Treball de fi de carrera

PANG 3D

MEMÒRIA

Jorge Cabezas García
mailto:jorgecabezas@uoc.edu

TFC

INDEX DE LA MEMÒRIA

1. Descripció del projecte i objectius

El projecte consisteix en una versió 3D del famós joc **Pang**. El joc original va ser publicat per *Mitchell*, una companyia que ja no existeix, al 1989 en format *coin-op*[1]. Des de la seva data de publicació va supossar un gran èxit de públic i de vendes per la companyia, que en va treure algunes versions més.

Més informació sobre aquest joc es pot trobar a Arcade History Database© (*http://www.arcade-history.com/history_database.php?page=detail&id=1929*)

L'objectiu del projecte és, doncs, aconseguir una versió 3D d'aquest joc clàssic, intentant de conservar la jugabilitat de la versió en 2D.

1.1. Mecànica del joc

La lògica del joc consta dels següents elements: un personatge, unes bombolles que es divideixen quan el personatge les dispara un tret, una pantalla delimitada per tots els costats i alguns elements com escales i plataformes on el personatge pot pujar i mantenir-se o disparar-les. També hi apareixen alguns bonus (temps, punts), en trencar-se algunes bombolles.

El àrea de joc està delimitada per la pantalla en si (hi ha un petit mur que indica que no es pot passar més cap a la esquerra o la dreta).

L'objectiu del joc es trencar totes les bombolles en un temps màxim. Quan això s'aconsegueix, es pot passar a la següent pantalla.

El personatge utilitza un arma que dispara en sentit vertical, i que a mesura que va pujant, va construïnt una espècie de fil.lament. Les bombolles es trenquen en dues bombolles més petites en ser tocades per aquest fil.lament. Les bombolles no col.lisionen entre elles. Quan el tret arriba al sostre, el fil.lament desapareix.

En aquest gràfic podem veure com dispara el personatge principal:

1. *Coin-op* : és el nom que reben les màquines recreatives que incorporen videojocs i que no poden concedir premis (en contrast a aquelles màquines recreatives que si que en donen, les famoses escurabutxaques). Els primers videojocs van aparèixer en aquest format, però han entrat en un declivi constant des de la segona meitat dels anys 90 i cada vegada és més difícil trobar-ne novetats.

El joc també té altres elements, com rellotges de temps que apareixen en trencar algunes bombolles i que les paral.litzen durant un breu instant de temps, així com plataformes que es poden trencar en ser disparades.

1.2. Plataforma de desenvolupament i producció

Tant per al desenvolupament del joc, com per a plataforma de producció (plataforma en la que ha de executar-se el programa final), s'ha decidit d'utilitzar un ordinador compatible PC amb sistema operatiu Windows.

El projecte es desenvoluparà en C (amb Visual C++ 6.0) i utilitzarà openGL 1.0 o superior. Els requeriments mínims del sistema d'execució estàn encara per determinar, però una posible configuració mínima seria:

PC Pentium IV compatible o superior
128 MB de RAM
Tarjeta de Video 3D compatible openGL (Nvidia o ATI recomanat)
Al menys 3 MB de espai en HD
Sistema operatiu Windows XP

1.3. Pang 3D

El funcionament del joc serà el mateix que en dues dimensions, tret que ara el moviment es realitzarà sobre un pla en tres dimensions. Això afegeix moltes complicacions de jugabilitat, ja que la visibil.litat no es ni de bon tros la mateixa, al poder-se produir oclussions entre els elements del joc (bombolles, plataformes) i el personatge principal.

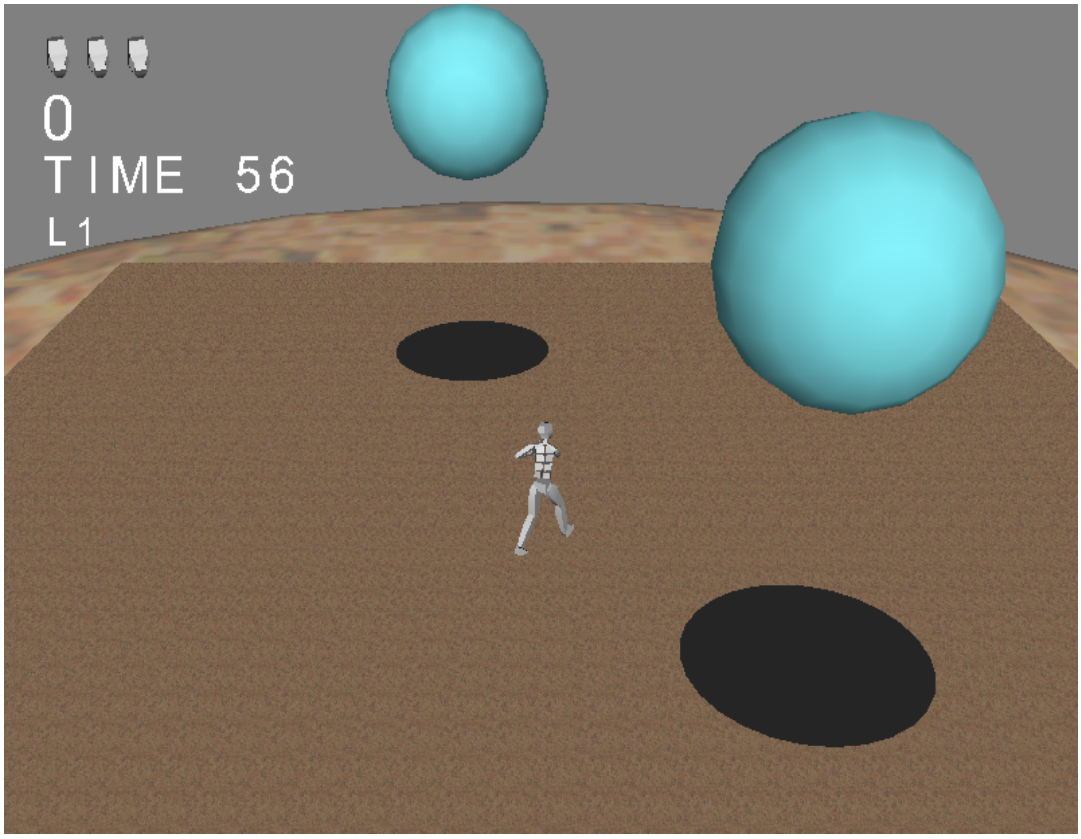
El àrea de joc estarà delimitada per una plataforma en 3 dimensions, sense cap mur.

La càmara, doncs, serà un dels aspectes més importants a tractar, així com la possibilitat de fer semitransparents

alguns elements, de manera dinàmica, per tal de millorar-ne la jugabilitat.

També s'implementarà la física de les bombolles, en tres dimensions i la possibilitat de tenir plataformes amb les que les bombolles col.lisionen

La vista principal de joc estarà fixe al moviment del jugador (és a dir, conservarà amb ell angle i distància), i serà com la de la imatge següent:



1.4. Modes de joc i menús

En principi el joc només constarà d'un menú principal d'entrada al joc, on hi haurà la opció de que un únic jugador jugui una partida, una altra opció de veure els rècords que existeixen fins ara, i una altra opció de sortir del programa.

2. Eines i llibreries a utilitzar en el desenvolupament del projecte

Eines comercials

Per a la realització del projecte s'utilitzaran les següents eines comercials:

- ***IDE:*** Microsoft Visual Studio 6.0`
- ***Compilador:*** Microsoft Visual C++ 6.0

- ***Programa de disseny 3D:*** 3D Studio MAX versió 5.0

Eines gratuites

Només s'utilitzarà una eina no comercial per a la realització del joc : *argoUML*, per als diagrames de l'anàlisi i el disseny.

Llibreries

Entre d'altres llibreries, s'hauràn d'utilitzar les següents:

Lib3ds: llibreria per la lectura de fitxers en format *.3ds* (un dels formats exportats pel programa 3D Studio MAX)

Link: http://lib3ds.sourceforge.net/

Glut: llibreria d'utilitats en openGL, que consisteix en un conjunt de funcions per activar càmeras, generar objectes, inicialitzar els dispositius de pantalla, etc...

Link: http://www.opengl.org/resources/libraries/glut.html

3. Requisits del programa

3.1. Paquets del programa

En base a la descripció del projecte realitzada al *Pla de Treball* (PAC 1), el programa hauria d'estar dividit en dos paquets ben diferenciats: joc i *engine* 3D. En la pràctica ens serà molt més útil dividir-ho en tres parts diferenciades: joc, llibreria matemàtica i *engine* 3D, ja que la llibreria de matemàtiques és suficientment autònoma com per a poder ser utilitzada en altres projectes diferents d'aquest.

Tant el joc com el *engine* 3D depenen d'aquesta llibreria matemàtica.

Aquí podem trobar el diagrama de paquets del anàlisi (pàgina 16)

3.1.1. Llibreria matemàtica (*mathLib*)

Implementa els TADs: Vector3, Matrix3, Matrix4

Aquesta llibreria ha de ser capaç de realitzar totes les operacions matemàtiques que pugui necessitar tant el joc com el *engine* 3D. Concretament, ha d'implementar un TAD de vectors, que serà *Vector3*, un TAD de matrius 3x3, que s'anomenarà *Matrix3*, i un altre de matrius de 4x4 (*Matrix4*).

La justificació de la necessitat de dos TADs diferents per a matrius de 3x3 i matrius de 4x4 , quan clarament unes formen part de les altres, bé donada per motius de velocitat i de flexibil.litat: el TAD *Matrix3* es de molta utilitat en procediments a on no s'ha de tenir en compte la component homogènia de la matriu, i per tant no hi ha desplaçament, com ara les rutines de dinàmica de sòlids rigids.

Vector3

Cada *Vector3* ha de constar de tres components (x,y i z), així com de les operacions necessàries per fer sumes, restes, normalitzar, consultar la longitud i fer el producte escalar i vectorial entre dos vectors.

Matrix3

Aquest TAD ha de oferir els serveis necessaris per a poder inicialitzar una matriu, multiplicar matrius entre si, ortonormalitzar-les (és a dir, garantir que formen una base), així com poder crear matrius de rotació en els eixos X,Y i Z. També ha de poder oferir serveis per a multiplicar vectors per una matriu i per una matriu inversa.

Aquestes matrius estan enmagatzemades en *Column Major Order*, tot i que el ordre d'enmagatzegament és el contrari al que utilitza openGL (openGL, per motius històrics, tot i utilitzar *column major order* per a les matrius, les enmagatzema per files)

Matrix4

Aquest TAD és una matriu 4x4 que ofereix tots els serveis de les matrius de 3x3 més un vector que fa de component homogènia. En la pràctica, i per motius de velocitat i espai, ens interessarà tenir enmagatzemat aquesta matriu en una taula de 4x3 elements.

Ofereix els mateixos serveis que *Matrix3* per la part de la matriu 3x3, i els mateixos serveis que *Vector3* per a la component homogènia.

3.1.2. *Engine* 3D

TADs i mòduls que implementa:

Geomètrics : objecte, base d'objectes, sòlid rígid, mòdul de col.lisions

Visualització: llum, camara, mòdul de textures.

Hardware: mòdul d'inicialització, mòdul d'errors

El *engine* 3D és el encarregat de realitzar tota la visualització en pantalla dels elements 3D, així com de gestionar la dinàmica i les col.lisions.

Ha de constar de les funcionalitats següents:

Un TAD d'objecte per enmagatzemar la informació geomètrica de cada objecte així com els seus materials i textures.

Cada objecte ha de tenir informació no només dels seus triangles (que conformen la seva geometria), sino també un conjunt de nodes que formen l'objecte (si és un objecte compost), així com informació per a poder descartar-lo pel *frustum* de la càmara en el moment de la visualització (en aquest cas, s'enmagatzema el radi tant dels nodes com de l'objecte final, així com el centre de cada node).

Tanmateix, els objectes també han de tenir informació de les normals tant a nivell de triangle com de vèrtex, per tal de poder realitzar la iluminació (s'utilitzaran llums standard d'openGL).

Aquest TAD està implementat a partir de la llibreria *lib3ds*, que ha estat modificada mínimament per poder afegir els radis als nodes, així com per poder afegir alguna informació addicional de textures.

Una base d'objectes 3D, que seràn tots els objectes utilitzables pel *engine* en un moment determinat de la execució del programa. Aquesta base ha de poder servir per carregar objectes des de disc i poder-los adreçar.

Una càmara, que ha de permetre poder establir la posició, direcció i angle de rotació, així com el angular, amb el qual volem veure un conjunt d'objectes 3D determinats en un moment concret.

Aquesta càmara ha de poder trasladar els objectes des de les coordenades del món a les coordenadas de pantalla, i també ha de poder determinar si un objecte és visible o no en pantalla abans d'haver-lo visualitzat.

A més, per simplicitat, aquesta càmara ha de fer servir un sistema de coordenades LH (Left handed), en oposició al sistema de coordenades RH (Right handed) que utiliza normalment openGL. Això ho volem així perque sempre ens serà molt més fàcil tractar amb coordenades de Z+ cap al fons de la pantalla.

Un mòdul d'inicialització del hardware, que en aquest cas inicialitza els paràmetres per defecte de openGL, així com la finestra de visualització en windows. Part d'aquestes tasques seràn realitzades a partir de la llibreria *GLUT*.

Una interfície per poder habilitar llums, que en aquest cas ha de permetre com a mínim poder activar fins a dues llums en coordenades de món. L'únic tipus de llum que es soportaran seran les direccionals infinites.

Un TAD de sòlids rígids, per tal de poder aplicar dinàmica als objectes que sigui necessari. Aquest TAD ha de poder, com a mínim, gestionar moviments lineals (sense rotacions).

Un mòdul de col.lisions, que està inclós al TAD de sòlid rígid. Aquest mòdul ha de poder determinar si dos sòlids rígids col.lisionen, i si un sòlid rígid col.lisiona amb l'escenari.

Un mòdul de gestió de textures, per poder carregar-les a la memòria d'openGL i gestionar la seva activació.

Per qüestions de eficiència en la compressió i en l'enmagatzegament en disc, s'ha decidit d'utilitzar textures en format *PNG*, i per això s'ha utilitzat la llibreria *libpng* (que utilitza, a la seva vegada, la llibreria *zlib*).

Un mòdul d'errors, per poder gestionar els errors que es poden produir durant la execució del *engine*.

3.1.3. Joc

El joc conté el punt d'entrada del programa, i és el encarregat d'inicialitzar el *engine* 3D (que inclou la inicialització del hardware), i el software en si.

Les funcionalitats requerides són les següents:

Un sistema d'estats, que permeti fer les transicions entre les diferents fases d'execució del programa (inicialització de bucle de joc, bucle de joc, inicialització de menú, menú, etc...)

Els estats mínims requerits per a la execució correcta del programa principal seràn (tret de l'estat d'inicialització del programa, que es crida en executar-se el *main*):

ST_INI_MENU : estat d'inicialització del menú
ST_MENU: estat d'execució del menú
ST_INI_FASE: estat que inicialitza una fase nova o ja feta (quan s'acaba el temps, o s'elimina una vida al jugador sense matar-lo)
ST_INI_GAMELOOP: estat d'inicialització del bucle de joc
ST_GAMELOOP: estat del bucle de joc
ST_INI_SHOWRECORDS: inicialitza l'estat d'ensenyar rècords
ST_SHOWRECORDS: ensenya els rècords actuals
ST_INI_INSERTRECORD: inicialitza l'estat d'inserir rècord
ST_INSERTRECORD: inserta rècord

Mentre que per a l'execució del bucle de joc, hem triat un sistema de subestats, que ha de contenir com a mínim els següents estats:

ST_GAMELOOP_START : subestat en el que es donen 3 segons al jugador abans de començar a jugar
ST_GAMELOOP_PAUSED : subestat de pausa de la fase
ST_GAMELOOP_ISGOD : subestat en el que el jugador es invulnerable (acaba de començar a jugar, bé perque ha començat la fase o li han matat una vida)
ST_GAMELOOP_PLAYING : subestat normal del bucle de joc
ST_GAMELOOP_DEAD : subestat en el que el jugador ha estat tocat per una bombolla
ST_GAMELOOP_ABORT : subestat per cancel.lar la partida en curs
ST_GAMELOOP_CONGRATULATIONS : subestat en el que s'ha acabat una fase i es felicita al jugador abans de passar a la següent
ST_GAMELOOP_GAMEOVER : subestat de game over

El diagrama complet d'estats del joc està al apartat del anàlisi: Diagrames d'estats (pàgina 25)

Un sistema de gestió dels missatges que arriben al programa, per tal de poder detectar la pulsació de tecles i ratolí. Aquesta sistema està gestionat en part per la llibreria *GLUT*.

