IE2KinectSupport: llibreria d'abstracció del SDK de Kinect per al motor InvasionEngine 2.

A. Castell Vilches

Resum– El mon dels videojocs está en constant canvi i una de les coses que més està canviant és el seu desenvolupament. Empreses que abans necesitaven equips de moltes persones amb vàries disciplines per poder desenvolupar un joc, a dia d'avui es veuen superades per produccions realitzades per nombres reduïts de persones, en ocasions persones que han portat tot el desenvolupament per compte pròpia. Això és en gran mesura propiciat per el desenvolupament de motors de joc cada cop més potents. Alguns exemples son Unity [1], Unreal Engine [2] o Source Engine [3]. En aquest article presentarem un dels mòduls de InvasionEngine 2, un motor de joc propi desenvolupat per Abel Cano Quintana, estudiant de l'Universitat Autònoma de Barcelona. El modul que presentem és el IE2KinectSupport (per les seves inicials Invasion Engine 2 Kinect Support) una llibreria que permet el ús de dispositius Kinect v1 en el desenvolupament de aplicacions amb el InvasionEngine 2.

Paraules clau– Kinect, Kinect V1, Natural User Interface, Depth Sensor, RGBCamera, AudioRecognition, SkeletonTracking, Bone-Track, Game Engine, InvasionEngine, InvasionEngine 2, IE2

Abstract– The gaming world is in constant change and one of the things that is changing most is the development, years before companiesneed many people with various disciplines to develop a game, today these developments are surpassed by productions made by small numbers of people, sometimes appear people who have taken all the development for their own. This is caused by the development of game engines increasingly powerful. Examples include Unity [1] Unreal Engine [2] or Source Engine [3]. In this article we present InvasionEngine2 module, a game engine developed by Abel Cano Quintana, a student of the Autonomous University of Barcelona. The module presented is IE2KinectSupport (by its initials Invasion Engine Support Kinect 2) a library that allows the use of Kinect devices v1 in developing applications with InvasionEngine 2.

Keywords– Kinect, Kinect V1, Natural User Interface, Depth Sensor, RGBCamera, AudioRecognition, SkeletonTracking, Bone-Track, Game Engine, InvasionEngine, InvasionEngine 2, IE2

1 Introducció

El desenvolupament de videojocs cada dia está més present en el món de les tecnologies, movent quantitats de diners cada cop més grans i a un públic cada cop més exigent i influenciat per la innovació i promesa de noves maneres de jugar. Davant d'aquesta societat tant interessada per els videojocs, i amb l'interés afegit de distanciar-se de les companyies habituals, es creen empreses de caràcter independent que gràcies als motors de jocs tenen la capacitat d'oferir propostes noves i de caràcter més personal o independent.

Tot i així també existeixen altres creadors que opten per la generació del seu pròpi motor de joc. D'aquesta manera no només eviten pagar les taxes o llicències que pot suposar utilitzar un motor ja desenvolupat, sinó que poden desenvo-

• E-mail de contacte: albert.castell@e-campus.uab.cat
• Menció realitzada: Computació
• Treball tutoritzat per: Enric Martí Gòdia (Computació)
• Curs 2016/17

lupar videojocs d'una manera més personalitzada i conèixer tota la lògica interna dels seus jocs de primera mà.
El motor InvasionEngine 2 neix com una progressió del seu antecessor InvasionEngine[4] un motor de joc desenvolupat per Abel Cano Quintana, també responsable del desenvolupament del motor InvasionEngine 2. Aquesta nova versió és un nou motor de joc que implementa funcions que permeten instànciar múltiples objectes en 3D i 2D, treballar amb fitxers de àudio i un sistema de Scripting basat en Mono[5].
La particularitat més interesant i que aprofitem en aquest treball és la gran modulartitat que permet aquest motor, que dona joc a poder implementar un número ilimitat de llibreries que augmentin de manera considerable les possibilitats de desenvolupament en Invasion Engine 2.
El modul IE2KinectSupport pretén ser una primera versió que permeti al motor treballar amb dispositius Kinect, una primera presa de contacte amb els dispositius NUI o Natural User Interface.

1.1 Motivació i objectius

Aquest treball té purament una intenció de autoformar-se. Per una banda és planteja el repte d'enfrontar-se a un motor de joc desconegut, que ja ha estat implementat amb unes especificacions concretes que s'han d'estudiar i comprendre per a que la llibreria IE2KinectSupport no doni problemes de compatibilitat. D'altra banda també existeix un repte a l'hora de comprendre el funcionament dels dispositius Kinect V1.
A l'hora de plantejar els objectius s'han determinat 3 classes d'objectius diferents:

- Objectius crítics: si no es compleixen comprometen directament en els requisits mínims de la llibreria.
- Prioritaris: Objectius que s'han d'implementar a la llibreria prioritàriament per assegurar les funcionalitats esperades.
- Secundaris: Objectius que si no s'acompleixen no comprometen el correcte funcionament de la llibreria, però donen qualitat.

Seguint aquestes directius s'han identificat els següents objectius:

- Objectius crítics :
  - Extreure i manipular informació obtinguda amb la càmera RGB de Kinect permetent realitzar captures d'imatge des del motor.
  - Extreure i manipular informació obtinguda des del receptor d'infrarrojos (Depth sensor).
  - Extreure i manipular informació obtinguda sobre seguiment d'esquelet (Skeleton/Bone Tracking) en temps real.
  - Extreure i manipular informació obtinguda d'àudio permetent la capacitat de reconeixement de àudio per part de la multiarray de micròfons del Kinect, permetent enregistrar àudio.
  - Assegurar la connexió i inicialització del dispositiu Kinect, així com de tots els sensors que intervindran en la llibreria (RGB, Depth, Audio i SkeletonTrack)
  - Realitzar tests de validació de totes les funcions de la llibreria realitzades.
  - "Parsejar" les funcions de la llibreria per a què puguin ser cridades des de un "Script".
- Objectius prioritaris:
  - Desenvolupar "demos" que permetin mostrar l'ús de les funcions del Kinect.
- Objectius secundaris:
  - Dissenyar i desenvolupar un petit joc amb IE2 que utilitzi funcionalitats proporcionades per la llibreria Kinect a través del sistema de Scripting.
  - Aprofitar les funcions de àudio per aconseguir realitzar reconeixement de veu.
  - Aprofitar el Skeleton-Tracking per grabar moviments offline i traspassar-los a un model 3D detallat.

