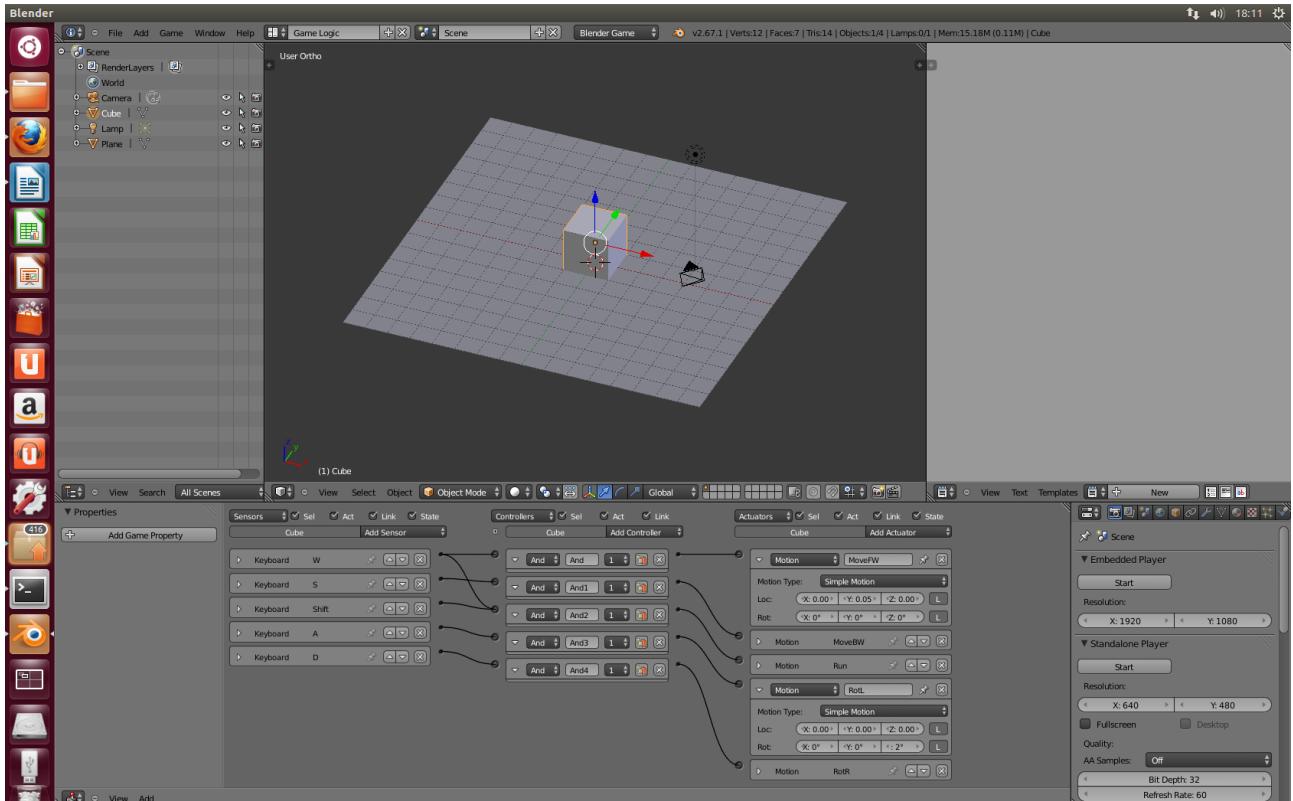
L'objectiu d'aquesta pràctica és aprendre les funcions bàsiques del Game Engine de Blender i veure com s'apliquen als objectes prèviament creats. Crearé uns connectors lògics bàsics que no requereixen programació d'scripts i els aplicaré sobre un entorn simple prèviament creat.



Com a la pràctica 1, podem observar la pantalla inicial de Blender. En aquest cas m'interessa conservar la càmera i la llum, ja que són necessàries pel joc. Em disposo a crear el pla que farà de terra per on es mourà el cub.



He desplaçat el cub a Z=1 per tal de que la seva base estigui alineada amb la graella. A més, he creat un pla en la posició de la graella que faré servir de terra. Un cop ja està creat l'entorn, passo a crear la lògica. Per fer això, canvio Blender Render per Blender Game.



Ja està creada la lògica del “joc”. A l’interfície inferior podem veure totes les ordres. En aquest cas, són molt simples, ja que són ordres que s’activen pel teclat. Veiem doncs com les tecles W i S s’utilitzen per avançar i retrocedir (la visibilitat de la imatge queda molt reduïda si obro totes les finestres, però les ordres donades a MoveBW són les mateixes que

a MoveFW amb el signe de Y canviat), A i D s'utilitzen per rotar el cub (passa el mateix que amb W i S) i, en aquest cas, l'únic una mica “excepcional” és la utilització del shift com a modificador de velocitat. En un joc, podríem utilitzar-la com a tecla per córrer, però en cap cas volem que només en prémer shift el nostre personatge avanci. Per tant, utilitzo el controller “And” per dir-li a l'ordinador que l'ordre només serà vàlida en el cas de que W i Shift estiguin premuts a la vegada. L'actuator “Run” té la velocitat de MoveFW doblada. Podríem aplicar el mateix per MoveBW, però no té massa sentit córrer cap enrere. Això ja és un tema que va a gustos, perfectament podríem assignar-hi una velocitat més ràpida que la de MoveBW però més lenta que la de Run. És completament opcional.