Un menú, que ha de poder permetre sortir de programa i passar a executar una partida del joc.

Un TAD per al jugador, que ha de incloure la gestió del moviment, així com la interacció amb la resta d'elements del joc (bombolles, escenari, etc...). El jugador haurà de ser un sòlid rígid.

Un mòdul per a gestionar les bombolles, que inclourà un TAD bombolla, així com una llista de totes les bombolles existents en un instant determinat del joc.

La llista de bombolles ha de permetre eliminar i afegir bombolles, així com poder dividir una bombolla existent en dues més.

El TAD bombolla ha de ser un sòlid rígid, i ha de poder gestionar les interaccions amb l'escenari i amb el jugador principal.

Un mòdul per a gestionar la visualització e inserció de rècords, que permetrà mantenir una llista de rècords en un fitxer de disc per tal d'implementar la persistència d'aquests entre diferentes execucions del programa.

Un mòdul per a gestionar el bucle principal de joc, que s'ha d'encarregar d'inicialitzar una partida, així com de gestionar la lògica del joc mentre la partida estigui en curs.

Un mòdul d'errors, per a gestionar els errors que es produeixin durant la execució del programa.

Un mòdul de debug, per tal de poder visualitzar informació de debug interessant durant la execució del joc.

3.2. Casos d'ús

Per a aquest joc només hi haurà tres estats que permeten interacció del usuari, i seràn l'estat de *MENU,* el de *INSERTRECORD* i l'estat de *GAMELOOP*.

El usuari bàsicament el que podrà fer és **jugar una partida**, **pausar la partida, abortar-la, visualitzar rècords**, **insertar rècords** i **sortir del programa**.

Per **jugar una partida** el jugador ha de presionar la opció del menú corresponent i entrarà en la pantalla de joc, on es podrà moure i disparar.

Per **pausar una partida**, l'usuari ha de pulsar la tecla *P* durant el joc, i per **abortar-la** ha de pulsar la tecla *ESC* i confirmar l'abortament.

Per **visualitzar rècords** el que farà és presionar la opció del menú corresponent.

Insertar rècord serà un cas d'ús que només es produira després d'haver fet un nou rècord en acabar una partida.

Sortir del programa és una opció del menú principal que dóna la possibilitat de tancar el programa i tornar al sistema.

3.2.1. Cas d'ús jugar partida:

Resum de la funcionalitat: cas d'ús de joc d'una partida de joc normal
Actors : jugador
Casos d'ús relacionats : insertar rècords
Precondició : el jugador es troba al menú principal i pulsa la opció de jugar
Postcondició : al jugador se li han acabat les vides (ha mort)
Fluxe d'events :

i. el jugador pulsa la opció de jugar partida
ii. s'inicia la fase amb 3 segons perque el jugador es prepari mentre es mostra el cartell "Start" amb un contador de temps
iii. el jugador comença a jugar amb 3 segons d'inmunitat i ja es pot moure i disparar.

3.2.2. Cas d'ús pausar partida:

Resum de la funcionalitat: cas d'ús per fer una pausa mentre s'està jugant
Actors : jugador
Casos d'ús relacionats : jugar partida
Precondició : el jugador està jugant
Postcondició : el jugador continua jugant
Fluxe d'events :

i. el jugador pulsa la tecla *P*
ii. el jugador es troba en pausa
iii. el jugador torna a pulsar la tecla *P* per continuar jugant

3.2.3. Cas d'ús abortar partida:

Resum de la funcionalitat: cas d'ús per abortar una partida mentre s'està jugant
Actors : jugador
Casos d'ús relacionats : jugar partida
Precondició : el jugador està jugant
Postcondició : el jugador continua jugant o surt al menú principal
Fluxe d'events :

i. el jugador pulsa la tecla *ESC*
ii. el jugador es troba amb un menú que li demana confirmació
iii. el jugador selecciona la opció que vol amb les tecles de direcció *esquerra, dreta*
iv. el jugador pulsa *espai* per confirmar l'abortament o continuar jugant

3.2.4. Cas d'ús visualitzar rècords:

Resum de la funcionalitat: visualització de les 10 millors puntuacions
Actors : jugador
Casos d'ús relacionats : insertar rècord

Precondició : el jugador es troba al menú principal i pulsa la opció de visualitzar rècords, o el jugador acaba d'insertar un nou rècord
Postcondició : cap
Fluxe d'events :

i. es mostren els 10 millors rècords
ii. el jugador pulsa la tecla *espai* per sortir al menú principal

3.2.5. Cas d'ús insertar rècords:

Resum de la funcionalitat: insertar les tres inicials d'una puntuació rècord entre les deu primeres
Actors : jugador
Casos d'ús relacionats : visualitzar rècords
Precondició : el jugador ha acabat una partida (ha mort), i ha obtingut una puntuació entre les deu primeres
Postcondició : el jugador ha insertat les tres inicials del seu rècord
Fluxe d'events :

i. el jugador mou amb les tecles de *dreta* i *esquerra* la inicial seleccionada fins que arriba a la que vol
ii. el jugador pulsa la tecla *espai* per validar la inicial seleccionada.
iii. el jugador inserta la resta d'inicials fins a completar les seves tres inicials

3.2.6. Cas d'ús sortir del programa:

Resum de la funcionalitat: abandonar la execució del programa
Actors : jugador
Casos d'ús relacionats : cap
Precondició : el jugador es troba al menú principal
Postcondició : el programa es tanca
Fluxe d'events :

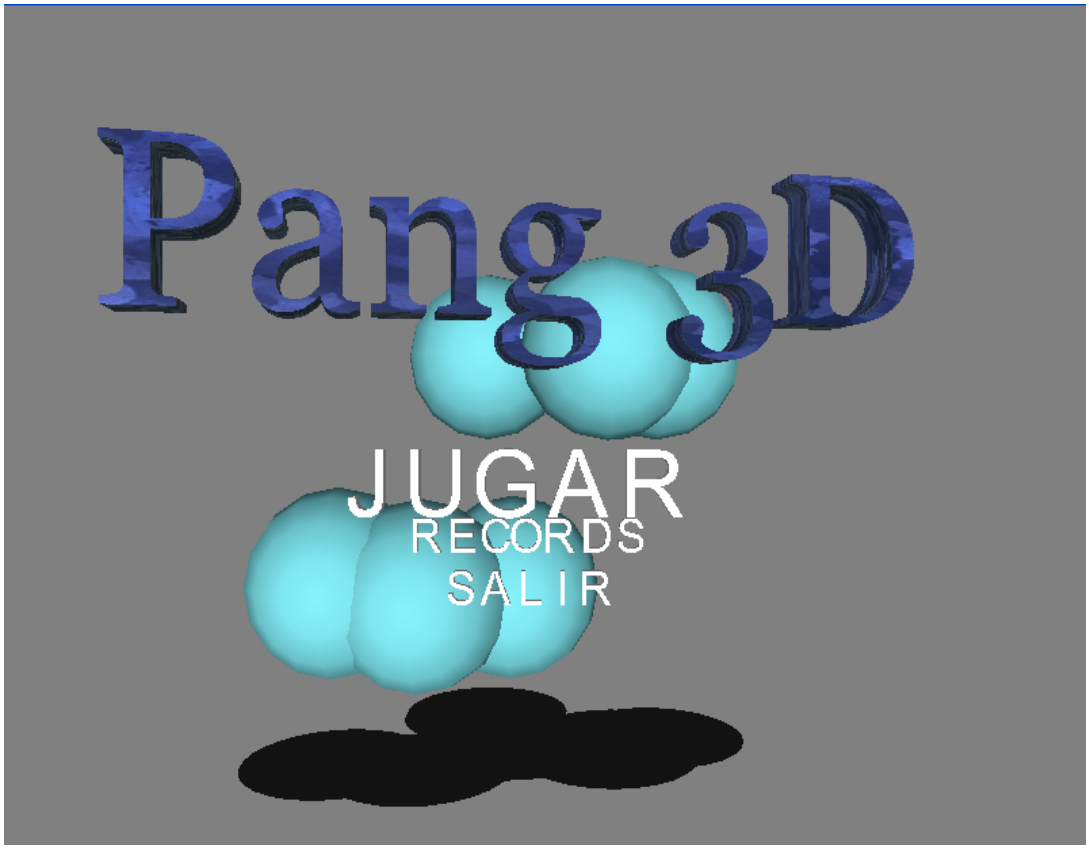
i. el jugador selecciona la opció *salir* i pulsa la tecla *espai*

3.3. Interfície d'usuari

La interacció del usuari amb el programa es farà mitjançant el teclat. Les tecles a utilitzar seran les tecles de direcció, així com la tecla *ESC* (per abortar partida), la tecla *P* (per pausar una partida), i la barra d'espai (per disparar).

Menú

En el menú, la pantalla principal visualitzarà un menú amb tres opcions possibles:

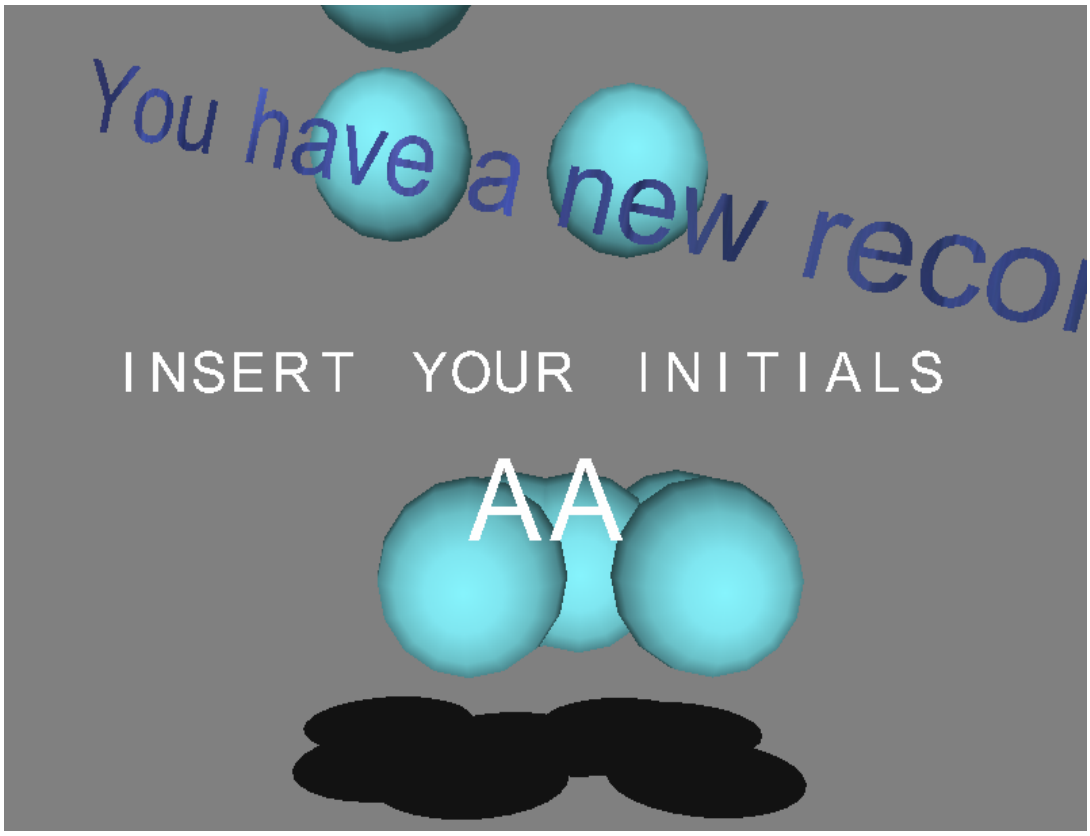


Per defecte estarà seleccionada la opció *Jugar Partida*. Quan una opció estigui seleccionada s'indicarà amb la opció del menú adient resaltada a través d'un escal.lat.

En pulsar les fletxes endavant o darrera la opció seleccionada canvia, i en pulsar la barra d'espai, la opció seleccionada passar a ser la opció a executar.

Insertar rècords

La pantalla d'inserció de rècords permetrà insertar les tres inicials del nom d'un jugador. Això es farà mitjançant les tecles de direcció, pulsant la barra d'espai per a poder fixar cada inicial com a vàlida. En acabar d'insertar les tres inicials, l'estat d'inserció de rècords terminarà, i es donarà pas a la pantalla de visualitzar aquests rècords.

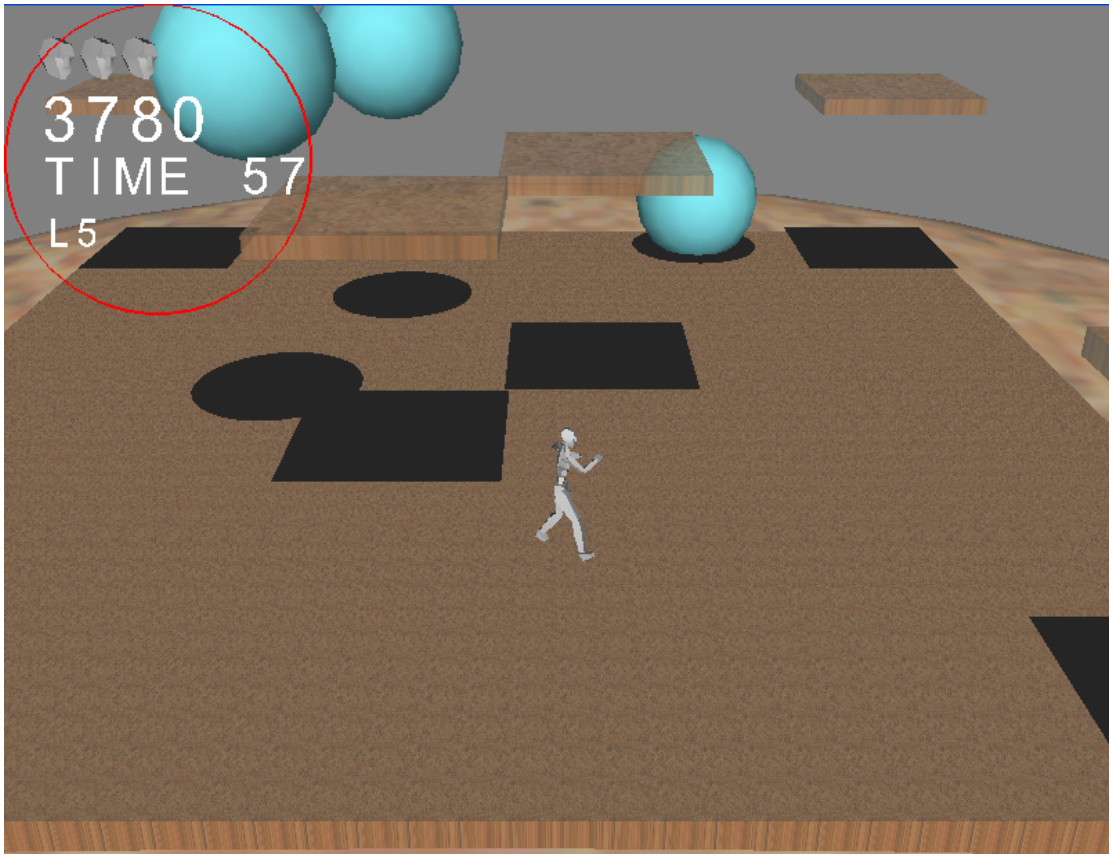


Joc

Durant la execució d'una partida, el usuari podrà moure's cap a la esquerra, dreta, endavant o darrera amb les tecles de direcció del teclat, i podrà disparar amb la tecla d'espai. Igualment, es podrà abortar la partida amb la tecla *ESC* (es demanarà confirmació), i pausar-la amb la tecla *P* (s'haurà de tornar a pulsar *P* per tal de continuar jugant)

Mentre duri la partida, el jugador tindrà en pantalla un contador de vides disponibles (cada nou joc començarà amb 3 vides), així com de la seva puntuació actual i el temps que li resta per trencar totes les bombolles que encara hi ha a pantalla. També tindrà informació del nivell actual de joc.

Tota aquesta informació estarà a la part superior esquerra de la pantalla de joc:



3.4. Jugabilitat

Un dels aspectes més importants d'un joc, sino la que més, és la seva jugabilitat, ja que el seu propòsit es fer passar al usuari una estona de diversió.

Aquesta jugabilitat va lligada al nivell de dificultat que té el joc, ja que aquest ha de ser progressiu, per tal que el jugador no trobi frustrant el no aconseguir passar de determinats nivells, i també ha de permetre al jugador anar millorant el seu nivell a mesura que vagi guanyant experiència.

Pang3D es bassa en un nivell de dificultat progressiu, en el qual a cada nivell s'aniran afegint nous elements per tal que la complicació vagi en augment.