1.2 Estat de l'art

L'any 2010 l'empresa Microsoft llença al mercat Kinect V1 technology per a XBOX360, aprofitant una tecnologia basada en construcció d'imatges 2D amb un sensor de distància creada per l'empresa Israelí PrimeSense (actualment comprada per Apple), donant al mercat de les aplicacions un dispositiu capaç de captar imatge, profunditat i àudio a un preu molt assequible. Microsoft va aprofitar el dispositiu per acostar al mon de l'entreteniment una nova manera de interactuar amb els videojocs basada en els moviments articulars del jugador, donant així a la seva consola del moment XBOX 360 la possibilitat de tant de jugar a videojocs com de controlar tota la interfície de menús que proposava la consola.

Arribats a l'any 2017 s'han fet molts avenços en l'àmbit de les Natural User Interfaces dins el mon dels videojocs i els gegants de l'industria tenen cadascú el seu propi sistema.A la figura *1* podem veure una taula comparativa d'alguns dispositius més coneguts
En el món de les Natural User Interfaces cada companyia treballa amb el seu dispositiu cadascun amb les seves avantatges i desavantatges donant la marca d'identitat a la companyia.
El nostre projecte utilitzarà la versió V1 del Kinect. El motiu principal és que ofereix una gran capacitat d'anàlisi de l'entorn de l'usuari (informació RGB, de profunditat i de so) amb una qualitat acceptable. A més a més Microsoft ofereix el SDK per Kinect V1 i V2 de manera completament gratuïta mentre d'altres com el SDK proposat per Sony, el Move.me, suposaria un cost addicional de 100 €.
D'altra banda preferim comptar per a desenvolupar aquesta primera fase del motor amb un Kinect V1 Technology, ja que tot i tenir unes prestacions notablement inferiors respecte el seu descendent, el Kinect V2 Technology, considerem que al ser més barat i també tenint una trajectòria de més de 10 anys amb molta documentació encaixa més bé amb el projecte InvasionEngine 2.

Dispositiu	Companyia	Pros	Contras
 *Kinect V1 Technology*	Microsoft	-Barat -Molt explotat (bona documentació) -Capaç de llegir *RGB,Depth, Audio, Skeleton Track* -Preu assequible -1 dispositiu permet multiplayer	-Especificacions Antigues
 *Kinect V2 Technology*	Microsoft	-Millora totes les especificacions del Kinect V1	-Preu elevat
 *Play Station Move*	Sony Computer Entertainment	-Mix NUI + Controller -Camp de visió més gran (85º)	-Multi Player requereix varis dispositius -No té infrarojos
 *WiiMote*	Nintendo	-Mix NUI + Controller	-Multi player requereix varis dispositius -No té infrarojos

Fig. 1: Dispositius NUI coneguts

2 METODOLOGIA

Per a assegurar la correcta realització dels objectius que s'han proposat, es realitzarà una metodologia de treball basada en les tècniques de desenvolupament àgil actuals. Cada una de les fases consistirà en un periòde de curta durada o *sprint* (màxim 20 hores) i en cas de què es sobrepassés el temps es realitzará una partició de la tasca en subtasques més simples.
Cada desenvolupament de cada funció es realitzarà seguint el diagrama proposat a continuació: *2*.

Fig. 2: Diagrama de la metodologia àgil que s'ha emprat

Afegidament a aquesta metodologia cal dir que s'ha realitzat un treball d'investigació general previ l'inici del desenvolupament on s'han indentificat les funcions bàsiques que s'han d'obtenir del SDK de Kinect V1 [6], així com el seu funcionament intern i com obtenir les dades que volem abstreure posteriorment al motor.

Per a mantenir una copia de seguretat en tot moment s'afegeix també l'ús d'un controlador de versions (GitLab[7]), gestionat amb SourceTree[8] un software que ens proveeix els repositoris Git amb una interfície gràfica, d'aquesta manera el projecte queda compartit entre els desenvolupadors i actualitzat en tot moment.

3 DISSENY DE LA LLIBRERIA

Possiblement una de les parts més complicades en el desenvolupament de tot aquest treball és el disseny, ja que si no tenim en compte les possibles incompatibilitats amb el motor que s'està desenvolupament paral·lelament, es poden originar conflictes interns donant peu a realitzar de nou tot el treball.
La IE2KinectSupport és una llibreria (.dll) que actua com ha mòdul adicional per al motor IE2. La idea principal és que IE2KinectSupport ha de donar tota informació al motor sent totalment independent, és a dir el motor únicament ha de rebre paràmetres que ja estiguin definits dins el seu àmbit de programació, per posteriorment poder interpretar correctament els scripts i poder utilitzar les funcions del IE2KinectSupport sense cap problemàtica. A la figura *3* queda representada la idea que s'ha plasmat.

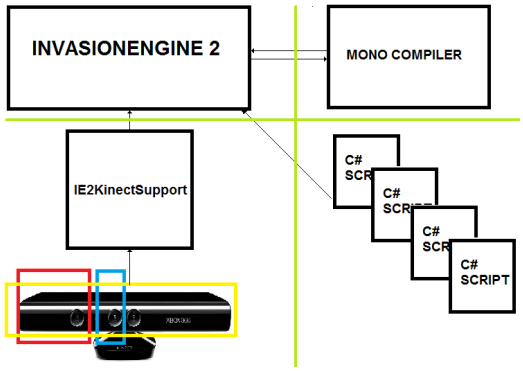

Fig. 3: Disseny a nivell de mòduls distingits dins els motor IE2, les linies de color verd representen la independència entre ells. La llibreria IE2KinectSupport ha de ser coherent amb aquest disseny i per tant abstreure informació sense que el motor, els scripts o el compilador hagin de conèixer variables pròpies del SDK.

Gràcies a aquest tipus de disseny assegurem que la nostra llibreria únicament serà utilitzada per enviar dades al motor sense interferir en la seva funcionalitat habitual, assegurant la modularitat del motor abans esmentada.
A nivell més intern a l'hora de plantejar quin seria el disseny de la IE2KinectSupport, s'han categortizat 3 parts indispensables.
Per una banda tenim la classe mestre IE2KinectSupport que és l'encarregada de definir els constructors que passarem al motor i serà la vía de sortida dels nostres objectes cap al motor.
Posteriorment tenim la classe *KinectTools*, és l'encarregada d'inicialitzar el Kinect i assegurar la seva connexió. A la vegada serà l'encarregada de gestionar les funcions que no depenen directament de cap sensor (moure el angle d'inclinació, inicialitzar els sensors, etc).
Finalment podem agrupar la resta de classes en un últim grup. Serà conformat per totes aquelles clases que depenen de algún sensor en concret i es generaràn a partir de l'objecte realitzat a la classe pare *KinectTools*. Si associem a cada sensor la seva pròpia clase per separat podem mantenir una molt bona organització de desenvolupament, podent per una banda centrar-nos en un sensor en concret per afegir funcionalitats o en cas de trobar alguna errada o *bug* saber a quina classe acudir segons on s'hagi produït l'error *4*.

