UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

PRODUCCIÓN DE CONTENIDO EN VIVO MULTIDISPOSITIVO Y MULTIPLATAFORMA

Trabajo de Fin de Grado
Realizado en la
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
Por
Ignacio Reimat Corbella

En cumplimiento parcial de los requisitos para el grado en Ingeniería en Sistemas Audiovisuales

Tutor: Josep Ramon Casas Pla
Director: Juan Antonio Núñez Aragón

Barcelona, Enero 2017

Abstract

Television has to be reinvented as Technology advances. Virtual reality devices begin to reach the market introducing new possibilities in the audiovisual world.
The goal of this project is to create a live content production tool that mixes omnidirectional videos with directive ones, to be able to generate audiovisual experiences through multiple devices (TV, tablet, virtual reality helmet). We want to extend the usual television experience introducing new available devices to the audience. This Final Degree Project is part of the European project ImmersiaTV that aims to create a more immersive television experience by offering additional content, played in multiple devices, and synchronized in time. In this context, this project adapts some of the tools developed in ImmersiaTV, currently used to produce on demand content, to allow the production and distribution of live content.

Resum

La televisió ha de reinventar-se a mesura que la tecnologia avança. Els dispositius de realitat virtual comencen a arribar als mercats introduint noves possibilitats al món audiovisual.
L'objectiu d'aquest projecte es crear una eina de producció de contingut en viu que barregi vídeos omnidireccionals amb vídeos directius, per poder generar experiències audiovisuals a través de múltiples dispositius (TV, tauleta, casc de realitat virtual). Volem així estendre l'experiència televisiva habitual introduint els nous dispositius disponibles a l'audiència. Aquest treball forma part del projecte europeu ImmersiaTV que pretén crear una experiència televisiva més immersiva oferint contingut addicional, reproduïble en múltiples dispositius, i que aquests estiguin sincronitzats en temps. En aquest context, aquest projecte adapta algunes de les eines de ImmersiaTV, fins al moment utilitzades per a produir contingut sota demanda, per a permetre la producció i distribució de contingut en directe.

Resumen

La televisión tiene que reinventarse a medida que la tecnología avanza. Los dispositivos de realidad virtual empiezan a llegar al mercado introduciendo nuevas posibilidades en el mundo audiovisual.
El objetivo de este proyecto es crear una herramienta de producción de contenido en vivo que mezcle vídeos omnidireccionales con vídeos directivos, para poder generar experiencias audiovisuales a través de múltiples dispositivos (TV, tableta, casco de realidad virtual). Queremos así extender la experiencia televisiva habitual introduciendo los nuevos dispositivos disponibles a la audiencia. Este trabajo forma parte del proyecto europeo ImmersiaTV que pretende crear una experiencia televisiva más inmersiva ofreciendo contenido adicional, reproducible en múltiples dispositivos, y que estos estén sincronizados en tiempo. En este contexto, este proyecto adapta algunas de las herramientas de ImmersiaTV, hasta el momento utilizadas para producir contenido bajo demanda, para permitir la producción y distribución de contenido en directo.

Agradecimientos

En primer lugar, dar gracias a la Fundació i2Cat por darme la oportunidad de realizar este trabajo en sus instalaciones y formar parte del equipo de desarrollo de ImmersiaTV.

En segundo lugar agradecer el papel de Juan Antonio Núñez Aragón quien se ha encargado de realizar el seguimiento y supervisión del trabajo desde i2Cat.

En tercer lugar quería dar las gracias a Josep Ramon Casas Pla por su colaboración como supervisor y tutor de este trabajo.

En último lugar, dar gracias a mi familia por su apoyo y por brindarme la oportunidad de realizar estos estudios.

Historial de revisiones y registro de aprobación

Revisión	Fecha	Propósito
0	7/12/2016	Creación del documento
1	4/01/2017	Primera Revisión del documento
2	5/01/2017	Modificación del documento
3	6/01/2017	Revisión del documento
4	10/01/2017	Primer borrador completo
5	12/10/2017	Revisión final
5	13/10/2017	Versión final

Lista de distribución del documento

Nombre	e-mail
Ignacio Reimat Corbella	nacho_reimat@hotmail.com
Juan Antonio Núñez Aragón	juan.antonio.nunez@i2cat.net
Josep Ramon Casas Pla	josep.ramon.casas@upc.edu

Escrito por:		Revisado y aprobado por:	
Fecha	7/12/2016	Fecha	12/01/2017
Nombre	Ignacio Reimat Corbella	Nombre	Juan Antonio Núñez Aragón
Posición	Autor del proyecto	Posición	Supervisor del proyecto

Tabla de contenido

Lista de figuras

Lista de tablas

Acrónimos

CSS	Cascading Style Sheets
DASH	Dynamic Adaptive Streaming over HTTP
GUB	GStreamer-Unity-Bridge
GUI	Guide User Interface
HMD	Head Mounted Display
HLS	HTTP Live Streaming
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
I+D+i	Investigación Desarrollo e internet
LGPL	Lesser General Public License
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group
RTP	Real-Time Protocol
RTCP	Real-Time Control Protocol
RTSP	Real-Time Streaming Protocol
RTMP	Real-Time Messaging Protocol
SRD	Spatial Relationship Description
TCP	Transport Control Protocol
UDP	User Datagram Protocol
XML	Extensible Markup Language

Introducción

Este trabajo se ha realizado en la Fundación i2CAT y forma parte del proyecto europeo ImmersiaTV [1] liderado por la unidad de Media e Internet e iniciado en el mes de enero del 2016. La Fundación i2CAT es un centro de investigación e innovación sin ánimo de lucro que promueve actividades I+D+i orientadas a la generación de arquitecturas, aplicaciones y servicios de Internet. i2CAT fomenta la cooperación entre distintas empresas, administraciones públicas, el entorno académico y los usuarios finales.

El proyecto ImmersiaTV

En la actualidad, la mayoría de consumidores de televisión europeos ven la televisión en un entorno multi dispositivo. El uso de segundas pantallas (teléfonos inteligentes, tabletas, ordenadores) para chequear información del evento televisivo divide la atención entre éstas y la televisión. Aunque muchos broadcasters han intentado orquestar distintas plataformas para complementar a la televisión, su éxito se ha visto limitado, en parte, debido a la diversidad de formatos en que se entrega esta información (aplicaciones web, aplicaciones móviles, redes sociales, etc).

La llegada de dispositivos inmersivos como las gafas de realidad virtual, o por sus siglas en inglés HMD (Head Mounted Display), ha introducido nuevas posibilidades, pero también nuevos retos. Los dispositivos inmersivos proporcionan una experiencia totalmente distinta y gracias a ellos tendremos que convertir la narrativa audiovisual. Por ejemplo el recurso de cambiar de plano o de cámara que constituye la base del lenguaje cinematográfico, no funciona bien en estos dispositivos, ya que para el usuario representa un cambio total en el mundo que lo rodea, eso implica una pérdida de referencias espaciales, y puede provocar mareos.

En paralelo, la evolución de los dispositivos de captura ha transformado la producción y difusión de vídeo omnidireccional pasando de ser una prueba experimental a una posibilidad de la industria. Además los avances en la generación de gráficos por ordenador han posibilitado la fusión de contenido sintético 3D y vídeo omnidireccional.

ImmersiaTV tiene el objetivo de crear un nuevo formato audiovisual [2] basado en vídeo omnidireccional, así como el software necesario para su producción y distribución. Este formato audiovisual permitirá al usuario acceder a una experiencia coherente a través de dispositivos de realidad virtual, tabletas y televisión tradicional, en lugar de tener la atención dividida entre las diferentes pantallas. ImmersiaTV integra un bloque de producción de contenido junto a herramientas de codificación y difusión adaptadas a este nuevo tipo de contenido y demostrará la viabilidad de este formato audiovisual en tres pruebas piloto, teniendo en cuenta los requerimientos tanto de la difusión de contenido bajo demanda como en directo. El primer piloto mostrará el caso de vídeo *On demand*, el segundo se centrará en el caso *en vivo* y el tercero añadirá nuevos niveles de interactividad como interacción entre dispositivos y transiciones más complejas.

Objetivo del trabajo

El trabajo descrito en este informe se inicia justo después de la presentación del primer piloto del proyecto ImmersiaTV (para vídeo bajo demanda) por lo que nuestro objetivo es trasladar el bloque de producción de contenido de ImmersiaTV de vídeo bajo demanda a un escenario en vivo, asumiendo sincronización entre los distintos flujos audiovisuales. Es decir, crear una herramienta multiplataforma que permita trabajar con fuentes de vídeo en vivo, editar la composición de una escena omnidireccional y reproducirla en distintos dispositivos. Por lo tanto se centra en la preparación de las herramientas para llevar a cabo el segundo piloto, el caso *en vivo* de ImmersiaTV.

Dado que el contenido a editar es en vivo, trataremos toda la cadena de montaje. Empezaremos con la captura de vídeo plano y omnidireccional para crear los flujos de vídeo que pasarán por la herramienta de edición que tenemos que implementar. La salida de esta herramienta será publicada en un servidor de modo que la cadena finalizará en la reproducción del resultado obtenido desde distintos dispositivos.

Tras esta breve introducción demos un repaso a la estructura del presente documento.
En el capítulo *1. Conceptos básicos y tecnologías utilizadas* repasamos los conceptos básicos y las tecnologías empleadas en el trabajo. En el capítulo *2. Desarrollo del proyecto* se describe su desarrollo, empezando por indicar los requisitos específicos del proyecto a partir de la situación del proyecto ImmersiaTV cuando se inició el trabajo. Se realiza una exploración de las distintas soluciones que dan paso al diseño y a la implementación de la solución. En el capítulo *3. Resultados* se listan los resultados obtenidos y en el capítulo *4. Conclusiones* y líneas futuras se presentan las conclusiones y líneas futuras del trabajo en el contexto de ImmersiaTV. Finalmente en el capítulo *5. Presupuesto* se puede ver una aproximación de los costes que ha supuesto la realización del trabajo.

1. Conceptos básicos y tecnologías utilizadas

A lo largo del documento aparecen distintos términos o herramientas específicas del contexto tecnológico en el que se sitúa el proyecto. Para facilitar su comprensión vamos a definir algunos conceptos y herramientas utilizadas.

1.1 Definiciones

1.1.1 Vídeo omnidireccional

El vídeo omnidireccional también llamado vídeo esférico o 360º es una escena panorámica que permite la visualización en todas direcciones donde el usuario se encuentra en el centro y puede ejercer control sobre ella. Puede ser tanto una grabación del mundo real como una escena generada por ordenador.

Para la grabación de escenas 360º se necesita una cámara con más de una lente para poder abarcar toda la escena. Hay cámaras compactas que realizan el procesado en el propio dispositivo y cuya salida es la escena omnidireccional, aunque el caso más típico es el uso de más de una cámara colocadas de forma circular y por lo tanto es necesario un procesado de imagen que una las distintas partes de la escena. Este proceso se define como *stitching* o cosido.

1.1.2 Transmisión de eventos en vivo

El streaming o retransmisión consiste en la distribución digital de contenido multimedia a través de la red con el objetivo de reproducir el contenido inmediatamente de forma continua.

Si sólo tenemos en cuenta el contenido podemos diferenciar dos tipos de streaming:

- Streaming de contenido capturado en tiempo real. Este es el streaming más "*puro"* o el que se ajusta más a la definición de streaming. En este caso se suelen utilizar protocolos como RTP+RTCP/RTSP y RTMP ya que permiten reducir el retraso de transmisión.
- Streaming de contenido pregrabado también llamado *bajo demanda* donde se transmite/descarga un archivo de forma progresiva.

Además según la tasa de bits podemos diferenciar entre:

- Streaming "estático". La tasa de bits es fija y la emisión ocupa siempre un ancho de banda determinado.
- Streaming adaptativo. La tasa de bits varía y la emisión se adapta al ancho de banda de la red. Encontramos soluciones comerciales como HTTP Dynamic Streaming de Adobe, HLS (HTTP Live Streaming) de Apple o Smooth Streaming de Microsoft y también soluciones estandarizadas internacionalmente como MPEG-DASH.

1.2 Tecnologías utilizadas

1.2.1 RTP

RTP [3] (de sus siglas en inglés *Real-Time Protocol*) es un protocolo orientado a transmitir datos en tiempo real. Trabaja sobre UDP y no garantiza la entrega de los paquetes. Para el control de errores se suele combinar con el *Protocolo de Control en Tiempo Real* (RTCP).

1.2.2 RTSP

RTSP [4] (de sus siglas en inglés *Real-Time Streaming Protocol*) es un protocolo utilizado para establecer y controlar sesiones multimedia cliente-servidor en tiempo real. Estas conexiones se realizan sobre TCP, aunque también es posible trabajar sobre UDP, y la comunicación se realiza en ambas direcciones. El protocolo no envía información, solamente realiza un control sobre los datos transmitidos. Normalmente se combina con RTP para enviar los streams.

1.2.3 RTMP

Adobe RTMP [5] (de sus siglas en inglés *Real-Time Messaging Protocol*) es un protocolo de transmisión basado en TCP desarrollado por Adobe Systems. Su objetivo es transmitir audio vídeo y datos entre un servidor Flash y un reproductor Flash. La comunicación se realiza en ambos sentidos y en tiempo real. Es válido tanto para contenido pregrabado como contenido en vivo.

1.2.4 MPEG-DASH

El *estándar internacional de transmisión de vídeo adaptativo y dinámico sobre HTTP*, también conocido como MPEG-DASH [6] (de sus siglas en inglés Dynamic Adaptive Streaming over HTTP), define un modelo de streaming con tasa de bits adaptativa. Permite la reproducción en tiempo real de contenidos multimedia de alta calidad mediante el uso de servidores HTTP convencionales.

Esta técnica es válida tanto para contenido *en vivo* como *bajo demanda* y proporciona contenido en distintas calidades (tasas de bits) y lenguajes con el fin de optimizar el ancho de banda y adaptarse a la situación de la red.

