Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Desenvolvemento dunha aplicación de realidade aumentada para incorporar contido multimedia nunha escena real

Estudante: Iago Martín Mato
Dirección: Julián Alfonso Dorado De La Calle

A Coruña, setembro de 2020.

A sabedoría suprema é ter soños bastante grandes para non perdelos de vista mentres se perseguen.

William Faulkner (1897-1962)

Agradecementos

A L. e J. por estar sempre ahí.

Resumo

O presente proxecto busca realizar unha aplicación a través da cal un usuario poida introducir un elemento dixital, que pode ser ou ben unha imaxe ou ben un vídeo, nunha escena real capturada por unha cámara. Para poder levar a cabo dita funcionalide lévase a cabo a creación dun módulo capaz de crear un espazo de proxección delimitado por un par de targets. O módulo realizará un seguimento da posición e visibilidade dos targets, modificando en consecuencia o espazo. Neste espazo xerado dixitalmente o usuario poderá escoller en tempo real que elemento multimedia desexa proxectar.

Deste modo, o usuario contará cun sistema capaz de xerar unha realidade aumentada modificable en tempo de execución, que poderá ser retransmitida posteriormente por medios convencionais.

Abstract

The present project seeks the creation of an application through the which an user can insert a digital element, that can be either an image or a video, in a real scene captured by a camera. In order to carry out such task we create a module to generate a space of projection delimited by a pair of targets. This module will perform a tracking of the position and visibility of the targets, modifying the space in consequence. In this digitally generated space the user will be able to choose in real time which multimedia element he wishes to project.

Thereby, the user will have a system with the ability to generate an augmented reality modifiable in execution time, which will be able to be broadcasted later by conventional means.

Palabras chave:

- Realidade Aumentada
- Vuforia
- Unity
- Image Target

Keywords:

- Augmented Reality
- Vuforia
- Unity
- Image Target

Índice Xeral

1 Introdución **1**

1.1 Orixe do proxecto . . . 1
1.2 Contexto do proxecto . . . 3
1.3 Obxectivos do proxecto . . . 4
1.4 Organización do proxecto . . . 4

2 Fundamentos **7**

2.1 Realidade Aumentada . . . 7
2.1.1 Historia da realidade aumentada . . . 7
2.1.2 Tipos de realidade aumentada . . . 10
2.2 Vuforia . . . 11
2.2.1 Funcionalidades principais de Vuforia . . . 12
2.3 Unity . . . 14

3 Estado da cuestión **15**

3.1 Reality Engine . . . 15
3.2 Max Reality . . . 16
3.3 Viz Virtual Studio . . . 16
3.4 Brainstorm . . . 16
3.5 tOG-VR . . . 17
3.6 Discusión . . . 17

4 Sistema **19**

4.1 Organización . . . 19
4.2 Tecnoloxías utilizadas . . . 21
4.3 Iteracións . . . 22
4.3.1 Primeira iteración . . . 23
4.3.2 Segunda iteración . . . 26

4.3.3 Terceira iteración . . . 29
4.3.4 Cuarta iteración . . . 32
4.3.5 Quinta iteración . . . 37
4.3.6 Sexta iteración . . . 40
4.3.7 Séptima iteración . . . 42
4.3.8 Oitava iteración . . . 46
4.3.9 Novena iteración . . . 49
4.4 Deseño final . . . 52

5 Conclusións **55**
5.1 Coñecementos utilizados . . . 55
5.2 Traballo futuro . . . 56

A Manual de instalación **61**
A.1 Instalación da aplicación de administración . . . 61
A.2 Instalación da aplicación cliente de retransmisión . . . 61

B Manual de usuario **63**
B.1 Aplicación de administración . . . 63
B.1.1 Funcionamento . . . 65
B.1.2 Control do teclado . . . 70
B.2 Aplicación cliente de retransmisión . . . 70

Relación de Acrónimos **73**

Glosario **75**

Bibliografía **77**

Índice de Figuras

1.1 Exemplos de RA . . . 2
1.2 Exemplos de RA en televisión . . . 2
1.3 Exemplos de RA en directo . . . 2

2.1 Espada de Damocles . . . 8
2.2 Escritura dun texto mediante Videoplace . . . 8
2.3 Primeiro prototipo de EyeTap . . . 8
2.4 Equipamento da plataforma Virtual Fixtures . . . 9
2.5 Detalle do sistema desenvolto por Wagner e Schmalstieg . . . 10
2.6 Exemplos de RA recentes . . . 10
2.7 Exemplos de distintos VuMarks . . . 13

3.1 Recital de poemas aumentado usando Reality Engine . . . 15
3.2 Evolución de tormentas mediante Max Reality . . . 16
3.3 Información dun partido de tenis con Viz Visual Studio . . . 17
3.4 Seguimento dunhas eleccións con software de Brainstorm . . . 17
3.5 Ubicación de partidos de fútbol utilizando tOG-VR . . . 18

4.1 Diagrama de bloques do sistema . . . 19
4.2 Diagrama de secuencia do sistema . . . 20
4.3 Targets exemplo da aplicación . . . 24
4.4 Imaxes con puntuación de cero estrelas . . . 24
4.5 Diagrama de obxectos da primeira iteración . . . 25
4.6 Diagrama de obxectos da segunda iteración . . . 26
4.7 Diagrama de clases da segunda iteración . . . 27
4.8 Plano creado sobre os targets detectados . . . 28
4.9 Diagrama de clases da terceira iteración . . . 30
4.10 Plano no que se está a proxectar o logo da UDC . . . 32

4.11 Diagrama de obxectos da cuarta iteración . . . 33
4.12 Opcións creadas no editor de Unity . . . 34
4.13 Diagrama da clase ”HandleMenu” . . . 34
4.14 Detalle do menú despregado . . . 37
4.15 Diagrama de clases da quinta iteración . . . 38
4.16 Menú de opcións no editor actualizado . . . 39
4.17 Fiestra mostrando os targets activos . . . 39
4.18 Diagrama de clases de LangResolver . . . 41
4.19 Localización de menú . . . 42
4.20 Diagrama da clase RenderScreenTexture . . . 43
4.21 Execución do sistema coa tv . . . 44
4.22 Diagrama da clase TVWindow . . . 44
4.23 Fiestra agrandada . . . 46
4.24 Diagrama da clase NetworkSender . . . 47
4.25 Diagrama da clase NetworkReceiver . . . 48
4.26 Diferenza entre aplicación de administración e de cliente . . . 48
4.27 Diagrama da clase ProjectionScheduled . . . 50
4.28 Diagrama da clase HandlePlane actualizado . . . 51
4.29 Programación de contidos multimedia . . . 52
4.30 Diagrama de clases final . . . 53

A.1 Importación dun paquete en Unity . . . 62
A.2 Fiestra de importación do SDK de Vuforia . . . 62

B.1 Targets iniciais do sistema . . . 63
B.2 Proceso para comprobar a detectabilidade dunha imaxe . . . 65
B.3 Visualización inicial do sistema . . . 66
B.4 Proceso para proxectar un elemento multimedia . . . 67
B.5 Proceso para cargar un novo target . . . 68
B.6 Proceso para ver os targets activos . . . 68
B.7 Proceso para previsualizar a emisión televisiva . . . 69
B.8 Proceso para programar o contido aumentado . . . 70
B.9 Exemplo de execución das aplicacións de administración e cliente . . . 71

Índice de Táboas

4.1 Valores posibles de StatusInfo 28
4.2 Formatos de imaxe e vídeo soportados en Unity 29

B.1 Formatos de imaxe e vídeo soportados polo sistema 66

Capítulo 1

Introdución

O presente documento ten como finalidade documentar todo o proceso ao redor do desenvolvemento dunha aplicación cuxa finalidade e proxectar elementos dixitais nunha escena real capturada cunha cámara. Esta situación na que a realidade recibe un tratamento dixital para mostrar uns elementos adicionais é o que se coñece como realidade aumentada.

1.1 Orixe do proxecto

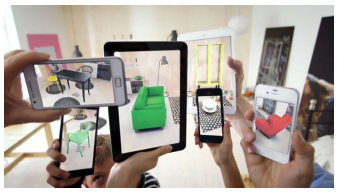
Na actualidade hai diversas tecnoloxías que contan con grandes desenvolvementos, como a intelixencia artificial, a robótica ou a que nos interesa neste proxecto, a realidade aumentada.

A realidade aumentada (RA) non é unha tecnoloxía nova, a súa orixe data de fai unhas décadas, pero é no mundo actual cando acadou unha gran popularidade. Sen lugar a dúbidas o principal motivo da divulgación desta tecnoloxía reside nos smartphones, eses dispositivos que duns anos a esta parte pasaron a formar parte da nosa vida diaria. Inicialmente os dispositivos necesarios para visualizar obxectos dixitais na realidade eran como pouco, aparatosos, e hoxe en día cun simple aparello que portamos no peto alcanzamos unha experiencia moito máis refinada.

Por suposto, non se trata só dos dispositivos. Se pensamos no mundo do cine de ciencia ficción, moitas grandes películas mostraban que o futuro contaría con tecnoloxías que permitirían visualizar elementos non presentes no mundo real e que proporcionan algún tipo de información adicional. Eses elementos do cine sempre contaron cun gran sentimento de desexo por parte do público, e hoxe en día xa somos capaces de acercarnos moito a esas situacións, non só grazas aos smartphones senón tamén a desenvolvementos que teñen lugar na actualidade en torno a lentes.

Esta tecnoloxía conta con infinidade de aplicacións no día a día da sociedade, como poder visualizar as veas dun paciente antes de realizar unha inxección ou probar como quedarían uns mobles en casa antes de compralos (ver figura 1.1).

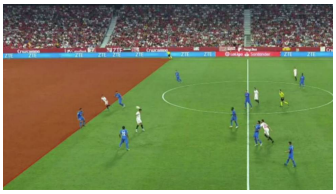
(a)



(b)

Figura 1.1: Exemplos de RA

Non obstante, o foco da nosa atención neste proxecto encóntrase noutro ámbito da nosa vida diaria, a televisión. A nivel teórico a realidade aumentada non é máis que a modificación dunha escena real mediante a inclusión de elementos dixitais. Con iso en mente podemos dicir que cando vemos a liña de fora de xogo na imaxe nunha repetición dun partido de fútbol (figura 1.2a) ou os nomes dos pilotos nunha carreira automobilística (figura 1.2b), estamos ante exemplos de realidade aumentada.



(a) Fora de xogo

(b) Pilotos na imaxe

Figura 1.2: Exemplos de RA en televisión

Do mesmo modo, xa fai un tempo que se utiliza a RA nos propios directos dos programas televisivos. Polo xeral os programas que fan uso da realidade aumentada xerada en directo son de corte informativo ou de actualidade, coa finalidade de ilustrar os datos que se estean a discutir nese preciso intre. Na figura 1.3 pódense observar exemplos desta realidade aumentada.

(a) Explicación das consecuencias dunha inundación

(b) Visualización dun resultados electorais

Figura 1.3: Exemplos de RA en directo

A partires desta realidade aumentada xerada durante os directos ideouse a realización

dunha ferramenta que permita introducir elementos multimedia (imaxe ou vídeo) intercambiables en tempo real, nun espazo xerado dixitalmente sobre unha escena real. Desta forma un técnico dun programa de televisión ou o realizador poderían engadir estes elementos fácilmente.

1.2 Contexto do proxecto

O proxecto que imos desenvolver enmárcase dentro do desenvolvemento dun proxecto de maior envergadura. O sistema completo usarase na sala de dirección dun programa televisivo para poder introducir elementos multimedia intercambiables na imaxe capturada polas cámaras en tempo real para a súa posterior retransmisión polas canles tradicionais, de maneira que a xente nas súas casas recibe a través dos seus televisores a imaxe modificada. A elección dos elementos multimedia pode ser múltiple, é dicir, que non todos os receptores da imaxe modificada verán os mismos elementos dixitais.

Isto ten unha gran aplicación de cara ao mundo da publicidade, posto que sería posible a utilización do sistema para realizar campañas publicitarias localizadas. Desta forma, unha empresa podería pagar por mostrar a súa publicidade nunha rexión determinada mentres que outra organización estaría a mostrar unha campaña distinta noutros lugares onde tamén se esté a recibir a señal televisiva do programa.

A introdución dos elementos dixitais na imaxe farase a través do recoñecemento dunhas imaxes coa axuda dun motor de realidade aumentada, que xerarán un espazo no que poder proxectar o elemento publicitario desexado polo encargado de uso do sistema.

Este gran proxecto pode fragmentarse en desenvolvementos máis pequenos e asequibles, que actúen como módulos separados que realicen as distintas tarefas. Dacordo cos obxectivos deste gran proxecto diferenciamos tres módulos:

- Un primeiro módulo encargarase da recepción da imaxe capturada polas cámaras, así como da creación do espazo de proxección no que emitir os elementos multimedia que poden ser intercambiados durante a propia execución do sistema.

- O segundo módulo será o encargado de realizar unha discriminación de contidos a enviar en caso de que a emisión non sexa igual para todos os receptores do programa.

- O último módulo será o responsable de convertir a nova imaxe xerada a través da realidade aumentada nunha señal que pode ser enviada polas canles tradicionais de televisión.

No presente proxecto recollemos a realización do primeiro módulo deste proxecto.

1.3 Obxectivos do proxecto

O obxectivo deste proxecto consiste en desenvolver unha aplicación que permita engadir contido multimedia sobre unha escena real, capturada por unha cámara, facendo uso da realidade aumentada.

Esta aplicación recoñecerá a presenza dun par de targets (imaxes detectables por un sistema de realidade aumentada) prefixados e creará entre eles un espazo sobre o que proxectar un elemento multimedia (un vídeo ou unha imaxe).

Partirase da captura da realidade mediante unha cámara e detectaranse os targets presentes na escena. Unha vez detectados, o usuario da aplicación poderá seleccionar que elemento multimedia quere proxectar. O sistema realizará un seguimento da proxección a través dos targets de maneira que se a cámara deixa de capturar o espazo de proxección, tanto parcial como totalmente, a proxección mostrada corresponderase coa parte do espazo que aínda estea a ser capturada pola cámara. O elemento multimedia poderá ser modificado en tempo real (cambiar un vídeo por outro, etc.).

Posteriormente engadiremoslle máis funcionalidades ao sistema para facelo máis completo. En primeiro lugar permitiremos modificar os targets dos que se realiza o seguimento por aqueles que aos usuarios do sistema lles interese. Do mesmo modo, para que poidan levar un control de que targets teñen seleccionados crearemos unha interface gráfica na que poidan visualizalos. Continuaremos coa localización da aplicación a outras linguaxes, de forma que usuarios que non entendan o castelán poidan tamén facer uso da ferramenta desenvolta.

Para que os usuarios poidan ter unha previsualización de como sería a emisión televisiva do vídeo aumentado desenvolveremos unha fiestra de tamaño variable que o usuario poderá ver o resultado tal cal o verían os televidentes. Continuando con esta idea crearemos unha aplicación cliente externa que unha vez aberta mostrará ese mismo contido aumentado, desta forma xa se podería iniciar a retransmisión masiva cara as televisións domésticas mediante a retransmisión da imaxe desta aplicación.

Finalmente traballaremos por incluir unha opción para programar a execución do contido multimedia. Desta forma, xa non sería necesario trastear co sistema durante o directo televisivo, bastaría con ter indicado anteriormente que elemento se desexa retransmitir, cando iniciar a sua emisión e cando rematala.

1.4 Organización do proxecto

Este proxecto conta con dous elementos principais, a memoria e a aplicación.

A memoria é o presente documento. Nela recóllese toda a documentación xerada durante a creación da ferramenta, xa sexan coñecementos teóricos ou a análise do desenvolvemento

da aplicación. Para facilitar a navegación, no documento existen unha serie de apartados que definimos a continuación:

- Introdución. O presente apartado. Aquí buscamos presentarlle ao lector os distintos elementos cos que se vai encontrar no resto do documento.
- Fundamentos. Profundízase nos distintos conceptos teóricos nos que se basea o proxecto. Do mesmo modo explícanse as ferramentas que se van utilizar no desenvolvemento da aplicación.
- Estado da cuestión. Recóllense alternativas ao sistema que se vai desenvolver.
- Sistema. Detalla as distintas iteracións a seguir para acadar os obxectivos do proxecto.
- Conclusións. Examínase o resultado final da aplicación desenvolvida en función dos obxectivos que se buscaban inicialmente. Do mesmo modo, coméntanse as posibles liñas futuras de traballo.
- Anexos. Engloba os distintos manuais para poder instalar e usar a ferramenta.

A aplicación, por outro lado, é o elemento software que se desenvolve de maneira concorrente á memoria. Para a súa creación utilízase Unity, un coñecido motor gráfico de desenvolvemento de videoxogos. Esta ferramenta é compatible cun SDK dedicado a traballar con realidade aumentada, Vuforia. Ambos elementos software son gratuitos.

Capítulo 2

Fundamentos

Neste apartado vanse explicar en profundidade aqueles termos teóricos e ferramentas sobre as que se constrúe o presente proxecto.

2.1 Realidade Aumentada

A RA [1] é unha tecnoloxía que combina o mundo real e o dixital de maneira que un ou varios usuarios poidan visualizar información xerada de maneira dixital sobre elementos presentes na realidade. Precisamente, a presenza da realidade é o que diferencia a realidade aumentada da realidade virtual, na que se crea un entorno completamente dixital separado do mundo real.

Na actualidade esta tecnoloxía goza de gran saúde grazas a que os smartphones son capaces de executar aplicacións de RA, facendo que obxectos dixitais sexan proxectados na pantalla do dispositivo cando o usuario apunta a determinados elementos da súa contorna.

2.1.1 Historia da realidade aumentada

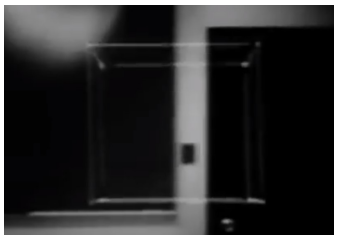
Para encontrar o primeiro intento de crear un dispositivo que fora capaz de proxectar elementos dixitais sobre o entorno real é necesario retroceder ata 1968, cando Ivan Sutherland, un recoñecido informático e considerado o pai da computación gráfica, presentou o dispositivo coñecido como Espada de Damocles [2]. Este prototipo de visor de RA proxectaba estruturas tridimensionais simples (ver figura 2.1).