4 IMPLEMENTACIÓ DE LES CLASSES

4.1 IE2KinectSupport.h

IE2KinectSupport és la classe que ens permet enviar informació al motor i encendre tots els sensors de manera immediata. D'aquesta manera el sistema de scripting només ha de fer la crida al *IE2KinectSupport* assegurant que tots els sistemes del dispositiu Kinect s'activen i s'instancien. Sen-

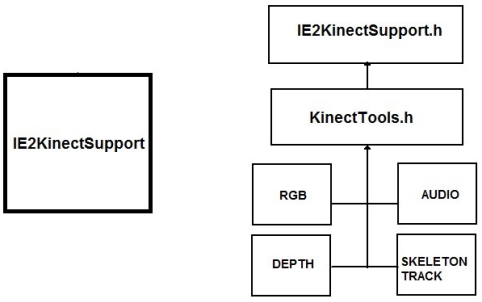

Fig. 4: Disseny a nivell intern de la llibreria IE2KinectSupport

se aquesta classe no podriem enviar informació al motor i per tant la llibreria no funcionaria de la manera esperada encara que internament totes les funcions estiguessin ben dissenyades.

4.2 KinectTools.h

La classe *KinectTools.h* és el nostre nucli i implementa totes les funcions "generals", aquelles funcionalitats bàsiques que no depenen de cap sensor en concret, és a dir tot el que fa referència a la inicialització o calibratge del dispositiu es pot considerar funció general de la llibreria.
Per a la seva implementació, el Kinect SDK v1.8 ens proveeix l'estructura d'un sensor Kinect bàsica i les funcions que permetran la seva localització del hardware, inicialització i modificació dels paràmetres de calibratge.
Cal afegir que d'ara en endavant, quan generem una estructura del tipus de la classe *KinectTools* utilitzarem la direcció de memòria d'aquest objecte per instanciar els objectes derivats generats pels diferents sensors. D'aquesta manera si utilitzéssim varis Kinect podríem seleccionar, per exemple, en un d'ells executar el sensor RGB i en un altre el mòdul Depth.

4.3 KinectRGB.h

La classe *KinectRGB.h* és la classe encarregada d'obtenir la informació rebuda per la càmera RGB del Kinect. A la nostra llibreria hem utilitzat les funcions proporcionades per el SDK per en primer lloc poder emmagatzemar de manera contínua un frame per a la seva posterior representació i en segon lloc una funcionalitat per a poder prendre una foto en un determinat moment i guardar-la al nostre sistema.
El SDK de Kinect es proveeix la funció *NuiImageStreamOpen*. Aquesta funció rep com a paràmetre un flag amb el nom de RGB o Depth en funció de quin tipus d'imatge que volem emmagatzemar, a més a més de variables internes de la classe de tipus HANDLE que emmagatzemaran el propi objecte per a l'obtenció de RGB (*RGBStreamHandle*) i el control d'esdeveniments (*NextColorFrameEvent*). Afegidament a la funció se li passa la qualitat d'imatge o ressolució, que pot ser alta (1280 x 960) o baixa (640 x 480). La creació d'un Stream genera un objecte de tipus *Handle*. Ara el nostre objecte KinectRGB ja por començar a treballar i rebre informació via la càmera RGB del sensor de Kinect. Per a fer-ho utilitzarem la funció *NuiImageStreamGetNextFrame* que rebrà com a paràmetre el objecte anterior que hem creat amb *NuiImageStreamOpen* i a més a més una variable de tipus *NUI_IMAGE_FRAME frame*. Aquesta variable permet a la funció *NuiImageStreamGetNextFrame* emmagatzemar la informació de un frame captat per la càmera, a la figura *5* podem observar que conté exactament l'estructura del frame.

```
typedef struct _NUI_IMAGE_FRAME {
    LARGE_INTEGER liTimeStamp;
    DWORD dwFrameNumber;
    NUI_IMAGE_TYPE eImageType;
    NUI_IMAGE_RESOLUTION eResolution;
    INuiFrameTexture *pFrameTexture;
    DWORD dwFrameFlags;
    NUI_IMAGE_VIEW_AREA ViewArea;
} NUI_IMAGE_FRAME;
```

Fig. 5: Disseny de l'estructura NUI_IMAGE_FRAME, aquesta estructura rep valors a l'hora d'executar la funció NuiImageStreamGetNextFrame.

La variable que conté el mapa de bits que volem posteriorment representar la trobem dins del propi frame a la variable *INuiFrameTexture *pFrameTexture.*. Aquest punter ens trasllada a un objecte de tipus *INuiFrameTexture*, que permet obtenir tota la informació de la textura que nosaltres hem aconseguit amb el sensor. Distingim però dos mètodes imprescindibles per al correcte funcionament de la classe *KinectRGB*. Per una banda tenim el mètode *LockRect* aquest mètode ens emmagatzema en una variable pròpia del SDK de Kinect anomenada *NUI_LOCKED_RECT* el mapa de bits de l'estructura, bloquejada en escriptura. Això es degut a que si no bloquegem correctament aquesta variable es podrien solapar mapes de bits, generant errors d'imatge no desitjats. Per altra banda tenim el mètode *UNLOCKrect* que allibera el frame permetent així que no es produeixin desbordaments de memòria per acumulació massiva de frames.
El motor però ha de ser completament independent a les funcions pròpies del SDK per tant i en ordre de que l'ús de la classe sigui més simple s'han generat 3 funcions bàsiques a l'hora de treballar amb la classe:

- RGBCámera: Constructor propi de la classe que s'encarregará de crear el streamRGB i cridar la funció *NuiImageStreamOpen*.
- RGBProcess: Funció encarregada de donar pas a la captació d'imatge per part de la funció *NuiImageStreamGetNextFrame* i treballar amb la textura per generar el LockedRect que posteriorment pot ser retornat.
- La funció de release per alliberar el frame

La figura *6* sintetitza les últimes explicacions en un diagrama de funcionament.

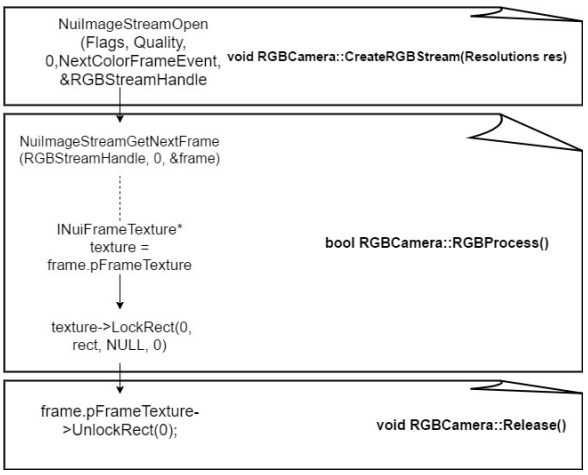

Fig. 6: Diagrama de funcionament per a la classe RGBKinect. A l'esquerra funcions propies del SDK que permeten l'obtenció de imatges en RGB, a la dreta funcions propies de la llibreria IE2KinectSupport.

