5025 - Launchageddon: Tecnologia

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica

Realitzat per Enric Martínez Ibarra
i dirigit per Enric Martí Gòdia

Bellaterra, 17 de Setembre de 2012

El sotasignat, Enric Martí Gòdia

Professor/a de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en **Enric Martínez Ibarra**

I per tal que consti firma la present.

Signat: Enric Martí Gòdia

Bellaterra, 17 de Setembre de 2012

Agraïments

El temps és un factor vital, per aquest motiu vull començar agraint a les 11 i la 1 per acompanyar-me i fer-me costat tota la vida, ells m'han fet convertir-me en el que sóc avui. Agraeixo a les 2 tots els bons moments que m'ha fet passar. Què he de dir de les 3? Gràcies per formar part de la meva vida. Una vida sense les 4, les 5 i les 6 seria monòtona, gràcies per donar-li corda a la meva creativitat i compartir-ne la vostra. Quantes tardes he passat amb les 7 i les 8... moltíssimes gràcies per aguantar-me durant aquest projecte i fer-me somriure quan les coses es compliquen. Les 9 i les 10 han estat imprescindibles durant tot aquest temps, la combinació d'afecte, saviesa i diversió que els caracteritza m'han donat forces per seguir endavant. I finalment vull donar les gràcies a les 12, qui comença i acaba els dies al meu costat, per estimar-me tal i com sóc.

Vull agrair al meu tutor Enric Martí per la confiança i la llibertat que ens ha proporcionat al realitzar un projecte d'aquesta magnitud, creient sempre en el nostre potencial.

A tots vosaltres, gràcies.

Resum

El següent document correspon a la memòria del projecte fi de carrera d'Enginyeria Informàtica, i conté el disseny i la implementació del videojoc Launchageddon. Aquest correspon a un projecte desenvolupat entre tres persones.

En aquesta memòria es troba la part corresponent al disseny i la implementació dels controls del videojoc a través del reconeixement corporal que proporciona la tecnologia Kinect. També s'han dissenyat i implementat aplicacions test en les que es controlaran diferents tipus de joc amb la mateixa tecnologia.

El correcte funcionament d'aquests controladors ha estat validat per un grup d'usuaris amb la finalitat de realitzar el *feedback* necessari per a l'obtenció d'una versió definitiva.

Resumen

El siguiente documento corresponde a la memoria del proyecto de final de carrera de Ingeniería Informática, y contiene el diseño y la implementación del videojuego Launchageddon. Éste corresponde a un proyecto desarrollado entre tres personas.

En esta memoria se encuentra la parte correspondiente al diseño y la implementación de los controles del videojuego a través del reconocimiento corporal que proporciona la tecnología Kinect. También se han diseñado e implementado aplicaciones test en las que se controlarán diferentes tipos de juego con la misma tecnología.

El correcto funcionamiento de estos controladores ha sido validado por un grupo de usuarios con la finalidad de realizar el *feedback* necesario para la obtención de una versión definitiva.

Abstract

The following document deals with the report of the final Information Technologies degree which contains the design and implementation of Launchageddon game. This belongs to a project developed by three people.

In this report you will find the relevant design and implementation of game controllers based on the body recognition provided by Kinect technology. Have also been designed and implemented test applications in order to control different types of game with the same technology.

The proper functioning of these controllers has been validated by a user group in order to perform the necessary feedback to obtain a final version.

ÍNDEX

1- INTRODUCCIÓ

Un videojoc és un software creat generalment per a l'entreteniment i basat en la interacció entre una o varies persones mitjançant un controlador i un dispositiu electrònic on s'executa aquest. El dispositiu pot ser un ordinador, una màquina "Arcade", una videoconsola o un dispositiu portàtil, els quals són coneguts com plataformes de desenvolupament.

Els videojocs varen ser originats a la dècada dels anys 40, on després del final de la segona guerra mundial, les potències vencedores van construir els primers supercomputadors programables. Aquests, com a medi de comunicació, són un producte cultural que corresponen a un context, a una societat i a unes finalitats. També canvien des de la seva capacitat com a tecnologia fins a la capacitat de contingut, en un procés que generalment està constituït per persones de diferents disciplines (programadors, dissenyadors gràfics, escriptors, etc.) i per això han estat revaloritzats durant l'ultima dècada

Hi ha diferents tipus de videojocs. Aquests es poden classificar en gèneres depenent de la seva representació gràfica, el tipus d'interacció entre el jugador i la màquina, l'ambientació i el sistema de joc, sent aquest l'últim criteri més habitual a tenir en compte. A continuació s'esmenten els diferents gèneres:

- **Acció:** Aquest gènere es caracteritza per l'objectiu d'evitar la mort del nostre personatge. Inclou jocs de trets, jocs de lluita i jocs de plataformes.
- **Simulació:** Marca un aspecte real de la vida, portat a un videojoc. Inclou simulació de combat, construcció, simulació de vida, simulació musical, esports i carreres.
- **Agilitat mental:** La finalitat d'aquests jocs és que l'usuari pensi i agilitzi la ment resolent exercissis amb una dificultat progressiva.
- **Aventura:** El jugador s'encarna en el protagonista amb la finalitat de resoldre incògnites i trencaclosques amb diferents objectius. Inclou aventura gràfica i rol.
- **Arcade:** Característics per la simplicitat d'acció ràpida en la jugabilitat, van tenir el seu moment de glòria als anys 80 (Màquines recreatives).

Donat el meu interès pel tema i el dels meus companys i tenint en compte les eines que se'ns proporcionen avui en dia, tres alumnes d'enginyeria informàtica hem proposat realitzar un

projecte conjunt amb l'objectiu de crear un videojoc que combini entreteniment, creació i destrucció, controlat amb el propi cos, sense necessitat d'estar en contacte amb un dispositiu extern com per exemple un teclat. El joc serà anomenat "Lauchaggedon".

Per tal d'aconseguir aquest control es farà ús de la tecnologia que ens proporciona Kinect, un dispositiu que ens permetrà reconèixer els moviments del nostre cos per tal d'interactuar d'una manera més natural amb el videojoc.

Personalment, el fet de controlar un videojoc a través dels moviments sempre m'ha fascinat. La ciència ficció, fa anys que ens il·lusiona amb el control de les tecnologies mitjançant gestos corporals, citant per exemple la pel·lícula futurista "*Minority Report*" on es mostrava un futur proper on la gent interactuava amb ordinadors d'aquestes característiques.

Les aplicacions de Kinect no es limiten al món dels videojocs, és a dir, en el moment en què es demostri que aquest és una eina suficientment estable i fiable, es podrà començar a ampliar el seu ús, tant el camp de la medicina, realitzant una operació a distància controlant un robot amb els moviments de les mans, com a l'ensenyament al permetre reproduir presentacions sense tenir cap mena de contacte físic amb un dispositiu extern.

Els components d'aquest projecte compartim l'afició per el món dels videojocs, vàrem coincidir a una assignatura enfocada als gràfics per computador, on conjuntament vam realitzar un videojoc, creant nosaltres mateixos un motor gràfic que ens permetés dur a terme el projecte en base a les nostres necessitats. A arrel d'aquella experiència, uns quants components ens vam adonar que formàvem un equip interessant.

Hem definit uns objectius comuns per tal de desenvolupar el videojoc:

- Realitzar un Gameconcept en comú.
- Desenvolupar un joc en 3D, tot i basant-nos en el concepte de joc proposat.
- Dividir la feina equitativament en tres mòduls, que en el seu conjunt formin el videojoc proposat.
- Dur a terme la integració dels tres mòduls i posteriorment verificar el seu correcte funcionament.
- Solucionar els problemes que sorgeixen durant la integració.

En el següent punt es procedeix a la descripció del Gameconcept, un document realitzat en comú a l'inici del projecte on es defineixen tots els aspectes del joc. Aquest ens ha servit a tots els components com a guia per a la implementació del videojoc.

1.1 LAUNCHAGEDDON: GAMECONCEPT

Aquest apartat explica a grans trets en què consisteix el projecte de crear un videojoc i com neix la idea, així com sobre quins objectius hem treballat per elaborar el concepte de joc i en què ens hem inspirat.

També detalla sobre quins objectius hem treballat per elaborar el concepte de joc i en què ens hem inspirat. S'inclou l'estat de l'art realitzat per posar sobre la taula els requeriments necessaris en la tecnologia per al desenvolupament i la metodologia de treball definida per dur a terme totes i cadascuna de les tasques. Al final veurem un diagrama de mòduls del projecte conjunt. Aquest ens servirà per identificar cadascuna de les parts que composen el projecte i les interaccions, tant internes entre ells com externes amb la tecnologia, interfícies, usuari, etc.

Abans de començar a treballar en el projecte en si, s'ha realitzat una valoració sobre les diferents opcions, tenint en compte les possibilitats de l'equip, factors que depenen bàsicament del temps i recursos disponibles.

Una forma de definir el concepte d'un joc és buscar els trets característics que millor el descriguin. Durant la valoració es va fer un llistat de paraules clau, les quals defineixin els objectius generals que ens hem marcat per al joc. Aquestes paraules són: divertit, modern, ràpid, senzill, competitiu , estratègic i portable.

1.1.1 Perquè i amb quina finalitat?

- **Divertit:** Com la pròpia paraula indica un videojoc no deixa de ser una experiència que ha de divertir, en concret una experiència interactiva, la seva raó de ser és fer passar una bona estona a la persona que està jugant.
- **Modern:** És molt difícil crear un concepte que sigui completament original avui dia. No obstant això, sempre pot haver-hi algun element fora del comú, que faci l'aplicació més interessant per als usuaris. Aquest és un dels motius principals pels quals s'inclou Kinect en el joc, ja que és una tecnologia relativament moderna.

- **Portable:** El fet de voler una aplicació que sigui descrita amb totes les paraules anteriors té molts “perquès” i un d'ells és que com més senzilla i actual sigui l'aplicació més fàcil serà de portar-la a plataformes mòbils (Android, iOS) i tenir un valor de mercat. Avui en dia aquesta característica és molt valorada, degut a que la gran majoria de companyies realitzen videojocs multi plataforma amb la finalitat de poder abastir una major quantitat i diversitat de públic.
- **Ràpid**. Per assegurar la característica anterior, hauria de ser un joc ràpid de jugar, una aplicació que es pugui usar en un àmbit casual que no requereixi massa concentració. D'aquesta manera resulta més senzill que l'usuari decideixi usar l'aplicació.
- **Senzill:** Considerem que un joc enfocat a plataformes mòbils a part de ser ràpid ha de ser senzill. D'aquesta manera a l'usuari li costarà menys comprendre la dinàmica de joc i la possibilitat que un major nombre de persones ho provin augmentarà. A part d'aquests motius el fet de les limitacions de recursos també influeixen, ja que no podem optar a un projecte excessivament complex, preferim reduir aquest factor a un nivell factible per a nosaltres i fer un producte de la màxima qualitat possible.
- **Competitiu:** No solament interessa en un joc que un usuari ho provi, sinó que a més torni a usar-ho. Fer un joc competitiu és una bona manera de mantenir actius als usuaris, a part de que resulta més divertit i motivador per a aquests.
- **Estratègic:** Per a la nostra pròpia motivació i la del propi usuari un dels objectius principals és crear un “repte” amb uns objectius clars. És un dels principis dels jocs i una dels nostres propis desafiaments. No hem de limitar-nos a crear una aplicació gràfica visual. L'objectiu és que l'aplicació tingui una funcionalitat ben definida.
- **En 3D:** Considerem que un joc en tres dimensions és més complex que un de dues. Les físiques actuen en tres eixos de coordenades i es pot enfocar un mateix nivell de moltes perspectives diferents, per tant, creiem que aquesta característica és un valor afegit.

En les següents línies es descriu la mecànica del joc, inclou objectius, jugabilitat i les fonts d'inspiració.

1.1.2 Mecànica de Joc

La nostra principal font d'inspiració ha estat el joc anomenat Angry Birds de *Rovio Mobile* [AngBir], un joc realitzat en dues dimensions que bàsicament tracta de llençar uns ocells per destruir una sèrie d'estructures i enemics.

Launchageddon és un joc que consisteix a enderrocar objectes situats dins d'unes determinades estructures mitjançant una variada però limitada gamma de llançaments en un entorn 3D.

Les **estructures** consisteixen en diferents tipus de peces o blocs amuntegats entre si formant una estructura global, com si es tractés de derrocar castells composts per les clàssiques peces de joguina fetes amb fusta. La figura 1 mostra el disseny d'un possible escenari on podem trobar la construcció principal a enderrocar i objectes decoratius com per exemple palmeres o roques.

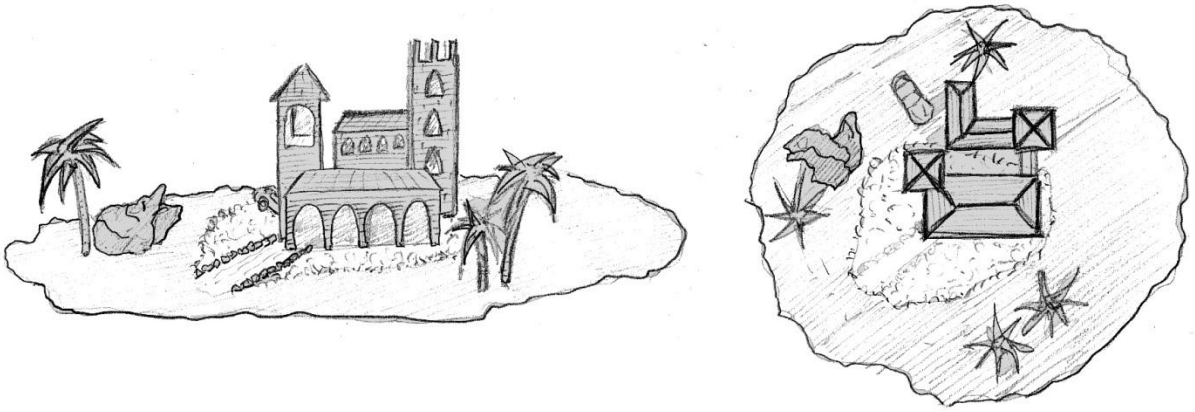

Figura 1. Disseny inicial d'un escenari

Dins d'aquestes estructures es troben una sèrie d'objectes especials que el jugador ha de picar per aconseguir el major nombre possible de punts.

Els trets s'efectuen amb un personatge que es llança des d'una posició determinada cap a lla direcció que el jugador triï, amb la finalitat d'enderrocar l'estructura (figura 4).

La jugabilitat es complementa amb dues variables diferents: la primera d'elles és el tipus de roba o vestit que porta el jugador, la qual modificarà el tipus de llançament que s'efectuarà

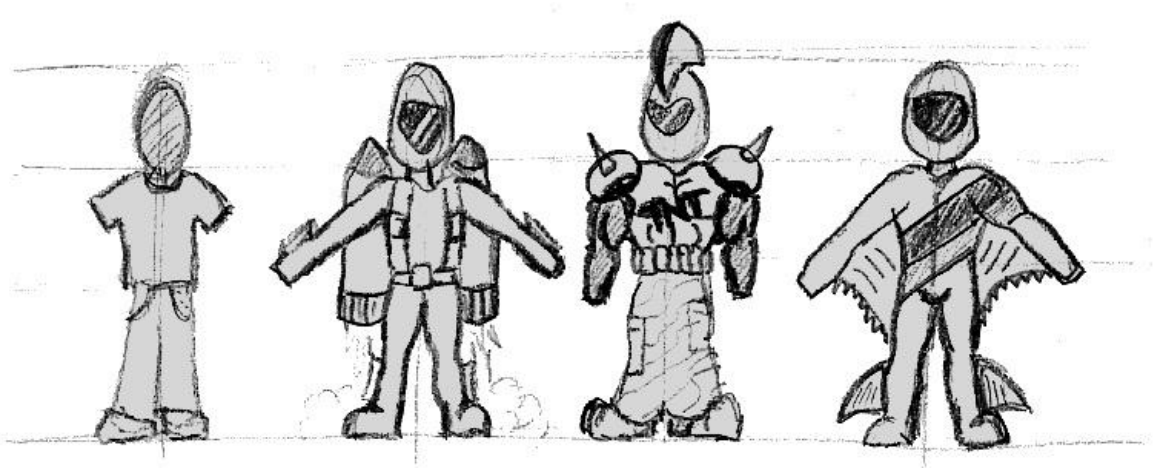

Figura 2. Disseny dels diferents tipus de llançament

Es podrà tirar entre diferents vestimentes, tal i com es mostra a la figura 2:

- **Vestimenta Normal**: Llançament més basic, un tir parabòlic.
- **Vestimenta Paracaigudista**: Mentre vola, permet modificar la trajectòria.
- **Vestimenta Propulsada:** Permet llançar-se a major velocitat contra l'escenari.
- **Vestimenta Antigravetat:** Ignora les lleis de la gravetat i provoca un rebot a xocar.

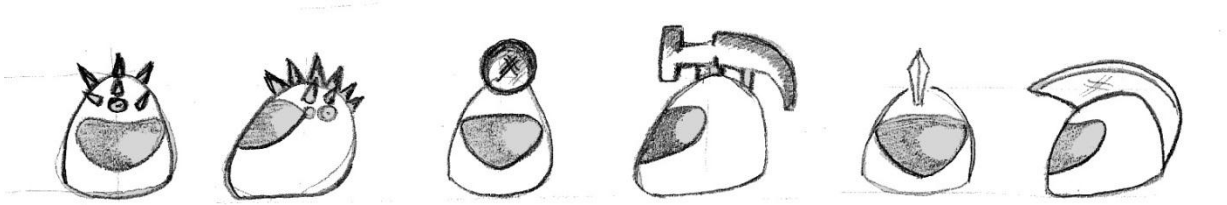

Figura 3. Disseny inicial dels cascs

D'altre banda podem triar entre una gamma de quatre cascs diferents per al personatge (figura 3), que atorguen diferents poders un cop el nostre personatge impacta contra l'escenari, aquests són:

- **Casc Normal:** Casc bàsic. No ofereix cap avantatge.
- **Casc Destral:** Destrueix tots els elements que toca gràcies a la seva destral.
- **Casc Explosiu:** Permet detonar una explosió per destruir parts de l'escenari.
- **Casc Científic:** Ofereix la visió a través dels elements per localitzar objectius ocults.

Una de les qualitats de la jugabilitat i que a més atorga un punt d'estratègia és que els tipus de llançament poden combinar-se amb els tipus de cascos, de manera que el jugador pot provar diferents combinacions per aconseguir els seus objectius.

Aquesta estratègia també consisteix que el jugador pensi bé quins trets utilitzarà, ja que aquests estaran limitats d'una forma diferent per a cada fase. Cada fase també tindrà un nombre de punts a batre per superar-ho, que s'hauran d'aconseguir amb un nombre de rondes diferents per a cada fase.

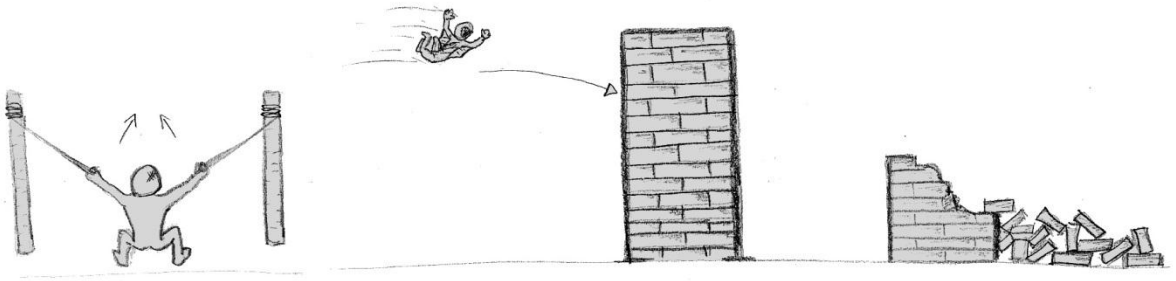

Figura 4. Llançament i destrucció de l'escenari

1.1.3 ELECCIÓ DE L'ENTORN DE DESENVOLUPAMENT

Prèviament al desenvolupament del projecte hem realitzat un estudi amb l'idea d'arribar a un consens sobre quin entorn i eines utilitzarem, tenint en compte les característiques definides al concepte de joc.

L'estudi s'ha basat en la cerca d'un motor gràfic que pogués servir-nos per dur a terme el projecte. També s'ha valorat la implementació d'un propi.

Per a realitzar l'estudi hem valorat tres possibilitats, dos són motors gràfics existents (*Unity3* i *Unreal Engine*), mentre que l'ultima d'elles és la de realitzar un motor gràfic propi.

Opció : Ús d'un motor gràfic extern

Entre les coses bones que aporta l'ús d'un motor gràfic extern, podem trobar un notable estalvi en temps en la codificació del projecte, ja que disposarem d'una sèrie d'eines ja realitzades. Un altre factor important serà la integració amb les altres parts del projecte, depenent de l'heterogeneïtat de tecnologies que s'hagin d'implementar.

El fet d'utilitzar un motor gràfic extern ens facilitaría molt la tasca de codificació del projecte i reduiria el temps de desenvolupament, a més d'una facilitat major a l'hora d'integrar

totes les parts. Aleshores, la tasca serà de saber quines coses es poden fer, aprendre a fer-les i quines coses no es poden fer a causa de les limitacions del motor escollit.

Ens hem centrat en l'estudi de dos motors gràfics, un és *Unity* i un altre és *Unreal Engine*. El motiu d'aquesta decisió és que són molt utilitzats avui dia i estan bastant ben documentats en llibres i per la xarxa.

a) Unity

Figura 5. Logotip pertanyent a Unity

Unity [Uty3] és una eina per al desenvolupament de videojocs. Consta d'un motor gràfic potent, capaç de simular físiques i il·luminació amb una qualitat visual força acceptable. Durant els últims mesos ha tret a la llum títols per a Android i iOS com "*Shadowgun*" o "*Blood & Glory*" on es mostra una molt bona qualitat gràfica en dispositius mòbils.

Unity proporciona una eina per a la programació anomenada *MonoDevelop*, que permet realitzar Scripts en 3 llenguatges diferents: JavaScript, C# i Boo (dialecte de *Python*) i posseeix un editor gràfic que permet crear escenaris, llums, SkyBox [SkyBx] i d'altres funcions facilitant el procés de desenvolupament.

La potència gràfica de *Unity* no és tant elevada com la de *Unreal Engine* i el seu editor tampoc és tan complert. No obstant, és més intuïtiu i hem trobat una gran quantitat de manuals per a principiants i també per a usuaris més experimentats.

La portabilitat no és un problema, ja que *Unity* està preparat per a la multi plataforma i per tant, realitzar el joc per a altres plataformes com Android o iOS, seria un procés possible si es realitza una bona estructuració del projecte.

Els llenguatges de Scripting que utilitza també són similars als que ja hem vist fins ara (JavaScript) o directament ja els hem tocat (C#). Per tant, si parlem del llenguatge a l'hora de programar, *Unity* ofereix un ventall de possibilitats major que la resta.

Un altre característica, és que existeixen moltes aplicacions que actualment treballen amb Kinect a través *Unity*. Per tant tenim a la mà un gran nombre d'exemples i demostracions lliures que podem fer servir per aprendre i enriquir els nostres coneixements.

Potser un dels punts més febles de *Unity* per a la nostra elecció és que la versió gratuïta no permet carregar llibreries externes. Això suposaria una complicació a l'hora d'utilitzar Kinect o al utilitzar altres llibreries externes (físiques per exemple) que volguéssim afegir al projecte.

Per solucionar aquest problema, hem buscat controladors alternatius per Kinect compatibles amb *Unity*. Si utilitzem el SDK oficial de Kinect, necessitaríem la versió Pro de *Unity*, ja que hauria d'accedir a les llibreries externes, així doncs, la millor eina per utilitzar Kinect amb la versió gratuïta de *Unity* que hem trobat és OpenNI , la qual serà explicada més endavant.

b) UDK

Figura 6. Logotip pertanyent a Unreal Development Kit

UDK (*Unreal Developement Kit*) **[UnrDev]** és una altre alternativa a l'hora de parlar de motors gràfics potents. El motor de "*Epic Games*" aposta més per la potència gràfica i no tant per la multi plataforma. Mostres d'aquesta potència són títols de l'envergadura de "Gears of War" o "Unreal Tournament". Aquesta eina ens permet desenvolupar jocs per dues plataformes: Windows i iOS.

UDK també utilitza un llenguatge propi (*UnrealScript*). Aquest però, en comparació als llenguatges de *Unity*, és molt més lent a l'hora d'executar-se. No obstant, l'editor integrat de proporciona *UDK* proporciona moltes més eines per a l'edició.

D'altra banda la integració amb programes externs, el joc Online i altres característiques com la portabilitat són un tant més complicats amb aquest motor. Són aspectes que podrien suposar un gran endarreriment en el desenvolupament del videojoc si triéssim aquesta opció.

Ús d'un Motor Gràfic Propi

L'altra alternativa és la creació d'un motor gràfic propi. Creiem que aquesta opció requereix una càrrega de treball excessiva per complir els objectius del projecte a causa de l'heterogeneïtat de les tasques que realitzarem.

Un aspecte positiu de crear una tecnologia és que pots tenir el control total sobre ella, sense dependre de tercers i saber en tot moment que estàs implementant i en què estàs focalitzant els teus esforços.

És una opció que ens obligaria a invertir massa temps de desenvolupament i difícilment ens podríem centrar en el desenvolupament del joc fins que el motor fos funcional.

Després de veure tots els punts positius i negatius de l'ús d'un motor gràfic extern o propi, ens decantem per l'ús d'un extern, estalviant bastant temps en desenvolupament de tecnologia, cosa que ens permetrà centrar-nos en el desenvolupament del joc en si, amb el cost d'haver d'invertir bastant temps en l'aprenentatge i l'assimilació del motor gràfic extern que utilitzem.

Després d'unes quantes tardes d'estudi i proves, i gràcies als manuals que hem utilitzat per aprendre més sobre *Unity* i *Unreal Engine*, hem arribat a tot un seguit de consideracions per seleccionar la tecnologia que utilitzarem.

