EPS
Escola Politècnica Superior
UdG

Projecte/Treball fi de carrera

Estudi: ETIS. Pla 2001

Títol: Entorn de creació d 'exercicis per l' estimulació de la memòria i la psicomotricitat fina

Alumne/a: Jordi Artau Moreno

Director/Tutor: Immaculada Boada Oliveras
Departament: IMA
Àrea: LSI

Convocatòria (mes/any): 09/2012

EPS
Escola Politècnica Superior
UdG

Projecte/Treball Fi de Carrera

Estudi: Eng. Tècn. Informàtica de Sistemes. Pla 2001

Títol: Entorn de creació d 'exercicis per l' estimulació de la memòria i la psicomotricitat fina

Document: Memòria

Alumne: Jordi Artau Moreno

Director/Tutor: Immaculada Boada Oliveras
Departament: Informàtica i Matemàtica Aplicada
Àrea: LSI

Convocatòria (mes/any): 09/2012

Índex

1 Introducció

1.1 Antecedents

La demència es caracteritza per una sèrie de dèficits cognitius múltiples que impliquen un deteriorament de la memòria. Entre els símptomes de la demència hi ha el deteriorament de la memòria i la pèrdua de la psicomotricitat fina. Per tractar aquestes pèrdues, una vegada fet el diagnòstic, els pacients poden assistir a tallers en els quals es plantegen diferents tipus d'exercicis per tal de recuperar o evitar incrementar la pèrdua d'aquestes facultats. Els exercicis que es realitzen en aquests tallers els preparen els diferents professionals especialitzats que hi treballen. Preparar les activitats pot ser un procés laboriós i és per aquest motiu que hi ha poca varietat d'exercicis.

La psicomotricitat fina determina la capacitat de realitzar moviments els quals requereixen certa precisió i control. La pèrdua d'aquesta capacitat es pot frenar o alentir amb la realització d'exercicis que estimulen aquestes extremitats (normalment mans i dits).

La falta de software per l'ajuda i seguiment del pacients amb demència ha fet que es plantegi desenvolupar un sistema que permeti als professionals de l'àmbit obtenir una eina que els faciliti la recuperació d'aquests pacients, a més d'estimular-los per a millorar la psicomotricitat fina.

Per tal d'exercitar aquesta capacitat es recomana realitzar exercicis on es combini la capacitat de concentració i precisió de les extremitats tals com:

- Realitzar construccions amb blocs.
- Obrir i tancar ampolles.
- Recollir objectes petits.
- Passar pàgines.
- Trossejar papers.
- Fer boles de plastilina.

- Pressionar tecles amb els dits.
- Dibuixar o escriure.
- Retallar amb tisores.
- Passar objectes petits per dins de forats.
- Tocar instruments musicals (sobretot de tecles).
- ...

En tots els exemples anteriors es pot observar que la capacitat de concentració i/o precisió son necessàries per a portar a terme qualsevol de les tasques.

Manipular el teclat o el ratolí d'un ordinador requereix una capacitat motriu mínima per tal d'utilitzar-los amb certa precisió, per tant, es poden aprofitar aquestes eines tan habituals per estimular-la.

A més de l'habilitat d'usar el teclat i ratolí d'un ordinador, també s'ha de tenir en compte la capacitat del pacient a concentrar-se per assolir el repte que li planteja el joc en qüestió, d'aquesta manera el pacient està estimulant tant la concentració com la precisió amb els dits i les mans.

Un cop trobada l'eina es planteja el repte de desenvolupar un software que faci que el pacient hagi de completar un seguit d'accions sobre el teclat i moviments amb el ratolí per tal d'assolir un repte.

Aprofitant els motors gràfics disponibles que hi ha avui en dia es poden realitzar tot tipus d'aplicacions i jocs que poden ser útils per fer front al problema.

Aquest projecte pretén unir la capacitat del motor gràfic i l'experiència mèdica per desenvolupar un aplicatiu que permeti entrenar la capacitat psicomotriu fina amb la finalitat d'exercitar-la. Al mateix temps se n'extreuen resultats i seguiments útils perquè els equips professionals puguin obtenir-ne unes valoracions.

1.2 Objectius

L'objectiu d'aquest projecte és explotar la tecnologia de videojocs per crear un entorn que permeti dissenyar diferents exercicis que potenciïn la memòria i la psicomotricitat fina. Els exercicis consistiran en la visualització d'un circuit en 3 dimensions pels quals el pacient haurà de fer circular un objecte. L'objectiu és desenvolupar un entorn que permeti crear qualsevol tipus de circuit, elaborar exercicis i poder-los puntuar de forma automàtica.

El sistema ha de permetre la creació de circuits en 3D indicant-ne una trajectòria i definint els gruixos de cada tram d'aquesta.

Aquests circuits han de poder-se guardar, editar i classificar segons la dificultat que l'expert cregui convenient.

Un cop definit el circuit, l'usuari pacient ha de poder dirigir o conduir un objecte pel seu interior mitjançant el teclat i/o el ratolí, controlant a temps real les errades i extreure'n un resultat final.

Els objectius d'aquest projecte queden agrupats en dues fases:

a) La primera s'assolirà amb les necessitats de l'usuari constructor (equip professional) que serà bàsicament el conjunt d'eines que permetrà crear, editar i guardar el circuit creat, i també l'anàlisi dels resultats de les accions del pacient.

b) La segona fase consistirà en dissenyar un entorn on el pacient sigui capaç de dirigir l'objecte per l'interior del circuit dissenyat per l'usuari constructor, mentre es guarden les seves dades de conducció.

Per tant, per assolir aquests objectius caldrà dissenyar i implementar un entorn que permeti:

a) Definir camins o trajectòries en 3D de qualsevol forma i amb amplades diferents a cada tram.

b) Creació d'un circuit amb 3D a partir de la trajectòria.

c) Moure l'objecte per l'interior del circuit marcant els llocs en els que surt de la trajectòria, el temps que tarda en seguir el camí, etc.

d) Guardar els camins de manera que es puguin editar i classificar en funció de la seva dificultat.

e) Guardar la solució que ha proposat el pacient a l'exercici.

f) Reproduir les accions que ha realitzat el pacient per poder analitzar-ho posteriorment.

1.3 Estructura de la memòria

La memòria del projecte està estructurada de la següent manera:

a) **Introducció -->** En aquest apartat s'introdueix en el marc del projecte, s'explica què és la demència i la pèrdua de la psicomotricitat fina, i com la falta de software per a la recuperació dels pacients motiva a desenvolupar un software que ajudi als equips professionals a tractar i seguir aquest afectats.

b) **Estudi de viabilitat -->** El punt en qüestió fa un estudi de la viabilitat del projecte, eines, materials, software i l'accessibilitat dels diferents usuaris al material descrit.

c) **Metodologia -->** El sistema de treball és el que s'argumenta en aquesta secció, mètodes de programació, estructura de treball i possibles variacions que puguin anar sorgint al llarg del projecte.

d) **Planificació -->** Aquí es podrà observar el calendari planificat del passos que es seguiran abans de començar la implementació i estudi del projecte.

e) **Marc de treball -->** És l'apartat on s'exposen els problemes que es volen resoldre, així com els motius pels quals s'ha decidit escollir aquest tema.

f) **Requisits del sistema -->** Aquesta secció llista els requisits que ha de complir el sistema per tal d'assolir els objectius del projecte.

g) **Estudis i decisions -->** Es descriu cada una de les eleccions que s'han pres al llarg del treball (llibreries, programari, solucions...).

h) **Anàlisi i disseny del sistema -->** Aquest apartat s'encarrega de detallar

i argumentar el model de dades i processos, i posteriorment, descriure les interfícies d'usuari, models d'objectes, processos ...

i) **Implementació i proves -->** En aquesta secció s'exposen els problemes sorgits durant el curs del projecte i les seves solucions, així com les classes i algorismes més rellevants.

j) **Implantació i resultats -->** Els resultats del projecte es mostren en aquest punt, de manera que els objectius es veuran assolits amb els exemples i proves exposats.

k) **Treball futur -->** Les possibles modificacions i ampliacions del projecte s'expliquen en aquest punt.

l) **Bibliografia -->** Aquí s'exposen les referències, articles, projectes..., els quals han estat consultats al llarg del projecte.

2 Estudi de viabilitat

La viabilitat del projecte depèn bàsicament de tres punts: les necessitats del usuaris afectats per la demència, els objectius a assolir en el software i la maquinària necessària amb els seus requisits.

2.1 Psicomotricitat fina

Tal i com s'explica a l'apartat d'antecedents per exercitar la psicomotricitat fina, es recomanen realitzar exercicis que requereixin concentració i precisió sobre certes extremitats. Per conseqüent, l'aplicació haurà de requerir als seus usuaris que portin a terme aquest tipus d'accions.

Per tal de treballar aquesta qualitat, els exercicis proposats en el projecte es basen en conduir certs objectes per unes trajectòries definides per un equip de professionals, de manera que l'usuari tingui l'obligació de concloure aquesta tasca amb el màxim de precisió possible.

Els ordinadors personals disposen de perifèrics que poden ajudar a entrenar la psicomotricitat ja que l'ús del teclat i ratolí requereixen de certa precisió amb les mans i dits, de forma que podem aprofitar aquest hardware per la finalitat del treball.

L'objectiu d'aquest exercici és que el pacient es vegi motivat per assolir el repte que se li planteja de la millor manera possible, per tal que vegi l'aplicació com un joc. Així el pacient deixa de percebre la recuperació com un seguit d'exercicis avorrits i sense finalitat, i ho veu com un videojoc més entretingut i divertit.

2.2 Requeriments

Veient els objectius exposats es pot observar que el sistema és molt semblant a qualsevol tipus de videojoc en molts aspectes: definició d'una escena, selecció d'un vehicle, conducció d'aquest, guardar escenes i veure estadístics. Per tant, el millor que es pot escollir pel desenvolupament d'aquest és un motor

gràfic estàndard amb 3D pel desenvolupament de videojocs.

A l'utilitzar tecnologia de videojocs la viabilitat del projecte no ha de ser un motiu de preocupació, doncs els motors gràfics actuals estan provats i funcionant sense problemes a un gran nombre de plataformes.

Simplement s'ha de tenir en compte quines plataformes volem abarcar i triar el motor gràfic que permeti adaptar el sistema als diferents dispositius.

A nivell de software només és necessari disposar de l'entorn de programació del motor gràfic seleccionat, i per l'usuari final un sistema operatiu que accepti aquest motor.

I a nivell de hardware és necessària la disposició d'un equip informàtic pel desenvolupador, amb uns requisits mínims per executar l'entorn de programació que requereix el motor gràfic i les seves execucions a nivell de test.

Per aconseguir que el projecte sigui viable a nivell d'usuari, aquest haurà de disposar de maquinari informàtic (ordinador personal amb perifèrics tals com teclat, ratolí i pantalla), de manera que els usuaris puguin interactuar amb el sistema.

2.3 Valoració final

Els requisits pel desenvolupament del projecte són perfectament viables, tenint en compte que les eines descrites anteriorment no necessiten grans pressupostos ni maquinari exclusiu o de difícil accés. A nivell de desenvolupador, el maquinari pot ser qualsevol ordinador personal amb una potència gràfica mitjana, per tal de treballar amb fluïdesa, mentre que a nivell d'usuari qualsevol ordinador no obsolet ha de ser capaç de resistir el sistema sense grans dificultats.

3 Metodologia de programació

Per la implementació del projecte s'ha escollit un tipus de programació àgil, que en aquest cas presenta una sèrie d'avantatges sobre els mètodes de programació tradicionals.

Les metodologies tradicionals son útils per equips de programació amb un gran nombre de programadors a l'equip, on s'han de precisar molt cada un dels processos de desenvolupament i són molt reàcies a modificacions. Per contra, les metodologies àgils estan més orientades a equips petits, on la comunicació entre els seus components és molt més fluïda, i la planificació del processos es va definint constantment durant el desenvolupament del projecte.

Tot i que les metodologies àgils estan pensades per portar-se a terme en equips amb un nombre reduït de components, també es poden adaptar a projectes amb un sol programador, com és el cas del present treball.

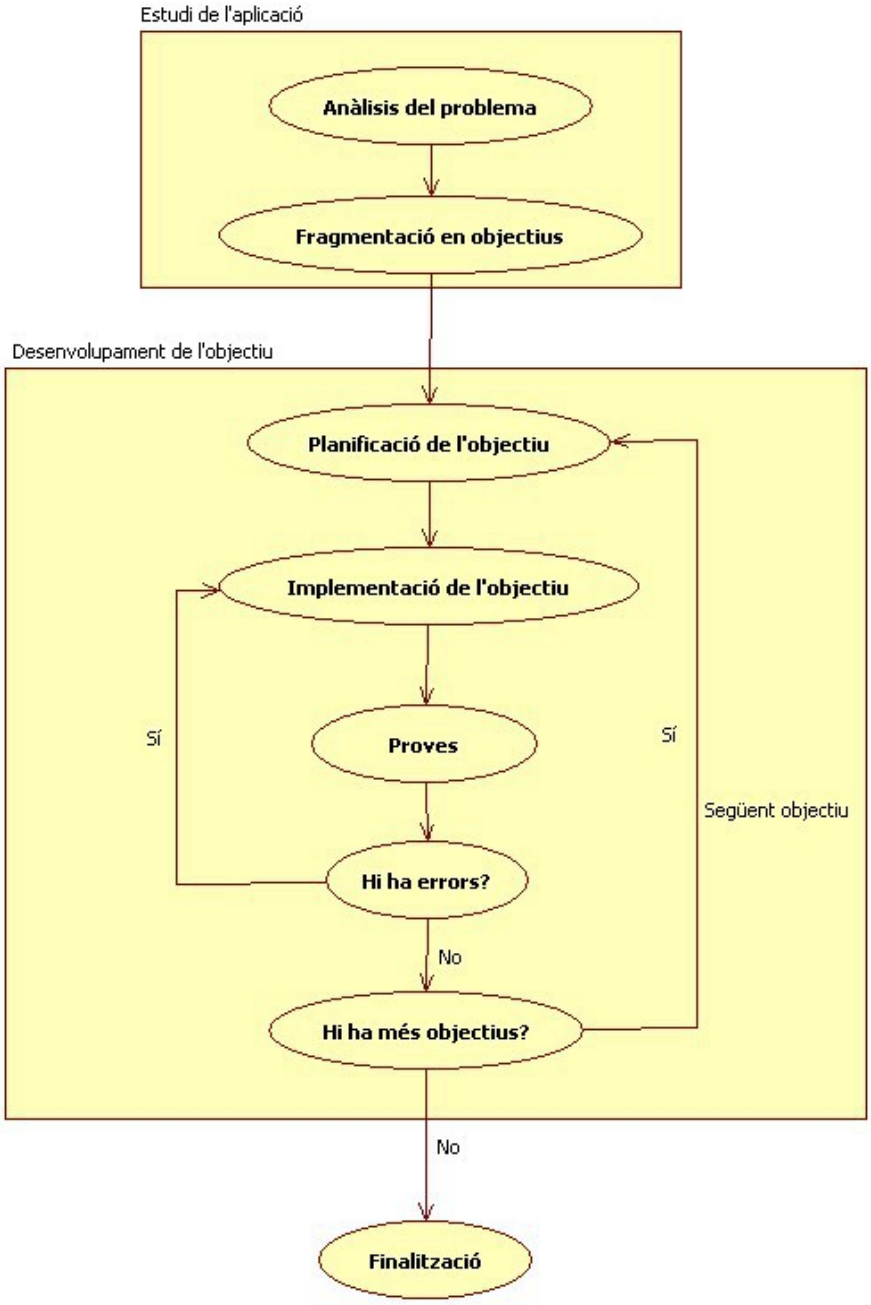


Figura 1. Esquema de desenvolupament

El procés de desenvolupament es realitzarà agrupant els objectius en etapes i, a mesura que aquestes es finalitzin, es testegin i s'acceptin, es procedirà a realitzar el mateix procés per la següent etapa.

3.1 Programació extrema (XP)

El mètode que s'escau més a la planificació és la programació extrema, que consisteix en afegir petites funcionalitats una darrera l'altre a mesura que s'assoleixen els diferents objectius de l'aplicació.

El motiu pel qual s'ha escollit aquest mètode és perquè els objectius del projecte són clars i viables, però a mesura que s'avança amb la programació van sorgint nous subobjectius, millores o rectificacions, que faran que aquest software sigui més usable i entenedor per l'usuari.

3.2 Característiques de l'XP

La programació extrema és una bona eina per elaborar aquells projectes on els usuaris finals no acaben de tenir clar fins a quin nivell d'usabilitat i utilitat arribarà el programa. En aquest cas els objectius són concrets, però a mesura que es vagin assolint les diferents fites marcades, aquestes podran tenir variacions per tal de fer l'aplicació més usable i útil pels usuaris.

Les característiques principals de la programació extrema són:

a) Desenvolupament iteratiu i incremental. Aplicar petites millores consecutivament.

b) Proves continuades. Realització de molts testos i a vegades repetitius.

c) Programació en parelles. El desenvolupament es realitza en parelles de forma conjunta i amb un alt grau de comunicació.

d) Integració de l'equip de programació amb el client. El client té contacte directe i molt freqüent amb els programadors.

e) Refractorització. El codi és corregit i millorat de manera continuada per tal d'optimitzar-lo i fer-lo més entenedor.

f) Propietat del codi compartida. Els desenvolupadors s'intercanvien el codi perquè tot l'equip en sigui coneixedor.

g) Simplicitat. El codi ha de ser simple i entenedor per tal que tothom el pugui entendre.

3.3 Adaptació XP al nostre cas

Per adaptar el codi a aquesta manera de treballar s'hauran d'aplicar algunes modificacions tenint en compte que només hi ha un programador al projecte. Aquestes són les adaptacions per tal de portar a terme la metodologia XP al projecte:

a) Programació en parelles. A l'haver-hi un sol component a l'equip de programació aquest punt no es podrà aplicar, tot i que el codi es pensarà sempre perquè el puguin llegir terceres persones.

b) Integració de l'equip de programació amb el client. El rol del client serà realitzat amb el tutor i l'alumne, que seran els que decidiran sobre la marxa el nivell d'usabilitat.

c) Propietat del codi compartida. Aquest punt no es podrà dur a terme ja que a l'equip només hi ha un programador.

Les proves en el codi del projecte seran de forma continuada, cada ampliació d'usabilitat serà provada per verificar el seu correcte funcionament i no s'avançarà cap al següent objectiu fins que aquest estigui funcionant al 100%.

La refactorització també serà present en el treball, a mesura que el projecte creixi es reestructuraran parts del codi per poder tenir un sistema ordenat, entenedor i ben estructurat. Aquestes modificacions es poden observar en la documentació.

A banda de les possibles variacions que puguin anar sorgint, es començarà dissenyant un nucli robust i totalment funcional d'edició de circuit, al que poc a poc, se l'hi afegiran les diferents funcionalitats per tal d'assolir l'objectiu global. La programació extrema és un dels millor mètodes per treballar d'aquesta manera.

Malgrat aquest sistema de treball està pensat per equips, el projecte s'hi adaptarà el màxim possible.

Els diferents rols dels components de l'equip els realitzarà l'únic programador participant al projecte juntament amb el tutor. Els rols seran:

Rol	Participant
Programador	Alumne
Client	Tutor
Entrenador	Tutor / Alumne
Rastrejador	Alumne
Tester	Alumne

La història de la implementació es podrà trobar emmagatzemada en les diferents versions del projecte, ordenades per dades amb una breu explicació de les millores i problemes.

4 Planificació

Per poder assolir l'objectiu s'ha planificat la feina en cinc fases. Aquestes són:

a) Anàlisi del problema. Enumeració de les necessitats i definició dels objectius.

b) Aprenentatge de l'entorn de treball. Valoració i anàlisi de les eines que es poden utilitzar per portar a terme l'implementació.

c) Disseny de l'aplicació. Definició dels requisits que l'aplicació ha d'assolir, definició de l'entorn de l'usuari i plantejament dels algorismes i els seus fluxos.

d) Implementació. Desenvolupament de l'aplicació en funció de l'anàlisi. Durant aquest procés es poden trobar problemes i limitacions que obligaran a tornar a l'apartat anterior per fer un replantejament de les solucions preses.

e) Documentació. Realització de la memòria del projecte. Aquesta tasca es desenvoluparà durant tot el projecte. Aquí és on s'aniran registrant totes les accions i decisions preses durant el curs del treball.

Les quatre primeres fases són necessàries per poder implementar el projecte i pel desenvolupament directe del codi font. Per tant, es faran seqüencialment.

El temps planificat per a la realització del projecte és de 6 mesos. Considerant que cada mes te quatre setmanes la temporalització planificada és la següent:

		Tasques				
		Anàlisi del problema	Aprenetatge de l'entorn	Disseny de l'aplicació	Implementació	Documentació
Setmanes	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					

Figura 2: Calendari

5 Marc de treball

La demència és una síndrome adquirida produïda per una patologia orgànica, psíquica o mixta, que, en un pacient sense alteració de nivell de consciència, ocasiona un deteriorament persistent de diverses funcions. Aquest projecte es centra en la funció basada en la psicomotricitat, concretament la fina.

La pèrdua d'aquesta capacitat provoca que els afectats tinguin dificultats per realitzar certes accions que requereixen concentració i precisió.

Actualment hi ha molta varietat de jocs i activitats físiques pensades per equips professionals que ajuden a la recuperació o a frenar aquestes afectacions, però moltes requereixen d'un equip de seguiment constant per als pacients i també, d'una gran quantitat de materials i infraestructures. Aquest projecte pretén ajudar a reduir aquests costos i aportar noves solucions.

5.1 Exemples d'exercicis

Actualment existeixen alguns jocs de seguiment de camins que exerciten o valoren la psicomotricitat fina, un exemple clar és el típic joc de l'examen mèdic del permís de conducció, que consisteix en conduir dos elements per dues carretes diferents, i segons la puntuació realitzada superes o no la prova.

Altres exemples són els jocs infantils, sobretot jocs realitzats amb Flash sobre navegador web, a la pàgina web del “proyecto agrega” se'n poden trobar alguns:

http://contenidos.proyectoagrega.es/visualizador-1/Visualizar/Visualizar.do?idioma=es&identificador=es_2008041013_0302400&secuencia=false#

http://contenidos.proyectoagrega.es/visualizador-1/Visualizar/Visualizar.do?idioma=es&identificador=es_2009042813_7300490&secuencia=false

Aquests jocs es basen sobretot en valorar la precisió que té l'usuari sobre el ratolí de l'ordinador i la majoria consisteixen en seguir camins o figures, tot i que n'hi ha d'altres que s'ha de fer punteria.