Finalment s'ha realitzat una funció anomenada ScreenShot que permet realitzar una captura en el moment que es llença la funció, emmagatzemant al nostre sistema el mapa de bits del LockedRect en un precís moment. Aquesta funció també ens ha servit com a test a l'hora de mostrar diferents imatges variant la qualitat. Aquesta funció no ha estat íntegrament generada en aquest projecte ja que no tenim els coneixements necessaris per a realitzar la correcte transformació de un mapa de bits a una imatge, i la investigació hauria comportat hores extres no considerades dins el projecte. Així que hem utilitat una funció OpenSource que té Microsoft per al ús de Kinect. A la figura *7* podem veure una fotografia en alta qualitat realitzada utilitzant les funcions de la classe KinectRGB.

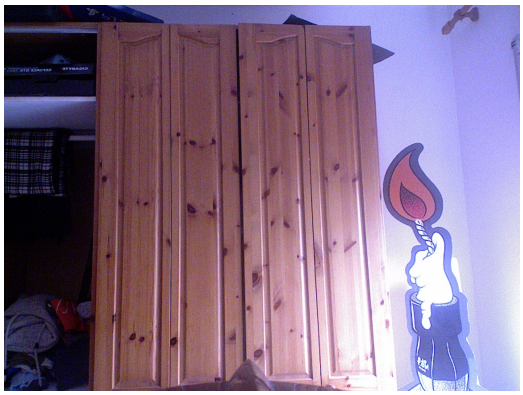

Fig. 7: Fotografia en qualitat 1280x960 RGB realitzada des de IE2KinectSupport

4.4 KinectDepth.h

La classe *KinectDepth.h* té com objectiu retornar mapes de profunditat generats a partir del sensor de proximitat que ens dona Kinect. A l'hora de la veritat aquesta funció quedaria una mica obsoleta ja que posteriorment es pot aconseguir un millor rendiment gràcies al Hardware integrat per al reconeixement d'esquelet ja que ofereix la possibilitat de tenir regions concretes de les quals obtenir la profunditat. Tot i així al ser un dels objectius inicialment marcats a l'hora de fer el treball hem realitzat la clase *KinectDepth* que permet a més generar una imatge del mapa de profunditat generat, aprofitant la funcionalitat Screenshot explicada a l'apartat KinectRGB.h.
El funcionament a l'hora de crear un Stream de tipus Depth és molt semblant al KinectRGB. Ens aprofitarem de la funció *NuiImageStreamOpen*, aquest cop amb un flag Depth i una resolució aquest cop pot ser alta (320 x 240) o baixa (80 x 60), ademés les variables necessàries per emmagatzemar tant el objecte que realitzarà la captura com el DepthEvent segueixen sent dos objectes de tipus HANDLE continguts dins la pròpia classe imitant a la clase KinectRGB.
A nivell de processat de la textura és diferent, ja que estem emmagatzemant un altre tipus de frame diferent al RGB, amb la seves pròpies variables i per tant, tot i utilitzar el *NUI_IMAGE_FRAME frame* la funció que utilitzarem per a capturar imatges dins aquest frame és per una banda la funció que utilitzàvem abans (*NuiImageStreamGetNextFrame*), però ara a més a més utilitzarem *NuiImageFrameGetDepthImagePixelFrameTexture*, una funció pròpia del SDK de Kinect que és capaç d'emmagatzemar dins un objecte tipus *INuiFrameTexture* Texture*, una textura que representa el mapa de bits a nivell de profunditat. Per fer-ho la funció *NuiImageFrameGetDepthImagePixelFrameTexture* rep com a paràmetres:

- El DepthStream HANDLE o variable privada de la clase on hem construit el objecte amb *NuiImageStreamOpen*.
- El frame on hem guardat la captura realitzada per la cámara amb *NuiImageStreamGetNextFrame*.
- Un paràmetre NearMode que informa si estem realitzant captura amb objectes molt pròxims a la cámera.
- La textura *INuiFrameTexture* Texture* passada per referència on guardarem el mapa de bits.

Per obtenir a el mapa de bits, el procés és exactament igual que el que hem realitzat dins del streamRGB. Hem d'aconseguir el LockedRect de la texture emmagatzemada cridant a *LockRect* dins la textura i capturar dins una variable privada tipus *NUI_LOCKED_RECT* la imatge que posteriorment mostrarem.
La funció *ScreenShot* és capaç d'imprimir el LockedRect, així que també formarà part de la classe permetent així poder comprovar el funcionament correcte de la classe, ja que capturar una imatge d'un mapa de profunditat realment no aporta un valor gaire elevat al motor.
A la figura *8* podem observar una fotografia d'un mapa de profunditat obtinguda des de la llibreria IE2KinectSupport

4.5 KinectAudio.h

Les funcionalitats relacionades amb la captació de àudio estan desenvolupades dins la classe *KinectAudio.h*. l'implementació d'aquesta classe ha estat una de les més dificils de tot el treball. En primera instància es volia implementar un reconeixement de veu possibilitant la interacció usuari-màquina mitjançant la veu, però atès que el tractament d'àudio és un procés complex que requereix coneixements sobre transformació a formats audibles, canals de sò,