Per a això hem elaborat la taula 1 on marquem amb ”+” els punts on un motor és millor a l'altre. Els punts que hem valorat són els següents:

- **Potència gràfica**: Una vegada vistes les demostracions i projectes que circulen per internet podem fer-nos una idea de quina és la potència gràfica capaç de suportar cadascun. Això implica el nombre de models que és capaç de gestionar en pantalla de forma fluïda, el seu nivell de detall, el nombre de partícules i la qualitat de la il·luminació que genera.
- **Editor**: Un cop instal·lats tots dos motors gràfics i realitzant una sèrie de tutorials hem pogut valorar el nivell d'edició de joc que suporta cadascun dels motors, temes com l'edició de nivells, l'edició d'animacions de personatges, l'edició de vídeos, so, etc.
- **Integració amb programes externs**: S'ha valorat quan el motor gràfic és capaç de permetre integració amb llibreries externes i realitzar importacions d'arxius generats d'altres programes.

• **Documentació**: Realitzant cerca per internet hem valorat el suport que té cadascun dels motors gràfics, les comunitats que generen continguts i les webs dedicades a oferir ajuda i tutorials.

• **Llenguatge Scripting**: Consultant la informació de cada motor gràfic hem trobat el llenguatge mitjançant el qual es programa en ells. Es valora la diversitat i la velocitat d'execució.

• **Preu llicencia professional i preu llicencia de publicació:** En la informació oficial de cadascun dels motors gràfics es pot trobar el preu d'us del motor, i del preu per poder publicar les aplicacions que realitzis.

Característiques:	**UNITY**	**UDK**
Qualitat Gràfica (Potència)		**+**
Editor		**+**
Integració amb programes externs	**+**	
Joc en xarxa	**+**	
Varietat de plataformes	**+**	
Programació i Documentació	**+**	
Llenguatge Scripting	**+**	
Preu Llicència Pro	**1500 $**	**2500 $**
Preu Llicència de Publicació	**Llicència Pro**	**99$ + 25% Beneficis (Superiors a 5000$)**

Taula 1. Comparativa entre motors gràfics

L'opció que més s'adequa als nostres requeriments és clarament *Unity*. Els factors de major pes han estat la documentació existent (per *Unity* hem trobat molta més informació, exemples i comunitat que desenvolupa); un altre motiu és que els llenguatges de scripting que utilitza *Unity* són més estàndard que *UnrealScript* (l'utilitzat per *Unreal Engine*) per tant serà un valor afegit que podrem aprofitar en un futur. I per finalitzar el tema de les llicències: si algun dia pensem portar el nostre producte a la venda té un preu molt més accessible que *Unreal Engine*.

Un cop decidit el motor gràfic, per començar a treballar en el projecte, s'han establert una sèrie de manuals [UtyMan] a seguir i la API de *Unity* [UtyApi] on es proporciona tota la informació necessària sobre les eines que proporciona el motor.

Aquest motor gràfic ens permet crear paquets amb objectes, scripts i escenaris. Aquests, faciliten molt la feina a l'hora d'integrar totes les parts que composen el projecte.

1.1.4 Diagrama de mòduls

Una vegada posat en comú el document de concepte de joc s'ha procedit a elaborar el diagrama de mòduls del joc *Launchageddon*. La divisió del treball s'ha realitzat de manera que queden tres mòduls ben diferenciats, amb una càrrega de treball que creiem ben balancejada, on cadascun d'ells serà desenvolupat per un integrant de l'equip.

Com podem veure a la figura 7 els mòduls són:

a) Gameplay [MarSue].
b) Editor [JorCar].
c) Tecnologia.

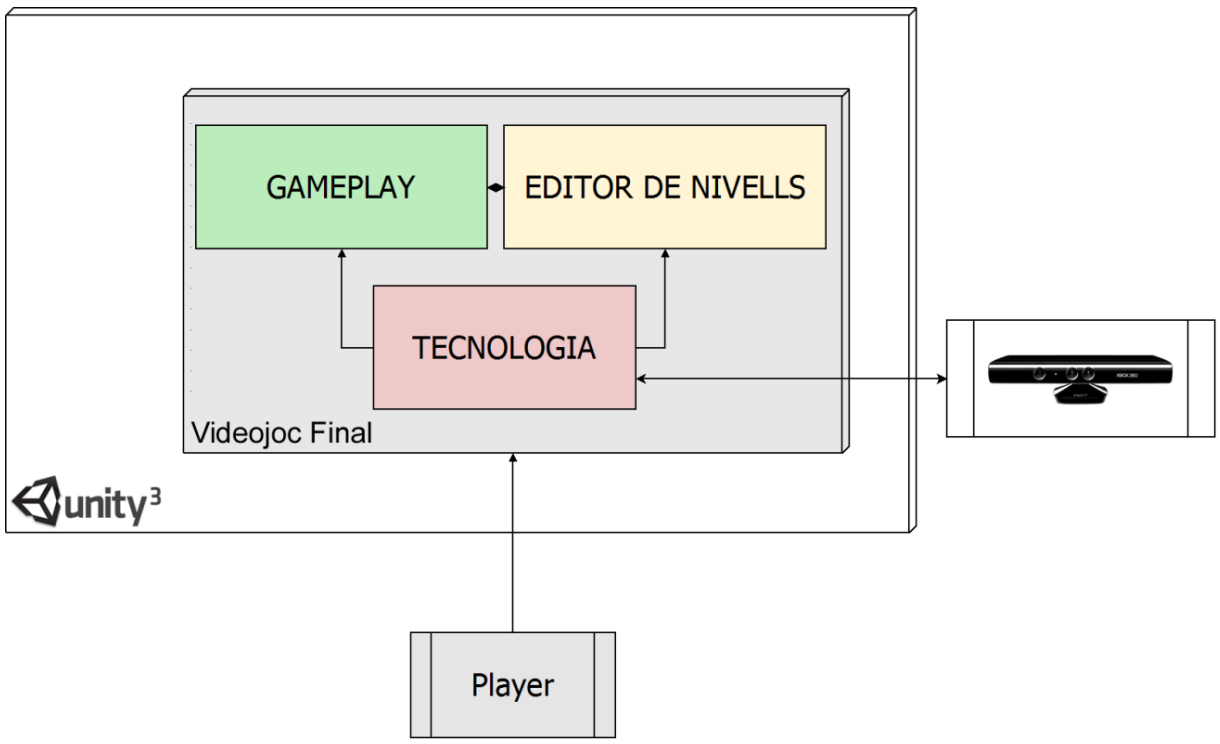

Figura 7. Diagrama de mòduls del projecte comú

a) **El Gameplay** és el mòdul encarregat de gestionar tota la lògica de joc, les físiques, la interfície d'usuari en joc i la interfície de menú principal i els nivells. A més serà l'encarregat de detallar i implementar el disseny de joc, sistemes necessaris per gestionar les puntuacions del joc, accés a arxius de configuració, d'informació sobre nivells i de localització del joc a altres idiomes.

b) El mòdul **Editor** s'encarrega de proporcionar les eines necessàries al jugador per dissenyar els seus propis escenaris personalitzats i exportar-los al mòdul de Gameplay per poder disposar d'aquests en el joc.

c) **La Tecnologia** estableix comunicació amb els altres dos mòduls i els proporciona les eines necessàries per poder treballar amb Kinect. Aquest mòdul requereix una comunicació externa amb el dispositiu Kinect per analitzar les dades físiques del jugador necessaris per als controls.

Per tal de fer la integració entre mòduls haurem de definir com es comunicaran entre ells, per quins motius i quines dades s'intercanviaran.

Comunicacions entre mòduls

Seguidament trobem un resum de les integracions més importants que es duran a terme:

GAMEPLAY: El mòdul encarregat del joc haurà de ser adaptat per a rebre el comandament del joc. Des del mòdul de tecnologia, es rebran les dades en un format estipulat per les dues parts que aquest utilitzarà per realitzar les accions pertinents al joc. També es comunicarà amb l'editor per detallar la forma en que els nivells hauran de ser carregats al joc per que funcionin, entre GAMEPLAY i EDITOR s'arribarà a un acord per l'estructura d'aquests tipus de dades i el seu funcionament.

EDITOR: Aquest mòdul estableix comunicació amb els altres dos. Per un cantó rep suport del mòdul TECNOLOGIA per poder implementar el control amb Kinect. Altrament, es comunica amb el mòdul de Gameplay a la vegada que comparteixen un conjunt de dades per poder accedir tots dos als escenaris personalitzats. En el cas de EDITOR per modificar-los i en el de GAMEPLAY per fer-los servir en el joc.

TECNOLOGIA: Per tal de que la resta de mòduls puguin emprar els controls que s'ofereixen en aquest, s'han dut a terme reunions per realitzar un anàlisi dels requeriments, on s'han estipulat els formats de les dades a enviar (explicats al capítol 2).

1.1.5 Integració del treball realitzat

Cadascun dels projectes és un treball individual. No obstant això, considerem interessant posar factors en comú de manera que els treballs interaccionin entre si com si formessin part d'un projecte global. Ho considerem un valor afegit, ja que comporta la dificultat de coordinar 3 treballs individuals de manera que cadascun no s'allunyi de la seva línia, però alhora comprengui les necessitats dels altres dos. Per poder integrar el treball realitzat entre mòduls hem definit una metodologia de treball. Aquesta s'ha complert mitjançant reunions de caràcter setmanal o quinzenal.

A la figura 8 es mostren les tasques que s'han realitzat durant el temps que han durat les reunions. Normalment el grup s'ha reunit una tarda per setmana.

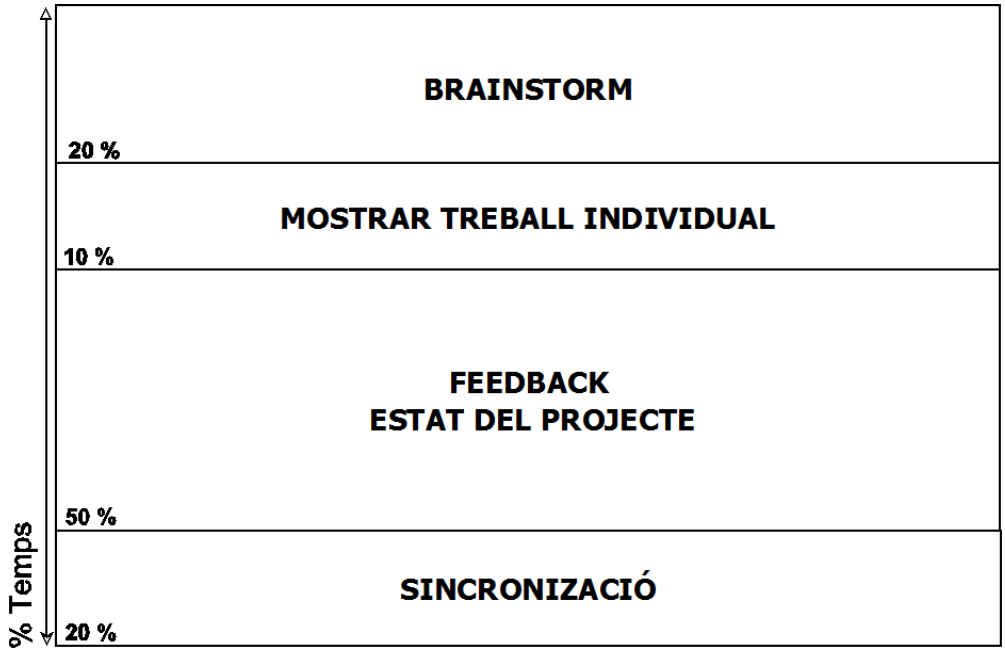

Figura 8. Percentatge de temps orientatiu de les reunions

Com es pot observar les reunions ens ha servit tant per comunicar-nos entre els membres de grup tant per sincronitzar treball, prendre decisions sobre el desenvolupament, valorar la modificació e inclús de característiques noves a l'aplicació, etc.

Durant el *Brainstorm* tots els components proposen les noves idees per al projecte, aquestes s'apunten per un anàlisi. Un cop s'han analitzat totes les propostes, cada un dels components mostra la feina realitzada fins el moment. En aquest moment es realitza un feedback de l'estat del projecte basant-nos en la feina feta fins el moment, i finalment es du a terme la sincronització, on s'estableixen les necessitats per a la següent reunió.

Per a la realització del projecte es van prendre algunes decisions sobre la tecnologia que usaríem, entre elles podem destacar: el projecte serà programat en un llenguatge d'alt nivell orientat a objecte, C#. La seva estructura bàsica estarà formada per classes que interactuaran entre elles utilitzant patrons i missatges. Per a la programació s'utilitzarà l'entorn MonoDevelop amb el compilador Mico, que és un editor que està perfectament integrat amb Unity. Aquest ens permetrà compilar, depurar i executar ràpidament sense trigar massa a sincronitzar els canvis amb *Unity*.

El plantejament inicial és fer un joc de gran envergadura i parts ben diferenciades per a que es puguin realitzar en tres blocs. Cada un d'aquests blocs serà realitzat per un component de l'equip. Posteriorment el treball realitzat es sincronitzarà i s'integrarà per tal de que funcioni conjuntament.

Seguidament proposarem els objectius que volem assolir amb el joc, i a continuació els objectius individuals de cada membre.

1.2 OBJECTIUS DEL PROJECTE

L'objectiu principal d'aquest projecte és proporcionar les eines bàsiques per tal d'utilitzar la tecnologia que ens ofereix Kinect per a interactuar amb el videojoc *Launchageddon*. Per tal d'assolir aquest objectiu principal, he de plantejar els següents objectius:

- Dissenyar els mòduls que composaran aquest projecte.
- Aconseguir un bon reconeixement de la figura del jugador, un seguiment estable que proporcioni dades fiables de la posició del cos.
- Seguir els moviments de les articulacions desitjades per tal de obtenir les dades per els controls de moviment.
- Reconèixer esdeveniments per part del jugador, com fer lliscar la mà, polsar un botó o simplement ajuntar les mans amb la finalitat de realitzar accions al videojoc.
- Gravar els moviments de les articulacions d'un jugador, per després reproduir la animació en el model virtual. Aquesta part serà aplicada per les animacions dels personatges en els moments en que el jugador no interactua amb el joc.
- Substituir el punter del ratolí per un altre controlat amb Kinect. En aquest cas s'haurà d'interactuar amb la interfície d'usuari per polsar botons, lliscar a través dels menús i seleccionar objectes entre d'altres.
- Provar aquesta tecnologia en altres jocs. Dissenyar controls específics per a altres tipus de jocs com per exemple jocs infantils o jocs de tirs.
- Controlar els moviments del jugador en el videojoc, amb una càmera en primera persona que permeti moure's lliurement per l'escenari.

Un cop definits els objectius del projecte s'esmentaran els requeriments previs per dur-lo a terme. El principal requeriment és el dispositiu extern Kinect i a continuació es detalla el seu funcionament.

1.3 KINECT

Després de vint anys de desenvolupament, Microsoft Research (MSR) treu a la llum la tecnologia de Kinect. Aquest va ser anunciat per primer cop el dia 1 de juny de 2009 en la "Electronic Entertainment Expo" amb el nom de *Project Natal*.

El nom en clau *"Project Natal"* fa referència a la tradició de Microsoft en emprar ciutats com a noms dels seus projectes. Aquest aquet cas, es refereixen a la ciutat brasilina Natal, com a homenatge al país d'origen del director de Microsoft, qui va incubar aquest projecte, Alex Kipman. Un dels altres motius, és que Natal té un significat que fa referència al naixement, i reflexa l'opinió de Microsoft en el projecte com: "El naixement de la propera generació d'entreteniment a la llar".

No cal buscar molt per la xarxa per descobrir que l'empresa responsable del disseny de Kinect és PrimeSense **[PrmSns]**, qui curiosament, proporciona els controladors del dispositiu "Open Source" adients per a treballar paral·lelament al Kinect SDK ("Software Developement Kit" oficial, proporcionat per Microsoft).

PrimeSense és una empresa d'Israel, la qual ja s'ha expandit a tot el món. Creada l'any 2005 per Aviad Maizels, Alexander Shpunt, Ophir Sharon, Tamir Berliner i Dima Rais. L'any 2011, aquesta va ser seleccionada per el MIT Technological Review **[TecRev]** com a una de les 50 companyies més innovadores del mon.

Actualment, el desenvolupament de videojocs utilitzant la tecnologia de Kinect s'està explotant a la videoconsola de Microsoft, però no trigarà gaire a utilitzar-se en ordinadors de forma oficial, ja que avui en dia no es desenvolupen videojocs per a l'ordinador compatibles amb Kinect de forma oficial.

El sensor Kinect té un disseny de barra horitzontal, amb una amplada de 28 centímetres, connectat a una base circular articulada per tal de poder abastir verticalment el seu camp de visió. Ha estat dissenyat per a ser col·locat a 1 o 1,6 metres d'alçada. A continuació s'analitzaran els elements que composen aquest dispositiu.

1.3.1 Components

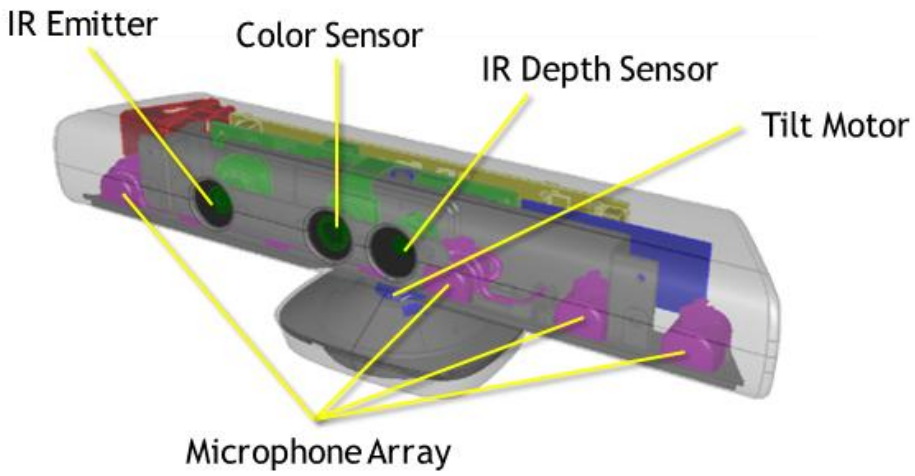

Figura 9. Components bàsics de Kinect

Kinect (figura 9), està principalment composat per:

- Chip de processament d'imatge (PS1080-A2).
- 512 Mb de RAM SDDR2
- Una càmera (COLOR CMOS) amb una resolució de 640x480 RGB 30fps VGA
- Una càmera d'infrarojos (IR CMOS) amb una resolució de 320x240.
- Un projector d'infrarojos.
- Un motor per calibrar verticalment la seva posició.
- Un ventilador per la refrigeració.
- 4 Micròfons (Sampling Rate: 16Hz)
- Un acceleròmetre per calibrar la imatge.

Un cop esmentats els components, en el següent punt s'explicarà quina funció realitza cadascun d'ells i de quina forma Kinect utilitza les dades obtingudes.

1.3.2 Com funciona Kinect?

Abans de l'arribada de la tecnologia de *PrimeSense*, la majoria de controladors de sistemes per gestos es basaven en els mètodes denominats com "time-of-light" que es van caracteritzar per la detecció de la variació de posició d'un objecte físic respecte a un sistema fixe.

Kinect no només es centra en la posició de l'objecte, sinó que també es dedica a detectar i codificar els diferents paràmetres dependents de la llum reflectida per els objectes.

Per tal de conèixer a la distància a la que es troba l'usuari, l'emissor d'infrarojos emet una gran quantitat de punts, formant uns patrons com els que observem a la *figura 10 (b i c)*. Aquests no poden ser captats per l'ull humà, però si per una càmera d'infrarojos. Aquesta detecta tots els punts emesos i Kinect comença a calcular la disparitat per a cada píxel , la diferència entre on es troba el punt al ser projectat i on es deuria trobar en la projecció. Aquesta tecnologia és anomenada Structured-light Scanner (Escàner de llum estructurada).

Figura 10. Visió nocturna de l'emissor d'infrarojos

Combinant les dades de la càmera de llum infraroja amb el sensor monocromàtic CMOS es genera una malla de punts, mitjançant els quals es pot generar una imatge en tres dimensions. Acte seguit, el chip de processament d'imatge descompon la imatge en els paràmetres necessaris per al seu tractament. Es centra en cerca de formes semblants a la humana (cap, braços, cames i pit) i el càlcul de com es podrien moure dins de l'entorn, fixant-se en les zones amb les que pot col·lisionar i els moviments que pot realitzar, a més de tractar d'esbrinar on estaran situades en un instant de temps posterior gràcies a una base de dades d'aproximadament 200 postures comuns de persones. La major part d'aquests càlculs són realitzats pel software dissenyat per Microsoft.

Les càmeres tenen un sistema de detecció que les permet identificar a 6 persones que es trobin en el seu camp de visió, però a l'hora de calcular els paràmetres necessaris per a la interacció està limitat a dues persones.

S'obtenen imatges de la càmera a una resolució de 640x480 píxels i ho realitza a 30fps. Degut a la baixa resolució que ens proporciona, els models no posseeixen una gran definició.

Els 4 micròfons s'encarreguen de localitzar la font acústica i suprimir el soroll d'ambient, permetent així un reconeixement de veu precís sense fer ús d'un micròfon quotidià. La videoconsola Xbox 360 ja implementa controls bàsics per veu tant en la seva interfície com en els jocs, intensificant l'experiència de joc.

El chip de processament d'imatge és el cervell de Kinect, ja que totes les dades dels sensors passen a través d'ell abans de transmetre el mapa de profunditat refinat i la imatge a color a la videoconsola o en aquest cas a l'ordinador.

Tot aquest hardware, necessita uns controladors per aprofitar les seves funcionalitats. El principal problema amb el que ens vàrem trobar, és que *Unity* no ens permet fer ús de llibreries externes amb la seva versió gratuïta. No obstant, vam descobrir *OpenNI*.

1.3.3 OpenNI (Descripció i funcionament)

OpenNI és una organització que promou la compatibilitat i interoperabilitat de dispositius, aplicacions i *middleware* de interacció natural, d'aquí el nom OpenNI "Open Natural Interaction".

Aquesta organització ha creat una llibreria per realitzar el seguiment de l'esquelet (*Skeleton Tracking*), que és inclosa al paquet anomenat NITE (*PrimeSense's Natural Interaction Technology for End-user*), i amb ell no cal utilitzar llibreries externes. NITE és un "middleware" que percep el món en tres dimensions basant-se en les imatges de profunditat proporcionades per Kinect, i trasllada aquestes percepcions en dades significatives de la mateixa manera que ho fem els éssers humans. Aquesta llibreria ens permet realitzar el seguiment del cos de l'usuari aportant les coordenades a l'espai de cadascun dels punts d'interès, així com extremitats i articulacions.

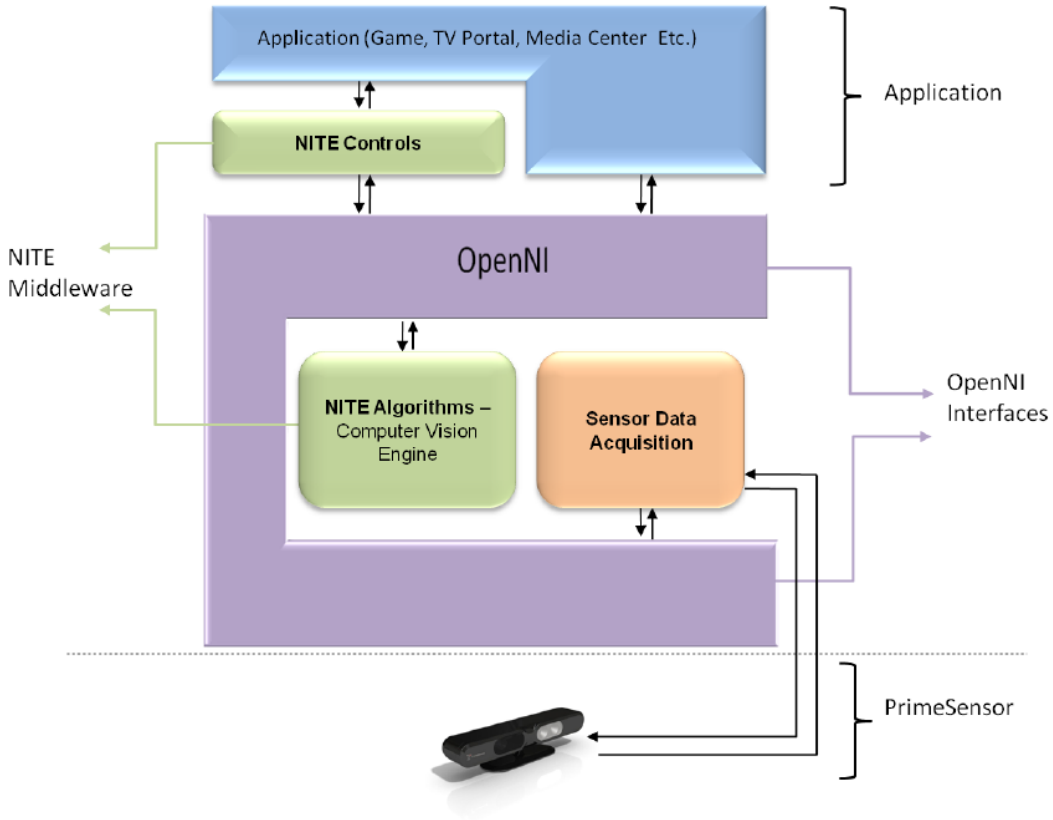

Figura 11. Visió per capes de l'adquisició i processament de la profunditat

Per tant, *OpenNI* permet comunicar-se amb el sensors d'àudio, vídeo i el sensor de profunditat, mentre que proporciona una API que serveix de pont entre el hardware de l'equip, NITE Middleware i les aplicacions i interfícies del sistema operatiu. L'idea principal és facilitar el desenvolupament d'aplicacions que funcionin amb interacció natural, com els gestos i moviments corporals.