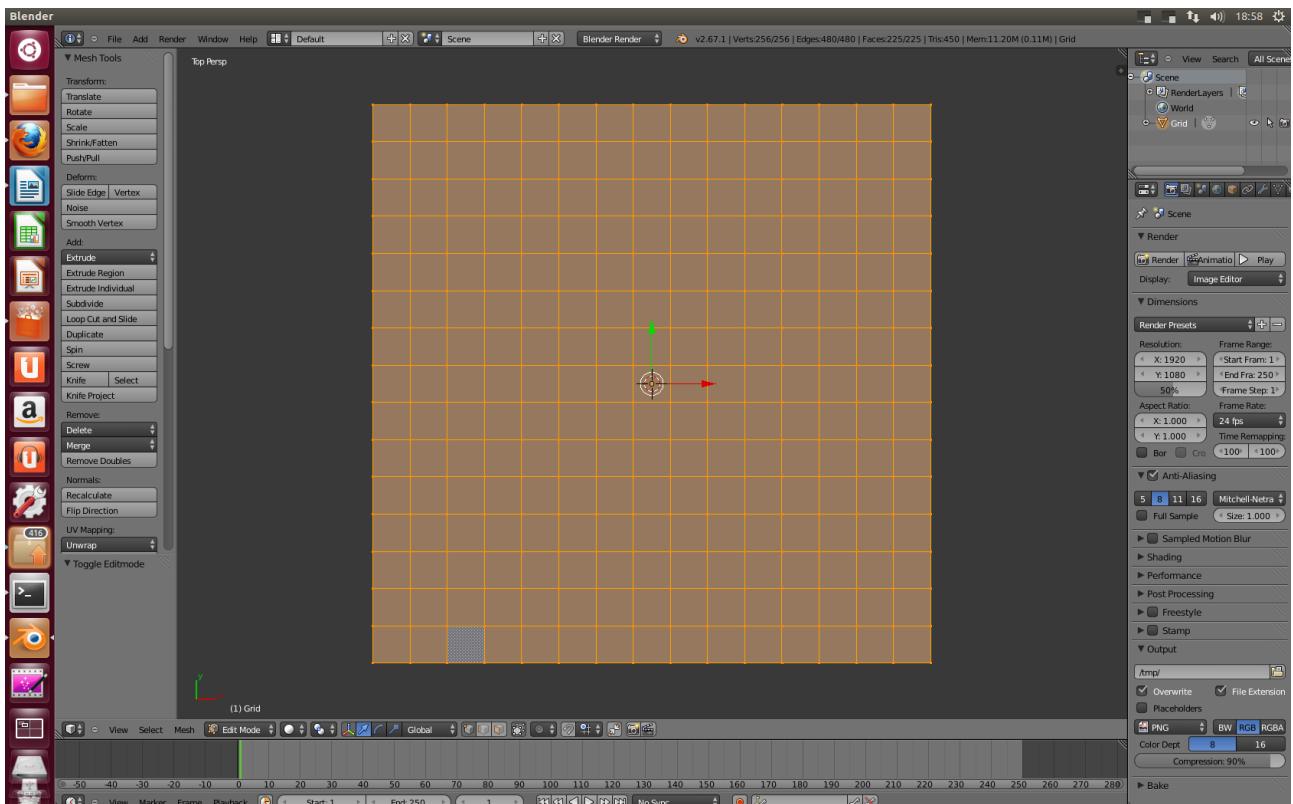
Bé doncs, no queda res més a fer. Les ordres estan donades i si premem P podem veure com el cub es mou a la perfecció.

Les conclusions d'aquesta pràctica són clares. Ha estat una molt bona introducció als connectors lògics de Blender. He entès com crear moviment i l'aplicació de la física sobre els plans. Però el primer repte de veritat ve a continuació.

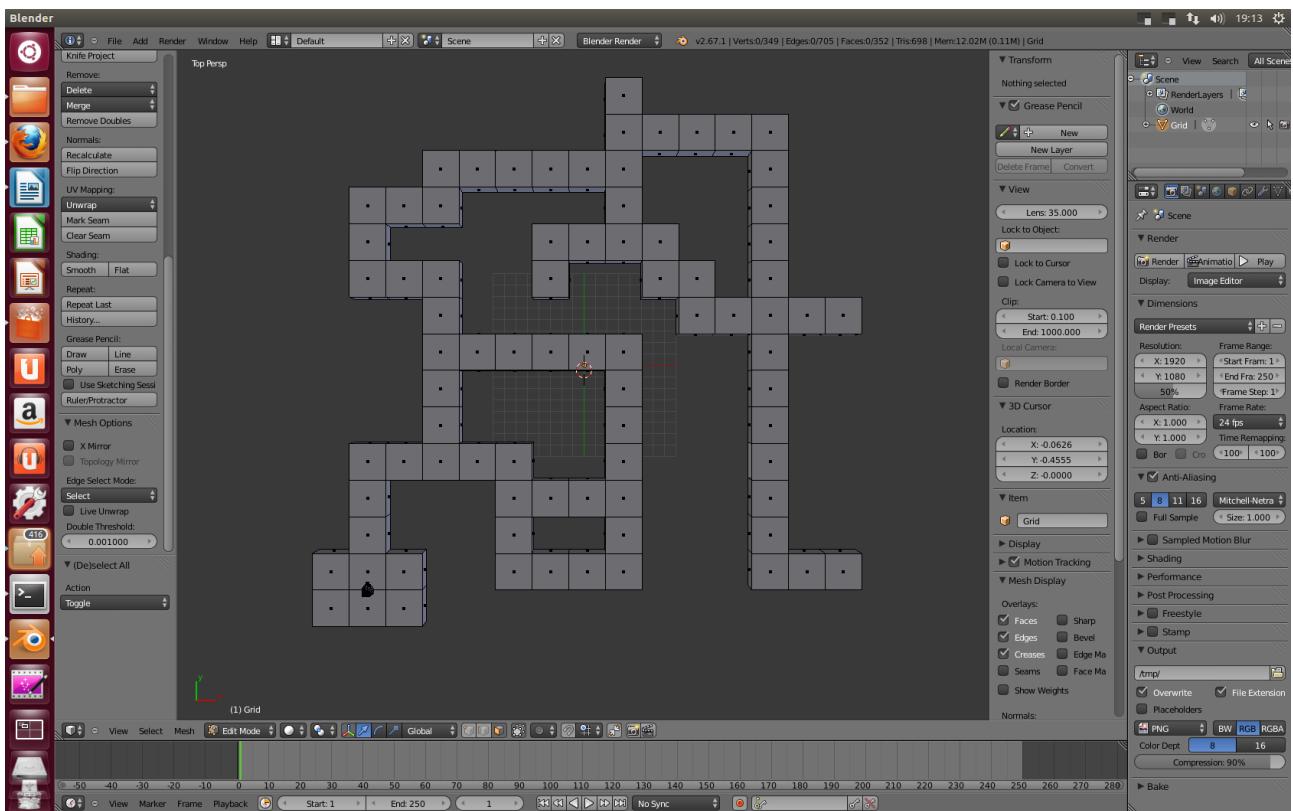
- **Pràctica 2: Creació d'un joc simple: Laberint** (*Font d'aprenentatge: http://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro/An_aMAZEing_game_engine_tutorial*)

L'objectiu d'aquest tutorial és aprendre a fer un joc amb les característiques clàssiques d'aquests (objectius, possibilitat de fallar, final...), utilitzant la lògica de Blender i, per tant, encara sense programar ordres avançades. Aquest joc contindrà diversos nivells, el que m'obligarà a aprendre com crear connectors lògics més complexos que tindran relació amb l'entorn i el metagame.