Por ejemplo, si un dispositivo solamente puede reproducir contenido de baja calidad no tiene sentido enviarle una calidad más alta, o si se quiere escuchar el vídeo en inglés no tiene sentido que se envíe la pista de audio en castellano. Del mismo modo, si la red está congestionada nos puede interesar recibir el contenido en una calidad más baja en vez de recibirlo en alta calidad con retraso o perdiendo información.

El streaming DASH se realiza sobre HTTP y comienza con la descarga de un archivo con metadatos llamado MPD (Multimedia Presentation Description) por sus siglas en inglés, donde se encuentran definidos los distintos flujos de audio, vídeo y datos, con sus distintas calidades y lenguajes disponibles. De este modo el cliente puede decidir concretamente el contenido que desea descargar teniendo en cuenta el tipo de dispositivo, el ancho de banda

de la red y las preferencias del usuario. En la **Figura. 1** vemos un esquema de la interacción que se realiza entre el Cliente y el Servidor DASH.

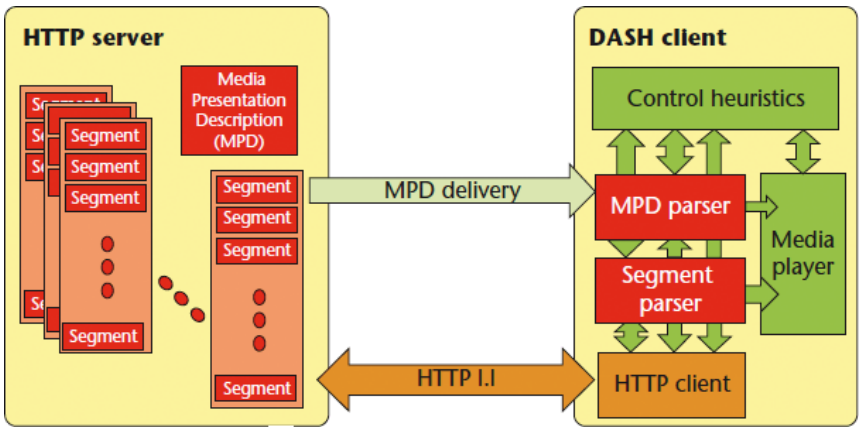


Figura. 1 *Interacción Cliente-Servidor DASH [7]*

El modelo de fichero MPD está escrito en formato XML por lo que sigue una estructura en forma de árbol. De modo que cada MPD puede contener múltiples periodos, cada periodo múltiples contenidos (Adaptation sets), cada contenido múltiples representaciones y cada representación múltiples segmentos. Vemos un ejemplo de esta estructura en la **Figura. 2**.

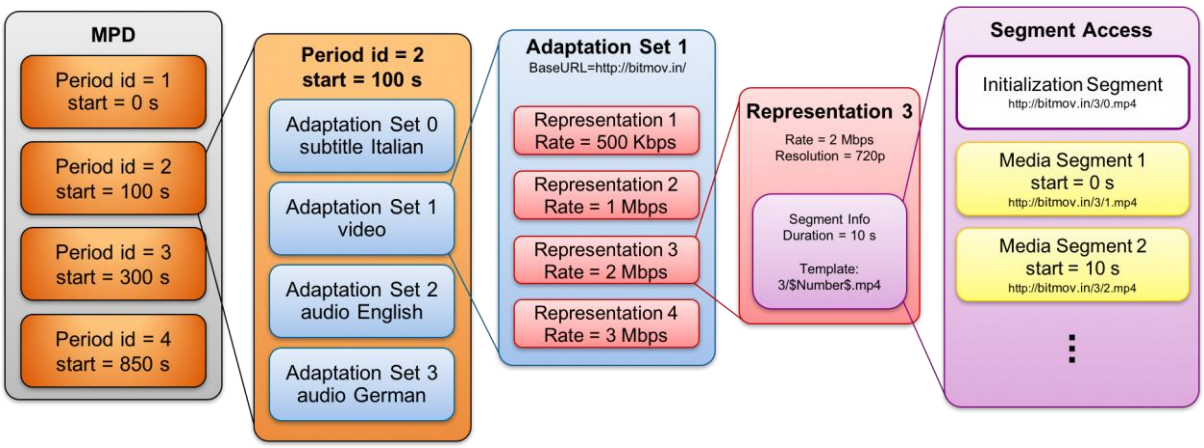


Figura. 2 *Modelo de Adaptation Sets de un fichero MPD [8]*

Hay que tener en cuenta que para poder disponer de vídeo en distintas calidades éstas han de ser generadas, por lo que este tipo de streaming introduce un retardo considerable para los casos en vivo, en ausencia de un esfuerzo considerable de paralelización, coordinación y sincronización de los múltiples niveles de compresión.

Al estándar se le han añadido recientemente técnicas muy útiles para contenido omnidireccional como es el caso de MPEG-DASH SRD [9] (Spatial Relationship Description) donde a parte de la resolución, canal de audio, subtítulos... es posible segmentar la escena de modo que solamente enviamos aquella parte que está siendo visualizada. De este modo para el caso de escenas omnidireccionales podríamos ahorrarnos enviar la información de la escena que queda en el espacio no visible por el usuario.

1.2.5 GStreamer

GStreamer [10] es un framework basado en software libre y multiplataforma que permite crear aplicaciones audiovisuales. Su funcionamiento está basado en la construcción de cadenas de elementos (en inglés pipelines). Estas cadenas empiezan con una fuente (source) y finalizan en un sumidero (sink). Entre la fuente y el sumidero se suelen colocar uno o más elementos de tipo filtro que modifican determinadas características del flujo que procesan. En la **Figura. 3** podemos ver el montaje de una pipeline que reproduce un archivo de vídeo.

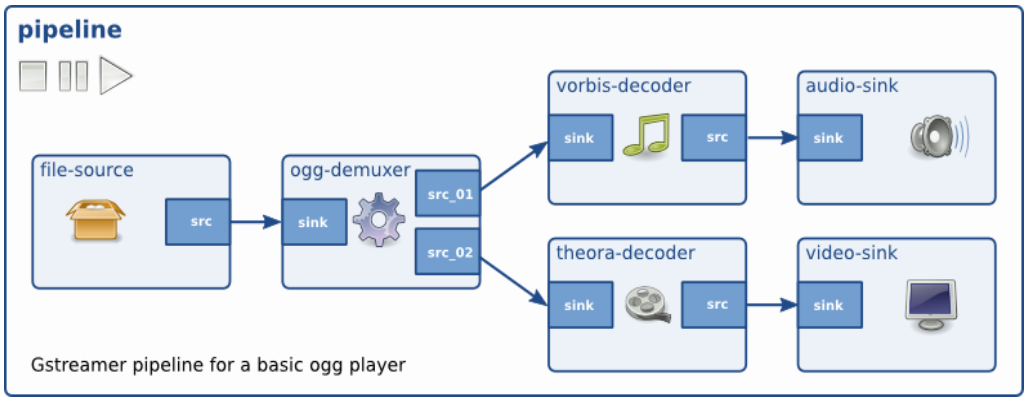


Figura. 3 *Esquema de una pipeline creada con GStreamer [11]*

Estas cadenas de elementos se pueden programar mediante las funciones provistas en las distintas librerías que componen GStreamer o mediante scripts aunque el programa dispone también de un comando llamado gst-launch para crearlas directamente desde la consola.

1.2.6 LiveMediaStreamer

LiveMediaStreamer [12] es un framework multimedia que permite manipular streams de audio y vídeo en tiempo real utilizando distintas configuraciones. Es un software con licencia LGPL creado por la fundación i2CAT y sólo corre en sistemas Linux aunque es posible utilizarlo desde Windows mediante el uso de contenedores Docker [13].

Al igual que GStreamer su funcionamiento consiste en crear pipelines formados por la concatenación de elementos de tipo fuente, filtros y sumidero.
En la **Figura. 4** vemos el esquema de un posible montaje de una pipeline donde podemos apreciar la estructura *fuente-filtros-sumidero así como el control de eventos*.

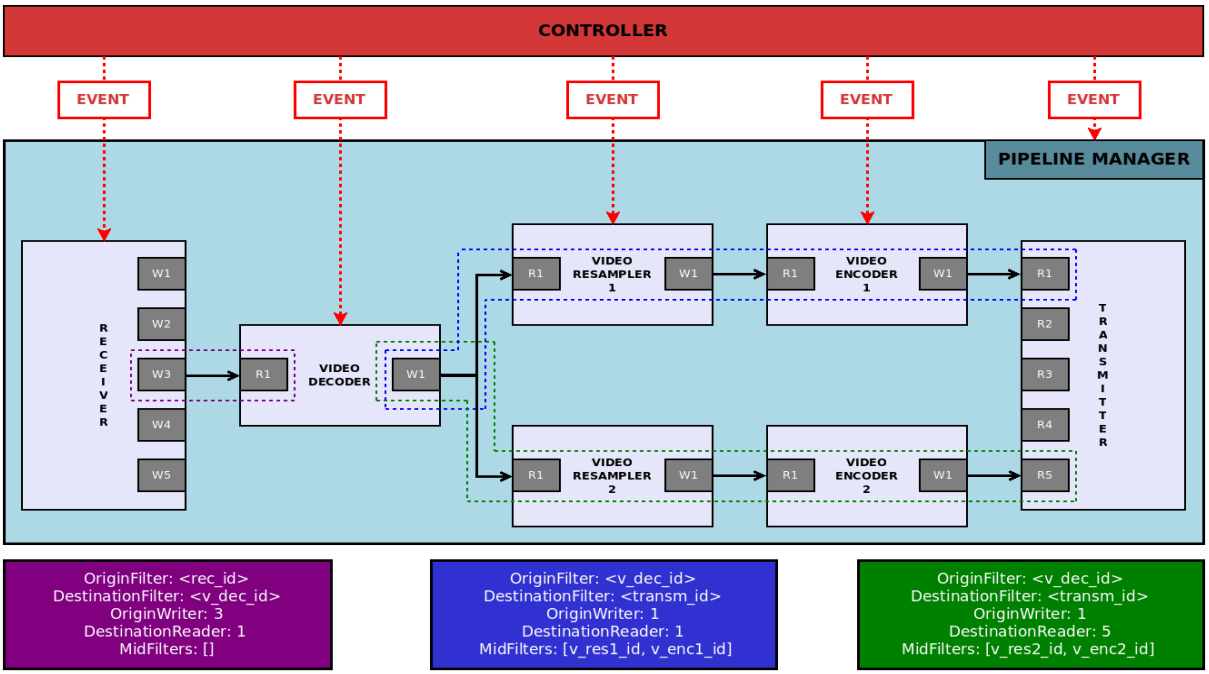


Figura. 4 *Ejemplo de pipeline de LiveMediaStreamer [14]*

1.2.7 Unity3D

Unity3D [15] es un motor de videojuego multiplataforma creado por Unity Technologies. Especialmente diseñado para la creación de juegos y contenidos 3D interactivos.
En el proyecto ImmersiaTV se usa Unity3D para generar los entornos omnidireccionales sintéticos y de test, y es el que se encarga de pintar los distintos streams de vídeo en esferas.

A su vez estos streams són gestionados previamente mediante GStreamer. Por lo que entre estas dos herramientas se utiliza el GStreamer-Unity Bridge (GUB). Tal como indica su nombre en inglés el GUB es un programa que actúa de puente entre las pipelines de GStreamer y el software Unity3D. La solución ha sido implementada por el equipo de Media de i2Cat especialmente para el proyecto ImmersiaTV aunque se ha publicado con código abierto bajo licencia LGPL para que todo el mundo pueda utilizarlo y modificarlo.

1.2.8 NGINX

Nginx [16] (pronunciado “Engine X”) es un servidor web de alto rendimiento. Es multiplataforma y está basado en software libre y de código abierto, bajo licencia BSD. Se cuenta entre los servidores web más utilizados del mundo y es utilizado por varios sitios web conocidos como WordPress, Netflix, GitHub, etc.

El servidor se puede utilizar también para publicar ficheros o como servidor de streaming, además de una larga lista de posibilidades. Y al ser de código abierto es posible adaptarlo a tu gusto.

1.2.9 Wowza Streaming Engine

Wowza Streaming Engine [17] es un servidor de streaming desarrollado por Wowza Media Systems. Aunque es una solución comercial cuenta con una versión de prueba de 6 meses un poco limitada respeto a la versión de pago, pero válida para realizar distintas pruebas.

El servidor es utilizado tanto para transmitir contenido "*en vivo*" como "*bajo demanda*". Además cuenta con un transcodificador que trabaja en tiempo real que decodifica los flujos de entrada y los re-codifica en múltiples formatos como MPEG-DASH, Apple HLS, Adobe RTMP, Adobe HDS, Microsoft Smooth Streaming y RTSP/RTP.

1.2.10 RTMP Camera

RTMP Camera es una aplicación desarrollada por MIV Dev Team para dispositivos Android. A través de su sencilla interfaz es posible realizar la captura de vídeo desde cualquier dispositivo y elegir el punto de publicación del flujo generado.

1.2.11 VahanaVR

VahanaVR [18] es un software de Vídeo-Stitch (partner tecnológico del proyecto ImmersiaTV) que permite la captura multi cámara, la realización del stitching de las distintas cámaras para generar una escena de vídeo 360º, la visualización en local y la transmisión de la escena en tiempo real a través de un único flujo.

Entre los posibles formatos de salida se encuentran las opciones de generar un fichero de vídeo, realizar streaming RTMP, publicar en Youtube o visualizarlo localmente en Oculus Rift y HTC Vive.

2. Desarrollo del proyecto

2.1 Punto de partida

Como se ha comentado en la introducción, el objetivo del proyecto es trasladar el bloque de producción de contenidos de ImmersiaTV (hasta ahora para vídeo bajo demanda) a un escenario en vivo, asumiendo que existe sincronización entre fuentes.

Para ponernos en situación y dejar claro lo que ha aportado el presente trabajo al proyecto ImmersiaTV, hemos creído conveniente explicar brevemente el estado en que se encontraba el proyecto europeo antes de nuestra incorporación.