Fig. 8: Fotografia en qualitat 340x240 Depth realitzada des de IE2KinectSupport

freqüencies, etc no s'ha pogut implementar.Tot i així s'ha aconseguit realitzar la classe KinectAudio.h que és capaç d'enregistrar audio i guardar-lo en format WAV.
El funcionament es basa en dues classes fonamentals. Una d'elles no ha estat realitzada dins aquest treball i forma part de l'OpenSource de Microsoft per treballar amb el Kinect, és l'anomenada *CStaticMediaBuffer*, aquesta classe ens permet definir una variable buffer dins KinectAudio, on emmagatzemar el audio enregistrat a través de l'array de microfons del Kinect.
L'altra clase és la *KinectAudio*, que s'encarregarà de proporcionar mètodes que cridin a les funcions pròpies del SDK amb l'intenció d'enregistrar àudio.
El procediment per enregistrar àudio comença en primer lloc amb la crida de la funció *NuiGetAudioSource* que rep com a paràmetre un objecte de la classe *NuiAudioBeam*, aquest objecte juntament amb la funció permet començar a enregistrar àudio i guarda dins el AudioSource informació de l'ona auditiva generada, per posteriorment si escau representar-la o treballar amb ella. En aquest treball no es treballa amb l'ona auditiva generada.
Tot seguit es criden funcions a través de l'AudioSource generat, en primer lloc cridem a la funció *QueryInterface* que rep com a paràmetres un IID_MediaObject i un punter doble a un objecte del tipus MediaObject contingut dins de la zona privada de la nostra classe KinectAudio. La funció *QueryInterface* ens retornarà un punter a les propietats a les interficies suportades per a la reproducció d'àudio disponibles o en la seva carència un error.
Finalment i per concloure la creació del Stream d'àudio es requereix cridar a través del AudioSource al mètode *QueryInterface* de nou, però aquesta vegada amb un objecte del tipus IID_IPropertyStore com a paràmetre i un punter doble a la variable tipus IPropertyStore per emmagatzemar la interfície de tipus IPropertyStore.
Un cop hem inicialitzat totes les variables necessàries per a l'enregistrament d'àudio toca començar a processar dins d'un buffer els sons captats per a posteriorment poder-los reproduir, per a fer-ho, en primer lloc ens hem de definir com serà el buffer i després fer ús de les funcions de la classe que ens dona Microsoft *CStaticMediaBuffer*. Utilitzarem per aconseguir-ho a més a més un paràmetre de bufferlength triat per l'usuari del motor, que permetrà triar la durada de l'àudio.
Malauradament i per la priorització d'altres objectius no s'ha parametritzat correctament l'àudio i per tant no es pot encara seleccionar a quin fitxer destinar la gravació, tot i així hem pogut comprovar el funcionament enregistrant en un fitxer constant una pista d'àudio i s'ha escoltat correctament.

4.6 SkeletonTracking.h

La part més interessant de la llibreria IE2KinectSupport és l'implementació de mètodes que permetin el SkeletonTracking, la funció més útil de Kinect i la més explotada dins el sector del videojoc.

4.6.1 Què és SkeletonTracking

Les tècniques de SkeletonTracking[9], a vegades també anomenades BoneTracking, permeten el reconeixement de l'esquelet humà a través de la identificació dels ossos i les articulacions que els relacionen i seguir les seves accions. Gràcies a això Kinect es capaç de reconèixer fins 6 esquelets humans al mateix temps i realitzar la representació de 2 d'ells de manera paral·lela. A la figura *9* podem veure les articulacions de l'esquelet que Kinect es capaç de reconèixer i localitzar al espai.

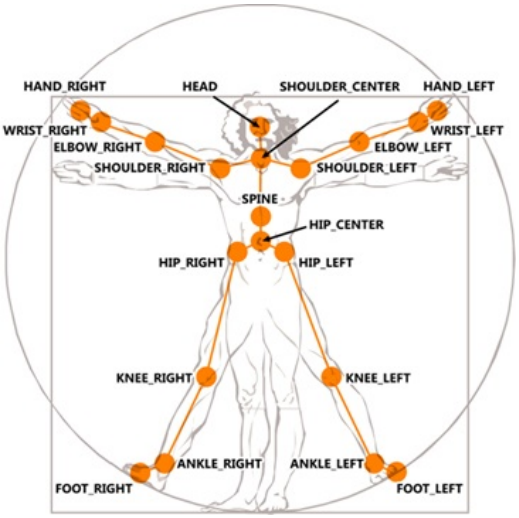

Fig. 9: Articulacions i el seu nom dins el SkeletonTracking de Kinect

Tenim 20 articulacions possibles que són representades en l'espai real (coordenades X, Y, Z i W, sent X, Y i Z coordenades a l'espai i W el quaternió) permetent la completa representació de l'esquelet humà.
Les possibilitats són molt grans una vegada podem aprofitar aquesta tecnologia, es pot realitzar un traspàs de les coordenades a models realistes de caràcter humanoide permetent el que s'anomena MotionCapture[10] en temps real. També es pot realitzar l'enregistrament d'aquests moviments en un fitxer que després sigui interpretat per un model realista 3D i tenir un MotionCapture en offline. D'altra banda també podem interactuar amb objectes dins la pantalla, ja siguin polsadors o objectes 3D que estiguin instanciats dins l'escenari, ja que al tenir també la seva posició dins l'espai podem fer que reacionin quan alguna part del nostre cos comparteixi la seva mateixa posició. Tot això i moltes coses més són posibles gràcies al reconeixement d'esquelet per part de Kinect.

4.6.2 Implementació del SkeletonTracking.h

Per a la realització del *SkeletonTracking.h* necessitarem a la nostra part privada dos objectes tipus HANDLE que s'encarreguin d'emmagatzemar l'enregistrador de SkeletonTrack (*SkeletonStreamHandle*) del Kinect i el control d'esdeveniments en la captura del frames *NextSkeletonFrameEvent*.
La funció que ens proposa Kinect per començar a rebre informació pel sensor de Kinect és *NuiSkeletonTrackingEnable*. Aquesta funció utilitzarà el HANDLE de *NextSkeletonFrameEvent* per iniciar la transmissió. Immediatament després ens hem de declarar un objecte tipus NUI_SKELETON_FRAME, objecte capaç d'emmagatzemar la informació de tot l'esquelet. Per actualitzar aquest frame de manera constant hem de cridar a la funció *NuiSkeletonGetNextFrame* que rebrà com a paràmetre el NUI_SKELETON_FRAME que hem creat anteriorment. Cal comentar que aquest frame no l'hem declarat de manera privada perquè no ens el volem endur sencer, ja que conté molta informació que no és necessària a l'hora de treballar amb el motor. Per finalitzar, es crida a la funció *NuiTransformSmooth* funció que permet suavitzar i tenir dades més correctes de la posició de cada articulació rebent com a paràmetre el propi NUI_SKELETON_FRAME. La figura *10* sintetitza l'obtenció del NUI_SKELETON_FRAME realitzada.