La figura 11 ens mostra una visió per capes de la producció, adquisició i processament de les dades de profunditat, fins la capa d'aplicació que ho utilitza per formar un mòdul basat en la interacció natural.

- La capa més baixa és el dispositiu Kinect, el qual s'encarrega de l'adquisició física on s'obtenen les dades sensorials i el conjunt d'imatges en profunditat.
- La següent capa en forma de "C" representa *OpenNI*. Proporciona interfícies de comunicació que interaccionen tant amb els controladors del dispositiu com amb els components *middleware* els quals analitzen les dades del sensor.

- La capa d'adquisició de dades del sensor (Sensor Data Adquisition) és una simple API que permet al host utilitzar el sensor. Aquest mòdul és compatible amb *OpenNI*.
- La capa *NITE Algorithms* és l'intermediari de la visió per computador i també està connectada a *OpenNI*. Es processen les imatges de profunditat generades pel dispositiu.
- La capa *NITE Controls* és una capa aplicativa que estableix el marc d'aplicació per a la identificació dels gestos i els controls basats en gestos de la interfície d'usuari. Aquesta es comunica amb la capa anteriorment explicada a través de les interfícies estàndards i tipus de dades definits per *OpenNI*.
- La capa superior és la *Natural Interaction Based Application*. Aquesta aplicació pot utilitzar els controls NITE i també pot apropar-se directament a *OpenNI* per tal d'accedir a les dades generades per els algoritmes NITE o fins i tot, les dades generades per el sensor.

Tal i com s'indica a l'apartat anterior, les llibreries del paquet NITE ens proporcionen la capacitat de realitzar el "*Skeleton Tracking*". Aquest algoritme es capaç de detectar la figura humana i les seves articulacions tal i com es mostra a la figura 12.

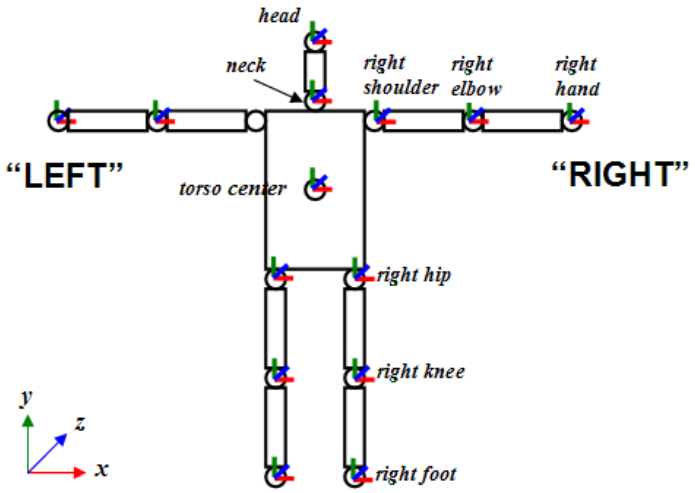

Figura 12. Punts de l'esquelet que reconeix OpenNI

La figura 13 representa el sistema de coordenades i la representació de l'esquelet quan el jugador està de cara a la càmera. Un cop obtingudes les posicions inicials dels components, comença el càlcul de les rotacions i desplaçaments corresponents als moviments del jugador. És important posicionar-se a una distància adequada per obtenir un funcionament correcte. Segons les especificacions, la distància idònia per al reconeixement es troba al voltant dels 2,5 metres.

Al reconèixer l'esquelet, ens retorna la posició i orientació de les articulacions. Les posicions de les articulacions són molt més precises que els angles.

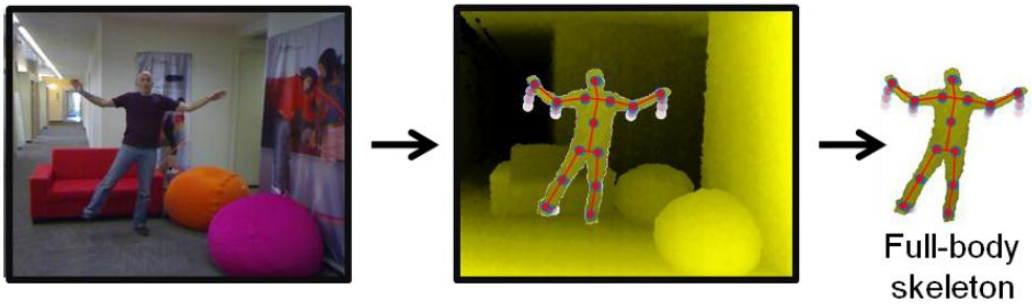

Figura 13. Imatge en profunditat i reconeixement de l'esquelet

Les orientacions de les articulacions es donen com a una matriu de rotació 3x3. Aquesta matriu, representa la rotació entre les coordenades locals de la articulació i les coordenades del món. La primera columna és la direcció de les articulacions del eix X, donades com vector de tres dimensions en el sistema de coordenades del món. La segona columna, és la direcció de les articulacions a l'eix Y, i la tercera la direcció a Z **[MRot]**.

La posició neutral és la posició T, mostrada a la *figura 12*. En aquesta posició, cada una de les orientacions de les articulacions està alineada amb el sistema de coordenades del món. Així doncs, la seva orientació és la matriu d'identitat.

Un problema comú es troba quan alguna de les articulacions no és captada per la càmera o té una mala il·luminació. En aquests moments, entren en acció les heurístiques.

1.3.4 Heurístiques de l'esquelet

Les heurístiques de l'esquelet fan referència a un conjunt d'heurístiques que proporcionen el comportament més plausible a l'esquelet quan les articulacions no tenen fiabilitat. Això succeeix en el moment en que aquestes es perden o s'oculten. Quan una articulació té una posició o orientació indefinida, les heurístiques s'encarreguen d'omplir aquestes dades amb valors raonables.

D'aquesta manera, podem obtenir dades de totes les articulacions a cada moment, sense haver de parar l'execució quan no reconeix bé una extremitat o articulació.

2 DESENVOLUPAMENT

En aquest capítol s'exposarà detalladament el treball realitzat al llarg del projecte. Aquest treball s'ha dividit en mòduls, on cadascun d'ells s'encarrega d'una sèrie de funcionalitats. A continuació es detallarà el procés seguit per al desenvolupament de totes les parts del projecte.

Aquest procés consta de quatre fases principals a través de les quals verificarem el correcte funcionament i agilitzarem el procés d'integració entre els mòduls. Aquestes fases són: anàlisi, disseny, implementació i validació.

La figura 14 mostra una visió global dels mòduls en que es divideix el projecte.

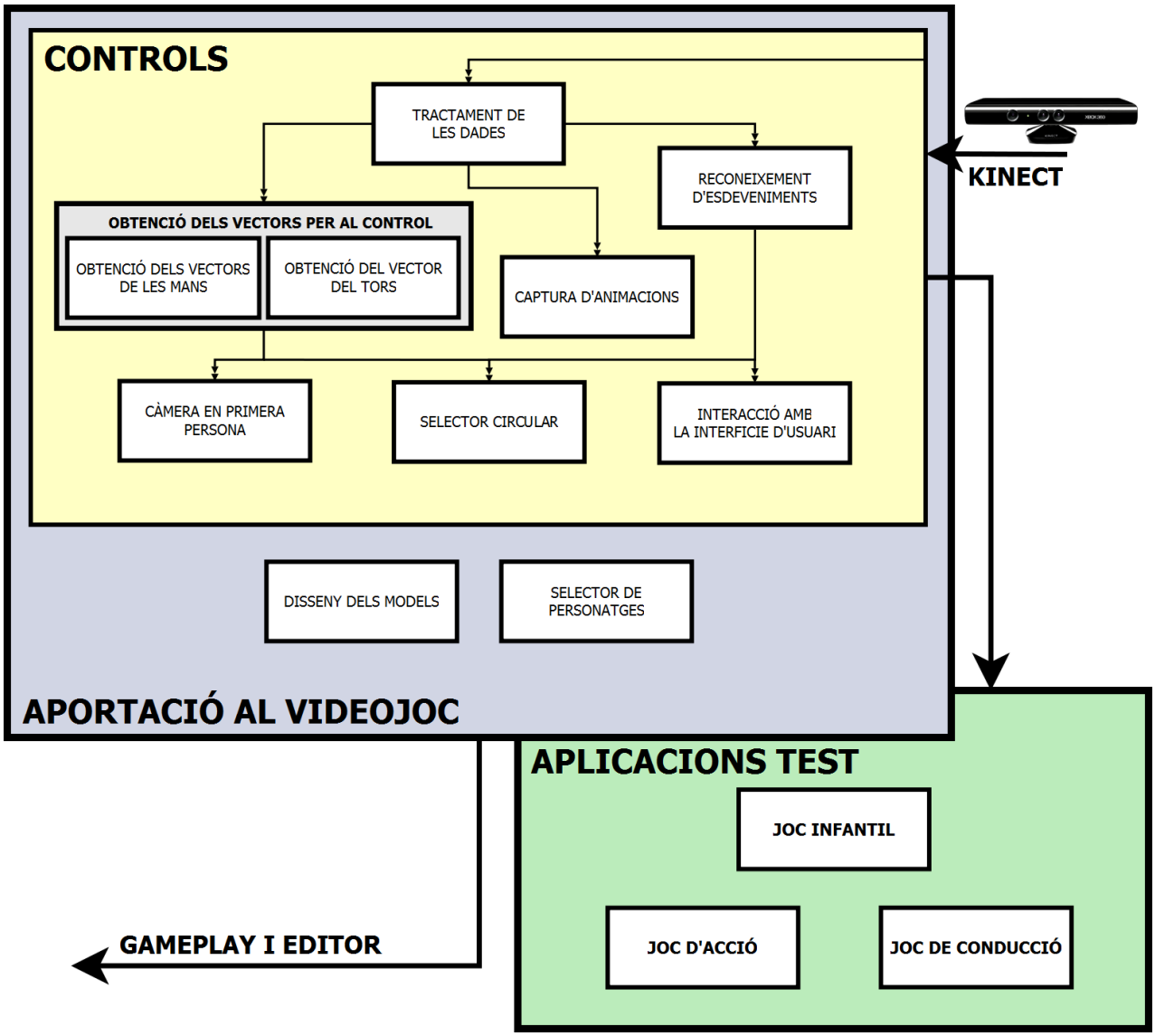

Figura 14. Diagrama de mòduls del projecte

Tal i com podem observar al diagrama, el projecte està dividit en tres grups principals. El primer grup, situat a l'extrem superior, està format per tots els mòduls relacionats amb Kinect que s'utilitzen al videojoc. Aquest utilitza les dades que ens proporciona el dispositiu extern per oferir les funcionalitats que permetin controlar i interactuar amb tots els elements.

Al segon grup, situat a l'esquerra, es troben els mòduls que estan vinculats únicament a "Launchageddon" i que formen part del procés d'integració.

El tercer i últim grup són els mòduls que s'han implementat fora de l'abast del videojoc. Aquests han estat dissenyats amb la finalitat de crear controls per a diferents tipus de jocs, per tal d'investigar el potencial d'aquesta nova tecnologia en diferents gèneres i obtenir conclusions que ens permetin veure si la tecnologia Kinect és un control més o podria arribar a substituir a la resta.

A continuació es realitzarà una breu descripció del funcionament dels mòduls:

Tractament de les dades: A partir de les dades que ens proporciona Kinect s'apliquen les rotacions i translacions corresponents a un model per tal que realitzi els mateixos moviments que nosaltres.

Obtenció dels vectors de les mans: Un cop podem controlar el model, tenim les posicions de les articulacions de l'esquelet i tractem les dades provinents de les mans per establir uns controls bàsics.

Obtenció del vector del tors: De la mateixa manera que es tracten les mans, tractem la posició del tors per obtenir un controlador de moviments en base a la nostra posició.

Reconeixement d'esdeveniments: S'aïllen les dades de les mans, per posteriorment ser capaços de reconèixer moviments del jugador que seran interpretats com a accions durant el joc.

Interacció amb la interfície d'usuari: Es substitueix el punter del sistema per un propi amb la finalitat que l'usuari pugui interactuar amb els elements del videojoc mitjançant els controls creats en el mòdul d'obtenció dels vectors de les mans.

Captura d'animacions: A través del primer mòdul, es defineix una estructura de dades on s'emmagatzema la informació dels moviments realitzats per l'usuari amb la finalitat que el model els reprodueixi en certs moments durant el joc.

Selector Circular: Consistent en una interfície governada pels esdeveniments del jugador, que faciliten l'experiència del joc amb aquest tipus de control.

Càmera en primera persona: Permetrà a l'usuari desplaçar-se arreu de l'escenari podent triar un control clàssic (teclat i ratolí) o un control mitjançant Kinect.

Disseny de models: A partir d'un model amb esquelet descarregat d'internet, s'han generat els personatges principals del videojoc realitzant modificacions, tant a l'estructura com en les textures, a través de 3D Studio Max.

Selector de personatges: Mòdul encarregat de representar per escena el moment en el que l'usuari tria l'equipatge abans de jugar el nivell.

Les següent agrupació de mòduls anomenada "APLICACIONS TEST", té com a finalitat provar la comoditat dels controls amb Kinect en altres tipus de jocs, en aquest cas tres:

KINECT Castle Logix: S'ha desenvolupat una petita demostració d'un joc existent anomenat "Castle Logix" . Un Joc infantil on els nens han d'apilar peces de fusta, formant les construccions indicades a l'inici.

Controls per a un joc de conducció: Controls per simular l'acció del volant amb les dues mans en un videojoc de conducció, com per exemple de cotxes o avions.

Controls per a un joc de guerra: S'han dissenyat controls alternatius per comprovar la seva aplicació en un joc de guerra. Aquest tipus de joc ha estat triat al poder utilitzar les dues mans per apuntar amb una arma.

Un cop presentats tots els mòduls, en els següents punts s'explicarà tot el procés de disseny i desenvolupament que s'ha dut a terme.

2.1 Tractament de les dades

Les dades que ens proporciona Kinect, es basen en les articulacions que es generen a partir de la captura del nostre cos. Aquest mòdul té com a objectiu aplicar tots els nostres moviments a un model 3D. A continuació s'exposen els conceptes necessaris per a la implementació.

El fet de que l'usuari pugui controlar els moviments corporals del model, permetrà que durant el joc, al realitzar el llançament del personatge i quan aquest es trobi a l'aire, adopti les mateixes postures que el jugador intentant assolir l'objectiu prioritari del joc, obtenir la màxima destrucció possible.

Les dades proporcionades per OpenNI s'han d'aplicar a un model 3D per tal que executi els mateixos moviments que els que es capturen a temps real. En el cas d'aquest projecte hem obtingut un model gratuït a través d'internet, amb un esquelet que conté les articulacions necessàries equivalents a les que reconeix Kinect. El format del model és FBX i *Unity* és capaç d'importar aquests i altres formats al projecte com a *GameObject* (classe base per a tots els components d'una escena de Unity).

Els elements amb aquesta classe poden contenir altres GameObject, com en aquest cas, on cadascuna de les articulacions es troba vinculada al model principal de forma jeràrquica. El nostre model posseeix moltes més articulacions de les que necessitem, però això no suposa cap problema degut a que no és necessari utilitzar-les totes. Un cop definits els conceptes d'aquest mòdul podem procedir al seu desenvolupament.

2.1.1 Implementació

El primer pas a dur a terme és vincular cadascuna de les articulacions de l'esquelet del model 3D a les que ens proporciona Kinect. Les articulacions del model 3D que no estan vinculades no cal tractar-les, ja que al realitzar els moviments, aquestes prendran noves posicions respecte a les articulacions actives.

A l'inici, el model es troba en la seva posició inicial per defecte "T" ja que encara no s'ha calibrat a l'usuari. En aquest instant es guarden les rotacions relatives de cada articulació respecte la rotació del model. Necessitem guardar aquests valors ja que s'establiran les noves rotacións de cada articulació, i necessitem mantenir les rotacions relatives per al nostre model.

OpenNI posseeix una classe anomenada "*UserTracker"* encarregada de comprovar en tot moment el nombre d'usuaris que es troben davant del dispositiu. Comprovarem la informació que aquesta ens proporciona fins que ens informi de que ja s'ha reconegut a l'usuari, aleshores comença el procés principal d'aquest mòdul.

A cada cicle d'execució s'actualitza l'esquelet del model en base a l'usuari que ha reconegut Kinect. L'articulació que es pren com a principal és el tors. Aquesta es pren com a referència ja que és el punt més proper al nostre centre de massa. Així doncs, aquesta serà la primera articulació a comprovar a cada iteració. Un cop s'actualitza la posició de l'articulació principal, es comproven una a una, la resta d'articulacions a través d'un mètode on es rep com a primer paràmetre l'articulació a actualitzar i com a segon una estructura de dades anomenada "*SkeletonJointTransformation*" que conté la informació respecte a la posició i orientació de l'articulació.

L'orientació de les articulacions s'actualitza en base a la confiança, una variable proporcionada per *OpenNI* que ens indica fins a quin punt hi ha una certesa de la nova posició. Els rangs utilitzats per aquesta variable van de 0 a 1. Si el valor d'aquesta és major a 0.5, procedim a la actualització de l'articulació. L'estructura de dades anteriorment esmentada ens proporciona la matriu de rotació 3x3 de l'articulació, la qual hem de convertir en el tipus quaternió **[Quat]** ja que és el tipus de dades amb la que treballa *Unity* internament al realitzar les rotacions.

Per modificar la rotació de l'articulació utilitzarem la funció Quaternion.Slerp() que rep com a paràmetre la rotació inicial i la rotació final i s'encarrega de realitzar aquesta rotació en base al temps i no al nombre d'execucions per segon. A l'actualitzar la posició treballarem amb la estructura Vector3, i totes les posicions seran calculades en base a la posició de l'articulació principal, el tors. D'aquesta manera, al modificar la posició de les articulacions de l'esquelet treballarem sempre amb la posició local.

Arribat aquest punt, podem aplicar els nostres moviments sobre un model 3D. No obstant, aquest mòdul només s'aplicarà en certs moments del joc i necessitem obtenir uns controls per poder interactuar amb ell de la mateixa manera que ho fem amb el teclat i el ratolí.

2.2 Obtenció dels vectors per al control

En aquest apartat es desenvoluparan els mòduls d'obtenció dels vectors de les mans i del tors, que tracten les dades que ens proporciona Kinect corresponent a les mans, obtenint finalment un vector normalitzat al que podran accedir la resta de mòduls.

Aquest format intenta ser semblant al que proporciona la resta de dispositius de control com per exemple els “Joysticks Analògics”, per facilitar la feina en el moment d’utilitzar-lo.

Un cop som capaços de reconèixer les parts principals de l’esquelet humà, podem centrar-nos en l’ús de les mans per als controls bàsics en el joc.

Un dels punts forts d’utilitzar Kinect en la interacció és que podem controlar elements d’un món virtual tridimensional a partir de les tres dimensions del nostre. Fins ara, utilitzàvem el teclat, el ratolí o altres dispositius, on cada botó o sensor com a màxim pot d’encarregar-se de dues dimensions.

Per aquest motiu, a l’hora de representar el control de les nostres mans sobre el videojoc necessitarem un Vector3D per poder registrar el desplaçament en cada una de les dimensions.

Unity ens proporciona una classe anomenada Vector3, que té tres components. Cadascun d’aquests és un número en punt flotant encarregat de definir un punt de l’espai en el seu eix coordenades. El primer valor fa referència a l’eix X, el segon al Y i el tercer al Z (figura 15).

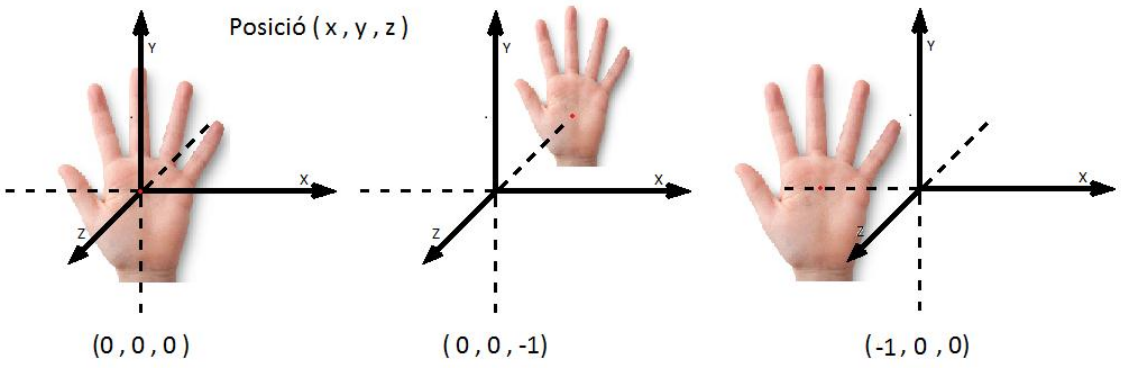

Figura 15. Exemple de la posició de la mà representat amb un Vector3

Inicialment, l’idea principal era poder utilitzar les mans sense la necessitat d’estar capturant a cada instant la resta d’elements del cos. Per fer això no cal reconèixer tot l’esquelet, i per tant, teòricament l’usuari pot interactuar a una menor distància del dispositiu.

La detecció per a una mà és molt ràpida i consumeix molts menys recursos, ja que només tracta el punt de la mà. El focus de la mà, és a dir, el punt inicial en que la mà se suposa que es troba en el seu punt d’origen, es detecta a traves d’un moviment “*swipe*”, que bàsicament es basa en desplaçar la mà d’un costat a l’altre fins que el dispositiu reconeix el punt a seguir.

Un cop tota la feina estava realitzada per a una sola mà, van començar a sorgir els inconvenients. El primer inconvenient va ser reconèixer les dues mans alhora. Al no tenir un esquelet per tal d'ubicar les dues mans, si aquestes s'encreuaven, el reconeixement es tornava imprecís i inestable. El control de la mà dreta passava a pertànyer a la mà esquerre i a l'inrevés depenent de com interpretés l'encreuament de mans.

La solució més coherent que es va trobar, va ser obtenir la posició de les mans a través del reconeixement de l'esquelet. D'aquesta manera, a través de les llibreries proporcionades per *OpenNI*, tenim accés a la posició de cada una de les parts del cos en temps real.

Les dades que ens proporciona l'esquelet han de ser tractades, ja que ens mostren la posició dels elements desitjats en el món.

El primer pas per a utilitzar les mans com a controlador és trobar la millor representació d'aquestes. Per a que les distàncies fossin equitatives, el primer plantejament va ser agafar la posició de les mans en base a un punt de referència: el cap. Utilitzant aquest mètode, teòricament es facilitava el procés de calibratge, ja que la distància entre les mans i el cap no es veia alterada en base a la distància entre la càmera i el jugador, permetent així trobar una escala suficientment bona per a treballar.

Al realitzar les proves, ens vàrem adonar de que al poc temps de mesurar la posició de les mans, aquestes començaven a variar, disminuint cada cop més la precisió inicial. A partir d'aquest error, es van dur a terme un seguit d'alternatives on finalment es va escollir la definitiva, treballar amb l'esquelet sencer.

2.2.1 Implementació

Un cop definides les funcionalitats d'aquest mòdul es procedeix a la implementació del mòdul anomenat obtenció dels vectors de les mans (figura 16). Aquest, s'executarà en mode recursiu mentre el videojoc estigui funcionant i calcularà les dades calculades als mòduls que les necessitin.

Ubiquem l'esquelet generat a partir de la captura de Kinect en el món, i recollim les posicions absolutes dels elements que ens interessen. D'aquesta manera obtenim una posició més fiable. Un cop obtinguda la posició del món, es necessari establir un focus, que serà la

posició a la qual considerem inicial. Per a obtenir la posició relativa al focus, caldrà de restar la posició en aquell instant al punt de focus.

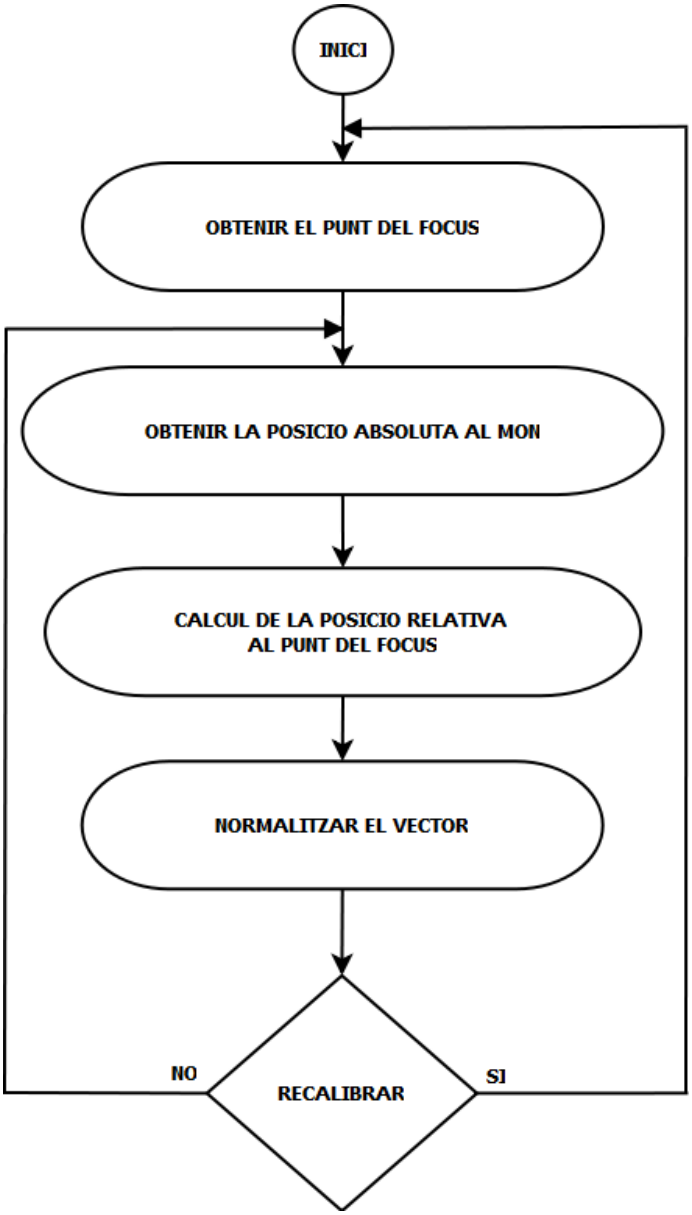

Figura 16. Diagrama de flux per a la obtenció de les dades de les mans

Arribat aquest punt, ja posseïm un vector3D amb la informació de la posició relativa de la mà. Fins ara només podem assegurar que la posició inicial serà un vector amb els seus components a zero, ja que no hi ha una escala establerta per a la resta de posicions. Degut a això, el següent i últim pas és normalitzar aquest vector a uns valors que siguin coneguts i semblants als que ofereixen la resta de dispositius externs. S'ha decidit utilitzar un rang de -1 a 1 ja que per a treballar amb ell és molt més còmode que tingui com a màxim i mínim un valor

unitari. Aquest últim pas, ofereix una major estabilitat en el control, ja que el vector mai superarà els valors definits com a màxim i mínim. El vector resultant, s'utilitzarà en el videojoc per moure o desplaçar un objecte a través d'un escenari en el editor de nivells, o intentar controlar el vol mitjançant la teva posició.