2.1.1 Generación y producción de contenido

A día de hoy el proyecto ImmersiaTV genera sus contenidos a partir de vídeos planos y omnidireccionales previamente grabados. Para la edición de las escenas se utiliza el software Adobe Premiere Pro. Para ello se han implementado varias herramientas que añaden funcionalidades extra al programa base. Llamamos a estas extensiones plugins.

Entre estos plugins encontramos transiciones de vídeo y los efectos de vídeo Rectangular*Portal y SphericalPortal* que permiten la inserción de vídeos secundarios sobre el vídeo omnidireccional principal. A este tipo de inserción lo llamamos portales.

También existe un plugin que se encarga de generar un archivo de metadatos y renderizar las pistas para la posterior publicación del contenido en un servidor web.

2.1.2 Reproducción del contenido

La reproducción de contenido se realiza a través de una aplicación que incorpora un reproductor especialmente desarrollado para el proyecto. Este reproductor está basado en Unity3D usando GStreamer para la reproducción de vídeo y audio, por lo que se utiliza el anteriormente mencionado GUB para unir ambas tecnologías. La aplicación lee los ficheros de metadatos publicados en un servidor y pinta la escena descrita.

2.2 Requisitos

A partir de la situación de ImmersiaTV tras la presentación del primer piloto de video bajo demanda, se establecieron los requisitos específicos del presente trabajo con el objetivo genérico de preparar las herramientas para el segundo piloto de vídeo en vivo, listados a continuación.

2.2.1 Captura de vídeo

- Los flujos habrán de ser sincronizados en origen y tener una referencia temporal que pueda servir para ajustar posibles desincronizaciones en la reproducción
- La salida de audio y vídeo deberá tener la menor latencia posible para poder ser distribuida desde un servidor que almacene flujos MPEG-DASH

2.2.2 Herramienta de edición

De forma más precisa, lo que queremos es implementar una herramienta multiplataforma que permita la edición del diseño de una escena omnidireccional. Eso implica

- Selección del vídeo omnidireccional “base”.
- Inserción de vídeos directivos y omnidireccionales en forma de portales.
- Asignación de la posición de estas inserciones sobre la escena.
- Asignación del sistema de referencia (si la posición es relativa al mundo virtual o al usuario).
- Asignación del tamaño (ancho y alto) de las inserciones.

Lo ideal sería poder hacer esta edición en un dispositivo con una pantalla táctil para facilitar su interacción, aunque opciones alternativas como un ratón convencional serán consideradas como posibles estrategias de implementación.

2.2.3 Metadatos

Para transmitir la información de la escena generada será necesaria la generación de un fichero de metadatos, siguiendo la estructura de los metadatos de ImmersiaTV descritos en el Apéndice 1. Metadatos de ImmersiaTV, donde se especifiquen las distintas fuentes de vídeo, posiciones, tamaño y referencias asignadas en la edición. La herramienta debe ser capaz de generar estos metadatos en tiempo real recogiendo los parámetros introducidos en la interfaz de usuario.

La salida de la producción en vivo engloba varios streams de audio, vídeo y metadatos. Por lo que será importante de cara a la retransmisión cumplir lo siguiente:

- Tanto los metadatos como los flujos de audio y vídeo deben proveerse por separado.
- Los metadatos deben ser actualizados regularmente imitando el comportamiento del procesamiento de un manifiesto MPEG-DASH en vivo.

2.2.4 Reproducción

Es necesario poder hacer una previsualización en directo de la composición de la escena antes de que ésta sea publicada. Tanto la previsualización como la visualización final ha de ser válida para múltiples dispositivos, como tabletas y dispositivos HMD.

2.3 Exploración de soluciones

Previamente a la fase de diseño se han explorado y testeado distintas herramientas. Empezaremos probando aplicaciones de captura y retransmisión de vídeo, seguidas de herramientas de manipulación de streams y finalmente un par de servidores de publicación de contenidos.

2.3.1 Captura de vídeo plano

Hemos testeado la aplicación móvil *RTMP Camera* en un dispositivo Android. Esta aplicación además de ocupar poco espacio, y como se muestra en la **Figura. 5**, tiene una interfaz sencilla que permite configurar varios parámetros de captura de vídeo y audio así como el punto de publicación del flujo de salida.

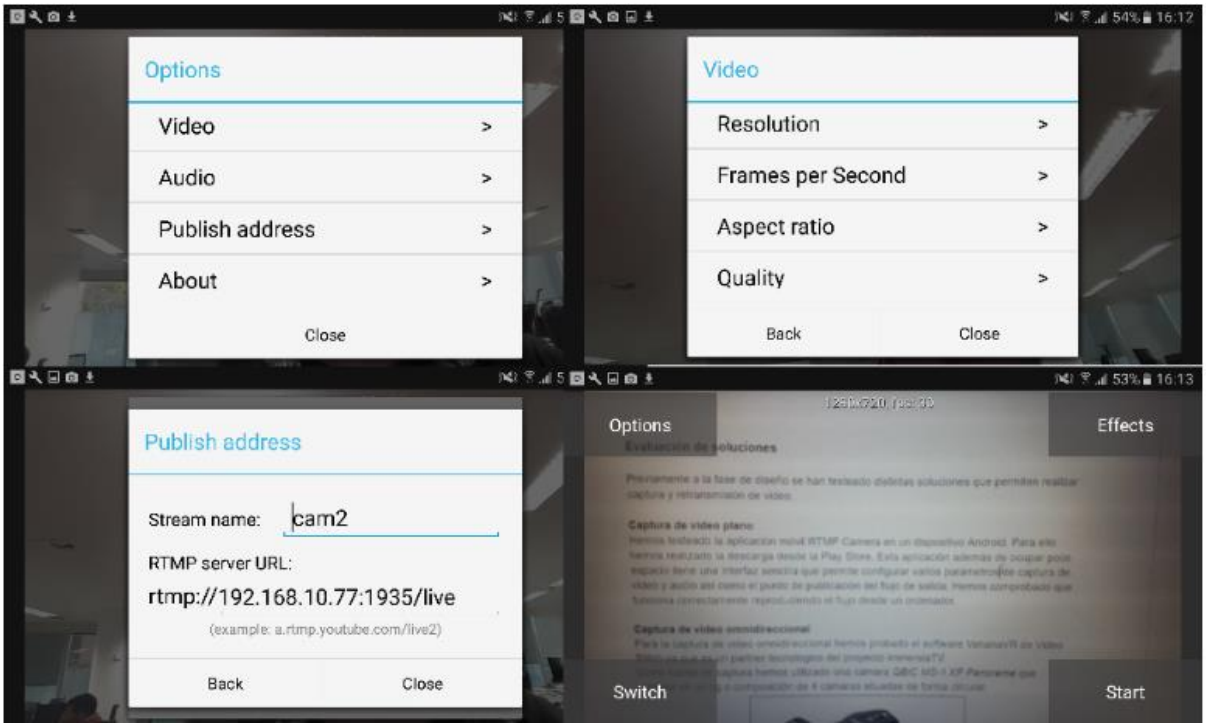


Figura. 5 *Capturas de pantalla de la aplicación RTMP Camera*

Hemos comprobado que la aplicación funciona correctamente enviando el flujo RTMP a nuestro servidor Nginx y reproduciendo el flujo desde el reproductor VLC.

2.3.2 Captura de vídeo omnidireccional

Para la captura de vídeo omnidireccional hemos probado el software VahanaVR de Vídeo Stitch. Como fuente de captura hemos utilizado una cámara *QBiC MS-1 XP Panorama*. Tal como podemos apreciar en la siguiente figura, esta consiste en un rig o cabezal de 4 cámaras situadas de forma circular, como se aprecia en la **Figura. 6**. Las hemos conectado al equipo a través de cables HDMI mediante un conversor HDMI a USB 3.0.

Figura. 6 *Cámara QBIC MS-1 XP Panorama [19]*

En VahanaVR hemos generado un proyecto nuevo y hemos seleccionado como entradas las cuatro cámaras. El programa muestra las cuatro entradas por separado en el panel *Sources* tal como se muestra en la **Figura. 7**.

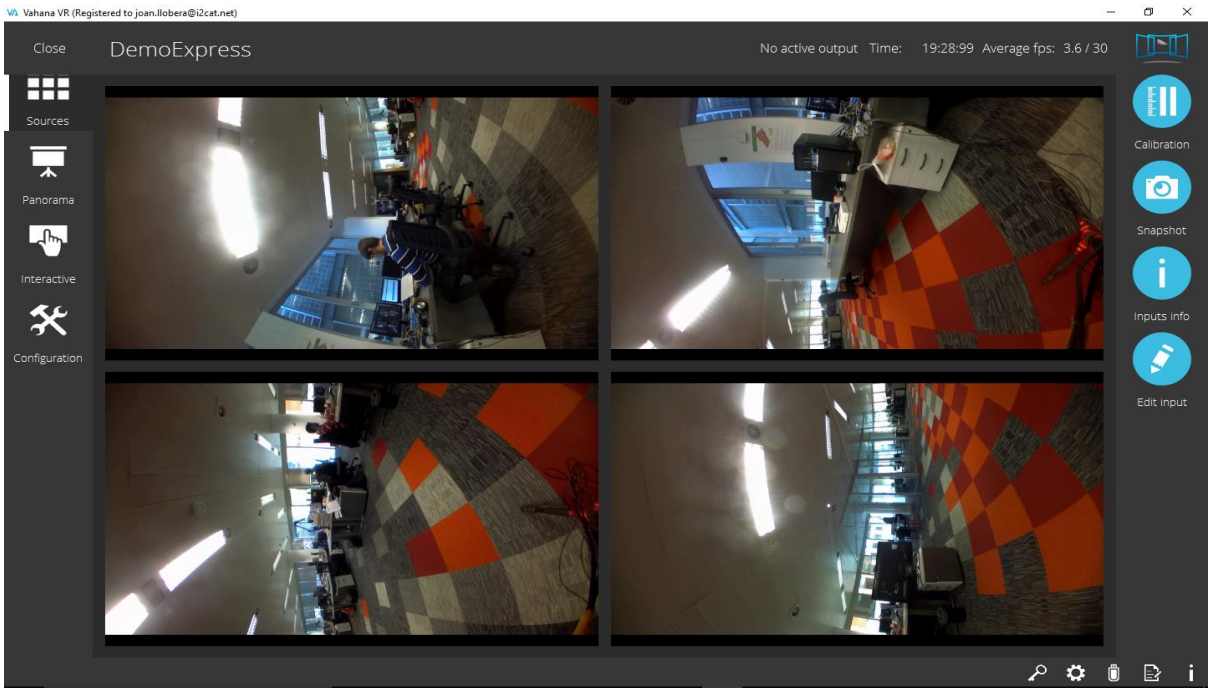


Figura. 7 *Vista del panel "Sources" en VahanaVR*

Hemos realizado la calibración automática que ofrece el mismo software y tras varios intentos se ha conseguido una calibración correcta de las 4 cámaras resultando un stitching bastante aceptable, como se puede ver en la **Figura. 8**, teniendo en cuenta que no hemos utilizado ningún patrón de calibración.

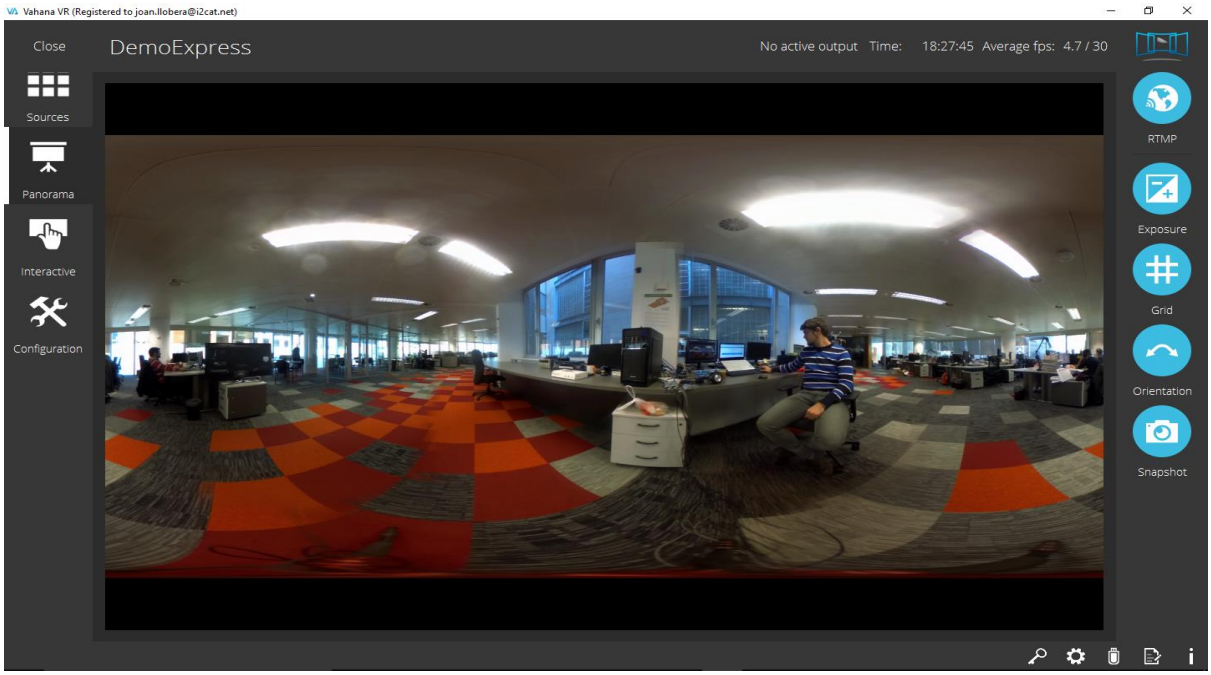


Figura. 8 *Vista del panel "Panorama" en VahanaVR*

Hemos configurado una salida en forma de flujo RTMP apuntando al servidor Wowza. Ajustando los parámetros tal como se muestran en la **Figura. 9** hemos logrado reducir el delay al máximo. Comprobamos que la transmisión es fluida reproduciendo el flujo desde otro equipo. Las salidas de Wowza en RTMP i RTSP presentan un delay de unos 2 segundos mientras que para los flujos DASH tenemos un delay de 8 segundos. Bastante aceptable.