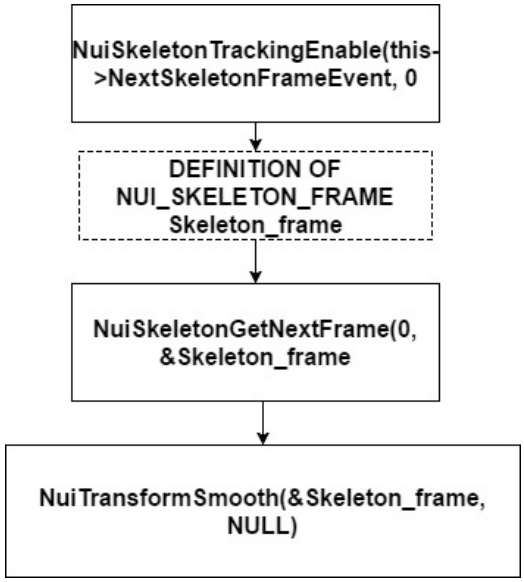

Fig. 10: Procés d'obtenció d'un frame d'esquelet per a la posterior obtenció de les dades de cada articulació

Un cop tenim el NUI_SKELETON_FRAME ens interessa realment poder retornar al motor IE2 les posicions de les articulacions i l'origen de l'esquelet, les altres dades del Frame en aquesta primera versió de la llibreria no són necessàries. Per poder retornar les articulacions únicament ens hem creat una estructura per cada esquelet (6 esquelets en cas que els trobem tots. Aquesta estructura, que hem anomenat SkeletonData té els següents paràmetres:

- int TracState: Aquesta variable informa de quin és l'estat de tracking de l'esquelet. Pot ser TRACKED (ha captat l'esquelet) NOT_TRACKED (no ha captat l'esquelet) o INFERRED (l'esquelet es dedueix a partir de l'escena).
- int SkeletonId: Aquesta variable assigna a un esquelet una variable per a la seva posterior identificació.
- int enrollId: Assigna un esquelet a un jugador.
- Vector4 position: Dona la posició de l'esquelet sencer (posició de l'origen)
- Vector4 SkeletonPositions[20]: Un array de posicions que conté per cada esquelet 20 posicions, una per articulació.
- int PositionStaten[20]: Un array que informa per cada posició el seu estat de tracking, sent TRACKED (ha captat la posició correctament) NOT_TRACKED (no ha captat la posició) o INFERRED (s'ha deduït la posició).
- int Quality: ens informa de la qualitat d' obtenció de l'esquelet.

Per omplir totes aquestes dades de tots els esquelets es realitza un bucle for que itera sobre tots els esquelets que apareixen per pantalla i accedeixen a la informació emmagatzemada dins el NUI_SKELETON_FRAME, d'aquesta manera ja tenim una estructura que conté tota la informació necessària sobre els usuaris davant del Kinect llesta per a ser enviada.
Per ajudar a l'usuari programador que fagi ús del motor s'han definit Getters per cada una de les variables continguda dins de l'estructura de manera que si per exemple l'usuari vol accedir únicament al moviment de la mà dreta ho podrà fer gràcies a la crida a un get tal que:
Vector4[20] Positions = Get SkeletonPositions(id); Vector4 ma = Positions[RIGHT_HAND].

5 PROVES I TEST

Per assegurar el correcte funcionament de les classes prèviament explicades, s'han realitzat diversos casos de test que comproven el seu funcionament i a la vegada tenen l'objectiu de trobar possibles errors que no s'han tingut en compte en una primera fase del desenvolupament. En aquesta secció explicarem els tests més complets que s'han realitzat a nivell de llibreria, ja que els tests a nivell de motor es veuran representats en demostracions completes.
La problemàtica que tenim a l'hora de testejar la llibreria Kinect és que la majoria de resultats només tindran sentit quan s'ajuntin amb el motor, així que ens aprofitem de la clase Log que té el motor InvasionEngine2 que permet mostrar per pantalla l'estat actual de les variables i comprovar que tenen el resultat esperat.

5.1 Test de les funcionalitats bàsiques

Aquest test té com a objectiu comprovar que el Kinect s'encen correctament i té la possibilitat de modificar el seu angle d'inclinació. Per a fer-ho s'intenta fer una declaració d'un sensor Kinect en els seus 3 possibles estats demostrables: Connectat, Desconnectat, Parcialment connectat (Connectat únicament amb usb, sense transformador). L'objectiu de la primera part del test és demostrar que el Kinect es pot

instanciar i a més a més retorna el missatge d'error esperat en cas que no es pugui realitzar correctament la declaració. El resultat obtingut d'aquest test està representat a la figura *11*

```
Tue May 16 16:11:09 2017 - Kinect status: 0
Tue May 16 16:11:09 2017 - Kinect conection 0
Tue May 16 16:11:09 2017 - Kinect initialization status: 5

Tue May 16 16:15:47 2017 - Kinect status: 1
Tue May 16 16:15:47 2017 - Kinect conection 1

Tue May 16 16:20:16 2017 - Kinect status: 1
Tue May 16 16:20:16 2017 - Kinect conection 1
```

Fig. 11: Log de resultats per part del primer test, en primer lloc el Kinect està conectat correctament i obtenim 0 en les primeres (0 READY) la sortida 5 a la inicialització simbolitza (5 Kinect initialized) En els altres dos test no s'aconsegueix inicialitzar correctament el Kinect per tant no s'arriba a inicialitzar i obtenim (1 NOT READY) en les dues primeres sortides

La següent part del test consisteix en poder modificar l'angle del Kinect assegurant-se que no es sobrepassa el límit màxim (-27º,+27º). Per a fer-ho mostrarem en el Log la posició del Kinect actual amb el GetAngle dins la classe i el modificarem amb el SetAngle.El resultat obtingut d'aquest test està representat a la figura *12*

```
Tue May 16 17:52:51 2017 - Kinect status: 0
Tue May 16 17:52:51 2017 - Kinect conection 0
Tue May 16 17:52:51 2017 - Kinect initialization status: 5
Tue May 16 17:52:51 2017 - Kinect's tilt Angle: 0
Tue May 16 17:52:56 2017 - Moving angle to 27 degrees...
Tue May 16 17:53:02 2017 - Kinect's tilt Angle: 27
Tue May 16 17:53:02 2017 - Moving angle to -27 degrees...
Tue May 16 17:53:10 2017 - Kinect's tilt Angle: -27
Tue May 16 17:53:10 2017 - Trying to move kinects angle to 50 degrees
Tue May 16 17:53:15 2017 - Kinect's tilt Angle: -27
Tue May 16 17:53:15 2017 - Moving kinects angle to 0
Tue May 16 17:53:17 2017 - Kinect's tilt Angle: 0
```

Fig. 12: Log del test que verifica el correcte funcionament del Set/Get Angle del Kinect

5.2 Test de RGB i Depth

Gràcies a la similitud entre aquestes dues classes i que el seu funcionament només el podem verificar amb la funció Screenshot hem realitzat un test que consisteix en prendre la mateixa fotografia des de 4 punts de vista diferents, 2 per cada tipus de filtre (RGB i Depth) i 2 per cada qualitat de cada filtre (HIGH i LOW). L'objectiu és comprovar que ambdós objectes es generen correctament i son capaços d'obtenir la imatge desitjada. El resultat obtingut d'aquest test està representat a la figura *13*

5.3 Test del SkeletonTracking

El test proposat per veure que el skeletontracking funciona és una mica abstracte, ja que no podem saber mai exactament quina posició estem ocupant a l'espai en un moment donat, però de tota manera hem estat capaços de veure que el Kinect reaccionava al nostre esquelet i donava sortides encertades del que podria ser la nostra posició a l'espai. Per fer-ho simplement ens hem generat un script que ens treu