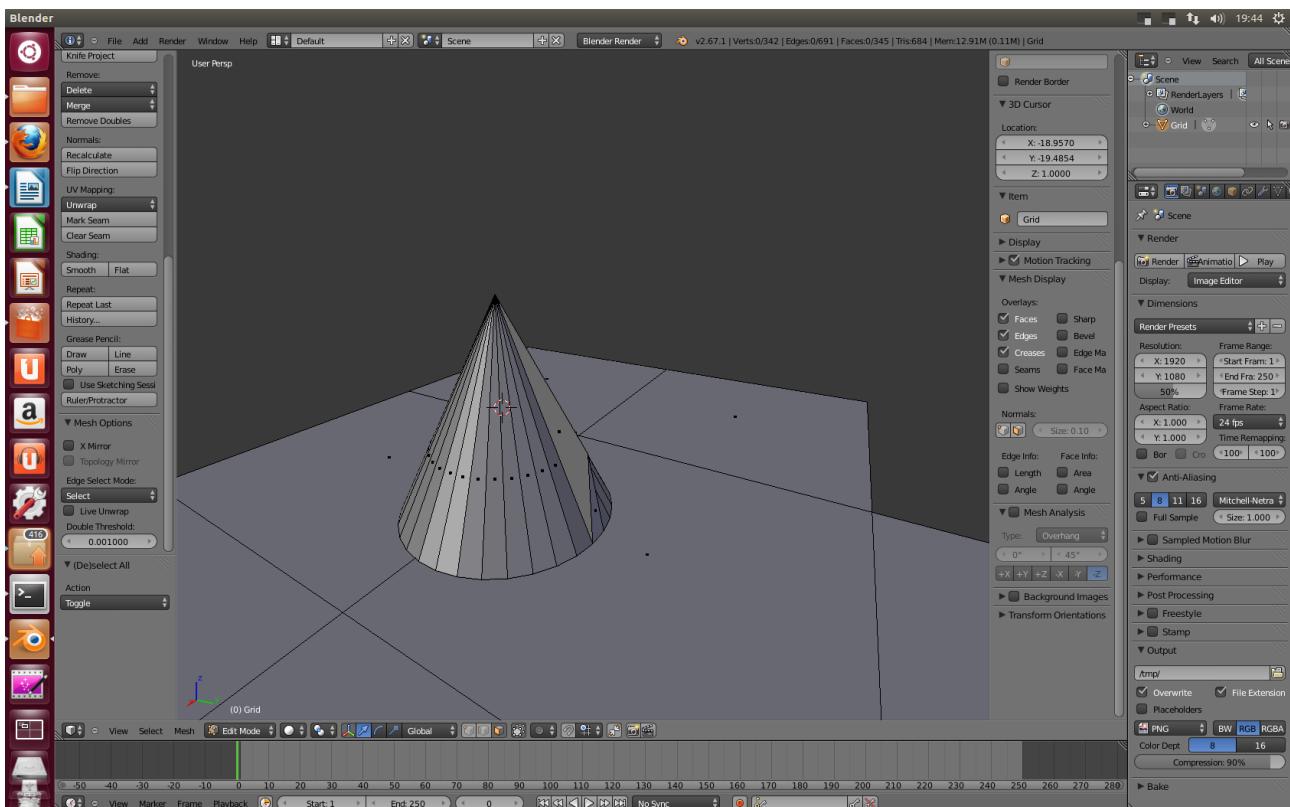
Interfície principal de Blender. En aquesta pràctica ho esborraré tot ja que el que vull crear és completament diferent.



En aquesta imatge es pot observar la graella que em farà de guia a l'hora de crear el laberint. Seleccioño les parts de les que constarà el meu laberint i les excloc de la graella, esborro la original i ja tinc la superficie del laberint. Hi faig una extrusió per crear-hi alçada.



Aquest és el meu laberint vist des de dalt. En el punt marcat en negre crearé el personatge que controlarà el jugador. Ara per ara, no m'interessa crear un personatge perfectament detallat i realista, ja que comportaria un munt de feina innecessària per l'objectiu real d'aquest joc i, a més, després hauria de crear animacions. En conclusió, no me'n sortiria, així que em basaré en la suggerència del tutorial i faré un con amb un indicador de direcció (una punxa). Podrem apreciar la seva magnificència a la següent imatge.