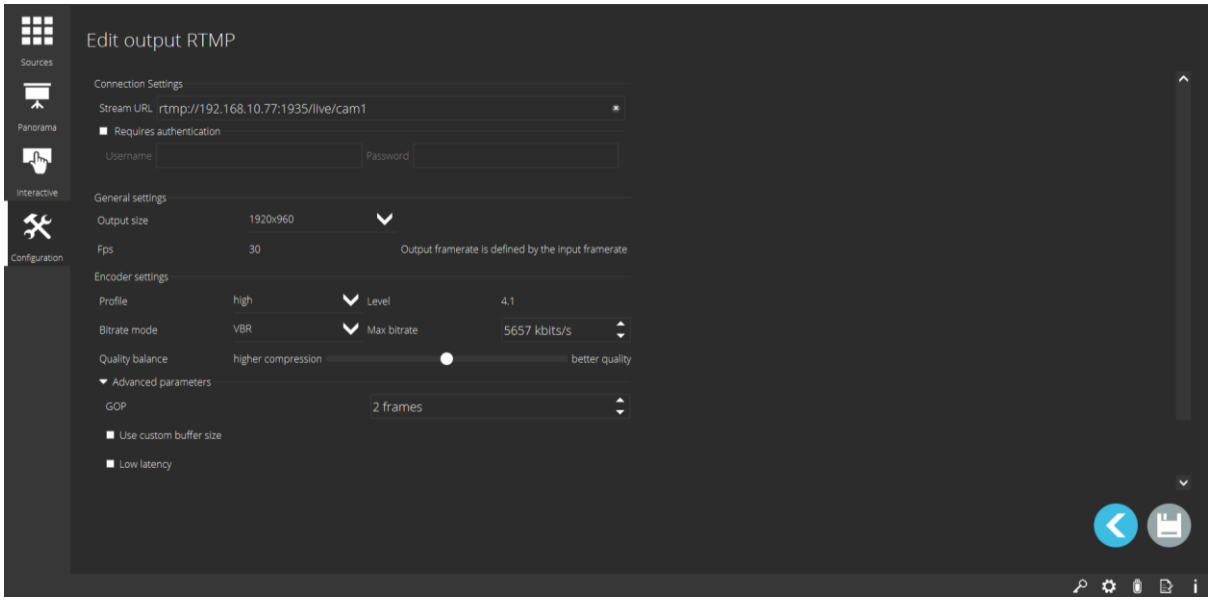


Figura. 9 *Vista del panel "Configuration-Outputs" en VahanaVR*

2.3.3 GStreamer

Hemos realizado varios tests para probar el funcionamiento de GStreamer. Por ejemplo la captura de vídeo a través de la webcam y el micrófono con su respectiva reproducción en local, la retransmisión de la salida en forma de flujo RTMP y la reproducción de un flujo RTMP.

Podemos ver este último caso en la **Figura. 10**, donde mediante los siguientes comandos y utilizando la aplicación *gst-launch,* hemos generado una pipeline que tiene como fuente un flujo RTMP seguido de los elementos de tipo filtro (un decodificador, una cola y un conversor de vídeo) y que finaliza en un elemento sumidero que muestra el vídeo por pantalla.

```
C:\Users\pc>gst-launch-1.0 rtmpsrc location="rtmp://192.168.10.77:1935/live/cam2" !
 decodebin ! queue ! videoconvert ! autovideosink
```

Figura. 10 *Ejecución de un comando de GStreamer mediante su gst-launch*

Después de realizar varias pruebas de funcionamiento nos damos cuenta de que el elemento que permite convertir los streams a formato MPEG-DASH todavía no ha sido implementado. Como es un requerimiento del proyecto, descartamos esta solución por el momento.

2.3.4 LiveMediaStreamer

Mediante LiveMediaStreamer hemos tratado de configurar un sistema que a partir de un flujo de entrada RTMP nos dé una salida DASH. Según la página web de LMS este sistema supuestamente funcionaba. Tras encontrarnos varios errores durante la configuración finalmente descubrimos que el software no ha sido mantenido y que hay que implementar todo el sistema de nuevo. Dado el poco tiempo que tenemos descartamos esta opción por el momento.

2.3.5 Nginx

Para la configuración del servidor de distribución de contenido, hemos modificado el archivo de configuración de Nginx para crear un servidor RTMP que escucha en el puerto 1935. Hemos creado una aplicación "*live*" donde se pueden publicar streams RTMP.

Hemos configurado una aplicación "*dash*" que convierte los streams de entrada RTMP a formato MPEG-DASH, para ello hemos generado el subdirectorio */dash* donde se publicarán los manifiestos y fragmentos generados.

Por último hemos generado el subdirectorio */live* que utilizaremos para publicar archivos y poder acceder a ellos desde distintos dispositivos.

```
rtmp {
    server {
        listen 1935;

        application live {
            live on;
        }
        application dash {
            dash on;
            live on;
            dash_path temp/dash;
        }
    }
}

http {
    server {
        listen 8084;

        location /dash {
            root temp;
            add_header Cache-Control no-cache;
            add_header 'Access-Control-Allow-Origin' '*';
        }
        location /live {
            root metadata;
            autoindex on;
        }
    }
}
```

Figura. 11 *Configuración del servidor Nginx*

Hemos obtenido buenos resultados para la aplicación RTMP "*live*" y la publicación de archivos en el directorio */live*, pero no estamos satisfechos con el funcionamiento de la aplicación "*dash*" ya que la reproducción es poco fluida.

2.3.6 Wowza Streaming Server

Para gestionar la distribución de flujos en directo y obtener los streams en formato RTSP, RTMP y MPEG-DASH hemos configurando varios flujos de entrada RTMP a través de la aplicación web Wowza Engine Manager. En la **Figura. 12** se muestra una captura de pantalla de la configuración de dicha aplicación donde los mejores resultados los hemos obtenido configurando el paquetizador con un tamaño de chunk de 2 segundos y 5 chunks por MPD. Hemos visualizado los distintos flujos de salida en varios dispositivos de manera fluida, así que de momento utilizaremos este servidor para realizar los tests.

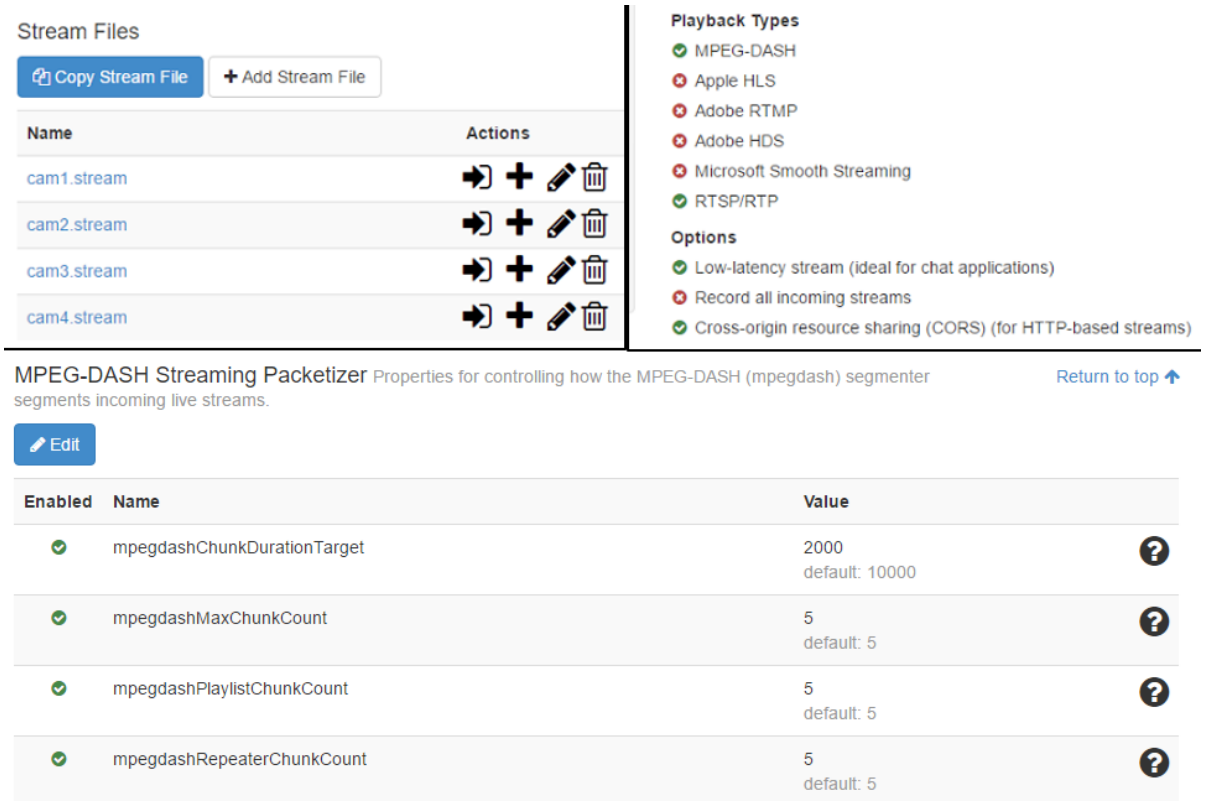


Figura. 12 *Configuración del servidor Wowza*

2.4 Diseño

Una vez concluida la fase de exploración de soluciones hemos diseñado un sistema como el que se muestra en la **Figura. 13**. Lo explicamos a continuación.

Para el bloque de captura utilizaremos la App móvil RTMP Camera para los vídeos planos y VahanaVR para el caso omnidireccional. De este modo todos los streams iniciales serán en formato RTMP y apuntarán a nuestro servidor Wowza.

Para el bloque de producción será necesario desarrollar una GUI. Lo ideal sería implementar una aplicación web para poder ejecutarla desde cualquier dispositivo, pero eso supone implementar también un reproductor de contenido omnidireccional interactivo que permita editar la escena. Descartamos esta opción por el momento ya que esta significaría emplear mucho tiempo y recursos, y no disponemos de ellos.

Hasta el momento se editaban todos los vídeos de contenido *On Demand* utilizando el software Adobe Premiere Pro [20]. Existen varios plugins de ImmersiaTV para la edición de portales y eso nos facilita las cosas. Por este motivo hemos decidido implementar otro plugin para Adobe Premiere orientado a la producción de escenas en vivo, que nos permita generar los metadatos. Por lo tanto la interfaz de usuario será un plugin para Adobe Premiere Pro

El bloque de distribución lo compondrán dos sub bloques. La preparación y codificación de DASH se realizará mediante el servidor Wowza Streaming Server y el servidor donde publicaremos los metadatos será Nginx.

Finalmente el bloque de reproducción se basará en la aplicación de ImmersiaTV que contiene el reproductor omnidireccional. Para ello será necesaria la modificación del GUB para que admita streams en directo ya que hasta momento solo leía ficheros. La adaptación la realizarán otros miembros del equipo de media de i2Cat.

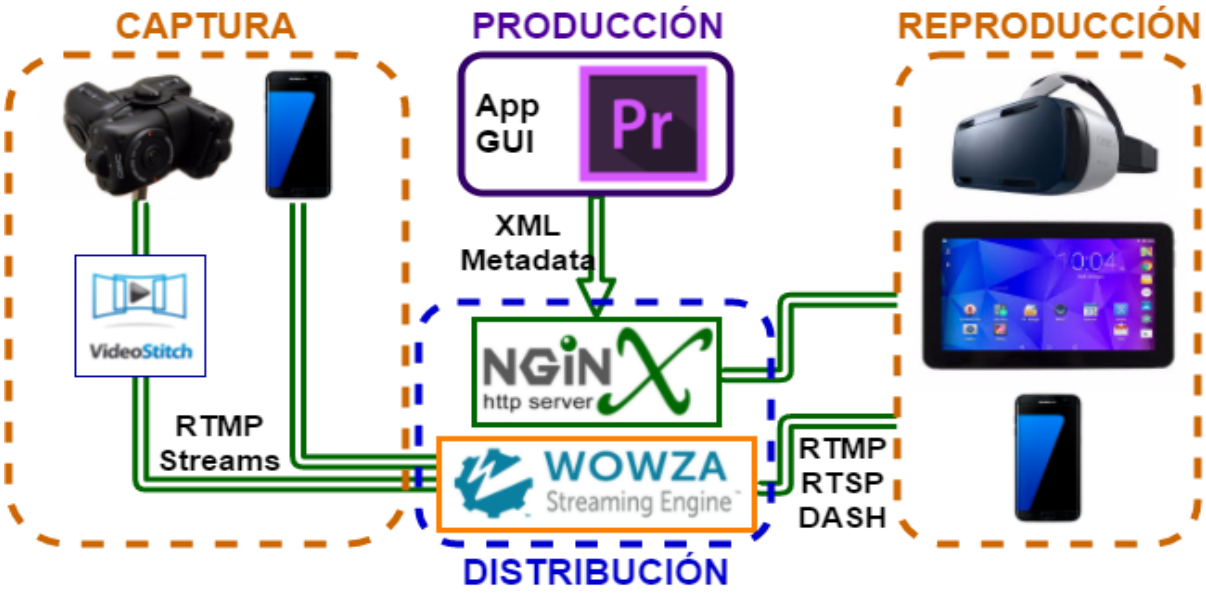


Figura. 13 *Diagrama del diseño propuesto*

2.5 Implementación

Para los bloques de producción, distribución y reproducción utilizaremos soluciones software existentes que ya hemos configurado en la fase de exploración. El bloque de producción es el que falta por implementar para el caso de vídeo en vivo en esta fase de preparación del segundo piloto de ImmersiaTV. Este bloque lo forma el plugin para Adobe Premiere Pro que es el que se encargará de producir la escena y generar los metadatos. Veámoslo con más detalle.

2.5.1 Plugin para Adobe Premiere Pro

Los plugins de Adobe Premiere Pro se implementan mediante el lenguaje de programación HTML5, para la capa gráfica, combinado con JavaScript para que el contenido a mostrar se genere dinámicamente. Por lo tanto el resultado es un documento HTML con su correspondiente hoja de estilos CSS y un archivo Javascript que gestiona la interactividad y se comunica con un archivo JSX en el que se encuentran las funciones básicas de Adobe Premiere Pro.