2.2.2 Tractament de les dades del tors

Seguint el mateix procés que ens proporciona els vectors de les mans, s'ha creat un mòdul anomenat obtenció del vector del tors, on obtenim un vector normalitzat amb la posició del tors. Aquest, ha estat dissenyat explícitament per controlar els desplaçaments durant el joc i pren un paper molt important en la càmera en primera persona i el llançament amb el vestit volador.

2.3 Reconeixement d'esdeveniments

En aquest apartat es desenvoluparà el mòdul de reconeixement d'esdeveniments, que serà l'encarregat de reconèixer els desplaçaments que realitza l'usuari amb les mans amb la finalitat de comunicar-ho als mòduls que l'observin.

El problema que es presenta en el moment en que no utilitzem un dispositiu extern com un teclat és que no disposem de cap botó. Situacions tant simples com una pausa en un joc, un canvi de càmera o d'altres, requereixen una prèvia dedicació per tal de definir esdeveniments el més còmodes possibles i intuïtius per a l'usuari final. Sembla un concepte bastant senzill, però no es pot abastar tota la seva complexitat fins que no s'implementa i s'observa la reacció d'una una persona aliena al projecte. Per tal de reemplaçar els botons, hem definit un conjunt d'esdeveniments que permetran la interacció amb el videojoc.

El primer dels esdeveniments a definir va ser la detecció de les mans juntes, ja que el vam trobar fàcil de realitzar per l'usuari, útil i intuïtiu. Aquest és l'únic esdeveniment on hi interactuen les dues mans. Més tard es van definir els cinc esdeveniments bàsics per a cadascuna de les mans:

- Desplaçament a la dreta
- Desplaçament a l'esquerra
- Desplaçament cap amunt
- Desplaçament cap avall
- Polsar (Desplaçament cap endavant)

Cada mà disposarà d'aquests cinc esdeveniments, així doncs, finalment obtenim un total d'onze esdeveniments per poder interactuar amb el videojoc. Els desplaçaments es basen en fer lliscar la mà d'una manera constant en el sentit desitjat. El mòdul implementat s'encarrega de detectar aquest desplaçament, identificar de quin esdeveniment es tracta i finalment comunicar-ho durant un període de temps. L'esdeveniment de polsar, tal i com el seu nom indica, consisteix en realitzar un desplaçament en l'eix Z, és a dir, moure la mà en la direcció a la càmera d'una manera constant com si la nostre intenció fos prémer un botó.

Aquests esdeveniments seran usats per desplaçar-se entre els diferents botons d'una barra d'eines a l'editor de nivells, seleccionar diferents opcions en un menú del joc o simplement afegiran noves funcionalitats de joc com l'activació d'atacs especials.

La posició de cadascuna de les mans s'obtindrà directament de l'esquelet. Treballarem amb les coordenades no normalitzades, ja que per ser capaços de detectar el moment en el que ajuntem les dues mans, necessitem conèixer les seves posicions respecte el mon. D'aquesta manera, quan la posició de les dues mans és la mateixa (amb un cert marge d'error), el mòdul ens avisa del que ha succeït. Una altre raó per no emprar les coordenades normalitzades, és que al sobrepassar els límits establerts, la posició s'estanca en el seu valor màxim, i per tant, no queden enregistrades totes les coordenades per les que ha passat la mà durant el desplaçament.

Les coordenades on es troba la mà al llarg de l'execució s'emmagatzemen en llistes de 10 posicions. Cada mà necessita tres llistes, una per a cada eix de les coordenades. Així doncs, per tractar ambdues mans declarem sis llistes.

Després de moltes proves per comprovar el correcte funcionament del reconeixement, diverses persones van aportar les seves sensacions amb la finalitat d'afinar la sensibilitat d'aquest. A partir d'aquestes aportacions externes, es va decidir que aquests desplaçaments no haurien de tenir una durada superior al mig segon. D'aquesta manera, el cicle d'execució del mòdul, que es l'encarregat d'executar el codi cada *"frame",* va ser limitat a executar-se cada 50 ms. Si tenim en compte que es recollirà una mostra en aquests intervals de temps i que al llarg

del desplaçament es recolliran 10 mostres, estem tractant un esdeveniment amb durada de mig segon.

Una màquina d'estats serà l'encarregada d'indicar quin esdeveniment s'ha produït. Definirem els cinc esdeveniments anteriorment esmentats i hi afegirem l'estat "*IDLE*", que és el estat en el que en trobem per defecte quan no es produeixen esdeveniments i que està associat al concepte de inactivitat. Es crearà una màquina d'estats per a cada mà, així doncs es defineixen "*leftHandState*" i "*rightHandState*".

2.3.1 Implementació

S'implementa el mòdul anomenat reconeixement d'esdeveniments. Aquest, igual que el mòdul anterior, s'executarà durant tot el joc.

Amb les llistes declarades i la màquina d'estats llesta, el primer pas a l'iniciar aquest mòdul és inicialitzar les posicions de les mans a zero i definir que l'estat inicial per ambdues mans sempre serà "*IDLE*". Quan un esdeveniment sigui detectat, canviarà el seu estat segons el desplaçament que es reconegui i pocs segons després, l'estat tornarà a ser l'inicial.

Segons s'ha esmentat en el punt anterior, hem limitat el nombre de cicles d'execució degut a que depenent de la computadora que executa el joc o dels elements que es mostren a l'escena, el nombre de execucions és variable. D'aquesta manera evitem que executi masses vegades el mètode principal, amb la conseqüència d'haver de realitzar el moviment molt més ràpid per tal de que sigui detectat , o que no l'executi les suficients, amb la conseqüència d'haver de realitzar els moviments amb molta lentitud. El nombre d'execucions s'ha establert en 20 per segon, equivalent a la velocitat a la que es realitzaria el reconeixement si s'executés sense estar limitat en un ordinador a 20 fps (Frames Per Second). Aquesta seria una execució massa lenta com per jugar-hi còmodament.

El nombre d'execucions, va acord tant amb la duració del moviment com amb la quantitat d'informació útil que es recull. Quantes més mostres prenguem, més carregarem l'execució del videojoc, afectant així a la velocitat final del videojoc.

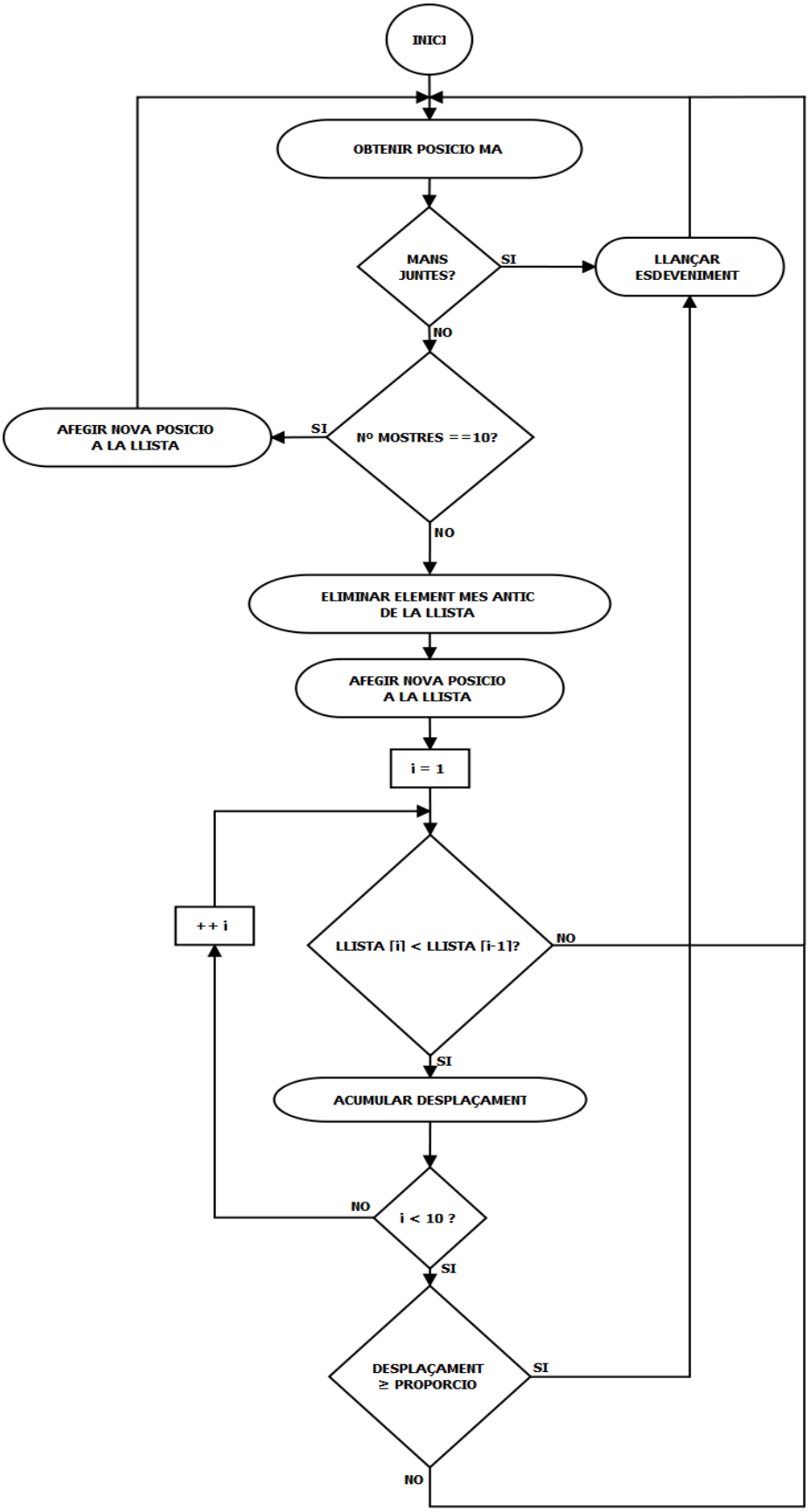

Figura 17. Diagrama de flux del reconeixement d'un desplaçament genèric

Tal i com s'indica al diagrama de flux de la *figura 17*, el primer pas en la comprovació dels esdeveniments és actualitzar la posició de les mans. Seguidament, es comprova si les mans estan juntes a través d'un mètode on es calcula un radi d'acció de cadascuna de les mans i es verifica si aquests dos radis col·lisionen.

Si és així, es llença l'esdeveniment de les mans juntes i es torna a l'inici. En el cas de que les mans no es trobin juntes, comencem a omplir llistes amb les coordenades pertanyents a la posició de les mans. Si la llista encara no conté deu mostres no procedim a l'anàlisi del contingut, sinó que afegim la nova posició i tornem a l'inici. Quan la llista ja és plena, s'elimina la mostra més antiga de la llista pertanyent a la posició zero i s'afegeix la nova mostra obtinguda al final. D'aquesta manera, sempre sabem que els elements que es troben a la llista estan ordenats segons l'instant de la captura.

Arribat aquest punt, entrem en un bucle on s'analitzarà el contingut de cadascuna de les llistes per tal de detectar els desplaçaments realitats per l'usuari. Es recorren les llistes comprovant que cada element compleixi una condició respecte el seu element anterior.

En el cas d'un desplaçament a la dreta, la llista conté l'eix de coordenades X de les posicions, i es comprova que cada element és major a l'anterior i a la inversa si es tracta d'un desplaçament a l'esquerra. Si no es compleix alguna condició, la llista deixa de ser analitzada per tal de no ocupar els recursos disponibles.

Cada cop que es compleix una condició s'acumula el valor absolut de la diferencia entre la posició anterior i la actual, obtenint al final un desplaçament total. Si es compleixen totes les condicions, finalment es comprova que el desplaçament total hagi estat igual o superior a la distància entre el canell i el colze de l'usuari. Si la condició es compleix es llença l'esdeveniment.

2.4 Interacció amb la interfície d'usuari

En aquest apartat es desenvoluparà el mòdul d'interacció amb la interfície d'usuari, encarregat de substituir el punter del sistema per un de propi amb la finalitat de controlar el videojoc.

El punter de sistema és l'encarregat d'interactuar amb tots els elements del joc. Els esdeveniments interns del motor gràfic que es generen quan aquest es situa sobre d'un element

de la interfície gràfica d'usuari estan programats en base a la posició del ratolí. Per tant, si volem que el nostre punter pugui interactuar de la mateixa manera que ho fa el ratolí necessitarem saber si aquest es troba situat sobre elements de la GUI i fer ús dels esdeveniments per simular l'acció dels botons del ratolí.

Per representar el nostre punter s'utilitza un Label, que no és més que una textura pintada sobre l'interfície d'usuari. Els Labels al igual que els botons se'ls indica una posició i una grandària. Per indicar tots aquests valors s'utilitzen els següents quatre paràmetres:

1- Posició X: Aquesta posició pren com a zero el costat superior esquerra de la pantalla (Top-Left).
2- Posició Y: També pren com a zero la posició de l'anterior.
3- Grandària X: Especifica l'ample de la imatge en píxels.
4- Grandària Y: Especifica l'alt de la imatge en píxels.

Un cop especificades les variables, finalment s'indica la textura que volem utilitzar. En aquest cas, s'emprarà un punter semblant al que es proporciona per defecte a Windows, caracteritzat per concordar amb l'aspecte del videojoc. Unity proporciona les eines necessàries per ocultar el punter del sistema, d'aquesta manera el nostre nou punter serà l'únic que es representi a la pantalla (figura 18).

Figura 18. Substitució del punter del sistema per un propi.

Moviment del punter

Ara que disposem del nostre punter, cal implementar el moviment d'aquest amb suavitat i comoditat per l'usuari. La posició inicial d'aquest, serà per defecte el centre de la pantalla per una major comoditat per a l'usuari. A diferència d'un ratolí, la posició de la mà es troba fixe en un punt, és a dir, la posició inicial de la mà serà interpretada com la posició d'origen del punter. Per tant, al ser desplaçada a un altre punt i tornar al centre, el punter també haurà de tornar al centre de la pantalla. El ratolí pot ser aixecat de la superfície on llisca per abastir una superfície més extensa de pantalla, ja que al variar la sensibilitat és necessari molt més espai per abastir la pantalla complerta.

Per implementar aquest moviment, partirem sempre de la posició inicial, i li sumarem la meitat de la grandària de la pantalla multiplicada per la posició de la nostra mà, que té un rang d'entre 1 i -1. D'aquesta manera, si la mà segueix a la seva posició inicial, el punter no es desplaçarà. La formula utilitzada per aquest càlcul és la següent:

*Posició_X = (Ample_Pantalla/2)+((Ample_Pantalla/2)*Desplaçament_Ma_X)*

*Posició_Y = (Alçada_Pantalla/2)+((Alçada_Pantalla/2)*Desplaçament_Ma_Y)*

La velocitat del desplaçament ha de ser la mateixa per a tots els ordinadors, i per tal de que no variï depenent de la potència l'hem de realitzar en base al temps.

Interacció amb els elements del videojoc

La interacció amb elements com el botons és un punt complex ja que *Unity* no proporciona cap mètode per modificar la posició del punter. Això és degut a que el punter que s'utilitza per defecte, és el punter del sistema operatiu, com a la gran majoria d'aplicacions.

Una opció per a resoldre aquest problema és importar funcions de llibreries externes del sistema operatiu, com pot ser "User32.dll" que posseeix una funció que ens permet modificar la posició del punter del sistema, o "System.Windows.Forms.dll" que també proporciona una funció amb aquestes característiques. No obstant, la versió de *Unity* amb la que treballem no ens permet importar llibreries externes, ja que la nostre llicència és gratuïta.

D'aquest mode, un cop descartada la importació de llibreries externes, s'ha buscat una solució alternativa que solucioni el problema i ens proporcioni una experiència semblant a l'ús del punter del sistema operatiu.

2.4.1 Implementació

S'implementa el nou punter del videojoc a través del mòdul d'interacció amb la interfície d'usuari. Aquest mòdul només s'executarà a l'hora d'interactuar amb elements pertanyents a la interfície de l'usuari (GUI). La GUI està formada per tots els elements que es mostren per pantalla i serveixen por donar suport a l'usuari, com els botons, les etiquetes o els menús. Per aquest motiu posseeix mètodes per activar-la i desactivar-la.

El desplaçament del punter es realitzarà a partir del vector pertanyent a la mà dreta que ens proporciona el mòdul obtenció dels vectors de les mans. Aquest vector s'haurà de reduir de tres dimensions fins a les dues de la pantalla, i aplicar els valors obtinguts a les formules del càlcul de posició. Amb aquestes formules ja som capaços de controlar el nou punter arreu de la pantalla. És important controlar la posició del nou punter en tot moment, ja que el del sistema està controlat per no sortir fora de la pantalla. Per aquest motiu es comprova en tot moment que el apuntador es trobi dins dels marges de la pantalla, i si se'n surt, establir una posició màxima i mínima tant en l'eix X com en el Y.

Si volem ser capaços de saber quan ens estem situats sobre d'un element de la GUI, l'única alternativa plausible de la que disposem és utilitzar un "*raycast*" que ens proporciona Unity. La finalitat d'aquest és llençar un raig des d'un punt de les coordenades de pantalla cap al món, que col·lisionarà amb els objectes de l'escenari proporcionant accés a les seves característiques, i per tant, permetent-nos interaccionar amb ells.

El raig serà llençat prenent com a punt d'origen les coordenades del nostre punter i la seva direcció serà un vector que apunta endavant (figura 19). En el nostre cas ens interessen només els elements que pertanyen a la GUI. Per aquest motiu i per tal d'optimitzar els recursos, treballarem amb "*Layers"* amb la finalitat de separar tots els elements que es mostren per pantalla en diferents capes. Així doncs ubicarem tots els botons i elements amb els que interaccionar en una única capa, d'aquesta manera sabrem que només col·lisionarem amb els objectes que ens interessen.

A l'hora d'indicar la posició de partida del raig és necessària fer una conversió de coordenades pantalla (ScreenSpace) a coordenades món (WorldSpace). El seu paràmetre d'entrada és la posició del nostre punter i ell genera el raig a les coordenades món, començant en el pla de la càmera i passant a través de les coordenades X i Y de la pantalla. Les coordenades de pantalla es defineixen en píxels. A la part inferior esquerra es troba el punt (0,0) i a la part superior dreta es troba els valors màxims de les coordenades (Amplada de Pantalla, Alçada de pantalla).

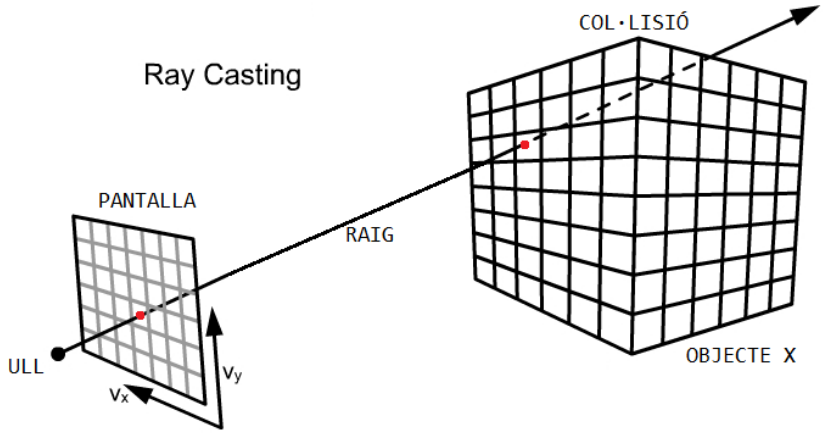

Figura 19. Representació d'un Raycast col·lisionant amb un objecte

Un cop tenim el raig enfocat a direcció desitjada, es llança el *Raycast, indicant-li el tipus de dades on rebrem la informació dels objectes amb els quals hem col·lisionat*. Finalment li indicarem quines capes són les que ens interessen per descartar la resta. Quan el raig detecta l'objecte desitjat, ens proporciona la informació d'aquest, com la seva grandària, la posició, el nom i d'altres. Un cop obtenim la informació, necessitarem fer ús dels esdeveniments per poder interactuar amb els elements, com per exemple els botons.

Unity comprova sempre la posició del punter del sistema respecte els botons amb la finalitat de fer-los destacar quan s'hi passa per sobre. Per simular aquesta acció amb el nostre punter utilitzarem els esdeveniments del motor gràfic per emular que el punter està posicionat sobre d'ell indicant la informació que ha retornat el "*Raycast*". Això farà que el boto sobre el qual estem posicionats destaqui sobre la resta (figura 20).

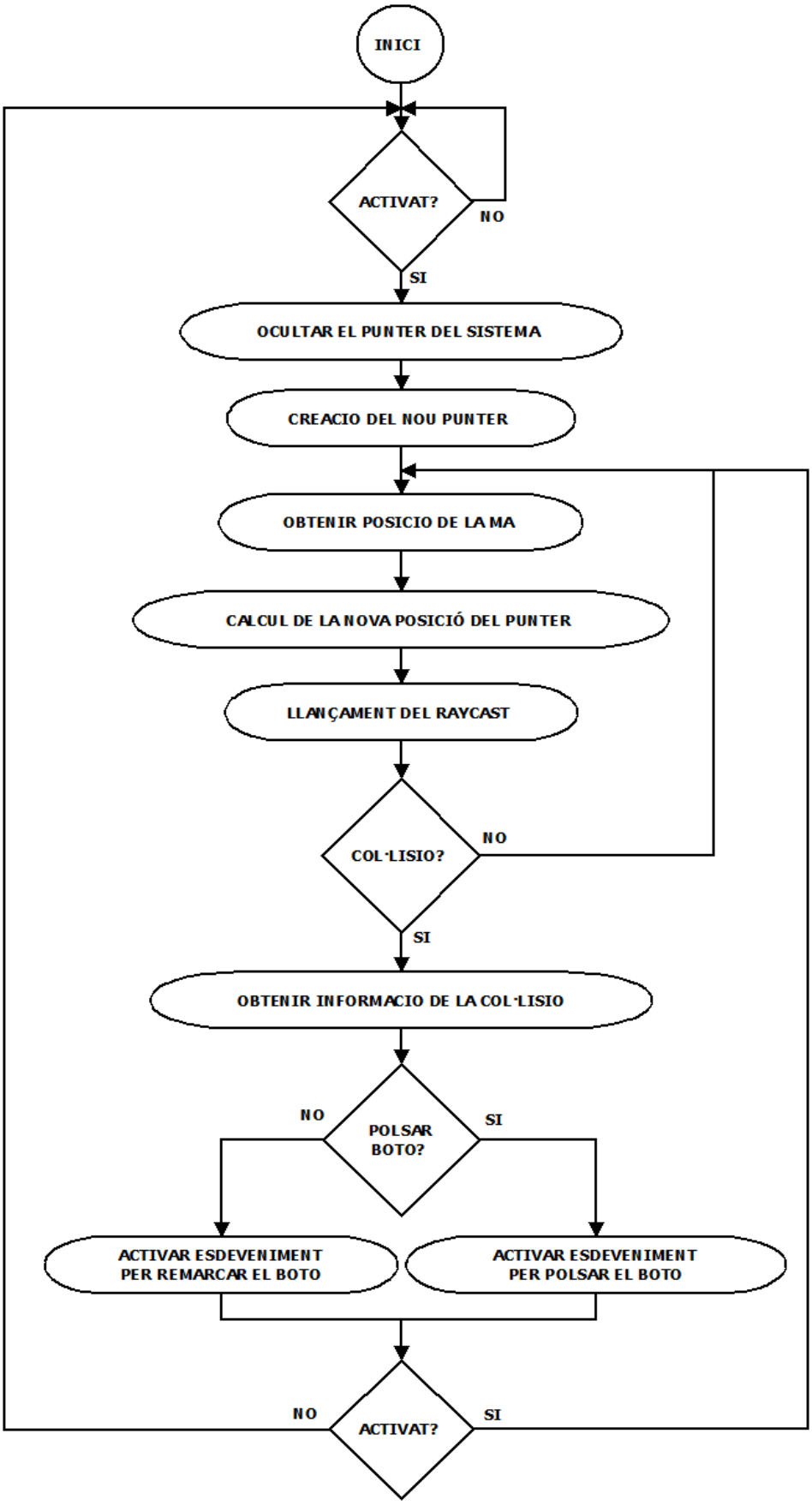

Figura 20. Diagrama de flux del punter controlat amb Kinect

El mòdul de reconeixement d'esdeveniments serà l'encarregat d'indicar-nos quins desplaçaments realitza l'usuari al llarg de l'execució. Segons les proves realitzades per usuaris aliens al projecte s'ha decidit definir dos esdeveniments per realitzar l'acció de polsar un botó. El definit per defecte és l'esdeveniment polsar, que consisteix en fer avançar la mà cap endavant d'una manera constant.