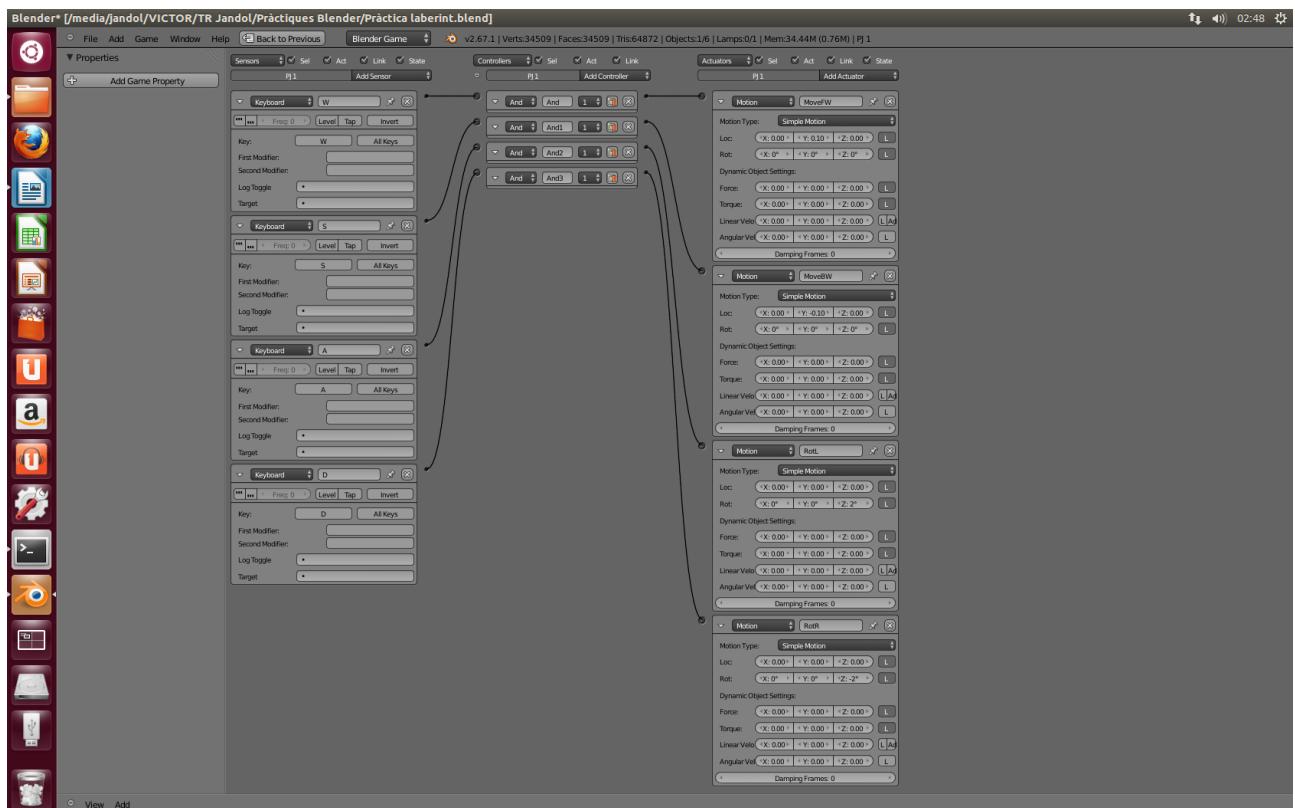


Un cop creat l'entorn, queda crear tota la lògica. Aquesta és una part complexa i si hagués de fer captures de pantalla de com ho vaig realitzant, ompliria 10 pàgines com a mínim. Per tant, explicaré de forma escrita i ordenada els passos que he seguit per crear, en primer lloc, el moviment i la física, en segon lloc les dificultats i en tercer lloc la meta i les reaccions del mapa en arribar-hi. Només faré captures de pantalla dels resultats finals d'aquests passos.

Moviment i física

La creació del moviment i la física és bastant simple i, de fet, ja m'hi havia familiaritzat a la pràctica anterior. Afegixo sobre la lògica del personatge (el con) uns sensors que reaccionin sobre el teclat (WASD), i uns actuadors motrius. Per W i S assigno modificadors de moviment en Y, positius i negatius, respectivament, mentre que per A i D assigno modificadors de rotació sobre l'eix Z. Ja tenim tot el moviment necessari creat.

La física necessària per aquest entorn és tant fàcil com donar-li dinamisme al personatge. Per tant, li dono aquesta propietat al con i li assigno una massa. Així, quan el jugador es surti de la superfície, caurà i haurà de tornar a començar (aquesta ordre l'assignarem més endavant). Ja tenim creada la física.



Interfície de la lògica del con (personatge).

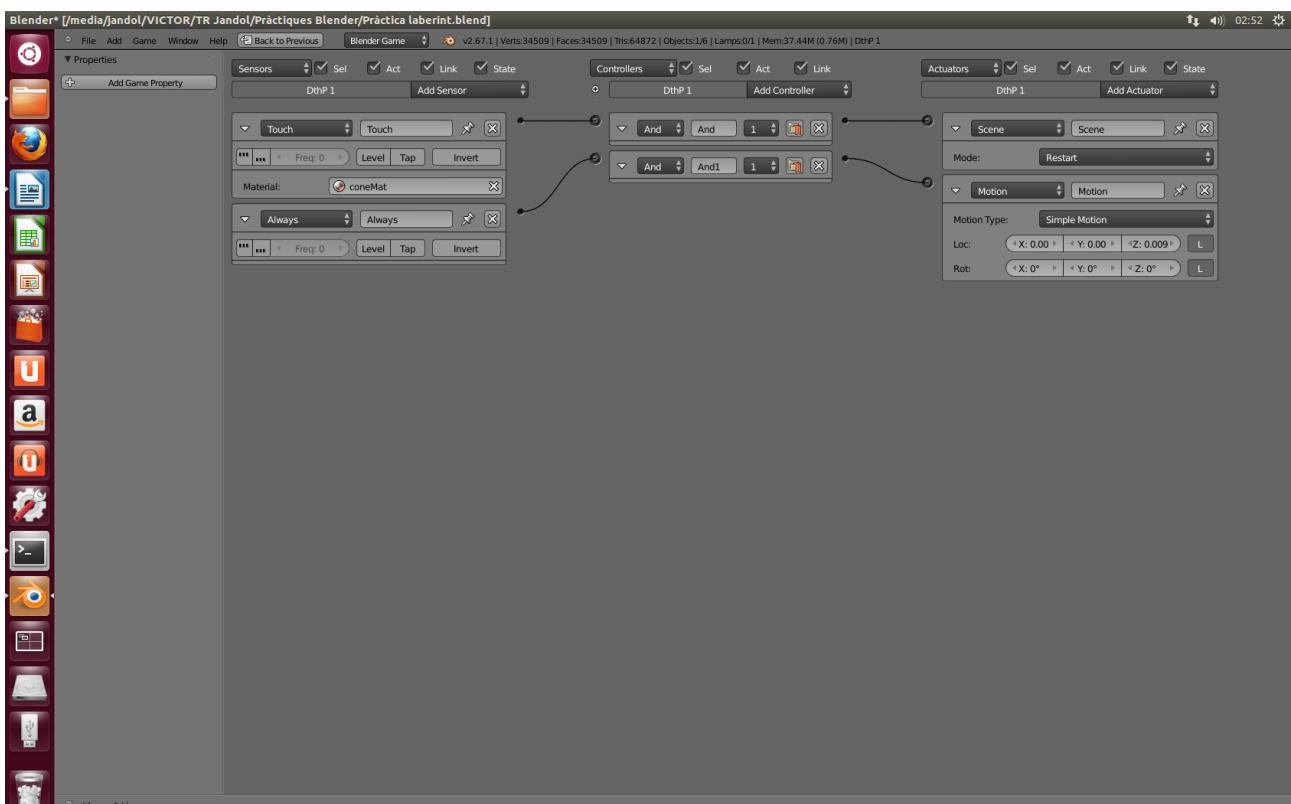
Dificultats (errors i temps)

Per fer un joc estimulant i donar-li un sentit, és necessari que aparegui alguna dificultat. Si no, el jugador s'avorriria ja que la victòria estaria regalada. Ara explicaré com he implementat la lògica que genera la “mort” del jugador al sortir del mapa i un sistema de temps que fa que el nivell s'hagi de superar en un temps X.

Com a l'apartat anterior li hem donat propietats físiques al personatge, aquest caurà al sortir del recorregut. Però, si a sota del mapa no hi hagués res, el personatge cauria en un pou sense fons i hauríem de tancar el joc i tornar-lo a obrir nosaltres, cosa que comportaria

tornar a començar des del primer nivell quan potser l'ordre de mort que volíem assignar feia tornar a començar des del nivell on el jugador havia mort. És més, encara que a sota del recorregut hi hagués una superfície, si no li assignem una propietat lògica, el cub es quedaria atrapat i estaríem en les mateixes. Per tant, ens trobem amb que hem d'idear una forma de que quan el personatge caigui, el joc ho detecti i el nivell torni a començar. Jo ho he fet creant un pla sota el mapa, i assignant-li al con un material. He creat un connector lògic de col·lisió que relaciona l'impacte de l'objecte “pla” sobre el material del con. Aquesta relació activa un actuador que fa reiniciar el nivell.