No ano 1975, Myron Krueger, doutor en ciencias da computación, ideou Videoplace, un sistema que permitía aos usuarios presentes nunha habitación interactuar con elementos dixitais a través da súa sombra proxectada nunha pantalla (ver figura 2.2).

Na década dos 80 encontramos dous grandes experimentos relacionados coa realidade aumentada. No ano 1980 Steve Mann presentou EyeTap, un dispositivo monocular deseñado para levar fronte ao ollo no que se proxecta a imaxe dun equipo remoto (ver figura 2.3). Anos

máis tarde, en 1987, Douglas George e Robert Morris desenvolveron un sistema de visualización que permitía mostrar información astronómica sobre o ceo.

(a) Casco de visualización



(b) Exemplo de estrutura proxectada

Figura 2.1: Espada de Damocles

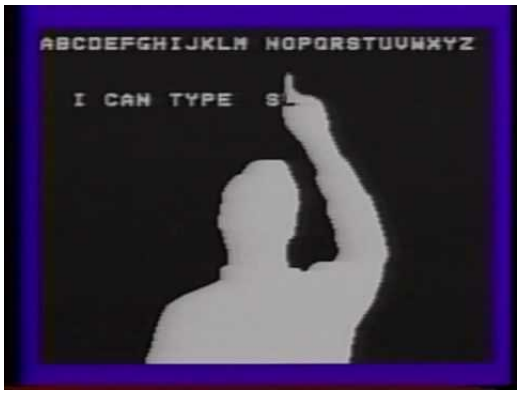


Figura 2.2: Escritura dun texto mediante Videoplace

Figura 2.3: Primeiro prototipo de EyeTap

Posteriormente, nos anos 90 Thomas Caudell e David Mizell acuñaron o termo ”realidade

aumentada" por primeira vez [3]. No 1992 Louis Rosenberg creou Virtual Fixtures (figura 2.4), unha plataforma pioneira nos campos da realidade aumentada e realidade virtual que tiña como obxectivo mellorar o rendemento en tareas feitas de maneira remota [4]. No 99, un equipo de investigadores liderado por Frank Delgado e Michael Abernathy desenvolveron un software de navegación que proporcionaba información visual adicional a pilotos [5]. Nese mesmo ano, Hirokazu Kato desenvolveu ARToolKit, a primeira libraría de realidade aumentada con software libre, aínda existente na actualidade.

Figura 2.4: Equipamento da plataforma Virtual Fixtures

A partires do ano 2000, o desenvolvemento de sistemas e aplicacións de RA foise acercando cara o gran público. No ano 2003, Wagner e Schmalstieg presentaron o primeiro sistema de realidade aumentada que corría nun asistente persoal [6], un claro precursor dos actuais sistemas que incorporan os smartphones (ver figura 2.5).

No 2013 Google presentou Google Glass (ver figura 2.6a), un sistema de gafas de realidade aumentada que xurdía co obxectivo de formar parte da vida diaria da xente. Dous anos despois, Microsoft presentou as súas propias gafas de realidade aumentada, HoloLens (figura 2.6b), cuxa tecnoloxía de detección e seguimento de obxectos baseouse en Kinect, un controlador de videoxogos desenvolto orixinalmente para a consola Xbox 360, tamén propiedade de Microsoft. En 2016 unha compañía de desenvolvemento de videoxogos estadounidense, Niantic, lanzou Pokemon Go (figura 2.6c), un xogo de realidade aumentada para dispositivos móbiles que ata a presente data é a aplicación de entretemento de RA máis exitosa da historia.

Figura 2.5: Detalle do sistema desenvolto por Wagner e Schmalstieg

(a) Google Glass

(b) Gafas HoloLens de Microsoft

(c) Pokemon Go

Figura 2.6: Exemplos de RA recentes

2.1.2 Tipos de realidade aumentada

En xeral os sistemas de realidade aumentada pódense agrupar en dúas grandes categorias, a RA baseada en marcadores e a RA sen marcadores.

A realidade aumentada baseada en marcadores parte de símbolos impresos ou imaxes sobre os cales se superpoñen os elementos dixitais. O elemento virtual aparece cando o dispositivo recoñece o marcador, do cal mantén un seguimento sobre a súa posición, de maneira que se o marcador deixa de ser visible, a proxección tamén desaparecerá. Este tipo de realidade aumentada é moi popular debido á gran facilidade de detección e recoñecemento coa que contan os dispositivos móbiles actuais, así como a súa presenza constante na vida diaria da xente. Esta realidade aumentada é a que se utiliza no desenvolvemento da aplicación

obxectivo deste proxecto.

A realidade aumentada sen marcadores non require recoñecer imaxes, senón que obtén información posicional de distintos sistemas do dispositivo, como a cámara ou o GPS. Esta información permite modelar un espacio tridimensional mediante un proceso coñecido como SLAM. Desta forma pódese introducir un obxecto dixital na escena real e mantelo no entorno. As librarías ARKit e ARCore, propiedade de Apple e Google respectivamente, traballan con esta modalidade de realidade aumentada.

Existen variacións da RA sen marcadores:

- RA baseada na localización. Este tipo de realidade aumentada ancora a experiencia aumentada a un sitio específico. A proxección dixital mapéase a lugares específicos e móstrase cando a información obtida do dispositivo coincide coa da localización do elemento virtual. O videoxogo Pokemon Go fai uso deste tipo de realidade aumentada.

- RA baseada na proxección. Nesta realidade aumentada, o sistema proxecta luz nunha superficie do mundo real e detecta a interacción humana con esa proxección. A detección realízase diferenciando entre unha proxección esperada e a proxección modificada pola interacción do usuario.

2.2 Vuforia

Vuforia [7] é un SDK utilizado no desenvolvemento de aplicacións que fan uso da realidade aumentada. Qualcomm foi o seu creador e inicialmente só podía recoñecer texto, pero no ano 2015 a compañía PTC Inc. adquiriu a súa propiedade. O SDK está programado en C++ e conta con APIs en varias plataformas. A API en IOS encóntrase en C++, en Android é en Java e en Unity, C#. No ano 2018 incluíuse unha API para UWP que fai uso de C++.

Para facer uso do SDK é necesario contar cunha licenza. Vuforia distribúe unha licenza con fins educativos de maneira gratuita que conta con gran parte das funcionalidades do SDK e certas limitacións, como unha marca de auga na pantalla durante a execución das aplicacións.

En caso de querer facer un uso comercial de Vuforia existen 3 plans de distribución:

- Basic. Ten un prezo 504 dólares anuais e permite o acceso ao Ground Plane, Image Targets, VuMarks e Model Targets.

- Basic+Cloud. Cun costo de 99 dólares ao mes inclúe as características do plan Basic e engade o acceso a bases de datos de imaxes na nube.

- Pro. É un plan sen limitacións no que o precio ten que ser especificado por PTC Inc.

2.2.1 Funcionalidades principais de Vuforia

Podemos dividir as funcionalidades de Vuforia en tres partes:

- Recoñecemento de imaxes.
- Recoñecemento de obxectos.
- Outras funcionalidades.

Recoñecemento de imaxes

Unha das principales funcionalidades do SDK é o recoñecemento de imaxes en tempo real. Estas imaxes coñécense como Image Targets e requiren dunha preparación para poder ser utilizadas. Os formatos admitidos son PNG e JPG e o tamaño non debe exceder os 2 Mb. As imaxes deben contar coa maior cantidades de detalles posibles para facilitar o seu recoñecemento. Unha vez que temos escollida a imaxe que queremos usar subímola a unha plataforma online de Vuforia coñecida como Vuforia Target Manager.

Despois de cargar a imaxe, Vuforia evalúa a imaxe en función da cantidade de características que encontra e otórgalle unha puntuación de 1 a 5 estrelas. A maior cantidade de estrelas, mellor recoñecemento.

Vuforia permite dúas maneiras de almacenar estes targets: de maneira local ou na nube. Se optamos por facelo en local descargamos a base de datos creada e cargámola na aplicación que desenvolvamos. Se fose necesario engadir unha nova imaxe será necesario volver a descargar toda a base de datos e volver a montala na aplicación. Se escollemos facer uso da nube non será necesario descargar a BD, o dispositivo conectarase a ela a traves de internet. En caso de engadir unha nova imaxe a aplicación xa a detectará automáticamente, mentres conte con conexión á rede.

Vuforia permite detectar imaxes que non sexan planas, como as etiquetas de botellas por exemplo. Para crear estes targets é necesario indicar información adicional, como o diámetro do obxecto cilíndrico.

Neste apartado tamén entra a detección das imaxes coñecidas como VuMarks. Os VuMarks son imaxes que conteñen outra imaxe codificada no seu interior. Básicamente é un código QR creado polos desenvolvedores de Vuforia. Estas imaxes son capaces de encodear texto, números ou bytes no seu interior. Como podemos apreciar na figura 2.7 estas imaxes poden ter deseños moi diversos, afastándose dos tradicionais códigos QR.

Recoñecemento de obxectos

Outra característica de Vuforia é a identificación de obxectos 3D en tempo real, coñecidos como Model Targets.

Figura 2.7: Exemplos de distintos VuMarks

Para o recoñecemento do obxecto hai que cargar no Target Manager un modelo CAD 3D ou un escaneamento 3D dun obxecto.

Esta tecnoloxía é moito máis complexa que a detección de imaxes. Para facilitar o recoñecemento dos obxectos é necesario seguir unha serie de pautas.

- O obxecto debe manterse estático. Isto non impide que o usuario poida moverse pola súa contorna e observarlo dende distíntos ángulos.
- Non debe ser un obxecto monótono. É dicir, se ten cor ou conta con patróns identificables, será máis fácil que Vuforia o identifique.
- O obxecto ten que ter complexidade. Canto menos simple sexa un obxecto, máis recoñecible será.
- O modelo debe ser ríxido. Se fixésemos uso dun modelo flexible, o modelo orixinal e o obxecto real poderían contar con moitas diferenzas.
- O modelo debería ser o máis opaco posible. É moi difícil para Vuforia recoñecer obxectos transparentes ou brillantes.

Outras funcionalidades

O Extended Tracking é unha tecnoloxía que permite seguir un target aínda que este xa no se encontre na imaxe capturada pola cámara. Vuforia conta cunha opción que de estar activada, permite realizar este seguimento de forma automática.

O Ground Plane permite detectar superficies horizontais e posicionar o contido dixital ancorandoo a uns puntos. Esta tecnoloxía usa a imaxe da cámara para analizar e mapear o espazo ao seu redor.

2.3 Unity

Unity [8] é un motor de desenvolvemento de videoxogos creado por Unity Technologies. A primeira versión foi lanzada en 2005 co obxectivo de facer accesible o desenvolvemento de videoxogos entre os desenvolvedores. O editor ten versións completamente funcionais para os sistemas operativos Windows e MacOS, así como unha versión experimental para Linux. Do mesmo modo, o motor soporta o desenvolvemento para unha gran multitude de plataformas, incluindo ordenadores, móbiles, consolas e realidade virtual e aumentada.

O motor conta cunha densa documentación online xestionada polos seus desenvolvedores. Esta documentación abarca dende o funcionamento das APIs ata unha gran colección de titoriais para iniciarse no desenvolvemento de xogos en xeral o explicar caracteristicas do motor en particular. Aparte da documentación xestionada polos creadores, a popularidade do motor na comunidade dos desenvolvedores de videoxogos fixo aflorar multitude de titoriais e documentación adicional creados polos membros da comunidade.

Dacordo coa páxina oficial de Unity, o 50% dos videoxogos do mundo están desenvoltos neste motor.

Unha das maiores ferramentas do motor é a Unity Asset Store, unha tenda virtual integrada no propio editor na que se poden obter funcionalidades implementadas ou obxectos creados. Os contidos da tenda poden ser gratuitos ou de pago. Desta forma, aqueles desenvolvedores que traballan por libre poden obter unha fonte de ingresos mediante a venta do seu traballo.

O desenvolvemento de aplicacións no motor xira en torno ao sistema de compoñentes. Estes compoñentes engádense aos obxectos para darlles funcionalidades. Existe unha gran variedade de compoñentes xa creados para permitir a interacción dos obxectos, pero tamén e posible añadir máis funcionalidades mediante o uso de scripts. Na actualidade Unity soporta C# como linguaxe de programación aínda que anteriormente foi compatible con Boo e cunha versión de JavaScript denominada UnityScript.

Unity conta cunha licenza gratuita e completamente funcional sempre e cando aquelas aplicacións desenvoltas co motor non superen unhas ganancias de 200000 dólares anuais. A partir dese momento é necesario pagar unha suscripción Pro cun prezo de 1800 dólares anuais. A principal característica desta suscripción é poder acceder ao código fonte do motor e modificalo en función das necesidades do cliente. Entre ambos plans de suscripción, o gratuito e o Pro, existe outra modalidade intermedia, o plan Plus.

Capítulo 3

Estado da cuestión

No presente apartado vanse comentar alternativas existentes á ferramenta que se vai desenvolver.

Como xa se comentou na sección 1.1, a RA realizada en tempo real na televisión xa existe. Non obstante, non hai só unha aplicación que realice esta tarea e no seguinte texto imos ver outras opcións.

3.1 Reality Engine

Reality Engine [9] é unha aplicación desenvolta pola compañía Zero Density. Para a creación desta aplicación fixeron uso de Unreal Engine, un motor de videoxogos propiedade de Epic Games. A ferramenta conta con Reality Keyer, unha tecnoloxía realizada tamén por Zero Density, que permite que a inclusión de obxectos transparentes na imaxe ou detalles pequenos como o cabelo teñan un aspecto realista en pantalla. Incluimos un exemplo desta ferramenta na figura 3.1.

Figura 3.1: Recital de poemas aumentado usando Reality Engine

3.2 Max Reality

The Weather Company, unha empresa subsidiaria de IBM, é a creadora de Max Reality [10]. Esta ferramenta de realidade aumentada está deseñada especificamente para mostrar en pantalla información relacionada coa climatoloxía e co tráfico. Dacordo cun estudio estadístico realizado pola compañía, máis dun 60% da xente permanecía atenta durante máis tempo á información meteorolóxica cando se facía uso da RA, o cal pode explicar o interese no desenvolvemento deste sistema. A imaxe 3.2 mostra un exemplo do uso desta ferramenta.

Figura 3.2: Evolución de tormentas mediante Max Reality

3.3 Viz Virtual Studio

Viz Virtual Studio [11] é unha aplicación desenvolta pola empresa noruega Vizrt no seu propio motor gráfico, Viz Engine. A utilización deste sistema require que o motor realice primeiro un mapeado do escenario que será posteriormente aumentado. Pódese apreciar un exemplo de uso da ferramenta na imaxe 3.3.

3.4 Brainstorm

Brainstorm [12] é unha compañía española dedicada á introdución de elementos dixitais en imaxe que será posteriormente emitida por televisión. Contan con varios softwares distintos que permiten a renderización de elementos nunha escena mediante realidade aumentada, como InfinitySet ou eStudy. Recollemos un caso de uso do software da empresa na figura 3.4.

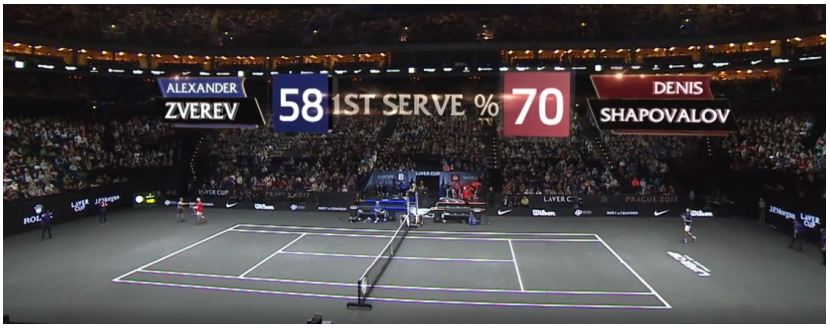


Figura 3.3: Información dun partido de tenis con Viz Visual Studio

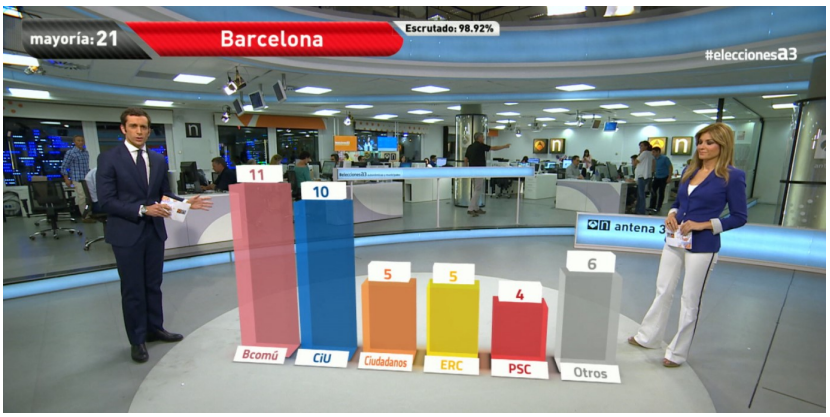


Figura 3.4: Seguimento dunhas eleccións con software de Brainstorm

3.5 tOG-VR

Aínda que o nome desta aplicación faga mención á realidade virtual, tOG-VR [13] está preparada para traballar con ambas tecnoloxías indistintamente. Esta ferramenta foi desenvolvida por RT Software partindo de Unreal Engine. Entre as televisións usuarias deste sistema encontramos á BBC. A figura 3.5 contén un exemplo de uso do sistema.

3.6 Discusión

Todos os sistemas contan cun gran abano de funcionalidades, facendo que o seu uso poida ser complexo e requira dun adestramento previo. A nosa ferramenta busca contar cunha funcionalidade que sexa simple e moi accesible para os usuarios do sistema.