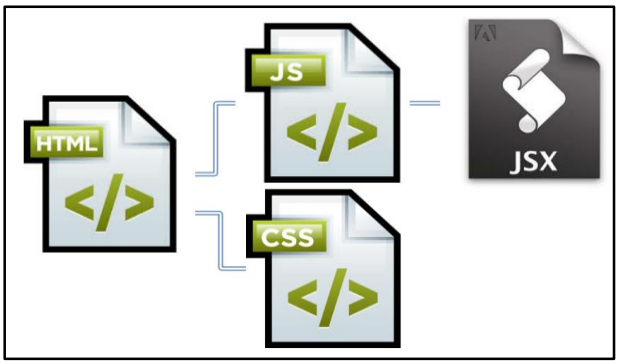


Figura. 14 *Esquema de ficheros que forman el plugin*

Hemos intentado mantener una estética parecida a la del propio programa Adobe Premiere Pro.

El plugin está formado por un panel inicial como el que se muestra en la **Figura. 15**. En este panel elegimos el número de cámaras que usaremos en la escena, la fecha y hora de inicio del evento y las rutas de publicación de los ficheros de previsualización y metadatos.

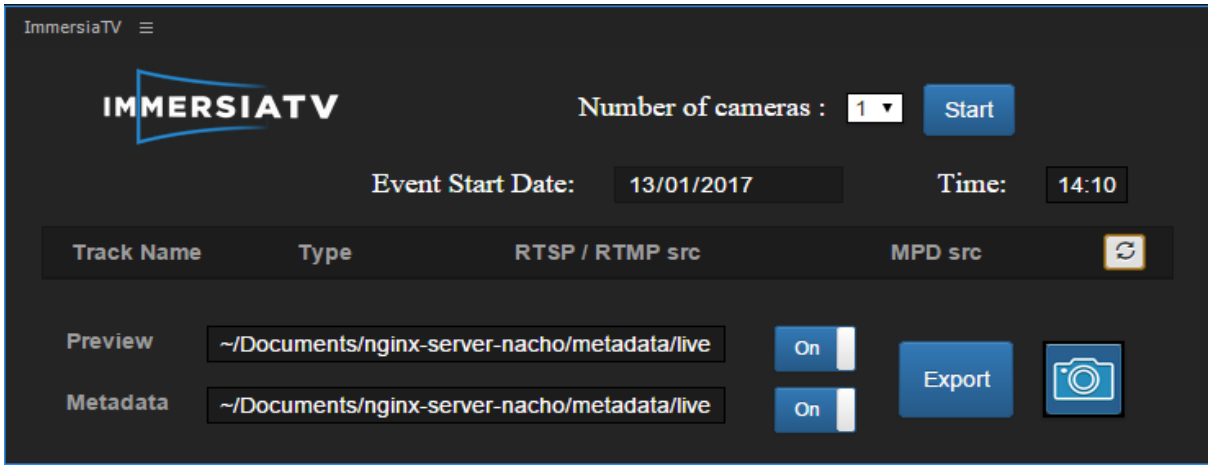


Figura. 15 *Vista inicial del panel "ImmersiaTV" en Adobe Premiere Pro*

Con el botón *Start* se genera una secuencia con tantas pistas de vídeo como cámaras seleccionadas. Estas pistas contienen una imagen de referencia a la que le podemos ajustar la duración y momento de aparición a través del panel de *Línea de tiempo* e insertarle transiciones de entrada y salida, tal como se puede ver en la **Figura. 16**.

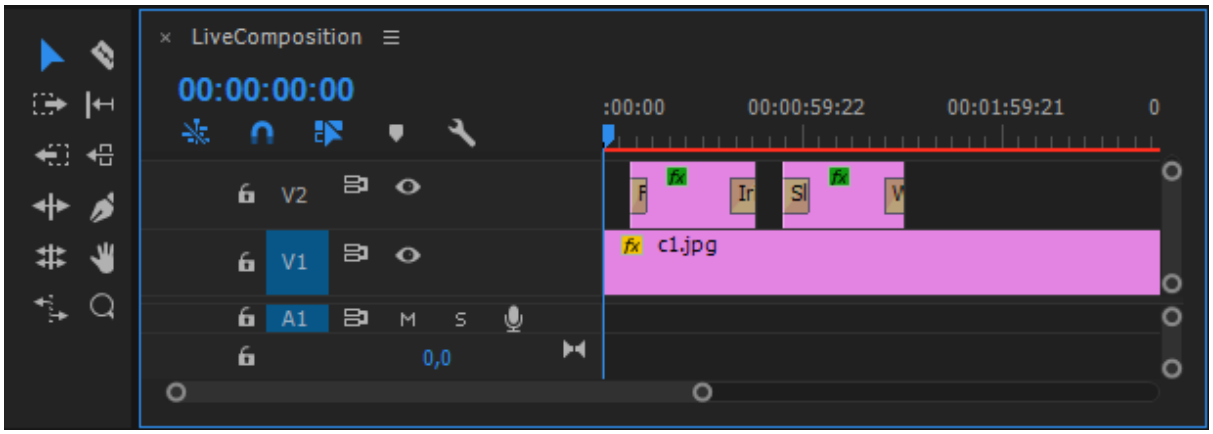


Figura. 16 *Captura del panel "Línea de tiempo" de Adobe Premiere Pro*

El plugin lee las distintas pistas del proyecto. Cada pista corresponde a una cámara y permite seleccionar el tipo de vídeo (plano / omnidireccional) y asignar a cada pista la ruta de acceso al contenido. Se fija tanto la ruta del stream de previsualización (RTSP/RTMP) como la del fichero MPD que servirá el contenido en DASH. Lo vemos en la **Figura. 17**.

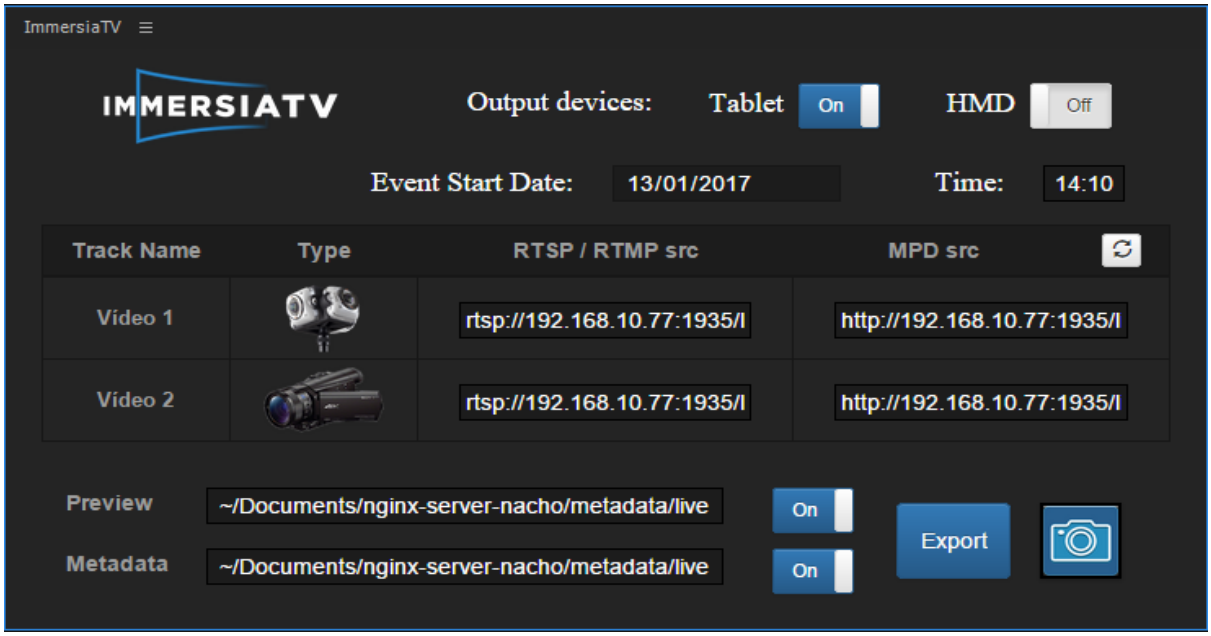


Figura. 17 *Vista del panel "ImmersiaTV" con las pistas generadas*

Aplicando a la pista secundaria el efecto portal es posible ajustarlo a nuestro gusto mediante el panel *Control de efectos* tal como se muestra en la **Figura. 18**. También es posible introducir transiciones de entrada y salida al portal mediante el plugin de transiciones perteneciente al proyecto ImmersiaTV.

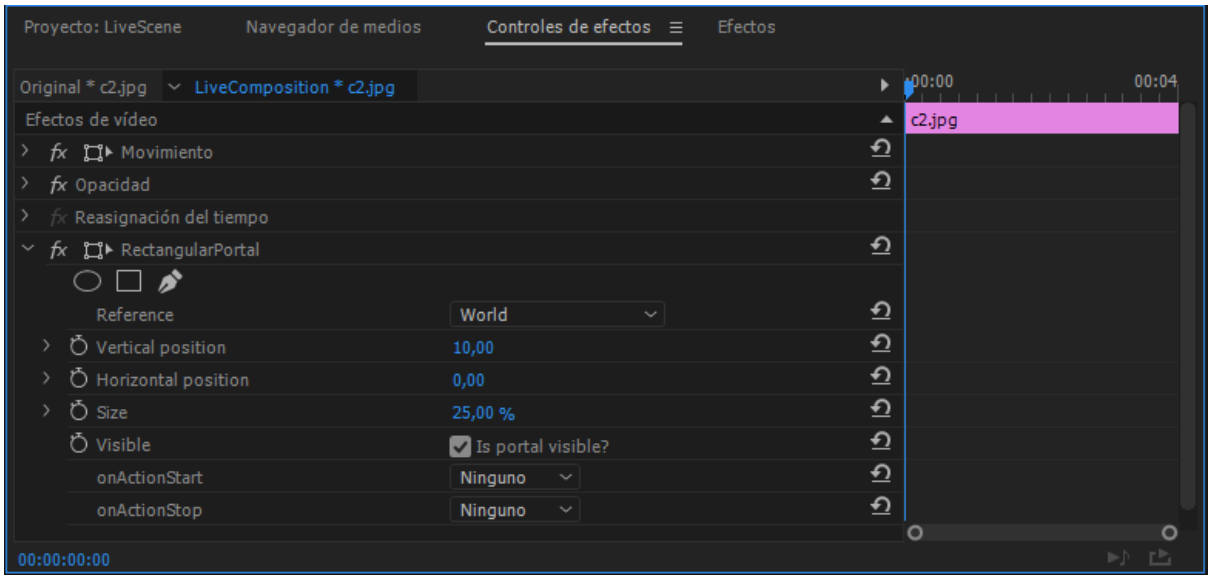


Figura. 18 *Captura del panel "Controles de efectos" de Adobe Premiere Pro*

Para facilitar la edición con imágenes de referencia, hemos añadido un botón que realiza una captura de un frame de los streams introducidos.
Al pulsar dicho botón se ejecuta un programa que hemos desarrollado en C#. Éste trabaja en segundo plano y realiza una llamada por consola a la aplicación ffmpeg [21]. A ésta se le pasa la ruta de cada stream y la ruta del fichero donde se guardará la imagen obtenida.

Una vez obtenidas las capturas, el panel *Programa* se actualiza automáticamente mostrando las nuevas imágenes. En la **Figura. 19** vemos un ejemplo donde se puede apreciar un portal insertado en el vídeo omnidireccional

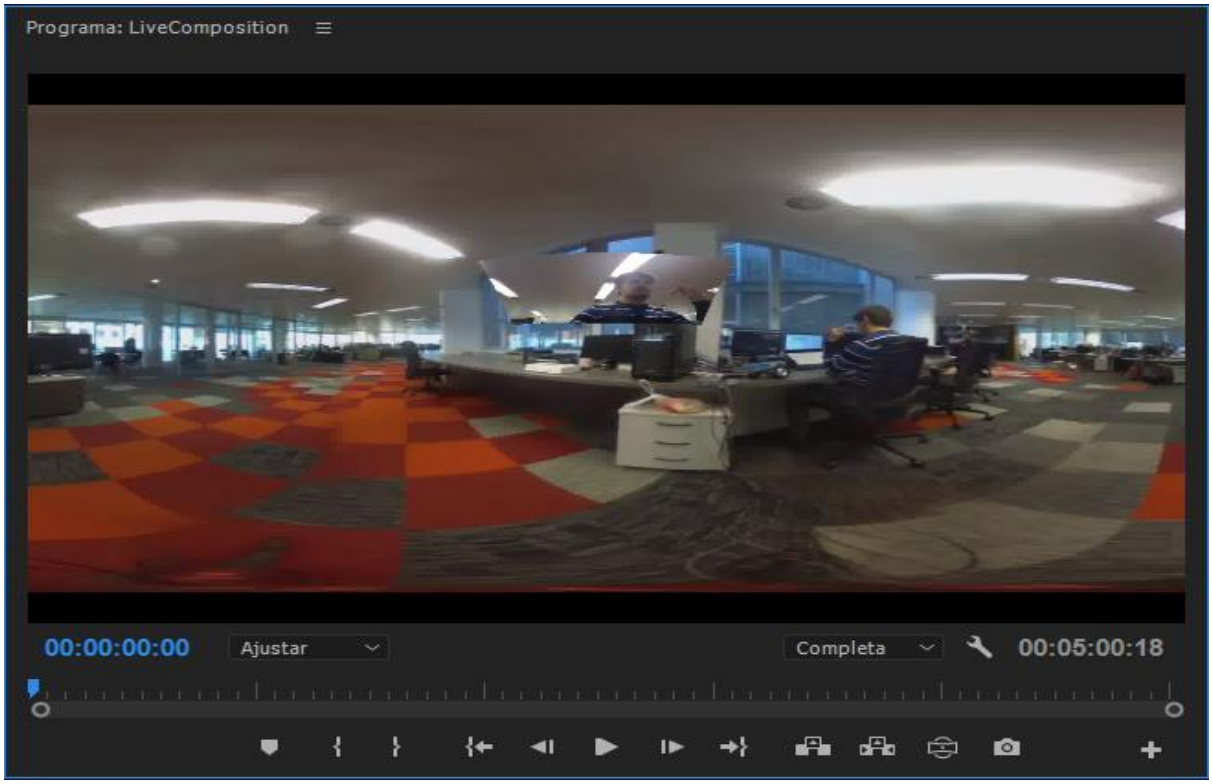


Figura. 19 *Captura del panel "Programa" en Adobe Premiere Pro*

En la imagen de la **Figura. 20** vemos como se muestra el conjunto de elementos explicado anteriormente. Arriba encontramos los paneles *Controles de efectos* y *Programa*. Abajo el plugin *ImmersiaTV* y el panel *Línea de tiempo*.