Per afegir-hi una dificultat extra, he implementat un sistema primitiu de temps. Aquest és ben senzill: el terra puja. Per fer això, és tan simple com assignar-li al terra una ordre de moviment ascendent (en Z), rumiem una mica el temps que volem posar-li al nivell (és molt important tenir en compte que el creador coneix la solució del laberint, però el jugador no). Pot ser una bona solució utilitzar *testers* per veure si el joc és o bé massa fàcil o massa difícil), i fem un càlcul amb el moviment que fa el pla.



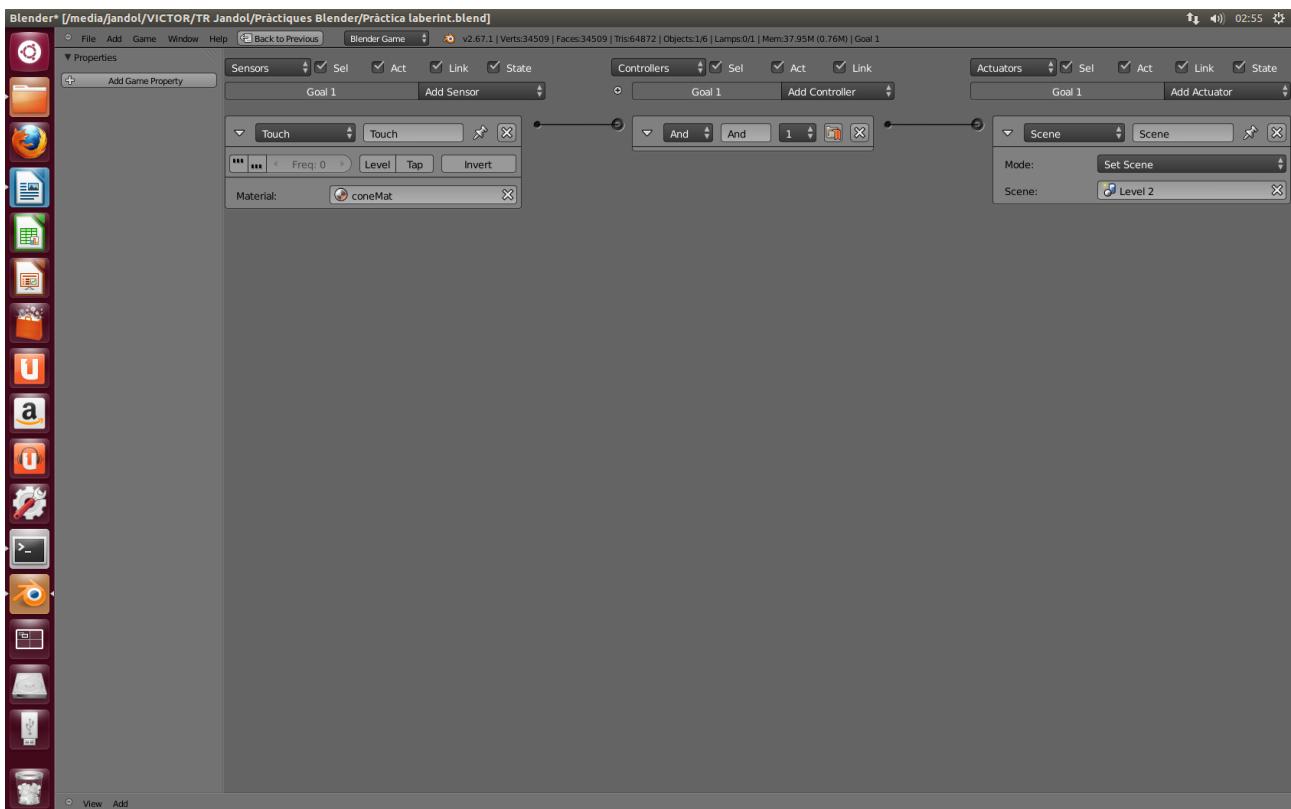
Observem com el pla té dues ordres assignades: la de pujar progressivament i la de reiniciar el nivell en cas de tocar al personatge

La meta i el canvi de nivell

Perquè el joc tingui sentit, és necessari un objectiu. En aquest cas, l'objectiu que busquem és una meta, un lloc al que arribi el personatge. Per la meva meta he creat un rectangle de poc gruix de color groc (destaca sobre l'entorn i és comprensible pel jugador). Ara cal crear-li la lògica.

L'activador de la meta és per contacte, igual que el del “pla de la mort” de sota el laberint. Però en aquest cas canvia l'actuador: l'ordre és la de canviar d'escena. Una escena és una espècie de “món” on hi poden haver diversos objectes, per tant, prenem les escenes com a nivells pel joc. Així, la meta de cada escena portarà a l'escena següent, excepte l'última, que

donarà al joc l'ordre de matar-se (s'ha de tenir en compte en tot moment que un joc és un programa).



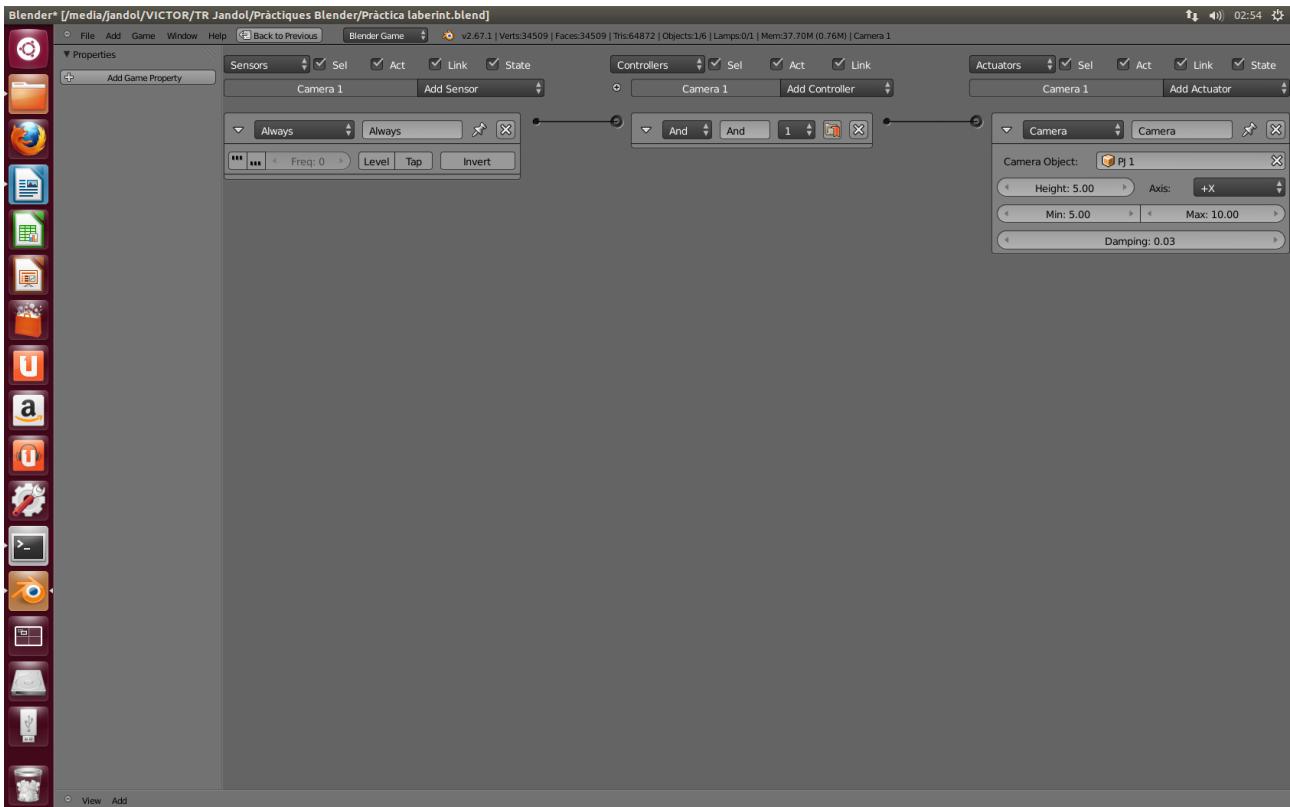
Lògica de la meta 1. Dóna l'ordre de passar al nivell 2.