Por outro lado, todos os sistemas ocultan o seu prezo na web de forma que para coñecelo é necesario contactar coa propia compañía para acordar o importe a pagar en función das necesidades de cada cliente, mentres que nós pretendemos que todas as funcionalidades sexan accesibles a todos os usuarios. Posto que Unity pode ser utilizado de maneira gratuita ata

Figura 3.5: Ubicación de partidos de fútbol utilizando tOG-VR

unhas certas ganancias, como xa se indicou na sección 2.3, e que os plans de uso comercial de Vuforia son accesibles (ver apartado 2.2) o noso sistema podería acadar un prezo de venta moi competitivo.

Capítulo 4

Sistema

Neste apartado imos profundizar en como foi o desenvolvemento da aplicación obxecto deste proxecto.

4.1 Organización

Ao longo deste capítulo imos ver os pasos necesarios para o desenvolvemento do sistema.

Recordemos que o obxectivo desta aplicación consiste en realizar un seguimento de dous targets capturados por unha cámara e xerar entre eles un espazo de proxección. Nese espazo de proxección o usuario do sistema poderá cargar un elemento multimedia da súa elección. O espazo de proxección modificará a súa posición en función da posición dos targets con respecto á cámara, o que pode desembocar en que o elemento se vexa recortado da imaxe se o espazo de proxección non se encontra na súa totalidade na escena capturada. Este elemento multimedia pode ser modificado polo usuario sen necesidade de reiniciar o programa.

Na figura 4.1 recollemos o diagrama de bloques do sistema que imos desenvolver para clarificar as ideas.

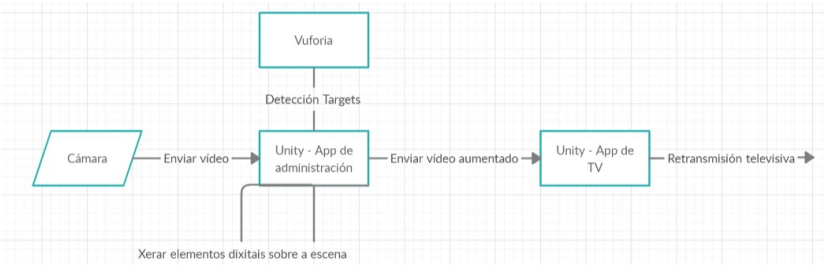


Figura 4.1: Diagrama de bloques do sistema

A idea que queremos transmitir neste gráfico é que tras a captura da escena real a través dunha cámara de vídeo, a aplicación de administración e Vuforia traballan de maneira

concurrente. Vuforia encargarase de detectar a presenza dos targets no vídeo e a aplicación aumentará a escena sobre eses targets engadindo contido dixital.

Á vez, a aplicación de administración enviaralle a escena aumentada á aplicación cliente de televisión, desde a que xa se poderá iniciar a retransmisión televisiva aos fogares da xente.

A continuación, na imaxe 4.2 mostramos o diagrama de secuencia da aplicación para detallar máis esta comunicación entre os elementos que acabamos de describir.

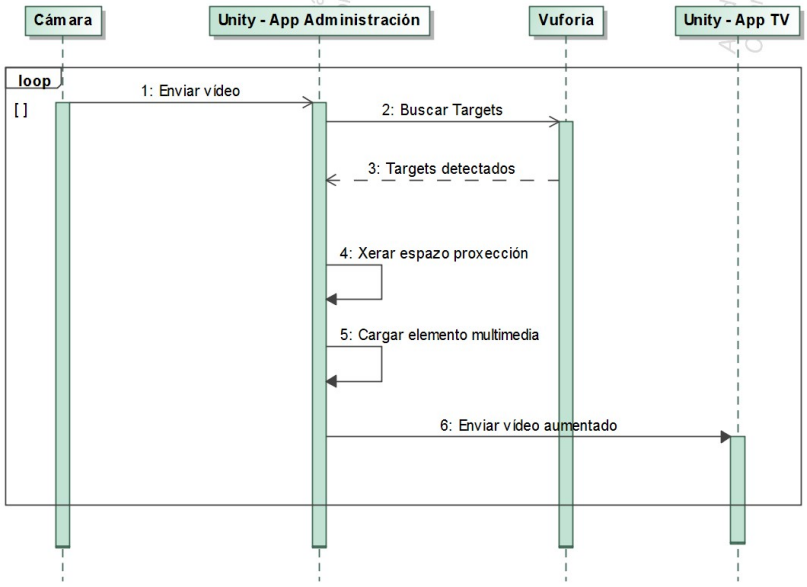


Figura 4.2: Diagrama de secuencia do sistema

Na imaxe podemos observar como é a comunicación entre os distintos elementos que interveñen no programa. Inicialmente, e ao longo de todo o programa, unha cámara estará a capturar unha escena real e enviaralle a imaxe a Unity.

Unity esperará a que Vuforia detecte os targets, como podemos observar nas mensaxes 2 e 3 da imaxe. No apartado 4.3.1 recollemos o proceso necesario para crear uns targets, no que será a primeira iteración do desenvolvemento. Do mesmo modo explicamos os requisitos que deben cumprir os targets para o seu correcto funcionamento.

Na segunda iteración (apartado 4.3.2) centrarémonos na xeración do espazo de proxección mentres os targets sexan detectados, como indica a mensaxe 4 da imaxe.

A continuación, como se recolle na mensaxe 5 do diagrama, o motor cargará aquel elemento multimedia que o usuario escolla. Recollemos o proceso necesario na terceira iteración do desenvolvemento (apartado 4.3.3).

Adicionalmente, incluimos iteracións adicionais (apartados 4.3.4, 4.3.5, 4.3.6, 4.3.7) nas que detallamos a inclusión doutras funcionalidades. En concreto, a xeración de novos targets por parte do usuario en tempo de execución, a visualización de qué targets estanse a usar para a

detección en cada momento, a localización do sistema a outras linguaxes, a previsualización da emisión do contido aumentado nunha fiestra adicional modificable polo usuario.

Continuando coa previsualización do contido, no apartado 4.3.8 crearemos unha aplicación cliente, dependente da aplicación de administración que imos desenvolver, na que só se visualizará a imaxe aumentada resultante do paso do vídeo pola aplicación de administración. Desta forma, esta aplicación cliente podería usarse como base para a retransmisión masiva cara os televidentes. Este sistema adicional aparece na figura 4.2 como o obxecto Unity - App TV recibindo a mensaxe 6.

Finalmente, no apartado 4.3.9 introduciremos na aplicación de administración a posibilidade de programar con antelación a emisión dos elementos multimedia. Desta forma non sería necesario interactuar coa aplicación durante o directo.

4.2 Tecnoloxías utilizadas

Para o desenvolvemento do sistema imos facer uso de dúas tecnoloxias corporativas, Unity e Vuforia.

A elección de Unity como sistema no que construir a aplicación ven dada principalmente pola súa compatibilidade coa realidade aumentada e pola inxente cantidade de documentación existente, tanto profesional como aficionada. Non obstante, tamén se tivo en conta que moitos desenvolvedores utilizan este motor para realizar aplicacións non relacionadas cos videoxogos, demostrando por tanto que Unity é versatil; así como que a curva de aprendizaxe deste motor en concreto está considerada máis asequible que a de outros sistemas similares, como Unreal Engine. Do mesmo modo, non hai que olvidar que aínda que usamos o motor na súa modalidade gratuita, todas as funcionalidades que necesitamos están desbloqueadas.

Unity non conta de maneira gratuita cun motor que permita a utilización de elementos de realidade aumentada. Por tanto é necesario buscar unha ferramenta externa coa que poder detectar un par de targets. Aquí é onde entra en xogo Vuforia. Este SDK conta con total compatibilidade con Unity e na súa versión gratuita permite facer un seguemento de targets, que é a única parte que necesitamos do motor de RA. Ao igual que con Unity, moitos desenvolvedores fan uso de Vuforia e a documentación sobre a súa utilización é moi extensa.

Co uso de Vuforia solucionamos o problema do recoñecemento de targets, pero ao longo deste texto imos explicar como a través do uso da programación podemos crear un espazo de proxección no que mostrar tanto imaxes como vídeos seleccionados durante a execución do programa e como facer que os targets dos que o sistema realiza un seguemento sexan modificados segundo as necesidades do usuario sen necesidade de deter o funcionamento do sistema.

4.3 Iteracións

Para levar a cabo o desenvolvemento vaise utilizar o método iterativo incremental.

Esta metodoloxía dicta que en cada nova iteración do desenvolvemento se repita un mesmo proceso de traballo para acadar unha nova funcionalidade, á vez que se melloran as xa existentes.

Dacordo con esta definición en cada nova iteración do desenvolvemento imos realizar unha análise do obxectivo que imos alcanzar, un deseño que ilustre como poder levar a cabo ese obxectivo e a implementación necesaria para acadalo.

Como xa se indicou na sección 1.2, este proxecto forma parte dun desenvolvemento máis grande dun sistema capaz de recibir, aumentar e retransmitir por televisión unha escena capturada por unha cámara en directo. Esta imaxe aumentada conterá ou ben unha imaxe estática, como pode ser o logo dunha empresa; ou cun vídeo, como por exemplo un anuncio promocional. Do mesmo modo, definimos que o proxecto conta cunha serie de módulos dos cales imos centrarnos no primeiro, que engloba a recepción da imaxe das cámaras e a introducción dos elementos multimedia seleccionables polo usuario nun espazo de proxección xerado entre untre uns targets presentes na escena.

Dacordo con este obxectivo e segundo a separación en tarefas que definimos no apartado 4.1, englobamos o desenvolvemento do sistema nas seguintes iteracións:

1. Na primeira iteración (apartado 4.3.1) inicializaremos o proxecto e crearemos un par de targets para realizar o seguimento. Tamén incluímos unha explicación dos requisitos que deben cumplir os targets para ter un funcionamento óptimo do sistema.

2. Continuando na segunda iteración (apartado 4.3.2), crearemos un espazo de proxección en base aos targets detectados polo sistema.

3. No apartado 4.3.3, correspondente á terceira iteración faremos que o espazo de proxección sexa interactuable co usuario, de forma que éste poida escoller unha imaxe estática ou un vídeo para emitilos nese espazo.

4. Ao longo da cuarta iteración (apartado 4.3.4) crearemos un menú de opcións para o usuario no que incluiremos a opción de cambiar calqueira dos dous targets dos que se está a realizar o seguemento por outro elixido polo usuario.

5. Na quinta iteración (apartado 4.3.5) engadiremos unha nova opción ao menú para que o usuario poida levar o control de cales son os targets que a aplicación trata de recoñecer na escena.

6. Durante a sexta iteración (apartado 4.3.6) incluiremos a localización da ferramenta a outras linguaxes en función da lingua que esté a utilizar o sistema do usuario.

7. Na séptima iteración (apartado 4.3.7) crearemos unha fiestra interactiva na que o usuario poderá previsualizar como está a ser a emisión aumentada de cara aos televidentes.

8. Ao longo da oitava iteración (apartado 4.3.8) desenvolveremos unha aplicación cliente adicional que recibirá o vídeo aumentado desde a aplicación orixinal. Será esta nova aplicación, na que non se visualizarán elementos de interface de usuario, a que se poida usar para a retransmisión cara as televisións.

9. Finalmente, na novena iteración (apartado 4.3.9) desenvolveremos a capacidade de poder planificar con antelación a emisión dos contidos multimedia.

4.3.1 Primeira iteración

Análise

Nesta primeira iteración do desenvolvemento imos centrarnos en que o sistema sexa capaz de recoñecer dous targets simultaneamente.

Os targets son obxectos que encapsulan imaxes de maneira que un motor de realidade aumentada poida recoñecelas. O motor extrae unhas características das imaxes que almacena no seu interior e cando detecta nunha escena real esas características empeza a realizar un seguimento desa rexión detectada.

Dacordo coa documentación de Vuforia as imaxes que encapsularemos nun target deben cumprir unha serie de requisitos:

- O formato das imaxes debe ser JPG ou PNG.
- O peso das imaxes non pode superar os 2.25 megabytes.
- As imaxes deben poseer unha anchura mínima de 320 píxels.

Por outro lado, para que as imaxes sexan recoñecibles deben seguir unhas recomendacións:

- Os obxectos da imaxe deben conter moitas esquinas. No campo da visión artificial a detección de esquinas é unha das formas máis sinxelas de detectar obxectos nunha imaxe, por tanto necesitamos que os elementos presentes na nosa imaxe conten con moitas esquinas para facilitar o proceso de detección. Cabe destacar que os obxectos redondos non teñen esquinas, polo tanto dificultan moito o proceso.
- O contraste entre os distintos elementos da imaxe debe ser o maior posible. Se o contraste entre elementos é moi baixo o sistema pode non detectar cando comeza un elemento e cando termina, de forma que a detección se vexa imposibilitada.

- Evitar imaxes borrosas ou cun exceso de iluminación. Estas imperfeccións da imaxe poden provocar que non se recoñezan suficientes características nas imaxes e por tanto a súa detección no mundo real pode verse comprometida.

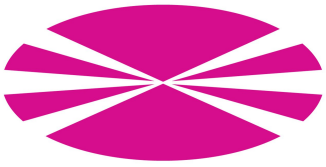
Seguindo estas indicacións buscamos unhas imaxes que cumprisen os requisitos antes mencionados. Para a decisión de que imaxes utilizar para facer un recoñecemento e seguimento fixemos uso da ferramenta web Target Manager que proporciona Vuforia, e da que xa falamos no apartado 2.2. Despois de facer un pool de quince imaxes e visualizar que puntuación obtiñan na ferramenta, decidimos usar como targets de exemplo no desenvolvemento da aplicación as imaxes recollidas na figura 4.3, que obtiveron unha puntuación de cinco estrelas cada unha. Non obstante, esta decisión está tomada en torno á puntuación na web e non implica que só se poidan utilizar estes targets.

(a)

(b)

Figura 4.3: Targets exemplo da aplicación

Neste pool de imaxes tamén encontramos imaxes que non deberían ser usadas para o recoñecemento posto que obtiveron cero estrelas, recollemos dous exemplos na figura 4.4.



(a)

(b)

Figura 4.4: Imaxes con puntuación de cero estrelas

Máis adiante, no apartado 4.3.4 veremos como o usuario poderá cambiar os targets durante a execución do sistema e a detección seguirá funcionando.

Do mesmo modo o sistema será invariable á posición dos targets un respecto doutro, é dicir, ambos van xerar un espazo entre eles sen importar cal esté á esquerda e cal á dereita.

Deseño

Para poder cumplir co noso obxectivo nesta iteración será necesario crear dous targets aos que lles asignaremos as imaxes desexadas. Desta forma crearemos un obxecto Target1 que realizará un seguimento da imaxe 4.3a e outro obxecto Target2 que se encargará de encapsular a imaxe 4.3b. O diagrama da figura 4.5 recolle unha representación gráfica do que imos realizar na implementación.

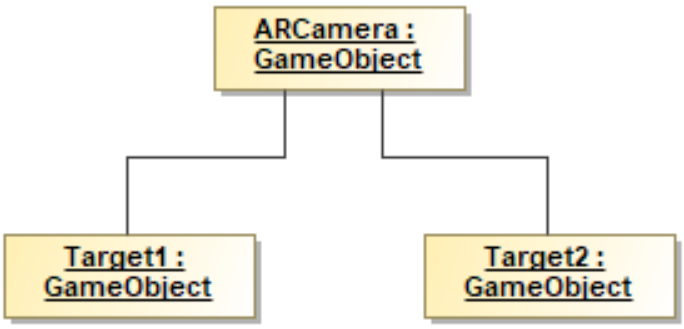


Figura 4.5: Diagrama de obxectos da primeira iteración

Implementación

En primeiro lugar é necesario crear un proxecto novo en Unity. Unha vez que o editor permita comezar a traballar hai que importar os assets que permiten facer uso do SDK de Vuforia en Unity. Unha vez que esta primeira inicialización do proxecto esté feita xa é posible comezar o desenvolvemento propio desta iteración.

Comezamos eliminando a cámara e a luz direccional que o proxecto ten creadas por defecto. A continuación creamos unha nova cámara de realidade aumentada e accedemos á configuración de Vuforia para realizar as seguintes modificacións:

- No apartado Max Simultaneous Tracked Images cambiamos o valor a 2. Desta forma a aplicación realizará un seguimento de dúas imaxes.

- Engadimos a base de datos na que se encontran as imaxes que usaremos para a detección. Esta base de datos créase na páxina de desenvolvedores de Vuforia (`https://developer.vuforia.com/`).

Unha vez que os anteriores cambios estean realizados xa podemos proceder á creación dos dous targets. Imos xerar unha xerarquía na escena facendo que estes targets sexan fillos

da cámara. Como xa definimos no apartado 4.3.1, un dos targets chamarase Target1 e conterá a imaxe 4.3a e o outro será Target2 coa imaxe 4.3b.

Chegados a este punto rematamos a primeira iteración. Se executamos a aplicación poderemos observar no editor como se apuntamos coa cámara a algún dos targets, ou aos dos á vez, o programa realizará un seguimento da súa posición.

4.3.2 Segunda iteración

Análise

O obxectivo desta segunda iteración vai ser a creación dun espazo de proxección que se localice entre os dous targets que xa xeramos no apartado 4.3.1.

Deseño

Para ter un espazo de proxección será necesario crear un novo obxecto cunha morfoloxía que lle permita funcionar como unha pantalla. Do mesmo modo, este obxecto debe ser capaz de modificar a súa posición en función da localización dos targets. Recollemos un diagrama de obxectos actualizado na figura 4.6.