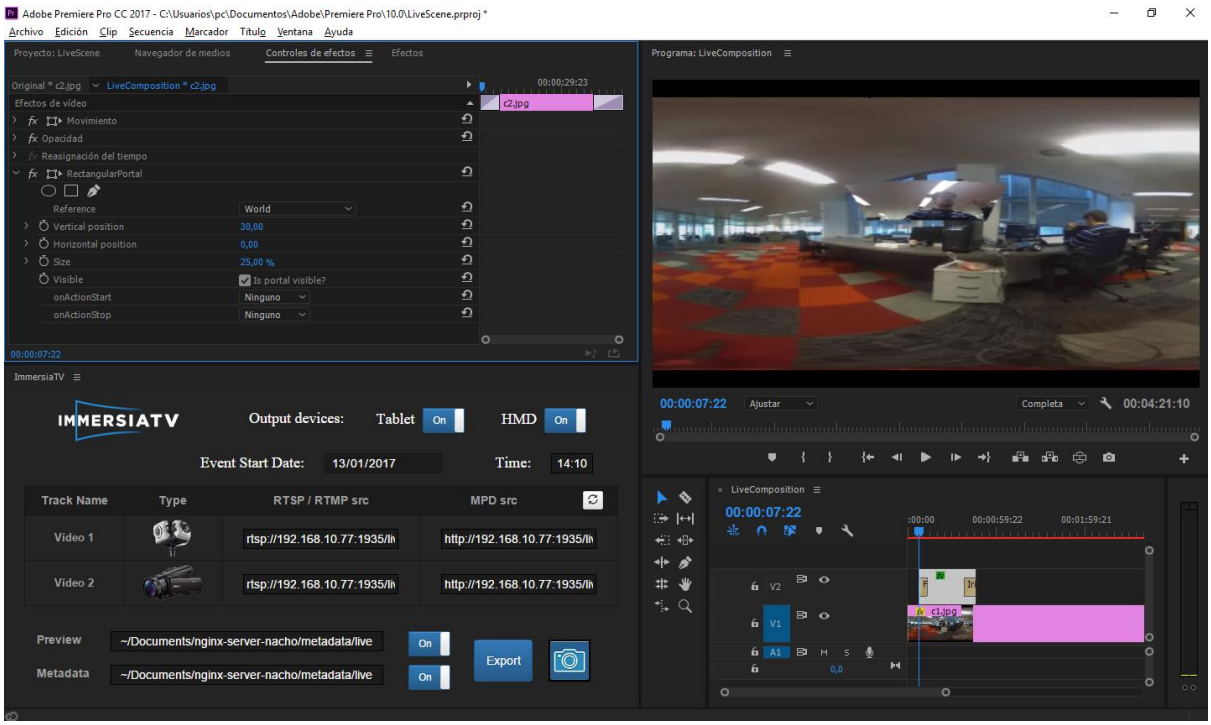


Figura. 20 *Vista general del espacio de trabajo en Adobe Premiere Pro*

Una vez finalizada la configuración de la escena y a través del panel *ImmersiaTV* podemos realizar una previsualización de la escena y exportar el fichero de metadatos que contiene toda la información necesaria para su reproducción. En nuestro caso publicamos los metadatos en el servidor Nginx que hemos configurado.

Si se tiene activada la opción Preview arranca automáticamente un reproductor de vídeo omnidireccional donde podemos ver la escena generada. Además es posible realizar tanto la previsualización como la visualización en DASH desde distintos dispositivos mediante la aplicación de ImmersiaTV.

2.5.2 Metadatos

Los ficheros de metadatos generados por la aplicación son archivos XML que siguen la misma estructura de los metadatos originales del proyecto ImmersiaTV (pueden verse en el Apéndice 1. Metadatos de ImmersiaTV. Los campos alterados para el caso de contenido en vivo son:

- el atributo *type* del elemento *ITVEvents* al que se le asigna el valor *dynamic* para indicar que es contenido en vivo. Para el caso on demand tiene el valor *static.*
- el atributo *time* del elemento DefineShape. Como trabajamos con eventos en tiempo real ahora el campo time es un tiempo UTC (del inglés *Universal Time Coordinated*).

- el atributo *mediafile* del elemento *DefineShape*. Antes se indicaba el nombre del vídeo contenido en el servidor. Ahora se indica una ruta absoluta que apunta a un flujo RTMP o RTSP en el caso *Preview* y a un fichero MPD para la reproducción DASH.

```
<?xml version="1.0" encoding="UTF-8"?>
<ITVEvents xmlns="urn:immersiatv:immersiatv01:2016:xml"
        xmlns:xsi="http://www.w3.org/2001/XMLSChema-instance"
        xsi:schemaLocation="urn:immersiatv:immersiatv01:2016:xml
http://server.immersiatv.eu/public_http/metadata/ImmersiaTV.xsd"
        type="dynamic">
        <DefineScene id="0" device="hmd" time="1484313000000">
                <DefineShape id="0" mediaFile="http://192.168.10.77:1935/live/cam1/manifest.mpd"
type="sphericalCap">
                        <Anchor id="0" distance="1.00"/>
                </DefineShape>
        </DefineScene>
        <DefineScene id="1" device="hmd" time="1484313008425.08">
                <DefineShape id="1" mediaFile="http://192.168.10.77:1935/live/cam2/manifest.mpd"
type="rectangle" size="0.2500" maskFile="fadein.shader,5.463791847229" visible="true">
                        <Anchor id="0" referenceFrame="world" longitude="0.00" latitude="30.00"
distance="0.79"/>
                </DefineShape>
        </DefineScene>
        <DefineScene id="2" device="hmd" time="1484313038413.38">
                <RemoveShape id="1" maskFile="irisinverse.shader,8.04970932006836"/>
        </DefineScene>
        <DefineScene id="3" device="hmd" time="1484313261678.1">
                <RemoveShape id="0"/>
        </DefineScene>
        <DefineScene id="4" device="tablet" time="1484313000000">
                <DefineShape id="0" mediaFile="http://192.168.10.77:1935/live/cam1/manifest.mpd"
type="sphericalCap">
                        <Anchor id="0" distance="1.00"/>
                </DefineShape>
        </DefineScene>
        <DefineScene id="5" device="tablet" time="1484313008425.08">
                <DefineShape id="1" mediaFile="http://192.168.10.77:1935/live/cam2/manifest.mpd"
type="rectangle" size="0.2500" maskFile="fadein.shader,5.463791847229" visible="true">
                        <Anchor id="0" referenceFrame="world" longitude="0.00" latitude="30.00"
distance="0.79"/>
                </DefineShape>
        </DefineScene>
        <DefineScene id="6" device="tablet" time="1484313038413.38">
                <RemoveShape id="1" maskFile="irisinverse.shader,8.04970932006836"/>
        </DefineScene>
        <DefineScene id="7" device="tablet" time="1484313261678.1">
                <RemoveShape id="0"/>
        </DefineScene>
```

Figura. 21 *Fichero de metadatos generados por el plugin ImmersiaTV*

En la **Figura. 21** vemos que se definen 2 contenidos, uno para HMD y el otro para tableta. En este caso para cada dispositivo tenemos una escena de tipo *SphericalCap* y otra de tipo *Rectangle* que se corresponden respectivamente al vídeo omnidireccional principal y al vídeo plano introducido en forma de portal. Observamos también que aparte del tipo de vídeo cada flujo tiene otros atributos que definen sus propiedades. Vemos como el tiempo introducido se encuentra en formato ntp y el atributo *mediaFile* contiene la ruta donde se encuentra el contenido, en este caso, un archivo de metadatos de MPEG-DASH.

3. Resultados

Tras analizar los requisitos iniciales del trabajo descritos en el apartado 2.2 Requisitos, veamos ahora los resultados obtenidos.

Empezemos por el bloque de captura. Por lo que se refiere al requisito de sincronización de flujos en origen y mínima latencia posible (2.2.1 Captura de vídeo), se han configurado los parámetros para reducir al mínimo la latencia, tanto en el tramo entre la fuente y el servidor como en el servidor que genera el contenido en formato MPEG-DASH. La sincronización en origen no es posible con las herramientas usadas por lo que para un correcto funcionamiento se tendrán que usar otro tipo de cámaras que incluyan esta función o realizar la sincronización en origen mediante un servidor.

Para el bloque de producción la herramienta implementada cumple todos los requisitos citados en el punto 2.2.2 Herramienta de edición. Funciona tanto Windows como en Mac OS y al ser una extensión de Adobe Premiere Pro se integraría perfectamente en sistemas de producción de video de la industria audiovisual. La herramienta permite editar la escena omnidireccional mediante la introducción manual de una serie de parámetros. Es posible seleccionar el vídeo omnidireccional “base”, e insertar otros vídeos directivos y omnidireccionales en forma de portales. A estos se les puede asignar transiciones de entrada y salida, el tamaño (ancho y alto) y la posición que tendrán sobre la escena. Además es posible seleccionar si la posición es relativa al mundo virtual o al usuario.

La salida del bloque de producción es un fichero de metadatos (requisito 2.2.3 Metadatos) que se provee por separado de los flujos de audio y vídeo tal como se pedía. A día de hoy la actualización de los metadatos se realiza mediante la exportación manual y no de forma regular, al contrario de lo que se pedía en los requerimientos. Creemos que es más conveniente hacerlo así dado que de este modo solo se introducen los cambios cuando el usuario está totalmente seguro de que está todo en su sitio. Una actualización de forma regular podría introducir un evento mientras este todavía se está editando y los resultados no serían los deseados.

Es posible realizar la previsualización en directo de la composición de la escena antes de su publicación, tanto desde el sistema de edición de contenidos como desde dispositivos móviles, tabletas y HMD (tal como pedía el requisito 2.2.4 Reproducción). La adaptación de los reproductores por parte del equipo de media de i2Cat ha necesitado de la interacción con otros miembros de dicho equipo para la integración de los nuevos metadatos generados, consiguiendo excelentes resultados.

4. Conclusiones y líneas futuras

El trabajo se ha desarrollado en el contexto del proyecto europeo ImmersiaTV y la herramienta implementada permitirá llevar a cabo el segundo piloto, cuyo objetivo principal es la producción y distribución de un evento en directo. Este ofrecerá una experiencia en vivo más atractiva, usando vídeo y sonido omnidireccional, para dar al espectador la sensación de "estar allí".

Como se ha podido observar, este es un proyecto innovador y complejo por lo que ha sido fundamental el trabajo en equipo para lograr encajar las distintas piezas que lo forman. Además ha habido un importante proceso de autoaprendizaje y uso de los recursos de información, dado que la mayoría de herramientas así como algunos de los lenguajes de programación utilizados eran desconocidos por el autor a la fecha de inicio de este trabajo.

Ha sido necesario realizar una planificación del trabajo (como queda reflejada en el Apéndice 2. Planificación del trabajo) para lograr los objetivos en el tiempo establecido. Como en todo proyecto durante el desarrollo han surgido dificultades pero estas han sido solucionadas de forma satisfactoria, pues tal como hemos visto en la sección de resultados, la solución propuesta cumple la mayoría de los requisitos que pedía el proyecto. Es posible editar la composición de una escena omnidireccional con portales, realizar una previsualización del contenido, publicarlo en un servidor y reproducirlo desde distintos dispositivos.

El hecho de que los flujos en origen no estén sincronizados es un problema grave dado que si se está capturando la misma escena desde puntos de vista distintos se pierde la coherencia. La solución vendrá con la compra de nuevos dispositivos que integren sincronización en origen o mediante el desarrollo de una solución de sincronización ad-hoc.

Y esto no termina aquí. En los meses venideros se investigará la introducción de audio 3D en las escenas, con el que se conseguirá una experiencia inmersiva todavía más real. Y está previsto que el proyecto culmine con la introducción de la solución comercial en uno o más organismos de radiodifusión de modo que todo el mundo pueda disfrutar de la experiencia ImmersiaTV desde el sofá de su casa.

5. Presupuesto

Para el cálculo del coste del proyecto primero de todo tendremos en cuenta la mano de obra, es decir, el coste del tiempo invertido en el desarrollo del trabajo.

	Cantidad	**Precio/Hora**	**Horas invertidas**	**TOTAL**
Ingeniero Junior	1	8.00€/h	496	**3968 €**
Ingeniero Senior	1	16€/h	60	**960 €**

Tabla 1. *Costes de mano de obra*

Para las pruebas se han utilizado distintos dispositivos (ordenadores, tabletas, móviles y cámaras) que pertenecen a la empresa i2Cat. Los dispositivos no se han comprado expresamente para este trabajo, por lo que el coste introducido en la Tabla 1 no es el coste total del producto sino una aproximación del coste proporcional al tiempo de uso. Este ha sido calculado teniendo en cuenta que el proyecto ImmersiaTV dura 2 años y medio (30 meses), mientras que este trabajo se ha realizado en 4 meses.

	Precio
PC de desarrollo – Sony Vaio SVE1513F4EW (4 meses)	99.86 €
Móvil - Samsung Galaxy S7 (4 meses)	95.86 €
Tablet – Nexus 7 (4 meses)	29.18 €
QBiC MS-1 XP Panorama 360 Camera (4 meses)	193.46 €
PC para realizar el Stitching (4 meses)	133.33 €
TOTAL	**551.69 €**

Tabla 2. *Costes de dispositivos*

Además se han utilizado algunos softwares que aunque cuentan con una versión de prueba disponen de una versión comercial. Por lo que incluiremos el coste proporcional de uso de las licencias. En este caso las licencias son anuales por lo que no se tiene en cuenta el tiempo total del proyecto ImmersiaTV sino los 12 meses de duración de las licencias.