Ja tenim la lògica del joc acabada, però encara queda una part fonamental abans de fer els retocs finals al joc: la càmera.

Lògica de la càmera

Hem d'entendre la càmera com els ulls del jugador. Si el nostre joc és en primera persona, el personatge és la càmera, i només crearíem un objecte que li faria de suport per a les col·lisions. Aquest objecte seria invisible (excepte en els FPS en els que és possible veure les cames al personatge), ja que només l'estaríem utilitzant per donar sensació al videojoc, però seria un prisma sense cap textura. Per altra banda, si és en tercera persona, la càmera segueix sempre de prop al personatge. En aquest cas, m'interessa fer una càmera en tercera persona, ja que vull veure com es mou el personatge per evitar que caigui.

En primer lloc, creo una càmera, i li assigno una posició inicial. La col·locaré aprop del con, ja que si no l'únic que aconseguiria seria que al principi del joc la càmera fes tot un viatge per la pantalla per col·locar-se darrere del con. Ara queda fer-li entendre que ha de seguir sempre a l'objecte personatge (el con). Com he dit, l'ha de seguir **sempre**, per tant l'activador serà **always**. Utilitzo l'actuador càmera i l'assigno a l'objecte personatge. A continuació determino l'altura que tindrà respecte el con, i els mínims i màxims moviments per actuar.



Lògica de la càmera

Ara sí, la lògica fonamental ja està completament acabada. Tenim un joc amb reaccions físiques, un objectiu i unes dificultats que donaran la gràcia al joc. Només em queda fer la decoració final, per fer el joc més atractiu, i crear tants nivells com vulgui. Per aquesta pràctica he creat tres nivells, de dificultat progressiva, he canviat els colors dels mapes de blau a vermell progressiu (com un indicador de l'augment de la dificultat) i he afegit música de fons per cada pista.

Bé doncs, fins aquí el meu primer videojoc en Blender. Puc treure molt bones conclusions: he après a crear relacions lògiques més avançades entre objectes, m'he vist immers en el concepte de *game loop*, he treballat en diversos nivells, he après a col·locar la càmera... En resum, ha sigut una pràctica en la que he après moltíssim i com a valoració personal puc dir que ja m'estic sorprendent de la velocitat amb la que m'estic endinsant en un món que des de fa uns pocs dies em semblava impenetrable.

Fase 3: Programació amb Python (*Fonts d'aprenentatge:*

<http://pyspanishdoc.sourceforge.net/tut/node8.html>

<http://pycartagena.pbworks.com/w/page/37765255/M%C3%B3dulos>

<http://docs.python.org/dev/reference/import.html>

Per aprendre a fer servir Python, he utilitzat bàsicament una recerca d'informació molt àmplia, i he estat programant tota mena d'ordres senzilles. He preguntat a persones entenedores del tema perquè m'assistissin a l'hora d'acabar d'assimilar els conceptes que havia trobat en línia i, després d'un llarg procés d'aprenentatge, he obtingut les següents conclusions referents al que hauré de tenir en compte durant la programació del meu videojoc:

- **Python és un codi basat en objectes:** Això vol dir que podem assignar qualsevol variable a un valor, funció, estructura condicional, etcètera. Per exemple, 'a = 2', però també 'holà = 2'. És important remarcar també que Python **fa distinció entre majúscules i minúscules**, per tant, no és la mateixa variable 'a' que 'A'.
- **Cada unitat singular de codi és anomenada "mòdul".** Dins d'aquests mòduls, es defineixen variables a les que es pot accedir des d'altres mòduls mitjançant l'ordre 'import i, més específica, 'from x import y'. Això és útil per esquematitzar la programació i fer-la més entenedora i fàcil a l'hora d'haver de treballar amb diversos factors que no comparteixen el mateix temps d'execució. Si necessitem que un mòdul ens determini una variable al moment $t=0$ i una altra al moment $t=1$, i després ens ha de fer uns càlculs en funció d'aquesta variable, repartir la feina és una solució més que viable. Crear un mòdul secundari que obtingui una variable mitjançant un procés independent del que utilitzarà un mòdul principal (que importarà aquesta variable) és molt més pràctic que crear un sol mòdul encarregat de fer-ho tot, doncs s'haurien de determinar intervals de funcionament.
- **Blender, i totes les interfícies gràfiques sobre les que es pot introduir codi, tenen uns mòduls predefinits per possibilitar les referències a objectes de la interfície des de la programació, totalment aliena.** Si això no fos així, seria impossible referir-nos a un objecte prèviament dissenyat, doncs Python i Blender són dos programes diferents sense cap relació. El mòdul necessari a importar en Python sempre que es vulgui fer una referència a un objecte o propietat de l'entorn gràfic és l'anomenat 'bge.py'. Per tant, sempre que es doni aquest cas, el mòdul haurà de començar amb la línia 'import bge'.