Molts usuaris no el trobaven suficientment còmode i se'ls va plantejar una alternativa. El segon esdeveniment és ajuntar les mans. La mà dreta s'encarrega de desplaçar el punter i quan l'usuari vol polsar un botó, apropa la mà esquerra fins la posició de la dreta.

Degut a la diversitat d'opinions respecte aquest tema, s'ha afegit la possibilitat de canviar el mode de polsar a través de les opcions del videojoc.

Per tal de realitzar l'activació dels botons, s'ha d'implementar una estructura de control del tipus "Switch" on diferenciarem els objectes sobre els que ens posicionem mitjançant el seu nom o etiqueta. Finalment, hem d'afegir una nova condició a la declaració del botó, indicant-hi que aquest podrà ser activat mitjançant el ratolí o una conjunció composta per el nom del botó sobre el qual està situat el punter i un booleà que indica si l'usuari vol realitzar l'acció de polsar-lo.

2.5 Càmera en primera persona

En aquest apartat es desenvoluparà el mòdul encarregat del control de la càmera en primera persona, amb la finalitat de que l'usuari pugui desplaçar-se lliurement arreu d'un escenari.

Una càmera en primera persona emula la visió dels essers humans en el món real. Aquesta permet un desplaçament arreu d'un món virtual sobre els eixos (X,Z) al igual que fem nosaltres al caminar. A més dels desplaçaments, aquest tipus de càmera ens permet ser capaços d'observar tot allò que ens envolta de la mateixa manera que nosaltres girem e inclinem el coll.

La finalitat d'aquesta classe serà oferir a l'usuari final la possibilitat de recórrer l'escenari a jugar (tant si és un nivell predeterminat com un creat per ell mateix) abans de que comenci l'acció. Això permetrà a l'usuari realitzar l'anàlisi dels punts febles, preparar millor l'estratègia a seguir o simplement admirar un escenari concret.

Aquest mòdul ha estat dissenyada per ser compatible tant amb la tecnologia Kinect com amb el teclat i el ratolí. Així doncs, si l'usuari no està convençut dels controls amb el cos, pot optar a realitzar aquesta part del joc amb els dispositius externs clàssics. El control per defecte serà Kinect, i si l'usuari ho desitja podrà canviar-ho al menú de les opcions.

Control amb teclat i ratolí

Començarem explicant la funcionalitat amb el teclat i el ratolí. El teclat, a partir de les quatre tecles preestablertes "W", "S", "A" i "D" podrà controlar els desplaçaments corresponents endavant, enrere, esquerra i dreta tal i com es mostra a la *Figura 21*. El ratolí ens permetrà observar el nostre entorn i girar 360 graus sobre el nosaltres mateixos. L'eix X del ratolí estarà vinculat a la rotació sobre l'eix Y del món, descrivint així una rotació equivalent a mirar en ambdós sentits abans de creuar un carrer. L'eix Y del ratolí estarà vinculat a la rotació sobre l'eix X del món, realitzant una rotació equivalent al moviment del coll per mirar amunt i avall.

Figura 21. Comparació entre l'ús d'un Joystick Analògic, Kinect i teclat

Control amb Kinect

Aquest control es divideix en dos grans blocs. El primer és l'equivalent al control amb teclat i s'encarrega de realitzar els desplaçaments. En comptes de quatre tecles, emprarem la posició del tors normalitzada a través de mòdul d'obtenció del vector del tors. Així doncs, segons la

posició del nostre cos, ens desplaçarem arreu del món virtual utilitzant els eixos (X,Z) del vector obtingut. El comportament serà exactament el mateix que al utilitzar un "Joystick analògic". Si ens trobem a la posició inicial i ens movem un pas endavant la càmera realitzarà un avenç continu fins el moment en que ens tornem a situar en el punt d'inici (0,0). Aquest tipus de control proporciona una major precisió en els moviments. Això és degut a que els valors obtinguts es troben dins d'un rang predeterminat i no prenen un sol valor binari com en el cas del teclat (*Figura 21).*

El segon bloc s'encarregarà de les rotacions de la càmera. De la mateixa manera que ratolí ens proporciona dues coordenades pertanyents la seva posició, amb Kinect obtindrem un vector amb les coordenades corresponents a la posició de la mà dreta. En aquest cas, s'aprofiten les dades que ens proporciona el mòdul d'obtenció dels vectors de les mans. Al desplaçar la nostre mà es realitzaran les rotacions pertinents sobre la càmera.

Quan parlem de les rotacions, utilitzarem com a mesura els angles d'Euler en graus. Aquests constitueixen un conjunt de tres coordenades angulars que serveixen per a especificar l'orientació d'un sistema de referència d'eixos ortogonals, normalment mòbil, respecte a un altre sistema de referència d'eixos ortogonals normalment fixos. Van ser introduïts per Leonhard Euler **[AngEul]** en mecànica del sòlid rígid per a descriure l'orientació d'un sistema de referència solidari amb un sòlid rígid en moviment.

2.5.1 Implementació

S'implementa el mòdul anomenat càmera en primera persona. Aquest mòdul conté una càmera pròpia i aquesta només s'utilitzarà en certs moments del joc, així doncs és necessari proporcionar a la resta de mòduls que l'utilitzaran el control d'aquest per tal d'activar-lo i desactivar-lo.

El control per defecte serà Kinect, però l'usuari tindrà l'oportunitat d'emprar el controlador estàndard en el menú de les opcions.

Al iniciar-se l'execució, el primer pas a realitzar és el de bloquejar i ocultar el punter del ratolí. S'inicialitzen els vectors a utilitzar a zero i s'estableixen la màxima i mínima rotació vertical que s'aplicarà a la càmera. Horitzontalment no hi ha cap mena de limitació, l'usuari podrà donar

voltes sobre si mateix tantes vegades com el desplaçament ho permeti. Verticalment s'ha establert un rang de rotació total de 120 graus, on 60 ens permetran observar la part superior de l'escenari i -60 la part inferior. La sensibilitat també és un punt important que ens permet calibrar les rotacions fins a trobar un valor còmode per a treballar.

El càlcul de la rotació horitzontal es realitza a partir de la rotació actual de la càmera. A aquest valor li sumarem el desplaçament sobre l'eix X obtingut tant amb el ratolí com amb Kinect (depenent del controlador triat), multiplicat per la sensibilitat. En la rotació vertical, obtenim el desplaçament sobre l'eix Y obtingut i el multipliquem per la sensibilitat, i utilitzant la funció *Clamp* utilitzada a la al mòdul d'obtenció dels vectors de les mans, el limitem amb les variables definides a l'inici com a mínim i màxim.

Un cop tenim el valor de les rotacions en els dos eixos, reescrivim el valor de *la rotació local* com a un nou vector composat per les rotacions en l'eix vertical i horitzontal corresponents als eixos (X,Y) i afegint un zero corresponent la rotació sobre l'eix Z.

Al realitzar el càlcul dels desplaçaments, a diferència de les rotacions, s'utilitzen mètodes diferents segons el controlador. Si utilitzem el vector proporcionat per el mòdul d'obtenció del vector del tors, necessitem saber quina és la rotació actual de la càmera per conèixer el vector direcció. En aquest cas, utilitzarem la funció proporcionada per Unity, la qual rep com a paràmetre un vector amb la direcció relativa a la que volem anar, com per exemple endavant (0, 0, 1). La funció ens retornarà un vector direcció aplicat a la càmera en coordenades món, així doncs, un cop obtenim la direcció desitjada, cal modificar la posició local de la nostre càmera a la nova posició, realitzant el desplaçament en base al temps a través de la funció Lerp().

Si utilitzem el teclat per els desplaçaments, no disposem de cap vector i per tant, cal controlar els esdeveniments de les tecles per saber en quina direcció es vol anar. Un cop establertes les condicions per a cadascuna de les tecles realitzarem els càlculs del vector direcció en cada cas. *Unity* ens proporciona vectors direcció estàndards com per exemple *forward* (endavant), *back* (enrere), *left* (esquerra) i *right* (dreta). D'aquesta manera, segons la tecla utilitzarem el vector corresponent a la direcció indicada. El càlcul de la nova posició es realitza del mateix mode que amb Kinect.

2.6 Selector circular

Un selector és una eina que ens permet triar entre diferents opcions, en el cas d'aquest projecte, per exemple, s'utilitza per seleccionar les peces que volem moure a l'editor de nivells.

Kinect és una eina molt còmode per controlar accions al jugar a un videojoc. No obstant, si no es disposa d'una interfície gràfica senzilla, treballar amb botons pot fer-se complicat. Per aquest motiu s'ha dissenyat un selector que ens facilitarà la feina a l'hora de realitzar les seleccions mentre juguem. Molts dels usuaris que han realitzat proves amb el videojoc, han coincidit en la complexitat que suposa utilitzar un punter controlat amb la mà en un cas comú, com per exemple seleccionar les peces que es volen introduir a l'escenari a l'editor de nivells.

Es tracta d'un selector composat per botons adoptant una forma circular. Aquest es trobarà situat a un lateral de la pantalla, mostrant només mitja circumferència (figura 22). Tal i com indica el seu nom, és un selector, per tant sempre hi haurà seleccionada una opció i no caldrà polsar sobre cap boto, només lliscar la circumferència fins arribar a la selecció desitjada.

La idea principal a l'hora de dissenyar un selector que proporcioni comoditat al jugador és eliminar o substituir els elements que afegeixen complexitat. En aquest cas aquests elements són el punter i les pulsacions per activar els botons. Aquests són útils a l'hora de desplaçar-se per els menús, s'utilitzen com un ratolí i quasi tothom està familiaritzat amb l'ús del ratolí.

Aquest selector ha estat dissenyat per situacions on no es vol perdre temps. Sempre hi ha una opció seleccionada, i per triar la resta d'opcions l'usuari utilitzarà els esdeveniments. Lliscant la mà cap amunt o cap avall farem girar aquest selector accedint a la següent opció. Això simplifica molt l'ús del videojoc, posant com a exemple un selector d'armes en un joc d'acció.

El següent apartat es centra en la realització d'aquest selector circular.

2.6.1 Implementació

Els elements que es visualitzaran en el selector, seran botons. Aquests es mostraran per pantalla com una agrupació circular, que girarà en ambdós sentits al canviar les seleccions, depenent de l'esdeveniment utilitzat per fer-ho. La posició del centre d'aquest selector circular es trobarà just al límit d'un lateral de la pantalla, és a dir, la posició sobre l'eix X serà zero, i sobre l'eix Y la meitat de l'alçada de la pantalla (Figura 22).

La base d'aquest selector sobre la qual es situaran els botons serà una textura de forma circular que rotarà amb la mateixa velocitat angular a la que es moguin aquests. Els botons es situaran a sobre, i al realitzar les rotacions tots els elements ho faran prenent com a centre la base del selector.

Figura 22. Selector circular a l'esquerra

El càlcul de les posicions dels botons es realitza a cada frame. La textura s'implementarà mitjançant un pla posicionat davant de la càmera i assignant-li la textura corresponent al cercle amb el fons transparent. Les rotacions es realitzaran indicant els graus desitjats.

Els botons hauran de rotar el mateix nombre de graus respecte el fons. Així doncs, per calcular les noves posicions ho farem a partir del centre d'aquest utilitzant equacions trigonomètriques per descriure una circumferència. La posició X la calcularem multiplicant el radi per el cosinus del nombre de graus en radiants a la que rotarem. Com que la posició del centre de rotació es troba a la coordenada 0 de l'eix X no caldrà sumar cap valor.

La posició Y la calcularem multiplicant el radi de la rotació per el sinus del nombre de graus en radiants a rotar. A aquest valor li sumarem la coordenada Y del centre de rotació, perquè aquesta prengui com a punt de referència la meitat de la pantalla. Les fórmules són les següents:

$$Posició\_X = (Radi\_Circumferència * Cos\ (angle)) – (Ample\_botó/2)$$

$$Posició\_Y = (Radi\_Circumferència * Sin\ (angle)) + Radi\_Circumferència$$

Al crear els botons, s'especifica la posició (X,Y) calculada anteriorment, aconseguint que els botons realitzin una trajectòria circular.

Els controls que s'utilitzen per realitzar les rotacions són els esdeveniments. Els desplaçaments cap amunt i cap avall s'encarregaran de fer rotar el selector, accedint al següent botó o a l'anterior. La mà encarregada de realitzar aquests desplaçaments serà la mà esquerra, i la mà dreta s'ocuparà d'interactuar amb el joc.

La correcta visualització de l'escenari és un factor molt important a l'hora de jugar, i per aquest motiu, per tal d'evitar que el selector no ens permeti veure per complert l'escenari s'han implementat dos nous esdeveniments encarregats d'ocultar-lo. Amb la mateixa mà que controlem la rotació utilitzarem els següents esdeveniments:

1. **Ocultar**: Realitzant un desplaçament cap a l'esquerra
2. **Mostrar**: Realitzant un desplaçament cap a la dreta.

2.7 Captura i reproducció d'animacions

En aquest apartat es desenvoluparà el mòdul encarregat de la captura d'animacions. Durant un període de temps s'emmagatzemaran les posicions que adopta l'usuari per després ser reproduïdes pel personatge del videojoc en certs moments.

Les animacions proporcionen un aspecte més humà als personatges d'un videojoc. Es produeixen quan el personatge realitza accions com caminar, saltar, etc. En el cas del nostre videojoc, degut a que els nostres models posseeixen esquelet i que aquest es controla mitjançant les dades que proporciona Kinect sobre els moviments de l'usuari, es poden emmagatzemar les posicions i rotacions de les articulacions en base al temps d'un mode mes senzill que no pas articulant-los manualment.

La finalitat de capturar animacions en aquest projecte serà proveir a l'usuari d'uns moviments de transició o celebració personalitzats. Que l'usuari pugui triar els moviments que realitza el seu personatge al finalitzar pot millorar la seva experiència de joc ja que s'afegeix el concepte d'implicació.

2.7.1 Implementació

El mòdul de tractament de dades és l'encarregat de modificar les posicions i les rotacions de les articulacions dels personatges en base als moviments realitzats per l'usuari. D'aquest mode, disposem de dues opcions a l'hora de emmagatzemar les dades del model. La primera opció és accedir a les dades de les articulacions que genera el mòdul anteriorment esmentat i emmagatzemar-les emprant l'estructura de dades que utilitza. La segona opció és accedir directament a les dades del model, permetent així una major llibertat i comoditat alhora d'obtenir les dades.

L'opció triada ha estat la segona, degut a que és més còmode accedir a les dades del personatge, degut a que les dades emmagatzemades s'hauran d'aplicar després sobre el mateix personatge.

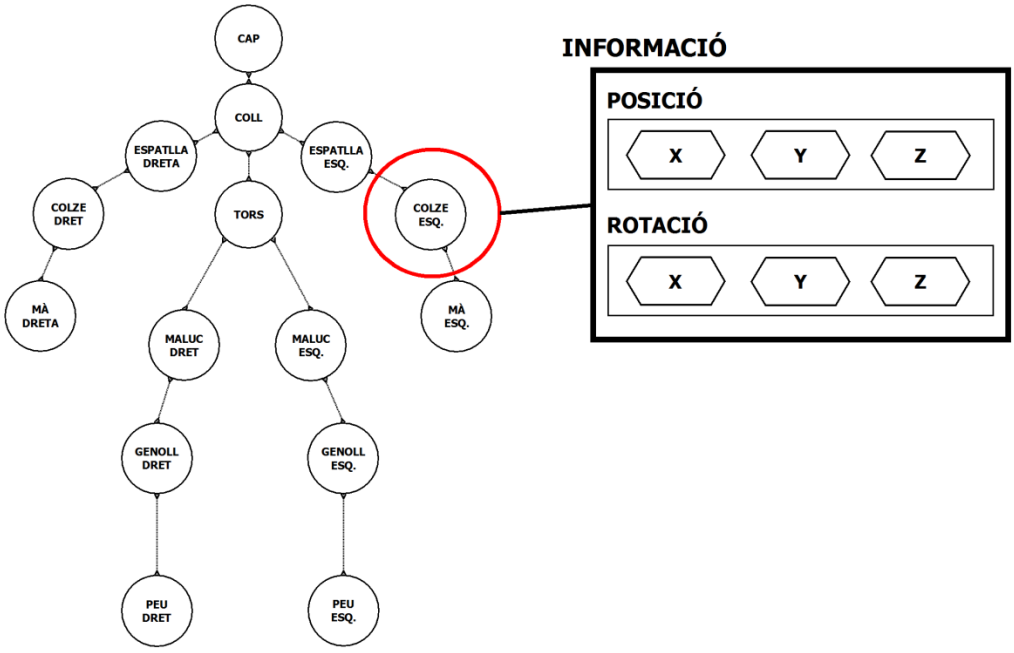

Figura 23. Articulacions de l'esquelet del model

Les posicions i rotacions a emmagatzemar seran les locals respecte al model, ja que es desconeix la seva posició en el moment de la reproducció. Segons s'ha establert, aquestes animacions tindran una duració aproximada d'uns cinc segons, per donar temps a l'usuari a realitzar el seu moviment. Un factor important són les captures que es prenen per segon, ja que si són poques el moviment no serà fluid, i si són masses desaprofitarem espai al emmagatzemar posicions que no donarà temps a utilitzar. Al realitzar un previ estudi, s'ha trobat factible realitzar

trenta captures per segon, ja que aquest és el "frame rate" que utilitzen les videoconsoles d'avui en dia (30fps). D'aquesta manera, al realitzar la captura necessitarem dos vectors per la rotació i la posició locals de les 15 articulacions (figura 23), 30 vegades cada segon durant 5 segons, obtenint un total de 2250 articulacions a emmagatzemar.

Els intervals entre les capturés seran de 33 ms, i així ho indicarem basant-nos en temps d'execució que ens proporciona Unity. Finalitzada la captura obtenim totes les dades de l'animació de l'usuari.

Finalment, en el moment de la reproducció, per tal de que no es realitzin salts entre la posició inicial del model i la primera posició de l'animació cal realitzar una transició fent que aquest desplaçament es realitzi amb continuïtat. Per fer-ho utilitzarem una interpolació lineal que realitzarà aquest desplaçament al llarg d'un nombre determinat de frames i no de cop. La posició inicial s'emmagatzemarà en memòria per tornar a ella al finalitzar la reproducció. Durant l'animació es modificarà directament la posició i rotació de les articulacions, i al finalitzar tornarem a utilitzar Lerp per la transició entre l'ultima posició de la animació i la que era la posició inicial del model.

2.8 Disseny dels models

El disseny és un apartat molt important alhora de realitzar un videojoc, ja que la feina del dissenyador és la que més podrà apreciar a simple vista per l'usuari. Un aspecte atractiu millora molt l'experiència de joc. Degut a que en aquest projecte no comptem amb cap dissenyador s'ha realitzat aquest procés dins de les nostres possibilitats.

S'han dissenyat els personatges principals del videojoc *Launchageddon,* algunes de les peces que s'utilitzen a l'editor de nivells, i les peces utilitzades en l'aplicació de Kinect Castle Logix®.

Això ha suposat un elevat cost de temps, ja que els editors d'objectes 3D són programes complexes on cal tenir una certa base de coneixements per tal d'obtenir uns resultats relativament acceptables. En aquest cas l'editor emprat és 3D Studio Max **[3DMax]**, i per aprendre el seu funcionament és necessària una prèvia formació a través d'un manual **[3DMn]**.

Començarem parlant del disseny dels personatges de Launchageddon. Tots els equipaments dels que disposa el jugador s'han extret d'un model base descarregat d'Internet. El model no s'ha realitzat des de zero degut a la complexitat d'implementar un esquelet amb les característiques que necessitem.

El format original del model és FBX (Abreviació de "Filmbox"). Aquest ens permet guardar informació d'animacions 2D, 3D, audio i video, però en aquest cas només utilitzarem el model 3D sense animar. 3D Studio Max permet importar aquest format i ens mostra informació del model com els seus vèrtex, les cares, l'esquelet, les textures i molts més.

Al realitzar les modificacions sobre model base, només modificarem les posicions dels vèrtex i cares desitjats amb l'objectiu de no modificar cap dels elements pertanyents a l'esquelet de la figura. Per tenir accés a l'edició dels vèrtex, cal seleccionar el model sencer i aplicar-li la utilitat "editable poly" que farà aparèixer els vèrtex del model.

Un dels problemes a l'hora d'editar figures en tres dimensions és que no es pot apreciar amb claredat la profunditat de l'objecte i quan des d'una certa perspectiva modifiquem la posició d'un o diversos vèrtex sembla que es troben en el punt desitjat com succeeix a la figura 24, però al rotar la figura sovint pot observar-se que aquest no es troba situat correctament. Degut a aquest engany provocat per la perspectiva, després de cada nova modificació realitzada sobre la figura cal observar-la des d'altres perspectives per poder certificar la modificació.

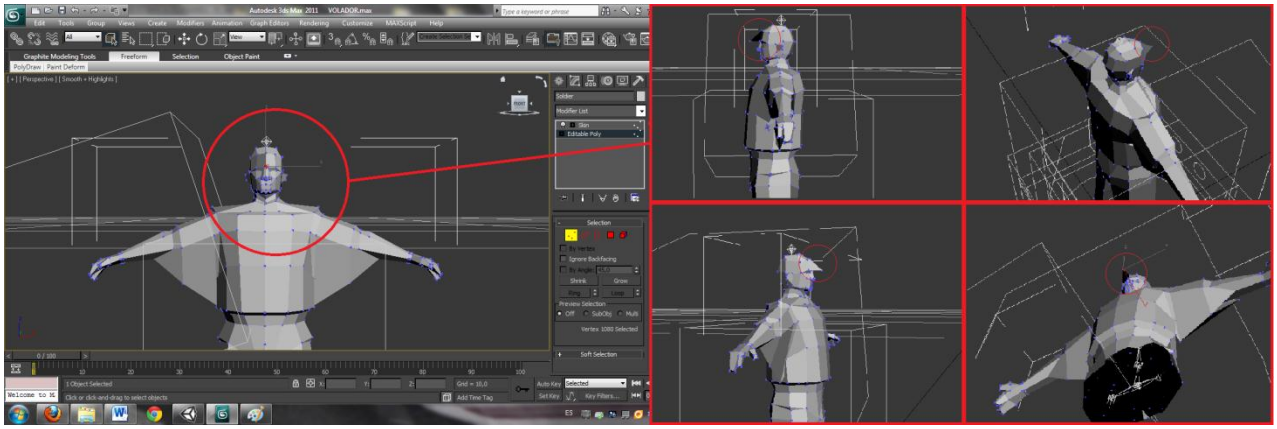

Figura 24. Vista de diferents perspectives un vèrtex mal posicionat

En el cas d'aquest model, els vèrtex superposats no es troben units. En un mateix punt de l'espai on hi ha una unió entre dos o més polígons es troben com a mínim dos vèrtex. Quan se selecciona un d'aquests punts d'unió realitzant-hi un clic amb el ratolí només un dels vèrtex és

seleccionat, aleshores al desplaçar-lo, la resta de vèrtex que es trobaven a les mateixes coordenades que aquest mantenen la seva posició inicial ocasionant una esquerda que permet la visualització de l'interior de la figura.

Una solució possible a aquest problema és seleccionar les unions dels polígons i unir tots els vèrtex que coincideixin a la mateixa posició a través del menú avançat dels vèrtex. Aquesta pot arribar a ser una tasca que requereixi massa temps ja que no es pot seleccionar tots els vèrtex de cop, per aquest motiu, una alternativa és deixar-los tal i com estan i en el moment de realitzar la selecció marcar una zona que contingui aquesta unió pitjant el botó del ratolí i arrossegant-lo fins que l'àrea de selecció contingui la unió desitjada. Així doncs, tots els vèrtex situats en aquesta àrea quedaran seleccionats.

Els polígons també són editables. Al seleccionar-ne un i modificar la seva posició, els vèrtexs pertanyents als extrems dels polígons que es troben en contacte amb aquest, de la mateixa manera que succeeix amb l'edició dels vèrtex es separen i permeten observar l'interior de la figura. Una altre modificació possible és la rotació, ja que al disposar de diversos vèrtex es genera un pla. Un cop les modificacions són satisfactòries, el model ha de ser guardat en un format propi de 3D Studio Max per ser finalment exportat al seu format original FBX.

L'últim pas abans de introduir el model a *Unity* és necessari aplicar-li textures per tal de simular la vestimenta, els detalls facials i la resta d'elements per obtenir un aspecte humà. El model base posseeix una única textura, la qual s'aplica en forma de embolcall recobrint per complert totes les cares de l'objecte. El format d'aquesta textura és PSD (Photoshop Document) que com el seu nom indica és un format d'un editor de imatges anomenat Photoshop **[PhoShp]**.

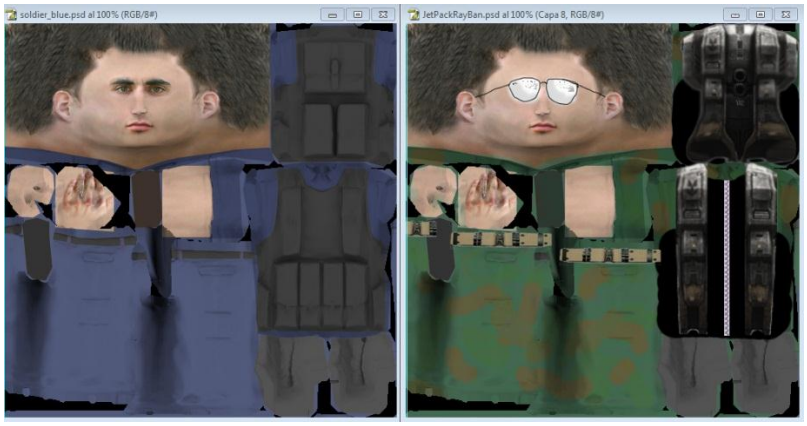

Figura 25. Textura original (esquerra), Textura modificada (dreta)

El model conté la informació corresponent a la textura, d'aquest mode cada polígon que el composa està associat a una zona de la imatge. Aquest tipus de texturització facilita la feina i són molt còmodes per a treballar ja que només cal editar una sola imatge. La figura 25 mostra la textura original del model i una modificada pertanyent a un dels personatges del videojoc.