Figura 3: Dibuixar línies

Figura 4: Seguiment de camins

Moltes de les activitats per ordinador que existeixen actualment permeten resoldre l'exercici però no estan pensades per fer-ne un seguiment. No tenen finalitat mèdica, l'usuari obté una puntuació i aquests resultats no queden guardats ni tampoc es poden usar per tenir un seguiment del pacient.

Les aplicacions de les quals es parla són aplicacions tancades, és a dir, es presenta un laberint però no hi ha possibilitat d'editar-lo per canviar la seva forma i adaptar-lo als interessos.

L'aplicació que es proposa ha de solventar aquestes limitacions, permetent personalitzar els exercicis per adaptar-los als pacients i poder tenir un control de com s'han resolt.

5.2 Avantatges dels jocs virtuals

Actualment les sessions de rehabilitació clàssiques presenten alguns problemes pel què fa a la seva realització tals com:

1. Exercicis presencials que obliguen als pacient a desplaçar-se sovint.
2. Seguiment del pacient per part del professionals de forma individual.
3. Gran volum de pacients, que implica:
    3.1. Saturació dels centres.
    3.2. Disminució dels recursos per individu.

3.3. Cues d'espera.

3.4. Major despesa per part de centres públics.

4. Nombre de professionals limitat.
5. Temps de espera per anàlisis i valoracions.
6. Exercicis monòtons i sense al·licient.

La majoria d'aquests problemes es deuen a la falta de personal i recursos de les entitats, ja que el seguiment d'un pacient a nivell individual comporta molta feina.

Amb aquest programa es proposa la informatització d'una part d'aquests exercicis, per tal de disminuir temps i recursos, així com alliberar càrrega als equips de professionals, millorant la qualitat dels serveis mèdics.

Alguns dels avantatges que presenta usar un software pel seguiment i recuperació dels pacients són:

5.2.1 Seguiment massiu i valoració instantània

Per obtenir un bon seguiment dels pacients durant la realització d'exercicis és necessària la presència d'un professional, per tal que aquest extregui un resultat de l'exercici i ho anoti per obtenir-ne uns resultats amb sessions anteriors. Això implica que el nombre de pacients que pot controlar un educador és limitat, doncs quants més usuaris ha de controlar més informació pot perdre i, de menor qualitat serà el seu seguiment.

L'avantatge d'usar una aplicació és que l'exercici queda guardat i valorat automàticament, per tant, l'equip mèdic pot fer-ne les valoracions posteriorment i, fins i tot, reproduir l'exercici tantes vegades com siguin necessàries. Això permet que es pugui realitzar l'exercici a múltiples persones sense la presència d'un professional, que tot seguit els podrà valorar de forma instantània.

5.2.2 Reducció d'infraestructures

Molts dels afectats per la demència han d'acudir habitualment a un centre mèdic per a la realització dels exercicis, això comporta que molts dels centres no tinguin suficients recursos i espais per cobrir les necessitats de tothom.

Si moltes d'aquests activitats es realitzen des d'un ordinador personal, no

només permetrien al pacient estalviar visites continuades, sinó que els centres de recuperació quedarien menys carregats i per tant, podrien atendre a més gent.

5.2.3 Valoració a temps real

El fet que el propi joc virtual valori la partida realitzada permet al professional obtenir un resultat precís i instantani, i a més, facilita al mateix pacient controlar el nombre d'errors i comparar-los amb altres partides. D'aquesta manera, per ell no és tan sols un exercici més, sinó que també és un repte a superar i així s'aconsegueix una motivació extra.

5.2.4 Personalització dels exercicis

Tal i com s'ha explicat, l'equip professional serà l'encarregat de dissenyar els exercicis de zero, aquest fet permetrà disposar d'una gran varietat de circuits de manera que els usuaris no veuran el joc com un exercici monòton. A més, es podrà adequar l'activitat a cada pacient, podent graduar així la dificultat.

5.3 Motiu de l'aplicació

El projecte pretén aportar les millores descrites en aquest apartat als equips que s'encarreguen de la recuperació del pacients afectats per la demència, i facilitar així les seves tasques, a banda de millorar la qualitat de vida dels afectats.

6 Requisits del sistema

S'entén l'aplicació del projecte com a una eina d'ajuda pels equips professionals, els quals l'utilitzaran pel seguiment i recuperació dels pacients amb una afectació de la psicomotricitat fina.

6.1 Rols dels usuaris

El sistema considerarà dos tipus d'usuaris diferents: l'equip mèdic i el pacient. Cada un d'ells disposarà d'unes eines i funcionalitats determinades en funció de les seves necessitats.

6.1.1 Equip mèdic

Les necessitats d'aquest usuari seran molt semblants a les necessitats que pot tenir qualsevol dissenyador de models 3D però molt més simplificades. Aquest haurà de tenir una interfície la qual li permeti dissenyar un objecte dins l'escena, de mides i forma que ell desitgi, en aquest cas un objecte cilíndric.

Inicialment haurà de definir la forma o trajectòria del circuit, de manera que a l'usuari se li presentarà un sistema capaç de definir una ruta base per on s'anirà construint el circuit punt a punt.

Durant aquest proces d'edició l'usuari ha de poder moure's lliurement per l'escena, per tal de veure el circuit des de tots els angles i modificar-lo segons li convingui.

A part d'això, també podrà carregar qualsevol circuit prèviament guardat a memòria per tal de poder modificar-lo. Per tant, també haurà de tenir les eines disponibles per poder guardar aquests circuits un cop creats i assignar-los uns camps de títol, descripció i dificultat.

Altrament, aquest usuari també ha de poder veure els resultats del pacient. Aquest segon objectiu s'ha de veure assolit per una segona interfície que li permeti reproduir les accions d'aquest segon, per tal de treure'n unes conclusions.

6.1.2 Pacient

L'usuari conductor tindrà uns requeriments menys complexes respecte a l'usuari constructor, ja que la seva missió és guiar l'objecte per dins del circuit.

La llibertat de moviments ha de ser total, així es podrà desplaçar mitjançant el teclat en les 3 direccions de l'espai X,Y i Z, i a més podrà orientar la direcció de l'objecte per mitja del ratolí.

6.2 Navegació

La navegació dins de l'escena consta de 2 fases clarament separades: l'edició del circuit i la conducció de l'objecte dins del circuit generat anteriorment.

A la primera fase, l'edició, consta de 2 estats per a la creació del circuit:

Constructor, on l'usuari definirà els punts de pas de la trajectòria i els radis dels diferents anells que compondran el circuit, o bé decidir si vol seleccionar un circuit guardat.

Opcions d'edició, on podrà decidir si vol guardar el circuit guardat, reeditar-lo o conduir-lo.

La segona fase estarà orientada a l'usuari conductor de l'objecte, on podrà seleccionar un circuit guardat prèviament, carregar-lo i començar a dirigir un objecte per l'interior del circuit.

Durant el curs de l'activitat es mostrarà a l'usuari un seguit d'informacions i senyals que l'ajudaran a desplaçar-se.

6.3 Sistema d'emmagatzematge

El sistema ha de permetre guardar els circuits de l'equip mèdic per tal que aquests puguin reeditar-los en sessions posteriors, a més, el pacient també hi ha de poder accedir per a poder-hi jugar.

Durant el procés de conducció del pacient, el sistema també guardarà els resultats perquè l'equip mèdic en pugui fer el seguiment.

7 Estudis i decisions

Per a implementar l'aplicació és necessari analitzar i escollir les eines de treball que més s'escaiguin.

7.1 Selecció de l'entorn de treball i motor gràfic.

Per a l'elecció del motor gràfic s'optarà per una eina que accepti un bon nombre de plataformes i un llenguatge de programació estàndard i amigable, però sobretot s'ha d'escollir una eina que faciliti el màxim conjunt d'eines per treballar amb models tridimensionals, de forma que faci menys carregosa la tasca de programar.

També serà important el seu entorn de treball, doncs un bon IDE ajuda molt en la implementació del codi font.

Motor de videojoc (Game Engine), fa referència a una aplicació de programari que permet el disseny, creació i representació d'un videojoc. Hi ha molts motors de jocs que estan dissenyats per treballar en consoles de videojocs i ordinadors personals. La funcionalitat principal d'aquests motors és la representació per pantalla d'una escena, simulant els objectes 2D o 3D que intervenen amb tots els seus elements, lleis físiques, textures, àudio...

Els motors de joc permeten reutilitzar elements i eines de desenvolupament simplificant i reduint temps d'implementació a l'hora de desenvolupar un joc o aplicació 3D. A més, inclouen nativament algoritmes matemàtics per transformar objectes a l'espai, simular lleis físiques, i renderitzar tots els objectes de l'escena... de forma molt eficient.

Avui en dia, aquests motors de joc estan totalment integrats en uns entorns de desenvolupament (IDE) que permeten als creadors de videojocs estalviar molta programació, ja que els objectes que s'utilitzen moltes vegades només s'han de definir i configurar les seves propietats (rigidesa, gravetat, pes, escala...), d'aquesta manera el desenvolupador només ha de programar les regles dels jocs i la interacció entre els diferents elements del joc.

Un dels grans avantatges comercials que proporcionen aquests motors de jocs

és que permeten crear aplicacions multi-plataforma, sense fer grans modificacions en el codi font. Això permet a les empreses ampliar el seu ventall d'usuaris ràpidament, sense grans pèrdues de temps.

La majoria d'aquests motors joc solen basar-se en un motor gràfic (DirectX, OpenGL, ...) que són els encarregats de renderitzar (transformar l'escena 3D a una imatge 2D per mostrar pantalla) i donar accés a la resta de dispositius d'entrada i sortida independentment del hardware usat.

Actualment, el ventall d'aplicacions i el mercat és tan ampli que tots aquests motors gràfics no només s'usen exclusivament en consoles i ordinadors personals, sinó que s'estan exportant cap a altres tecnologies anteriorment impensables com ara mòbils, navegadors web...

Els llenguatges de programació que accepten aquests motors també estan creixent, de manera que ja es pot programar en llenguatges de molt alt nivell, com pot ser Java, c# .NET... La potència actual dels hardwars a nivell d'usuari és tan alta que la ralentització d'aquest llenguatges d'alt nivell és insignificant, mentre que l'augment de productivitat que ofereixen aquests és molt significativa.

Hi ha una gran quantitat de motors de joc, cada un amb les seves característiques:

a) **Unity -->** Motor lliure per a web, Windows i Mac OS X, pagant llicència també per iPhone, Android, Nintendo Wii, Playstation 3 i Xbox 360.

b) **Adventure Game Studio -->** Un dels més usats per a videojocs en primera persona, amb llicència gratuïta.

c) **Unreal Engine -->** Motor per a PC, i amb llicència pagada també apte per Xbox i PS3.

d) **Brender -->** Motor gràfic amb llicència comercial apte per a jocs d'ordinador, simuladors i eines gràfiques.

e) **Goggi Game Engine -->**Multi-plataforma 2D per a PC, Mac, iPhone i iPad.

En aquest projecte s'utilitzarà el motor anomenat Unity 3D, ja que la varietat de plataformes que l'accepten és alta, encara que només interessa inicialment que

corri sobre PC, independentment del sistema operatiu. I, si més endavant es volgués exportar sobre tecnologia mòbil (smartphone, tablets...) els costos no serien elevats, ja que també pot córrer sobre Android i iOS.

A més de ser un motor amb un gran ventall de plataformes, també està molt ben aconseguit el seu IDE, l'entorn de treball, que consta d'una interfície gràfica molt potent que permet manipular els diferents elements de l'escena i les seves configuracions de forma molt intuïtiva i ràpida, sense haver de fer modificacions sobre el codi font directament.

Un altre punt decisiu cap a aquest motor és que accepta bastants llenguatges de programació, tals com JS, C, C# i Boo, encara que en el projecte només es treballarà amb C#, doncs és un llenguatge d'alt nivell orientat a objectes, igual que el JS, però presenta millores notables en tema de detecció d'errors sobre aquest.

7.2 Sistema d'emmagatzematge

El sistema d'emmagatzematge de l'aplicació és un punt molt important pel desenvolupament de l'aplicació, s'ha d'escollir un mètode que satisfaci els requisits i a ser possible també que pugui assolir necessitats futures. Per tant, s'ha d'escollir un mètode flexible, àgil i simple.

L'XML (extensible markup language) és un metallenguatge extensible, d'etiquetes, desenvolupat pel World Wide Web Consortium (W3C) i és un sistema estàndard per definir llenguatges per a diferents necessitats. Tot i que la seva aparició va començar per aplicacions d'Internet, s'ha convertit en un dels millors mètodes per emmagatzemar i enviar informació per a diferents plataformes, sistemes i aplicacions.

El motiu pel qual s'ha escollit aquest sistema i no, per exemple, un de base de dades tradicional és perquè l'XML permet estructurar les dades d'una forma molt entenedora i visual, a més de permetre representar l'estructura d'aquests models i classes, tal i com estan representats a nivell de codi.

El fet de ser un sistema simple, entenedor i molt estandarditzat ha fet que gairebé tots els llenguatges de programació disposin de llibreries pel tractament

d'aquest tipus de fitxers, així doncs la seva utilització a nivell de codi fa que sigui molt simple, eficaç i eficient.

Un altre avantatge d'utilitzar aquests fitxers per a l'emmagatzematge de dades és que en ampliacions futures es podran llegir i manipular aquest fitxers, per tal d'utilitzar-los per a altres aplicacions o per ampliacions de la mateixa, de forma que no s'hagin d'implementar grans sistemes ni fer adaptacions al codi, o a la mateixa estructura de l'XML.

8 Anàlisi i disseny del sistema

Per entendre el software en qüestió, primerament s'ha de fer una explicació de com s'entén el circuit i els diferents components a nivell de programació.

8.1 Sistema de circuits

Si s'agafa qualsevol conducte corbat es pot representar com a la unió de molts cilindres units entre ells per els extrems, formant una cadena que dóna forma a un circuit.

L'element cilindre s'entén com un objecte de revolució, generat a partir d'un rectangle on l'eix de revolució és un dels costats d'aquest.

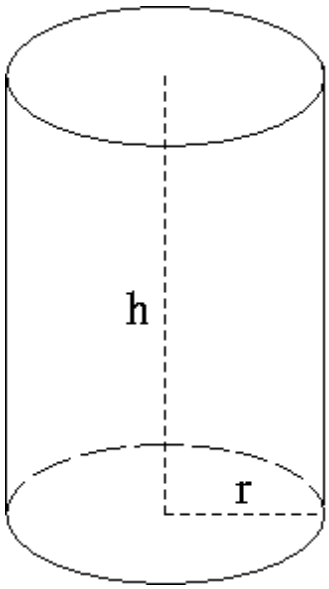


Figura 6: representació cilindre

Per tant, per obtenir un cilindre és necessari disposar d'una alçada (h) que indicarà la longitud d'aquest cilindre, i un radi que determinarà la seva amplada.

Així doncs, s'obté un pla corbat sobre una recta que en forma el cilindre, ara bé, en el món 3D tot element ha de tenir 3 dimensions, llavors aquest pla ha de tenir un gruix per ser viable. Per aconseguir això es posarà un altre cilindre amb la mateixa alçada i sobre la mateixa recta amb un radi més gran, de manera que s'obtinguin dos cilindres concèntrics, i l'espai entre els 2 cilindres serà el gruix del circuit.

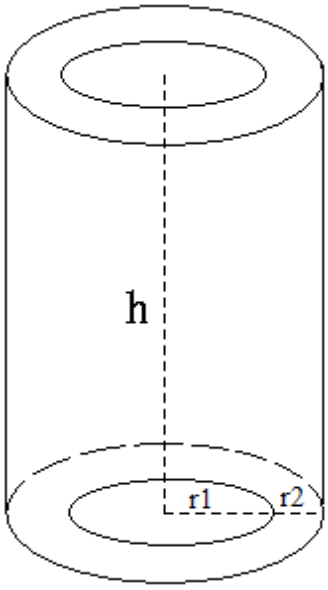


Figura 7: representació cilindre amb volum

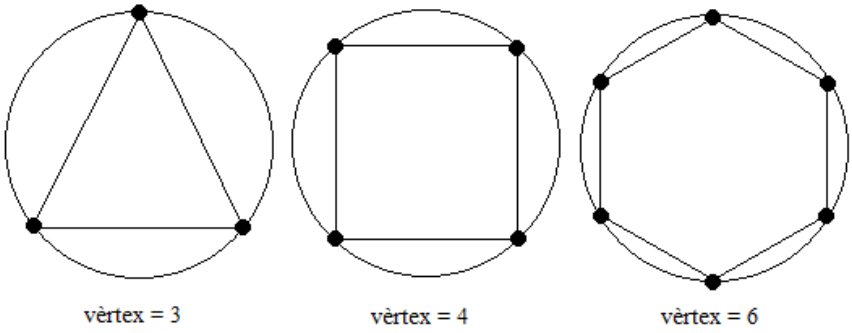
Cada cilindre està definit per un diàmetre intern (r1) i un extern (r2), que defineixen el gruix d'aquest objecte.

Amb aquest objecte base es pot definir qualsevol conducte, concatenant cilindres de diferents mides, un darrere l'altre.

S'entén un circuit com a una estructura dintre l'escena amb 2 orificis d'entrada i sortida de parets tancades, on les parets tindran diferents gruixos a cada tram d'aquest objecte.

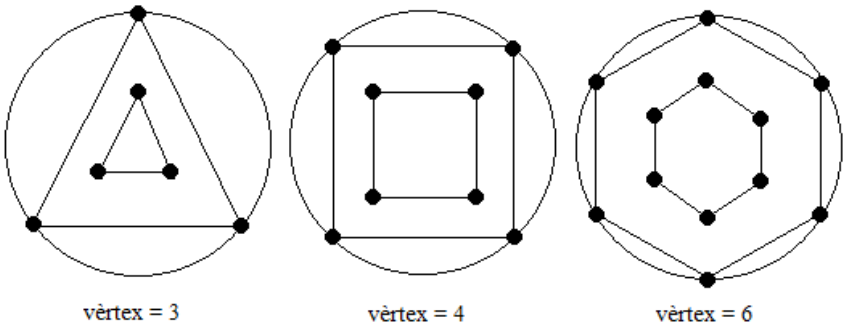
Si s'observa la secció d'un circuit tancat, sigui de la forma que sigui, es pot veure que aquest està format per dos circumferències no concèntriques, per tant, es pot definir la secció com a un **anell**, que està format per dos **cercles,** un intern i un altre extern, amb els seus radis pertinents.

Ja se sap que no es pot representar un cercle perfecte en el món virtual, per tant s'haurà de fer una aproximació usant polígons amb un cert nombre de costats definits per **vèrtexs**, per exemple:



Veient la imatge es pot deduir que, com més vèrtexs tinguin els cercles, més pròxims a una circumferència seran, i per tant, de més qualitat serà el circuit presentat.

Definits els cercles s'interpreta que les seccions del circuit (anells) tindran la següent forma:



Aquests anells seran els que donaran la forma i la mida al circuit.

Un cop definit l'element anell, es pot entendre un **circuit** com a un conjunt d'anells sobre una trajectòria units seqüencialment pels seus vèrtexs.

Així doncs, s'observen 4 objectes importants:

Cercle --> Definit per un radi, centre, direcció i un nombre de vèrtexs. Es representa com un pla a l'espai delimitat per un perímetre circular, on el centre determina la seva posició dintre l'escena, la direcció marca la seva rotació i el radi la seva mida.

Anell --> Compost per un cercle exterior i un cercle interior. Aquest objecte ha de complir que els vèrtexs del cercle intern sempre han d'estar compresos entre els del cercle extern.

Trajectòria --> Una trajectòria està composta d'un seguit de punts a l'espai que marcaran la ubicació del centre dels cercles, que defineixen el circuit.

Circuit --> És l'element final del sistema. Estarà representat per una trajectòria on es situaran cada un dels anells. Aquest element serà el que generarà el conjunt d'anells, i llavors s'uniran per formar la malla que definirà l'estructura del circuit.

En la següent imatge s'aprecia l'estructura definida anteriorment, els segments vermells formarien els cercles, i cada parella d'aquests construiria un anell. En aquest cas, hi ha un circuit de 3 seccions (anells) que s'uneixen entre ells pels seus vèrtexs (segments de color blau), per formar-ne la malla del cos 3D.

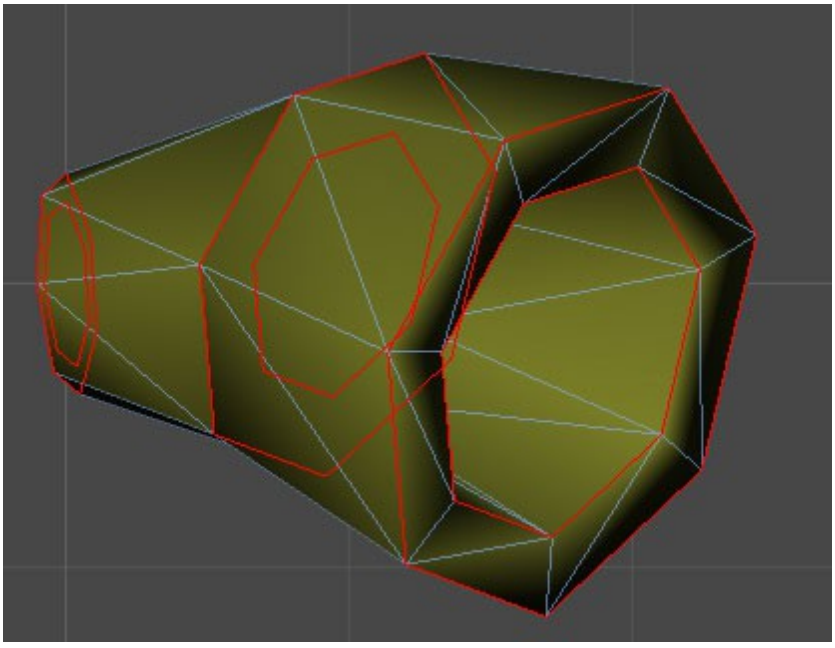


Figura 8: circuit de 3 seccions i 7 vèrtexs

Aquí es pot veure la representació 3D de l'objecte cilíndric dintre l'escena.

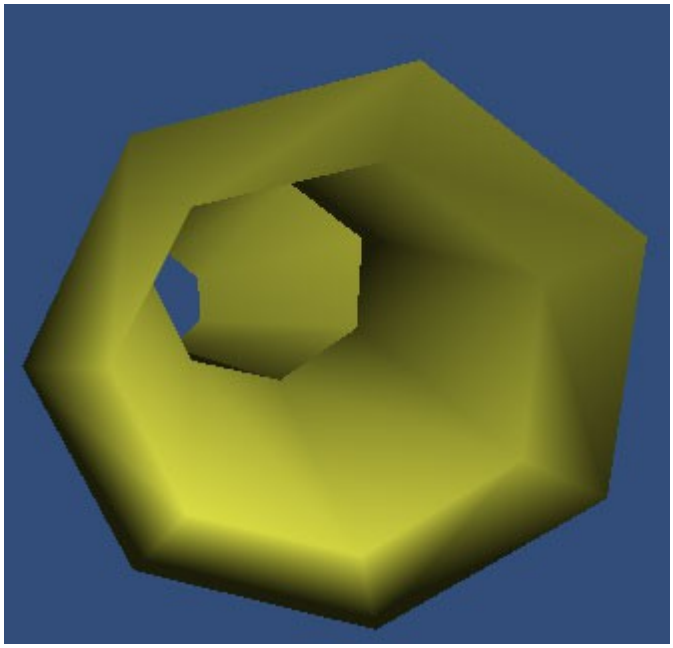


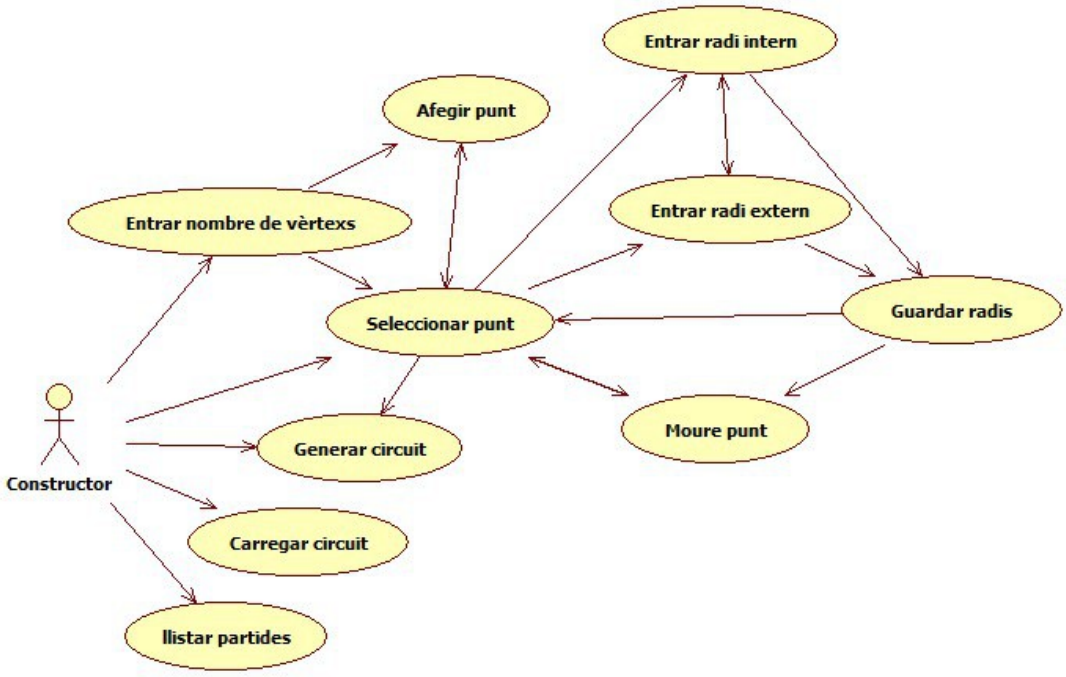
Figura 8: circuit de 3 seccions i 7 vèrtexs sobre l'escena

8.2 Fitxes de cas d'ús

Tot seguit es presenten les fitxes de cas d'ús, que serviran per entendre les eines de què disposa cada un dels usuari i la seva seqüència d'ús.

Aquestes fitxes es separen en les diferents pantalles de l'aplicació, on cadascuna té un objectiu bàsic.

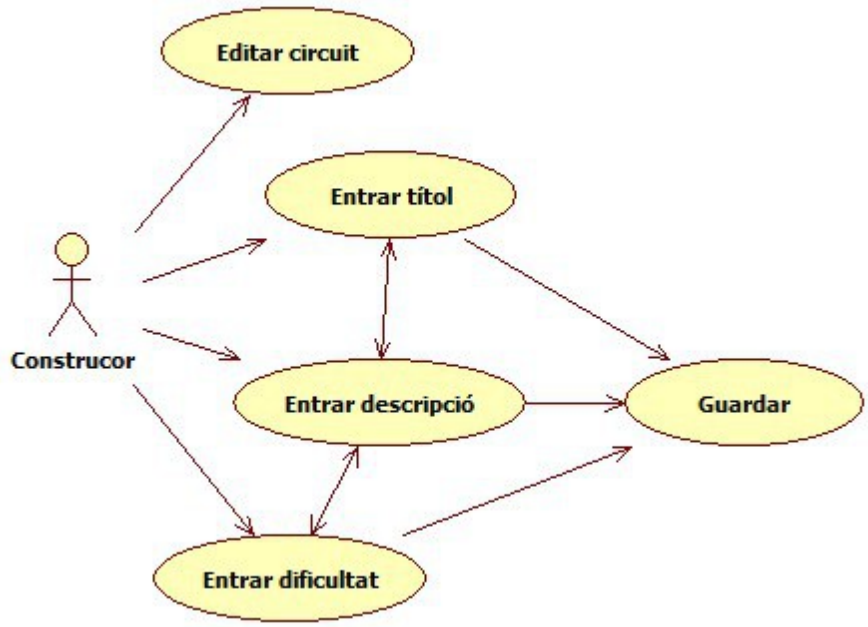
A l'iniciar l'aplicació es presenta la pantalla d'edició, on l'usuari constructor podrà crear un circuit des de zero i llavors generar-lo. També podrà dirigir-se a la pantalla de càrrega dels circuits i partides guardats.



Cas d'ús 1: Edició del circuit

Cas d'ús	Edició del circuit
Descripció	Permet crear o editar un circuit, dirigir-se a la pantalla de càrrega de circuits guardats o bé a veure les partides guardades
Precondició	L'usuari conductor executa el sistema o prové d'editar un circuit ja carregat
Flux principal	1 Si carregar circuit 1.1 Dirigir a pantalla de carregar circuit guardat 2 Si llisar partides 2.1 Dirigir a pantalla de carregar partides guardades 3 Mentre no Generar 3.1 Cas: 3.1.1 Entrar nombre de vèrtexs 3.1.2 Seleccionar punt 3.1.3 Mentre no guardar radis 3.1.3.1 Entrar radi intern 3.1.3.2 Entrar radi extern 3.1.4 Fi mentre 3.1.5 Moure punt 3.1.6 Generar circuit 3.1.6.1 Dirigir a pantalla d'opcions 4 Fi mentre
Postcondició	Deixar un circuit generat a l'escena

Un cop l'usuari ha definit bé la trajectòria del circuit i els seus punts, té l'opció de generar-lo de manera que aquesta acció crearà l'objecte 3D i el portarà a la pantalla d'opcions, on podrà guardar-lo a memòria o reeditar-lo si s'escau.



Cas d'ús 2: Opcions del circuit

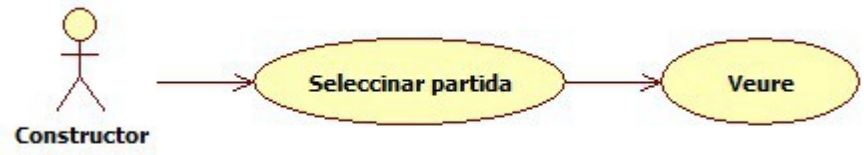
Cas d'ús	Opcions del circuit
Descripció	Permet editar un circuit a l'escena o guardar-lo
Precondició	L'usuari conductor ja ha generat un circuit a l'escena a partir de la pantalla d'edició
Flux principal	1 Si editar circuit 1.1 Enviar a la pantalla d'edició 2 Mentre no editar 2.1 Entrar títol 2.2 Entrar descripció 2.3 Entrar dificultat 2.4 Guardar 3 Fi mentre
Postcondició	Guarda el circuit carregat en escena a la memòria

Si l'usuari no desitja crear un circuit a la pantalla d'edició i en selecciona l'opció carregar, serà dirigit a la pantalla de càrrega on disposarà d'un llistat de circuits prèviament guardats, on en seleccionarà un i l'editarà, i tornarà a l'inici de l'aplicació amb la trajectòria configurada per si el vol modificar.

Cas d'ús 3: Carregar circuit guardat

Cas d'ús	Càrrega de circuit guardat
Descripció	Permet carregar la configuració d'un circuit a la pantalla d'edició
Precondició	Hi ha circuits guardats prèviament
Flux principal	1 Mentre no editar 1.1 Seleccionar circuit 1.2 Si editar 1.2.1 Enviar a pantalla d'edició 2 Fi mentre
Postcondició	Carrega el circuit guardat a la pantalla d'edició

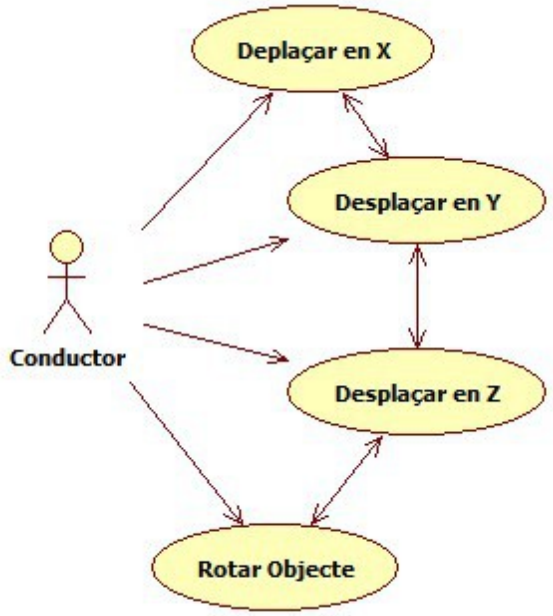
L'equip mèdic també pot optar per veure les partides guardades que han realitzat el pacients amb l'opció de llistar partides.



Cas d'ús 4: Llistar partides

Cas d'ús	Veure partides guardades
Descripció	Permet llistar i carregar les partides guardades a memòria
Precondició	Hi ha partides guardades prèviament
Flux principal	3 Mentre no veure 3.1 Seleccionar partida 3.2 Si veure 3.2.1 Enviar a pantalla reproducció de partida 4 Fi mentre
Postcondició	Carrega una partida guardada i la reprodueix

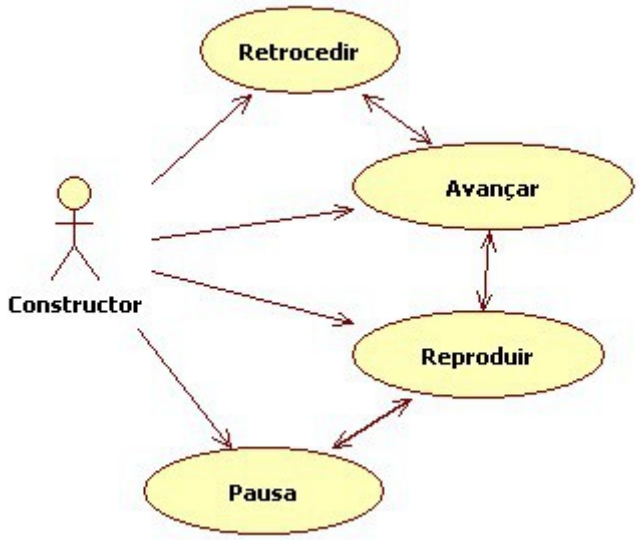
En el cas de l'usuari conductor la seva interfície serà més simple. Les seves accions es limitaran a desplaçar-se en qualsevol direcció X,Y,Z mitjançant teclat i canviar la rotació de l'objecte usant el ratolí del PC.



Cas d'ús 5: Conducció pel circuit

Cas d'ús	Conducció de l'objecte pel circuit
Descripció	Permet a l'usuari conductor guiar l'objecte per l'interior del circuit
Precondició	Circuit carregat en escena
Flux principal	1 Mentre no Final 1.1 Desplaçar en X 1.2 Desplaçar en Y 1.3 Desplaçar en Z 1.4 Rotar objecte 2 Fi mentre
Postcondició	L'usuari obté una puntuació i una partida guardada

Després de què un usuari hagi realitzat una partida el sistema la guardarà automàticament, per tant l'equip mèdic podrà realitzar una reproducció seleccionant-ne una.



Cas d'ús 6: reproducció d'una partida

Cas d'ús	Reproducció de una partida
Descripció	Permet a l'usuari constructor visualitzar una partida
Precondició	Circuit i partida guardats a memòria
Flux principal	3 Mentre no Final 3.1 Si reproduir 3.1.1 Si avançar 3.1.1.1 reproduir endavant 3.1.2 Si retrocedir 3.1.2.1 reproduir enrrera 3.2 Si pausa 3.2.1 no desplaçar 4 Fi mentre
Postcondició	L'usuari ha visualitzat la partida

Mentre l'usuari està conduit l'objecte, el sistema està escoltant les possibles col·lisions que poden sorgir. En cas que se'n detecti una, haurà de marcar l'objecte indicant que s'està col·lisionant i incrementar el nombre de col·lisions, per llavors guardar-la. Un cop deixin de col·lisionar els dos cossos la indicació haurà de desaparèixer.

Cas d'ús 7: detecció de col·lisions

Cas d'ús	Detecció de col·lisions
Descripció	El sistema esta escoltant per tal de indicar possibles condicions i comptabilitzar-les
Precondició	L'usuari conductor ha iniciat una partida
Flux principal	5 Mentre no Final 5.1 Si col·lisió 5.1.1 Marcar col·lisió 5.1.2 Afegir col·lisió 5.2 Si fi de col·lisió 5.2.1 Desmarcar col·lisió 6 Fi mentre
Postcondició	La partida està finalitzada i hi ha guardades totes les col·lisions

9 Implementació

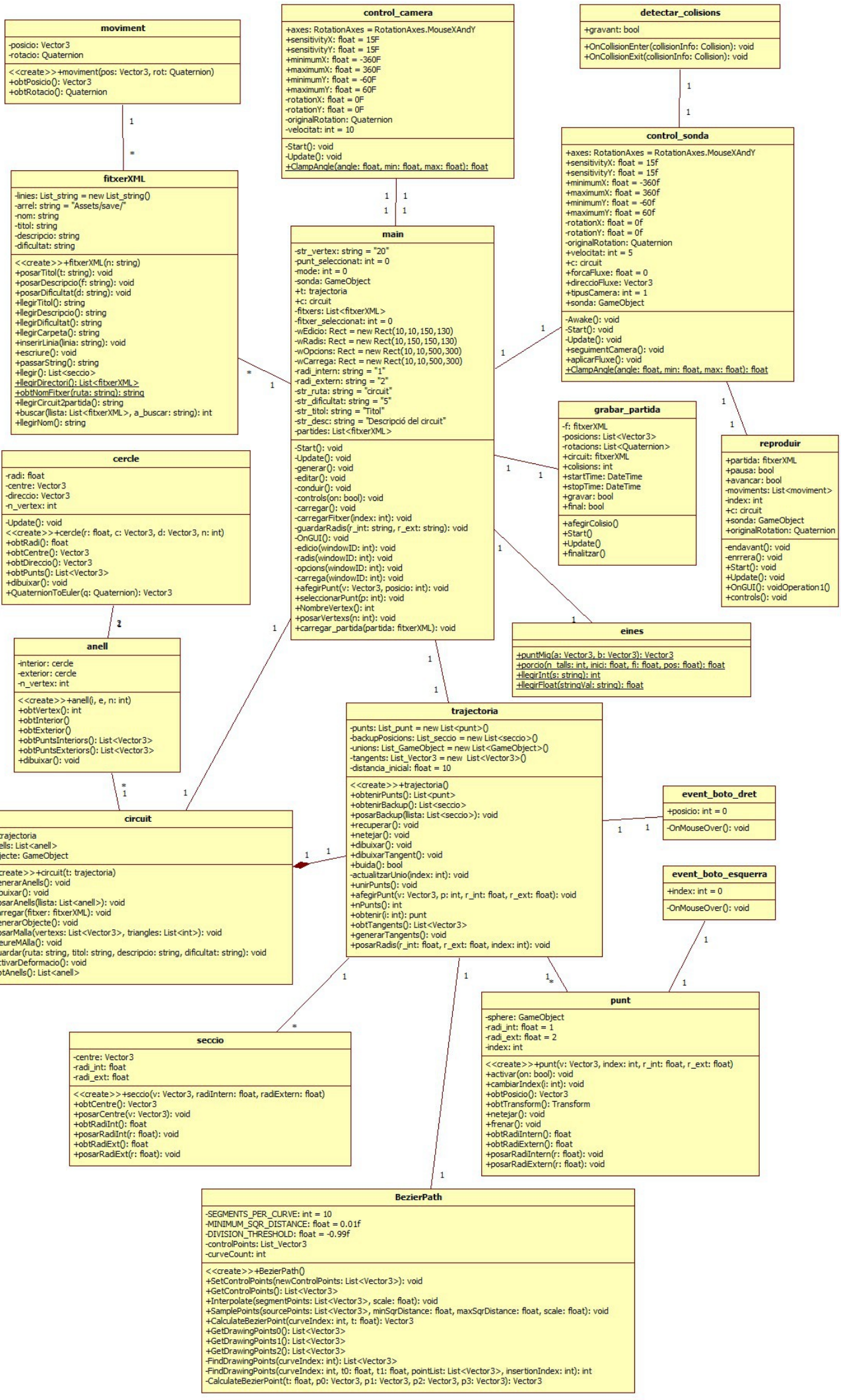
En aquest apartat es descriurà tècnicament tot el codi font de la aplicació

9.1 Diagrama de classes

El sistema està representat per un conjunt de classes que generen i controlen els objectes i faciliten les eines necessàries per treballar-hi.

Les classes més rellevants són les descrites a l'apartat de sistema de circuits, que són les encarregades de definir el circuit amb la seva estructura.

La resta de classes solen ser eines d'edició i control per facilitar les tasques als usuaris.


moviment
-posicio: Vector3
-rotacio: Quaternion
<<create>>+moviment(pos: Vector3, rot: Quaternion)
+obtPosicio(): Vector3
+obtRotacio(): Quaternion
control_camera
+axes: RotationAxes = RotationAxes.MouseXAndY
+sensitivityX: float = 15F
+sensitivityY: float = 15F
+minimumX: float = -360F
+maximumX: float = 360F
+minimumY: float = -60F
+maximumY: float = 60F
-rotationX: float = 0F
-rotationY: float = 0F
-originalRotation: Quaternion
-velocitat: int = 10
-Start(): void
-Update(): void
+ClampAngle(angle: float, min: float, max: float): float
detectar_colisions
+gravant: bool
+OnCollisionEnter(collisionInfo: Collision): void
+OnCollisionExit(collisionInfo: Collision): void
control_sonda
+axes: RotationAxes = RotationAxes.MouseXAndY
+sensitivityX: float = 15f
+sensitivityY: float = 15f
+minimumX: float = -360f
+maximumX: float = 360f
+minimumY: float = -60f
+maximumY: float = 60f
-rotationX: float = 0f
-rotationY: float = 0f
-originalRotation: Quaternion
+velocitat: int = 5
+c: circuit
+forcaFluxe: float = 0
+direccioFluxe: Vector3
+tipusCamera: int = 1
+sonda: GameObject
-Awake(): void
-Start(): void
-Update(): void
+seguimentCamera(): void
+aplicarFluxe(): void
+ClampAngle(angle: float, min: float, max: float): float
fitxerXML
-linies: List_string = new List_string()
-arrel: string = "Assets/save/"
-nom: string
-titol: string
-descripcio: string
-dificultat: string
<<create>>+fitxerXML(n: string)
+posarTitol(t: string): void
+posarDescripcio(f: string): void
+posarDificultat(d: string): void
+llegirTitol(): string
+llegirDescripcio(): string
+llegirDificultat(): string
+llegirCarpeta(): string
+inserirLinia(linia: string): void
+escriure(): void
+passarString(): string
+llegir(): List<seccio>
+llegirDirectori(): List<fitxerXML>
+obtNomFitxer(ruta: string): string
+llegirCircuit2partida(): string
+buscar(llista: List<fitxerXML>, a_buscar: string): int
+llegirNom(): string
main
-str_vertex: string = "20"
-punt_seleccionat: int = 0
-mode: int = 0
-sonda: GameObject
+t: trajectoria
+c: circuit
-fitxers: List<fitxerXML>
-fitxer_seleccionat: int = 0
-wEdicio: Rect = new Rect(10,10,150,130)
-wRadis: Rect = new Rect(10,150,150,130)
-wOpcions: Rect = new Rect(10,10,500,300)
-wCarrega: Rect = new Rect(10,10,500,300)
-radi_intern: string = "1"
-radi_extern: string = "2"
-str_ruta: string = "circuit"
-str_dificultat: string = "5"
-str_titol: string = "Titol"
-str_desc: string = "Descripció del circuit"
-partides: List<fitxerXML>
-Start(): void
-Update(): void
-generar(): void
-editar(): void
-conduir(): void
-controls(on: bool): void
-carregar(): void
-carregarFitxer(index: int): void
-guardarRadis(r_int: string, r_ext: string): void
-OnGUI(): void
-edicio(windowID: int): void
-radis(windowID: int): void
-opcions(windowID: int): void
-carrega(windowID: int): void
+afegirPunt(v: Vector3, posicio: int): void
+seleccionarPunt(p: int): void
+NombreVertex(): int
+posarVertexs(n: int): void
+carregar_partida(partida: fitxerXML): void
grabar_partida
-f: fitxerXML
-posicions: List<Vector3>
-rotacions: List<Quaternion>
+circuit: fitxerXML
+colisions: int
+startTime: DateTime
+stopTime: DateTime
+gravar: bool
+final: bool
+afegirColisio()
+Start()
+Update()
+finalitzar()
reproduir
+partida: fitxerXML
+pausa: bool
+avancar: bool
-moviments: List<moviment>
-index: int
+c: circuit
+sonda: GameObject
+originalRotation: Quaternion
-endavant(): void
-enrrera(): void
+Start(): void
+Update(): void
+OnGUI(): voidOperation1()
+controls(): void
cercle
-radi: float
-centre: Vector3
-direccio: Vector3
-n_vertex: int
-Update(): void
<<create>>+cercle(r: float, c: Vector3, d: Vector3, n: int)
+obtRadi(): float
+obtCentre(): Vector3
+obtDireccio(): Vector3
+obtPunts(): List<Vector3>
+dibuixar(): void
+QuaternionToEuler(q: Quaternion): Vector3
eines
+puntMig(a: Vector3, b: Vector3): Vector3
+porcio(n_talls: int, inici: float, fi: float, pos: float): float
+llegirInt(s: string): int
+llegirFloat(stringVal: string): float
anell
-interior: cercle
-exterior: cercle
-n_vertex: int
<<create>>+anell(i, e, n: int)
+obtVertex(): int
+obtInterior()
+obtExterior()
+obtPuntsInteriors(): List<Vector3>
+obtPuntsExteriors(): List<Vector3>
+dibuixar(): void
trajectoria
-punts: List_punt = new List<punt>()
-backupPosicions: List_seccio = new List<seccio>()
-unions: List_GameObject = new List<GameObject>()
-tangents: List_Vector3 = new List<Vector3>()
-distancia_inicial: float = 10
<<create>>+trajectoria()
+obtenirPunts(): List<punt>
+obtenirBackup(): List<seccio>
+posarBackup(llista: List<seccio>): void
+recuperar(): void
+netejar(): void
+dibuixar(): void
+dibuixarTangent(): void
+buida(): bool
-actualitzarUnio(index: int): void
+unirPunts(): void
+afegirPunt(v: Vector3, p: int, r_int: float, r_ext: float): void
+nPunts(): int
+obtenir(i: int): punt
+obtTangents(): List<Vector3>
+generarTangents(): void
+posarRadis(r_int: float, r_ext: float, index: int): void
event_boto_dret
+posicio: int = 0
-OnMouseOver(): void
event_boto_esquerra
+index: int = 0
-OnMouseOver(): void
circuit
-t: trajectoria
-anells: List<anell>
-objecte: GameObject
<<create>>+circuit(t: trajectoria)
+generarAnells(): void
+dibuixar(): void
+posarAnells(llista: List<anell>): void
+carregar(fitxer: fitxerXML): void
+generarObjecte(): void
+posarMalla(vertexs: List<Vector3>, triangles: List<int>): void
+treureMalla(): void
+guardar(ruta: string, titol: string, descripcio: string, dificultat: string): void
+activarDeformacio(): void
+obtAnells(): List<anell>
seccio
-centre: Vector3
-radi_int: float
-radi_ext: float
<<create>>+seccio(v: Vector3, radiIntern: float, radiExtern: float)
+obtCentre(): Vector3
+posarCentre(v: Vector3): void
+obtRadiInt(): float
+posarRadiInt(r: float): void
+obtRadiExt(): float
+posarRadiExt(r: float): void
punt
-sphere: GameObject
-radi_int: float = 1
-radi_ext: float = 2
-index: int
<<create>>+punt(v: Vector3, index: int, r_int: float, r_ext: float)
+activar(on: bool): void
+cambiarIndex(i: int): void
+obtPosicio(): Vector3
+obtTransform(): Transform
+netejar(): void
+frenar(): void
+obtRadiIntern(): float
+obtRadiExtern(): float
+posarRadiIntern(r: float): void
+posarRadiExtern(r: float): void
BezierPath
-SEGMENTS_PER_CURVE: int = 10
-MINIMUM_SQR_DISTANCE: float = 0.01f
-DIVISION_THRESHOLD: float = -0.99f
-controlPoints: List_Vector3
-curveCount: int
<<create>>+BezierPath()
+SetControlPoints(newControlPoints: List<Vector3>): void
+GetControlPoints(): List<Vector3>
+Interpolate(segmentPoints: List<Vector3>, scale: float): void
+SamplePoints(sourcePoints: List<Vector3>, minSqrDistance: float, maxSqrDistance: float, scale: float): void
+CalculateBezierPoint(curveIndex: int, t: float): Vector3
+GetDrawingPoints0(): List<Vector3>
+GetDrawingPoints1(): List<Vector3>
+GetDrawingPoints2(): List<Vector3>
-FindDrawingPoints(curveIndex: int): List<Vector3>
-FindDrawingPoints(curveIndex: int, t0: float, t1: float, pointList: List<Vector3>, insertionIndex: int): int
-CalculateBezierPoint(t: float, p0: Vector3, p1: Vector3, p2: Vector3, p3: Vector3): Vector3

En els següents apartats es descriuen les classes i tots els seus mètodes i funcions.

9.1.1 Classe MonoBehaviour

En Unity totes les classes que hereten de MonoBehaviour representen components que defineixen el comportament del objectes als quals estan assignats. Aquesta classe pare ja té alguns mètodes definits tals com start, update, onGui ... que s'executen a partir de certs events:

void Start() --> Aquest mètode s'executa en el moment de la creació de l'objecte dins l'escena, és molt usat per iniciar variables i assignar atributs.

void Update() --> Tots els objectes amb components del tipus MonoBehaviour criden aquest mètode a cada *frame*, de manera que es poden controlar els estats del sistema i modificar les propietats dels objectes constantment.

void OnGui() --> En aquest segment de codi és on es defineixen tots els elements d'interfície gràfica, així com finestres, botons, caixes de text ...

void OnMouseOver() --> Quan el ratolí es situa sobre l'objecte amb la classe actual assignada, aquest crea un event que executa aquest mètode.

void OnCollisionEnter(Collision collisionInfo) --> En l'instant en què dos objectes de l'escena es toquen s'executa el mètode, facilitant per paràmetres la informació de la col·lisió(posicions, direccions, objectes afectats...)

void OnCollisionExit(Collision collisionInfo) --> al contrari que l'anterior, aquest event s'inicia en el moment que els dos cossos deixen d'estar en contacte.

9.1.2 Classe Main

Aquesta classe és un component de la càmera principal de l'aplicació, és l'encarregada de generar tota la interfície gràfica, els objectes de treball i assignar-los els components necessaris.

string str_vertex --> Nombre de vèrtex actual que es carregarà al circuit quan es generi.

int punt_seleccionat --> Punt de la trajectòria seleccionat actualment.

float radi_int --> Valor per defecte del radi intern dels punts de la trajectòria.

float radi_ext -->Valor per defecte del radi extern dels punts de la trajectòria.

private int mode --> Estat actual de l'escena, s'utilitza per generar la pantalla corresponent, en cada moment els seus valor són:

0- Edició del circuit.

1- Circuit carregar.

2- Conducció del circuit.

3- Llistat de circuits.

4- Llistat de partides**.**

GameObject objecte --> Objecte 3D que recorre el circuit controlat per teclat i ratolí.

public trajectoria t --> Objecte que guarda totes les característiques de la trajectòria del circuit.

public circuit c --> Objecte circuit on es guarden els vèrtex i arestes que generen el model 3D.

private List<fitxerXML> fitxers --> Llista de circuits guardats a memòria.

private List<fitxerXML> partides --> Llista de partides guardades a memòria.

private int fitxer_seleccionat --> Índex del fitxer circuit seleccionat actualment.

public bool hi_ha_partida_seleccionada --> Indica si hi ha una partida seleccionada i guardada.

void Start() -->Crea la trajectòria inicial on es formarà el circuit.

void Update() --> Segons l'estat del sistema i les accions de l'usuari, aquest mètode s'encarrega de situar els punts de pas de la trajectòria i actualitzar-ne constantment les dades, obtenint-ne un circuit pregenerat sense l'objecte en 3D.

private void generar() --> Elimina tots els objectes d'edició de l'escena i genera l'objecte 3D del circuit actual.

private void editar() --> Elimina l'objecte 3D de l'escena i torna a posar totes les eines d'edició de circuit.

private void conduir() --> Prepara interfície per conduir, crea un objecte que recorrerà el circuit dirigit per l'usuari constructor i li assigna els components control_sonda, circuit, gravar_partida i fitxerXML.

private void controls(bool on) --> Habilita el component control_camera si la variable *on* és igual a cert, altrament el deshabilita.

private void carregar() --> Obté tots els circuits guardats en el directori de fitxers XML i els emmagatzema a l'atribut fitxers.

private void carregarFitxer(int index) --> Genera la trajectòria i circuit guardats en el fitxer XML de la taula fitxers amb posició index, i deixa l'escena preparada per editar-lo.

private void guardarRadis(string r_int,string r_ext) --> Assigna els radis r_int i r_ext al punt seleccionat en l'escena per l'atribut punt_seleccionat.

private void llistarPartides() --> Guarda a l'atribut partides totes les partides guardades en el directori de fitxers XML

private int buscar(List<fitxerXML> llista, string a_buscar) --> Busca a la taula de partides llista una partida amb nom igual a l'string a_buscar.

private void carregar_partida(fitxerXML partida) --> Prepara l'escena per tal de poder reproduir una partida guardada, busca el circuit associat a la partida, el genera, assigna el component gravar_partida a l'objecte i n'habilita els controls.

void OnGUI() --> Genera les finestres, botons i caixes de text en funció de l'estat de l'escena.

void edicio(int windowID) --> Genera les finestres inicials, que permeten entrar un nombre de vèrtexs, generar un circuit definit i carregar circuits guardats o partides.

void radis(int windowID) --> Crea una finestra per entrar el valor dels radis del punt seleccionat.

void opcions(int windowID) --> Defineix la finestra que, un cop generat el circuit permet conduir-hi, guardar-lo o reeditar-lo.

void carrega(int windowID) -->Llista els circuits guardats a memòria per tal de

seleccionar-ne un i carregar-ne la configuració.

void llista (int windowID) --> Llista les partides per tal de seleccionar-ne una i començar el procés de reproducció.

public int NombreVertex() --> Llegeix el nombre de vèrtexs actual.

public void posarVertexs(int n) --> Canvia el valor del nombre de vèrtexs actual assignant-li *i*.

9.1.3 Classe trajectòria

Aquesta classe representa un conjunt de punts units seqüencialment representant una línia a l'espai.

private List<punt> punts --> Llistat de punts a l'espai per poder moure dintre l'escena.

private List<seccio> backupPosicions --> Llistat de les seccions usat per guardar els punts i les seves propietats quan no s'està editant el circuit.

private List<GameObject> unions --> Objectes 3D que contenen els punts.

private List<Vector3> tangents --> Llistat de punts a l'espai que representen la trajectòria real del circuit.

private float distancia_inicial --> Distància inicial en l'eix Z entre punts.

public trajectoria() --> Mètode constructor que crea una trajectòria de dos punts amb els paràmetres inicials.

public List<punt> obtenirPunts() --> Retorna tots els punts de la trajectòria actual.

public List<seccio> obtenirBackup() --> Retorna les seccions guardades abans de l'última càrrega de la trajectòria.

public void posarBackup(List<seccio> llista) --> Guarda la llista a la taula de backups.

public void recuperar() --> Torna a deixar la trajectòria tal i com estava en l'últim backup.

public void netejar() --> Elimina tots els punts i el seus objectes 3D per netejar l'escena un cop finalitzada l'edició.

public void dibuixar() --> Dibuixa a l'escena les unions entre punts. Aquest mètode només és útil en el procés de desenvolupament.

public void dibuixarTangent() --> Dibuixa a l'escena les unions tangents entre punts. Aquest mètode només és útil en el procés de desenvolupament.

public bool buida() --> Retorna cert si a la trajectòria no hi ha punts, altrament fals.

private void actualitzarUnio(int index) --> Aquest mètode regenera la unió index en funció de les propietats modificades per l'usuari, per tal que la trajectòria sigui editable a temps real.

public void unirPunts() --> Actualitza totes les unions i genera les tangents de la trajectòria.

public void afegirPunt (Vector3 v, int p, float r_int, float r_ext) --> Afegeix un nou punt amb centre v, radi intern r_int, radi extern r_ext, a la posició p de la trajectòria.

public int nPunts() --> Retorna el nombre de punts de la trajectòria.

public punt obtenir(int i) --> Retorna l'i-èssim punt de la trajectòria.

public List<Vector3> obtTangents() --> Retorna totes les tangents de la trajectòria.

public void generarTangents() --> Genera totes les tangents entre punts a partir del teorema de les corbes de Beizer.

public void posarRadis(float r_int, float r_ext, int index) --> Posa el radi intern r_int i el radi extern r_ext en el punt index de la llista de punts.

9.1.4 Classe circuit

Conté totes les propietats d'un circuit per tal de generar un objecte 3D.

private trajectoria t --> Objecte trajectòria que marcarà el pas del circuit.

private List<anell> anells --> Llistat d'anells que representen les seccions del circuit.

private GameObject objecte --> Objecte 3D resultant de la generació del circuit.

public circuit(trajectoria t) -->Mètode constructor, que genera un objecte circuit a partir d'una trajectòria t.

public void generarAnells() --> Genera els anells del circuit a partir de les seccions de la trajectòria.

public void dibuixar() --> Dibuixa a l'escena les unions entre punts. Aquest mètode només és útil en el procés de desenvolupament.

public void posarAnells(List<anell> llista) --> Afegeix el llistat d'anells a l'atribut anells del circuit.

public void carregar(fitxerXML fitxer) --> Genera un circuit a partir de fitxer.

public void generarObjecte() --> Calcula tots els vèrtexs i arestes que composen el model 3D del circuit i en generen l'objecte 3D.

private void posarMalla(List<Vector3> vertexs, List<int> triangles) --> A partir dels llistats de vèrtexs i triangles genera un objecte 3D i n'assigna les propietats.

public void treureMalla() --> Elimina l'objecte 3D del circuit per tal que no es mostri en pantalla.

public void guardar(string ruta,string titol,string descripcio,string dificultat) --> Genera un fitxerXML amb la ruta títol i descripció passada per paràmetres i la configuració del circuit i el guarda a memòria.

public void activarDeformacio() --> Assigna la propietat deform a l'objecte 3D per tal que es pugui deformar.

public List<anell> obtAnells() --> Retorna els anells del circuit.

9.1.5 Classe anell

Aquesta classe guarda parelles de cercles i representen les seccions de tall del circuit.

private cercle interior --> Cercle intern de l'anell.

private cercle exterior --> Cercle extern de l'anell.

private int n_vertex --> Nombre de vèrtexs dels cercles.

public anell(cercle i, cercle e, int n) -->Mètode constructor que genera un anell a partir d'un cercle intern i, un extern e, i un nombre de vèrtexs.

public int obtVertex() --> Retorna el nombre de vèrtexs de l'anell.

public cercle obtInterior() --> Retorna el cercle interior.

public cercle obtExterior() --> Retorna el cercle exterior.

public List<Vector3> obtPuntsInteriors() --> Retorna els punts a l'espai del cercle interior.

public List<Vector3> obtPuntsExteriors() --> Retorna els punts a l'espai del cercle exterior.

public void dibuixar() --> Dibuixa a l'escena els cercles. Aquest mètode només es útil en el procés de desenvolupament.

9.1.6 Classe cercle

Aquesta classe representa un cercle dintre l'escena.

private float radi --> Radi del cercle.

private Vector3 centre -->Posició del cercle dintre l'escena.

private Vector3 direccio --> Vector perpendicular al pla format pel cercle.

private int n_vertex --> Nombre de vèrtexs del polígon.

void Update() --> A cada frame dibuixa el cercle a l'escena.

public cercle(float r, Vector3 c, Vector3 d, int n) --> Mètode constructor, crea un cercle a partir d'un radi r, un punt central c, una direcció d i un nombre de vèrtexs n.

public float obtRadi() --> Retorna el radi del cercle.

public Vector3 obtCentre() --> Retorna el centre del cercle.

public Vector3 obtDireccio() --> Retorna la direcció del cercle.

public List<Vector3> obtPunts() --> Retorna el llistat de punts que formen el perímetre del cercle.

public void dibuixar() --> Dibuixa el cercle a l'escena. Aquest mètode només és útil en el procés de desenvolupament.

9.1.7 Classe fitxerXML

Aquesta classe és l'encarregada de gestionar els fitxers en format XML.

private List<string> linies --> Informació línia a línia que compon o compondrà el fitxer en forma d'strings.

private string arrel --> Directori arrel on es guarden els circuits i partides en format XML.

private string nom --> Nom del fitxer dintre el directori arrel.

private string titol --> Títol del circuit definit per l'usuari.

private string descripcio --> Descripció del circuit definida per l'usuari.

private string dificultat --> Dificultat del circuit de 0 a 10 (10 el més difícil).

public fitxerXML(string n) --> Mètode constructor, donat un nom n crea un objecte fitxer.

public void posarTitol(string t) --> Assigna el valor t a l'atribut titol.

public void posarDescripcio(string f) --> Assigna el valor de l'string f a l'atribut descripcio.

public void posarDificultat(string d) --> Assigna el valor de l'string d a l'atribut dificultat.

public string llegirNom() --> Retorna el valor del títol.

public string llegirDescripcio() --> Retorna el valor de la descripció.

public string llegirDificultat() --> Retorna el valor de la dificultat.

public string llegirCarpeta() --> Retorna la ruta de la carpeta de treball.

public void inserirLinia(string linia) --> Afegeix un string al final de la llista de strings que formen el contingut del fitxer.

public void escriure() --> Crea i guarda un fitxer .xml amb la informació definida a l'objecte. En cas que el fitxer existeixi en crea un de nou, anomenant-los de la forma: circuit.xml, circuit(1).xml, circuit(2).xml ,..., circuit(n).xml

public string passarString() --> Llegeix un fitxer xml i retorna el contingut en forma d'string.

public List<seccio> llegir() --> Llegeix un fitxer circuit, i retorna la llista de seccions que el formen.

public List<moviment> llegirPartida() --> Llegeix un fitxer partida i retorna la llista de seccions que el formen.

public static List<fitxerXML> llegirDirectori(int tipus) --> Llegeix tots els fitxers del directori de treball i els retorna en una llista, segons el paràmetre tipus busca els circuits si tipus val 0, o bé les partides si tipus val 1.

public string llegirCircuit2partida() --> Retorna el fitxerXML que guarda el circuit de la partida actual.

public static string obtNomFitxer(string ruta) --> Retorna el nom del fitxer guardat a memòria sense extensió ni ruta de carpetes.

9.1.8 Classe control_camera

Aquesta classe s'ocupa del control de la càmera en el procés d'edició, és un component assignat a la càmera per tal de controlar-la mitjançant teclat i ratolí.

public enum RotationAxes --> Rotació dels eixos en funció del ratolí.

public RotationAxes axes --> Diferència de rotació del ratolí entre els eixos X i Y.

public float sensitivityX --> Sensibilitat del ratolí en l'eix X.

public float sensitivityY --> Sensibilitat del ratolí en l'eix Y.

public float minimumX --> Màxim d'angles de rotació en X.

public float maximumX --> Màxim d'angles de rotació en Y.

public float minimumY --> Mínim d'angles de rotació en X.

public float maximumY --> Mínim d'angles de rotació en X.

float rotationX --> Rotació inicial en X.

float rotationY --> Rotació inicial en Y.

Quaternion originalRotation -->Rotació inicial de la càmera.

int velocitat -> Velocitat de desplaçament.

void Start() --> Assigna component de llum a la càmera.

Update() --> Modifica la rotació i posició de la càmera a cada frame segons els events de teclat i ratolí.

public static float ClampAngle (float angle, float min, float max) --> Donat un valor angle inicial, en resta les voltes sobrants a 360º i en retorna el valor més pròxim entre min i max.

9.1.9 Classe control_sonda

Aquesta classe és molt semblant a la classe control_camera, aquesta intervé en el moment de conducció de l'usuari pacient, dirigint l'objecte per l'interior del circuit. A més, també s'encarrega de gravar la partida i finalitzar-la quan s'ha arribat al final del circuit.

public enum RotationAxes --> Rotació dels eixos en funció del ratolí.

public RotationAxes axes --> Diferència de rotació del ratolí entre els eixos X i Y.

public float sensitivityX --> Sensibilitat del ratolí en l'eix X.

public float sensitivityY -->Sensibilitat del ratolí en l'eix Y.

public float minimumX --> Màxim d'angles de rotació en X.

public float maximumX --> Màxim d'angles de rotació en Y.

public float minimumY --> Mínim d'angles de rotació en X.

public float maximumY --> Mínim d'angles de rotació en X.

float rotationX --> Rotació inicial en X.

float rotationY --> Rotació inicial en Y.

Quaternion originalRotation --> Rotació inicial de la càmera.

int velocitat ---> Velocitat de desplaçament.

public circuit -> Objecte circuit.

public float forcaFluxe --> Mòdul del vector flux.

public Vector3 direccioFluxe --> Vector director del flux.

public int tipusCamera --> Tipus de càmera.

public GameObject sonda --> Objecte 3D que circula per dins del circuit.

public bool grabar --> Control de gravació, si cert es va guardant la partida a cada frame.

void Start() --> Assigna component de llum a la càmera.

Update() --> Modifica la rotació i posició de la càmera a cada frame segons els events de teclat i ratolí.

public void seguimentCamera() --> Canvia la rotació i posició de la càmera de manera que sempre segueixi a l'objecte.

public void aplicarFluxe() --> Aplica una lleugera desviació a l'objecte segons el tram que s'estigui recorrent del circuit.

public static float ClampAngle (float angle, float min, float max) --> Donat un valor angle inicial, en resta les voltes sobrants a 360º i en retorna el valor més pròxim entre min i max.

9.1.10 Classe detectar_colisions

Aquest és un component de l'objecte que comptabilitza i indica a l'usuari les col·lisions que realitza amb les parets del circuit.

public bool gravant --> Variable de control, s'incrementaran les col·lisions mentre gravant sigui cert.

void OnCollisionEnter(Collision collisionInfo) --> S'executa en el moment d'entrar en contacte amb el circuit, suma una col·lisió més a la partida i canvia el color de l'objecte a vermell, indicant que s'esta tocant el circuit amb l'objecte.

void OnCollisionExit(Collision collisionInfo) --> S'executa en el moment que l'objecte i el circuit deixen de tocar-se, posant el color de l'objecte altre cop a blanc.

9.1.11 Classe eines

Aquí es troben algunes eines, mètodes estàtics útils a l'aplicació.

public static Vector3 puntMig(Vector3 a, Vector3 b) --> Donada una posició a i una b retorna el punt mig entre les dues.

public static float porcio(int n_talls,float inici,float fi,float pos) --> Donat un conjunt de n_talls valors situats equitativament entre els valors inici i fi es retorna el valor proporcional del punt amb posició pos.

public static int llegirInt(string s) --> Retorna un enter representat per la cadena de caràcters s.

public static float llegirFloat(string stringVal) --> Retorna un valor numèric representat per la cadena de caràcters stringVal.

9.1.12 Classe event_boto_dret

Aquesta classe és un component dels objectes que representen les unions entre punts de la trajectòria, la seva funció és afegir un punt a la unió que s'ha seleccionat amb el botó dret.

public int posicio --> Index de la unió dins la trajectòria.

void OnMouseOver() --> Afegeix un nou punt a la trajectòria en el centre de la unió que s'ha seleccionat amb el botó dret. Aquesta funció només s'executa quan el ratolí està sobre l'objecte 3D.

9.1.13 Classe event_boto_esquerra

Aquesta classe és un component dels objectes que representen els punts de la trajectòria, la seva funció és indicar a la classe main quin és el punt seleccionat en cada moment.

public int posicio -->Index del punt dins la trajectòria.

void OnMouseOver() --> Indica a la classe main quin és el punt que s'ha seleccionat amb el botó esquerre del ratolí. Aquesta funció només s'executa quan el ratolí està sobre l'objecte 3D.

9.1.14 Classe grabar_partida

private fitxerXML f --> Objecte que conté el fitxer XML on es guardarà la partida.

private List<Vector3> posicions --> Llistat de posicions que ha seguit l'objecte durant la partida.

private List<Quaternion> rotacions --> Llistat de rotacions que ha seguit l'objecte durant la partida.

public bool gravar --> Indica quan la partida està en procés, i aquesta serà gravada mentre el seu valor sigui cert.

public bool final --> Indica el final de la partida, quan el seu valor és cert s'atura l'execució de la classe.

public fitxerXML circuit --> Objecte circuit on es desenvolupa la partida.

public int colisions --> Nombre de col·lisions que s'han produït durant la partida.

DateTime startTime --> Marca horària de l'inici de la partida.

DateTime stopTime --> Marca horària del final de la partida.

public void afegirColisio() --> Incrementa en una unitat el nombre de col·lisions.

void Start() --> Posa gravar a cert, final a fals i guarda la marca de temps actual startTime.

void Update() --> A cada frame guarda la posició de l'objecte a la llista de posicions, la rotació a la llista de rotacions mentre gravar sigui cert, si final és igual a cert finalitza la partida guardant-la.

void finalitzar() --> Guarda la marca de temps actual a stopTime i en calcula la diferència amb strarTime, genera tot el contingut XML de la partida amb les rotacions, posicions, el títol, l'hora de joc, les col·lisions i la descripció per escriure-ho a memòria.

9.1.15 Classe moviment

Aquest objecte guarda una posició i una rotació a l'espai.

private Vector3 posicio --> Posició en l'espai.

private Quaternion rotacio --> Rotació en l'espai.

public moviment(Vector3 pos, Quaternion rot) --> Crea un moviment a partir d'una posició pos i una rotació rot.

public Vector3 obtPosicio() --> Retorna la posició actual.

public Quaternion obtRotacio() --> Retorna la rotació actual.

9.1.16 Classe punt

Aquest és un component dels objectes 3D que representen els punts de pas de la trajectòria, per tal que l'usuari els pugui moure i guardar-ne la informació.

GameObject sphere --> Objecte 3D a l'escena.

private float radi_int --> Radi intern per defecte de la secció situada sobre l'objecte.

private float radi_ext --> Radi extern per defecte de la secció situada sobre l'objecte.

private int index --> Index que ocupa l'objecte sobre la trajectòria.

public punt(Vector3 v,int index,float r_int, float r_ext) --> Objecte constructor, crea una esfera amb centre v, posició index, radi intern r_int i radi extern r_ext.

public void activar (bool on) --> Canvia el color de l'objecte indicant a l'usuari quin és el punt que està editant, si on és igual a cert posa l'esfera en vermell, altrament la torna a deixar blanca.

public void cambiarIndex(int i) --> Assigna l'index i a l'objecte.

public Vector3 obtPosicio() --> Retorna la posició actual de l'objecte 3D dins l'escena.

public Transform obtTransform() --> Retorna el component Transform de l'objecte 3D.

public void netejar() --> Elimina l'objecte de l'escena.

public void frenar() --> Posa la velocitat actual de l'objecte a zero per evitar les forces d'inèrcia que pugui desviar l'objecte 3D.

public float obtRadiIntern() --> Retorna l'atribut radi_int actual.

public float obtRadiExtern() --> Retorna l'atribut radi_ext actual.

public void posarRadiIntern(float r) --> Assigna a radi_int el valor r.

public void posarRadiExtern(float r) --> Assigna a radi_ext el valor r.

9.1.17 Classe reproduir

Aquest component de l'objecte s'encarrega de reproduir una partida exactament igual de com s'ha gravat, facilitant controls de reproduir endavant, enrere i pausa.

public fitxerXML partida --> Fitxer XML que guarda la partida.

public bool pausa --> La reproducció està en pausa si pausa val cert.

public bool avancar --> La reproducció de la partida avança si avancar és cert, altrament retrocedeix.

private List<moviment> moviments --> Llistat de moviment que ha de realitzar l'objecte.

private int index --> Index del moviment actual que s'està representant.

public circuit c --> Objecte circuit de la partida.

public GameObject sonda --> Objecte de la partida.

Quaternion originalRotation --> Rotació actual de l'objecte.

private void endavant() --> Incrementa en una unitat l'index actual si pausa val cert i hi ha més moviments següents per realitzar.

private void enrrera() --> Decrementa en una unitat l'index actual si pausa val cert i hi ha moviments anteriors per realitzar.

void Start() --> Abans de començar la reproducció extreu els moviments del fitxerXML partida.

void Update () --> A cada frame aquest mètode canvia la posició i la rotació de l'objecte pel moviment indicat per l'index actual, tot seguit avança o retrocedeix segons l'atribut avancar.

void OnGUI() --> Executa la finestra on es representen el controls de la reproducció.

void controls(int windowID) --> Crea un botó per avançar, retrocedir o posar en pausa la reproducció de la partida.

9.1.18 Classe secció

Els objectes d'aquesta classe guarden la informació de tots els trams del circuit, per tal de poder-ne realitzar backups i generar-lo.

private Vector3 centre ---> Centre de la secció a l'escena.

private float radi_int -->Valor del radi intern de la secció.

private float radi_ext --> Valor del radi extern de la secció.

public seccio(Vector3 v, float radiIntern, float radiExtern) --> Crea una secció assignant-li un centre v, un radi intern radiItern i un radi extern radiExtern.

public Vector3 obtCentre() --> Retorna el centre actual de la secció.

public void posarCentre(Vector3 v) --> Assigna el valor v a l'atribut centre.

public float obtRadiInt() --> Retorna el radi intern actual de la secció.

public void posarRadiInt(float r) --> Assigna el valor r a l'atribut radi_int.

public float obtRadiExt() --> Retorna el radi intern actual de la secció.

public void posarRadiExt(float r) --> Assigna el valor r a l'atribut radi_int.

9.2 Procés de desenvolupament i algorismes més rellevants

Alguns del objectius més costos i importants estan relacionats amb la generació del model del circuit en 3D i la trajectòria.

El primer gran objectiu que es va assolir en el desenvolupament del codi font va ser la creació de la trajectòria i calcular-ne les tangents.

9.2.1 Mètode generarTangents()

Aquest mètode de la classe trajectòria analitza tots els punts de pas de la trajectòria per calcular-ne les tangents i obtenir-ne una corba el més suau possible.

El sistema escollit és traçar aquestes corbes aplicant el teorema de Beizer, que

consisteix en obtenir uns punts intermedis (nodes) a partir d'uns conjunts de punts i les seves tangents, les quals tracen una trajectòria.

Aquests nodes s'obtenen dels polígons formats per la connexió del punts de la trajectòria original amb les rectes, començant pel primer punt fins l'últim.

Per calcular aquests nodes s'ha utilitzat la classe BeizerPath, extreta de la comunitat de Unity3D.

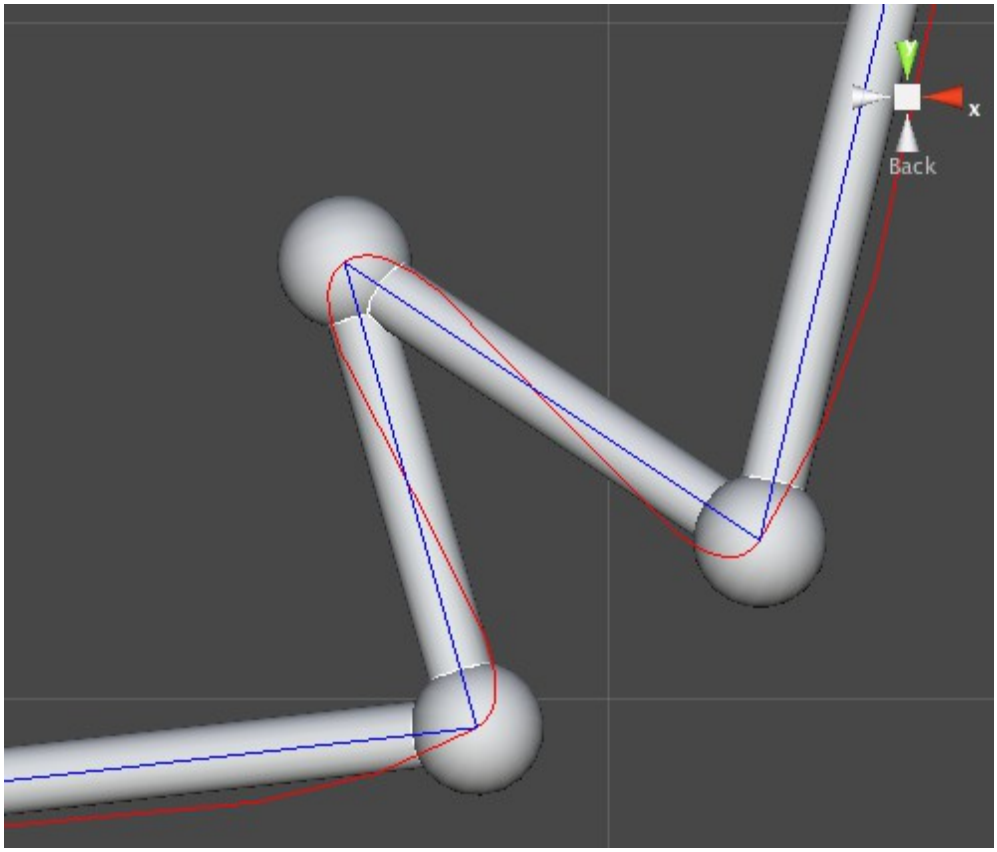


Figura 10: Exemple de tangents

En la imatge anterior es pot observar la trajectòria inicial en color blau i la trajectòria generada a partir de les tangents de color vermell.

Un cop trobats els punts de pas de la trajectòria amb les seves tangents, el següent pas és generar les seccions (anells) que es formen en cadascun dels punts on s'ha trobat una tangent.

9.2.2 Mètode generarAnells()

Aquest mètode de la classe circuit és l'encarregat de generar tots els anells que componen l'objecte, a fi d'obtenir els punts del perímetre que formen el circuit. Per tal d'aconseguir aquests punts, el mètode recorre la trajectòria punt a punt, situant 2 cercles amb radis definits per l'usuari amb centre al punt actual de la trajectòria, perpendicularment a la tangent d'aquell punt.

L'algorisme recorre la llista de tangents que es guarda al circuit, durant aquest recorregut es poden trobar 2 tipus de punts, per una banda els que són tangents pròpies de la generació del teorema de Beizer, i per l'altra, els punts que ha definit l'usuari i que, per tant, tenen les propietats de radi definides i que serviran per trobar el radi dels punts associats a les tangents.

Per trobar aquests radis s'invoca el mètode de la classe eines, anomenat porció, on per paràmetres se li passa el nombre de punts tangent trobats entre 2 punts de la trajectòria amb radi definit, els radis d'aquests dos punts i l'index del punt tangent que es vol consultar, i llavors retorna el valor proporcional que li toca per tal d'aconseguir un tram amb una obertura regular, sense “escalons”.

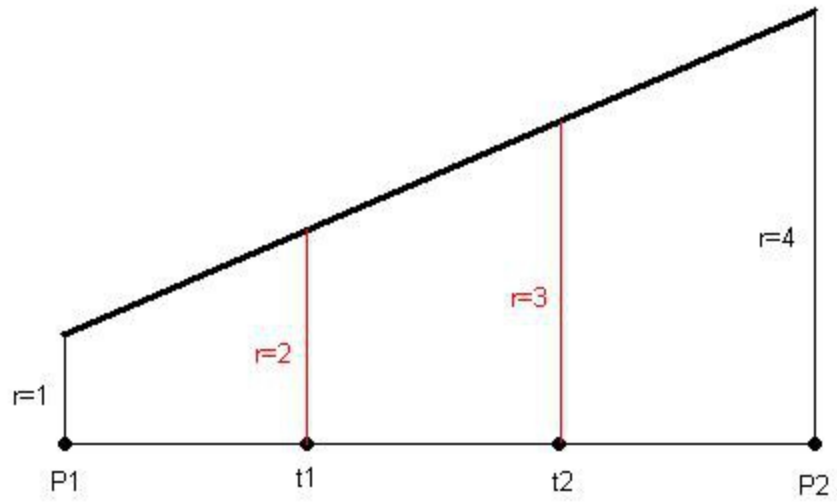


Figura 11: Radis de tangents

En la figura anterior es pot veure un exemple d'assignació dels radis de les tangents. En aquest cas el radi de P1 val 1 i el radi de P2 val 4, que són els radis que ha assignat l'usuari, i durant la generació del circuit en aquest tram de trajectòria han sorgit 2 punts més que no tenen valor assignat, per tant, en

calcular els radis que tocarien a t1 i t2, 2 i 3 respectivament.

Amb aquests passos s'obtenen tots els punts de pas del circuit, els seus radis i tangents pertinents. D'aquesta manera es poden obtenir els anells que no deixen de ser 2 cercles a l'espai. Per això és necessari el constructor de la classe cercle passant-li les variables radi, centre, direcció i nombre de vèrtexs.

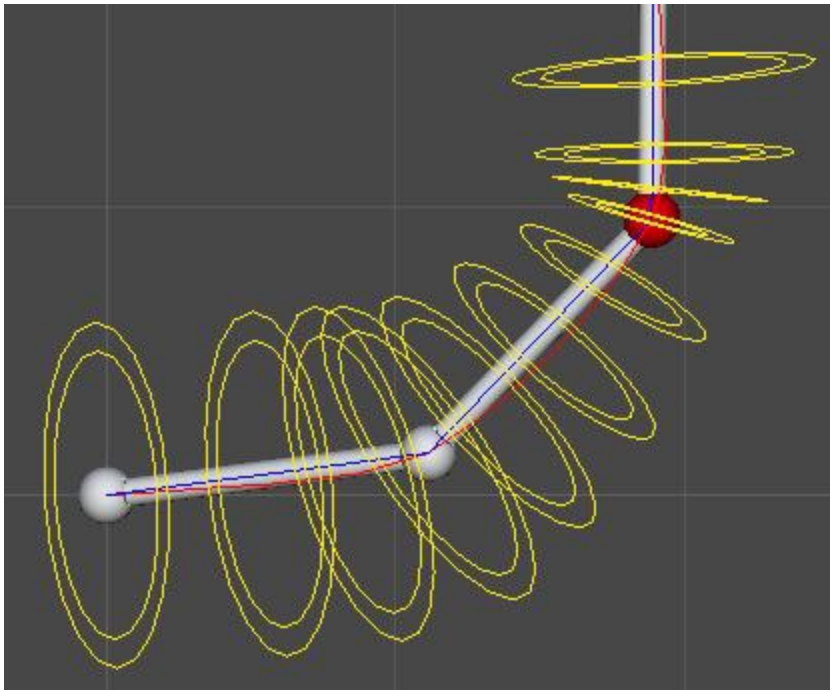


Figura 12: visualització dels anells i cercles

Tot seguit, es creen els anells amb el seu constructor amb les parelles de cercles que hi ha a cada punt.

9.2.3 Mètode GenerarObjecte()

Aquest mètode de la classe circuit és un dels més importants de l'aplicació, a partir dels anells creats anteriorment genera tots els vèrtexs i arestes de la malla per tal de poder obtenir-ne un objecte 3D.

El primer pas és obtenir els contorns dels cercles que componen els anells, per aconseguir-ho es recorre cada un dels anells i s'executa el mètode obtPunts() sobre cada cercle, creant una llista de punts de tants valors com nombre de vèrtexs, i tot seguit en calcula cada un mitjançant trigonometria bàsica i una rotació.

Primerament es genera punt a punt, com si el cercle estigués perpendicular a l'eix Z, per trobar les 3 coordenades dels vèrtexs s'apliquen les següents

fórmules.

```
angle = (2 * PI) / nombre vèrtex
x = centre(x) + radi * Cos(angle * i)
y = centre(y) + radi * Sin(angle * i)
z = centre(z)
```

A continuació s'aplica la rotació a cada punt, que s'extreu de la diferencia entre el vector (0,0,1) i el vector direcció assignat al cercle.

Un cop aconseguits tots els vèrtexs que compondran la malla s'han de definir els triangles, que són les agrupacions de tres d'aquests vèrtexs.

Per definir els triangles es crea una taula de valors, la qual conté els index dels vèrtexs dins la seva taula. Tres index consecutius representen un triangle.

Per exemple, per generar una malla com la següent:

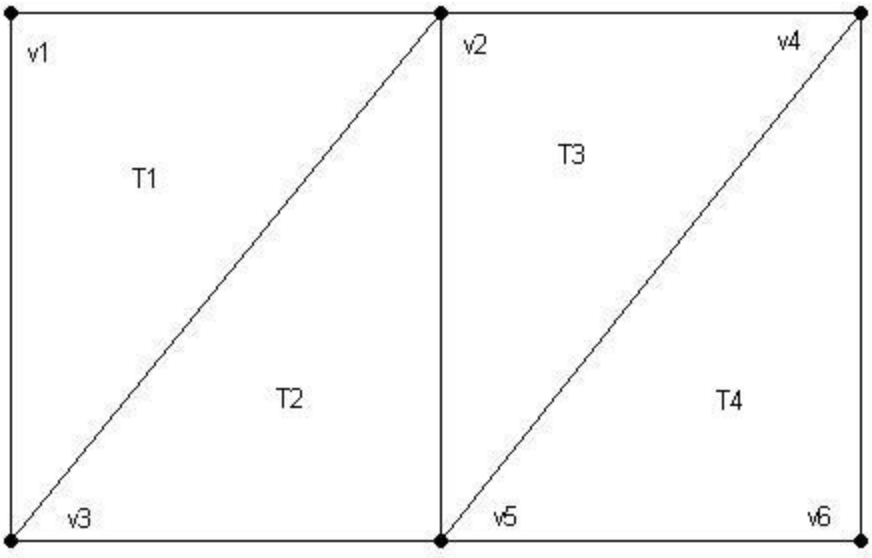


Figura 13: exemple de malla

Les taules de vèrtexs i triangles representatives són:

Posició	0	1	2	3	4	5
Vector	v1	v2	v3	v4	v5	v6

Triangles	T1			T2			T3			T4		
Index	0	1	2	1	2	4	1	4	3	3	4	5

Un cop creades les taules de triangles i vèrtexs ja es pot generar el GameObject i assignar-los a la component Mesh.

Un altre tema important d'aquesta aplicació és l'emmagatzematge a memòria, d'això se'n encarrega la classe fitxerXML, que guarda les dades en fitxers amb format .xml.

9.2.4 Mètode guardar()

Un cop definit el circuit, la mateixa classe disposa d'un mètode anomenat guardar, que passant-li per paràmetres la ruta dels fitxer final, el títol del circuit, la descripció i la dificultat genera una llista de strings que representen el circuit exactament amb l'estructura que s'ha definit en el projecte.

Aquesta mètode genera una estructura xml de la següent forma:

```
<?xml version="1.0" encoding="utf-8"?>
<circuit vertex="20">
<info>
<titol>forma d'essa</titol>
<descripcio>Això té forma d'essa</descripcio>
<dificultat>5</dificultat>
</info>
<trajectoria>
<punt>
<centre x="0,9891877" y="-2,66347" z="9,19671" />
<radi_intern valor="5" />
<radi_extern valor="6" />
</punt>
<punt>
<centre x="10,06122" y="-1,226457" z="5,489118" />
<radi_intern valor="5" />
<radi_extern valor="6" />
</punt>
<punt>
<centre x="15,51337" y="4,655309" z="4,471102" />
<radi_intern valor="5" />
<radi_extern valor="6" />
</punt>
</trajectoria>
</circuit>
```

Un cop generat el cos del fitxer aquest es passa a un objecte de tipus fitxerXML, mitjançant el mètode inserirLinia i, posteriorment, s'executa escriure per tal de generar un nou fitxer a memòria i inserir-hi el contingut.

9.2.5 Mètode finalitzar()

El mateix passa amb el mètode de la classe gravar_partida, un cop s'ha finalitzat una partida aquest guarda a memòria tots els moviments que ha fet l'usuari durant el joc. L'estructura d'una partida a memòria és la següent:

```
<?xml version="1.0" encoding="utf-8"?>
<partida>
<fitxer>Essa</fitxer>
<titol>forma d'essa</titol>
<colisions>8</colisions>
<data>22/8/2012 18:16:03</data>
<temps>00:00:12.3730000</temps>
<moviments>
<mov>
<pos x="0" y="0" z="0" />
<rot x="0" y="0" z="0" w="1" />
</mov>
<mov>
<pos x="-0,0009171838" y="-9,030549E-05" z="0,0003880964" />
<rot x="0" y="0" z="0" w="1" />
</mov>
<mov>
<pos x="-0,001834368" y="-0,000180611" z="0,0007761928" />
<rot x="0,01308932" y="-0,006544378" z="8,56706E-05"
w="0,999893" />
</mov>
<mov>
<pos x="-0,001834368" y="-0,000180611" z="0,0007761928" />
<rot x="0,01308932" y="-0,006544378" z="8,56706E-05"
w="0,999893" />
</mov>
<mov>
<pos x="-0,002751551" y="-0,0002709165" z="0,001164289" />
<rot x="0,01308932" y="-0,006544378" z="8,56706E-05"
w="0,999893" />
</mov>
<mov>
<pos x="-0,003668735" y="-0,000361222" z="0,001552386" />
<rot x="0,01308932" y="-0,006544378" z="8,56706E-05"
w="0,999893" />
</mov>
</moviments>
</partida>
```

Quan s'han aconseguit totes les eines d'edició i emmagatzematge ja es pot passar al procés de conducció. Aquest procés està dirigit principalment per la classe control_sonda, que a part d'anar guardant els moviments de l'usuari per posteriorment escriure'ls a memòria, controla mitjançant teclat i ratolí l'objecte i la càmera, i també controla en cada moment a quin tram de l'aplicació es troba l'objecte per saber quan la partida ha finalitzat.

Per saber si s'ha arribat al final de la partida es comprova la posició de

l'objecte, i es busca entre quins plans formats per les seccions del circuit es troba. Si es troba entre l'últim i el penúltim pla, llavors la partida finalitza guardant el fitxer xml resultant.

9.3 Diagrames de seqüència

Per tal de descriure el flux d'execució de l'aplicació, a continuació es mostren els diagrames de seqüència separats per funcionalitats.

9.3.1 Editar

Aquest és l'apartat més extens en quant a flux d'execució, s'han separat els diagrames en les diferents accions que té l'usuari en l'apartat d'edició.

a) Construir

Bloc d'execució de la construcció i edició de la trajectòria.

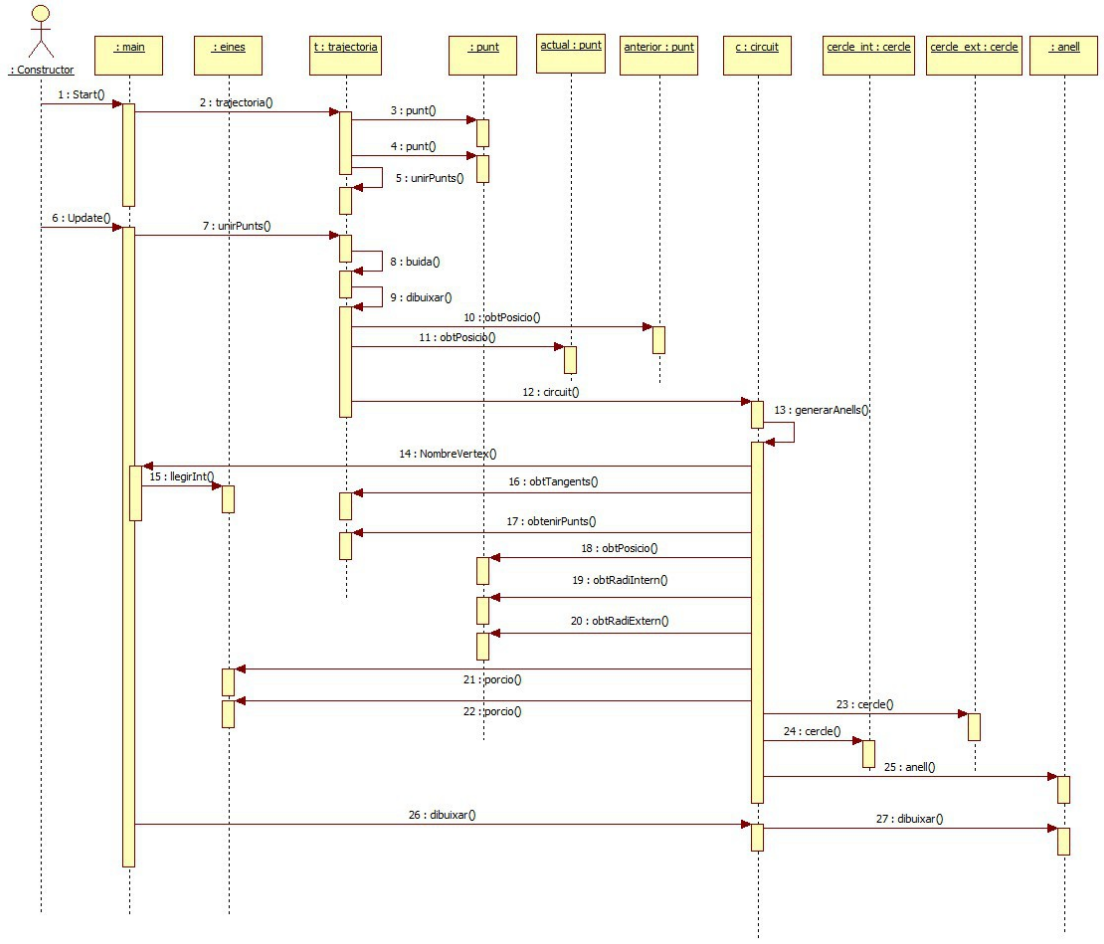


Diagrama de seqüència 1: Edició de la trajectòria

b) Selecció de circuits guardats

El següent diagrama llista els circuit guardats a memòria.

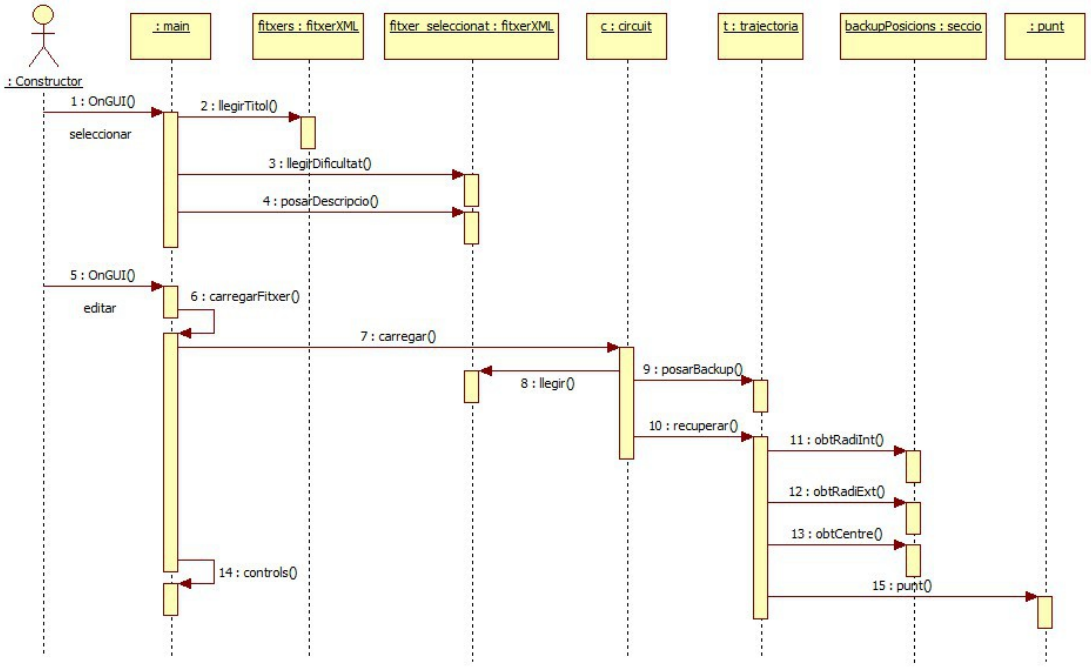


Diagrama de seqüència 2: Selecció de circuits

c) Carrega del circuit seleccionat

Aquesta seqüència prepara l'escena per editar un circuit a partir del fitxer guardat a memòria.

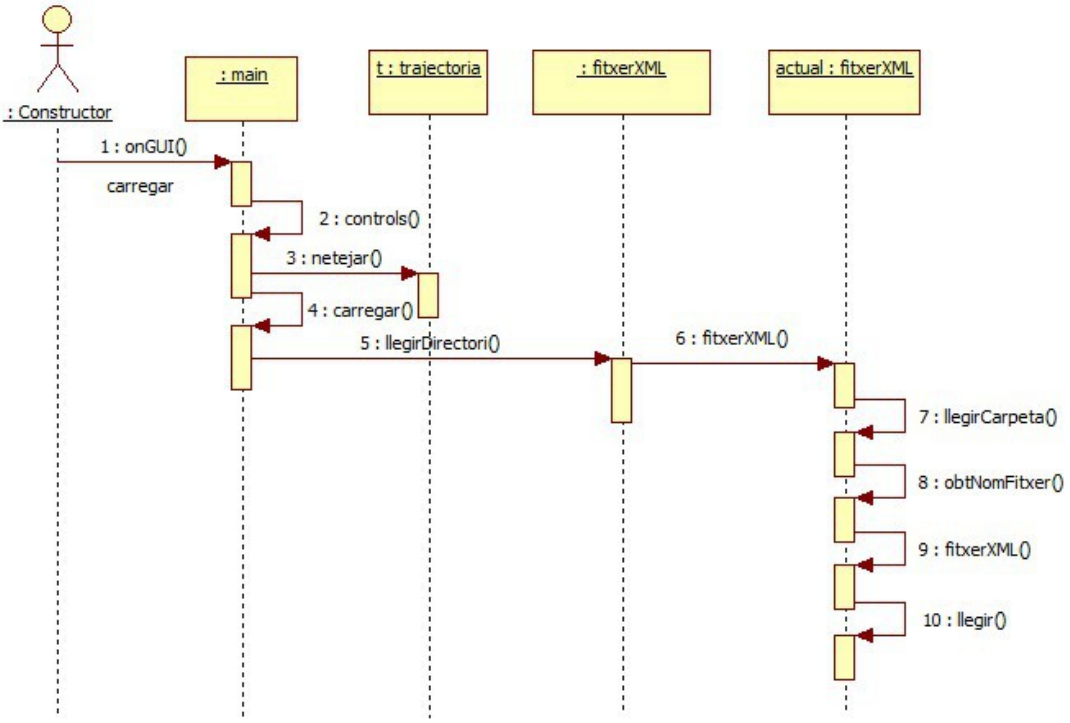


Diagrama de seqüència 3: Carregar circuit seleccionat

d) Generació del circuit editat

En aquest punt l'usuari vol generar el circuit després d'editar-ne la trajectòria.

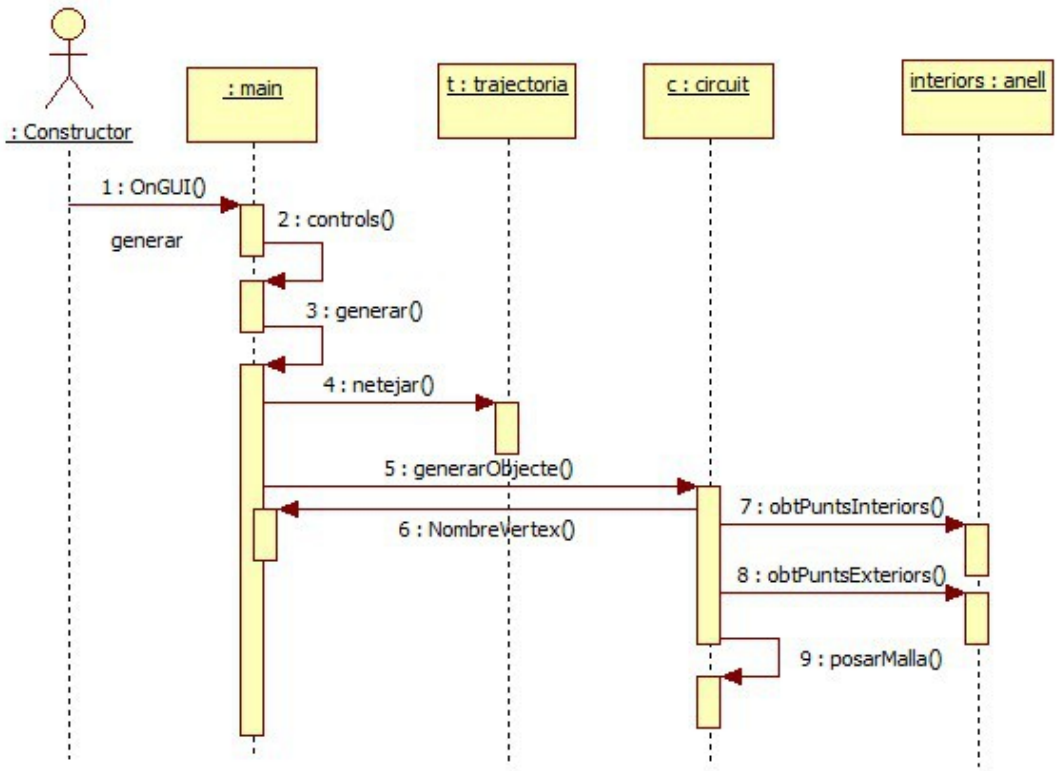


Diagrama de seqüència 4: Generació de l'objecte 3D

e) Selecció de reeditar

L'usuari ha carregat un circuit però vol reeditar-lo, aquesta és la seqüència:

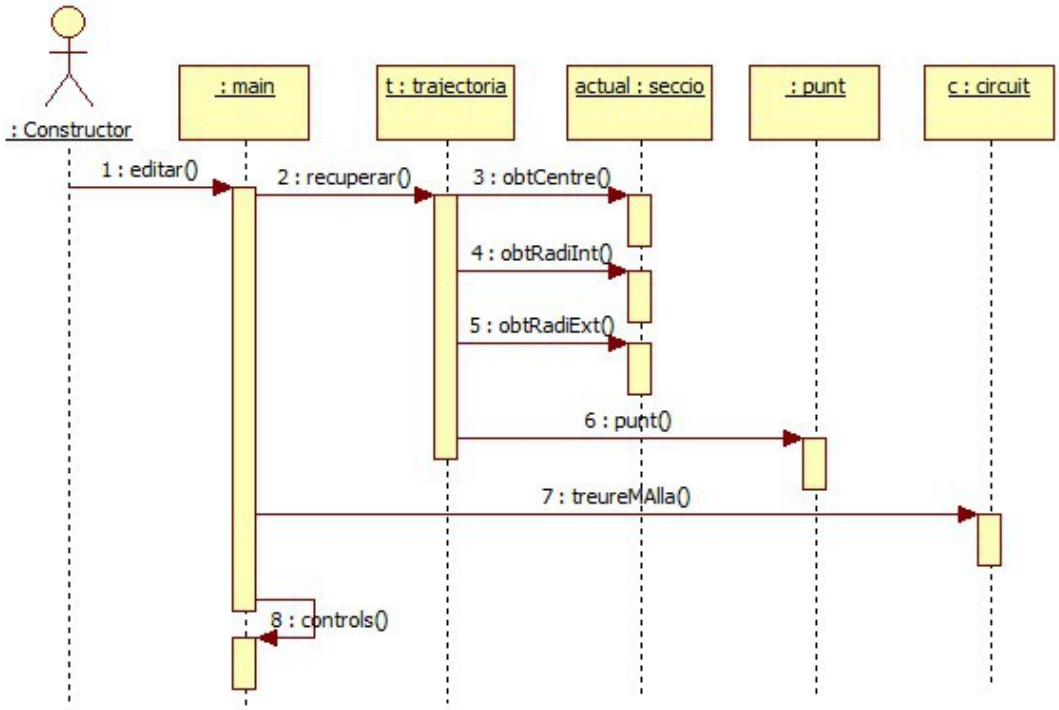


Diagrama de seqüència 5: Reedició

f) Guardat circuit

Un cop generat l'objecte l'usuari selecciona guardar el circuit.

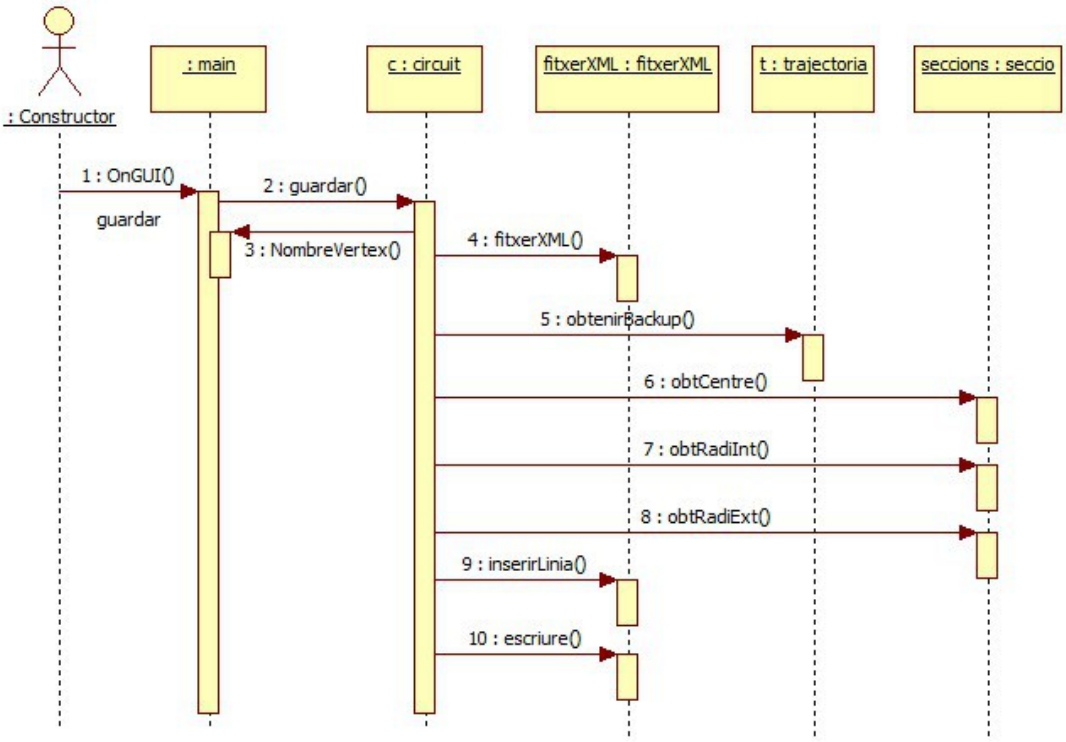


Diagrama de seqüència 6: Guardar circuit

9.3.2 Conducció del circuit

Un cop el circuit ha estat generat ja es pot començar la conducció, en el següent diagrama es pot veure el flux.

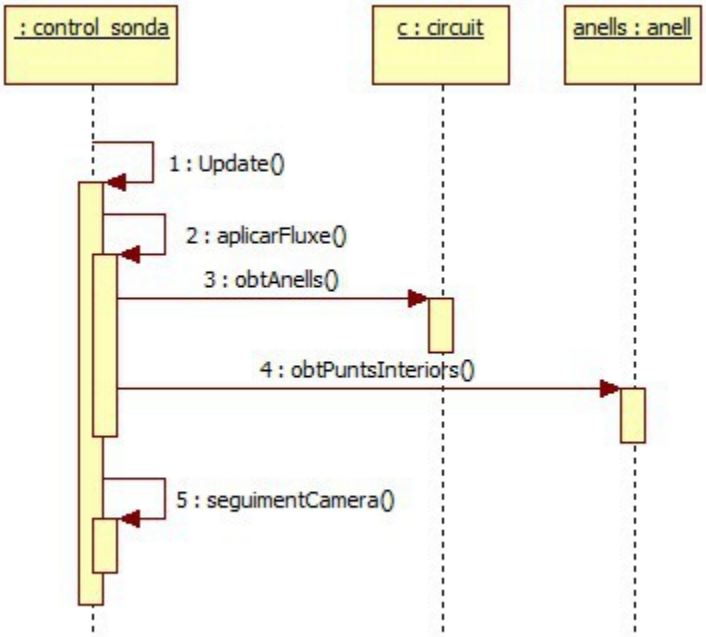


Diagrama de seqüència 7: conducció

9.3.3 Detecció de col·lisions

Al mateix temps que s'està executant la conducció, també ho fa el codi de detecció de col·lisions.

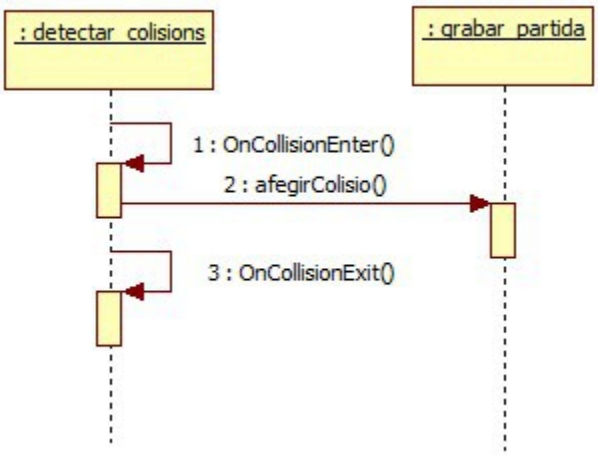


Diagrama de seqüència 8: Detecció de col·lisions

9.3.4 Gravar partida

En aquest moment també s'executa la gravació de la partida que genera el fitxer xml i l'emmagatzema.

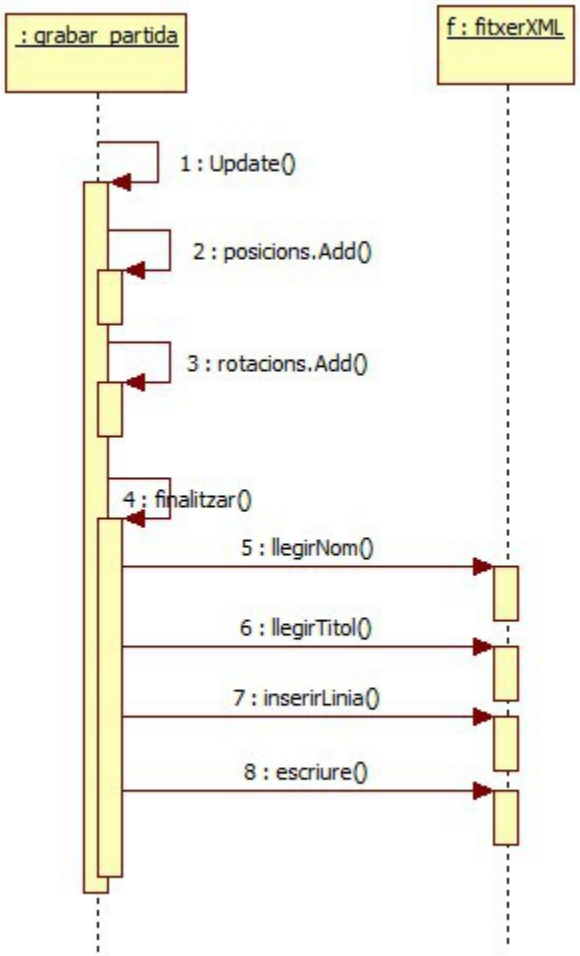


Diagrama de seqüència 9: Gravació de la partida

9.3.5 Llistat i càrrega de partides

Un cop hi ha partides guardades a memòria, l'usuari constructor ja les pot llistar per carrega-ne una, aquí es pot observar el flux d'execució.