	Precio
Licencia VahanaVR (4 meses)	836.07 €
Licencia Wowza (4 meses)	247.18 €
Licencia Adobe Premiere Pro (4 meses)	145.14 €
TOTAL	**1228.39 €**

Tabla 3*. Costes de licencias*

Sumando los costes de cada una de las tablas anteriores obtenemos que el coste total del proyecto asciende a los **6708.08 €.**

Bibliografía

[1] «ImmersiaTV project,» [En línea]. Available: http://www.immersiatv.eu/the-project-2/. [Último acceso: 30 Diciembre 2016].

[2] J. Llobera, «A new content format for immersive experiences,» 20 Octubre 2016. [En línea]. Available: https://arxiv.org/ftp/arxiv/papers/1610/1610.07577.pdf.

[3] T. I. Society, «RTP: A transport protocol for Real-Time Applications,» Julio 2003. [En línea]. Available: https://www.ietf.org/rfc/rfc3550.txt.

[4] T. I. Society, «Real Time Streaming Protocol (RTSP),» Abril 1998. [En línea]. Available: https://www.ietf.org/rfc/rfc2326.txt.

[5] A. Systems, «Adobe's Real TIme Messaging Protocol,» 21 Diciembre 2012. [En línea]. Available: http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/rtmp/pdf/rtmp_specification_1.0.pdf.

[6] «MPEG-DASH ISO/IEC 23009-1:2014,» 15 Mayo 2014. [En línea]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=65274.

[7] «Scope of the MPEG-DASH standard,» [En línea]. Available: https://www.computer.org/csdl/mags/mu/2011/04/mmu2011040062-abs.html. [Último acceso: 30 Diciembre 2016].

[8] «DASH Data Model,» [En línea]. Available: https://bitmovin.com/what-is-cmaf-threat-opportunity/. [Último acceso: 30 DIciembre 2016].

[9] «MPEG-DASH SRD ISO/IEC 23009-1:2014/Amd 2:2015,» 1 Julio 2015. [En línea]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=66486.

[10] «GStreamer - Open source multimedia Framework,» [En línea]. Available: https://gstreamer.freedesktop.org/. [Último acceso: 30 Diciembre 2016].

[11] «GStreamer pipeline for a basic ogg player,» [En línea]. Available: https://gstreamer.freedesktop.org/documentation/application-development/introduction/basics.html.

[12] «LiveMediaStreamer,» i2Cat Foundation's Media Internet Area, [En línea]. Available: http://livemediastreamer.i2cat.net/. [Último acceso: 30 Diciembre 2016].

[13] «Docker containers,» [En línea]. Available: https://www.docker.com/what-docker. [Último acceso: 30 DIciembre 2016].

[14] «LiveMediaStreamer - Conceptual diagram of the logical structure of the control and dataflow layers,» [En línea]. Available: http://livemediastreamer.i2cat.net/architecture/. [Último acceso: 30 Diciembre 2016].

[15] «Unity 3D,» [En línea]. Available: https://unity3d.com/es/unity. [Último acceso: 30 Diciembre 2016].

[16] W. Reese, «Nginx: the High-Performance Web Server and Reverse Proxy,» 1 Septiembre 2008. [En línea]. Available: http://www.linuxjournal.com/article/10108.

[17] «Wowza Streaming Engine,» Wowza Media Systems, [En línea]. Available: https://www.wowza.com/products/streaming-engine. [Último acceso: 30 Diciembre 2016].

[18] «Video-Stitch VahanaVR,» [En línea]. Available: http://www.video-stitch.com/live-vr/. [Último acceso: 30 Diciembre 2016].

[19] «Cámara QBIC MS-1 XP Panorama,» [En línea]. Available: https://filmora.wondershare.com/virtual-reality/360-camera-rigs-for-vr-video.html. [Último acceso: 30 Diciembre 2016].

[20] «Adobe Premiere Pro CC,» Adobe Systems, [En línea]. Available: http://www.adobe.com/es/products/premiere.html. [Último acceso: 30 Diciembre 2016].

[21] «About FFmpeg,» [En línea]. Available: https://ffmpeg.org/about.html. [Último acceso: 30 Diciembre 2016].

Apéndices

Apéndice 1. Metadatos de ImmersiaTV

Table of Contents

Group by: Namespace

urn:immersiatv:immersiatv01:2016:xml

Elements
- ITVEvents
- ITVEvents/DefineScene
- ITVEvents/DefineScene/DefineShape
- ITVEvents/DefineScene/DefineShape/Anchor
- ITVEvents/DefineScene/RemoveShape

Simple Types
- AnchorMethodType
- DeviceType
- ITVEventsType
- ProjectionType
- ReferenceFrameType
- RelativeValueType
- ShapeType
- mergeModeType
- playbackState

Main schema ImmersiaTV.xsd

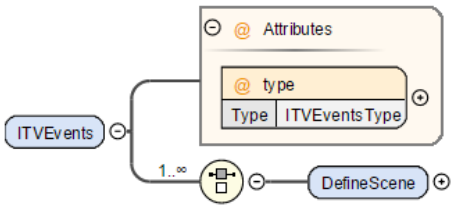
Properties		
	Attribute Form Default	**unqualified**
	Element Form Default	**qualified**

[top]

Element ITVEvents

Annotations: Root node containing a list of timestamped events.

Diagram:



Properties		
	Content	**complex**
Children	DefineScene	

Attributes:

QName	Type	Use	Annotation
type	ITVEventsType	required	Type of event stream.

[top]

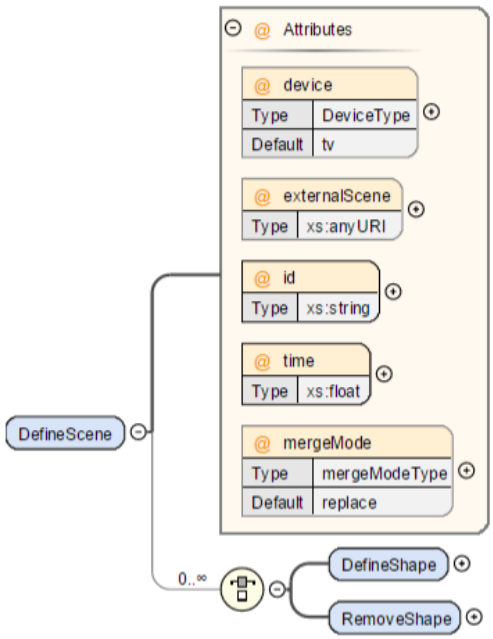
Element ITVEvents / DefineScene

Annotations: This event describes a scene. A scene is a collection of shapes with media projected onto them, along with an optional Unity scene. Media is identified with a file name, and is composed of texture media, an optional mask and an optional transition (a second mask, controlled by the program).

Scenes are considered to be inside a "Background Sphere" of radius 1000 Unity units, although this size is relevant only when using external scenes (Unity files). All measures defined in this specification are relative to the reference Background Sphere.

A scene can be associated with a specific kind of device (like a TV or a tablet), and multiple scenes can be defined simultaneously. Each device should show the scene intended for its kind and disregard the others. A typical scene for a TV would include a rectangle shape facing the camera and displaying a flat (conventional) media, whereas a typical scene for a Head-Mounted Display would include the largest possible spherical shape (size=1, the size of the Background Sphere) displaying an omnidirectional media, plus possibly some portals. Portals are shapes (with any geometry) placed in the scene, showing some media, and possibly capable or triggering actions.

Diagram



Properties — Content: **complex**

Children: **DefineShape**, **RemoveShape**

Attributes

QName	Type	Default	Use	Annotation
device	**DeviceType**	tv	optional	Device intended for this Scene.
externalScene	**xs:anyURI**		optional	External Unity3D scene file to load. Its contents will be merged with the Shapes defined in the MPD and some of its objects can have media rendered over them using the Mesh Shape.
id	**xs:string**		required	Id for the Scene.
mergeMode	**mergeModeType**	replace	optional	Whether this scene should replace the previous one with matching ID, or merge with it.
time	**xs:float**		required	Media time at which this event should be executed, in seconds.

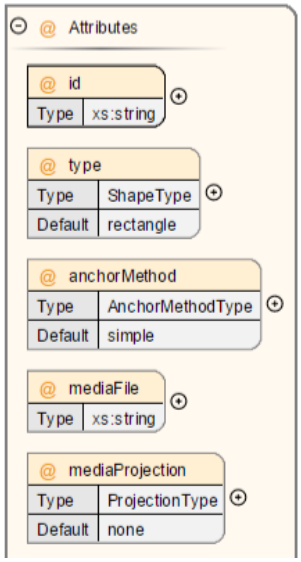
[top]

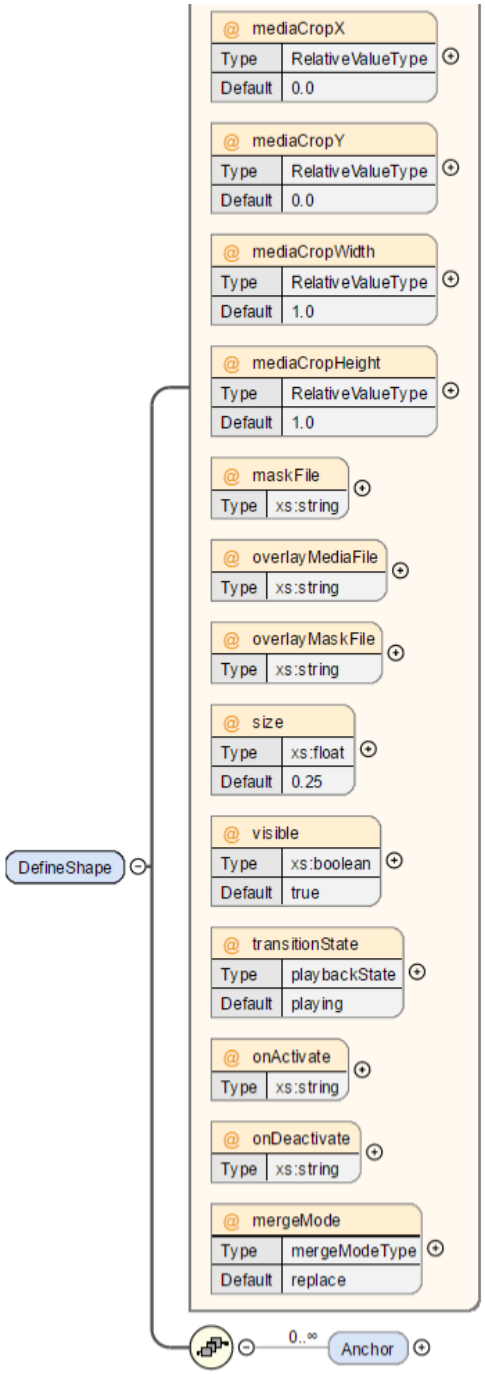
Element ITVEvents / DefineScene / DefineShape

Annotations

This event defines a new Shape, or updates an existing one. Shapes are the geometrical figures used to display media. They are positioned in the scene through Anchors. They can have multiple anchors, which are then combined using the anchorMethod attribute.

Diagram





Properties — Content: **complex**

Children: **Anchor**

Attributes

QName	Type	Default	Use	Annotation
anchorMethod	**AnchorMethodType**	simple	optional	How multiple anchors are combined to generate the final Shape position in the scene.
id	**xs:string**		required	Id for this Shape, so it can be updated or removed in future events.
maskFile	**xs:string**		optional	Optional mask to use on the texture media. This mask is also a file name (see mediaFile), so it can either be a video or a still image. It uses a "secondary timeline", independent from the mediaFile, since it can be controlled by the application (it can be shorter and loop, or triggered by user actions). It must be a gray image or video. This allows, for example, to have circles with blurry edges shown in the scene, using only rectangular shapes, or transitions.

	mediaCropHeight	RelativeValueType	1.0	optional	Optional cropping height for the texture media. 0 means no height, 1 means original media height.
	mediaCropWidth	RelativeValueType	1.0	optional	Optional cropping width for the texture media. 0 means no width, 1 means original media width.
	mediaCropX	RelativeValueType	0.0	optional	Optional left crop for the texture media. 0 means left border, 1 means right border.
	mediaCropY	RelativeValueType	0.0	optional	Optional top crop for the texture media. 0 means top border, 1 means bottom border.
	mediaFile	xs:string		optional	Texture media (color) to display on this Shape. This is the base name (without path or extension) of a file which must be available from the same location where the metadata XML was recovered. Its playback is synchronized across all devices (they all show the same frame at the same time) and cannot be stopped. This is known as the "main timeline".
	mediaProjection	ProjectionType	none	optional	Projection used for all media used on this shape (texture media, mask and transition).
	mergeMode	mergeModeType	replace	optional	Whether this shape should replace the previous one with matching ID, or merge with it.
	onActivate	xs:string		optional	Player method to call when the shape has been "activated". The exact action that triggers the activation depends on the device and the player (could be a "touch" action on a tablet, and "look for more than 3 seconds" on an HMD). The list of available methods is outside the scope of this metadata description.
	onDeactivate	xs:string		optional	Player method to call when the shape has been "deactivated". The exact action that triggers the deactivation depends on the device and the player (could be "stop looking" on an HMD). The list of available methods is outside the scope of this metadata description.
	overlayMaskFile	xs:string		optional	Alpha mask for the overlayMediaFile.
	overlayMediaFile	xs:string		optional	A second texture media (with companion alpha mask in overlayMaskFile) which is rendered on top of the mediaFile. It uses the "secondary timeline" (see maskFile). It can be used to make a "beauty mask", embellishing transitions.
	size	xs:float	0.25	optional	Relative size of the shape. Its meaning depends on the type of shape. For rectangles, 1 means the shape is as wide as the Background Sphere. For sphericalCaps, 1 means the cap covers the whole 360 degrees (and therefore it is a sphere). In both cases, the height of the shape is derived from the media's aspect ratio.
	transitionState	playbackState	playing	optional	Playback state of the transition. Interaction can change it.
	type	ShapeType	rectangle	optional	Particular geometry that this Shape should have.
	visible	xs:boolean	true	optional	Whether this shape is visible in the scene or not. Hidden shapes can be activate through interaction.