Així, doncs, està previst que fase a fase, es vagin afegint plataformes cada vegada més complicades, per tal que les col.lisions de les bombolles amb aquestes dificultin cada vegada més el joc, i també està previst anar afegint bombolles a cada nou nivell de dificultat.

El nombre de subdivisions d'aquestes bombolles es també un factor important, però finalment serà un detall de la implementació de la jugabilitat (pàgina 30), ja que dificilment sabrem quantes subdivisions necessitem per fer el joc jugable fins a provar-lo.

Per cada bombolla que el jugador esclata, se li donarà un número de punts, ja que aquest serà un factor més que crearà competitivitat entre els diferents usuaris.

Un altre aspecte que hem de tenir en compte serà el de les col.lisions, ja que aquestes poden afectar molt negativament a la jugabilitat del joc degut a que en 3D ja podem imaginar que serà molt més difícil esquivar les bombolles i disparar-les.

També, per ajudar als jugadors i com a al.licient, es donarà una vida extra cada cert número de punts. Per qüestions de jugabilitat, la primera vida es dóna amb un número de punts menor que el període de la resta de vides, per ajudar així als jugadors més dolents.

3.5. Notes addicionals sobre el joc

Com a requisits addicionals a la mecànica de joc, tal com es va especificar al *Pla de Treball*, s'ha inclós el fet que el joc no finalitzi mai, i per tant la única manera d'acabar serà que el jugador mori.

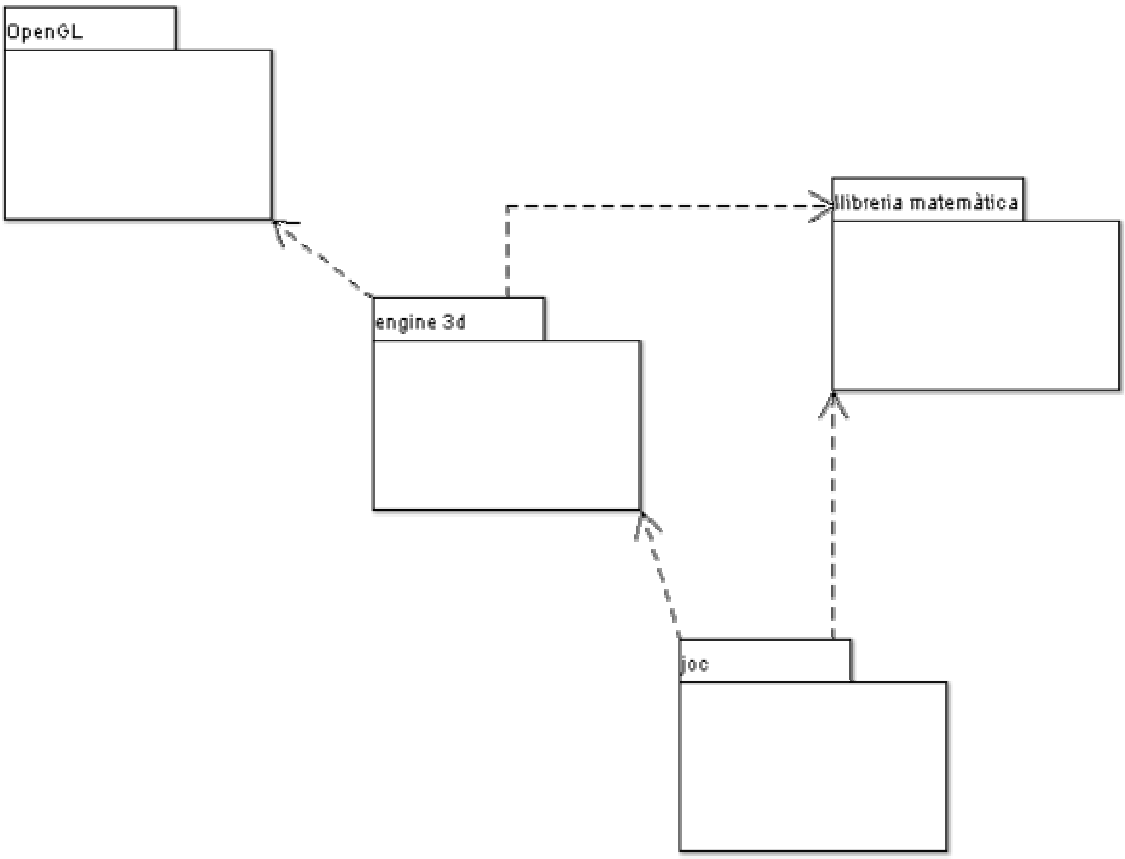
4. Anàlisi del programa

Com ja hem explicat als requeriments, el programa consistirà en tres paquets diferenciats, que seràn el joc, el *engine* 3D de visualització, i la llibreria matemàtica.

Aquests tres paquets estàn relaccionats entre si, ja que el joc dependrà tant del *engine* 3D com de la llibreria matemàtica, mentre que el *engine* 3D dependrà únicament de la llibreria matemàtica.

En quan als diagrames que utilitzarem per especificar aquest anàlisi seràn els diagrames de paquets, el de classes dels diferents paquets, i un diagrama d'estats del joc. En principi no ens farà falta cap diagrama de casos d'ús, ja que aquests són molt simples, ni utilitzarem cap més diagrama per la senzillesa del cas.

El diagrama dels paquets i les seves relacions, és el següent:

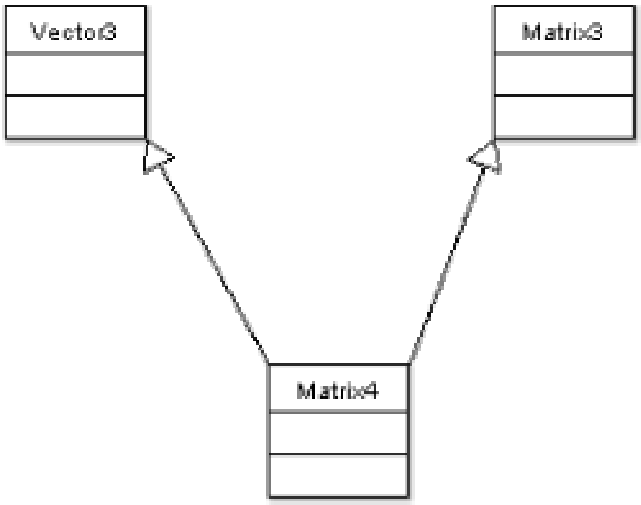


Aquí es veuen les dependències explicades anteriorment i la dependència del *engine* del *API* de openGL.

4.1. Diagrames de classes

Llibreria matemàtica

Aquesta llibreria consta de tres classes, que són *Vector3*, *Matrix3* i *Matrix4*. El seu diagrama de classes és el següent:



En aquest diagrama podem veure que *Matrix4* heredarà directament de *Matrix3* i *Vector3*, ja que en realitat és una ampliació d'aquesta amb una fila més (recordem que als requisits hem dit que finalment

aquesta classe seria de 4x3 elements i no de 4x4, com es podria supossar).

Més detalladament, cada classe (en realitat seran TADs).

Vector3:

Vector3
- **x**, **y**, **z** : float
+*vvadd*(v0 : Vector3,v1 : Vector3) : Vector3 +*vvsub*(v0 : Vector3,v1 : Vector3) : Vector3 +*vsmul*(v0 : Vector3,v1 : Vector3) : float +*vvmul*(v0 : Vector3,v1 : Vector3) : Vector3 +*vlength*(v : Vector3) : float +*normalize*(v : Vector3)

La classe (TAD) *Vector3* té les operacions descrites als requisits, i totes aquestes operacions són públiques. Per tant, no és possible accedir a les components *x,y,z* del vector, però en la pràctica aquesta situació es pot arribar a donar per qüestions de velocitat i senzillesa en el procés.

Matrix3:

Matrix3
-**m[9]** : float
+*copy*(src : Matrix3) +*scale*(scale : Vector3) +*identity*() +*invert*() +*rotate*(angle : float,axis : int) +*mvmul*(d : Vector3) : Vector3 +*imvmul*(d : Vector3) : Vector3 +*mmul*(src : Matrix3) +*m2gl*() : oglMatrix +*gl2m*(m : oglMatrix)

Aquesta classe té definides les operacions que s'han descrit als requisits. Les úniques operacions remarcables, són *imvmul* i *mvmul*, que multipliquen un *Vector3* per la matriu o la matriu inversa, i les dues operacions *m2gl* i *gl2m*, que en realitat el que fan es convertir un objecte del tipus *Matrix3* des de'l nostre format al format *openGL*.

Matrix4:

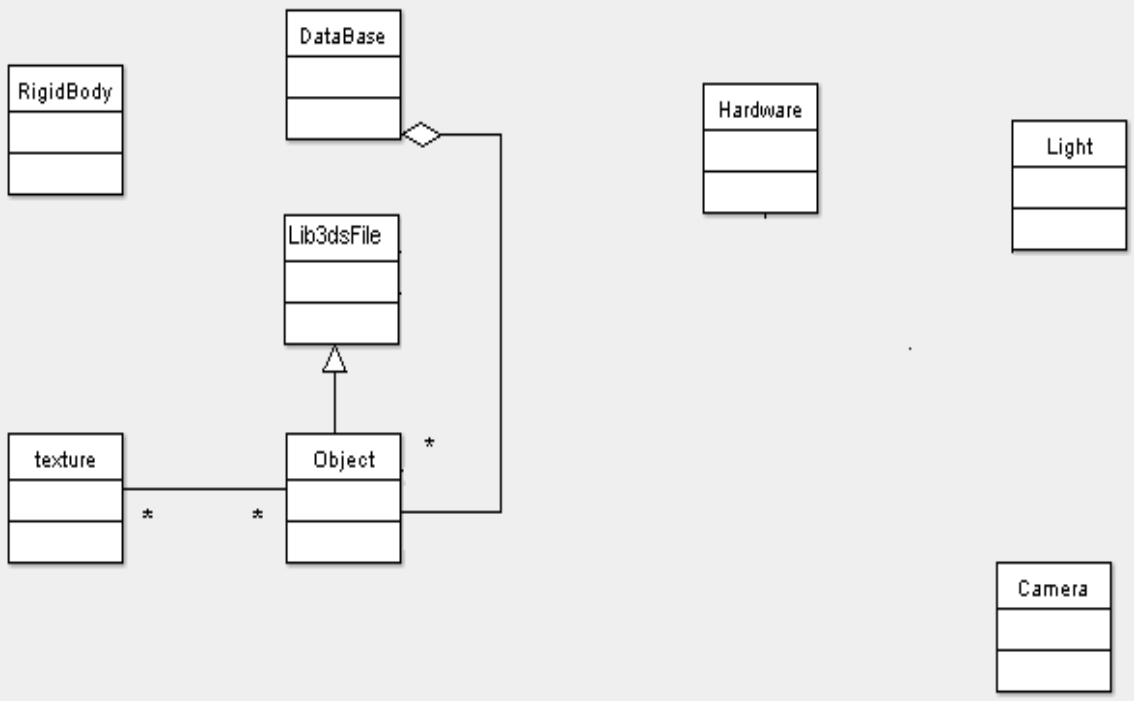
Matrix4
-**r** : Matrix3 -**off** : Vector3

```
+copy(src : Matrix4)
+identity()
+mmul(src : Matrix4)
+m2gl():oglMatrix
+gl2m(m : oglMatrix)
+transformVect(d : Vector3) : Vector3
+rotate(angle : float, axis : int)
+translate(d : Vector3)
```

Aquesta classe, al igual que la classe *Matrix3*, ofereix operacions per tal de convertir entre el seu format intern i el format utilitzat per openGL. La resta d'operacions són molt semblants, tret de les dues noves operacions : *transformVect* i *translate*, que multipliquen un vector per la matriu de 4x3, i generen una matriu de translació que es multiplica per la matriu actual de l'objecte sobre el que s'executa la operació, respectivament.

***Engine* 3D**

El diagrama de classes del *engine* 3D és el següent:



En aquest diagrama ja podem veure alguns aspectes d'implementació que no hem pogut deixar de banda, com que la classe *Object* deriva d'una classe anomenada *Lib3dsFile*, que és la definició d'una escena *3ds* que ens ofereix la llibreria *lib3ds*.

Una anàlisi més detallada d'aquesta classe, es pot trobar a http://lib3ds.sourceforge.net

La única funcionalitat afegida a la classe *Object* que no está a *Lib3dsFile*, és la sintaxi de noms, per poder diferenciar els models per noms diferents al que tenien en ser exportats des de *3dsmax*, i la associació amb les textures, que és una funcionalitat que no oferia aquesta llibreria.

També cal recalcar el paper que juga la classe *DataBase*, ja que és el contenidor que manté tota la llista d'objectes carregats (també podria haver-se fet una associació o agregació amb les textures i els materials que conformen la totalitat de la base de dades, per evitar duplicitats, però la estructura de la llibreria *lib3ds* ens ho impedeix en part per als materials, i per tant no té molt sentit fer-ho unicament per a les textures)

Ara detallaré una a una les classes més importants:

Object:

Aquesta classe deriva de *lib3ds*, ja que n'hereda molts dels mètodes d'aquesta (de càrrega, cerca de jerarquies, etc...)

Object
-**name** : String -**radius** : float -**obj** : Lib3dsFile
+*setName*(name : String) +*getName*() : String +*load*(filename : String) +*free*() +*render*(v : Vector3, m : Matrix3, flags: int, frame : int) +*renderNode*(name : String, v : Vector3, m : Matrix3, flags : int, frame: int) +*getNodeTM*(name : String) : Matrix4

Els métodes principals d'aquesta classe, a més de poder llegir un objecte des de fitxer, i alliberar-lo, són els que s'encarreguen de renderitzar l'objecte en pantalla. Aquests són *render* i *renderNode*. El primer renderitza l'objecte complet, i el segon només un node de l'objecte (l'objecte té nodes amb jerarquia, com està especificat al format *3ds*).

El mètode *getNodeTM* serveix per demanar la matriu de 4x3 del pivot d'un node determinat.

Els flags de pintat disponibles i els seus significats són:

NOCLIP : renderitza l'objecte tot i que estigui fora de pantalla
PIVOTCOORDINATES : renderitza l'objecte en coordenades de pivot

DataBase:

DataBase
-**nObjectes** : int -**llistaObjectes[]** : Object
+*load*(resources[] : Resource) +*getObject*(name : String) : Object +*free*()

La classe *DataBase*, manté una llista d'objectes i té mètodes per carregar-los des de disc a la memòria a partir de una llista de recursos (en aquest cas s'ha utilitzat una classe anomenada *Resource*, que només és una estructura que associa a cada fitxer en disc un nom d'objecte 3D). També té un mètode per alliberar tota aquesta llista d'objectes.

Tot i això, la funcionalitat més utilitzada d'aquesta classe és el mètode *getObject*, que a partir del nom d'un objecte 3D ens retorna un punter a aquest.

Càmara:

Camara
-**dir** : Vector3 -**pos** : Vector3 -**tilt** : float -**angular** : float
+*isVisible*(pos : Vector3, radius : float) : boolean +*set*(pos : Vector3, dir : Vector3, tilt : float, angular : float) +*setOrtho*(pos : Vector3, dir : Vector3, tilt : float)

Aquesta classe manté una associació directa amb openGL, i el que fa és activar una càmara en una posició, amb una direcció, angle i angular determinats.

També pot fer servir càmeres amb projecció plana, per poder visualitzar objectes sense perspectiva.

A més de poder activar càmares, la classe permet saber si un determinat objecte amb un determinat radi i posició és visible dintre del camp de visió.

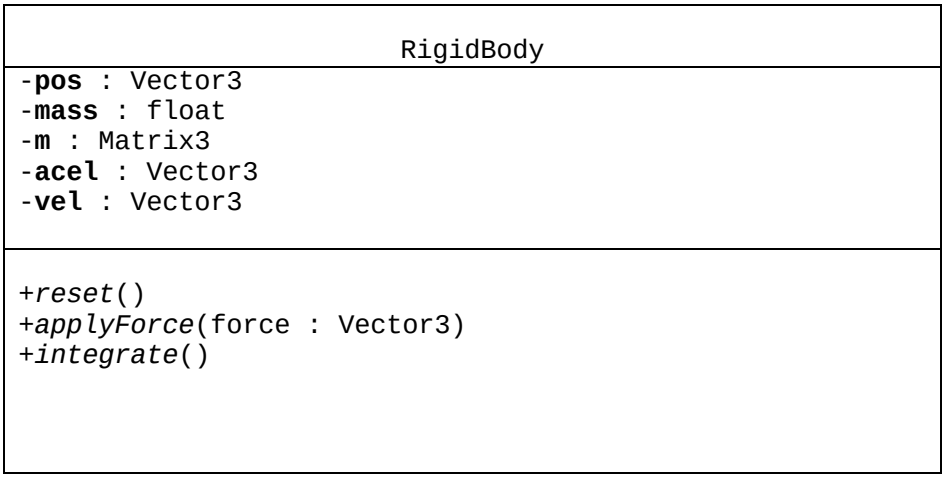
Light:

Light
-**dir** : Vector3 -**ambient** : Vector3 -**diffuse** : Vector3 -**specular** : Vector3
+*set*(nLight : int, dir : Vector3, ambient : Vector3, diffuse : Vector3, specular : Vector3)

Aquesta classe permet activar una llum d'openGL. Les llums han de ser tipus direccional infinites, i poden especificar les components ambiental, difusa i especular.