Els cascs dels personatges s'han realitzat apart a partir de models gratuïts d'internet (Figura 26). De la mateixa manera que amb el disseny dels personatges, s'han utilitzat dos models de casc que estableixen la base i s'han afegit elements nous com dinamita o una destral per aconseguir els models dels dissenys inicials (Punt 1.2).

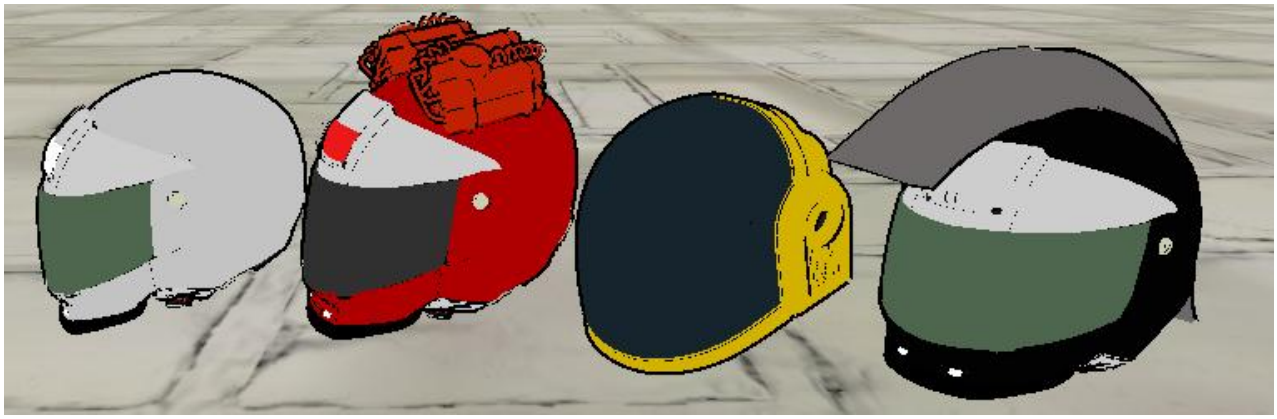

Figura 26. Cascs (Normal – Explosiu – Científic - Destral)

Els models dels cascs han estat dissenyats apart degut a que el jugador podrà combinar els quatre tipus de llançaments amb els quatre cascs. Així doncs, les 16 possibles combinacions s'han realitzat amb *Unity*, creant *GameObjects* vinculant els cascs al model del jugador i prenent com a referència la posició del cap dels models

Figura 27. Llançaments (Normal – Propulsat – Paracaigudista - Antigravetat)

A tots els models se'ls hi ha aplicat una renderització "*Toon Shading*" que dibuixa les figures amb un estil de còmic remarcant les línies exteriors.

Durant el joc, l'usuari podrà crear nous escenaris mitjançant l'editor de nivells. Els objectes disponibles per a la construcció tenen una mida unitària i s'han realitzat a partir de models bàsics com el cub, aplicant després les seves corresponents textures.

Per la realització de les rampes s'ha creat un cub amb els models basics de 3D Studio Max, prenent com a origen una de les arestes inferiors paral·leles a l'eix X, des d'aquesta aresta es realitza un tall diagonal que travessa el cub fins arribar a la seva aresta oposada de la part superior. Un cop realitzat el tall, la part superior restant és eliminada i el forat resultant es cobreix amb un nou polígon que s'uneix als vèrtex resultants del tall. Les rampes amb diferents pendents han estat realitzades seguint el mateix procés però partint d'un paral·lelepípede rectangular.

Kinect Castle Logix®

Com a exemple de l'aplicació de Kinect en altres tipus de jocs s'ha desenvolupat una demostració d'un joc anomenat "Castle Logix®" (explicat al punt 2.10.1). Es composa de tres torres i quatre peces de fusta amb diferents perforacions horitzontals i transversals, on l'objectiu és agrupar-les totes i construir un castell de la mateixa manera que se'ns mostra en una imatge. Aquest joc és un físic, i per tant, el disseny de les peces s'ha realitzat des de zero.

El disseny de les torres és basic. Es parteix d'un cilindre d'alçada unitària, a la part superior d'aquest s'hi fixa un con amb un radi superior al del cilindre, i a la punta del con s'hi fixa una petita esfera. Aquest disseny és realitzat basant-se en les peces físiques.

Hi ha tres torres, i la diferència entre elles és l'alçada dels cilindres. El cilindre més petit és d'una unitat, el mitjà és de dues i el gran de tres. Les textures d'aquestes peces són llises i en tres colors: la base és color fusta, el con color blau cel i l'esfera de color groc.

La part complicada és el disseny dels paral·lelepípedes. Aquestes quatre peces s'inicien amb un cub i tres paral·lelepípedes. L'alçada i la profunditat són unitàries per a totes les peces, i la amplada canvia segons el color. La peça vermella té una amplada d'una unitat, la verda i la groga de dues unitats i per últim la blava de tres.

Tal i com s'observa a la figura 28 on es veu la implementació final de les peces, aquestes tenen perforacions cilíndriques verticals i horitzontals. El radi d'aquestes perforacions ha de ser inferior al dels cilindres ja que aquests les han de travessar.

El primer pas a seguir per a realitzar les perforacions consisteix en marcar una circumferència que ens servirà de guia al realitzar el tall. Per fer-ho, és precís marcar el centre d'aquesta basant-nos en els vèrtex que composen els extrems del polígon. Seguidament s'estableix el radi de la circumferència i marquem totes les arestes que la composen.

El següent pas és eliminar la part central de l'agrupació d'arestes. Un cop realitzada la perforació de la cara desitjada apliquem la simetria per realitzar la mateixa acció a la cara oposada, obtenint així un forat que travessa la peça. L'interior de la figura és negre, degut a que l'interior de les cares no té cap vector normal que reflecteixi la llum. Per obtenir un efecte visual igual a la peça s'ha de crear un nou cilindre amb el mateix radi que la perforació i invertir les seves normals per tal de que la par interior sigui visible. L'alçada d'aquest cilindre serà la mateixa que la profunditat de la perforació.

Aquest procés es repeteix per a totes les peces. La peça vermella té una sola perforació vertical, la verda en té dues, la groga té les mateixes que la verda més una perforació horitzontal que es comunica amb les altres dues i la blava en té tres de verticals i una horitzontal.

Les peces vermella i verda ja s'han finalitzat, però a la resta, les perforacions es comuniquen i els polígons que formen els cilindres es superposen, per tant, cal eliminar aquests polígons restants.

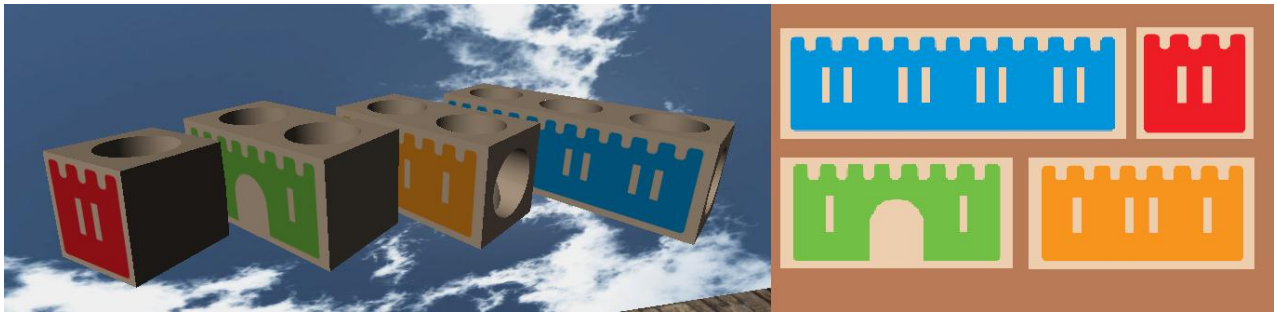

Figura 28. CASLTE LOGIX – Disseny final de les peces i les textures

El darrer pas consisteix en eliminar les parts dels cilindres que es comuniquen, i per fer-ho necessitarem utilitzar el selector de cares. D'una en una se seleccionen les cares que obstrueixen

els forats i s'eliminen. Aquest és un pas molt important ja que si alguna cara no s'elimina correctament, a l'introduir una torre dins del forat col·lisionaria amb ella i no es podria travessar la figura.

Un cop l'estructura de les figures, s'apliquen les textures del joc sobre les cares desitjades. La cara frontal conté la textura d'una torre horitzontal (figura 28) i la cara oposada a aquesta la d'una torre vertical.

2.9 Selector de personatges

En aquest apartat es desenvoluparà el mòdul de selecció de personatge. La finalitat d'aquest és oferir una pantalla de selecció on l'usuari pugui triar el casc i el vestit desitjat per superar el nivell.

Aquest mòdul s'executarà abans de jugar un nivell. L'aspecte visual és un factor important ja que és el que veurà l'usuari abans de jugar. Perquè l'usuari pugui observar els dissenys dels personatges, aquests es visualitzaran amb una rotació constant sobre el seu eix Y. Amb la finalitat de limitar el camp visual de la càmera al jugador, s'afegeixen dos plans a l'escena que ocultin els laterals de la càmera.

2.9.1 Implementació

Per a la selecció es necessitaran 8 botons, 4 corresponents als cascs i 4 als vestits. Tal i com s'ha esmentat en el punt anterior, es posicionaran dos plans perpendiculars a la càmera amb la finalitat de reduir el seu camp de visió (figura 29). Aquests plans s'aprofiten per situar-hi a sobre els botons de la selecció. Els botons de selecció dels cascs es situaran al costat esquerra i els dels vestits a la dreta. Al posicionar el punter sobre del botó de selecció, aquest mostrarà unes imatges explicant la funcionalitat d'aquest. Això s'aconsegueix gràcies al mètode OnFocus que en tot moment comprova la posició del punter respecte els botons de l'escena.

Degut a que hi ha 16 combinacions possibles, s'ha dissenyat una estructura de control per realitzar la selecció. El mètode més ràpid per fer-ho és assignar un valor a cada botó i sumar els dos valors obtinguts. La selecció inicial del selector sempre serà el casc normal i la vestimenta normal. Així doncs, els valors de les vestimentes seran múltiples de 4 començant per zero (0,4,8,12), i els valors corresponents als cascs aniran des de 1 fins a 4. Un cop definits els valors,

per saber la selecció que ha realitzat l'usuari només caldrà sumar els dos valors seleccionats obtenint finalment un valor amb un rang entre 1 i 16.

Figura 29. Menú de selecció del personatge

Com ja sabem que la configuració per defecte serà el model 1, aquest ja es troba situat davant la càmera a l'inici de l'execució. A aquest li aplicarem una rotació constant sobre si mateix en l'eix Y a través del mètode Rotate que ens proporciona *Unity*, i per que la velocitat de rotació sigui sempre constant independentment de l'ordinador on s'executi, es multiplica el nombre de graus a girar per Time.deltatime, que farà que l'objecte roti els graus especificats al llarg d'un segon. En el moment que l'usuari realitza una selecció diferent a la que hi ha per defecte, s'emmagatzema la posició i rotació actual del model. S'elimina el model que s'estava mostrant per pantalla i es carrega el model a través de la selecció de l'usuari. A aquest model se li aplicarà la posició i la rotació de l'anterior per tal de no realitzar una transició brusca.

2.10 Aplicació de Kinect en altres tipus de videojoc

Al llarg d'aquest projecte s'ha pogut observar com el control de Launchageddon mitjançant Kinect millora l'experiència de joc per a l'usuari, trobant-se una mica més immers als escenaris gràcies a que les accions físiques que cal dur a terme aporten una mica més de realitat. En el cas d'aquest tipus de videojoc, el control corporal proporciona les mateixes funcionalitats que la resta de dispositius que hi ha al mercat, però si volem realitzar un anàlisi de major abast amb la finalitat de descobrir si aquest dispositiu podria ser el substitut de la resta, cal implementar controls per altres tipus de joc i avaluar els resultats obtinguts. Així doncs, s'implementaran tres demostracions on es pugui observar la reacció dels controls.

S'han triat tres gèneres a desenvolupar on es posaran a prova les funcionalitats de Kinect: un joc infantil, un joc de conducció i un joc d'acció.

2.10.1 Joc Infantil: "Kinect Castle Logix®"

Castle Logix® [CstLog] és un joc d'enginy infantil dissenyat amb la finalitat d'augmentar la capacitat lògica i desenvolupar les habilitats espacials. L'objectiu principal és encaixar els blocs i les torres de fusta per construir el castell que mostra una fitxa triada (figura 30).

Els controls que utilitzarem en aquest joc seran únicament les mans. La mà esquerra serà l'encarregada de seleccionar la peça que volem triar, indicar quan volem agafar-la per modificar la seva posició i quan volem desar-la, i la mà dreta s'encarregarà del desplaçament sobre els tres eixos de les peces.

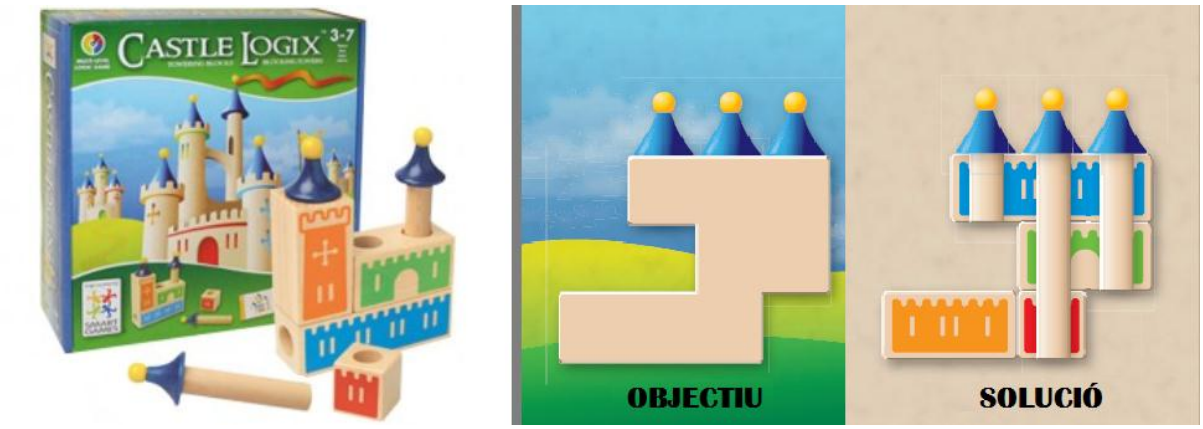

Figura 30. CASLTE LOGIX (Joc Original) – Fitxa de reptes

Aquesta petita demostració constarà d'un sol nivell on se'ns proposarà una agrupació concreta i nosaltres l'haurem de dur a terme. Els procediments per al disseny de les peces es troben a l'apartat de disseny.

Implementació

L'escenari es composa per un terreny que realitzarà la funció de base on es durà a terme la construcció i tres plans posicionats perpendicularment, dos al extrems laterals i un altre a l'extrem més llunyà obtenint una estructura en forma d'escenari.

La càmera es posiciona sobre el terreny, visualitzant així un pla frontalment i els altres dos als laterals. Al pla frontal se li aplica una textura de pedra, i als laterals una que simula el cel, d'aquesta manera, el camp de visió de la càmera només capta els tres plans i el terreny.

S'importen a *Unity* les figures dissenyades, aquestes es situaran en tot moment al fons de l'escenari fins que no siguin seleccionats. El selector de les peces serà el que s'ha dissenyat anteriorment (Selector Circular) i ens permetrà seleccionar amb la mà esquerra la peça que vulguem moure. Els esdeveniments que modificaran aquesta selecció seran el desplaçament cap amunt per la peça següent i el desplaçament cap avall per la peça anterior.

La peça seleccionada s'il·luminarà i no es començarà a moure fins que l'usuari no ho indiqui amb l'esdeveniment polsar. Aquest canviarà l'estat de la peça a mòbil, permetent modificar la seva posició. Els dos estats de la peça emulen l'acció d'agafar una peça o deixar-la. En aquest moment la mà dreta controlarà la posició de la peça i ens proporciona un vector a través de la posició de la nostra mà. La sensibilitat d'aquests desplaçaments es veurà disminuïda, obtenint així uns desplaçaments més lents, ja que per aquesta demostració és necessària una major precisió que no pas amb el control d'un punter.

L'esdeveniment polsar tornarà a ser el detonant per canviar l'estat de la peça a immòbil, quedant aquesta fixa a la seva posició. La única condició per deixar anar una peça és que aquesta estigui col·lisionant o bé amb el terra o amb una altre peça.

Per aquesta demostració s'ha desenvolupat un mòdul que proporciona a les figures un efecte magnètic entre elles. La decisió d'implementar aquest mòdul recau en que les torres han de travessar les perforacions circulars, i a causa de la perspectiva, és molt difícil alinear perfectament dues figures i encaixar els dos orificis cilíndrics per passar-hi una torre a través d'ells. Gràcies a aquest mòdul, quan es detecta una col·lisió entre dues figures les alinea exactament en la posició més propera. A continuació es descriu la implementació d'aquesta ajuda al jugador.

El magnetisme s'executa sempre des de la figura unitària, la figura seleccionada se situa correctament modificant les dues coordenades restants. Els paral·lelepípedes realitzen el magnetisme quan la peça es posiciona a la part superior o als laterals d'una altra, en canvi, les torres, només a la part superior per tal d'encaixar amb l'orifici.

Al detectar una col·lisió amb una altre figura, s'accedeix al seu nom per identificar-la. A través de la direcció del vector normal que ens retorna la col·lisió podem saber amb quin costat de l'altre figura estem en contacte. Si la figura no és una torre, coneixent la seva grandària i tenint en compte que les figures són unitàries podem conèixer la zona d'encaix.

Si posicionem la nostra figura sobre una altra (figura 31), en el moment en el que es produeix el contacte, la posició de la nostre figura sobre l'eix Y és la correcta, aleshores es modifiquen les coordenades X i Z en el moment de la col·lisió per tal que les peces encaixin perfectament. Si desplacem la figura després del magnetisme, aquesta tornarà a quedar lliure fins el moment en que es produeixi la següent col·lisió.

En una col·lisió com la de la figura 31, els valors que prenen els eixos X i Z de la peça són els de la zona d'encaix més propera. Si la peça vermella col·lisionés a la zona central de la peça blava, l'efecte magnètic la posicionaria sobre el següent orifici, i si ho fes a l'extrem dret encaixaria amb l'últim.

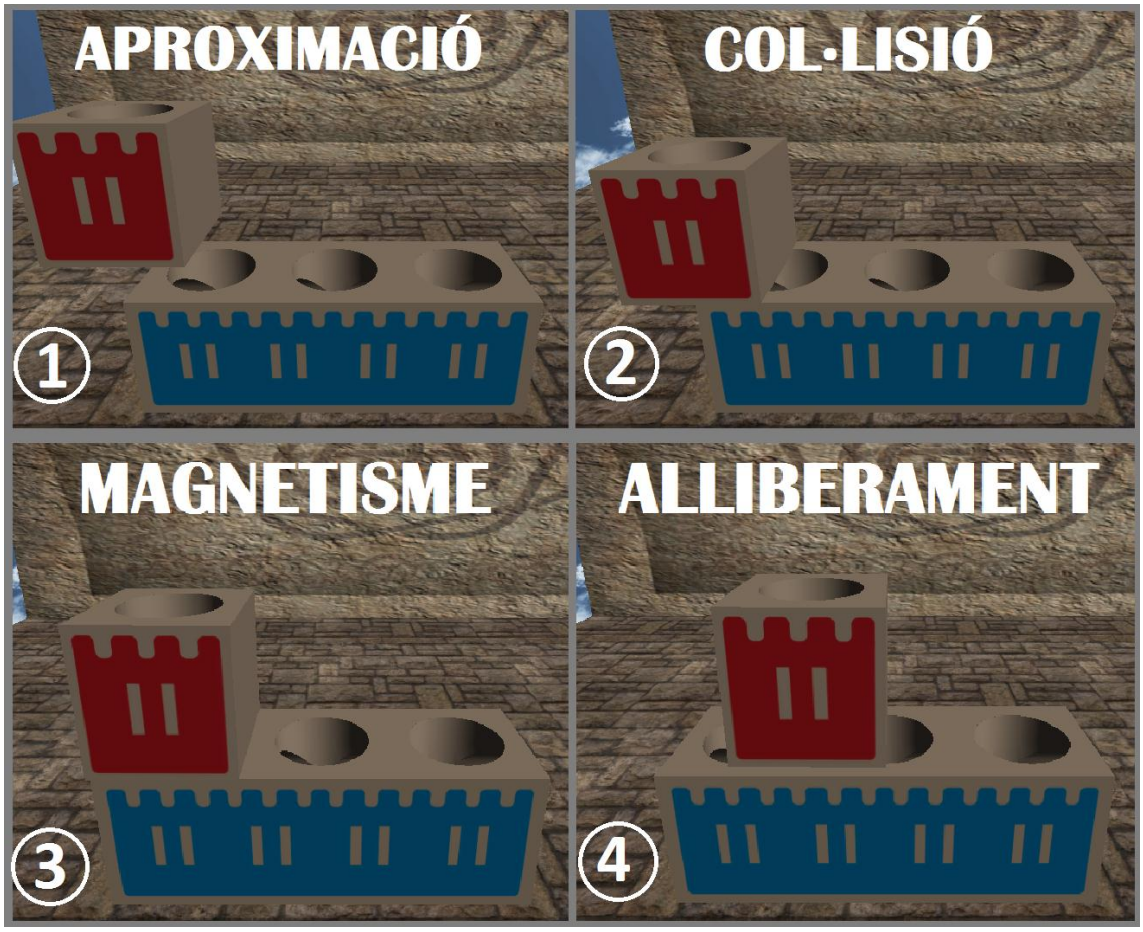

Figura 31. Fases de l'efecte magnètic

Un cop posicionades totes les peces es comprova que la solució sigui correcta a través d'un vector de sis posicions on s'emmagatzema la posició de les peces respecte a la peça blava. Si les posicions dels objectes es corresponen amb les dades emmagatzemades finalitza el nivell.

2.10.2 Joc de conducció

En aquest apartat s'esmenta el procés seguit alhora de dissenyar una demostració sobre els controls de Kinect en un joc de conducció. Òbviament els controls per al desplaçament seran les mans, però a diferència de la resta d'implementacions, en aquest cas l'usuari adoptarà una postura que emularà l'acció d'agafar un volant amb elles.

Aquest control és el més proper a la realitat de tots, ja que les accions a realitzar per controlar el vehicle són exactament les mateixes que les que s'utilitzen per controlar un vehicle. No ens hem volgut limitar a la conducció d'un automòbil. Per aquest motiu, la demostració consisteix a controlar un avió. Les raons d'aquesta decisió es basen en que l'avió afegeix dues funcionalitats més : l'ascensió i el descens.

No s'han aplicat físiques reals a la demostració, degut a que aquesta té com a objectiu demostrar la funcionalitat dels controls i no la simulació en si.

A continuació passarem a explicar el funcionament dels controls. La posició inicial per a l'usuari és la mateixa que s'adopta al conduir un cotxe. Les dues mans s'han de trobar posicionades a l'alçada del pit, amb una separació entre elles d'uns 20 o 30 centímetres. Els moviments bàsics per realitzar els girs són els mateixos que fer girar un volant, rotant les nostres mans prenent el centre del volant com a l'eix de rotació. Quan apropem les dues mans a la càmera els controls interpretaran aquesta acció com a un descens, amb la corresponent sensibilitat vinculada a la distancia desplaçada per les mans. Al allunyar les mans realitzarem un ascens amb la mateixa sensibilitat que l'acció anterior.

Les coordenades de les mans que necessitarem per aquesta implementació seran coordenades món, ja que necessitem conèixer la posició de cadascuna d'elles en referència a un mateix espai amb la finalitat de calcular la rotació realitzada quan l'usuari realitza un gir.

Per obtenir aquesta rotació tractarem els dos vectors corresponents a la posició de les mans, restant-los per així obtenir un vector direcció. Quan coneixem aquest vector, a través de

les formules trigonomètriques podrem calcular els graus corresponents al gir realitzat per l'usuari.

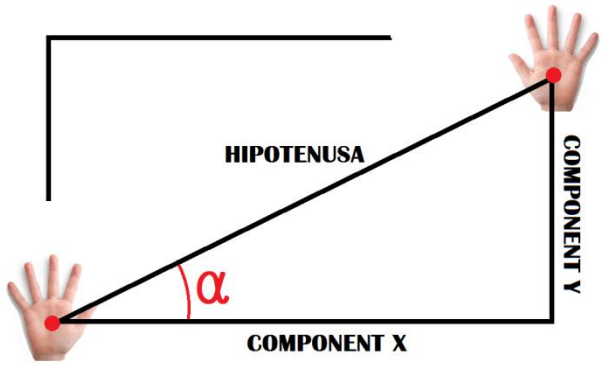

Figura 32. Càlcul de l'angle generat entre les dues mans

Aquest vector contindrà les dues coordenades (X,Y). Seguidament especificarem el valor absolut de les coordenades com la dimensió d'un rectangle. Aquest fet ens permetrà calcular l'angle que hi ha entre les dues mans (la mà esquerra seria el vèrtex inferior esquerra i la mà dreta el vèrtex superior dret), si tallem el rectangle obtingut a través d'una recta que passi per els dos vèrtexs corresponents a les mans obtindrem un triangle rectangle (figura 32). A partir d'aquí necessitem calcular l'angle corresponent a la zona inferior esquerra i per fer-ho utilitzarem una fórmula trigonomètrica. Primer de tot necessitem conèixer el valor de la hipotenusa, sumant els valors elevats al quadrat dels components (X,Y) i seguidament realitzant l'arrel quadrada del resultat.