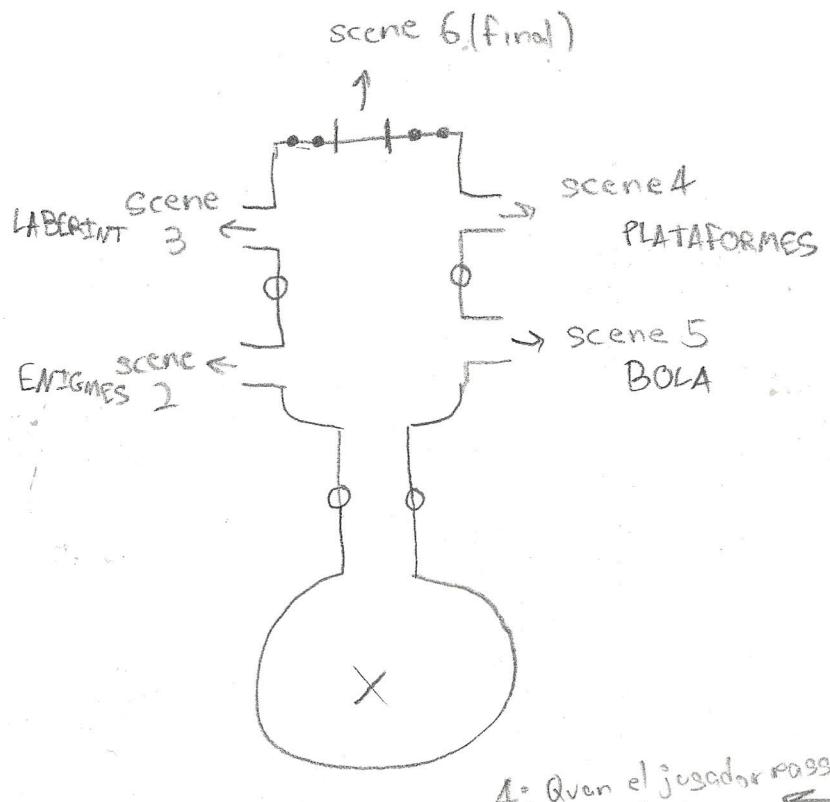
Això és el que s'ha de tenir en compte sempre a l'hora de programar amb Python. La part complicada és que, com a llenguatge de programació, té la seva sintaxi, que he hagut d'aprendre a base de provar i equivocar-me un cop i un altre. Per sort, cada cop que es comet un error sintàctic, al executar el mòdul i fallar s'indica la línia on s'ha comès l'error, el que facilita molt més la identificació del problema i la seva solució.

En aquest punt de la recerca només em queda plantejar-me el format del videojoc que haig de realitzar, i fer un plantejament previ al seu disseny. Aquest serà l'últim punt del meu diari de recerca, mitjançant la confecció del qual he obtingut les competències necessàries abans de començar amb el desenvolupament del meu videojoc.

Fase 4: Plantejament del videojoc (plànols)

Deixant de banda la tria de la temàtica i l'explicació detallada dels factors lògics i físics del videojoc (cosa que constarà a la part del desenvolupament del videojoc de la memòria), dedicaré aquesta part del diari a mostrar els plànols inicials que he realitzat per a la confecció del mateix. Hi ha algunes sales que no presenten disseny gràfic, degut a que és més fàcil la modelització directa des de Blender que a mà, però totes tenen determinades la física dels objectes i les interaccions que s'esperaven en inici. Aquests plànols seran un bon punt per fer una comparació entre els resultats esperats i els obtinguts en acabar el desenvolupament del videojoc.

SALA PRINCIPAL (scene 1)



4: Quan el jugador passa a la
scène 6, el cronòmetre
es para (si hi ha cronò).

El jugador apareix a X i se li dóna
una introducció en forma de text que
envolta la sala circular.

- Objectes dinàmics: Jugador
- Objectes amb propietats físiques: Jugador
- Reaccions de l'entorn:
 - 1: Cada porta canvia la scena quan s'interactua amb ella (per donar "credibilitat" al joc les portes seràn teleports amb una placa al terra que tindrà la propietat del canvi d'escena).
 - 2: La porta final només s'obrirà quan s'hagin superat les proves de les scènes 2, 3, 4 i 5.
 - 3: Uns indicadors (•) que en un principi seràn de color negre canviaran de color quan una sala es completa (cada indicador anirà associat a una sala).
- Il·luminació: A sobre de X hi haurà un focus de llum blanca que prové "de l'exterior". Aquest focus il·luminarà la sala circular, mentre que la resta de la sala estarà il·luminada per torxes (○).

Idees per estimular la competitivitat:

- Contador d'errors/morts: Forçarà al jugador a prendre's el joc seriosament en sales com la dels enigmes. S'evita el "probar per probar" i s'incita a intentar millorar.
- Cronòmetre: Fa l'efecte contrari al contador, evita que la gent s'adormi, i la incita a resoldre les proves amb agilitat.
- Puntuació final: Després del beta-testing, es poden fer mitjanes d'errors i temps i assignar unes "notes" a les puntuacions (exemple darrere).

Puntuació final: Exemple

63

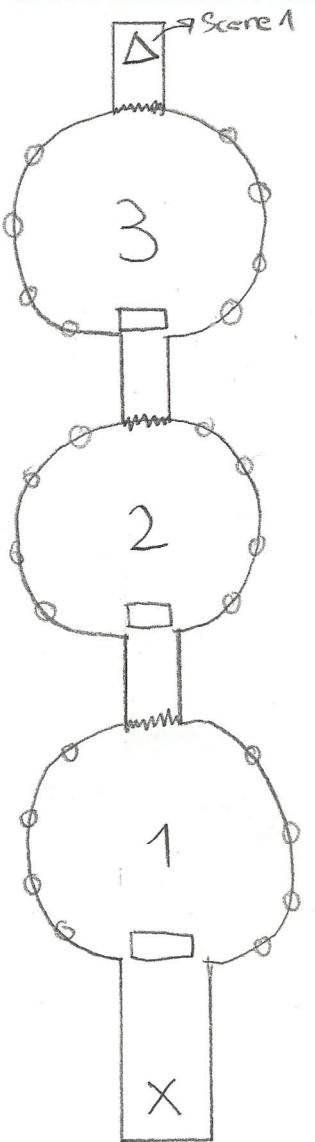
Després del beta-testing, hem obtingut uns resultats mitjans de 20 errors i 30 minuts. Podriem fer les assignacions següents:

QUALIF.	ERRORS	TEMPS
SSS	<5	<15m
SS	5-10	15-20m
S	10-15	20-25m
A	15-20	25-30m
B	20-25	30-35m
C	25-30	35-40m
D	30-35	40-45m
E	35-40	45-50m
F	>40	>50m

... I la qualificació total
seria la mitjana de les dues
truncant els decimals (si tens
SSS i SS, has fet SS).