[top]

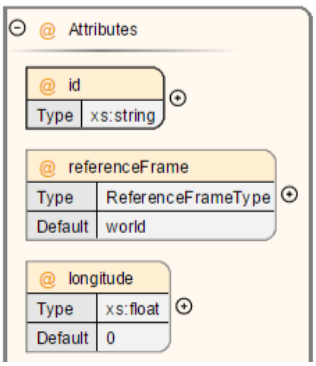
Element ITVEvents / DefineScene / DefineShape / Anchor

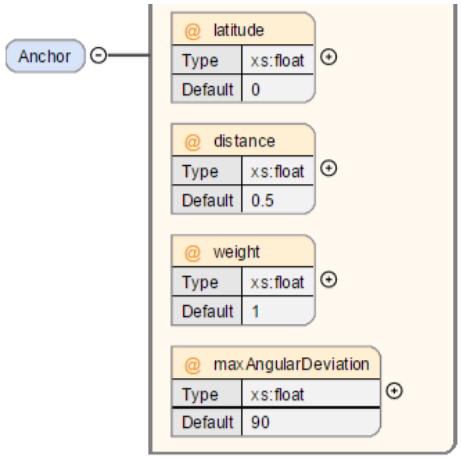
Annotations

Anchors specify a point in the scene, tied to a fixed polar coordinate, or relative to the camera. The exact position of the shape depends on its type:

- point: The Shape is located exactly at the anchor point
- rectangle: The Anchor is located at the lower left corner of the rectangle.
- sphericalCap: The Anchor is located at the lower left corner of the spherical cap.
- mesh: Its position cannot be changed by an Anchor. Mesh shapes should not have Anchors.

Diagram





Properties

Content	**complex**
MinOccurs	**0**
MaxOccurs	**unbounded**

Attributes

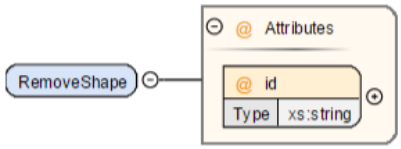
QName	Type	Default	Use	Annotation
distance	**xs:float**	0.5	optional	Distance from the camera. 0 means the Anchor is touching the camera, 1 means it is touching the Background Sphere.
id	**xs:string**		required	Id for this Anchor, so it can be updated or removed in future events.
latitude	**xs:float**	0	optional	Degrees above or below the horizontal plane.
longitude	**xs:float**	0	optional	Degrees on the horizontal plane.
maxAngularDeviation	**xs:float**	90	optional	Maximum distance (in degrees) that the Shape can move from this Anchor, if multiple Anchors are mixed. Beware that multiple Anchors with multiple maxAngularDeviation constrains might not be satisfiable at the same time.
referenceFrame	ReferenceFrameType	world	optional	Reference frame for the Anchor.
weight	**xs:float**	1	optional	When multiple anchors are defined and the "simple" anchorMethod is used, all anchors are averaged using their respective weights. Longitude, Latitude and Distance are each one averaged independently.

[top]

Element ITVEvents / DefineScene / RemoveShape

Annotations: This event removes an existing shape from the scene.

Diagram



Properties

Content	**complex**

Attributes

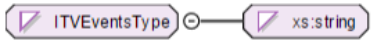
QName	Type	Use	Annotation
id	**xs:string**	required	Id of the Shape to remove.

[top]

Simple Type ITVEventsType

Annotations: What kind of event stream is this.

Diagram



Facets

Enumeration	**static**	The content of this event stream file will not change, the player does not need to periodically re-download it.
Enumeration	**dynamic**	The content of this event stream might change in time, the player should re-download it periodically to check if new content has been added.

Used by

Attribute	**ITVEvents/@type**

[top]

Simple Type DeviceType

Annotations	Categorization for all kinds of devices supported by ImmersiaTV.		
Diagram	DeviceType — xs:string		
Facets	Enumeration	**tv**	This scene is intended for TV devices: A flat screen, viewed from a distance, with little or no interactivity or immersive capacity.
	Enumeration	**tablet**	This scene is intended for Tablet devices: An interactive flat screen, viewed from close on with an orientation sensor.
	Enumeration	**hmd**	This scene is intended for Head-Mounted Displays: A screen worn close to the eyes (providing immersive illusion), with orientation tracking and little or no interactivity.
Used by	Attribute	**ITVEvents/DefineScene/@device**	

[top]

Simple Type mergeModeType

Annotations	How this information is merged with the previous one.		
Diagram	mergeModeType — xs:string		
Facets	Enumeration	**update**	This node updates the previous one, this is, attributes and child nodes with matching IDs are replaced, and the rest are unaffected.
	Enumeration	**replace**	This node completely replaces any existing node with matching ID.
Used by	Attributes	**ITVEvents/DefineScene/@mergeMode**, **ITVEvents/DefineScene/DefineShape/@mergeMode**	

[top]

Simple Type ShapeType

Annotations	Geometry of a shape.		
Diagram	ShapeType — xs:string		
Facets	Enumeration	**point**	A point, with no orientation. Only meaningful for Audio sources. Requires an Anchor.
	Enumeration	**rectangle**	A rectangle, always facing the camera. Requires Anchor and size. Aspect ratio is fixed by the media shown on it.
	Enumeration	**sphericalCap**	A section of a sphere centered on the camera. The Anchor marks its lower-left corner, which is also the origin for the texture coordinates. It requires size, and the aspect ratio is fixed by the media shown on it.
	Enumeration	**mesh**	A mesh loaded from an external file, specified at the DefineScene level with the externalScene attribute. The "id" of the Shape element must match the name of an object in the scene. Anchors are ignored, objects are already positioned in the world in the external file.
Used by	Attribute	**ITVEvents/DefineScene/DefineShape/@type**	

[top]

Simple Type AnchorMethodType

Annotations	Mechanism to combine multiple anchors into a final location in the scene.		
Diagram	AnchorMethodType — xs:string		
Facets	Enumeration	**simple**	The final anchor position is the independently weighted average of the Longitude, Latitude and Distance of all child anchor nodes.
	Enumeration	**alwaysOnScreen**	If the anchor is in the field of view, it is used. Otherwise, the player is free to choose any location which appears on screen.
Used by	Attribute	**ITVEvents/DefineScene/DefineShape/@anchorMethod**	

[top]

Simple Type RelativeValueType

Annotations	A value between 0 and 1.	
Diagram	RelativeValueType — xs:float	
Facets	MaxInclusive	1.0
	MinInclusive	0.0
Used by	Attributes	ITVEvents/DefineScene/DefineShape/@mediaCropHeight, ITVEvents/DefineScene/DefineShape/@mediaCropWidth, ITVEvents/DefineScene/DefineShape/@mediaCropX, ITVEvents/DefineScene/DefineShape/@mediaCropY

[top]

Simple Type ReferenceFrameType

Annotations	Reference frame for an Anchor.		
Diagram	ReferenceFrameType — xs:string		
Facets	Enumeration	user	The coordinates given for this anchor are relative to the camera (the user).
	Enumeration	world	The coordinates given for this anchor are relative to the world.
Used by	Attribute	ITVEvents/DefineScene/DefineShape/Anchor/@referenceFrame	

[top]

Simple Type ProjectionType

Annotations	Type of projection used on a given media file.		
Diagram	ProjectionType — xs:string		
Facets	Enumeration	none	This media has not been projected, this is, it is a conventional flat image.
	Enumeration	equirectangular	The original image was omnidirectional, and was flattened by mapping longitude and latitude directly to the x and y texture coordinates.
Used by	Attribute	ITVEvents/DefineScene/DefineShape/@mediaProjection	

[top]

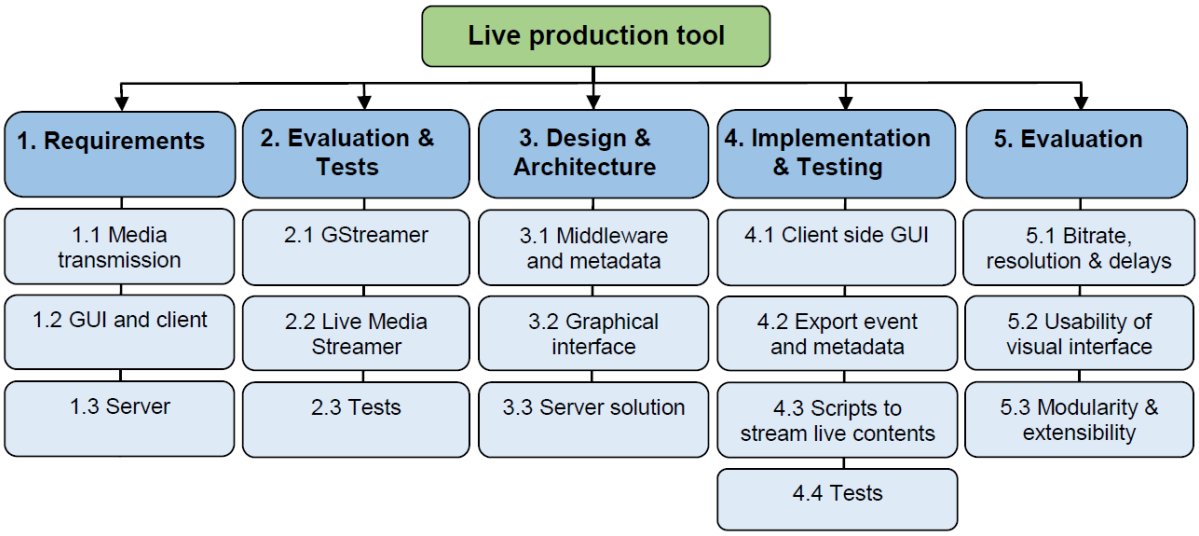
Simple Type playbackState

Annotations	Whether a given media is playing or is paused.		
Diagram	playbackState — xs:string		
Facets	Enumeration	playing	Media is playing (time is advancing, frames are changing).
	Enumeration	paused	Media is paused (time is not advancing, the same frame is shown all the time).
Used by	Attribute	ITVEvents/DefineScene/DefineShape/@transitionState	

[top]

Apéndice 2. Planificación del trabajo

Work Breakdown Structure



Work Packages

Project: Requirements	WP ref: 1	
Major constituent: software	Sheet 1 of 5	
Short description: Describe the technical requirements for the whole project.	Planned start date: 19/09/2016 Planned end date: 30/09/2016	
	Start event: End event:	
Internal task T1: Requirements for media transmission software. Internal task T2: Define requirements for GUI and client side. Internal task T3: Define requirements for server.	Deliverables:	Dates:

Project: Evaluation of streaming alternatives and tests	WP ref: 2	
Major constituent: software	Sheet 2 of 5	
Short description: Evaluate LMS, GStreamer and other media frameworks for live environments.	Planned start date: 03/10/2016 Planned end date: 21/10/2016	
	Start event: End event:	
Internal task T1: GStreamer software evaluation. Internal task T2: Live Media Streamer software evaluation. Internal task T3: Tests with the chosen solution.	Deliverables:	Dates:

Project: Design and architecture	WP ref: 3	
Major constituent: software	Sheet 3 of 5	
Short description: Design and architecture for the modular software solution, client and server side.	Planned start date: 10/10/2016 Planned end date: 28/10/2016	
	Start event: End event:	
Internal task T1: Design of the middleware architecture and metadata. Internal task T2: Design graphical interface. Internal task T3: Design server solution.	Deliverables:	Dates:

Project: Implementation and Testing	WP ref: 4	
Major constituent: software and hardware	Sheet 4 of 5	
Short description: Develop and test client and server part of the live streaming solution	Planned start date: 31/10/2016 Planned end date: 30/12/2016	
	Start event: End event:	
Internal task T1: Develop client side GUI Internal task T2: Export events and metadata to be used in player. Internal task T3: Implement scripts to stream live contents with the chosen media framework solution. Internal task T4: Functional testing.	Deliverables:	Dates:

Project: Technical and usability evaluation	WP ref: 5	
Major constituent: software	Sheet 5 of 5	
Short description: Objective and subjective tests with the software developed in order to evaluate the quality of the project.	Planned start date: 02/01/2017 Planned end date: 10/01/2017	
	Start event: End event:	
Internal task T1: Evaluate bitrate, resolution and delays obtained with the media framework used. Internal task T2: Evaluate usability of the visual interface Internal task T3: Evaluate modularity and extensibility of the chosen proposal.	Deliverables:	Dates:

Gant Diagram

Task	Start	End	Dur. (days)
A tool for Live multi-platform content production	19/9/16	10/1/17	82
1. Requirements	19/9/16	30/9/16	10
1.1 Media transmission	19/9/16	21/9/16	3
1.2 GUI and Client	22/9/16	26/9/16	3
1.3 Server	27/9/16	30/9/16	4
2. Evaluation & Tests	3/10/16	21/10/16	15
2.1 GStreamer	3/10/16	7/10/16	5
2.2 Live Media Streamer	10/10/16	14/10/16	5
2.3 Tests	17/10/16	21/10/16	5
3. Design & Architecture	10/10/16	28/10/16	15
3.1 Middleware and metadata	10/10/16	14/10/16	5
3.2 Graphical interface	17/10/16	21/10/16	5
3.3 Server solution	24/10/16	28/10/16	5
4. Implementation & testing	31/10/16	30/12/16	45
4.1 Client side GUI	31/10/16	18/11/16	15
4.2 Export events metadata	21/11/16	2/12/16	10
4.3 Scripts to stream live contents	5/12/16	23/12/16	15
4.4 Tests	26/12/16	30/12/16	5
5. Evaluation	2/1/17	10/1/17	7
5.1 Bit rate, resolution & delays	2/1/17	4/1/17	3
5.2 Usability of visual interface	5/1/17	6/1/17	2
5.3 Modularity and extensibility	9/1/17	10/1/17	2