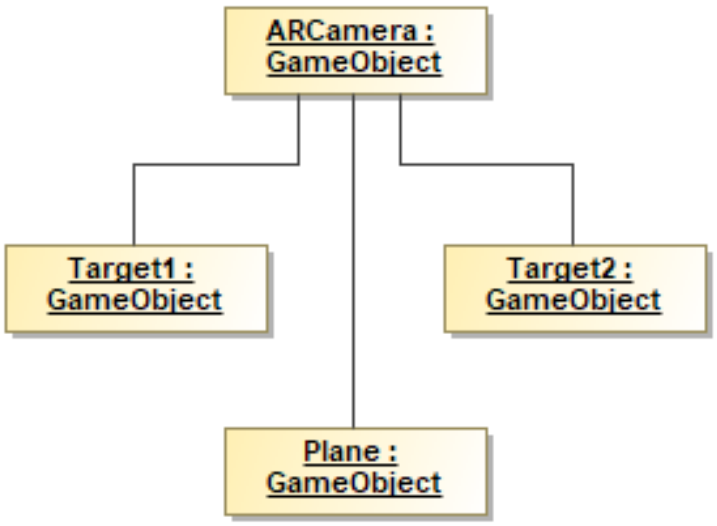


Figura 4.6: Diagrama de obxectos da segunda iteración

Implementación

Para acometer este obxectivo desenvolvemos un script ao que chamaremos ”CreatePlane” e que llo asignamos á cámara como un compoñente máis. Este script encargarase tanto da

creación do espazo no que posteriormente introduciremos elementos multimedia, como de posicionalo correctamente na escena capturada pola cámara.

Neste script hay dúas funcións básicas que necesitamos coñecer. Unha é a función ”Start()” e a outra é ”Update()”. A primeira delas execútase unha única vez en canto o script pasa a estar dispoñible. A outra execútase unha vez por fotograma mentres dure a execución da aplicación e o script estea habilitado. A figura 4.7 recolle un diagrama de clases con este script.

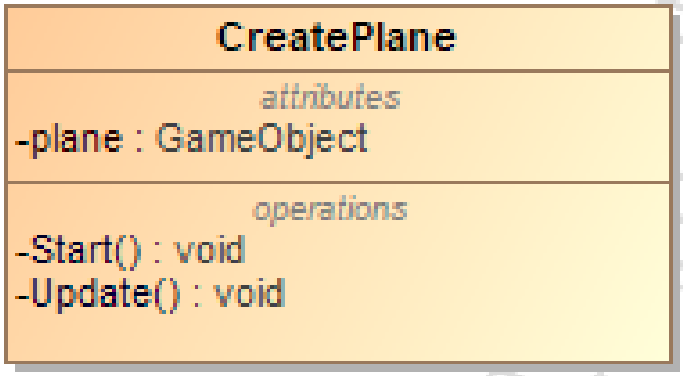


Figura 4.7: Diagrama de clases da segunda iteración

Posto que o script é un compoñente da cámara, a función ”Start()” execútase en canto o programa comeza a súa execución. Neste momento creamos o obxecto que servirá como elemento proxectable. De entre as distintas primitivas coas que conta Unity escollemos o plano, posto que a súa morfoloxía representa o aspecto de pantalla de proxección que estamos a buscar. Non obstante, posto que non queremos que o plano sexa visible sempre, e esta creación sucede ao inicio da execución do programa, desactivamos o obxecto, de forma que non se renderice na imaxe.

Por outro lado, o método ”Update()” é o que se encarga de posicionar o plano en cada fotograma. Como xa se comentou anteriormente a posición do plano vai vir determinada polas posicións dos targets detectados, de modo que temos que comezar por comprobar cal é o estado de ambos os dous targets.

Os targets contan cunha variable interna chamada StatusInfo que pode ser consultada mediante un getter. Esta variable almacena o estado do target en todo momento, de forma que necesitamos realizar un control do valor que toma para actuar en consecuencia sobre o noso plano.

Na táboa 4.1 recollemos os distintos valores que pode tomar StatusInfo:

Cando o valor de StatusInfo é NORMAL, os targets están a ser detectados e o sistema

StatusInfo	
Estado	Descripción
NORMAL	O target non presenta problemas
UNKNOWN	O target presenta algún problema de orixe descoñecida
INITIALIZING	Aínda falta información para poder realizar un bo seguimento
RELOCALIZING	O sistema trata de relocalizar o target
EXCESSIVE_MOTION	O target móvese demasiado rápido
INSUFFICIENT_FEATURES	O target non presenta suficientes características para realizar un seguimento preciso
INSUFFICIENT_LIGHT	Non hai iluminación suficiente para realizar un seguimento do target preciso
NO_DETECTION_RECOMMENDING_GUIDANCE	Non se puido detectar o target

Táboa 4.1: Valores posibles de StatusInfo

realiza un seguimento deles. Así, programamos que cando ambos os targets teñan ese estado, o plano se active, de forma que sexa renderizado na pantalla. A continuación, tomamos a posición de ambos targets, calculamos o punto medio entre eles e asignámoslle este novo punto ao plano como posición. Do mesmo modo modificamos o tamaño do plano en función da posición dos targets na pantalla.

Tamén é necesario programar como actuar en caso de que os targets pasen a ter outro estado distinto. Neste caso simplemente desactivamos o plano, de forma que xa non sexa visible na escena. Desta forma evitamos comportamentos non desexables no programa se por exemplo o sistema tratase de calcular o punto medio cando un dos targets xa non está a ser detectado.

Con todo o anterior programado, o noso sistema xa ten a capacidade de crear un plano e posicionalo en función dos targets detectados. Na figura 4.8 recollemos un exemplo da execución do programa no que podemos visualizar o resultado da implementación desta iteración.

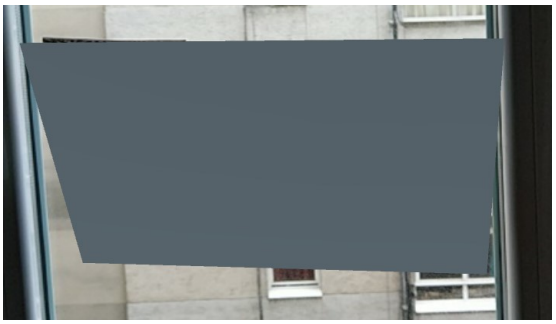


Figura 4.8: Plano creado sobre os targets detectados

4.3.3 Terceira iteración

Análise

Na terceira iteración do desenvolvemento da aplicación imos engadir a funcionalidade de cargar elementos multimedia no espazo de proxección que creamos durante a sección 4.3.2.

Para poder levar a cabo este obxectivo, temos que recurrir á documentación de Unity para saber que formatos de imaxe e vídeo soporta o motor. Tendo en conta que o sistema operativo que se está a utilizar durante o desenvolvemento é Windows, recollemos na táboa 4.2 os formatos soportados.

Formatos soportados	
Imaxe	vídeo
BMP	ASF
EXR	AVI
GIF	DV
HDR	M4V
IFF	MOV
JPG	MP4
PIC	MPG
PNG	OGV
PSD	MP8
TGA	WEBM
TIFF	WMV

Táboa 4.2: Formatos de imaxe e vídeo soportados en Unity

Deseño

Imos facer que o plano reciba un comando do usuario a través do cal cargue unha imaxe ou un vídeo seleccionados polo usuario que se proxectarán no obxecto. Por simplicidade do usuario faremos que a interacción co plano sea a través dun clic do rato sobre o propio obxecto. Toda esta funcionalidade recollerase nunha clase á que chamaremos ”HandlePlane” e que será un compoñente do plano engadido cando se crea ao inicio da execución do sistema.

Do mesmo, xa que é necesario manter un control do formato dos arquivos a cargar, imos crear un arquivo de configuración no que recoller a información que recollimos na táboa 4.2 para obtela desde outras clases mediante o uso de getters e setters. Nomearemos este arquivo ”MultimediaCompatible”.

Na figura 4.9 recollemos un diagrama de clases coas modificacións no deseño que imos implementar nesta iteración.

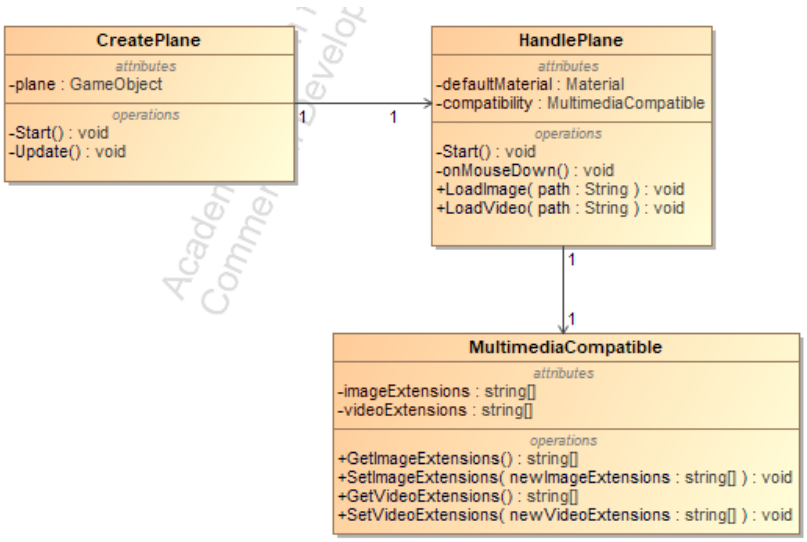


Figura 4.9: Diagrama de clases da terceira iteración

Implementación

En primeiro lugar comezamos por crear a clase ”MultimediaCompatible” e creamos os atributos ImageExtensions e VideoExtensions cos valores que xa recollimos na táboa 4.2. Do mesmo modo creamos os getters e setters que permitan acceder a eles desde outras clases.

Por outro lado creamos a clase ”HandlePlane” que é a que se encargará de darlle funcionalidade ao plano que creamos na anterior iteración.

Comezamos creando un método ”Start()” co único obxectivo de almacenar cal é o material inicial do que consta o plano. Farémolo na variable ”defaultMaterial”.

Do mesmo modo que ”Start()” é un método que se executa cando se crea o obxecto ao que pertence e ”Update()” faino en cada fotograma mentres o obxecto non sexa destruido; Unity proporciónanos un método que actúa cando o usuario fai clic sobre o obxecto ao que pertence. Este método é ”onMouseDown()”.

Programamos este método para que ao seu inicio devolva o plano ao seu estado inicial, de forma que se estaba a proxectar un vídeo, elimina o reprodutor de vídeo, e se estaba a mostrar unha imaxe, cambiaa polo material orixinal que almacenamos en ”defaultMaterial”.

A continuación facemos que o sistema abra o explorador de arquivos para que o usuario busque e escolla o elemento multimedia que quere proxectar. Cando un arquivo é seleccionado almacenamos a súa ruta no sistema. Da ruta extraemos a extensión do arquivo e realizamos unha comparación para saber se é algún dos formatos soportados en Unity. En caso de que o arquivo sexa unha imaxe aceptable polo sistema delegamos a súa carga no método ”LoadImage()” e se é un vídeo válido, en ”LoadVideo()”. Na chamada a cada un dos métodos enviamos como atributo de entrada a ruta do arquivo a cargar.

```
public void LoadImage(string path)
{
  var image = new Material(Shader.Find("Standard"));
  var fileContent = File.ReadAllBytes(path);
  var texture = new Texture2D(1, 1);
  texture.LoadImage(fileContent);
  image.mainTexture = texture;

  MeshRenderer mr = gameObject.GetComponent<MeshRenderer>();

  mr.material = image;
}
```

No método ”LoadImage” creamos un novo material que pasará a substituir o que ten o plano (de ahí que en cada chamada a ”onMouseDown” cambiásemos o material que estivese presente polo que había orixinalmente). Este novo material terá como textura a imaxe que o usuario seleccionou.

```
public void LoadVideo(string path)
{
  var videoPlayer =
    gameObject.AddComponent<UnityEngine.Video.VideoPlayer>();
  videoPlayer.url = path;

  videoPlayer.isLooping = true;

  videoPlayer.Play();
}
```

O método ”LoadVideo” é o que se encarga de cargar e proxectar un video sobre o plano. Para iso facemos uso dun compoñente que Unity xa ten prefabricado, ”VideoPlayer”. Este reprodutor de vídeo xa conta coa funcionalidade de cargar un vídeo a través da súa ruta. Do mesmo modo activamos a propiedade ”isLooping” do compoñente para que en caso de que o vídeo remate a súa execución, volva a comezar. Finalmente facemos unha chamada ao método ”Play” para iniciar a reprodución do vídeo.

Xa temos un script que é capaz de modificar un obxecto para que proxecte elementos multimedia, non obstante falta que o plano teña acceso a esta funcionalidade. Para iso hai que modificar o script ”CreatePlane”. No método ”Start()” de dito script engadimos unha nova liña de código para engadirlle o script ”HandlePlane” ao plano como un compoñente. Do mesmo modo, para que o plano poida recibir un clic do usuario é necesario engadirlle un compoñente de tipo ”Collider”, que é un compoñente que fai que os obxectos poidan sufrir colisións, no noso caso do rato.

Neste momento, cando se faga clic no plano o usuario xa pode seleccionar e proxectar o elemento dixital que desexe. Na imaxe 4.10 recollemos un exemplo da execución do sistema no que se insertou unha imaxe estática sobre o plano.

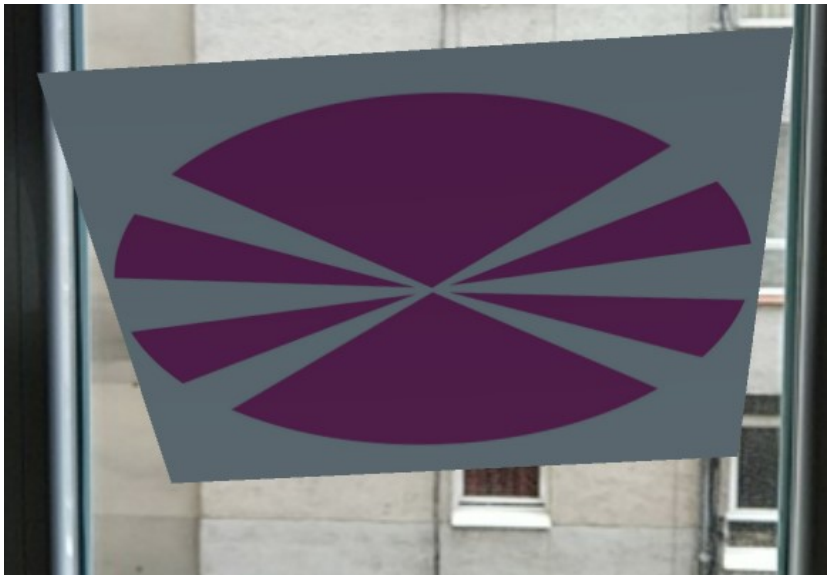


Figura 4.10: Plano no que se está a proxectar o logo da UDC

4.3.4 Cuarta iteración

Análise

Chegados a este punto, o sistema xa é capaz de realizar as accións que se esperaban del. Non obstante, imos ir un paso máis alá e intentar engadir unha nova funcionalidade. Queremos poder modificar os targets en tempo real. Desta forma o usuario podería cambiar as imaxes que a aplicación recoñece para a creación do espazo de proxección durante a execución do programa. Do mesmo modo, posto que imos a empezar a crear novos targets tamén é interesante engadir algunha maneira de poder saber que targets están a ser detectados.

Posto que o usuario poderá cargar novos targets no sistema hai que destacar que pode crear os targets de dúas maneiras diferentes. Por un lado poderá imprimir unha imaxe desde o seu equipo e situala na escena real ou ben poderá realizar unha fotografía dun escenario real e cargala no equipo.

Deseño

Para cumprir con este novo obxectivo imos crear un menú que o usuario poida despregar. desde ese menú o usuario poderá escoller entre dúas opcións. Estas opcións servirán para

modificar cada un dos targets por separado. Para a realización desta funcionalidade crearemos unha clase chamada HandleMenu. Na figura 4.11 móstrase o diagrama de obxectos actualizado co cambio que pretendemos aplicar.

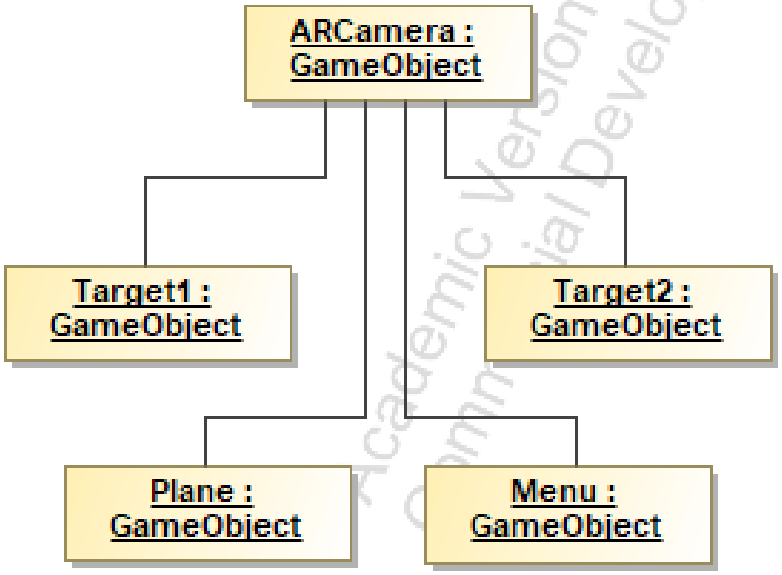


Figura 4.11: Diagrama de obxectos da cuarta iteración

Para evitar crear targets repetidos (enténdase targets repetidos por aqueles targets que encapsulen a mesma imaxe) controlarase que cada novo target a crear non estivese xa creado anteriormente, en cuxo caso simplemente se modificará cal é o target activo. Do mesmo modo, non se eliminarán os targets creados con anterioridade, senón que serán desactivados para que en caso de que o usuario desexe volver a utilizar algún deles non sería necesario cargalo outra vez no sistema, só reactivalo.

Do mesmo modo, o menú non estará sempre renderizado en pantalla. O usuario poderá facelo aparecer e desaparecer pulsando a barra espaciadora e o tabulador respectivamente.

Implementación

Imos comezar por crear o menú interactivo. Unity xa conta con obxectos prefabricados para conformar a interface de usuario. De entre os obxectos dispoñibles, creamos un obxecto Dropdown, que é precisamente un menú despregable. Este obxecto pode ser configurado engadindo as opcións que mostrará (ver figura 4.12) e indicándolle o método ao que chamar cando se faga clic nalgunha delas.

O novo script que imos crear, chamado ”HandleMenu”, asignaremosllo á cámara como un novo compoñente. Na imaxe 4.13 recollemos o diagrama UML da clase que imos implementar.

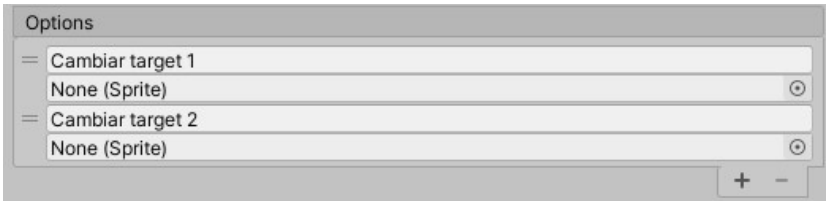


Figura 4.12: Opcións creadas no editor de Unity

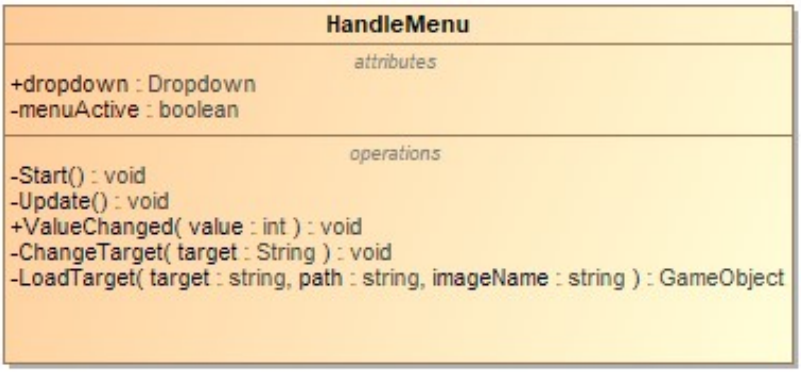


Figura 4.13: Diagrama da clase ”HandleMenu”

Este script conta con dúas variables globais:

- dropdown. Obxecto de tipo Dropdown ao que lle asignamos o menú despregable que creamos no editor.
- menuActive. Variable booleana que toma o valor true cando o usuario fai aparecer o menú e o valor false cando o menú desaparece da pantalla. O uso desta variable ven dado polo feito de que se o usuario fai clic no menú cando este está encima do plano de proxección, o sistema toma como válido o clic sobre o plano. Desta forma utilizamos esta variable para condicionar o método onMouseDown de HandlePlane.

O script conta cun método ”Start()” que entra en execución en canto o obxecto ao que pertence sexa creado. Posto que engadímoslle o script á cámara, o momento de creación ten lugar en canto o programa comece a executarse. Neste momento desactivamos o menú, de forma que non se renderice o obxecto en pantalla. Do mesmo modo asignamos o valor false á variable ”menuActive”.

”Update()” vaise encargar en cada fotograma de supervisar se o usuario pulsou as teclas Espacio ou Tabulador. En caso de que a Barra Espaciadora fose pulsada e a variable ”menuActive” tivese o valor false, dámoslle o valor true e activamos o menú, para que sexa renderizado na pantalla. En caso de que o Tabulador fose pulsado e ”menuActive” fose true, cambiamos o valor a false e desactivamos o menú para facer que desapareza.

O método ”ValueChanged()” vai ser o que se encargue de redirixir a selección de opcións do menú a outro método específico, así como de facer que cando esas chamadas sexan atendidas, o menú sexa desactivado e a variable ”menuActive” reciba o valor false. Para facer que o menú conecte con este método é necesario indicarllo no editor.

Para poder realizar a modificación dos targets activos creamos o método ”ChangeTarget”, que recibe como parámetro de entrada cal é o target que vai ser modificado, que indicamos na chamada ao método desde ”ValueChanged”.

```
1  private void ChangeTarget(string target)
2  {
3    var path = EditorUtility.OpenFilePanel("Seleccione unha imaxe",
       "", "");
4    if (path != null)
5    {
6      var completeName = path.Split("/"[0]);
7      var name = completeName[completeName.Length - 1];
8      var found = GameObject.Find(name);
9
10     if (found != null)
11     {
12       GameObject old = GameObject.Find(target);
13       old.GetComponent<ImageTargetBehaviour>().enabled = false;
14       old.gameObject.SetActive(false);
15       old.gameObject.name =
     old.GetComponent<ImageTargetBehaviour>().TrackableName;
16
17       found.gameObject.GetComponent<ImageTargetBehaviour>().enabled
     = true;
18       found.gameObject.SetActive(true);
19       found.gameObject.name = target;
20     }
21     else if (found == null && GameObject.Find("Target1").gameObject
22     .GetComponent<ImageTargetBehaviour>().TrackableName != name &&
     GameObject.Find("Target").gameObject
23     .GetComponent<ImageTargetBehaviour>().TrackableName != name)
24     {
25       GameObject old = GameObject.Find(target);
26       old.GetComponent<ImageTargetBehaviour>().enabled = false;
27       old.SetActive(false);
28       old.name =
     old.GetComponent<ImageTargetBehaviour>().TrackableName;
29
30       LoadTarget(target, path, name).transform.parent =
     gameObject.transform;;
31     }
```

```
32    }
33 }
```

Comezamos por abrir o explorador de arquivos para que o usuario escolla aquela imaxe que queira utilizar como novo target e obtemos a súa ruta. A partires da ruta obtemos o nome da imaxe e realizamos unha busca entre todos os obxectos para ver se algún ten ese nome.

Se algún dos obxectos cargados posúe ese nome quere dicir que xa existe un target no sistema con esa imaxe. Desactivamos o target que xa non vai ser usado e dámoslle o nome da imaxe que encapsula. Do mesmo modo activamos o novo target e modificamos o seu nome para que sexa o do target que substituíu (Target1 ou Target2) posto que a través de ese nome o script CreatePlane realiza o seguimento da posición dos targets.

A outra posibilidade que require un control programático é se o target non está entre ningún outro cargado anteriormente nin é ningún dos que xa se está a usar. Isto quere dicir que hai que crear un novo target de cero. Primeiramente realizamos sobre o target que vai ser modificado o mismo tratamento que fixemos na outra opción, e a continuación realizamos unha chamada á función ”LoadTarget” mandando como atributos de entrada o nome do target a crear (Target1 ou Target2), a ruta da imaxe a cargar e o propio nome da imaxe. Finalmente facemos que o target devolto por LoadTarget pase a ser tamén fillo da cámara, como os demais.

Recollemos a implementación da función ”CreateTarget” no seguinte fragmento de código:

```
1  private GameObject LoadTarget(string target, string path, string
       imageName)
2  {
3    var objectTracker =
       TrackerManager.Instance.GetTracker<ObjectTracker>();
4    var runtimeImageSource = objectTracker.RuntimeImageSource;
5    runtimeImageSource.SetFile(VuforiaUnity.StorageType.
6      STORAGE_ABSOLUTE, path, 0.25f, imageName);
7
8    var dataset = objectTracker.CreateDataSet();
9    var trackableBehaviour =
       dataset.CreateTrackable(runtimeImageSource, target);
10   trackableBehaviour.gameObject.
11     AddComponent<DefaultTrackableEventHandler>();
12   return GameObject.Find(target);
13 }
```

Nesta función comezamos por cargar un manexador a través do cal cargamos a imaxe que o usuario seleccionou anteriormente. A continuación creamos un novo obxecto que pode ser detectado polo SDK que encapsula a imaxe cargada e que recibe como nome aquel que vai dentro de ”target” que pode ser Target1 ou Target2. Finalmente engadímoslle un manexador estándar que Vuforia proporciona para os obxectos detectables e devolvemos o target creado.

Neste punto finalizamos a implementación desta iteración. Na figura 4.14 podemos apreciar o aspecto final do menú cando está despregado.

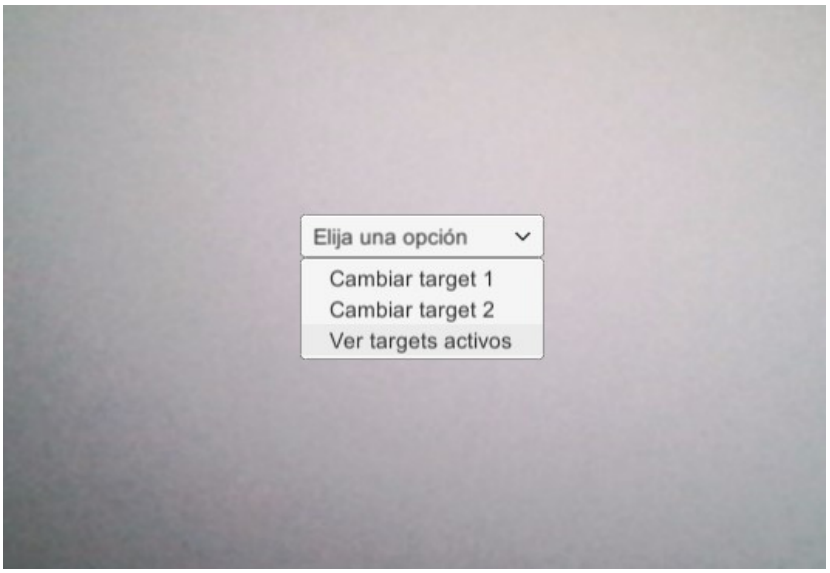


Figura 4.14: Detalle do menú despregado

4.3.5 Quinta iteración

Análise

Agora que o sistema ten a capacidade de cargar novos targets durante a súa execución, imos engadir unha nova opción no menú do programa para poder saber en todo momento os targets que están a ser detectados.

Deseño

Posto que na anterior iteración xa creamos un menú despregable, engadiremos a opción ”Ver targets activos”. Cando o usuario a seleccione, abrirase unha nova fiestra na que se mostrarán as imaxes que conforman os targets. Esta fiestra contará cun botón OK encargado de pechar a fiestra en caso de ser clicado. Esta funcionalidade implementarase sobre a clase HandleMenu xa creada anteriormente.

Unity non permite acceder ás imaxes contidas nos targets durante a execución do programa, polo que imos crear unha clase na que almacenar unha imaxe o seu nome nome e instanciaremos un obxecto desa clase cada vez que se cargue un novo target no programa para introducilo nunha lista. Chamaremos ImageLoaded a esta nova clase.

Na imaxe 4.15 recollemos o diagrama de clases actualizado con estes cambios.

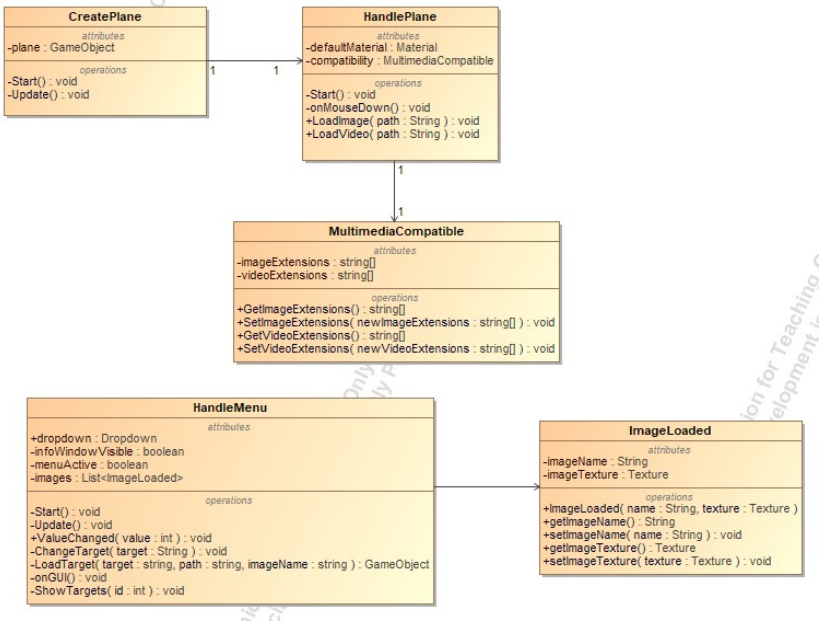


Figura 4.15: Diagrama de clases da quinta iteración

Implementación

Comezamos por implementar a clase ImageLoaded, que conta con dous atributos. O primeiro deles é imageName, un string que representa o nome da imaxe; e o outro é imageTexture, a textura na que almacenaremos as imaxes. Aparte dos getters e setters dos atributos, nesta clase incluimos un constructor para poder crear instancias do obxecto de maneira simple.

O seguinte paso consiste en programar a creación das instancias de ImageLoaded e a súa introdución nunha lista. Farémolo no script HandleMenu que xa creamos anteriormente.

Primeiramente engadimos un novo atributo no script, unha lista de tipo ImageLoaded á que chamaremos images.

Seguidamente, no método Start(), creamos dous obxectos ImageLoaded que encapsulan os targets predefinidos da aplicación e engadímolos á lista.

Do mesmo modo, tamén é necesario engadir á lista todos aqueles targets que sexan creados durante a execución da aplicación, polo que é necesario engadir esa funcionalidade no método LoadTarget que desenvolvimos no apartado 4.3.4.

Agora que xa temos creada unha estructura da que extraer as imaxes cando as necesitemos imos implementar esa extracción. A primeira acción que levamos a cabo é engadir a opción ”Ver targets activos” no menú dropdown que creamos no apartado 4.3.4 (ver imaxe 4.16).

Agora que existe unha nova opción no menú hai que actualizar o método ValueChanged para que actúe cando teña lugar o evento de elección da visualización dos targets.

Aquí entra en xogo unha nova variable booleana á que chamamos ”infoWindowVisible”

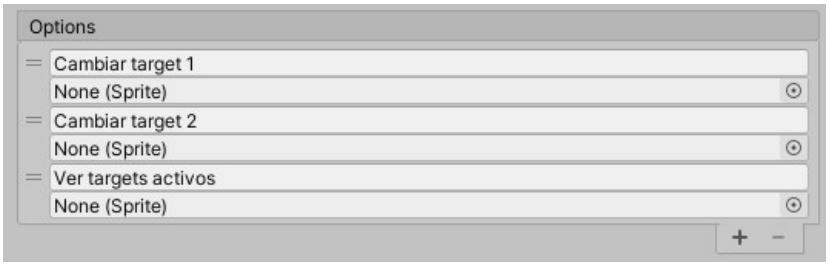


Figura 4.16: Menú de opcións no editor actualizado

que se encargará de facer aparecer e desaparecer a fiestra na que se mostran os dous targets activos. Inicialmente toma o valor false, pero ao marcar a opción Ver targets activos no menú despregable, ValueChanged modifica o seu valor a true.

Neste momento entra en acción o método ”onGUI()” de Unity. Este método está en funcionamento durante toda a execución do programa, pero inhabilitámolo mediante a variable ”infoWindowVisible”. Cando a variable ten o valor true xeramos unha fiestra que cubrimos con información mediante a execución do método ”ShowTargets”. Esta información consiste en mostrar a imaxe que encapsulan os dous targets activos no momento da chamada ao método. Tamén creamos un botón OK que ao ser pulsado pecha a fiestra e actualiza a variable infoWindowVisible para indicar que xa non está activa a fiestra, é dicir, dalle o valor false. Na figura 4.17 mostramos un exemplo desta fiestra.

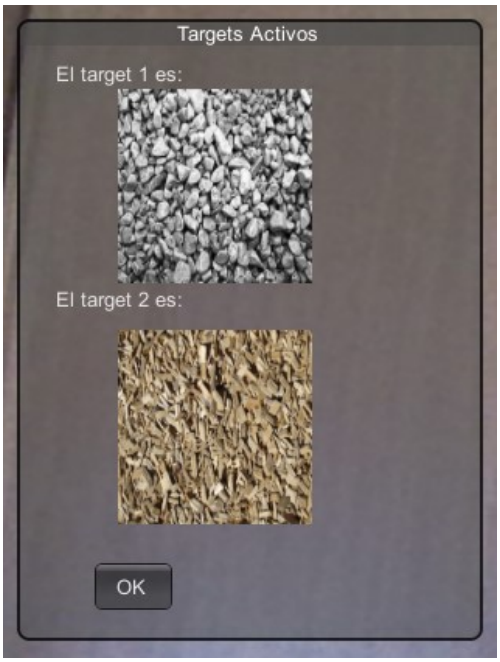


Figura 4.17: Fiestra mostrando os targets activos

4.3.6 Sexta iteración

Análise

Neste intre hai un punto moi interesante a tratar, que é a localización da aplicación. Actualmente, esta encóntrase en completo castelán, e ao longo desta iteración imos desenvolver a compatibilidade con outros idiomas.

Deseño

Para realizar a localización do programa imos desenvolver un script chamado LangResolver que se encargará de ler uns arquivos de texto nos que incluiremos o dicionario dos diferentes idiomas que desexamos incluir en función de cal sexa o idioma do sistema no que se esté a executar a aplicación. Do mesmo modo contará cun método ao que chamar cada vez que queiramos incluir un texto do dicionario.

Os dicionarios estarán compostos por pares de datos tipo chave-valor, no que a chave será o identificador do texto que imos localizar e que será o mesmo en todos os dicionarios e o valor será o texto localizado. O separador entre as chaves e os valores será o caracter "=".

Finalmente cambiaremos os textos que inserimos nas anteriores iteracións por chamadas a este novo script para que sexan localizados.

No manual de usuario de Unity podemos visualizar as linguaxes soportadas polo motor (https://docs.unity3d.com/ScriptReference/SystemLanguage.html) e podemos observar queo galego non é unha delas, polo que soamente incluiremos a localización ao inglés, e en caso de que o sistema esté noutra linguaxe distinta tomaremos o español como lingua por defecto. Do mesmo modo o nome de cada dicionario ten que coincidir co nome do idioma según se recolle na páxina web anteriormente indicada.

Implementación

Recollemos o diagrama de clases do script LangResolver na figura 4.18.

En primeiro lugar encontramos o método Awake. Este método é propio de Unity e execútase no mesmo momento en que o obxecto do que forme parte sexa inicializado. A razón de que usemos este método e non o Start é que a chamada ao método Awake realízase antes, poidendo por tanto crear unha orde de inicialización dos scripts. Neste método realizaremos unha chamada a ReadProperties.

```
1 private void ReadProperties()
2 {
3   language = Application.systemLanguage;
4   var file = Resources.Load<TextAsset>(language.ToString());
5   if (file == null)
```

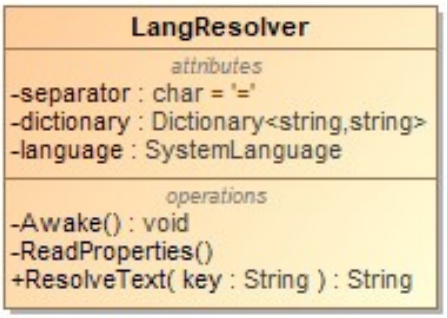


Figura 4.18: Diagrama de clases de LangResolver

```
6   {
7     file =
      Resources.Load<TextAsset>(SystemLanguage.Spanish.ToString());
8     language = SystemLanguage.Spanish;
9   }
10  foreach (var line in file.text.Split('\n'))
11  {
12    var prop = line.Split(separator);
13    dictionary[prop[0]] = prop[1];
14  }
15 }
```

ReadProperties é o método que se vai encargar de ler e parsear os dicionarios. En primeiro lugar obtén cal é a linguaxe do sistema para a continuación cargar na aplicación o dicionario con ese nome. Se o arquivo non existe quere dicir que esa linguaxe non está implementada, en cuxo caso cambiamos ao español e cargamos o dicionario correspondente. En último lugar parseamos o arquivo liña a liña e almacenamos a información na variable dictionary.

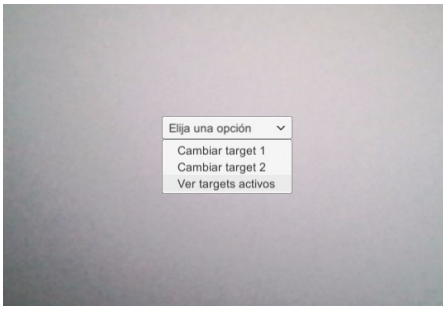
O método ResolveTexts recibe un string coa clave do valor que desexamos recuperar do dicionario e devólveo.

Creamos un novo obxecto vacío no editor ao que chamamos LangResolver e engadímoslle este script como compoñente.

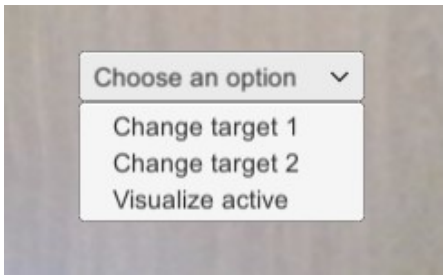
O seguinte paso consiste en modificar todos os textos da aplicación que se mostrasen ao usuario de forma que agora consistan en chamadas a ResolveTexts. Non require de gran esforzo excepto polas opcións que incluimos no dropdown, xa que no inspector non podemos incluir a chamada a este método. A solución que temos que levar a cabo consiste en eliminar as opcións do editor e modificar o método Start de HandleMenu para realizar unha chamada ao método AddOptions propio dos obxectos de tipo Dropdown para incluir as opcións coa chamada a ResolveTexts incluida.

Na figura 4.19 recollemos o menú de opcións nas dúas localizacións implementadas, cas-

telán (subfigura 4.19a) e inglés (subfigura 4.19b).



(a) Castelán



(b) Inglés

Figura 4.19: Localización de menú

4.3.7 Séptima iteración

Análise

Unha utilidade que actualmente o noso sistema non posúe é a previsualización televisiva do contido aumentado. É dicir, actualmente a vista que ten o usuario da aplicación inclúe cousas que os televidentes non deben poder ver, como por exemplo o menú despregable. Ante esta situación imos incluir unha vista adicional na aplicación na que o usuario poida asegurarse de que é o que se estaría a retransmitir desde a ferramenta.

Deseño

Para a previsualización televisiva decidimos facer unha segunda pantalla na principal de modo similar a como son os minimapas nos videoxogos. Esta segunda pantalla proxectará a imaxe que será retransmitida desde a aplicación. Do mesmo modo o usuario do sistema poderá interactuar con esta pantalla facéndoa aparecer e desaparecer da vista de maneira que non lle moleste e modificar o seu tamaño de modo similar a como podemos facelo coas fiestras de calqueira ordenador.

Implementación

A primeira tarea que temos que realizar consiste en cambiar o renderizado da cámara para que renderice cara unha textura en vez de cara a pantalla. Para iso primeiro temos que crear no editor un asset de tipo Render Texture ao que chamaremos tv.

Os Render Textures son un tipo especial de texturas que se crean en tempo de execución e que hai que asignarllos a unha cámara para que renderice a imaxe que captura neles.

Isto xera un problema, posto que Unity de forma nativa renderiza ou ben cara a pantalla ou ben cara unha textura, de forma que agora ao executar a aplicación móstrasenos unha pantalla en negro. Para solucionar isto necesitamos facer uso do script "RenderScreenTexture", cuxo diagrama de clase recóllese na figura 4.20.

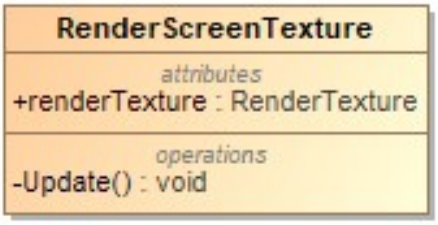


Figura 4.20: Diagrama da clase RenderScreenTexture

Este script conta co método Update no que en cada chamada durante a execución do sistema fai que a cámara renderice cara a tv e a continuación fai que a cámara deixe de renderizar na textura. Desta forma conseguimos que tanto a textura como a pantalla mostren a imaxe capturada pola cámara. Do mesmo modo o script conta con un atributo público que é a textura na que queremos renderizar, por tanto, unha vez que engadamos o script como compoñente á cámara, seleccionamos a textura tv no editor.

Aínda que poida parecer que esta estratexia de activar e desactivar o renderizado na textura faría que a imaxe sufrise unha caída na taxa de fotogramas, tras realizar unha proba do sistema non se viu ningunha caída no rendemento.

Agora que xa temos a imaxe duplicada podemos facer a segunda pantalla. No apartado 4.3.4 cando creamos o menú despregable, Unity creou automáticamente un Canvas ao que anexalo. O Canvas é o obxecto que encapsula a área na que se recolle toda a interface de usuario.

Neste Canvas creamos un novo obxecto de tipo Raw Image ao que chamamos TV. Escollemos un obxecto deste tipo porque Raw Image pode mostrar calqueira tipo de textura. A continuación colocamos TV na posición que nos interesa na interface, que no noso caso é a parte inferior dereita da pantalla a axustámola ao borde do Canvas. Finalmente, indicamos no inspector de TV que a textura que ten que mostrar é tv, a Render Texture que creamos antes. Na imaxe 4.21 vemos un exemplo da execución do sistema no seu estado actual, reparando en como o menú despregable non se visualiza en TV e por tanto non o verían os televidentes.

Coa fiestra que imita a unha televisión xa feita pasamos a darlle as funcionalidades de aparecer e desaparecer da vista do usuario da aplicación e de modificar o seu tamaño mediante o método de clicar e arrastrar, método habitual de agrandar e empequeñecer as fiestras nos ordenadores.

Para a creación destas funcionalidades creamos o script TVWindow, cuxo diagrama de

Figura 4.21: Execución do sistema coa tv

clases recollemos na figura 4.22.

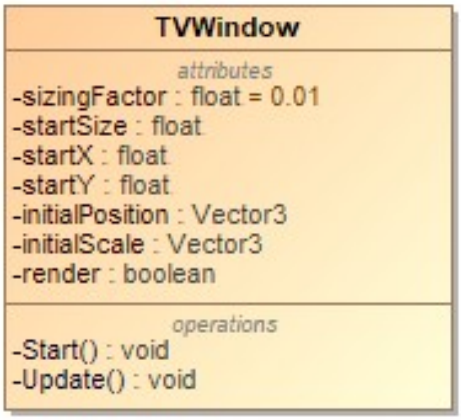


Figura 4.22: Diagrama da clase TVWindow

O método Start de TVWindow serve para darlle un valor inicial ás variables initialPosition e initialScale, estes valores iniciais son os que ten a fiestra na vista do editor, é dicir, na esquina inferior dereita do Canvas. Do mesmo modo, inicialmente non queremos que a fiestra sexa visible polo que necesitamos ocultala. Para facelo accedemos á posición do obxecto e modificámola por outra que se encontre fora dos límites da pantalla, neste caso utilizamos a posición arbitraria (2500, 2500, 0). Adicionalmente, dámoslle o valor false ao atributo render, que é o que utilizaremos para saber cando mostrar a fiestra en pantalla e cando non.

No seguinte fragmento de código recollemos o método Update:

```
1 private void Update()
2 {
3   if (Input.GetKeyDown(KeyCode.A))
4   {
5     if (!render)
6     {
7       gameObject.transform.position = initialPosition;
```

```
8         gameObject.transform.localScale = initialScale;
9       }
10      else
11      {
12        gameObject.transform.position = new Vector3(2500f,2500f,0f);
13      }
14      render = !render;
15    }
16
17    if (Input.GetMouseButtonDown(0))
18    {
19      Vector3 position = new Vector3(Input.mousePosition.x,
        Input.mousePosition.y, 0f);
20      startX = position.x;
21      startY = position.y;
22      startSize = gameObject.transform.localScale.z;
23    }
24
25    if (Input.GetMouseButton(0))
26    {
27      Vector3 size = gameObject.transform.localScale;
28      size.x = startSize - (Input.mousePosition.x - startX) *
        sizingFactor;
29      size.y = startSize + (Input.mousePosition.y - startY) *
        sizingFactor;
30      gameObject.transform.localScale = size;
31    }
32  }
```

Inicialmente, comprobamos se a tecla A foi pulsada polo usuario. En caso afirmativo comprobamos cal é o valor da variable render:

- Se ten o valor true, cambiamos a súa posición á inicial, visible para o usuario, e modificamos o seu tamaño para que sexa o que tiña inicialmente no editor.
- En caso contrario simplemente modificamos a súa posición ao punto (2500,2500,0), fora da vista do usuario.

En calqueira dos dous casos, modificamos o valor da variable render ao seu contrario para que na próxima chamada a Update o bloque if funcione como queremos.

A continuación encontramos dous bloques if que entran en funcionamento en caso de que o usuario faga clic, coa diferenza de que o método de Unity GetMouseButtonDown actívase no momento no que o usuario fai a pulsación do clic, momento no que devolve true; e GetMouseButton devolve o true cando o usuario solta a pulsación do clic. Desta forma podemos ter cal é a posición inicial de pulsación do rato e modificar o tamaño gradualmente da fiestra a

medida que o rato desplázase pola escena. Incluimos unha variable chamada sizingFactor cun valor fixo de 0.01 para reducir o impacto do arrastre, de maneira que un mínimo movemento do rato non faga un aumento máximo da fiestra.

Con este método Update xa podemos modificar o tamaño da fiestra televisiva así como a súa aparición en función dos desexos do usuario do sistema. Na imaxe 4.23 mostramos o agrandamento da fiestra nunha execución.

Figura 4.23: Fiestra agrandada

4.3.8 Oitava iteración

Análise

O presente proxecto, como xa explicamos no apartado 1.2, ten por finalidade a emisión televisiva do contido aumentado que xeramos na aplicación. Non obstante, a presente aplicación ten un funcionamento só apto para o seu uso na sala desde a que se dirixe a emisión televisiva, falta a creación dunha saída de vídeo aumentado definitiva que poida ser retransmitida.

Partindo desta premisa imos crear unha nova aplicación que tome a imaxe aumentada do actual sistema (como xa fai a fiestra que creamos no apartado 4.3.7) sen ningún tipo de elemento administrativo visible que a dirección televisiva poida utilizar directamente para a súa retransmisión.

Deseño

O proceso que imos levar a cabo para poder transmitir o vídeo consiste en convertir á aplicación actual nun servidor ao que unha nova aplicación cliente poida conectarse. O servidor aceptará a conexión desde o cliente e comezará a enviar o vídeo. O cliente recibirá os bytes do vídeo e tras decodificalos mostraraos na pantalla.

A conexión entre o servidor e o cliente realizarase a través da interface de loopback, de forma que ambas aplicacións funcionaran na misma máquina. No funcionamento real do sistema podería facerse uso dun equipo con dúas pantallas, na que nunha delas esté a aplicación

cliente a pantalla completa, esta segunda pantalla é a que se usaría na central de emisión para o envío ás televisións.

Implementación

En primeiro lugar imos habilitar o servidor na nosa aplicación. Este desenvolvemento irá encapsulado nun novo script ao que chamaremos NetworkSender, e cuxo diagrama de clases está recollido na figura 4.24.

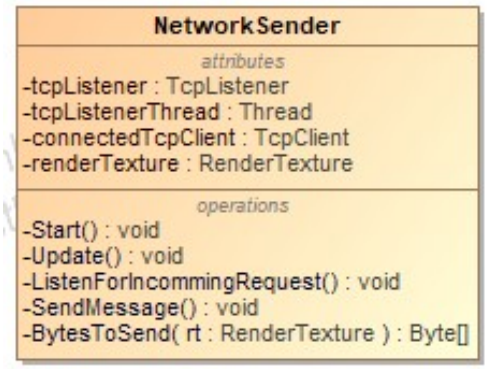


Figura 4.24: Diagrama da clase NetworkSender

A execución deste script, que colocamos como novo compoñente á cámara, comeza no método Start. Cando o script se inicia inicializamos a variable tcpListenerThread creando un novo thread que executará constantemente o método ListenForIncommingRequests sen afectar ao funcionamento do thread principal.

No método ListenForIncommingRequests inicializamos o atributo tcpListener como un novo obxecto de tipo TcpListener, facendo que escoite na interface de loopback (127.0.0.1). En caso de que un cliente intente conectarse, acéptao e almacénao na variable connectedTcpClient.

Por outro lado en cada nova chamada a Update necesitamos enviar a imaxe que xera o sistema, para o cal facemos a chamada a SendMessage. Neste método comprobamos se hai algún cliente conectado e en caso afirmativo, se o envío é posible, convertimos o frame actual do render texture que creamos na sección 4.3.7 a bytes e procedemos ao seu envío. A codificación en bytes do render texture ten lugar na función BytesToSend.

Chegados a este punto xa temos o servidor desde o que xerar a retransmisión aumentada, agora pasamos á creación da aplicación cliente.

Creamos un novo proxecto en Unity ao que chamamos TV e no editor creamos un novo obxecto de interface de usuario de tipo Raw Image. O sistema creará automáticamente un Canvas co tamaño da pantalla, de modo que redimensionamos o obxecto Raw Image para

ocupar todo o espazo do Canvas. Este obxecto será o que reciba a emisión desde a aplicación servidora e mostrará a imaxe, para o cal imos crear un novo script denominado NetworkReceiver (ver diagrama de clase na figura 4.25)

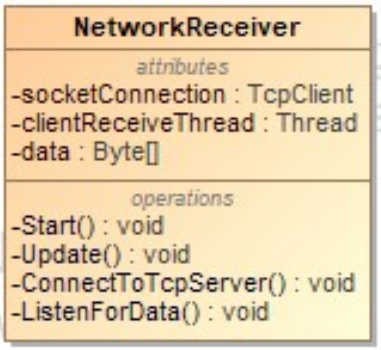


Figura 4.25: Diagrama da clase NetworkReceiver

Cando o script se inicia, no Start facemos unha chamada ao método ConnectToTcpServer, que crea un novo thread en segundo plano executando o método ListenForData. Este método abre unha conexión co servidor e recibe os bytes que envía.

Non obstante, Unity non permite modificar a textura dun obxecto nun thread que non sexa o principal, polo que os datos recibidos en ListenForData envorcámolos na variable global data e necesitamos facer uso de un Update que en cada execución cargue a textura embebida nesa variable e asígnella ao Raw Image.

Con isto rematamos o desenvolvemento da aplicación cliente. Na figura 4.26 observamos como na aplicación de administración estamos a visualizar os targets activos, algo que non pode ser retransmitido aos televidentes nunca, e na aplicación cliente vemos como non se está a nostrar na retransmisión.

(a) (b)

Figura 4.26: Diferenza entre aplicación de administración e de cliente

4.3.9 Novena iteración

Análise

Nas salas de dirección responsables da emisión televisiva teñen unha planificación temporal preparada con anterioridade de cando teñen que entrar en pantalla os distintos contidos, ben sexan informativos o publicitarios. Deste modo imos introducir unha nova funcionalidade que permita ao usuario do sistema preparar a reproducción dos elementos multimedia a priori.

Deseño

Para desenvolver esta iteración dividimos o traballo en dúas partes:

- Por un lado o plano ten que recoñecer que chegada unha determinada hora ten que cargar un contido multimedia sen que o usuario faga clic nel.
- Por outro lado o usuario ten que ser capaz de indicarlle ao sistema o elemento que desexa cargar, así como o momento no que ten que empezar a reproducirse e o momento no que debe deixar de facelo.

No primeiro caso crearemos unha nova clase, ProjectionScheduled, que encapsulará un novo tipo de obxecto. Este obxecto contará con tres atributos:

- A ruta do equipo na que se encontra o elemento multimedia.
- O momento de inicio da reproducción desexado.
- O momento de finalización da reproducción desexado.

No que respecta aos tempos faremos uso do tipo de dato DateTime propio de C#. Unha vez teñamos creado este obxecto, modificaremos o script HandlePlane, encargado da reproducción dos elementos multimedia, para que leve o control de cando teñen lugar os tempos de inicio e finalización.

Para a interacción co usuario incluiremos unha nova opción no menú interactivo despregable que ao ser seleccionada abrirá o explorador de arquivos para seleccionar o elemento que desexa cargar. Unha vez que obteñamos a ruta, o sistema mostrará un campo de entrada de datos no que o usuario poderá introducir o inicio e remate da reproducción.

Implementación

O primeiro paso que realizamos consiste en crear a clase ProjectionScheduled que xa deseñamos (ver figura 4.27).

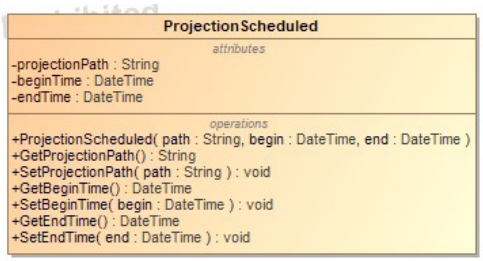


Figura 4.27: Diagrama da clase ProjectionScheduled

A continuación pasamos a modificar HandlePlane. Primeiramente creamos unha lista na que almacenaremos os distintos obxectos ProjectionScheduled que o sistema vaia recibindo. A introducción de elementos nesta lista faise a través dun novo método chamado AddProjectionSchedule.

Con isto feito necesitamos crear dous métodos máis, un que será o encargado de iniciar a reproducción dun elemento programado e outro que a rematará. Non obstante para facer isto necesitamos comparar as horas programadas coa hora do sistema e o método de comparación de tempos da clase DateTime é tan exacto que ao realizar probas con distintas comparacións, nunca se obtiña a igualdade, polo que foi necesario crear un novo método de comparación ao que chamamos TimeCompare.

Unha vez que temos o método de comparación pasamos a desenvolver os dous métodos anteriormente comentados:

```
1  private void StartScheduledProjection()
2  {
3      var now = DateTime.Now;
4      var compare = projections.Find(scheduled =>
       TimeCompare(scheduled.BeginTime, now) == true);
5      if (compare != null) {
6          broadcasting = compare;
7          projections.Remove(broadcasting);
8          LoadMultimedia(broadcasting.ProjectionPath);
9      }
10 }
11
12 private void EndScheduledProjection()
13 {
14     if(broadcasting != null)
15     {
16         if (TimeCompare(broadcasting.EndTime, DateTime.Now) == true)
17         {
18             broadcasting = null;
19             RestartPlane();
```

```
20              }
21          }
22  }
```

En cada chamada ao método StartScheduledProjection, obtemos a hora do sistema e miramos se na lista de obxectos ProjectionScheduled (chamada projections) existe algún cuxo tempo de inicio coincida, en cuxo caso extraémolo nunha variable global chamada broadcasting, e eliminámolo da lista. Finalmente iniciamos a reproducción do elemento multimedia realizando unha chamada ao método LoadMultimedia, que encapsula todo o proceso de discriminación de tipos de contido multimedia que antes faciamos en OnMouseDown.

O método EndScheduledProjection, comproba en cada chamada se hai algún elemento reproducíndose e se é o caso comproba se a hora de finalización da reproducción dese elemento é a actual, en cuxo caso anula o obxecto, e reinicia o plano mediante o método RestartPlane que desenvolvimos relocalizando o reinicio do plano que antes faciamos en OnMouseDown.

Posto que queremos que o sistema chame a estes métodos constantemente durante a execución do sistema para controlar os tempos, creamos un método Update no que introducimos as chamadas.

Na figura 4.28 recollemos o diagrama de HandlePlane que inclúe estes cambios.

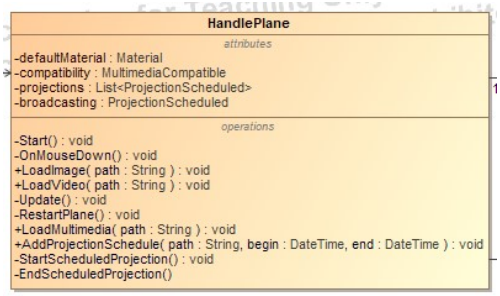


Figura 4.28: Diagrama da clase HandlePlane actualizado

Agora falta a interacción co usuario. Para iso comezamos por crear no editor un novo obxecto de tipo Input Field para o Canvas, e que chamaremos TimeProgrammer. Estes obxectos son campos de entrada de datos nos que os usuarios poden introducir texto desde o seu teclado.

A continuación pasamos a modificar HandleMenu. No método Start facemos que TimeProgrammer se desactive, para facer que non se renderice ao inicio da execución. Do mesmo modo engadimos unha nova opción ao menú despregrable tendo en conta que o texto ten que estar nos dicionarios que creamos no apartado 4.3.6.

No método ChangeValue, encargado de recibir a selección do menú despregable engadimos o novo caso correspondente a este desenvolvemento facendo que o sistema abra o explorador de arquivos para que o usuario seleccione o elemento multimedia que desexa cargar.

Unha vez seleccionado gardamos a súa ruta e activamos TimeProgrammer, de maneira que aparezca en pantalla.

O método que vai tomar a información escrita polo usuario en TimeProgrammer é Schedule-Multimedia. Neste método tomamos o string que o usuario introduciu cos tempos de inicio e finalización e separámolos. A continuación parseámolos como obxectos de tpo DateTime e finalmente realizamos unha chamada ao método AddProjectionSchedule de HandlePlane para introducir crear este novo elemento programado.

Na figura 4.29 mostramos o menú actualizado (imaxe 4.29a) e un exemplo de execución de TimeProgrammer (4.29b)

(a) (b)

Figura 4.29: Programación de contidos multimedia

4.4 Deseño final

Tras o traballo realizado ao longo das distintas iteracións que compoñen o desenvolvemento do sistema, contamos con dúas aplicacións funcionais, unha de administración na que o usuario do sistema, desde a sala de control nun plató de televisión pode xerar contido multimedia sen necesidade de ter coñecementos previos; e outra que serve de base para retransmisión televisiva mostrando o contido aumentado xerado na aplicación de administración pero sen mostrar a interface de usuario propia dela.

Este desenvolvemento foi posible a través da programación de diferentes scripts para cumplir as distintas necesidades, de modo que o diagrama de clases foi aumentando en cada nova iteración do desenvolvemento. Así, na imaxe 4.30 recollemos o diagrama de clases definitivo tras o desenvolvemento do sistema.

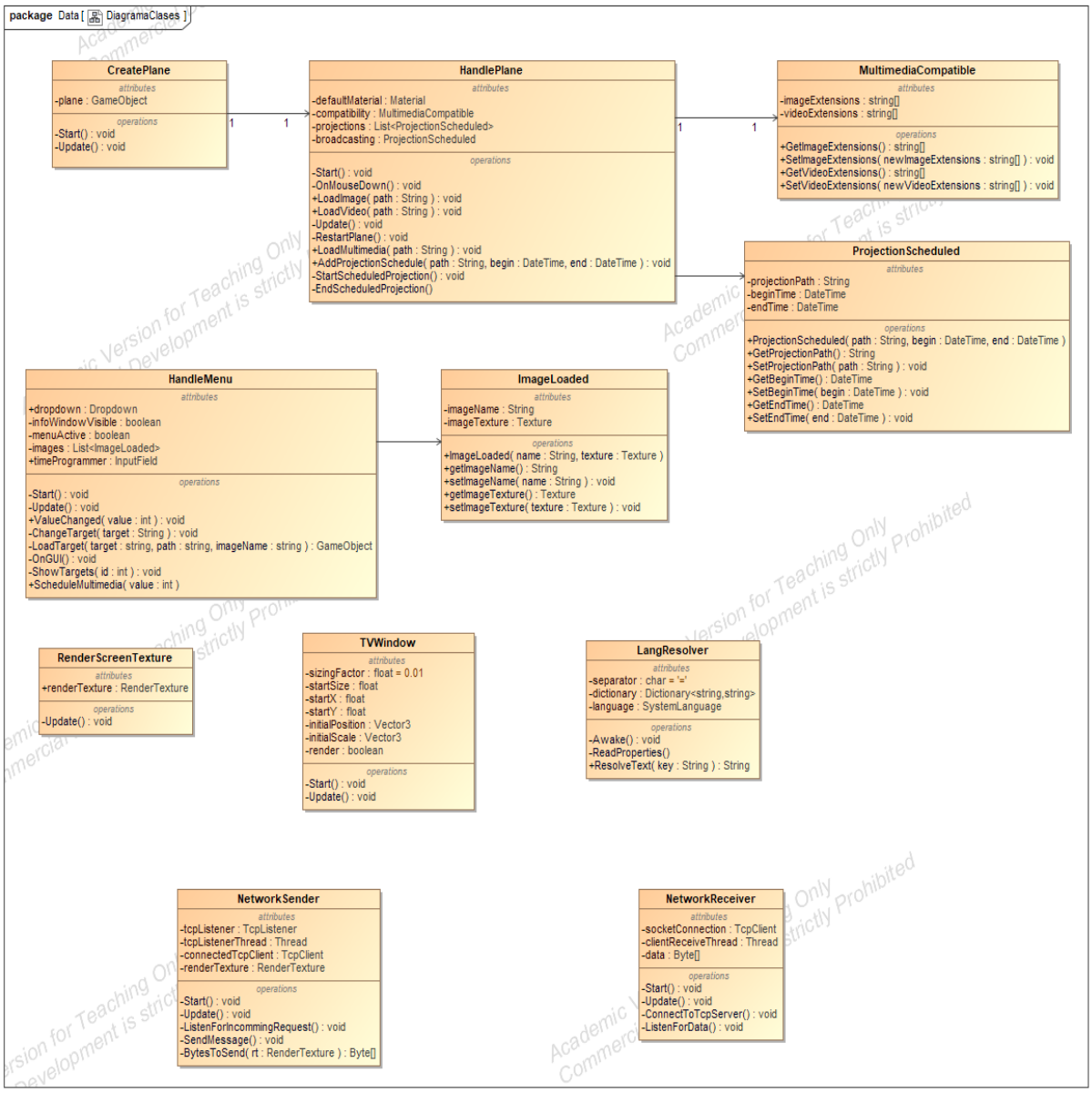


Figura 4.30: Diagrama de clases final

Capítulo 5

Conclusións

O obxectivo deste proxecto, como xa se comentou na sección 1.3, era desenvolver un sistema capaz introducir elementos dixitais en tempo real nunha escena capturada por unha cámara mediante o uso da realidade aumentada. No capítulo 4 fomos detallando os diferentes pasos para facer realidade o obxectivo, así como a introdución de funcionalidades adicionais que completasen as opcións que aporta o sistema de cara á súa incorporación ás salas de dirección de televisións xa establecidas.

A situación actual na que encontra o sistema desenvolto xa é un estado funcional, isto quere dicir que xa podería ser utilizado en transmisións en directo. Previo aprendizaxe dun operador do funcionamento da aplicación.

Chegados a este punto, por tanto, só resta facer unha análise posterior ao desenvolvemento do proxecto.

5.1 Coñecementos utilizados

Ao longo do desenvolvemento da aplicación foi necesario o uso de coñecementos adquiridos no grao así como a búsqueda de novos coñecementos.

Os coñecementos teóricos fundamentais do proxecto, a realidade aumentada e o funcionamento de Unity forman parte da materia Contornos Inmersivos, Interactivos e de Entretemento pertencente ao itinerario de Computación do grao. Na docencia da materia explícase que é a RA, como funciona e cales son os principais SDKs, entre os que se encontra Vuforia. Do mesmo modo, a teoría relacionada co desenvolvemento de videoxogos realízase sobre o motor Unity pola súa curva de aprendizaxe. Non obstante, ao longo da materia non se realiza unha integración entre ambos conceptos de maneira práctica, de modo que foi necesario un estudo do funcionamento da API de Vuforia en Unity para poder levar a cabo as funcionalidades que o sistema que se buscaba desenvolver requiría.

Por outro lado, durante a creación da ferramenta fíxose uso do sistema iterativo incre-

mental, unha das metodoloxías de traballo máis importantes e presente no temario da materia Proceso Software.

No que respecta á localización do sistema, a materia Interfaces Persoa Máquina do itinerario de computación introduce o concepto e realiza un especial fincapé nel.

O aprendizaxe en torno ao funcionamento das redes e do protocolo TCP, do que facemos uso na creación da aplicación cliente, forma parte do temario da materia Redes. Do mesmo modo, para poder traballar nesta conexión foi necesario a introdución de threads adicionais durante a execución. O grao en Enxeñaría Informática introduce a teoría relativa aos threads na materia Concorrencia e Paralelismo.

5.2 Traballo futuro

O sistema realizado conta agora mesmo con limitacións sobre as que se podería traballar en futuros desenvolvementos sobre a aplicación.

Neste momento o sistema só pode traballar cunha cámara, é dicir, non se pode modificar cal é a cámara da que se está a recibir a imaxe durante a execución. Para contar con esta nova funcionalidade engadiríase unha nova opción no menú despregable co texto ”Cambiar a cámara” que a súa vez despregase un submenú coas distintas cámaras detectadas. O usuario escollería aquela cámara que lle interesase e a imaxe que se mostra en pantalla pasaría a ser a desa cámara. Non obstante este desenvolvemento non é posible na actualidade posto que a API de Vuforia non permite acceder mediante script á cámara máis que cun getter, non cun setter, de forma que non se pode modificar en tempo real.

Actualmente, Unity non permite, de momento, xerar un executable dun programa de realidade virtual para ordenadores, por tanto a execución da ferramenta está supeditada ao uso do editor. En caso de que nalgún momento esta situación cambie, xa se poderia crear un arquivo executable que podería ser comercializado por si mesmo. Cabe destacar que actualmente o sistema podería ser distribuido como un paquete para o editor coas funcionalidades implementadas para futuros desenvolvedores.

Durante o desenvolvemento da aplicación, cada vez que era necesario facer uso do explorador de arquivos utilízase un método pertencente á clase EditorUtility, que pertence á libraría Unity Editor de Unity. Esta libraría implementa APIs propias do editor e non pode ser utilizada en aplicacións compiladas. Desta forma, se nalgún momento Unity permitise a creación de executables para computadoras de aplicacións de realidade aumentada, sería necesario modificar esa chamada. Para realizar a modificación hai dúas vías posibles. Por un lado podemos realizar un sistema que imite o comportamento do explorador de arquivos permitindo navegar polo sistema e seleccionar o arquivo que o usuario desexe. Por outro lado podería facerse uso da tenda virtual de Unity e buscar se xa existen exploradores de arquivos

xa desenvoltos por outros usuarios do motor.

Do mesmo modo, será necesaria a realización do módulo de discriminación de contidos para que o desenvolvemento da aplicación orixinal sexa completo.

Apéndices

Apéndice A

Manual de instalación

O sistema actual conta con dúas aplicacións, unha de administración e outra cliente para as retransmisións, que funciona en conxunción coa primeira. A continuación pasamos a explicar como instalar cada unha delas:

A.1 Instalación da aplicación de administración

Imos explicar os pasos necesarios para instalar a ferramenta desenvolta no editor de Unity. Para este manual de usuario a versión de Unity da que se fai uso é Unity 2019.3.11f1.

En primeiro lugar é necesario que o proxecto no editor sexa capaz de traballar con Vuforia. En caso de que non estea ainda preparado, nesta versión de Unity é necesario descargar o SDK de Vuforia dende a páxina web `https://developer.vuforia.com/downloads/sdk` e importalo ao proxecto de Unity.

Para importar o SDK no proxecto seleccionamos no menú do editor Assets>Import Package>Custom Package...(ver figura A.1).

Ao facer clic abrirase o explorador de arquivos e será necesario seleccionar o arquivo descargado dende a web. Unha vez que o usuario o seleccione o editor mostrará a fiestra que podemos observar na figura A.2 na que será necesario pulsar sobre Import.

Unha vez que o proxecto xa está preparado para traballar con Vuforia realízase a importación da ferramenta desenvolta. O proceso a levar a cabo é o mesmo que acabamos de explicar. Ao finalizar esta segunda importación xa se pode usar a aplicación no editor.

A.2 Instalación da aplicación cliente de retransmisión

A aplicación cliente conta cun executable chamado TV.exe. Deste modo basta con facer clic sobre el para iniciar a aplicación e comezar a súa execución. Non son necesarios pasos adicionais para facela funcionar.

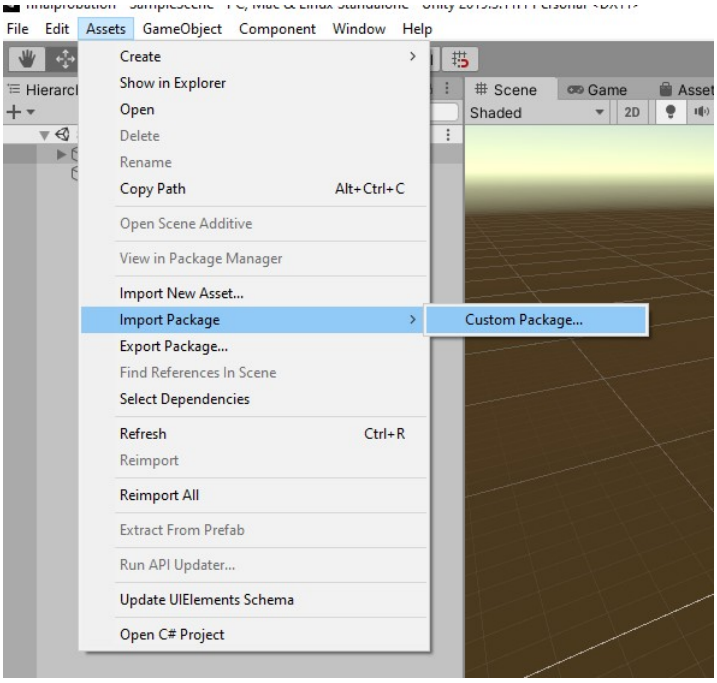


Figura A.1: Importación dun paquete en Unity

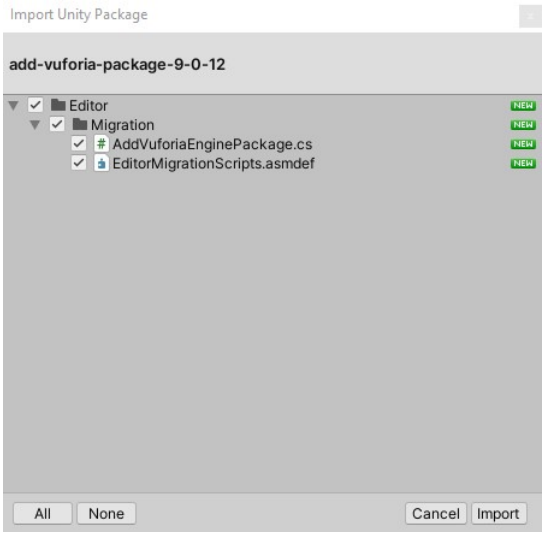


Figura A.2: Fiestra de importación do SDK de Vuforia

Apéndice B

Manual de usuario

Neste manual imos explicar como funcionan as dúas aplicacións que compoñen este sistema de cara ao usuario:

B.1 Aplicación de administración

Mediante este sistema o usuario pode engadir elementos multimedia á imaxe capturada por unha cámara. Estes elementos multimedia imprimiranse sobre un plano de proxección visible na pantalla cando dous targets estén presentes na pantalla. Os targets son imaxes detectables por un sistema de realidade aumentada. A posición dos targets é indiferente, non importa cal dos dous se encontre á esquerda e cal á dereita.