També s'ha de tenir en compte
que la mitjana dels resultats
no vol dir res per si sola, s'han de
comprovar els resultats individuals
per veure si el repte és massa complicat/

SALA DELS ENIGMES (scène 2)



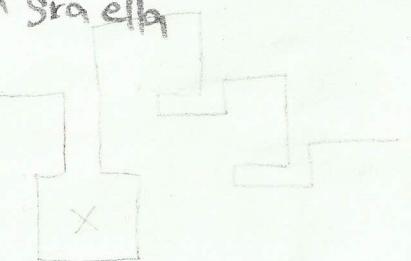
La Sala dels Enigmes presentarà tres enigmes a resoldre pel jugador, de dificultat progressiva.

- Dificultat = nombre d'objectes seleccionats necessaris per passar al següent enigma (enigma 1: 1 obj., enigma 2: 2 obj., enigma 3: 3 obj.)
- □ : Les portes s'obren quan l'enigma és resolt.
- □ : Placa al terra (del mateix aspecte que la resta de la sala) que s'activa al contacte amb el jugador. Aquest contacte provocarà l'activació d'un so (si explicaria l'enigma). Es considera la idea de que, a més, l'enigma aparegui escrit a la paret.
- ○ : Il·luminació (torxes).
- Els objectes estaran dispersos sobre un mostrador (no amagats)
- Siemetrà un so de confirmació quan es tria l'objecte correcte (en el cas de l'enigma 1 no importa ja que la porta s'obre amb un objecte) perquè el jugador sapiga que va bé als enigmes 2 i 3.
- Si es tria un objecte incorrecte, es suma un error i pels enigmes 2 i 3 es desselecciona/en els objectes correctes en cas que hi haguessis fet una selecció correcta prèvia (els objectes si han de triar en un ordre concret).
- Δ: Transportador de volta a X en scene 1 (funciona al tocar-lo). La seva activació comporta també el canvi de color d'un dels indicadors de la porta final de Scene 1.

SALA DEL LABERINT (scene 3)

(SALA DISSENYADA DIRECTAMENT
SOBRE BLENDER)

L'és absurd dissenyar-la aquí
quan es crea mitjançant
una graella



A la Sala del Laberint no hi haurà possibilitat d'error (per això el plantejament de l'implementació del sistema de temps) i puntuacions)

- Dins del laberint, el jugador haurà de trobar la sortida; Així es posarà a prova la seva capacitat d'orientació espacial.
- Il·luminació; Torxes distribuïdes pel laberint.
- Sortida: Teleportador de volta a X en Scene 1. Activa també un indicador de la porta final.

SALA DE LES PLATAFORMES (Scene 4)

99

La sala de les plataformes o "parkour" obligarà al jugador a arribar superant obstacles mitjançant gran quantitat de salts. Aquests presentaran dificultats tals com llunyania entre plataformes, plataformes molt petites, camins estrets...

(BLENDER)

- Objectiu: Arribar al final (el camí no és lòs, sinó complicat. No és un laberint).
- Error: Caure. Reinicia la scene (contacte amb fons).
- Il·luminació: Torxes, èlava que ilumini?
- Arribar al final transporta al jugador a scene 1(x) i activa un dels indicadors de la porta final.

SALA DE LA BOLA (scene 5)

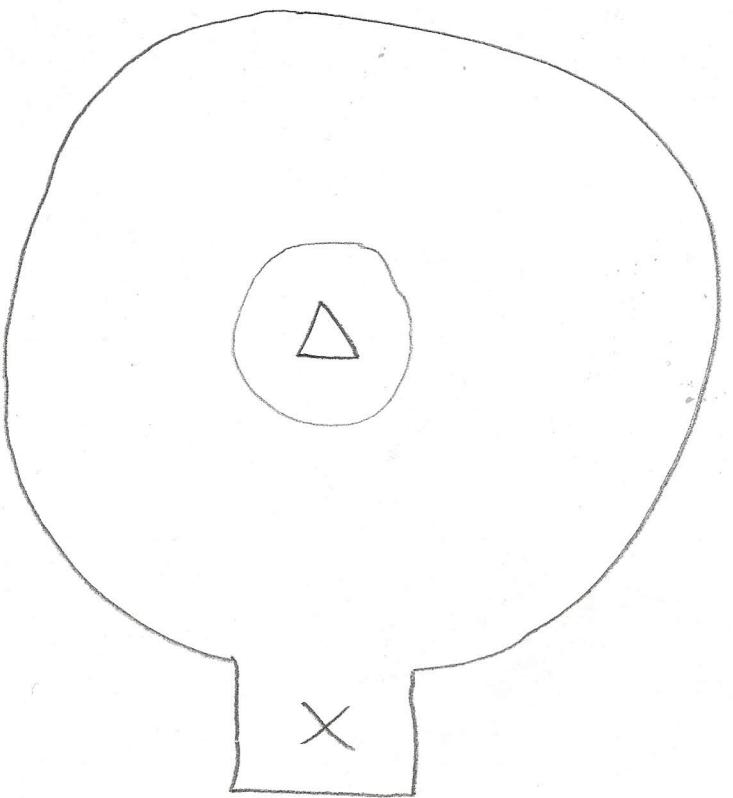
A la sala de la bola, el jugador ha de moure una bola sobre un pla evitant que caigui, amb l'objectiu de portar-la a un recipient que està al final.

(BLENDER)

- Física: A diferència de les altres sales, en aquesta tenim un altre objecte amb propietats físiques: la bola.
- Comportament de la bola: Objecte rígid (es mou lluïvement segons com la moguem amb el PJ).
- Dificultat: Al pla per on desplaçem la bola hi haurà forats, i si ho d'evitar que la bola hi caigui.
- Reaccions de la sala:
 - 1: Col·lisió amb el "pla de la mort" (bola, jugador) → Restart scene
 - 2: La bola toca el recipient final → Porta del teleportador s'obre.
 - 3: Col·lisió amb el teleportador: Transporta al jugador a X (scene 1); activa un dels indicadors de la porta final.

SALA FINAL (scene 6)

89



A la sala final, el jugador només s'ha d'acostar al centre (que representarà una decoració pertinent) per tal d'acabar el joc.

- Disseny de la sala: La sala final és un cilindre obert per adalt, el que permetrà al jugador veure el cel i el sol, cosa que denota llibertat (les sales anteriors eren totes bastant fosques, o al menys no presentaven il·luminació natural)
- Il·luminació: Sol.
- Reaccions de la sala: △ acaba el joc, es mostra la puntuació final, un vídeo (fet amb Movie Maker, no sé d'on) i els crèdits (creador, música, agraïments, etc)