Calcularem el valor de l'angle α a través del cosinus, però per fer-ho hem de dividir el valor del component X entre el valor de la hipotenusa. Finalment obtindrem el valor de l'angle calculat l'arccosinus del resultat. En el cas en que el component Y del vector direcció sigui negatiu, per calcular l'angle real, li restarem l'obtingut a 90 i finalment el multiplicarem per -1. Un cop conegut l'angle que realitza l'usuari al girar, podem aplicar aquesta rotació entorn a l'eix Y de la càmera. La fórmula és la següent:

$$\alpha = ArcCos(Component\_X / Hipotenusa)$$

Per poder realitzar l'ascens i el descens s'utilitzarà la informació que ens proporciona el mòdul d'obtenció dels vectors de les mans. Degut a que aquests valors ja estan normalitzats els utilitzarem directament per realitzar les rotacions a la càmera sobre l'eix X. D'aquesta manera,

quan l'usuari apropi les mans cap al seu cos, la càmera realitzarà un ascens en proporció a la distància del desplaçament i quan les allunyi realitzarà un descens.

L'ultima de les funcionalitats és la que ofereix a l'usuari la possibilitat de incrementar o reduir la velocitat de la càmera. Per fer-ho emprarem les dades que ens proporciona el mòdul de reconeixement d'esdeveniments. Quan l'usuari realitzi un desplaçament cap a l'esquerra amb la mà esquerra, els controls de l'avió (càmera) passaran a valdre zero, mantenint així la direcció establerta, i a través de la dreta podrà incrementar la velocitat allunyant la mà del seu cos o reduir-la apropant-la com si es tractes del control de velocitat d'un avió. En el moment en que la velocitat sigui la desitjada, per tornar al mode normal, amb la ma esquerra haurà de realitzar un desplaçament a la dreta i tornar a col·locar les mans en la posició inicial. En aquest moment els vectors de les mans prenen com a punt focus la posició de les mans i es continua amb el pilotatge.

Aquest control demostra una correcta sensibilitat als moviments, i és còmode per als usuaris ja que l'ús d'un volant és el principal mètode de control en un vehicle i resulta familiar.

2.10.3 Joc d'acció

En aquest apartat s'implementen els controls per a un joc d'acció. En aquesta demostració el personatge principal portarà una pistola, i per tal d'apuntar, l'usuari haurà d'adoptar la mateixa postura com si també en portés una. Amb aquest control s'intenta que els moviments que ha de realitzar l'usuari siguin semblants als que hauria de dur a terme en una situació real.

A l'hora de desplaçar-se per l'escenari, es realitzarà el càlcul de la seva posició mitjançant les dades proporcionades per el mòdul d'obtenció del vector del tors. L'usuari haurà d'avançar respecte a la seva posició inicial per que la càmera es mogui, de la mateixa manera que s'ha implementat en el mòdul de la càmera en primera persona.

A l'hora d'apuntar, es prendran com a referència dos posicions, la de la mà i la del cap. Restant aquestes dues posicions s'obtindrà el vector direcció que ens indicarà cap a on apunta l'usuari.

Començarem bloquejant la posició del punter del sistema per tal de que aquest quedi fixe al centre de la pantalla, i l'ocultarem mitjançant els mètodes que ens proporciona el motor gràfic.

Per tal de simular que el jugador és una persona, s'ha descarregat un model gratuït d'internet i s'ha posicionat en front de la càmera. Aquest s'ha de vincular a la posició de la càmera per tal de que no desaparegui al realitzar els desplaçaments o les rotacions (figura 33).

Figura 33. Captura de pantalla del joc d'acció. Pistola a la mà.

Per calcular el vector direcció necessitem utilitzar les coordenades món. Per tant accedirem al mòdul de tractament de dades per obtenir-les. Per tal de calcular els graus horitzontals i verticals realitzarem el mateix càlcul trigonomètric que en el punt anterior, però en aquest cas per duplicat. El càlcul dels graus a rotar sobre l'eix Y es calcularan a través del components (X,Z) del vector direcció, i els graus a rotar sobre l'eix X es calcularan a través dels components (Y,Z).

Així doncs, un cop obtinguts els graus corresponents a les dues rotacions, les aplicarem sobre la càmera per que l'usuari pugui observar tot l'escenari realitzant modificacions a la seva postura.

En el cas dels desplaçaments, els controls utilitzats són els mateixos que els de la càmera en primera persona (Punt 2.5). La posició del tors de l'usuari en el moment en que comença la demostració es pren com a zero, per tant, si l'usuari fa un pas enrere, la càmera retrocedirà fins que aquest es torni a col·locar en la posició inicial.

Un dels inconvenients a l'hora d'utilitzar aquest control mitjançant Kinect és que OpenNI només proporciona les dades de les articulacions bàsiques de l'esquelet (Punt 1.3.3, figura 12) i només és capaç de reconèixer la mà; per poder realitzar accions com ara disparar l'arma d'una manera semblant a com ho faríem a la realitat, seria necessari un reconeixement més complert,

que ens permetés obtenir la posició dels dits, podent així controlar les accions realitzades per aquests.

3 RESULTATS

A continuació s'explicaran els processos realitzats amb la finalitat de validar el correcte funcionament de cadascun dels mòduls implementats i els resultats obtinguts en base als usuaris que els han provat. Més tard es mostrarà l'aspecte final de les aplicacions test realitzades, els problemes trobats a l'hora de realitzar la integració i finalment parlarem del futur del projecte.

El concepte de validació és molt important per a qualsevol aplicació, ja que permet trobar la gran majoria d'errors i corregir-los per obtenir una versió definitiva del producte, eficient i de qualitat.

Validació del codi

Per a la validació dels mòduls a nivell intern s'han dut a terme diferents proves. Per assegurar que el codi dels mòduls s'executa correctament s'han utilitzat les eines que proporciona *Unity* per recórrer el codi pas a pas durant l'execució, comprovant els valors que prenen les variables i el correcte funcionament dels mètodes i funcions implementats.

Un cop comprovada l'execució del codi, la primera de les proves és mostrar per pantalla les dades que calcula cadascun dels mòduls. En el cas de la obtenció dels vectors de les mans, durant un cert temps es modifica la posició de les mans i es verifica que els vectors obtinguts es corresponen amb les posicions d'aquestes.

En el cas del reconeixement dels esdeveniments, es mostra per pantalla els resultats obtinguts, realitzant durant un període de temps els moviments desitjats i comprovant que la precisió dels reconeixements no es vegi afectada en base al temps d'execució. Per regular la sensibilitat d'aquests és necessari modificar els paràmetres i tornar a provar el mòdul fins que el reconeixement sigui fiable i estable.

Després de validar el codi, és molt important realitzar una sèrie de proves sobre el dispositiu Kinect, per descartar possibles errors de reconeixement. Per fer-ho, s'han realitzat proves amb tres diferents il·luminacions per veure si la llum afecta al correcte funcionament del dispositiu. La figura 34 ens mostra les tres il·luminacions amb les que s'ha realitzat la prova. La primera és la il·luminació màxima, portada a terme durant el dia i afegint les llums de l'habitació.

La segona il·luminació només utilitza la llum solar, i la tercera és la mínima, on la visibilitat és reduïda. A l'esquerra de cada imatge s'hi troba la seva corresponent imatge en profunditat, que és en la que es basa Kinect per a realitzar els càlculs.

Figura 34. Diferents proves del reconeixement en base a la il·luminació.

Tal i com s'observa a les imatges en profunditat, la il·luminació no afecta al reconeixement, degut a que la càmera només s'utilitza per capturar textures en el cas de voler-les aplicar sobre algun objecte. Un cop testejat el dispositiu, sabem que entre aquestes tres configuracions d'il·luminació Kinect treballa correctament.

Quan tots els mòduls s'executen correctament i proporcionen les dades desitjades, s'utilitzen les dades que calculen per a controlar diferents elements del joc, i finalment un grup

d'usuaris externs proven aquests controls per a proporcionar les seves opinions al respecte i millorar així el procés de validació.

Validació amb els usuaris

Amb la finalitat de millorar les funcionalitats que ofereixen els controls per Kinect, un grup d'usuaris els han provat al llarg del procés de desenvolupament i se'ls ha realitzat enquestes on es recullen les seves sensacions i opinions. Aquestes es troben adjuntades a l'annex I.

Les opinions dels usuaris no solament s'han tingut en compte en el moment de la validació sinó que també han permès refinar i redefinir el comportament dels diferents mòduls. La figura 35 mostra el diagrama de flux que descriu el procés de validació i la interacció d'usuaris externs en aquest. En el procés de desenvolupament de la figura, un cop implementat el disseny de cadascun dels mòduls, s'ha validat la seva funcionalitat, que en aquest cas correspon a la manera en que l'usuari ha de dur a terme les accions (desplaçaments, pulsacions, etc.). Un cop aquesta funcionalitat és comprovada per l'equip, es posa a prova per diferents usuaris amb l'objectiu de refinar-la en base als seus criteris.

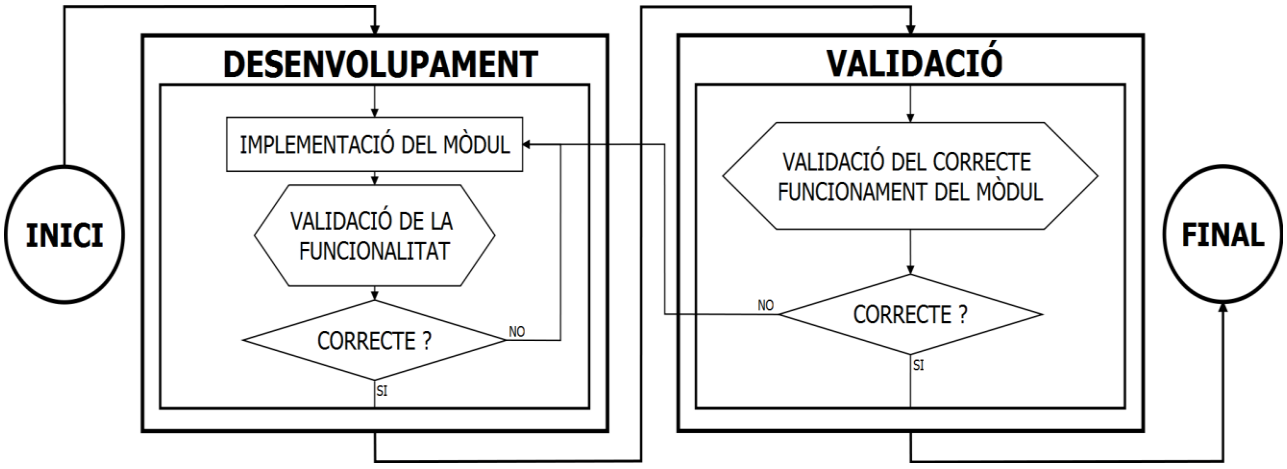

Figura 35. Diagrama de flux del procés de validació.

Cada cop que és necessari modificar característiques com la sensibilitat en el cas del punter o la velocitat dels desplaçaments en el cas del reconeixement d'esdeveniments, es torna a la implementació del mòdul, fins obtenir els resultats desitjats.

Un cop s'ha finalitzat la implementació del mòdul comença el procés de validació. A diferència de la validació de la funcionalitat on s'han analitzat les sensacions de l'usuari, en aquest cas, és necessari centrar-se en l'anàlisi del comportament del mòdul durant l'execució. Per fer-ho s'han realitzat totes les possibles accions, posant a prova els mòduls unitàriament i posteriorment en conjunt. Les proves que realitzen els usuaris també ens permeten detectar errors d'execució. Si es detecta qualsevol anomalia, es torna al procés de desenvolupament amb la finalitat de trobar la causa de l'error i corregir-la. Aquest procés es du a terme fins que no siguem capaços de detectar cap més error. Tot i així, encara que nosaltres no en detectem cap, mai podrem assegurar al 100% que la nostra aplicació estigui lliure d'errors, però se'n detectaran una gran part.

En les enquestes, primer es puntuen diferents aspectes dels mòduls desenvolupats, aquests aspectes son: intuïció, comoditat, estabilitat, sensibilitat, utilitat i fiabilitat. L'usuari els valorarà ràpidament a través del quadre de d'avaluació.

La gran majoria d'usuaris han coincidit al puntuar la primera part, indiquen que els controls són intuïtius i fàcils. No obstant, molts usuaris han penalitzat l'ús del punter controlat per les mans. Segons moltes opinions, els menús on s'utilitza aquest control serien molt més còmodes si aquests utilitzessin els desplaçaments. Sempre hi hauria una selecció activa que destacaria sobre la resta i a través dels desplaçaments modificarien la selecció (figura 36 B), en comptes de desplaçar el punter fins posicionar-se sobre el botó desitjat (figura 36 A).

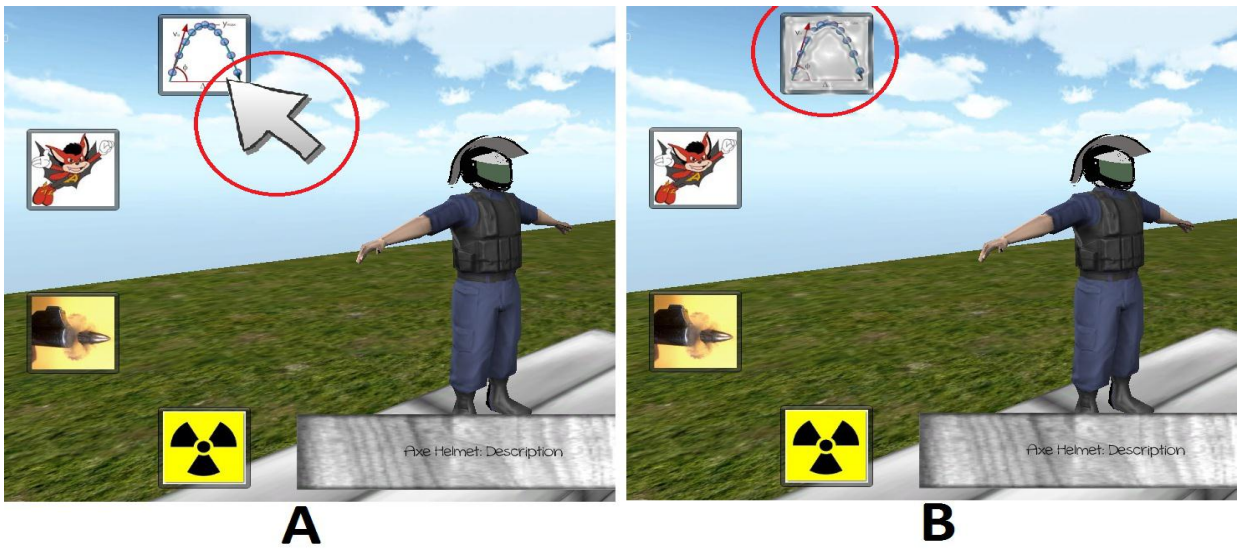

Figura 36. Tipus de controls en el menú.

Finalment es demana una enumeració dels avantatges i inconvenients d'utilitzar la tecnologia Kinect per a controlar les aplicacions test. En aquesta part hi trobem més diversitat d'opinions. Així doncs, s'exposen els principals avantatges i inconvenients que resumeixen les sensacions dels usuaris, indicant també el percentatge d'aquests que comparteixen opinió:

KINECT CASTLE LOGIX

Figura 37. Usuari provant el funcionament de Kinect Castle Logix®.

Avantatges:

- El control de les peces és molt intuïtiu (80%).
- L'efecte magnètic ajuda molt a col·locar correctament les peces (60%).
- El selector circular facilita la selecció de les peces (30%).

Inconvenients:

- Les peces no es poden agafar tancant el puny (20%).
- No es poden agafar dues peces alhora (10%).

La gran majoria dels usuaris han quedat satisfets amb aquesta aplicació, ja que és un videojoc senzill i el control amb les mans ofereix sensibilitat.

JOC DE CONDUCCIÓ

Figura 38. Usuari provant el funcionament del joc de conducció.

Avantatges:

- La conducció és molt semblant a la realitat (90%).
- La sensibilitat està molt aconseguida (60%).

Inconvenients:

- No es pot modificar la velocitat sense deixar de controlar el volant(80%).
- Al no tenir un volant físicament, no hi ha tant de control al realitzar els girs (10%).

En aquesta aplicació test, els controls són iguals a tenir un volant entre les mans, per aquest motiu a la gran majoria dels usuaris els ha resultat fàcil. El principal inconvenient en el que tothom coincideix, és el problema de modificar la velocitat, ja que per fer-ho s'ha de deixar de controlar el volant.

JOC D'ACCIÓ

Figura 39. Usuari provant el funcionament del joc d'acció.

Avantatges:

- La postura adoptada per apuntar és molt real (45%).

Inconvenients:

- No es pot disparar l'arma d'una marera còmode (95%).
- Al apuntar desorienta molt quan realitzes girs laterals (85%).

Molts dels usuaris han coincidit en que els dos primers jocs són còmodes per jugar-hi amb aquest control, però aquest tercer no hi estan d'acord. El major inconvenient del joc d'acció és que al tenir una visió en primera persona és molt complicat desplaçar-se i realitzar les accions pertinents alhora com disparar. Al no poder reconèixer articulacions com els dits, amb OpenNI aquest tipus de joc està molt limitat. Un altre dels comentaris dels usuaris com a millora ha estat utilitzar comandes de veu durant el joc amb la finalitat de proporcionar més funcionalitats al jugador.

Les edats dels usuaris enquestats es troben entre els 20 i els 60 anys. D'aquesta manera aconseguim unes opinions més variades, tenint en compte que els joves estan més acostumats a jugar de manera més habitual, i a mesura que augmenta l'edat és menor el contacte amb el món dels videojocs. D'aquesta manera s'ha obtingut un rang d'edats més variat tal i com es mostra a la gràfica A de la figura 40. A tots aquests usuaris se'ls ha realitzat l'enquesta durant el procés de desenvolupament (per comprovar la comoditat de les accions) i el de validació (per comprovar el correcte funcionament del mòdul a testejar).

A la primera part de l'enquesta, l'usuari ha de marcar amb un signe "+" els punts que compleixen aquests aspectes esmentats i amb el signe "-" els que no ho fan. D'aquesta manera, s'ha realitzat una mitjana de la satisfacció dels usuaris amb el joc en base a la seva edat, tal i com es veu a la gràfica B de la figura 40.

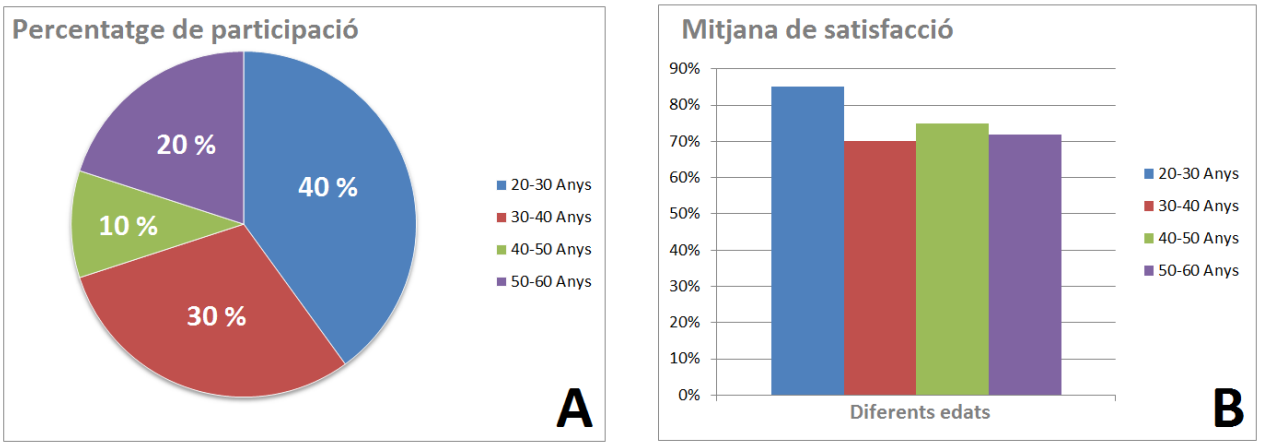

Figura 40. Resultats de les proves als usuaris.

El fet que l'ultima agrupació d'edats entre 50 i 60 anys hagi puntuat el joc positivament, indica que els controls són intuïtius i còmodes. Així doncs, gràcies a la participació d'aquests 10 usuaris hem validat els mòduls d'una manera fiable i hem obtingut idees per possibles millores en els controls de Launchageddon.

Aspecte final

L'aspecte visual de les aplicacions test s'han basat en els dissenys inicials. La figura 41 mostra els dissenys de les tres aplicacions a l'esquerra i a la dreta el seu aspecte final un cop implementats. El punt 1 correspon al disseny inicial i l'aspecte final de l'aplicació test Kinect

Castle Logix®, el punt 2 fa referència al joc de conducció i per últim el punt 3 pertany al joc d'acció.

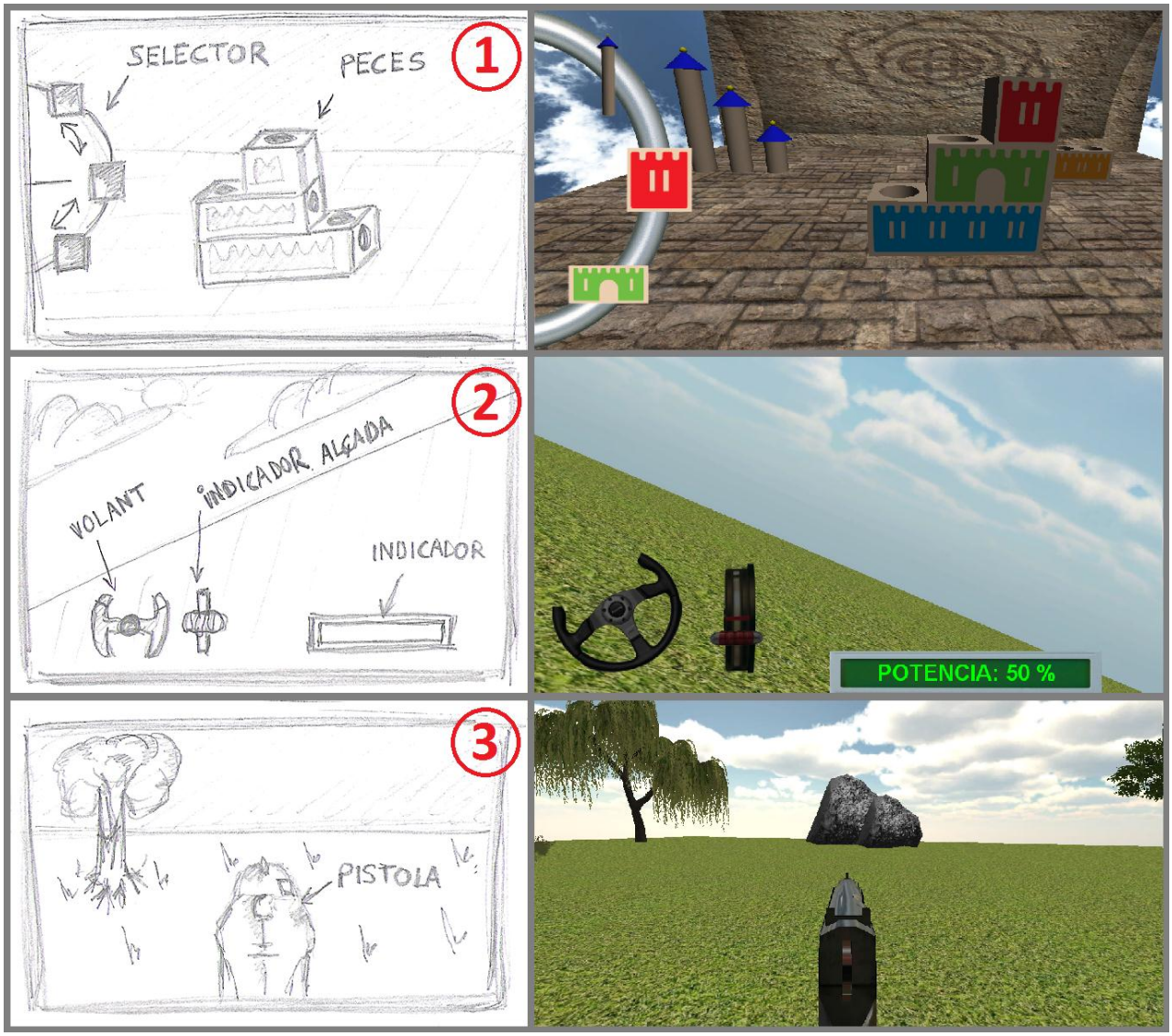

Figura 41. Resultats visuals de les aplicacions: 1 – Castle Logix, 2 – Conducció, 3 - Acció.

Integració

En aquest apartat es mostren tots els passos que s'han realitzat en el moment de la integració entre mòduls. Començarem parlant de la integració dels controls amb Kinect, que s'ha realitzat en comú amb el mòdul de Gameplay i amb l'Editor, i seguidament s'explicarà la integració realitzada per als mòduls realitzats que utilitzarà el Gameplay.

Integració dels controls (Gameplay i Editor)

A les reunions realitzades durant aquest projecte, es van definir tots els accessos als controls del mòdul de tecnologia. D'aquesta manera la resta de mòduls podien conèixer com accedir i utilitzar les dades proporcionades.

Quan es va implementar el mòdul d'Obtenció dels vectors (Capítol 2.2), es va decidir realitzar un mòdul que simulés el comportament d'aquest mòdul però utilitzant el teclat. Aquest mòdul va ser especialment implementat perquè tant el Gameplay com l'Editor poguessin familiaritzar-se amb el tipus de dades amb les que treballaria el mòdul final sense la necessitat d'utilitzar Kinect.

En el cas del mòdul de reconeixement d'esdeveniments (Capítol 2.3), també es van pactar els estats corresponents a cada esdeveniment realitzat per l'usuari, facilitant així la posterior integració.