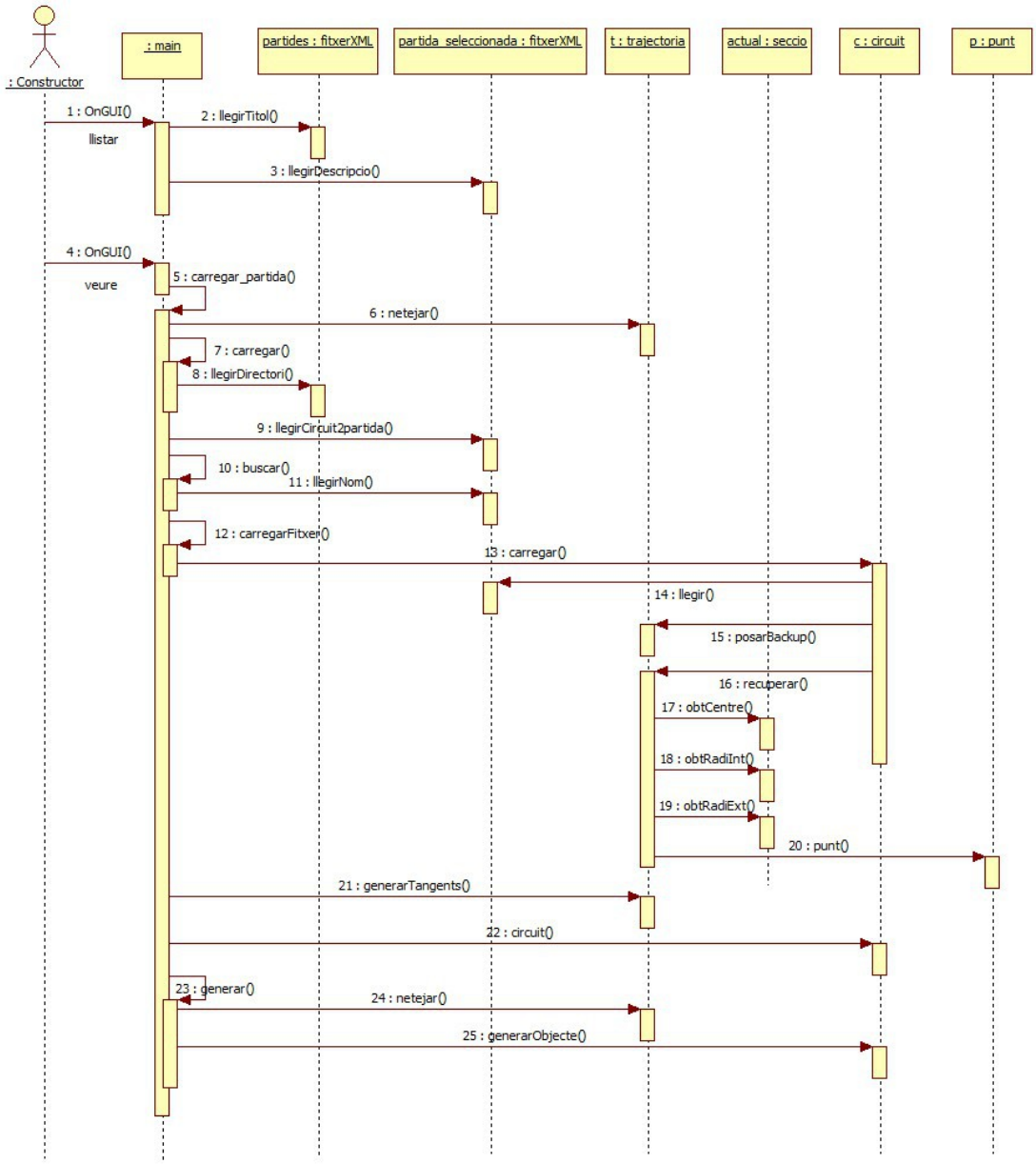


Diagrama de seqüència 10: Llistat i carrega de partides

9.3.6 Reproducció de la partida

En aquest diagrama es pot veure el flux d'execució de la reproducció, on en el mateix moment s'executa el diagrama de detecció de col·lisions per indicar-les a l'usuari.

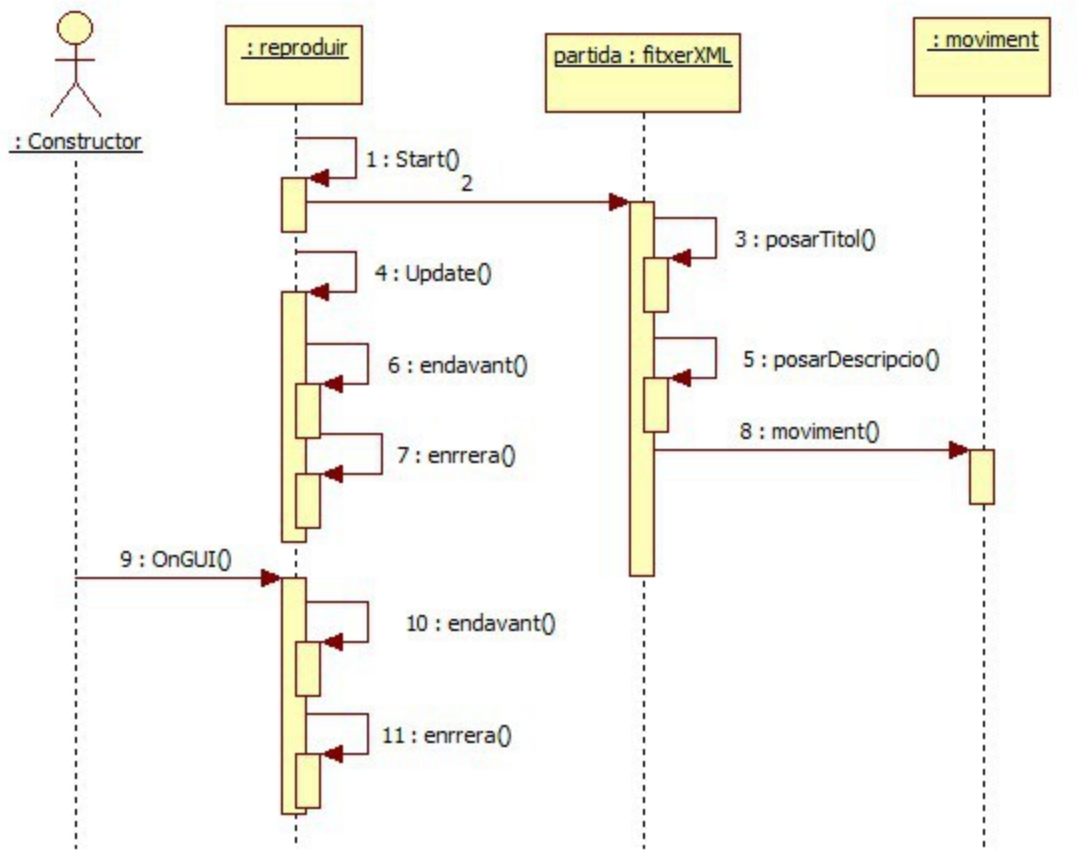


Diagrama de seqüència 11: Reproducció de la partida

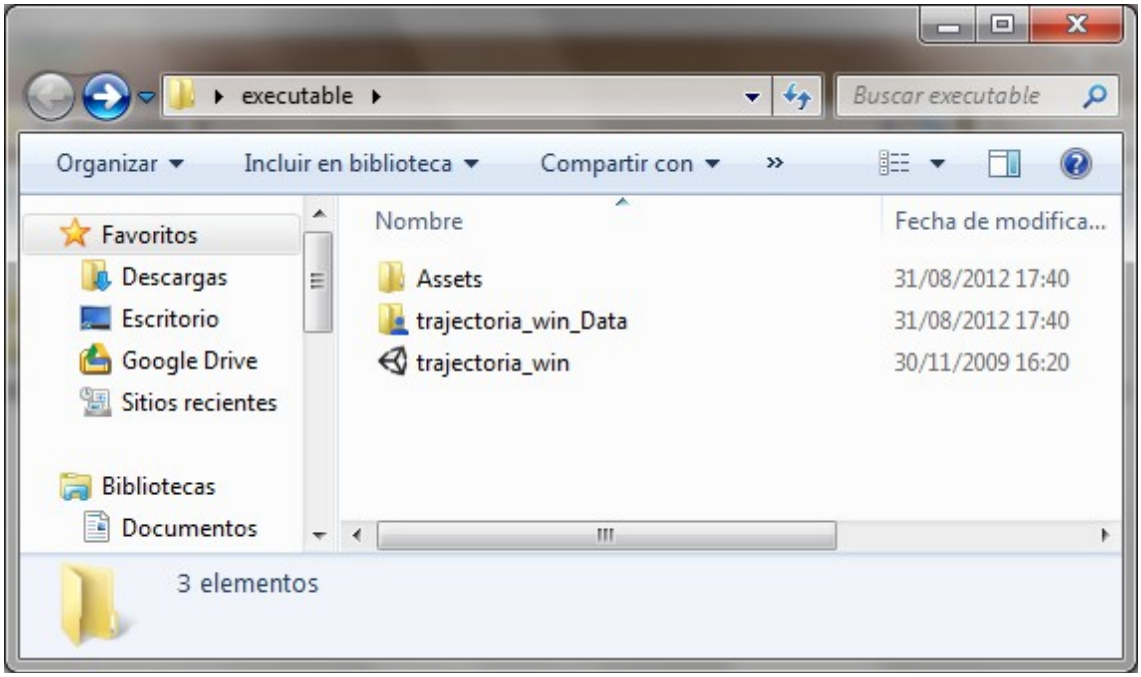
10 Resultats

En aquest apartat es mostren les pantalles resultants de l'aplicació, la seva funcionalitat, i com utilitzar-ne les opcions. A més de servir també com a manual d'usuari.

L'aplicació final del projecte es presenta en un executable, capaç d'iniciar-se en qualsevol ordinador, amb un sistema operatiu amb Windows XP, Windows Vista, o Windows 7. També es pot executar des de Mac OS o Linux si aquests disposen d'una màquina virtual de Windows, o d'algun programa que pugui executar .exe, com per exemple Wine per Linux.

10.1 Carpeta de treball

Per tal d'executar l'aplicació s'ha de descomprimir el .rar de l'aplicació en qualsevol directori, donant com a resultat una carpeta com la següent:



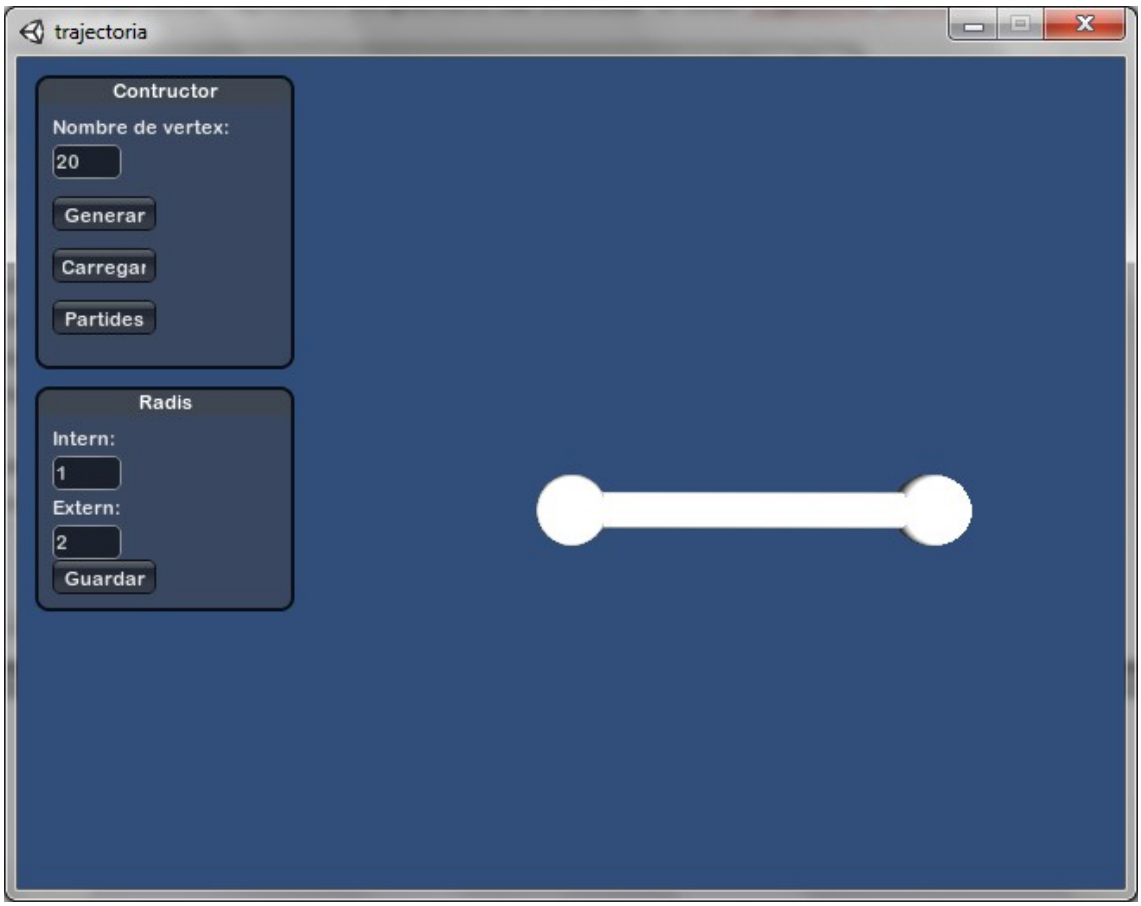
L'executable s'anomena **trajectoria_win.exe**, la carpeta **trajectoria_win_Data** conté diferents llibreries i complements necessaris per l'execució, i a la carpeta **Assets** s'hi troben les partides guardades, així com també els circuits.

En sistemes operatius amb restricció de permisos d'usuari o on l'usuari no és l'administrador, s'aconsella descomprimir la carpeta en un directori on no hi hagi restricció de lectura ni escriptura. També es poden assignar permisos a tots els fitxers i directoris que permetin a l'usuari llegir i escriure, ja que sinó, no es podrien guardar ni carregar partides i circuits.

Per iniciar l'aplicació tan sols cal executar el fitxer **trajectoria_win.exe**

10.2 Creació i definició de la trajectòria

Un cop executada l'aplicació apareix una pantalla com aquesta:



Pantalla 1: inicialització

Aquesta pantalla serveix la trajectòria per defecte, la qual es representa amb una recta i dos esferes a cada extrem.

El desplaçament per l'escena es realitza mitjançant teclat i ratolí, els cursors permeten moure la càmera endavant, enrere, a la dreta i a l'esquerra,

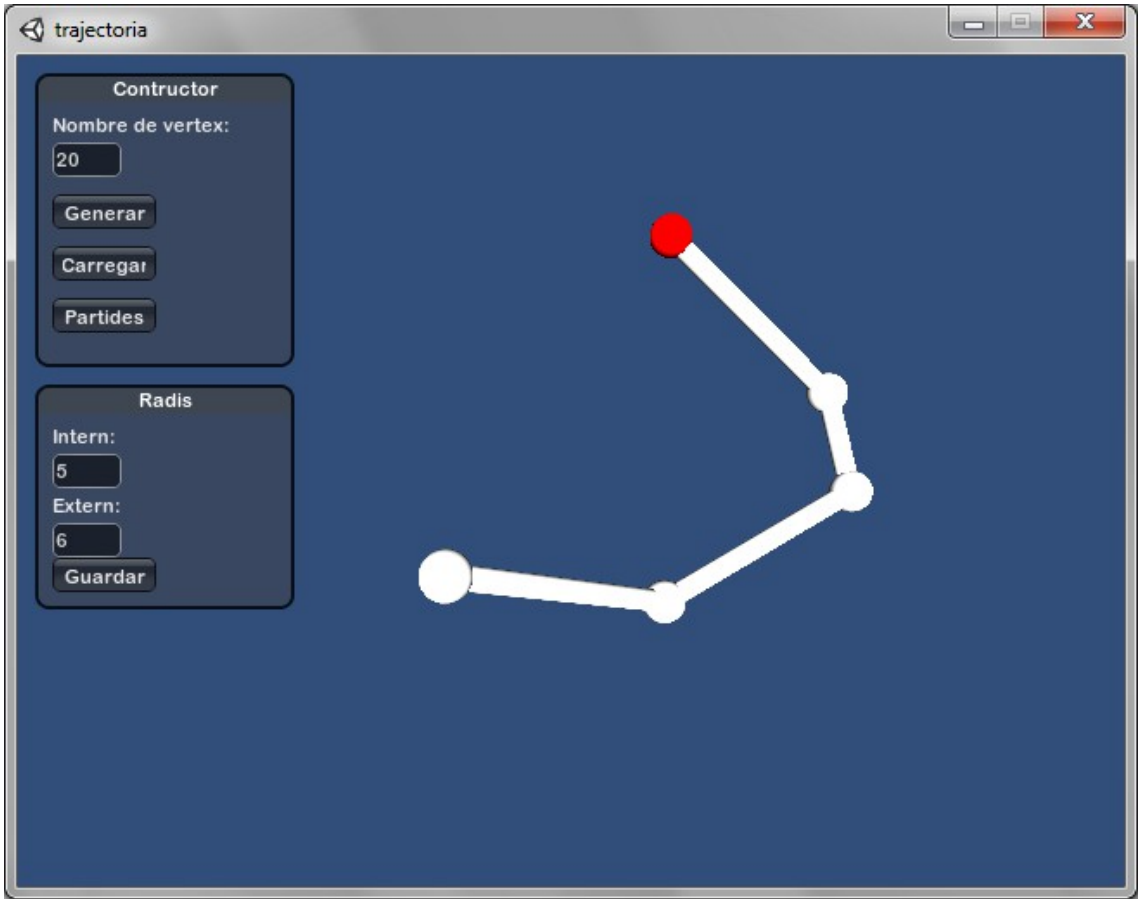
juntament amb les tecles Z i A que permeten moure-la avall i amunt, respectivament.

La rotació de la càmera es realitza amb el ratolí i la tecla d'espai, al pressionar aquesta tecla i moure el ratolí la càmera rota sobre si mateixa en la direcció desitjada.

Per a modificar la trajectòria n'hi ha prou amb arrossegar les esferes per l'escena amb el ratolí, pressionant el botó primari a l'hora que es mou el ratolí, d'aquesta manera es podran situar les esferes al punt on convingui.

Cada esfera representa una articulació de la trajectòria, aquestes es poden afegir pressionant amb el botó secundari sobre la recta, que uneix les esferes i automàticament apareix una nova articulació entre elles.

Amb tot això es pot aconseguir una trajectòria com la següent:



Pantalla 2: edició de la trajectòria

En aquesta pantalla es poden observar dues finestres a la part esquerra, **Constructor** i **Radis**.

La segona finestra, **Radis**, guarda els valors dels radis intern i extern al punt seleccionat amb vermell, el procés és el següent:

1. Per seleccionar un punt s'ha de pressionar amb el botó primari sobre d'ell.
2. Modificar els valors assignats al punt que es mostren a les caixes de text **Intern** i **Extern**. Per modificar-los n'hi ha prou en entrar 2 valors nous.
3. Pressionar **Guardar**.

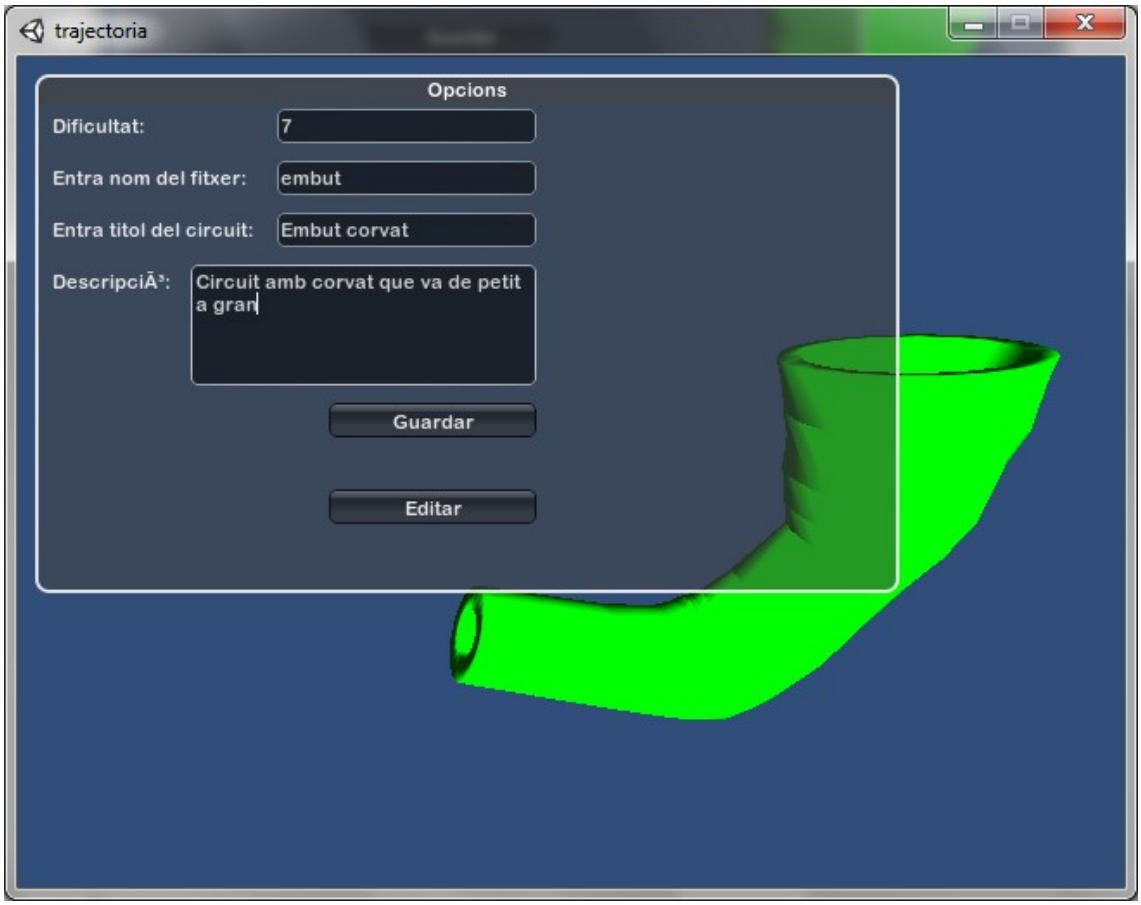
Un cop definida tota la trajectòria hi ha la possibilitat de generar el circuit de la següent manera:

1. Entrar el nombre de vèrtexs que tindran les seccions del circuit a la caixa anomenada **Nombre de Vèrtexs**.
2. Pressionar **Generar**.

Aquesta acció carrega una nova pantalla d'opcions.

10.3 Opcions del circuit

En aquest pas es presenta el circuit 3D generat amb les diferents opcions del circuit, guardar-lo o editar-lo.