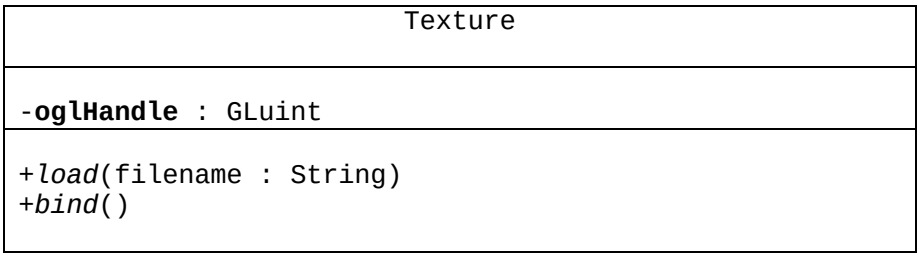
La implementació d'aquest TAD només supporta dues llums actives a la vegada.

RigidBody:



En aquesta classe no s'especifiquen els *setters/getters*, ja que els mètodes importants són el *reset*, que permet inicialitzar un objecte sòlid rígid, *applyForce*, que permet aplicar-li una força, i *integrate*, que integra el sòlid rígid.

Texture:



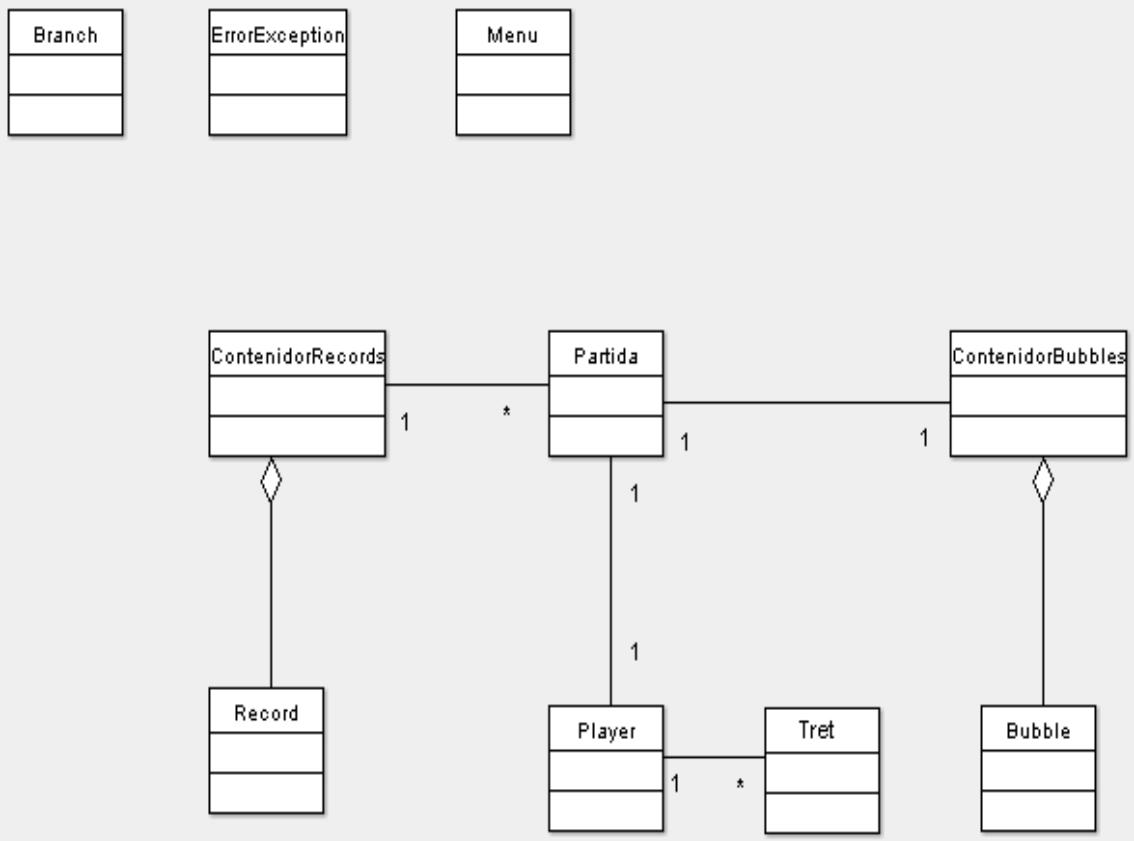
Aquesta classe només té la funcionalitat de carregar i assignar la textura al dispositiu de video hardware, per tal de poder ser utilitzada, des d'un fitxer *.PNG* en disc.

Hardware:

Bàsicament, *hardware* és un conjunt de mètodes que permeten d'inicialitzar una ventana en 3D (a través de *GLUT*), així com inicialitzar els estats openGL. Aquesta classe no requereix cap més especificació, ja que només consta d'un mètode *init* que fa tota aquesta funcionalitat.

Joc

El diagrama de classes del joc és el següent:



En aquest diagrama podem veure que una de les classes més important és *partida*. *Partida* està relacionada amb un contenidor de *records*, que és només una llista de rècords, i també amb un jugador *(player)* i un contenidor de bombolles *(bubbles)* (llista de bombolles).

També hem de comentar que tant *Player* com *Bubble* i *Tret* heredarien de *RigidBody*, però al ser una classe del *engine* només ho comentem textualment.

A més d'aquestes classes, també trobem la classe *Menu*, que serveix perque el usuari pugui entrar al joc, així com algunes classes d'utilitat, com *ErrorException*, que s'encarrega de la gestió d'errors, i *Branch*, que s'encarrega de la lògica dels estats.

Ara passem a comentar cada classe amb més detall:

Partida:

Partida
-**jugador** : Player -**bombolles** : ContenidorBubbles -**records** : *ContenidorRecords -**temps** : int
+*iniciar*() +*iniciarFase*() +*jugar*()

La partida l'únic que conté és un jugador, unes bombolles i una referència a la llista de rècords. Per tant, els seus mètodes són *iniciar*, que inicia una partida, *iniciarFase*, que inicia una fase i *jugar*, que executa un bucle de joc.

Record:

Record
-**inicials** : String -**puntuacio** : int

Aquesta classe ha de poder mantenir un rècord de joc, i per tant només necessita les inicials i la puntuació.

No s'han detallat els mètodes perque només conté *setters/getters*.

Player:

Player
-**body** : RigidBody -**vidas** : int -**obj** : *Object -**frame** : int -**nTrets** : int -**trets** : *Tret
+*iniciar*() +*moure*() +*renderitzar*()

El jugador ha de mantenir una estructura *RigidBody*, ja que s'ha de comportar com a sòlid rígid, un contador de vides, un punter al objecte que s'utilitzarà per renderitzar-ho, i el frame actual d'animació.

Els seus únics mètodes seran *iniciar, moure* i *renderitzar*, que inicialitzen, processen la lògica del jugador i ho renderitzen, respectivament.

Bubble:

Bubble
-**body** : RigidBody -**tamany** : int -**obj** : *Objecte
+*iniciar*() +*moure*() +*renderitzar*()

Al igual que el jugador, la classe bubble ha de mantenir un sòlid rigid, a més d'un identificador del tamany actual que té, i un punter al objecte geomètric que la representa.

En quan a mètodes té els mateixos d'inicialitzar, moure i de renderitzar que el jugador.

ContenidorBubbles:

ContenidorBubbles
-**nBubbles** : int -**arrayBubbles**[] : Bubble
+*inicialitzar*() +*afegirBubble*(b : Bubble) : boolean +*isColliding*(t : Tret) : int +*dividir*(index : int) : boolean

El contenidor de bubbles manté una array d'aquest tipus d'objecte, i conté els mètodes bàsics per inicialitzar el array i afegir un altre bubble a aquest. A més, conté dos mètodes més complicats, com es *isColliding*, que donat un tret determinat detecta una colisió amb una bombolla de la llista, i el mètode *dividir*, que divideix una bombolla en dues.

Tret:

Tret
-**body** : RigidBody -**posicioOrigen** : Vector3

Cada tret és disparat pel jugador, i només consta d'un sòlid rígid, tot i que també ha de enmagatzemar la seva posició inicial. Els mètodes que té aquesta classe són els *setters/getters*.

ContenidorRecords:

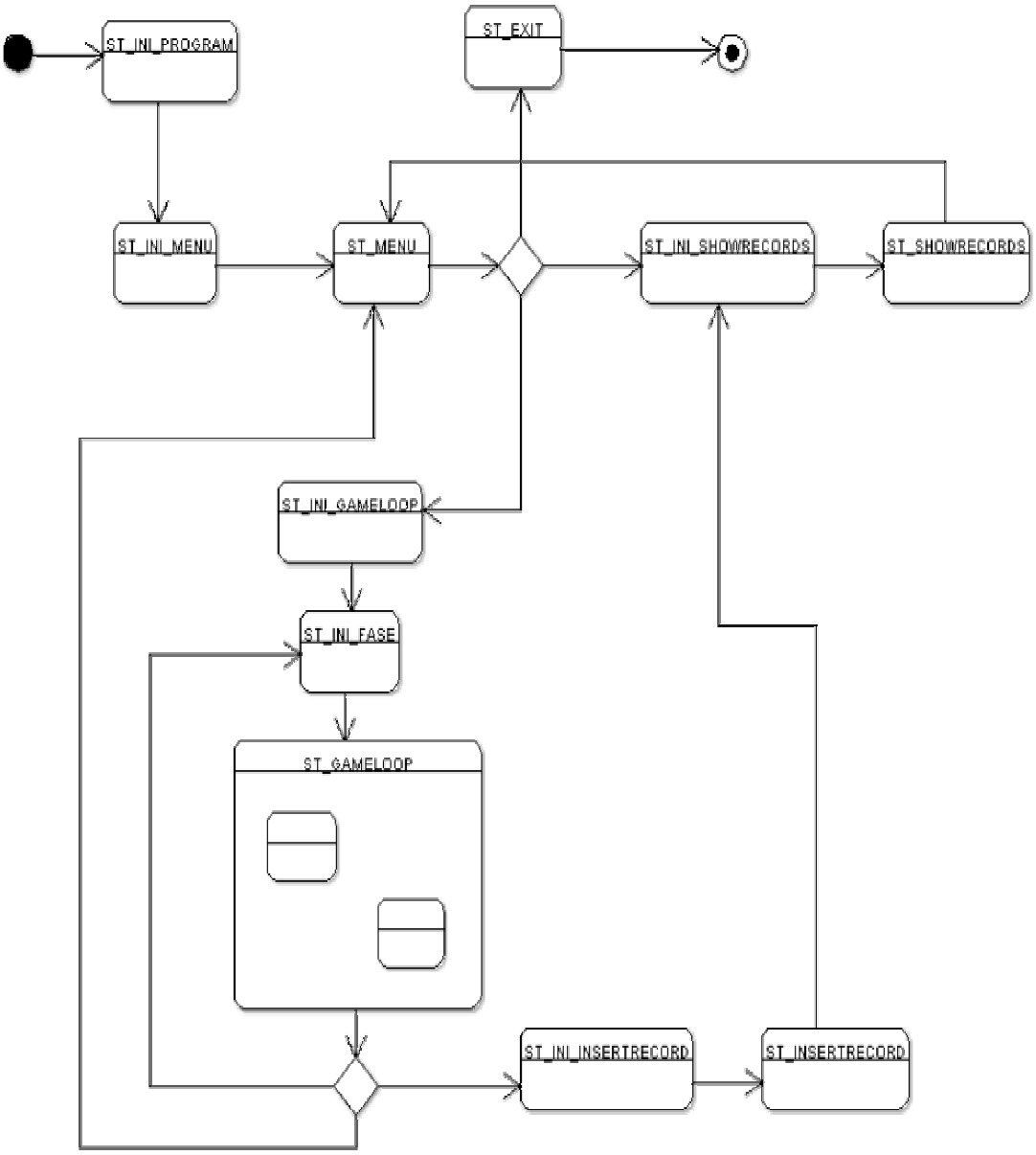
ContenidorRecords
-**nRecords** : int -**arrayRecords**[] : Record
+*inicialitzar*() +*afegirRecord*(r : Record) : boolean +*getFirst*() : *Record +*getNext*() : *Record

Aquesta classe és només una llista de rècords, amb un mètode per inicialitzar la llista a buida, afegir un rècord nou, així com dos mètodes per llistar tots els rècords actuals a la taula: *getFirst* i *getNext*.

4.2. Diagrames d'estats del joc

En aquest apartat tenim dos diagrames, el del joc principal i el del bucle de joc. El primer conté el fluxe del programa principal (transicions entre pantalles de joc, fluxe de partida normal), mentre que el segon conté el fluxe de joc d'una fase. Els estats del bucle de joc són subestats de l'estat *ST_GAMELOOP* del joc principal.

- El diagrama d'estats del joc és el següent:



Primer, el programa entra en l'estat *ST_INI_PROGRAM*. Aquest estat inicialitza hardware i software, i correspon al *main* del programa. Després passa inmediatament a la inicialització de menú, i després al menú. Des de menú, es pot sortir del programa, passar al estat de ensenyar rècords, o d'inicialitzar una partida (*ST_INI_GAMELOOP*).

S'ha de fer notar que si per exemple el programa va a parar a un estat de joc, com el *ST_GAMELOOP*, primer passarà per un estat d'inicialització de joc (*ST_INI_GAMELOOP*), i així amb la majoria de possibles transicions reflectides en aquest diagrama.

ST_GAMELOOP és un estat compost que conté bastant subestats.

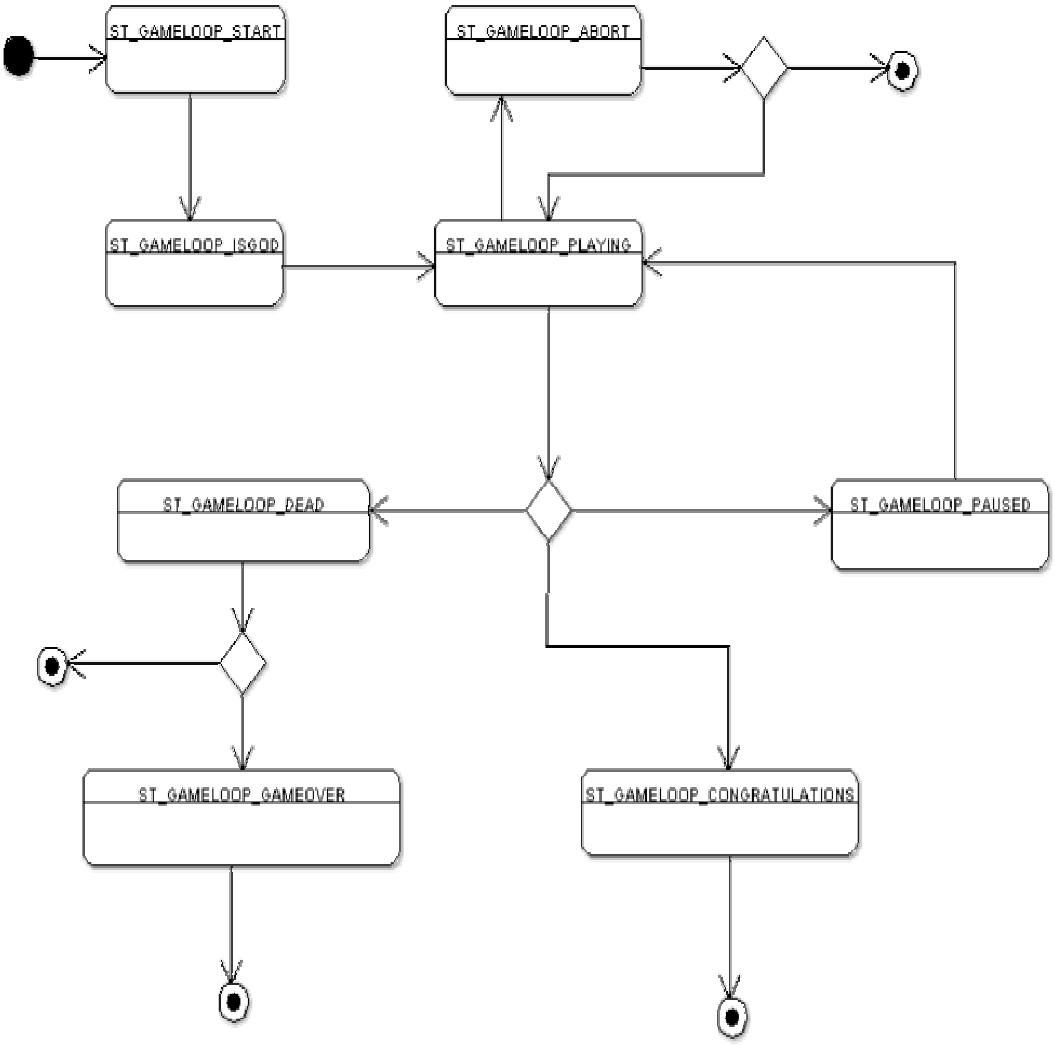
Des de *ST_GAMELOOP* es pot passar a *ST_MENU*, *ST_INI_INSERTRECORD* o *ST_INI_FASE* :

Quan al jugador li treuen un vida per temps o per col.lisió amb bombolles, o quan elimina totes les bombolles, ha de passar a una fase nova o inicialitzar l'anterior, i el joc passarà al estat *ST_INI_FASE*.

Quan el jugador ha mort, ja que s'ha previst que el joc no acabi mai tret de que et matin, el joc passarà o bé al estat *ST_INI_INSERTRECORD* o bé al estat *ST_MENU* (si no hi ha rècords a insertar)

Si el jugador ha abortat la partida en joc, es passa al estat *ST_MENU*.

- El diagrama d'subestats del bucle de joc és:



L'estat inicial és *ST_GAMELOOP_START*, que no és més que una espera de tres segons donant al jugador temps per preparar-se a començar a jugar la fase.

Des d'aquest estat es passa a *ST_GAMELOOP_ISGOD*, un estat en el qual el jugador és invulnerable durant un cert període de temps, per tal d'evitar que només començar la fase li caigui una bombolla al damunt sense que tingui cap possibilitat d'esquivar-la.

Des de l'estat *ST_GAMELOOP_ISGOD*, es passa al estat de jugar normalment *ST_GAMELOOP_ISPLAYING*, des d'on es pot passar a quatre estats diferents: *ST_GAMELOOP_DEAD*, *ST_GAMELOOP_PAUSED*, *ST_GAMELOOP_CONGRATULATIONS* i *ST_GAMELOOP_ABORT*.

A *ST_GAMELOOP_DEAD* s'arriba quan el jugador ha estat tocat per una bombolla o se li acaba el temps i per tant se li ha de restar una vida. Des d'aquest estat es passa a *ST_GAMELOOP_GAMEOVER* si al jugador ja no li resten vides, o sortir de *ST_GAMELOOP* cap a *ST_INI_FASE*, perque torni a començar a jugar a la mateixa fase. Des de *ST_GAMELOOP_GAMEOVER*, es surt de l'estat *ST_GAMELOOP* (i es passarà a insertar rècords o al menú principal)

A *ST_GAMELOOP_PAUSED* s'arriba si el jugador ha pausat la partida mitjançant la tecla *'P'*, i des d'aquí només es pot tornar a continuar jugant.

A *ST_GAMELOOP_CONGRATULATIONS* s'arriba només si el jugador es passa la fase, i des d'aquí es sortirà fora de l'estat general *ST_GAMELOOP* i s'anirà cap a tornar a inicialitzar una altra fase (*ST_INI_FASE*)

L'estat *ST_GAMELOOP_ABORT* és un petit menú al que s'arriba després de pulsar la tecla *ESC* i que permet cancel.lar la partida. Si es cancel.la, es surt de *ST_GAMELOOP* i es passa al estat *ST_MENU*, i sino es continua jugant.

5. Implementació del programa

Per a la implementació, anem a plantejar les adaptacions del anàlisi que hem de realitzar per tal de complir els requeriments del llenguatge de programació, tot i que moltes de les decisions durant el anàlisi van tenir en compte que anàvem a utilitzar un llenguatge no orientat a objectes (C standard).

La utilització de C standard va ser suggerida pel consultor.

Els únics punts que no caldrà retocar respecte als punts tractats anteriorment serà el disseny de la interfície d'usuari, ja que la especificació feta als requisits romandrà inalterada, i el diagrama d'estats, ja que aquest és pràcticament la versió definitiva (tret que surtin nous estats).

5.1. Modificacions respecte al anàlisi

Transformació de classes a TADs.

Ja que hem de suprimir les classes existents al anàlisi, el que farem serà crear mòduls definits de programa amb les propietats com a variables *static* i les operacions que operen sobre aquests tipus.

Totes les operacions sobre un TAD determinat, o pertanyents a un mòdul concret del programa, portaran com a nom, primer el nom del TAD (o el mòdul), i després el nom de la operació a realitzar.

Per exemple, la operació *afegirBombolla* del TAD *ContenidorBombolles*, s'anomenarà *contenidorBombollesAfegirBombolla().*

Per tal d'adaptar la longitud dels noms d'alguns mètodes (eren massa llargs si utilitzem aquesta estructura estrictament), s'han utilitzat abreviatures, com per exemple en la operació que suma dos vectors: *vvadd*.

Modificacions a la llibreria matemàtica.

La llibreria matemàtica, a més d'haver de modificar els noms del mètodes a utilitzar, haurà de definir una estructura per a cada TAD a utilitzar. En el cas de *Matrix3* ho implementarem amb un typedef (no necessitarem cap estructura).

Les operacions per a *Matrix3* afegiran *rm* al principi del nom, per a *Matrix4* afegiran *tm*, i per a *Vector3* afegiran *vv* algunes, i d'altres no res.

L'herència múltiple de *Matrix4* es resoldrà afegint les dues estructures que defineixen *Matrix3* i *Vector3* per generar la nova estructura de *Matrix4*.

Modificacions del *engine*

El *engine* tindrà com a TADs *Object*, *Camera* i *RigidBody*. La resta seran mòduls que oferiran els serveis concrets.

La classe Light serà un mòdul (com *Hardware*), i mantindrà uns flags *static* indicant les llums activades actualment.

Texture en realitat serà un mòdul que només s'encarregarà de carregar un fitxer *.PNG* i fer un *bind* cap a una textura openGL. El identificador del bind a la textura openGL es guardarà al propi objecte 3D.

DataBase serà un mòdul, i la llista d'objectes serà una taula de longitud fixa (del màxim número d'objectes 3D a carregar).

Tant *Camera*, com *RigidBody* i *Object*, en realitat seran estructures amb alguns mètodes i variables privades (seràn definides com *static*), i el mètodes propis que s'han definit al anàlisi.

Modificacions al joc.

Com ja s'havia especificat al anàlisi, els contenidors (*ContenidorRecords* i *ContenidorBubbles*), no deriven de cap classe llista, i tenen un rang determinat i unes operacions fixes d'afegir i treure elements.

Les bombolles i el jugador, deriven de *SolidRigid*, però en la pràctica això vol dir que mantenen una estructura tipus *SolidRigid* dintre de la seva propia estructura, i accedeixen als mètodes propis de la dinàmica des d'ella.

La partida pròpiament no serà una classe, sino un mòdul que anomenarem *GameLoop*, i que inclourà tota la gestió de la lògica d'una partida.

La classe *Menu* passarà a ser un mòdul de la gestió del menú, i la classe d'excepció d'errors en realitat no utilitzarà excepcions, sino que tindrà alguns mètodes per poder abortar el programa amb un determinat codi d'error.

La classe *Branch*, estarà implementada a través d'un array d'estats amb els mètodes que corresponen a cada estat, i algunes variables privades que indiquin l'estat actual i anterior, així com un mètode per canviar d'estat i consultar l'estat actual.

A més, en aquest punt també haurem de mantenir alguns mòduls addicionals per tal de poder cumplir alguns dels requisits, com per exemple un módul de debug per tal de poder llençar missatges de debug durant el joc, així com un mòdul que contindrà el llistat de la base de joc (la llista d'objectes 3D a carregar), per tal de poder cridar als mètodes adients del *engine* per carregar-los.

5.2. Disseny de la persistència

En aquest joc només tenim persistència dels rècords de joc. Aquesta persistència és molt senzilla, ja que només consistirà en gravar tants registres de rècords a disc, com registres de rècords existeixin (un nombre fixe, a més). La gravació de cada rècord consistirà en la gravació de totes les seves dades : nom (les tres inicials), i rècord (un integer).

El nom del fitxer de rècords serà fixe, i el seu contingut es carregarà en començar el joc, i es gravarà a disc cada vegada que s'inserti un nou rècord.

5.3. Implementació de la jugabilitat

En el moment de la implementació la jugabilitat és un aspecte molt important, ja que requereix, a més de la pròpia implementació en sí, una fase de prova suficientment apropiada per al joc en concret.

El procés ideal consisteix en poder distribuir còpies no finalitzades per tal de que alguns usuaris de confiança puguin jugar i emetre la seva opinió sobre el producte.

Per tant a la hora de fer la implementació de la jugabilitat, m'he bassat en la opinió d'alguns familiars i amics per tal de poder ajustar aquest nivell de dificultat.

5.3.1. Tests de jugabilitat

Aquests tests de jugabilitat han estat realitzats amb la col.laboració de 4 persones: el meu germà, un amic i dos companys del treball. Tots ells havien jugat a la versió original de *pang*.

Les proves han consistit en fer jugar a aquestes persones unes quantes partides amb una versió del joc quan aquest estava pràcticament acabat.

Aquestes proves han estat iteratives, ja que se'n va tornar a passar un altre test amb una segona versió del joc modificada.

Les conclusions més importants que es van extreure d'aquestes proves van ser les següents:

i. Al joc original era molt més fàcil encertar les bombolles (tots van coincidir en que era un problema inherent al fet de que en 3D era més difícil precisar a on disparaves, degut a la perspectiva)

ii. Algunes col.lisions entre el personatge i les bombolles semblaven massa ajustades. De fet el problema es que de vegades semblava que no s'havia col.lisionat però en realitat havia hagut col.lisió.

iii. El temps original de 60 segons (com al joc original) era inapropiat a partir de la tercera i quarta fase. La possibilitat de fer aquest temps variable en les fases superiors confonia als provadors.

iv. El número de bombolles que es va planificar inicialment era excessiu (la primera fase començava amb 5 bombolles originalment)

v. El número de subdivisions de les bombolles era excessiu (hi havia 4 subdivisions originalment)

vi. Les plataformes dificultaven la visibil.litat

A més també es van detectar altres problemes menors, com el fet de que quan s'acabava el temps, no s'informava de que havia hagut un *timeout*, sino que simplement el jugador es moria, i això confonia als jugadors. També es van detectar alguns errors a la inserció de rècords, que van ser corregits.

5.3.2. Modificacions en base als tests

Llavors, en base a aquestes conclusions es van fer els següents canvis:

5.3.2.1. Temps de joc

El temps de joc s'ha fixat en 75 segons. Aquest temps és absolutament excessiu per als primers nivells, però apropiat a partir de determinades fases del joc.

5.3.2.2. Bombolles

El número de bombolles parteix de dues, i es va incrementant progressivament, aproximadament cada 4 ó 5 nivells, fins a un màxim de 7 bombolles al nivell màxim de dificultat.

Cada bombolla té un màxim de 2 nivells de subdivisió, és a dir, es pot dividir com a molt en 4 bombolles. A més, s'ha decidit que el tamany de cada bombolla subdividida sigui la meitat que la bombolla pare, ja que amb aquest tipus de subdivisió i els 2 nivells esmentats, el tamany de les bombolles més petites és suficientment gran com perque no sigui impossible encertat un tret per trencar-les.

Per tal de facilitar la jugabilitat, s'ha inclós una ombra a cada bombolla que serveix de punt de referència a la hora d'intentar trencar-les i esquivar-les.

5.3.2.3. Plataformes

Les plataformes tenen unes quantes configuracions possibles, que van passant a mesura que es va jugant el joc. En el primer nivell no n'hi ha plataformes, fins que poc a poc s'hi va incrementant el nombre, fent més complicat trencar les bombolles.

Tot i que n'era molt dubtós, per coherència, s'ha optat perque els trets del jugador no puguin travessar les plataformes (com al joc original), lo que complica encara més la dificultat.

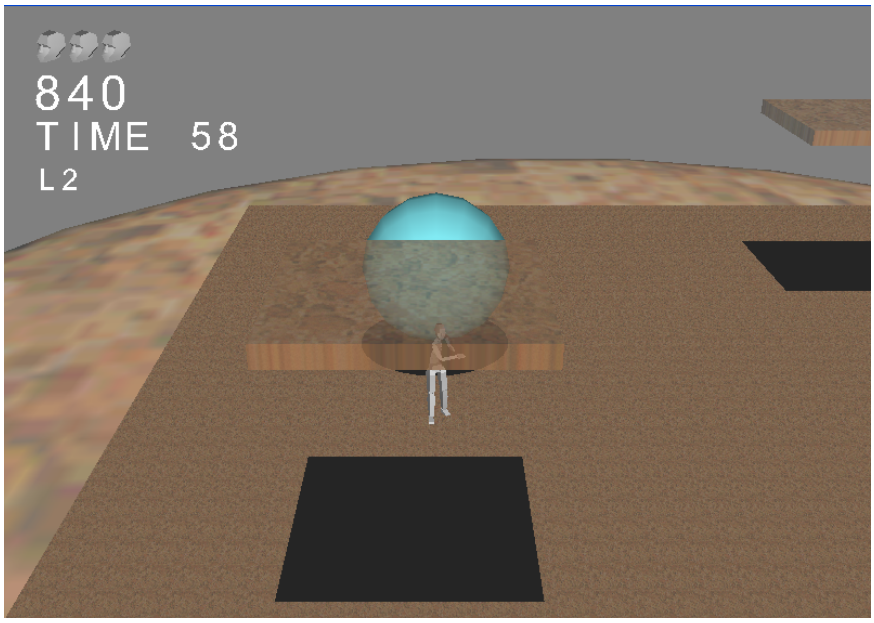
Per fer que la mesura de les plataformes no confongui als jugadors s'ha optat per afegir-hi una ombra, i també fer-les sempre del mateix tamany (es poden fer més grans, però amb combinacions)

La col.lisió de les bombolles amb les plataformes es fa com si fos una col.lisió entre dues caixes (com al joc original), ja que això fa molt més previsible la col.lisió i fa que el joc sigui una mica menys complicat.

5.3.2.4. Visualització

En aquest apartat és molt important destacar les ombres, que serveixen de punt de referència per al jugador (ja que en 3D és més complicat fixar-ne el punt de referència).

A més, hem de destacar que al final s'ha decidit fer les plataformes una mica transparents, per tal que no dificultin la visió del jugador quan passa per darrera d'elles.



Exemple de transparència per facilitar la visió del jugador

També s'ha fet la màscara de col.lisió de les bombolles una mica més petita que el seu tamany de render (aproximadament un 10% menys), ja que això facil.lita que el jugador pugui esquivar-les millor.

5.4. Base de dades 3D

Degut a la natura 3D del joc, els elements gràfics que hi trobem són models tridimensionals. Aquest models, com ja s'ha explicat, han estat generats amb el programa de disseny 3D *3D Studio Max* versió 5.

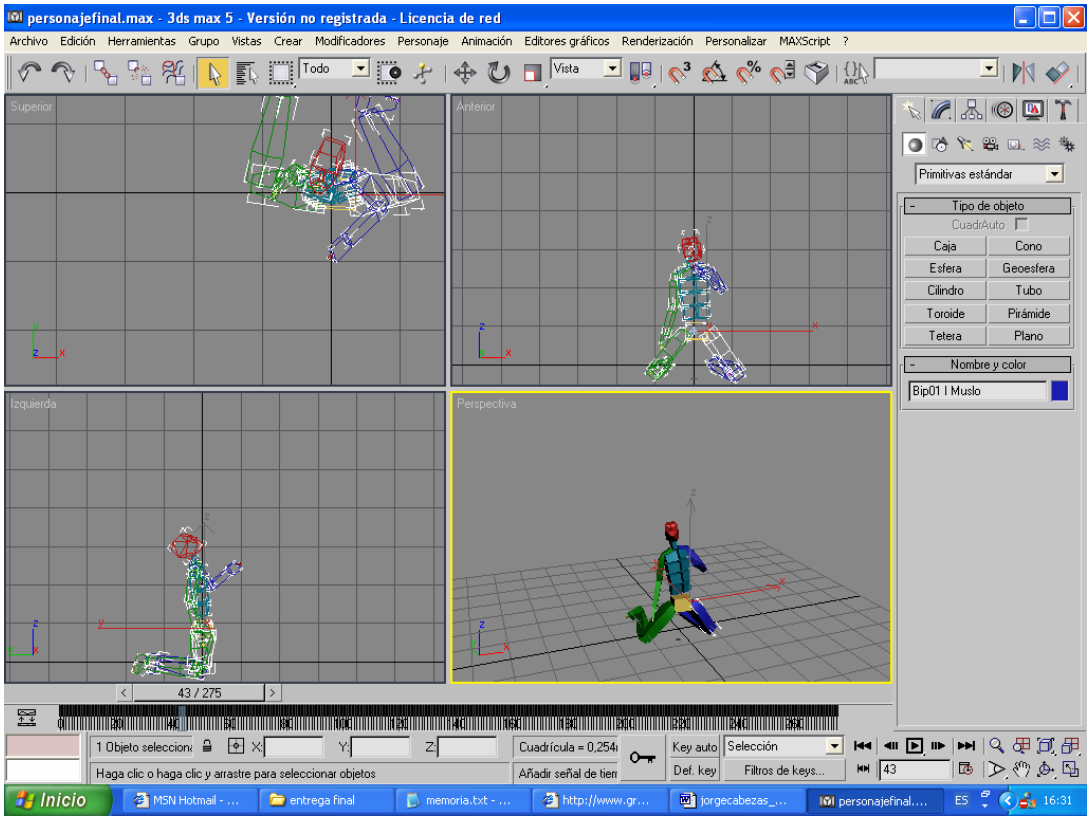
Bàsicament, els models utilitzats al joc han estat els següents:

El personatge

El model del personatge ha partit d'un model *biped* de *character studio* (un plugin del programa *3D Studio Max* que serveix per dissenyar caràcters).

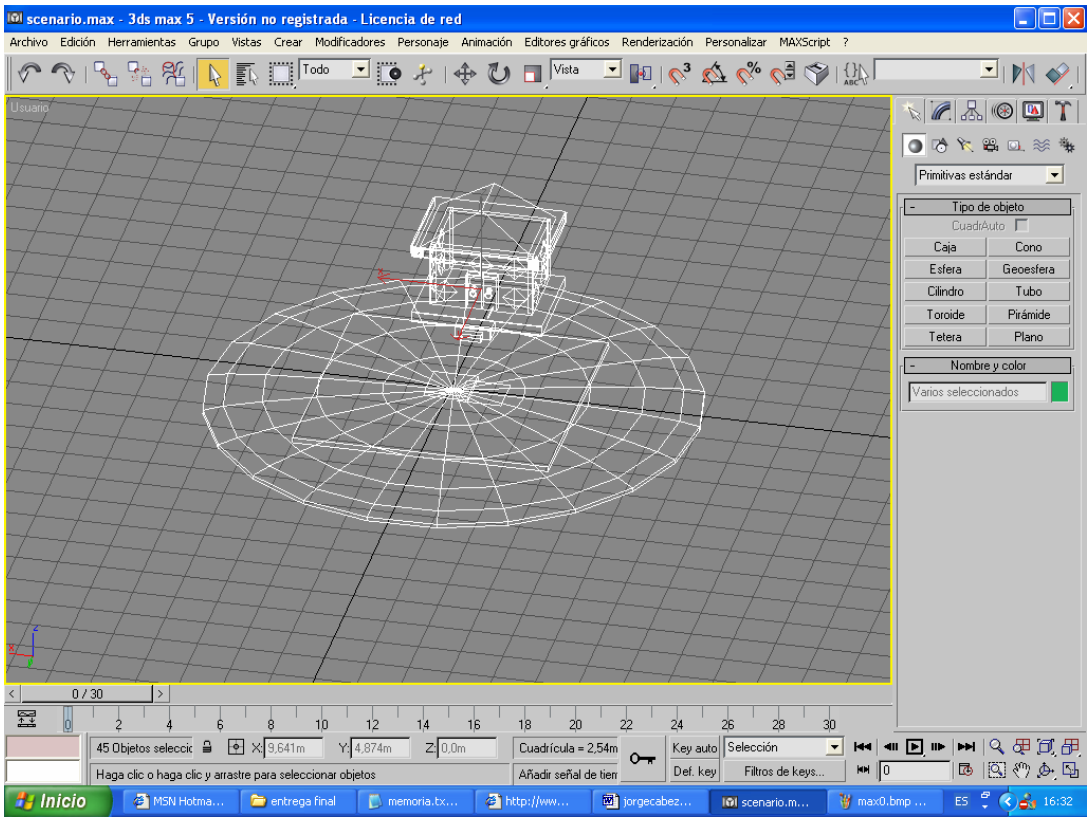
Aquest model conté les animacions de moviment i de morir, tot i que per problemes tècnics, degut a la incompatibilitat dels models tipus *biped* amb els models *3DS* (els interpretats per la

llibreria *lib3ds*), s'han hagut de passar aquestes animacions *frame* a *frame* mitjançant la transformació al format intermig *DXF* (un format utilitzat pel programa *AutoCAD*), i des d'aquest al format *3DS*.



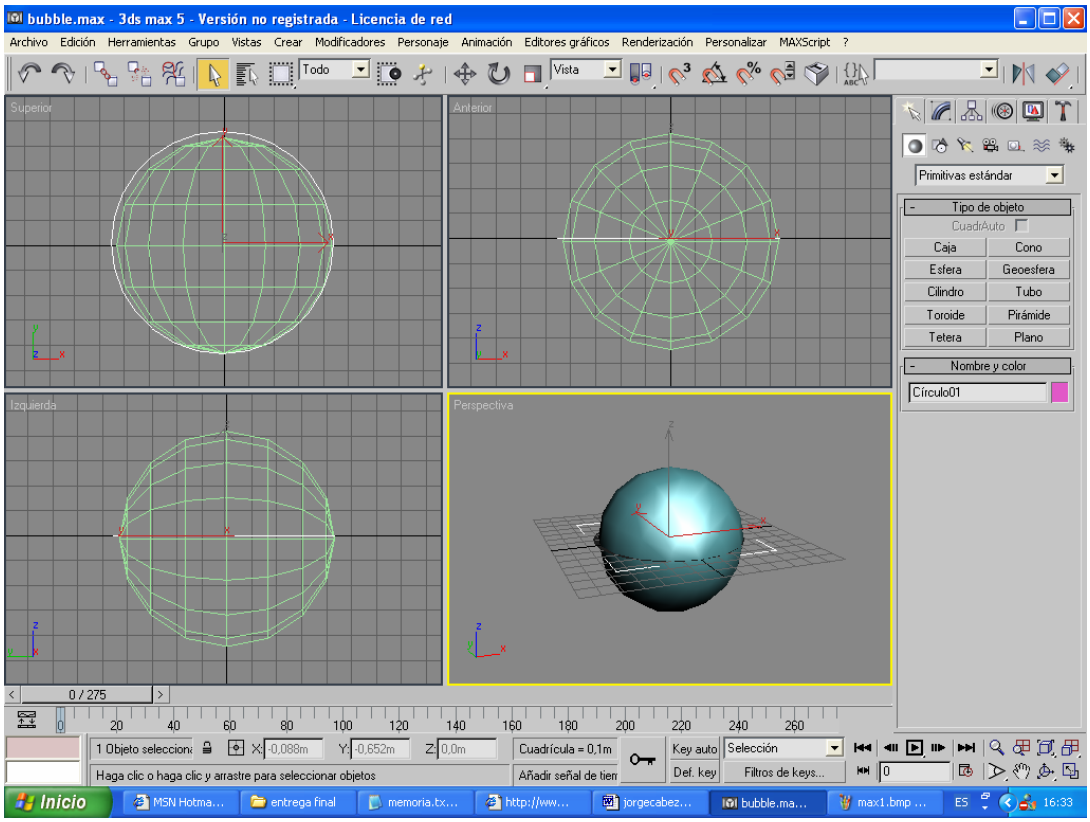
Un escenari

El model de l'escenari consisteix simplement en un pla amb una plataforma a on es desenvolupa tota l'acció de joc.



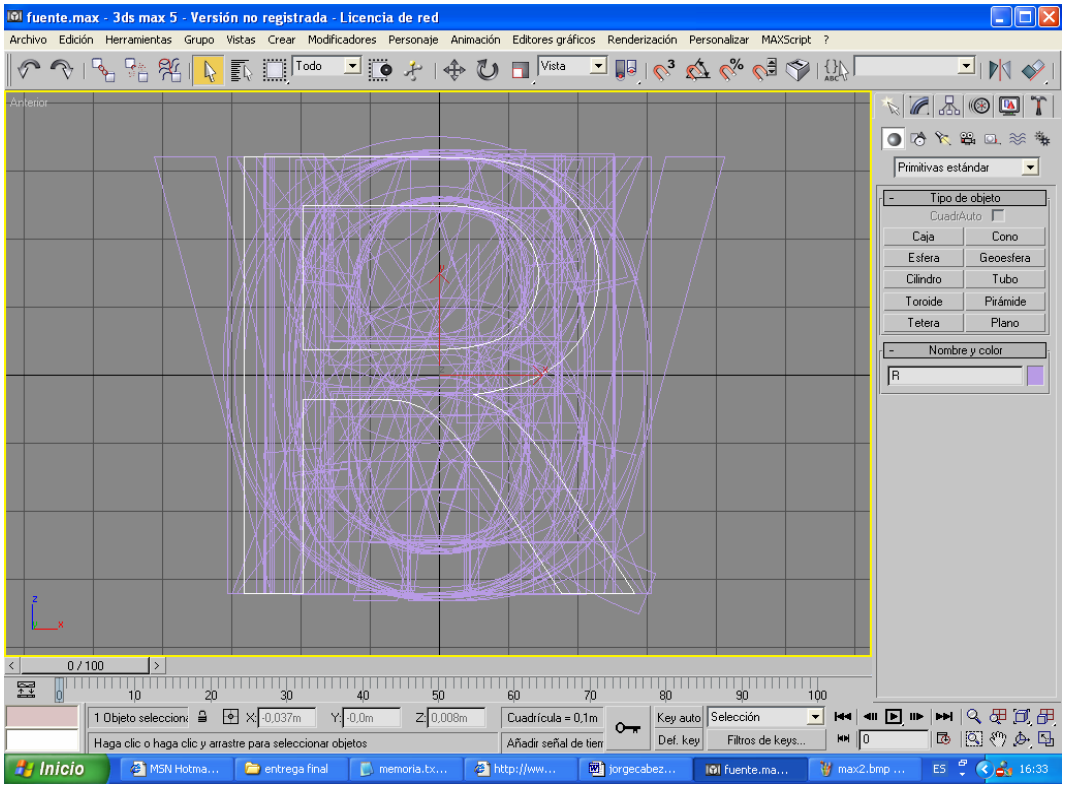
Una bombolla amb la seva ombra

Aquesta bombolla i la seva ombra es passen amb una única mida (que correspon a la bombolla més gran), ja que el seu tamany s'escala per programa.



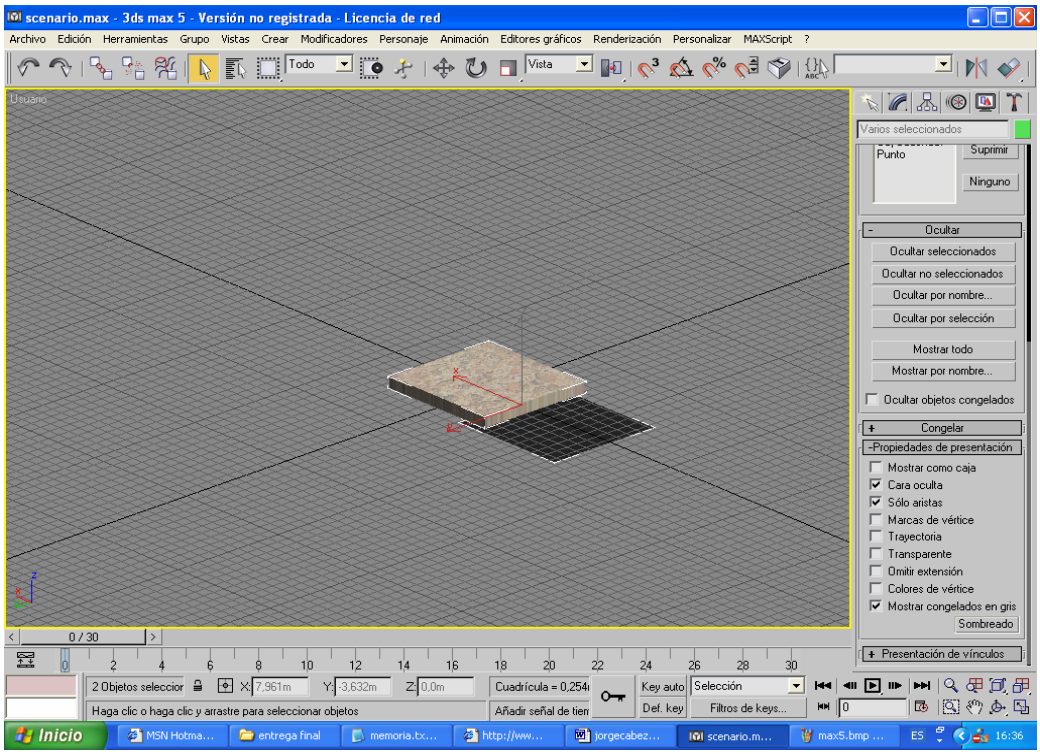
Una font de text

Aquesta font de text inclou tots els caràcters que es poden imprimir al programa.



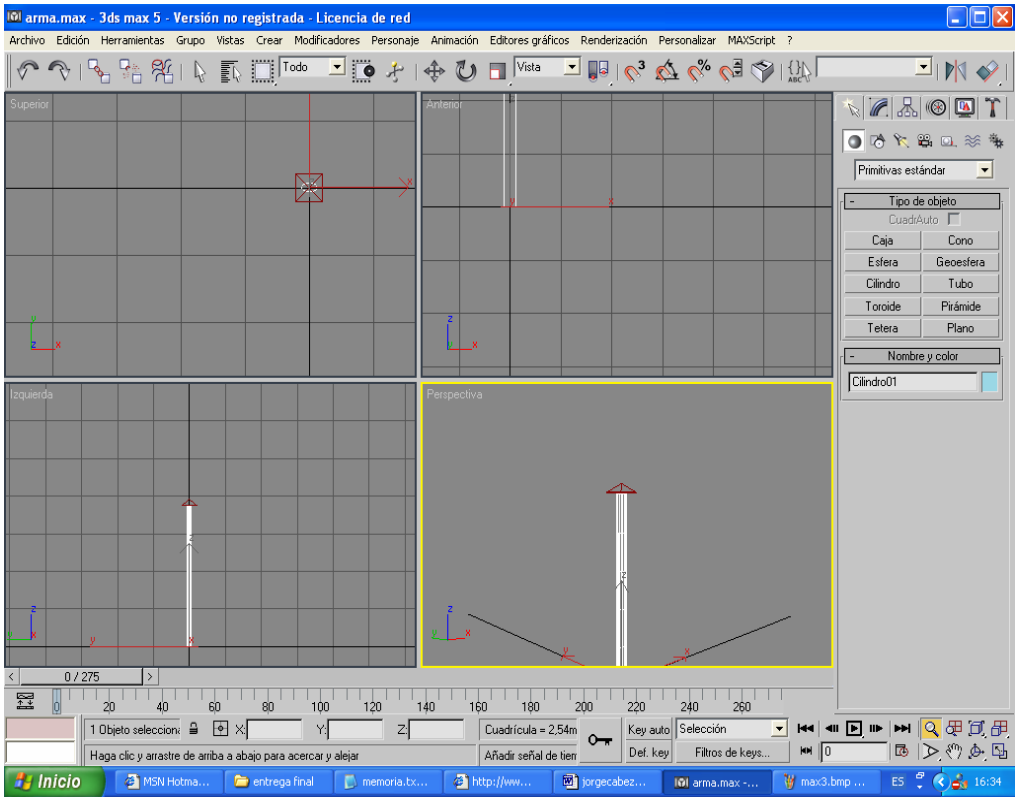
Una plataforma amb la seva ombra

La plataforma és simplement una caixa, i la seva ombra un rectangle.



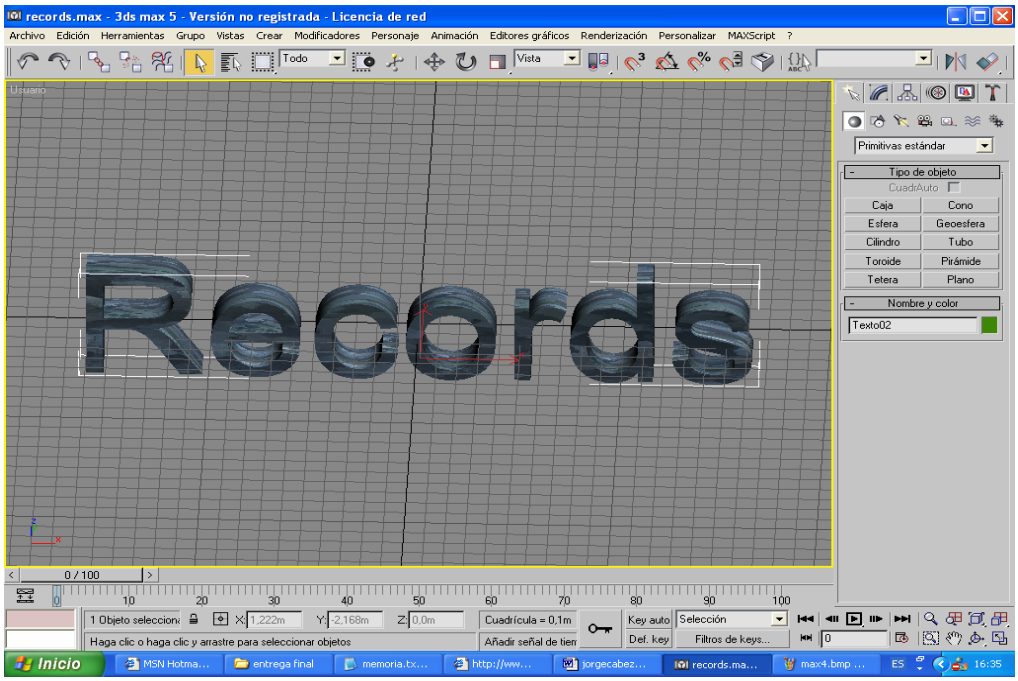
El tret del personatge

Aquest model del tret té una mida fixa, i en disparar-se el programa s'encarrega de escalar-lo per tal de que vagi ampliant-se en pantalla per formar un filament.



Alguns rètols addicionals

Aquests rètols inclouen el títol de la pantalla principal, així com el de la pantalla d'inserció de rècords.



6. Manual d'instal.lació i d'ús

6.1. Requisits mínims i recomanats

Els requisits mínims per a la correcta execució del joc són:

PC Pentium IV compatible o superior
128 MB de RAM
Tarjeta de Video 3D compatible openGL amb 16 MB de video RAM
3 MB de espai en HD
Sistema operatiu Windows XP

Tot i això, per a poder gaudir del joc en les millors condicions, es recomana el següent equip:

PC Pentium IV 2'66GHZ+
256 MB de RAM
Tarjeta de video 3D Nvidia Geforce4 amb 64 MB de video RAM
3 MB de espai en HD
Sistema operatiu Windows XP

6.2. Manual d'instal.lació

6.2.1. Instal.lació del joc

Per instal.lar el joc només hem de clickar sobre el fitxer *pang3dv10.exe*, que és el fitxer amb la distribució del joc, i indicar-li el directori a on volem instal.lar-ho.

Per executar el joc hem d'anar a aquest directori i clickar sobre el fitxer *pang3d.exe*, i el joc començarà la seva execució.

Per desinstalar-ho, és suficient amb borrar el directori a on vam fer la instal.lació.

6.2.2. Instal.lació del codi font

El codi font del joc es troba al fitxer *sources.zip* . Per instal.lar-ho, només hem de descomprimir el fitxer a un directori qualsevol, ja que tant el projecte del joc, com l'executable que genera i les dades necessàries per a la correcta execució utilitzen directoris relatius.

Una vegada descomprimit, només hem de carregar el fitxer *pang3d.dsw* al *visual studio* per poder compilar el projecte.

6.3. Manual d'ús

Al executar el joc passem directament al menú principal:

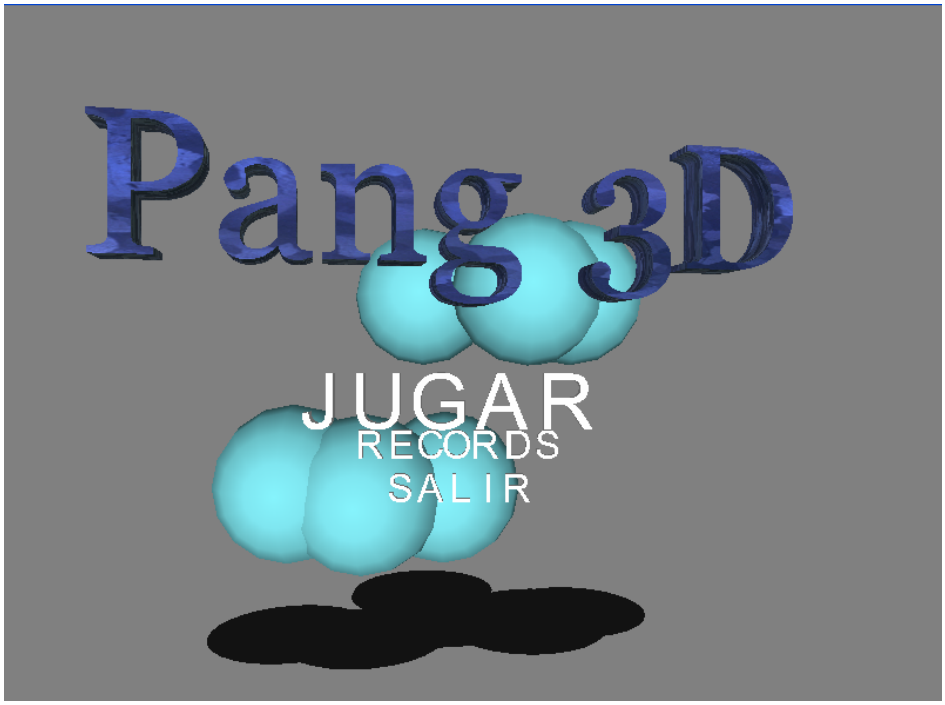
Menú principal

El menú principal dóna les opcions disponibles, que són *jugar*, *veure records* i sortir *del programa*.

La opció resaltada és la activa en cada moment, i es poden utilitzar les tecles de direcció *amunt* i *avall* per tal de modificar la selecció, i la tecla *espai* per seleccionar la opció activa.

Si es pulsa *jugar*, s'anirà a la pantalla de joc per començar una nova partida, mentre que si es pulsa *records* s'anira a la pantalla de visualització de rècords.

Finalment, la opció *salir* serveix perque es sorti del programa cap al sistema operatiu.

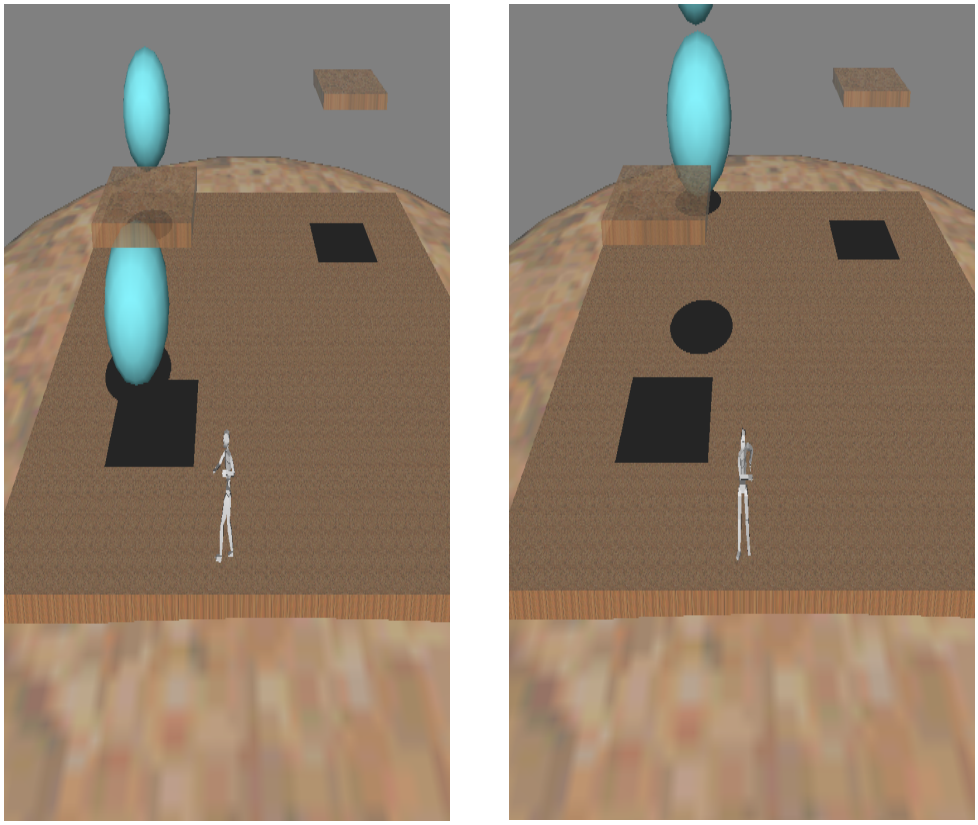


Joc

Objectiu del joc

L'objectiu del joc consisteix a esquivar les bombolles que hi ha en pantalla, i anar-les trencant. Cada vegada que trenquem una bombolla aquesta es divideix en dues, fins a un limit de divisions. Cada vegada que es trenca una bombolla s'aconsegueixen punts. Entre més petita és la bombolla, més punts dóna.

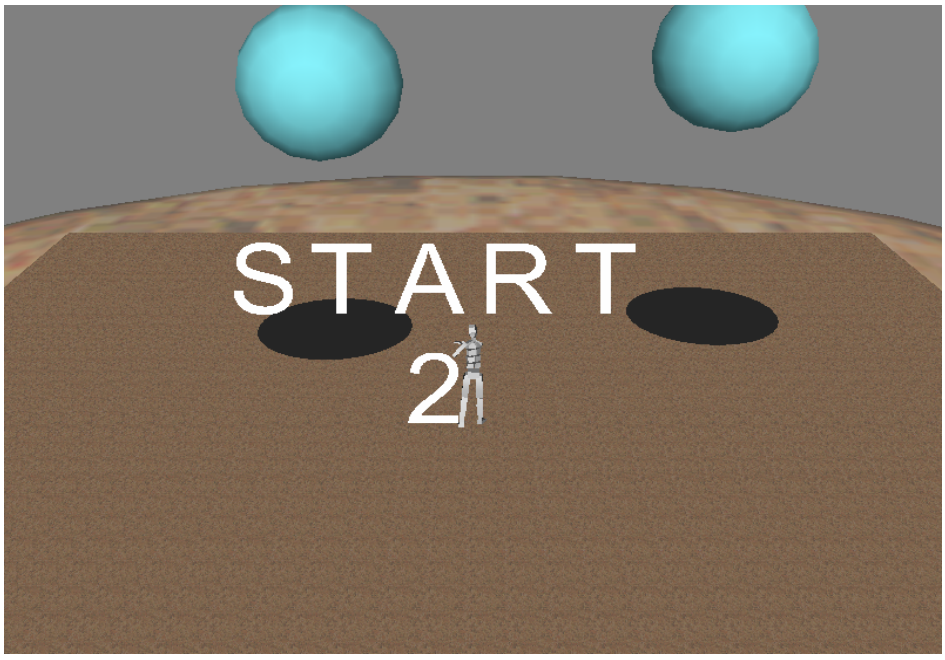
Les bombolles col.lisionen contra el terra, els límits de la plataforma de joc, i les plataformes. A més, l'alçada de bot de les bombolles es sempre la mateixa, tot i que es pot veure afectada per les plataformes de forma temporal:



En aquestes dues imatges es pot veure una bombolla col.lisionant amb una plataforma i després recuperant la seva alçada normal de bot.

Mecànica del joc

Al començar la partida, hi ha un missatge inicial amb un compte enrere de tres segons per tal de poder-se preparar abans de començar a jugar:

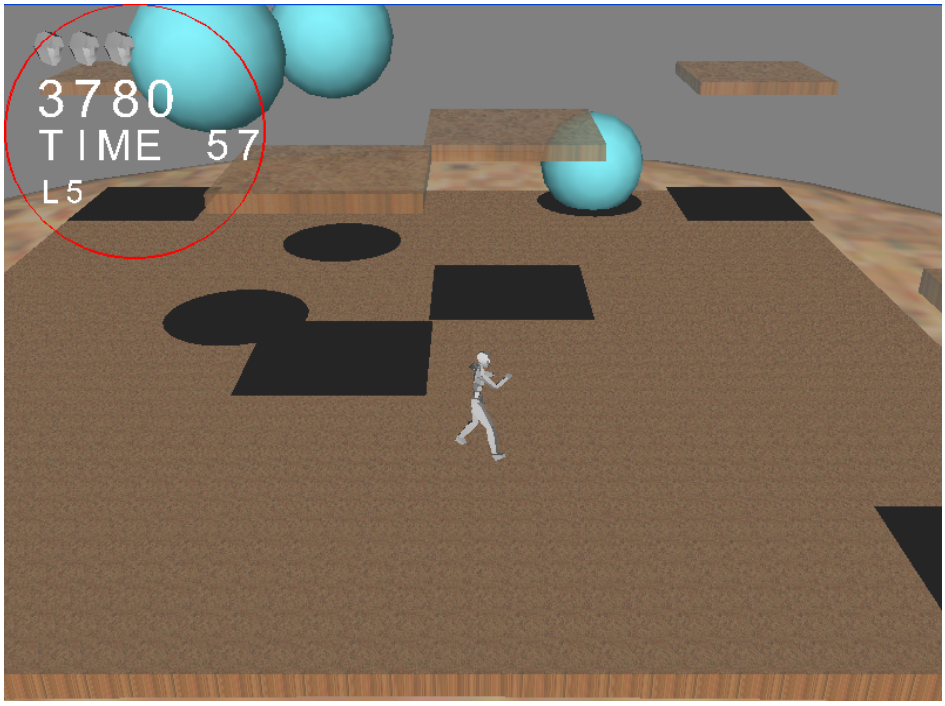


Després d'això es comença a jugar, tot i que durant 3 segons es dispossa de inmunitat (el personatge principal parpadeja).

Durant la partida hi ha les opcions de moure enrera, endavant, a l'esquerra o la dreta mitjançant les *tecles de direcció*, i també hi ha la possibilitat de disparar, amb la tecla *espai*. El personatge no es pot moure més enllà de la plataforma que delimita el àrea de joc.

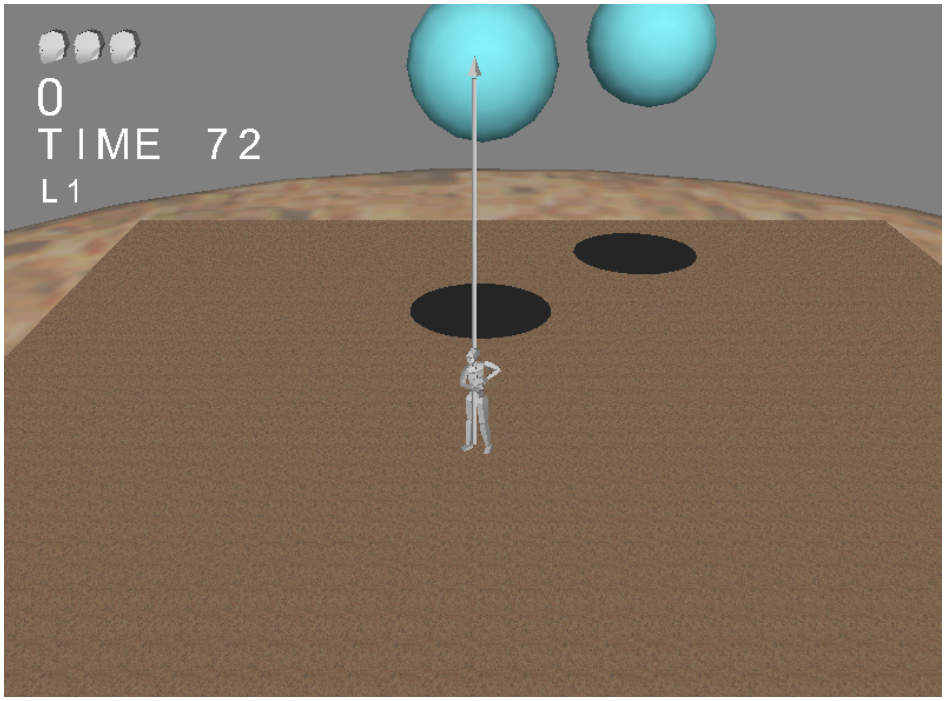
Marcadors

Els marcadors indiquen l'estat del joc en tot moment. Als marcadors trobem el nombre de vides restants, el nombre de punts, així com el nivell actual i el temps que manca per acabar la fase:



Trets

Els trets surten cap amunt des de la posició del personatge i només pot haver-hi un a la vegada. Un tret finalitza quan es trenca una bombolla, es toca una plataforma o s'arriba a una alçada màxima des de la posició del personatge (es triga mig segon en arrivar a aquesta alçada)



Pèrdua d'una vida i game over

Es perd una vida o bé quan el temps s'esgota o quan una bombolla col.lisiona amb el personatge.

En perdre totes les vides el joc finalitza i es mostra la pantalla de game over.

Si no s'han perdut totes les vides, es torna a començar en la mateixa fase, en el mateix estat en que s'havien deixat les bombolles, i amb el personatge en la mateixa posició.

Passar a la següent fase

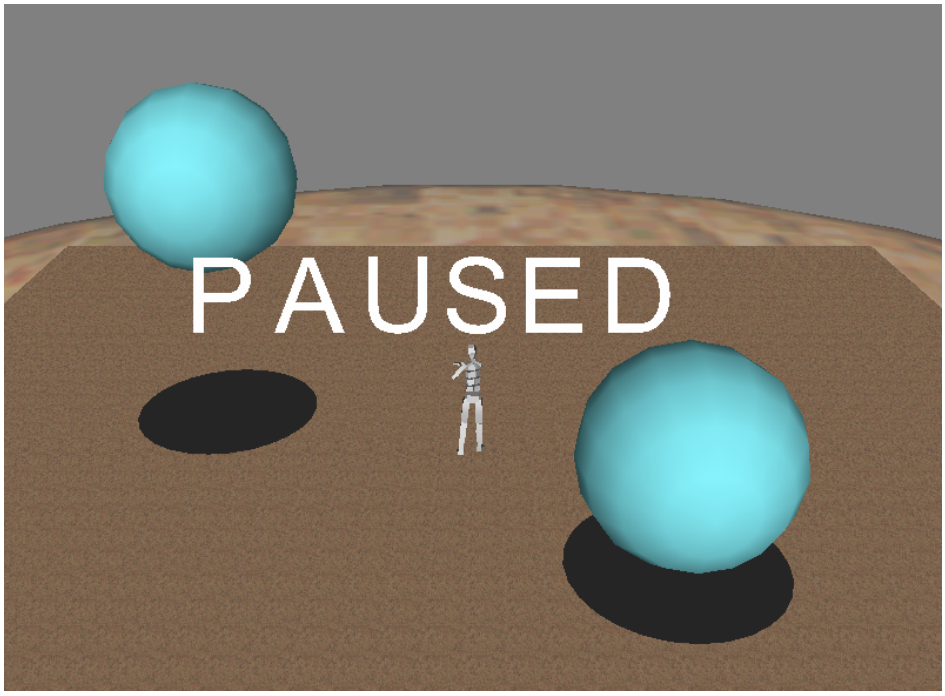
Per tal de passar a la següent fase s'han d'eliminar totes les bombolles que hi ha a la pantalla abans de que el temps s'esgoti. Quan això s'aconsegueix, es mostra un missatge en pantalla i es comença de nou a jugar en una fase amb un nivell de dificultat més elevat que l'anterior.

Vides addicionals

A mesura que es van aconseguint punts, es poden anar aconseguint vides extra. La primera vida extra és als 3000 punts, i després hi ha una vida extra cada 5000 punts.

Pausa

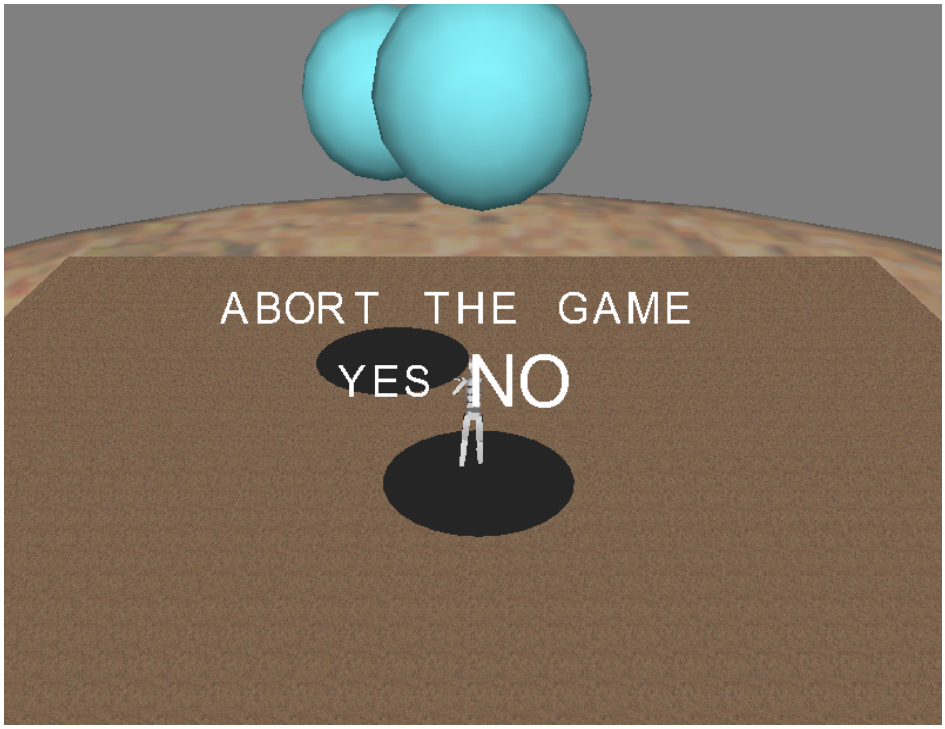
Mentre s'està jugant a una fase es pot pausar el joc pulsant la tecla *P*. Es mostrarà en pantalla un missatge que indica que s'està en pausa, i tornant a pulsar la mateixa tecla es continuarà jugant.



Abortament de partida

Mentre es juga, es pot pulsar la tecla *ESC* per tal d'abortar el joc.

En pantalla sortirà un menú que ens demana la confirmació. En aquest menú canviem entre les opcions amb les tecles de direcció *esquerra* i *dreta*, i confirmem amb la tecla *espai*:

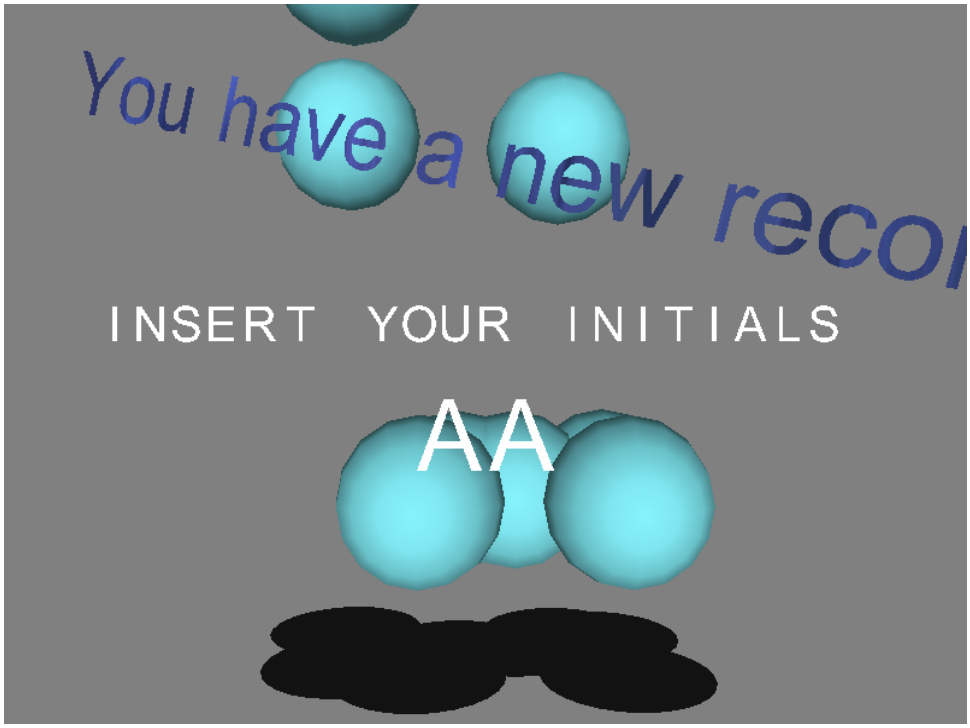


Inserció de rècords

A la pantalla d'inserció de rècords s'arriba quan s'ha aconseguit fer una puntuació que està entre les deu primeres.

A aquesta pantalla s'inserten les tres inicials del jugador, mitjançant les tecles de direcció, *esquerra* i *dreta*, que permeten que el jugador pugui anar variant cadascuna de les inicials. Les inicials es confirmen amb la tecla *espai*. La inicial que s'està canviant en cada moment és la que parpadeja.

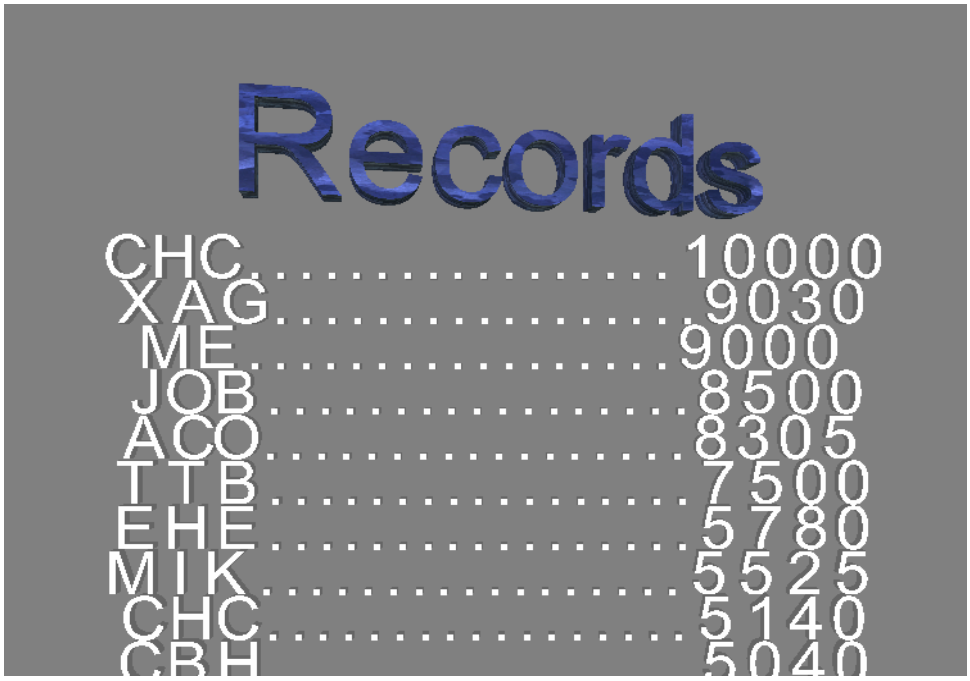
Una vegada confirmades les tres inicials, s'inserta el rècord i es passa a la pantalla de visualització de rècords.



Visualització de rècords

A aquesta pantalla s'arriba des del menú principal o des de la pantalla d'inserció de rècords.

Es surt cap al menú principal pulsant la tecla *espai*.



7. Memòria econòmica

El cost del projecte s'ha dividit en cost de maquinari, cost de programari i cost de desenvolupament.

Cost del maquinari

El cost de maquinari ha estat el d'un únic PC, que té un cost aproximat de 750 euros (Pentium IV, 2.66GHZ, 1GB RAM, 120 GB HD, monitor TFT)

Cost del programari

Microsoft Visual Studio 6.0 : descatalogat . En el seu lloc es posa el preu de Microsoft Visual Studio .NET Enterprise Architect 2003 : 2738 Euros. Font : acuista (*http://www.acuista.com*)

3D Studio MAX 7.0 : 4250 euros. Font : discreet (*http://www.discreet.com*)

La resta del programari (openGL, argoUML, glut i lib3ds) és gratuït.

Cost de desenvolupament

El temps de desenvolupament ha estat de aproximadament 3 mesos (60 dies laborals), a raó de 2 hores al día. Això ens dóna unes 120 hores de treball.

El temps de l'anàlisi ha estat des del 28 de setembre fins al 17 d'octubre, com es va detallar al pla de treball. Tot i així s'han requerit retocs i modificacions al anàlisi que fan que hagi estat aproximadament un mes de treball. La resta del temps ha estat dedicat a la programació.

Si comptem com a sou brut d'analista 24000 euros anuals, i de programador 18000 euros anuals, i sabent que el número d'hores anuals és aproximadament de 1800 hores (depend del conveni), tenim que cada hora d'analista surt a 13,3 euros , i de programador a 10 euros.

40 hores d'analista x 13,3 euros/hora : 532 euros
80 hores de programador x 10 euros/hora : 800 euros

Cost total

El cost total per tant serà:

cost maquinari + cost programari + cost desenvolupament: 750 euros + 4250 euros + 2738 euros + 1332 euros = 9070 euros

Evidentment, queda clar que el cost total del producte és tant alt pel cost excessiu del programari *3D StudioMAX* i *Visual Studio .NET* i del maquinari (PC). Sense aquests gastos extras, que només serien necessaris en la primera versió del programa, els costos són bastant ajustats.

8. Conclusions i treball futur

Conclusions

Com a conclusions, podem dir que els objectius al principi del projecte s'han complert de forma satisfactòria. El resultat final és un producte amb totes les opcions previstes (inclús algunes més), i amb tota la funcionalitat que es pot demanar a un programa d'aquestes característiques.

El temps d'anàlisi i desenvolupament han estat els fixats al pla de treball, amb desviacions mínimes. També, tal i com ja es va predir al pla de treball, la etapa d'anàlisi ha requerit de petits refinaments a mesura que el projecte anava avançant, tal com suceeïx al desenvolupament de qualsevol producte comercial actual.

A més s'ha de dir que el desenvolupament del codi ha seguit uns estandards el més alts possibles en quan a estructuració. Tot el codi està plenament comentat, segueix uns criteris de nomenclatura i documentació, i permet la futura ampliació de les funcionalitats de forma fàcil.

Hem de fer especial ènfasi en alguns aspectes com la jugabilitat, que en realitat és la part més important del producte final, ja que tot i que al pla de treball no vam poder mesurar quan de temps s'hauria de dedicar a aquest aspecte, en la pràctica podem dir que ha estat un temps considerable.

Tot i això, aquest és un aspecte (el de la jugabilitat) que requerirà un treball posterior en futures ampliacions.

Un altre aspecte a destacar és que ha hagut bastants problemes amb la llibrería *lib3ds*, ja que aquesta librería no està en res pensada per a poder optimitzar el temps de procés de les targes actuals, i en la pràctica ha donat molts problemes en quan a rendiment. Igualment, aquest és un problema que haurem de tractar també en futures ampliacions.

Treball futur

N'hi ha moltes futures ampliacions que podem fer al producte. Les podem dividir entre ampliacions de funcionalitat i ampliacions gràfiques.

Ampliacions gràfiques

Possibilitat de visualitzar personatges amb pell (*skinning*), bombolles amb materials de reflexió tipus *cubemap*, materials amb iluminació per pixel, *bump mapping*.

També seria molt interessant la optimització del codi 3D per tal de poder visualitzar més poligons i així obtenir escenaris més detallats.

Realització d'una llibreria 3D pròpia per poder prescindir de *lib3ds*.

Ampliacions de funcionalitat

Un dels aspectes més interessants seria afegir una opció per a dos jugadors. El codi actual del joc està pensat perque aquesta opció sigui relativament fàcil d'implementar mitjançant el *TAD player*.

En quan a jugabilitat, seria molt interessant poder afegir nous elements, com ara escales, bonus que rebria el jugador en esclatar les bombolles (com bonus de temps, punts extra, etc...)

Una altra opció recomanable seria afegir al joc un final, quan s'hagin passat un nombre determinat de fases, i poder donar en aquest cas un bonus especial al jugador per tal d'inscriure un rècord encara més alt.

9. Bibliografia

Arcade History Database© (*http://www.arcade-history.com/history_database.php?page=detail&id=1929*) : conté informació sobre el PANG original

http://www.mame.net/ : un emulador que conté la versió original de PANG

http://lib3ds.sourceforge.net/ : aquesta web conté els manuals així com el codi font del projecte *lib3ds*

http://www.opengl.org/resources/libraries/glut.html : informació sobre la llibreria *glut*

http://www.opengl.org : informació general sobre *opengl*, així com especificacions. Els *forums* són especialment interessants i de gran utilitat