Aínda que ao inicio da execución do programa o sistema conte con dous targets prefixados, que recollemos na figura B.1, o usuario poderá modificalos segundo as súas necesidades. Para obter os targets o usuario pode ou ben imprimir as imaxes que necesite dende o seu equipo e ubicalas no escenario que capturará coa cámara ou ben pode realizar fotografías de elementos do escenario e cargalas no seu equipo.

(a) (b)

Figura B.1: Targets iniciais do sistema

O usuario pode utilizar aqueles targets que desexe, non obstante hai unha serie de normas que as imaxes deben cumprir para ser utilizadas no sistema:

- O formato das imaxes debe ser JPG ou PNG.
- O peso das imaxes non pode superar os 2.25 megabytes.
- As imaxes deben poseer unha anchura mínima de 320 píxels.

Do mesmo modo indicamos unhas recomendacións que deben seguir as imaxes para que a detección sexa o mellor posible:

- Os obxectos da imaxe deben conter moitas esquinas. No campo da visión artificial a detección de esquinas é unha das formas máis sinxelas de detectar obxectos nunha imaxe, por tanto necesitamos que os elementos presentes na nosa imaxe conten con moitas esquinas para facilitar o proceso de detección. Cabe destacar que os obxectos redondos non teñen esquinas, polo tanto dificultan moito o proceso.
- O contraste entre os distintos elementos da imaxe debe ser o maior posible. Se o contraste entre elementos é moi baixo o sistema pode non detectar cando comeza un elemento e cando termina, de forma que a detección se vexa imposibilitada.
- Evitar imaxes borrosas ou cun exceso de iluminación. Estas imperfeccións da imaxe poden provocar que non se recoñezan suficientes características nas imaxes e por tanto a súa detección no mundo real pode verse comprometida.

En caso de que o usuario desexe facer unha comprobación de se unha imaxe será detectable ou non pode recurrir a unha ferramenta web que proporciona o motor de realidade aumentada do que a presente ferramenta fai uso. A continuación recollemos os pasos a realizar para facer a comprobación:

1. Acceder á URL `https://developer.vuforia.com/vui/auth/login`. Nesta páxina poderemos identificarnos co noso usuario e contrasinal se xa estamos rexistrados (ver figura B.2a) ou ben acceder ao rexistro se é a primeira vez que accedemos.
2. Unha vez rexistrados, accedemos ao portal de desenvolvedores, onde seleccionaremos a pestana Target Manager e faremos clic no botón ”Add Database” (ver figura B.2b).
3. Surxirá unha fiestra emerxente na que introducir o nome da base de datos na que engadir os targets posteriormente (figura B.2c). Unha vez que o nome estea introducido pulsaremos sobre Create.
4. Ao acceder a esta base de datos creada poderemos engadir targets ao seleccionar o botón Add Target (figura B.2d).
5. Abrirase unha fiestra na que poderemos seleccionar a imaxe que queremos cargar nun target así como indicar o ancho da mesma (figura B.2e).

6. Agora xa poderemos ver como de detectable será a imaxe mediante unha puntuación dada a través dun sistema de estrelas que vai de unha a cinco (figura B.2f). Cantas máis estrelas obteña a imaxe, máis fácil será para o sistema recoñecer o target no entorno.

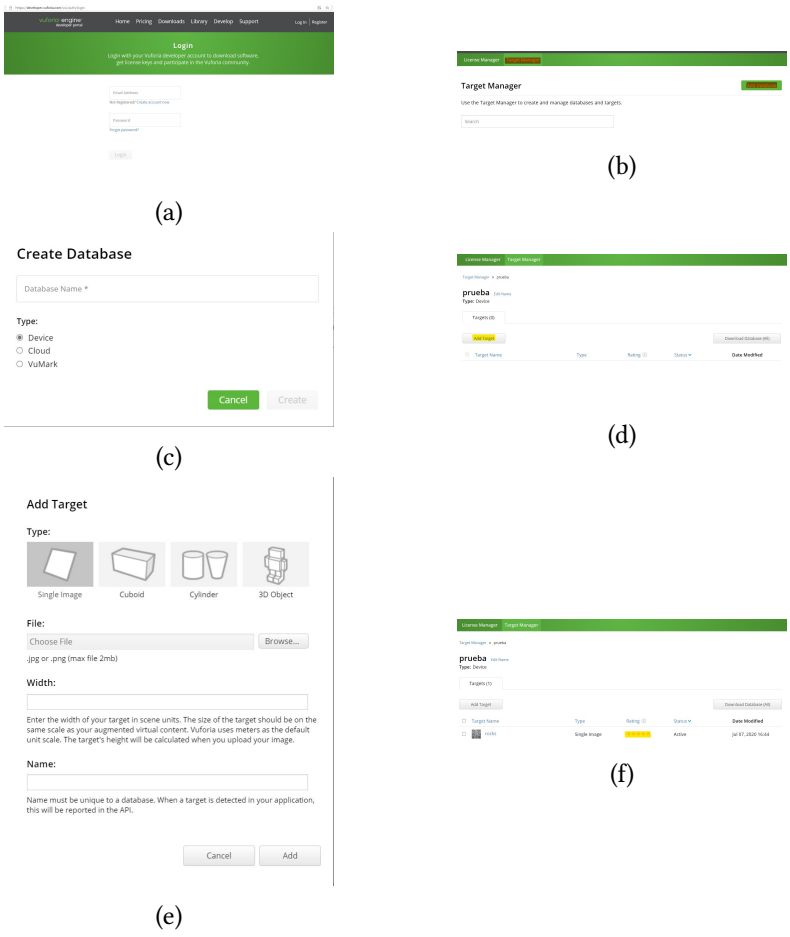


Figura B.2: Proceso para comprobar a detectabilidade dunha imaxe

Os elementos multimedia que o usuario pode cargar no sistema para a súa proxección poden ser unha imaxe estática ou un vídeo sempre e cando o formato do arquivo sexa soportado polo sistema. Na táboa B.1 recollemos os formatos soportados polo sistema para os arquivos multimedia.

B.1.1 Funcionamento

Ao inicio da execución do sistema o usuario visualizará na pantalla do dispositivo que execute a aplicación a imaxe que capta a cámara de vídeo (ver imaxe B.3).

Formatos soportados	
Imaxe	vídeo
BMP	ASF
EXR	AVI
GIF	DV
HDR	M4V
IFF	MOV
JPG	MP4
PIC	MPG
PNG	OGV
PSD	MP8
TGA	WEBM
TIFF	WMV

Táboa B.1: Formatos de imaxe e vídeo soportados polo sistema

Figura B.3: Visualización inicial do sistema

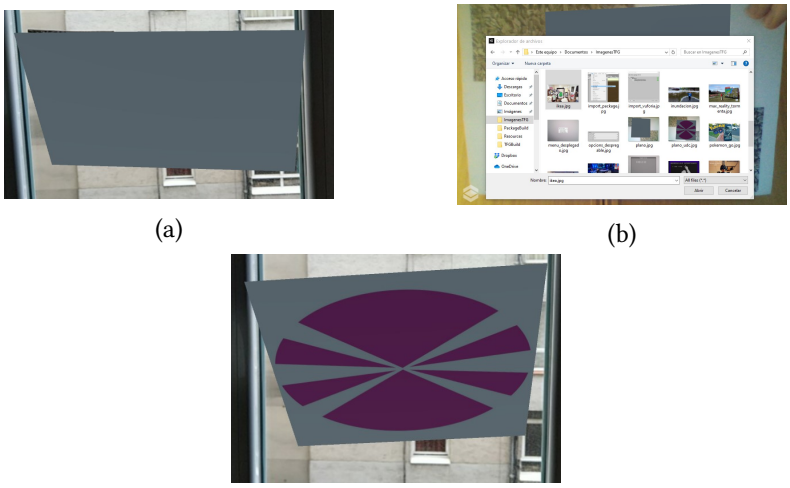
Proxección de elemento multimedia

Se o usuario quere cargar un elemento multimedia debe realizar os seguintes pasos:

1. Enfocar coa cámara os targets. Cando ambos targets sexan detectados o sistema xerará un plano entre eles na pantalla (ver figura B.4a).
2. Clicar sobre o plano. O sistema abrirá o explorador de arquivos, onde o usuario pode buscar que arquivo multimedia cargar (imaxe B.4b).
3. Ao seleccionar un arquivo, aparecerá proxectado sobre o plano (figura B.4c).

Cargar novos targets

Para cargar novos targets no sistema e iniciar a súa detección é necesario realizar o seguinte proceso:



(a) (b)

(c)

Figura B.4: Proceso para proxectar un elemento multimedia

1. Pulsar a Barra Espaciadora para facer aparecer un menú interactivo (figura B.5a).

2. Desplegar o menú e seleccionar a opción Cambiar target 1 ou Cambiar target 2 en función de que target se desexa modificar (imaxe B.5b).

3. Abrirase o explorador de arquivos dende onde o usuario poderá buscar a imaxe que desexe cargar (figura B.5c).

4. Ao seleccionar a imaxe o sistema pasará a detectar o novo target en lugar do antigo (figura B.5d).

Ver targets activos

Durante a execución do programa o usuario pode visualizar que targets están activos. Isto é, que targets están a ser detectados para a creación do espazo de proxección. O proceso para realizar esta acción é o que segue:

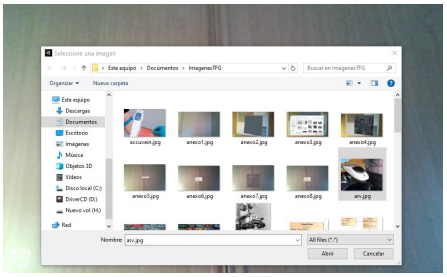
1. Pulsar a Barra Espaciadora para facer aparecer un menú interactivo (figura B.6a).

2. Desplegar o menú e seleccionar a opción Ver targets activos (imaxe B.6b).

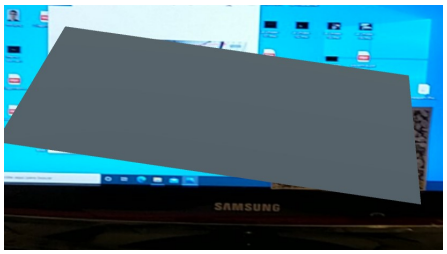
3. Aparecerá unha fiestra na que se mostran os dous targets activos (imaxe B.6c). Se o usuario fai clic no botón OK a fiestra desaparecerá.

(a)

(b)



(c)



(d)

Figura B.5: Proceso para cargar un novo target

(a)

(b)

(c)

Figura B.6: Proceso para ver os targets activos

Previsualización televisiva

Este sistema inclúe a posibilidade de visualizar nunha fiestra como se está a realizar a emisión de salida do programa de cara ás televisións. Para que o usuario poida asegurarse de que se está a emitir o que desexa debe realizar o seguinte proceso:

1. Pulsar a tecla A para facer aparecer unha fiestra na pantalla, na que se retransmite o contido que se está a emitir aumentado sen aqueles elementos que só o usuario da aplicación pode ver (figura B.7a).
2. Se o usuario desexa pode modificar o tamaño da fiestra facendo un clic e arrastrando sobre a fiestra (figura B.7b).
3. Pulsar de novo a tecla A para facer desaparecer a fiestra (figura B.7c).

(a) (b)

(c)

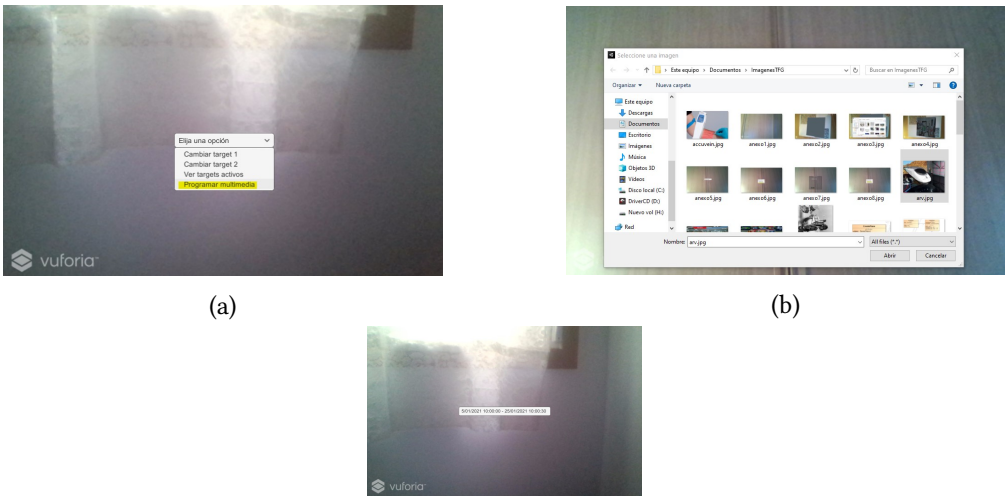
Figura B.7: Proceso para previsualizar a emisión televisiva

Programación de contidos multimedia

Se o usuario desexa, pode programar a emisión dos contidos multimedia con anterioridade á súa emisión. O proceso necesario é o seguinte:

1. Pulsar a Barra Espaciadora para facer aparecer o menú despregable e seleccionar a opción Programar Multimedia (imaxe B.8a).
2. Abrirase o explorador de arquivos para que o usuario seleccione cal é o contido que quere programar(imaxe B.8b).

3. Aparecerá un campo de texto onde o usuario debe introducir o momento de inicio e de finalización da reproducción do contido e pulsar Enter. Esta información debe ser introducida coa data e horas de comezo e remate separadas cun guión, é dicir, co seguinte formato: DD/MM/AAAA HH:MM:SS - DD/MM/AAAA HH:MM:SS (imaxe B.8c).



(a) (b)

(c)

Figura B.8: Proceso para programar o contido aumentado

B.1.2 Control do teclado

Nesta sección recollemos aquelas teclas do teclado que teñen funcionalidade no programa:

- Barra Espaciadora. Fai aparecer o menú interactivo.
- Tabulador. Fai desaparecer o menú interactivo.
- A. Fai aparecer e desaparecer a fiestra coa emisión televisiva.

B.2 Aplicación cliente de retransmisión

Aparte da aplicación de administración, o sistema conta cunha aplicación cliente adicional que funciona de forma concorrente coa outra.

Esta aplicación cliente recibe o vídeo aumentado xerado en directo na aplicación de administración e móstrao en pantalla. Este vídeo recibido non mostra ningún tipo de interface de usuario propio da aplicación de administración, como podemos comprobar na figura B.9. Posto que a recepción do vídeo entre aplicacións é en directo, teñen que estar ambas aplicacións en funcionamento á vez, isto é, a aplicación cliente de retransmisión non funcionará se non está en execución a aplicación de administración no mesmo equipo.

(a) Aplicación de administración

(b) Aplicación cliente

Figura B.9: Exemplo de execución das aplicacións de administración e cliente

Desta forma, unha vez aberta a aplicación e mentres estea en execución, o usuario poderá utilizar a imaxe que se visualiza en ela como aquela que vai retransmitir aos televidentes. Para iso bastaría con conectar a pantalla na que se está a executar o sistema en pantalla completa á central de retransmisión, unindo desta forma a imaxe desta aplicación á captada polas outras cámaras que se encontran no plató de televisión.

Relación de Acrónimos

RA *Realidade Aumentada.*

SDK *Software Development Kit.*

GPS *Global Positioning System.*

SLAM *Simultaneous Localization and Mapping.*

API *Application Programming Interface.*

CAD *Computer-aided design.*

Glosario

Script Documento que contén instrucións escritas nalgún código de programación.

Getter Método que permite obter o valor dunha variable.

Setter Método que permite modificar o valor dunha variable.

Bibliografía

[1] A. B. Craig, *Understanding Augmented Reality: Concepts and Applications*, 1st ed. Morgan Kaufmann Publishers Inc., 2013.

[2] I. E. Sutherland, "A head-mounted three dimensional display," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I.* American Federation of Information Processing Societies, 1968.

[3] T. Caudell and D. Mizell, "Augmented reality: An application of heads-up display technology to manual manufacturing processes," vol. 2, 1992, pp. 659 – 669.

[4] L. Rosenberg, "The use of virtual fixtures as perceptual overlays to enhance operator performance in remote environments," USAF Armstrong Laboratory, Wright-Patterson AFB OH, Tech. Rep. AL-TR-0089, 1992.

[5] F. Delgado, M. Abernathy, J. White, and W. Lowrey, "Real-time 3d flight guidance with terrain for the x-38," *Proceedings of SPIE - The International Society for Optical Engineering*, 1999.

[6] D. Wagner and D. Schmalstieg, "First steps towards handheld augmented reality," 2005, pp. 127– 135.

[7] [En liña]. Dispoñible en: https://www.ptc.com/en/products/augmented-reality/vuforia

[8] J. Xie, "Research on key technologies base unity3d game engine," in *2012 7th International Conference on Computer Science Education (ICCSE)*, 2012, pp. 695–699.

[9] [En liña]. Dispoñible en: https://www.zerodensity.tv/products/reality-engine/

[10] [En liña]. Dispoñible en: https://www.ibm.com/products/max-reality

[11] [En liña]. Dispoñible en: https://www.vizrt.com/products/viz-virtual-studio

[12] [En liña]. Dispoñible en: https://www.brainstorm3d.com/

[13] [En liña]. Dispoñible en: https://rtsw.co.uk/tog-vr/