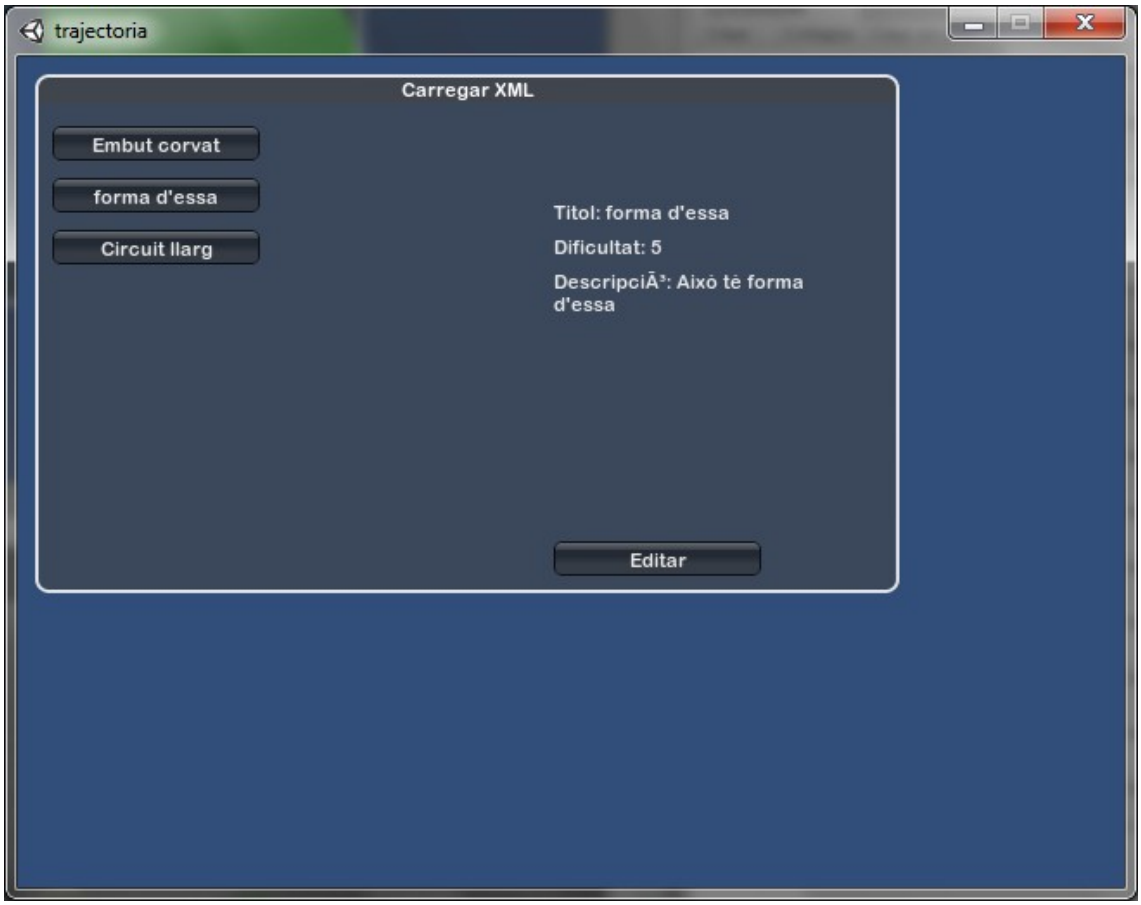
Pantalla 3: opcions del circuit

Per emmagatzemar el circuit a memòria és necessari omplir les dades que es sol·liciten per pantalla: **dificultat**, **nom del fitxer**, **títol** i **descripció**, i tot seguit pressionar **Guardar**.

El botó **Editar** de la part inferior de la finestra torna a la pantalla d'edició amb la trajectòria definida, en el moment abans de carregar el circuit, per fer-hi modificacions.

10.4 Veure circuits guardats

Tornant a la pantalla inicial, es pot observar que hi ha un botó anomenat Carrega. Aquest permet accedir al llistat de circuits guardats pressionant-lo.



Pantalla 4: llistat de circuits guardats

Es pot veure una llista de circuits a l'esquerra de la pantalla, on pressionant sobre ells es carrega la informació guardada a la dreta de la pantalla.

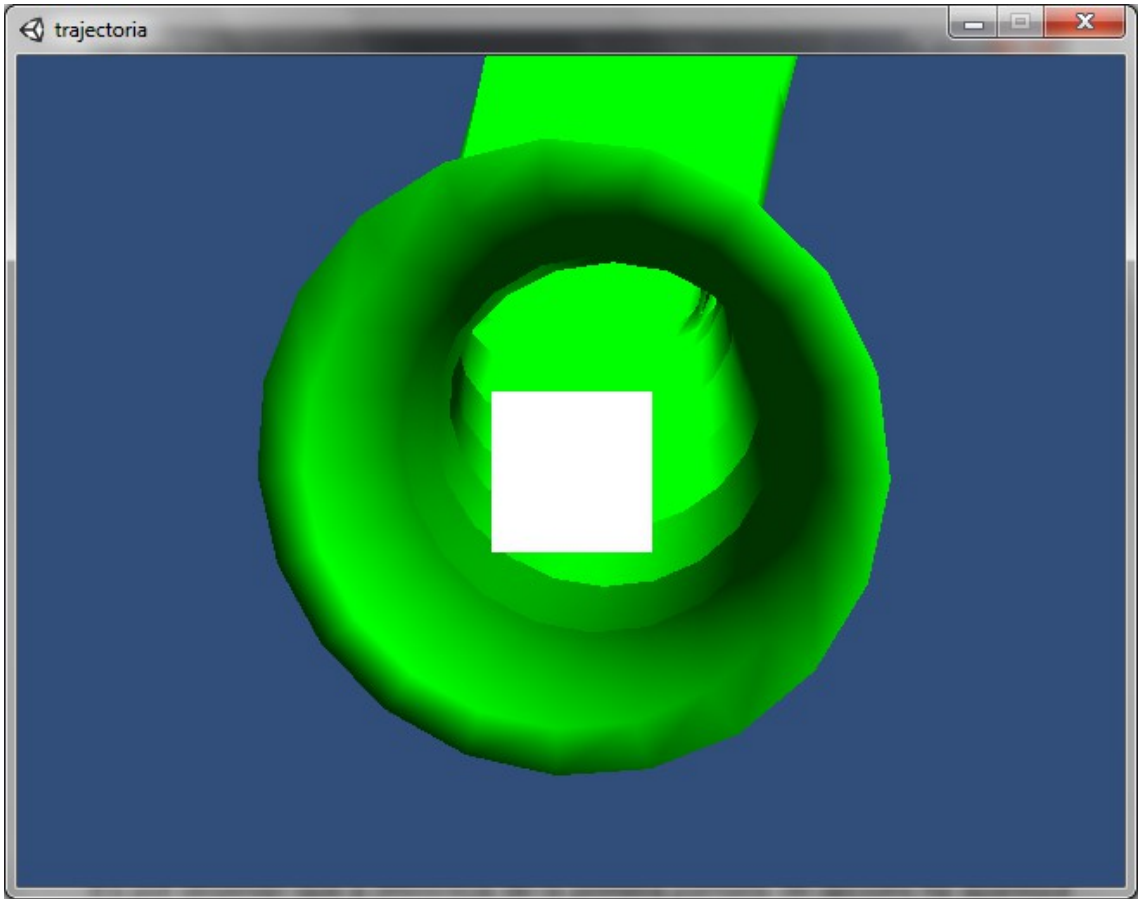
Un cop seleccionat el circuit prement **Editar** es torna altre vegada a la pantalla inicial, per tal d'editar-lo o generar-lo. Si es genera es torna a les opcions del circuit amb una opció més a la finestra d'opcions.

Pantalla 5: Opcions de circuit carregat

Es pot observar que a diferència de la primera pantalla d'opcions, en aquesta ocasió ha aparegut un nou botó anomenat **Conduir**. Aquest botó permet entrar en el procés de conducció del circuit i generar una partida a memòria, per posteriorment ser reproduïda.

10.5 Inici de conducció, partida nova

El procés de conducció es basa en el control de l'objecte mitjançant ratolí i teclat, igual que el procés d'edició el desplaçament s'efectua amb els cursors i les tecles A i Z, i la direcció ve marcada pel ratolí.

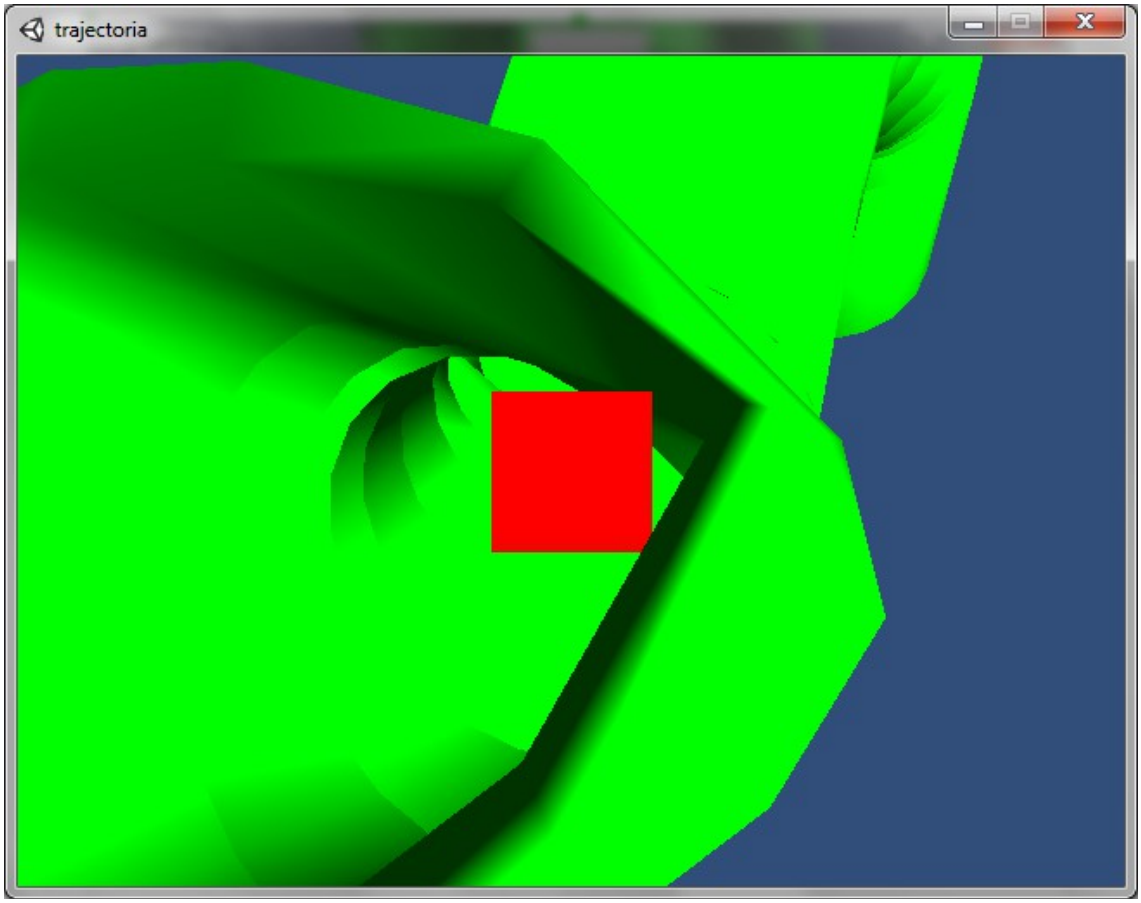


Pantalla 6: Inici de la conducció

Si durant el procés de conducció es col·lisiona amb el circuit es poden observar dos fets:

1. El color de l'objecte canvia a vermell indicant a l'usuari que està col·lisionant.
2. La paret del circuit es deforma.

Aquests dos fets es poden observar en la imatge que es mostra a continuació.



Pantalla 7: Detecció de col·lisions

El curs de la partida finalitza quan l'usuari ha arribat a l'altre extrem del circuit, guardant un fitxer nou a memòria on han quedat emmagatzemats tots els moviments que s'han fet, a més d'informació extra per l'usuari mèdic.

10.6 Llistar partides

A la pantalla inicial es pot observar un botó a la finestra **Constructor** anomenat **Partides**, seleccionant-lo dirigeix a una nova pantalla amb un funcionament igual al de llistar els circuit, però en aquest cas es mostren totes les partides guardades a memòria.

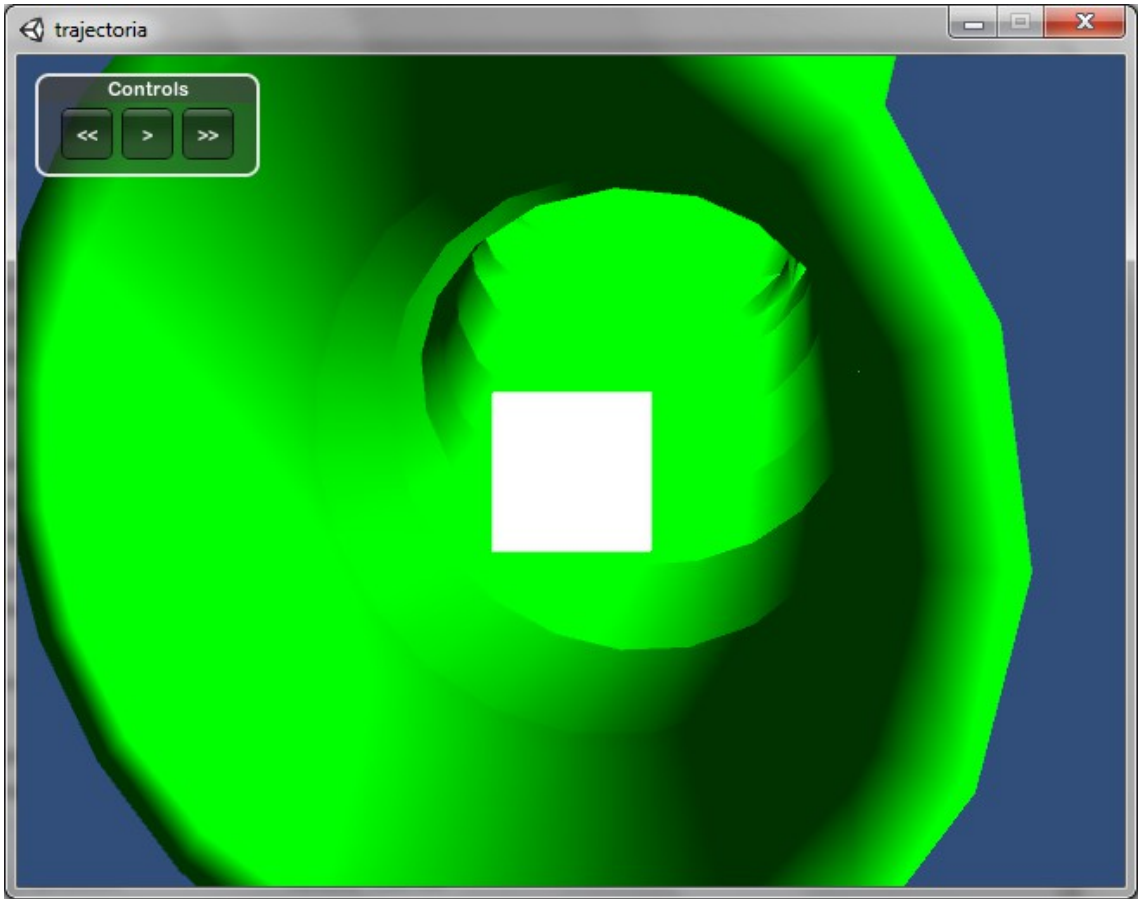
Pantalla 8: Llistat de partides

Es pot veure la informació de les partides a la part dreta de la pantalla: el títol, el nombre de col·lisions, la data de joc i els temps de joc.

El botó **Veure** dirigeix a la pantalla de reproducció, on l'usuari mèdic pot observar el procés de la partida.

10.7 Reproducció de la partida

En aquesta pantalla s'observa que hi ha 3 botons a la part superior esquerra. El botó central és el de **play** o **pausa**, que canvia en funció de quin està actiu, els altres dos són els botons d'**avançar** o **retrocedir**.



Pantalla 9: Reproducció de la partida

11 Conclusions

L'objectiu d'aquest projecte era desenvolupar un entorn informàtic que permetés definir circuits de forma tubular i tridimensionals per tal de fer navegar un objecte pel seu interior. Aquest entorn es podria usar en entorns de rehabilitació per estimular la memòria i la psicomotricitat fina dels pacients amb lesions cerebrals.

L'entorn que hem desenvolupat permet:

- **Definició d'una trajectòria a l'espai**

L'usuari és capaç de crear una trajectòria punt a punt dins l'escena de la forma i mida que necessiti, durant aquest procés l'usuari pot definir el nombre de trams per tal de poder-los posicionar i definir-ne els gruixos.

- **Crear circuits**

A partir de la trajectòria anterior, l'equip mèdic por crear un objecte 3D en forma de tub cilíndric de la forma i mida que ha definit anteriorment, d'aquesta manera queda assolit el segon objectiu.

- **Guardar circuits**

Després de crear el circuit es disposa d'un seguit d'eines, una de les quals permet emmagatzemar-lo a memòria per disposar d'ell en sessions posteriors.

- **Editar els circuits**

Disposant d'un o més circuits a memòria l'usuari pot carregar altra vegada el circuit a l'escena, per tal de modificar-lo.

- **Realització de partides dintre el circuit**

Amb els objectius anteriors assolits l'usuari pacient disposa d'un sistema de càrrega de circuits, el qual li permet seleccionar-ne un i iniciar una partida dirigint un objecte per l'interior d'aquest, mitjançant teclat i ratolí.

- **Guardar les partides**

Mentre el pacient realitza l'exercici, el sistema s'ocupa de guardar-ne tots els seus moviments i comptabilitzar les col·lisions que realitza, indicant-les a l'usuari a temps real.

- **Reproduir les partides**

Un cop finalitzada la partida, aquesta queda emmagatzemada a memòria per tal que l'usuari mèdic la pugui reproduir posteriorment. En el llistat de partides es pot veure el temps que s'ha tardat en finalitzar-la, l'hora d'inici i el nombre de col·lisions que s'han produït. A més, també permet carregar-la altre cop, i simular tots els moviments que ha realitzat el conductor disposant de botons tals com reproduir, avançar, retrocedir i pausa.

Valoració personal

La realització d'aquest projecte m'ha aportat un conjunt de coneixements molt enriquidors per a la meva formació. En primer lloc cal tenir en compte que no havia desenvolupat mai en un món tridimensional i això m'ha agradat molt, ja que les possibilitats en aquest tipus d'aplicacions són molt àmplies i penso que el món dels videojocs sempre ha estat molt atractiu.

Una altra novetat que m'ha aportat el projecte és conèixer el llenguatge C sharp, que és com el Java Script però a nivell personal l'he trobat més interessant pel fet que tracta el tema d'errors i debugar.

El que més m'ha motivat a l'hora de fer aquest treball és treballar amb l'entorn de Unity, a part del seu entorn de desenvolupament que està molt ben aconseguit, és molt intuïtiu i potent. M'ha cridat molt l'atenció el gran nombre de plataformes que accepta, sobretot amb tema mòbil i tablets que és un món que sempre m'ha atret, doncs ja havia realitzat algunes aplicacions utilitzant tecnologia web i crec que hi ha molt de futur en aquest àmbit.

Per tant, puc dir que la realització del treball ha estat una bona experiència per mi, tant a nivell personal com professional. Crec que a partir d'ara, aprofitant que ja tinc una base de coneixements en el món 3D, seguiré formant-me en aquest àmbit encara que sigui per satisfacció personal, però també per obrir nous ventalls en el meu món laboral.

12 Treballs futurs

Tot i que l'aplicació satisfà els requisits establerts, es pot valorar la possibilitat de millores i ampliacions per tal d'optimitzar-ne el funcionament.

Un dels punts millorables en usabilitat és l'edició de la trajectòria, mentre l'usuari va configurant la trajectòria podria mostrar una previsualització del circuit final a temps real.

Una altra qüestió a millorar seria la presentació de la pantalla de conducció, visualment es pot millorar aplicant objectes i imatges de decoració per tal que el pacient tingui una visió del joc més divertida.

La millora en el rendiment de l'aplicació és un tema a valorar, tot i que es pot executar en qualsevol PC actual sense problemes en alguns moments del procés de conducció, quan l'objecte col·lisiona amb el circuit i aquest últim es deforma, alguns ordinadors de característiques limitades es saturen i ralentitzen el procés.

Podria ser que l'usuari pacient vegi el joc una mica monòton a mesura de usar-lo, per a solucionar aquest fet es podria adaptar el joc i en comptes de generar un circuit tancat, podria crear camins o punts que l'usuari hagués de seguir o unir, d'aquesta manera hi hauria diferents varietats de joc.

Veient que les aplicacions realitzades amb Unity accepten un notable nombre de plataformes, es podria valorar exportar l'aplicació a altres sistemes.

Un altre punt a nivell tècnic que pot millorar el projecte és la possibilitat de crear camins alternatius al circuit (bifurcacions), per tal que l'usuari no només hagi de seguir una trajectòria, sinó que tingui la possibilitat de decidir per on passar.

13 Bibliografia

Unity Manual [pàgina a Internet], [consultada 2012] Disponible: http://docs.unity3d.com/Documentation/Manual/index.html

Unity Scripting Reference [pàgina a Internet], [consultada 2012] Disponible: http://docs.unity3d.com/Documentation/ScriptReference/index.html

Unity Community [pàgina a Internet], comunitat de usuari de Unity [consultada 2012] Disponible: http://forum.unity3d.com/

Unity 3D tutorials [pàgina a Internet], [consultada 2012] Disponible: http://unity3dtutorial.com/

Blackpawn [pàgina a Internet], Point in triangle test [consultada 15 maig 2012] Disponible: http://www.blackpawn.com/texts/pointinpoly/default.html

Herman Tulleken, DEV MAG [pàgina a Internet], Bézier Curves for your Games: A Tutorial [creada 5 abril 2011; consultada 8 juny 2012] Disponible: http://devmag.org.za/2011/04/05/bzier-curves-a-tutorial/

StarManta, Unifycomunity [pàgina a Internet], Tube Render [consultada 10 abril 2012] Disponible: http://wiki.unity3d.com/index.php?title=TubeRenderer

Unity Comunidad en Español [pàgina a Internet], script para deformar objetos en los impactos[creat 25 juliol 2010; consultat 5 juliol 2012] Disponible: http://unity3d-es.info/viewtopic.php?f=10&t=2510

Wikipedia [pàgina a Internet], Polígono regular [consultat 25 abril 2012] Disponible: http://es.wikipedia.org/wiki/Pol%C3%ADgono_regular

Wikipedia [pàgina a Internet], Curva de Béizer [consultat 1 juny 2012] Disponible: http://es.wikipedia.org/wiki/Curva_de_B%C3%A9zier

Wikipedia [pàgina a Iternet], Dessarrollo ágil de software [consultat 20 març 2012] Disponible: http://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software