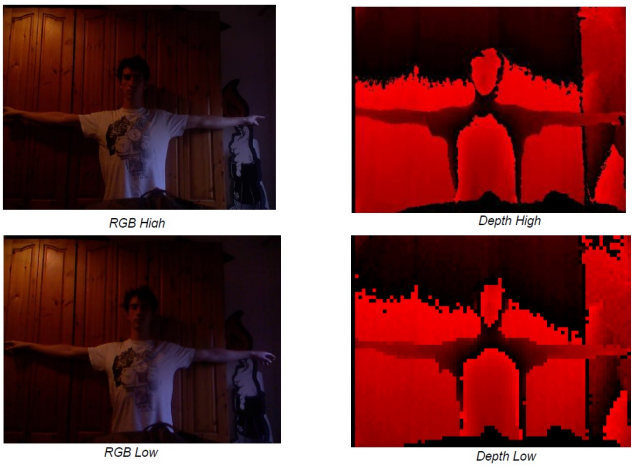

Fig. 13: Representació d'imatges en RGB i en mapa de profunditat Depth en alta i baixa qualitat obtingudes amb IE2KinectSupport.

pel log les 20 possibles posicions de l'esquelet que veu el Kinect. El resultat obtingut d'aquest test està representat a la figura *14*.

```
Sat May 20 22:49:27 2017 - frame number -30
Sat May 20 22:49:27 2017 - central o HIP: x=0.040970, y=-0.197886, z=1.156197

Sat May 20 22:49:27 2017 - espina: x=0.043869, y=-0.162351, z=1.182834

Sat May 20 22:49:27 2017 - hombro centro: x=0.053774, y=0.037508, z=1.202968

Sat May 20 22:49:27 2017 - cabeza : x=0.064171, y=0.220604, z=1.167808

Sat May 20 22:49:27 2017 - espatlla esquerra : x=-0.088755, y=-0.080491, z=1.228322

Sat May 20 22:49:27 2017 - braç dret: x=-0.305079, y=-0.138731, z=1.299537

Sat May 20 22:49:27 2017 - canell esquerra: x=-0.414605, y=-0.197886, z=1.145495

Sat May 20 22:49:27 2017 - ma esquerra: x=-0.435666, y=0.087314, z=1.118557

Sat May 20 22:49:27 2017 - espatlla dreta: x=0.192792, y=-0.091374, z=1.287097

Sat May 20 22:49:27 2017 - braç dret: x=0.410750, y=-0.141149, z=1.282892

Sat May 20 22:49:27 2017 - canell dret: x=0.297016, y=-0.083551, z=1.289975

Sat May 20 22:49:27 2017 - ma dreta: x=0.277675, y=-0.077399, z=1.299244

Sat May 20 22:49:27 2017 - centre esquerra o HIP esquerra: x=-0.024867, y=-0.263816, z=1.130453

Sat May 20 22:49:27 2017 - genoll esquerra: x=-0.162847, y=-0.110980, z=1.308047

Sat May 20 22:49:27 2017 - turmell dret: x=-0.071001, y=-0.349099, z=1.116915

Sat May 20 22:49:27 2017 -  Peu esquerra: x=-0.089460, y=-0.414619, z=1.137538

Sat May 20 22:49:27 2017 - Centre dret o HIP dret: x=0.104411, y=-0.271361, z=1.163609

Sat May 20 22:49:27 2017 - genoll dret: x=0.294113, y=-0.108515, z=1.273535

Sat May 20 22:49:27 2017 - turmell dret: x=0.191140, y=-0.383824, z=1.197737

Sat May 20 22:49:27 2017 - peu dret o HIP: x=0.123740, y=-0.390348, z=1.176710
```

Fig. 14: Log que informa de les posicions obtingudes d'un esquelet situat davant el Kinect amb IE2KinectSupport.

6 DEMOSTRACIONS FINALS

Per demostrar que el treball s'ha realitzat correctament i que la integració de la llibreria amb el motor ha estat correcte. Hem realitzat demostracions que aprofiten la tecnologia del motor IE2 incloent els mòdul IE2KinectSupport. No obstant ens hem trobat amb algunes dificultats a l'hora de representar imatge en temps real obtinguda per la càmera RGB o Depth donat que el motor només és capaç de descodificar i mostrar textures DDS (DirectDraw Surface) mentre que el *IE2KinectSupport* genera imatges bitmap, el que genera una incompatibilitat entre textures. Per fer-ho caldria afegir una capa de conversió BMP cap a DDS, el que afegiria un objectiu nou que no ha donat temps a realitzar.
Tot i així estem satisfets amb el treball realitzat, ja que la

funcionalitat més útil que pot proveir Kinect és la de Skeleton Track i el mòdul IE2KinectSupport otorga eines suficients per a la programació de jocs que depenguin de SkeletonTrack.
Cal comentar que les demostracions s'han programat amb scripts, és a dir estem treballant directament amb el motor IE2 conservant la independència entre mòduls del motor.

6.1 Primera demostració: Generació d'esquelet amb cubs 3D

Un dels objectius de test que era interessant testejar era poder mostrar si realment el motor està reconeixent correctament les posicions. L'única manera de poder-ho comprovar verídicament era realitzant la representació completa de les 20 posicions que conformen un esquelet humà. A més a més també calia comprovar si realment estem reconeixent posicions de dos esquelets diferents.
Per a la realització de la demostració i aprofitant que el motor és capaç d'instanciar diversos objectes en una escena hem realitzat cubs per cada una de les posicions de l'esquelet i hem realitzat les translacions necessàries perquè es col·loquin exactament on el SkeletonTracking ens diu que tenim una articulació. D'aquesta manera som capaços de representar l'esquelet humà sencer. Ademés gràcies a la correcta programació de la classe de SkeletonTracking la demostració permet que dues persones entrin a l'escena i capturar correctament les articulacions dels dos esquelets.
A la figura *15* podem veure una captura de la demostració.

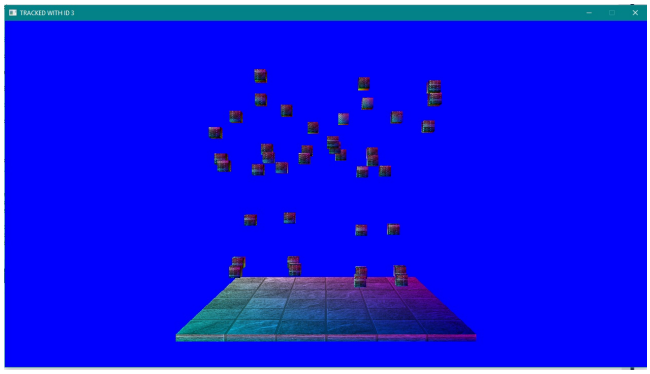

Fig. 15: Demostració de dos esquelets diferents representats amb cubs 3D utilitzant el sistema de scripting del motor InvasionEngine 2.

Aquesta demostració ens ha permès també verificar que l'esquelet s'ha reconegut correctament, Per tant també podriem considerar aquesta demo com un test de validació de la pròpia llibreria.

6.2 Segona demostració: Interacció de tecles de piano amb les mans

Els dispositius Kinect en el món de les aplicacions tenen com objectiu permetre al jugador interactuar de manera directa amb objectes que apareixen a la pantalla (d'allà el nom Natural User Interface) en ordre de verificar que la llibreria IE2KinectSupport té capacitat per a donar aquestes utilitats s'ha realitzat una demostració que consisteix a generar un piano (construït a base de cubs escalats i traslladats de manera equidistant a l'escena) el qual serà tocat per les mans dels usuaris que es posin davant el Kinect. Per fer-ho es comprova únicament que les articulacions etiquetades com a RIGHT_HAND i LEFT_HAND entren en contacte amb alguna de les tecles de piano tenint en compte la seva localització a l'espai.
Les mans les hem representat amb cubs 3D per veure que efectivament quan estem sobre una tecla aquesta emet so. El so és gestionat internament pel mòdul d'àudio del motor InvasionEngine 2. Afegidament a part del so s'aplica una il·luminació a les tecles quan aquestes són premudes per l'usuari.
A la figura *16* podem veure una captura de l'escena proposada dins la demostració.

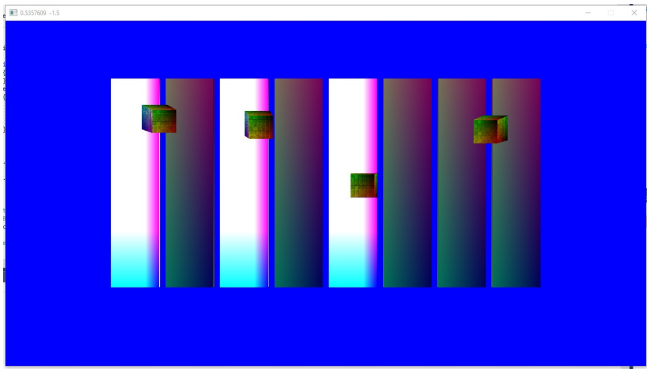

Fig. 16: Demostració de 4 mans (2 per esquelet) representades amb cubs interaccionant amb una representació de tecles de piano, quan els cubs comparteixen un interval de posició amb les tecles aquestes s'il·luminent i emeten un sò

Amb la realització d'aquesta demostració queda demostrat que la llibreria IE2KinectSupport és capaç de donar eines necessàries per a la implementació de Natural User Interfaces dins dels videojocs programats amb InvasionEngine 2.

7 Conclusions

En aquest treball s'ha implementat la llibreria IE2KinectSupport, una llibreria capaç d'aconseguir informació de tots els sensors de la versió de Kinect 1.0 i treballar amb ella en entorns dissenyats amb el motor de joc InvasionEngine. Les funcionalitats més destacables d'aquesta llibreria consisteixen en gran mesura en la possibilitat de treballar amb múltiples esquelets i la posició de les seves articulacions gràcies a les tècniques de SkeletonTracking implementades.
Les incidències més destacables dins aquest treball han estat per una banda la falta de coneixement en algunes àrees del desenvolupament. Kinect és un dispositiu molt complet que treballa amb objectes d'alta complexitat com poden ser buffers d'àudio o textures d'imatge, la falta de coneixement en aquestes àrees ha dificultat el projecte de manera què alguns objectius que a l'inici semblaven més senzills no s'hagin pogut dur a terme, com per exemple el reconeixement de veu i també a l'hora de tenir en compte la compatibilitat amb el motor, com l'error de no poder mostrar textures en BMP a través del motor. Una altra incidència molt rellevant a tenir en compte que s'ha pogut resoldre es el temps de inicialització de sensors, el dispositiu Kinect triga alguns mil·lisegons en posar en

marxa els seus sensors, tant de captació d'imatge, àudio o SkeletonTrack. Per tant s'han hagut de tenir en compte aquest milisegons a l'hora de treballar amb imatges, ja que si no es tenien en compte el motor donava errors fins que no s'encenia el sensor.
El punt fort d'aquest projecte són les línies futures, ja que la llibreria està en una versió inicial. Les millores són moltes però la inclusió de la capa de transformació de BMP a DDS seria una prioritat, ja que permetria la mostra d'imatges en temps real dins el videojoc que s'està desenvolupant. D'altra banda també seria molt interessant el desenvolupament d'un reconeixedor de veu dins la llibreria, així com funcionalitats que permetin el reconeixement gestual, ambdues modificacions permetrien un ús més extens de la Natural User Interface del motor.

AGRAÏMENTS

Al meu company de treball i desenvolupador en cap del motor de joc per compartir el seu coneixement i ajudar en el desenvolupament de la llibreria gràfica i les demostracions.
A Enric Martí Gòdia per tutoritzar aquest treball de principi a fi, aportant correccions, consells i ajuda en tot el seu desenvolupament a més a més d'oferir el material per a poder treballar.
A Adrià Garcia Castejón per aconsellar en fases específiques del treball i donar suport en els moments més complicats.
Als companys del Club de Rem Badalona, per la flexibilitat i suport donats durant tot el desenvolupament del treball.
A la meva família per tot el suport entregat durant tots els estudis.

REFERÈNCIES

[1] Unity enllaç ofical https://unity3d.com/es Darrer Accés Juny 2017

[2] Unreal Engine enllaç ofical https://unrealengine.com/ Darrer Accés Juny 2017

[3] Source Engine https://developer.valvesoftware.com/ Darrer Accés Juny 2017

[4] Invasion Engine http://invasion.uab.es/ Darrer Accés Juny 2017

[5] Monodevelop oficial http://www.monodevelop.com/ Darrer Accés Juny 2017

[6] Kinect SDK https://developer.microsoft.com/Darrer Accés Juny 2017

[7] Gitlab oficial https://about.gitlab.com/ Darrer Accés Juny 2017

[8] SourceTree oficial https://www.sourcetreeapp.com/ Darrer Accés Juny 2017

[9] SkeletonTracking https://blogs.msdn.microsoft.com/esmsdn/2011/08/09/reto-sdk-de-kinect-detectar-posturas-con-skeletal-tracking/ Darrer Accés Juny 2017.

[10] Motion Capture http://www.organicmotion.com/motion-capture/ Darrer Accés Juny 2017.