La resta de mòduls com en el cas de la interacció amb la interfície d'usuari (Capítol 2.4) no necessiten integració, ja que no proporcionen dades a altres mòduls, sinó que directament interactuen amb els elements del videojoc. Així doncs, només ha estat necessari modificar la manera en que el Gameplay i l'Editor els encenen i apaguen

Problema amb els models 3D (Gameplay)

A l'importar els models 3D dissenyats en el mòdul de tecnologia, ens vam adonar que les escales dels models no eren les mateixes que la del model original. Aquest canvi es produeix a l'editar el model original en format (FBX). Alhora d'obrir aquest format amb 3D Studio Max, no es guarda l'escala original del model, així doncs a l'exportar el model modificat, aquest no comparteix la mateixa escala que el model original (vestimenta normal).

La solució al problema proposada va ser utilitzar el model original i modificar la seva textura en temps d'execució per canviar de vestimenta. Això ha provocat que tots els models tinguin el mateix volum i grandària. A l'aspecte final del videojoc ja no es mostren les deformacions que s'han mostrat al capítol 2.8 com per exemple el JetPack de la vestimenta propulsada, o les ales de la vestimenta paracaigudista. Així doncs, a la figura 42 es mostra el disseny dels models anteriors i el disseny final un cop solucionat el problema.

Figura 42. Disseny dels models – Superior: Disseny inicial, Inferior: Disseny final

D'altra banda, a l'integrar els cascs del jugador no vam trobar el mateix problema. Al ser models que tridimensionals que anaven separats dels models de les vestimentes es van poder integrar fàcilment sense masses problemes. Només modificant l'escala i les rotacions per a que fossin iguals i substituint els cascs per a cadascun dels models vam obtenir els models finals. Els cascs al igual que les vestimentes es modifiquen en temps d'execució al selector de personatge.

El problema amb les escales dels models en va obligar a modificar el mòdul de selecció de personatges (Capítol 2.9), ja que ara només hi ha un model base a través del qual es modifiquen les textures al realitzar la selecció. El resultat final del selector de personatges es mostra a la figura 43, on es mostren els botons pertinents a la selecció de la vestimenta al costat esquerre, i al costat dret els botons de la selecció del casc.

Figura 43. Selector de personatges final

Integració de la càmera en primera persona (Gameplay)

La integració de la càmera en primera persona realitzada al mòdul de tecnologia (Capítol 2.5) va ser bastant senzilla. Inicialment van ser analitzades les dependències de codi necessàries, i més tard, es va dur a terme la importació del codi i els GameObjects associats.

Per canviar de càmera només va ser necessari implementar un mètode capaç d'activar i desactivar el funcionament del mòdul en el moment que ens interesses en temps d'execució.

Futur del projecte

Encara no s'ha parlat de les intencions en un futur, però veient la capacitat de treball en equip dels membres del projecte no hi ha dubte que es podria arribar a comercialitzar aquest producte o treballar en un de nou amb la mateixa intenció.

Ha estat una feina molt dura partir d'una idea inicial i repartir-la entre els components per finalment obtenir com a resultat aquests tres projectes. La constància i la compenetració han estat aspectes fonamentals per concloure amb èxit el projecte comú.

4 CONCLUSIONS I MILLORES

Pel que fa al projecte global:

- S'ha desenvolupat un videojoc 3D anomenat *Launchageddon.*
- S'ha dissenyat un Gameconcept inicial per aquest joc. Document que ha servit per dissenyar la implementació i dividir les tasques entre els membres de l'equip i establir els mòduls i les seves comunicacions.
- S'ha dut a terme la integració dels tres mòduls que composen el projecte de forma satisfactòria i s'han solucionat els problemes provinents de la integració.

Pel que fa al projecte concret:

- S'ha investigat el funcionament de *Unity* i l'ús de les llibreries proporcionades per *OpenNI*.
- S'han dissenyat i implementat els mòduls corresponents a la part de tecnologia per proporcionar les dades calculades a la resta de mòduls.
- S'han dissenyat i implementat el mòdul per a tractar les dades proporcionades per Kinect en base al nostre esquelet.
- S'han dissenyat i implementat els mòduls que proporcionen els vectors corresponents a les mans i el tors.
- S'ha dissenyat i implementat un reconeixedor d'esdeveniments en base als moviments que realitza l'usuari durant el joc.
- S'ha dissenyat i implementat un mòdul que permet gravar els moviments realitzats per l'usuari i després reproduir aquests moviments sobre un model 3D.
- S'ha implementat un mòdul que substitueix el ratolí, proporcionant les mateixes funcions que aquest, sense necessitat d'estar en contacte amb cap dispositiu.
- Les aplicacions test desenvolupades han permès demostrar que Kinect és una eina molt potent i que en el cas del nostre videojoc permet a l'usuari jugar-hi sense perdre funcionalitats. No obstant, també ens han fet veure que certs tipus de joc com els d'acció necessitarien un reconeixement més detallat del jugador per ser capaços de realitzar totes les accions del videojoc d'una manera real.

- Les principals incidències han estat associades a la planificació inicial. En base als objectius proposats i a la inexactitud dels càlculs del temps necessari per desenvolupar cadascun d'ells, s'han dedicat moltes més hores de les planejades per a la realització d'aquest.
- Els divers proporcionats per *OpenNI* són molt complicats d'instal·lar i han sorgit incidències a arrel d'aquesta complicada instal·lació, ja que en certs moments l'ordinador no reconeixia al dispositiu.
- Recomanaria l'ús de *Unity* per a futurs usuaris, ja que és un motor potent i estable que ha permès desenvolupar aquest projecte.
- A nivell personal ha estat un plaer tenir l'oportunitat de realitzar un projecte on s'ha partit d'una idea pròpia i s'ha disposat d'una plena llibertat en el desenvolupament.

Com a millores caldria destacar:

- Utilitzar altres llibreries per Kinect que permetin un reconeixement més exhaustiu de l'usuari, afegint el reconeixement dels dits i les rotacions del cap i de les mans.
- A través de la càmera de Kinect realitzar un escaneig facial a l'usuari per tal de que el seu personatge pugui tenir la seva cara.
- Permetre a l'usuari escanejar objectes personals amb la finalitat d'incloure'ls als seus escenaris.

5 BIBLIOGRAFIA I REFERÈNCIES

[JorCar] : Jordi Cartes Rosell, "Launchageddon: Editor de nivells", Projecte de Final de Carrera d'Enginyeria Informàtica, Escola d'Enginyeria UAB, 2012.

[MarSue] : Marcos Sueiro Eglicerio, "Launchageddon: Gameplay", Projecte de Final de Carrera d'Enginyeria Informàtica, Escola d'Enginyeria UAB, 2012.

[3DMax] : www.autodesk.es/3dsmax, últim accés: 25/08/2012, web principal del producte.

[3DMn] : www.foro3d.com/f112/manual-3d-studio-max-8-instituto-tecnologico-durango-60725.html, últim accés: 12/09/2012, manual d'usuari per a 3D Studio Max.

[AngBir] : www.rovio.com/en/our-work/games/view/1/angry-birds, últim accés: 05/06/2012, pàgina principal del videojoc *Angry Birds*.

[AngEul] : www.euclideanspace.com/maths/geometry/rotations/euler/index.htm, últim accés: 10/03/2012, definició i càlcul dels angles d'Euler.

[CstLgx] : www.smartgames.eu/en/smartgames/castle-logix, últim accés: 15/04/2012, pàgina web del fabricant on s'explica la metodologia del joc.

[MRot] : mathworld.wolfram.com/RotationMatrix.html, últim accés: 10/01/2012, definició i funcionament de les matrius de rotació.

[PhShp]: www.adobe.com/es/products/photoshop.html, últim accés: 15/03/2012, pàgina web de l'editor d'imatges Adobe Photoshop.

[PrmSns] : www.primesense.com, últim accés: 08/05/2012, pàgina web de PrimeSense, empresa propietària de les llibreries *OpenNI*.

[Quat] : mathworld.wolfram.com/Quaternion.html, últim accés: 21/06/2012, definició i càlcul dels quaternions per a les rotacions.

[SkyBx] : docs.unity3d.com/Documentation/Components/class-Skybox.html, últim accés: 19/05/2012, aplicació de *Skybox* a *Unity.*

[TecRev] : www.technologyreview.com, últim accés: 15/03/2012, pàgina web publicada per l'Institut Tecnològic de Massachusetts (MIT) on es mostren les noves tecnologies emergents.

[UnrDev] : www.unrealengine.com/udk, últim accés: 02/03/2012, pàgina web del motor gràfic *Unreal Development Kit* (UDK).

[Uty3D] : www.unity3d.com, últim accés: 05/07/2012, pàgina web del motor gràfic *Unity*.

[UtyApi] : docs.unity3d.com/Documentation/ScriptReference, últim accés: 12/09/2012, API on s'explica el funcionament de les classes del motor gràfic de *Unity*.

[UtyMan] : docs.unity3d.com/Documentation/Manual/index.html, últim accés: 10/05/2012, manual d'usuari per a principiants de *Unity*.

6 ANNEX

VALIDACIÓ DELS MÒDULS

PROVES REALITZADES AMB KINECT

Dades del provador: Sexe: [X] H [ ] D Edat: 27 Data: 7-8-12

AVALUACIÓ PERSONAL

Indica amb "+" o "-" les característiques que creus poseeix cadascun dels mòduls testejats:

	Intuïtiu	Còmode	Estable	Sensible	Útil	Fiable
CONTROLS DISSENYATS PER A LAUNCHAGEDDON						
Obtenció dels Vectors (Mans i Tors)	+	+	+	+	+	+
Reconeixement d'Esdeveniments	+	+	+	+	+	+
Control del Punter	+	-	-	-	-	-
Selector Circular	+	+	-	+	+	+
Càmera en Primera Persona	+	+	+	+	+	+
Gravació d'Animacions	+	+	+	+	+	+
APLICACIONS TEST SOBRE EL CONTROL AMB KINECT						
Controls Kinect Castle Logix®	+	+	+	+	+	+
Controls Joc de Conducció	+	+	+	+	+	+
Controls Joc d'Acció	-	-	-	-	-	-

LAUNCHAGEDDON Proporciona una breu explicació sobre els aspectes a millorar en els controls d'aquest videojoc

El control del ratolí amb les mans em resulta més complicat que un ratolí corrent. Desde el meu punt de vista els menus serien més facils amb desplaçaments.

APLICACIONS TEST Enumera dos avantatges d'utilitzar Kinect a les aplicacions test

Detecta molt bé els girs del volant.
Quan portes la pistola el moviment és molt real.

Enumera dos inconvenients d'utilitzar Kinect a les aplicacions test

Afegiria un canvi de marxes a la conducció.
M'agradaria poder llençar granades amb la mà.

Observacions:

M'agrada molt el control del joc amb Kinect.

VALIDACIÓ DELS MÒDULS

PROVES REALITZADES AMB KINECT

Dades del provador: Sexe: [X] H [ ] D Edat: 57 Data: 23/8/12

AVALUACIÓ PERSONAL

Indica amb "+" o "-" les característiques que creus poseeix cadascun dels mòduls testejats:

	Intuïtiu	Còmode	Estable	Sensible	Útil	Fiable
CONTROLS DISSENYATS PER A LAUNCHAGEDDON						
Obtenció dels Vectors (Mans i Tors)	+	+	+	+	+	+
Reconeixement d'Esdeveniments	+	+	+	+	+	+
Control del Punter	+	+	+	+	+	+
Selector Circular	+	+	+	-	+	+
Càmera en Primera Persona	-	+	+	+	-	+
Gravació d'Animacions	+	+	+	+	+	+
APLICACIONS TEST SOBRE EL CONTROL AMB KINECT						
Controls Kinect Castle Logix®	+	+	+	+	+	+
Controls Joc de Conducció	+	+	+	+	+	+
Controls Joc d'Acció	-	-	-	-	-	-

LAUNCHAGEDDON Proporciona una breu explicació sobre els aspectes a millorar en els controls d'aquest videojoc

Me gusta mucho la manera en que se controla el videojuego

APLICACIONS TEST Enumera dos avantatges d'utilitzar Kinect a les aplicacions test

Es divertido ver como responden los controles, en la conducción. No necesitas ningun aparato solo tus manos

Enumera dos inconvenients d'utilitzar Kinect a les aplicacions test

Me cuesta lo mismo que jugar con un mando
El juego de acción me marea

Observacions:

—

VALIDACIÓ DELS MÒDULS

PROVES REALITZADES AMB KINECT

Dades del provador: Sexe: [X] H [ ] D Edat: 26 Data: 25/08/2012

AVALUACIÓ PERSONAL

Indica amb "+" o "-" les característiques que creus poseeix cadascun dels mòduls testejats:

	Intuïtiu	Còmode	Estable	Sensible	Útil	Fiable
CONTROLS DISSENYATS PER A LAUNCHAGEDDON						
Obtenció dels Vectors (Mans i Tors)	+	+	+	+	+	+
Reconeixement d'Esdeveniments	+	+	+	+	+	+
Control del Punter	+	+	+	+	−	+
Selector Circular	+	+	+	+	+	+
Càmera en Primera Persona	−	+	+	+	+	+
Gravació d'Animacions	+	+	+	+	+	+
APLICACIONS TEST SOBRE EL CONTROL AMB KINECT						
Controls Kinect Castle Logix®	+	+	+	+	+	+
Controls Joc de Conducció	+	+	+	+	+	+
Controls Joc d'Acció	−	−	+	−	−	+

LAUNCHAGEDDON Proporciona una breu explicació sobre els aspectes a millorar en els controls d'aquest videojoc

Crec que es bastant complet, no hi afegiria res.

APLICACIONS TEST Enumera dos avantatges d'utilitzar Kinect a les aplicacions test

La conducció es molt real ja que es com portar un volant.
Castle Logix: moure les peces es còmode.

Enumera dos inconvenients d'utilitzar Kinect a les aplicacions test

Estric bé poder disparar l'arma amb els dits.
Quan agafes la peça estaria millor tancar els dits.

Observacions:

Cap comentari.

VALIDACIÓ DELS MÒDULS

PROVES REALITZADES AMB KINECT

Dades del provador: Sexe: ☐ H ☒ D Edat: 49 Data: 1/9/12

AVALUACIÓ PERSONAL

Indica amb "+" o "-" les característiques que creus poseeix cadascun dels mòduls testejats:

	Intuïtiu	Còmode	Estable	Sensible	Útil	Fiable
CONTROLS DISSENYATS PER A LAUNCHAGEDDON						
Obtenció dels Vectors (Mans i Tors)	+	-	+	+	+	+
Reconeixement d'Esdeveniments	+	+	+	-	+	+
Control del Punter	+	+	+	+	+	-
Selector Circular	+	-	+	+	+	+
Càmera en Primera Persona	+	+	-	+	-	+
Gravació d'Animacions	+	-	+	+	+	+
APLICACIONS TEST SOBRE EL CONTROL AMB KINECT						
Controls Kinect Castle Logix®	+	-	+	+	+	+
Controls Joc de Conducció	+	+	+	-	+	+
Controls Joc d'Acció	-	-	-	-	-	-

LAUNCHAGEDDON Proporciona una breu explicació sobre els aspectes a millorar en els controls d'aquest videojoc

Sería divertido que el personaje tuviese mi cara en el videojuego.

APLICACIONS TEST Enumera dos avantatges d'utilitzar Kinect a les aplicacions test

El juego de las piezas es muy intuitivo, podría servir para niños.
Reconoce muy bien los gestos y movimientos.

Enumera dos inconvenients d'utilitzar Kinect a les aplicacions test

Hay veces que te desorienta al no tener nada en las manos. El juego de acción es difícil de controlar.

Observacions:

Ha sido divertido probar esta nueva tecnología.

VALIDACIÓ DELS MÒDULS

PROVES REALITZADES AMB KINECT

Dades del provador: Sexe: ☐ H ☒ D Edat: 25 Data: 4/09/2012

AVALUACIÓ PERSONAL

Indica amb "+" o "-" les característiques que creus poseeix cadascun dels mòduls testejats:

	Intuïtiu	Còmode	Estable	Sensible	Útil	Fiable
CONTROLS DISSENYATS PER A LAUNCHAGEDDON						
Obtenció dels Vectors (Mans i Tors)	+	+	+	+	+	+
Reconeixement d'Esdeveniments	+	+	+	+	+	+
Control del Punter	+	-	+	+	+	+
Selector Circular	-	-	+	+	+	+
Càmera en Primera Persona	-	+	+	+	+	+
Gravació d'Animacions	+	+	+	+	+	+
APLICACIONS TEST SOBRE EL CONTROL AMB KINECT						
Controls Kinect Castle Logix®	+	-	+	+	+	+
Controls Joc de Conducció	+	+	+	+	+	+
Controls Joc d'Acció	-	-	-	-	-	-

LAUNCHAGEDDON Proporciona una breu explicació sobre els aspectes a millorar en els controls d'aquest videojoc

No m'agrada gaire jugar, però crec que els controls són bastant còmodos per al tipus de joc.

APLICACIONS TEST Enumera dos avantatges d'utilitzar Kinect a les aplicacions test

El control de los peces amb la mà és molt bo
La conducció és molt real.

Enumera dos inconvenients d'utilitzar Kinect a les aplicacions test

No m'agrada el joc d'acció, és molt complicat
No es poden utilitzar els dits per controlar los peces

Observacions:

VALIDACIÓ DELS MÒDULS

PROVES REALITZADES AMB KINECT

Dades del provador: Sexe: ☐ H ☒ D Edat: 56 Data: 4/09/2012

AVALUACIÓ PERSONAL

Indica amb "+" o "-" les característiques que creus poseeix cadascun dels mòduls testejats:

	Intuïtiu	Còmode	Estable	Sensible	Útil	Fiable
CONTROLS DISSENYATS PER A LAUNCHAGEDDON						
Obtenció dels Vectors (Mans i Tors)	+	+	+	+	+	+
Reconeixement d'Esdeveniments	+	+	+	+	+	+
Control del Punter	+	−	−	−	−	+
Selector Circular	+	+	−	+	+	+
Càmera en Primera Persona	+	−	+	+	+	+
Gravació d'Animacions	−	+	+	+	+	+
APLICACIONS TEST SOBRE EL CONTROL AMB KINECT						
Controls Kinect Castle Logix®	+	+	+	+	+	+
Controls Joc de Conducció	+	−	+	−	+	+
Controls Joc d'Acció	−	−	−	−	−	−

LAUNCHAGEDDON Proporciona una breu explicació sobre els aspectes a millorar en els controls d'aquest videojoc

Hay algunos momentos en que el aparato tarda en reconocerte.

APLICACIONS TEST Enumera dos avantatges d'utilitzar Kinect a les aplicacions test

Muy comodo trabajar con los brazos.
Los movimientos adaptados a la realidad.

Enumera dos inconvenients d'utilitzar Kinect a les aplicacions test

Hay gestos que no reconoce con los dedos.
A veces es más comodo usar un teclado y un raton.

Observacions:

Un juego.

VALIDACIÓ DELS MÒDULS

PROVES REALITZADES AMB KINECT

Dades del provador: Sexe: [X] H [ ] D Edat: 37 Data: 4/09/2012

AVALUACIÓ PERSONAL

Indica amb "+" o "-" les característiques que creus poseeix cadascun dels mòduls testejats:

	Intuïtiu	Còmode	Estable	Sensible	Útil	Fiable
CONTROLS DISSENYATS PER A LAUNCHAGEDDON						
Obtenció dels Vectors (Mans i Tors)	+	+	+	+	-	+
Reconeixement d'Esdeveniments	-	-	+	+	+	-
Control del Punter	+	+	+	-	+	+
Selector Circular	+	+	+	+	+	+
Càmera en Primera Persona	-	+	+	+	+	+
Gravació d'Animacions	+	+	+	+	+	-
APLICACIONS TEST SOBRE EL CONTROL AMB KINECT						
Controls Kinect Castle Logix®	+	+	+	+	+	+
Controls Joc de Conducció	+	-	+	+	+	-
Controls Joc d'Acció	-	-	+	-	-	-

LAUNCHAGEDDON Proporciona una breu explicació sobre els aspectes a millorar en els controls d'aquest videojoc

CREC QUE EL JOC ESTÀ MOLT BE, ELS CONTROLS SÓN COMODES I FÀCILS.

APLICACIONS TEST Enumera dos avantatges d'utilitzar Kinect a les aplicacions test

LA CONDUCCIÓ ÉS MOLT DIVERTIDA
EL JOC D'ACCIÓ ÉS MOLT REAL PERO MOLT COMPLICAT

Enumera dos inconvenients d'utilitzar Kinect a les aplicacions test

LA POSICIÓ DEL JOC D'ACCIÓ ÉS REAL PERO MOLT INCOMODE
ES DIFICIL PROBAR EL PUNT O QUAN MOUS LES PECES DEL CASTLE LOGIX.

Observacions:

NO HI HAN COMENTARIS

VALIDACIÓ DELS MÒDULS

PROVES REALITZADES AMB KINECT

Dades del provador: Sexe: [X] H [ ] D Edat: 35 Data: 5/9/12

AVALUACIÓ PERSONAL

Indica amb "+" o "-" les característiques que creus poseeix cadascun dels mòduls testejats:

	Intuïtiu	Còmode	Estable	Sensible	Útil	Fiable
CONTROLS DISSENYATS PER A LAUNCHAGEDDON						
Obtenció dels Vectors (Mans i Tors)	+	+	+	+	-	+
Reconeixement d'Esdeveniments	-	+	+	+	+	+
Control del Punter	+	-	+	+	+	+
Selector Circular	+	+	+	+	+	+
Càmera en Primera Persona	+	-	+	+	+	+
Gravació d'Animacions	+	+	+	+	+	+
APLICACIONS TEST SOBRE EL CONTROL AMB KINECT						
Controls Kinect Castle Logix®	X	+	+	+	X	+
Controls Joc de Conducció	+	+	+	+	+	+
Controls Joc d'Acció	+	-	+	-	-	-

LAUNCHAGEDDON Proporciona una breu explicació sobre els aspectes a millorar en els controls d'aquest videojoc

(REL QUE ÉS UN CONTROL MOLT CÒMODE, MILLOR QUE EL TECLAT, JA QUE ET FA MOURE EL COS I PARTICIPAR

APLICACIONS TEST Enumera dos avantatges d'utilitzar Kinect a les aplicacions test

EL CONTROL DEL VOLANT S'ASSEMBLA MOLT A LA REALITAT
M'HA SEMBLAT UTIL QUE LES ~~PE~~GES S'ENGANXIN SOLES AL CASTLE LOGIX

Enumera dos inconvenients d'utilitzar Kinect a les aplicacions test

EL INCREMENT DE LA VELOCITAT ÉS COMPLICAT AMB EL JOC DE CONDUCCIÓ
EL JOC D'ACCIÓ ÉS MOLT COMPLICAT, ET DESUBICA I NO ES POT DISPARAR.

Observacions:

NO HI HA CAP OBSERVACIÓ

VALIDACIÓ DELS MÒDULS

PROVES REALITZADES AMB KINECT

Dades del provador: Sexe: [✓] H [ ] D Edat: 27 Data: 6/9/2012

AVALUACIÓ PERSONAL

Indica amb "+" o "-" les característiques que creus poseeix cadascun dels mòduls testejats:

	Intuïtiu	Còmode	Estable	Sensible	Útil	Fiable
CONTROLS DISSENYATS PER A LAUNCHAGEDDON						
Obtenció dels Vectors (Mans i Tors)	+	+	+	+	+	+
Reconeixement d'Esdeveniments	+	+	+	−	+	+
Control del Punter	+	−	+	+	+	+
Selector Circular	+	+	+	+	+	+
Càmera en Primera Persona	−	+	+	−	+	+
Gravació d'Animacions	−	+	−	+	+	+
APLICACIONS TEST SOBRE EL CONTROL AMB KINECT						
Controls Kinect Castle Logix®	+	+	+	+	+	+
Controls Joc de Conducció	+	+	+	−	+	+
Controls Joc d'Acció	−	−	+	−	−	+

LAUNCHAGEDDON Proporciona una breu explicació sobre els aspectes a millorar en els controls d'aquest videojoc

Afegir reconeixement dels dits ~~vers~~ reconeixement veu Sugar més amb la profunditat dels moviments

APLICACIONS TEST Enumera dos avantatges d'utilitzar Kinect a les aplicacions test

Afegeix diversió!!! T'estalvies volant en conducció. Ajuda a viure més el joc.

Enumera dos inconvenients d'utilitzar Kinect a les aplicacions test

Que passi algú entre la Kinect i el jugador. L'espai necessari.

Observacions:

Tots els jocs que he vist amb Kinect no utilitzen més que les que he vist en les aplicacions a excepció del reconeixement o reconstrucció del cos.

VALIDACIÓ DELS MÒDULS

PROVES REALITZADES AMB KINECT

Dades del provador: Sexe: ☐ H ☒ D Edat: 32 Data: 10/9/2012

AVALUACIÓ PERSONAL

Indica amb "+" o "-" les característiques que creus poseeix cadascun dels mòduls testejats:

	Intuïtiu	Còmode	Estable	Sensible	Útil	Fiable
CONTROLS DISSENYATS PER A LAUNCHAGEDDON						
Obtenció dels Vectors (Mans i Tors)	+	-	+	+	+	+
Reconeixement d'Esdeveniments	+	+	+	-	+	+
Control del Punter	+	+	-	+	+	+
Selector Circular	+	-	+	+	+	+
Càmera en Primera Persona	+	+	+	-	+	+
Gravació d'Animacions	-	+	+	+	+	+
APLICACIONS TEST SOBRE EL CONTROL AMB KINECT						
Controls Kinect Castle Logix®	+	+	+	+	+	+
Controls Joc de Conducció	+	-	+	+	+	+
Controls Joc d'Acció	-	-	-	+	-	-

LAUNCHAGEDDON Proporciona una breu explicació sobre els aspectes a millorar en els controls d'aquest videojoc

No se m'acudeix cap aspecte a millorar.

APLICACIONS TEST Enumera dos avantatges d'utilitzar Kinect a les aplicacions test

Els controls s'assemblen molt a la realitat

Enumera dos inconvenients d'utilitzar Kinect a les aplicacions test

Els dits podrien ofrir més possibilitats. M'agradaria utilitzar la meva cara escanejada.

Observacions: