

CREACIÓ D'UNA APLICACIÓ
DE LIVE VIDEO MIXING AL NÚVOL,
UTILITZANT TECNOLOGIA DE VIDEOJOCOS

laSalle

UNIVERSITAT RAMON LLULL

Treball Final de Grau Universitat la Salle URL Barcelona
Enginyeria Multimèdia - Menció en Videojocs

realitzat per

Àlex Vicente Carpio
Departament d'Enginyeria Multimèdia

Tutor

Gabriel Fernández Ubiergo

Contents

1	Introducció	3
2	Part Teòrica	5
2.1	Explicació del projecte	5
2.1.1	Projecte Actual	5
2.1.2	Requeriments del nou Mixer	6
2.2	Eines Gràfiques	7
2.2.1	Headless Chrome	7
2.2.2	WebGL Headless	11
2.2.3	Unreal Engine 4	11
2.2.4	Unity	14
2.3	Tractament dels fluxes de video	16
2.3.1	Formats Compatibles	16
2.3.2	Protocols de Streaming	19
2.3.3	Llibreries de processament	23
3	Proposta tècnica	28
3.1	Elecció de l'eina i justificació	28
3.1.1	Chrome Headless	28
3.1.2	Unreal Engine 4	36
3.1.3	Unity	38
3.2	Taula Comparativa Eines Gràfiques	40
3.3	Elecció de les llibreries a utilitzar	41
3.3.1	Entrada de fluxos	41
3.3.2	Sortida del frame	43
3.4	Elecció del hardware i sistema	43
3.4.1	Plataforma Virtual AWS	43
3.4.2	Sistema operatiu	46
4	Part Pràctica	47
4.1	Arquitectura del sistema	47
4.2	Possibilitats	48
4.2.1	Resolució	48
4.2.2	Compatibilitat fluxes d'entrada	50
4.2.3	Control de l'app	55
4.2.4	Compatibilitat audio	55

4.2.5	Transicions	56
4.2.6	Animacions	57
4.2.7	Transformacions	58
4.2.8	Layer Sorting	62
4.2.9	Cropping	65
4.2.10	Outline and Shadows	69
4.2.11	Chroma Key	71
4.2.12	Textos	73
4.2.13	Imatges	76
4.2.14	Scenes Management	78
4.2.15	Debugging	81
4.3	Sortida Aplicació	83
4.3.1	Video	83
4.4	Rendiments Màxims i Tests	83
5	Conclusions	87
6	Línies de futur	89
7	Referències Imatges	90

1 Introducció

L’evolució de les tecnologies en el sector multimèdia és imparable, cada cop tenim a la nostra disposició més serveis de streaming, «video on demand» i eines de producció molt professionals a l’abast de qualsevol tipus d’usuari. I a la vegada, la demanda per aquest tipus d’eines també creix, ja que la gent s’interessa en donar-se a coneixer, retransmetre conferències, entretenir, i ho volen fer amb la major qualitat possible. El gran problema és que no tothom disposa d’una màquina suficientment potent per suportar eines de producció de vídeo tant potents, o dels coneixements tècnics necessaris per utilitzar-los.

Partint d’aquesta base, s’ha marcat com a objectiu principal d’aquest projecte crear una aplicació de live video mixing usant tecnologies alternatives, amb els punts clau: remot, escalable, senzill i potent. Es vol posar a disposició de qualsevol usuari, una eina al núvol capaç d’aconseguir uns resultats equivalents o superiors a les eines de vídeo més professionals. Al estar executada al núvol, vol dir que no s’utilitzaran recursos de la màquina local, per tant qualsevol tipus d’ordinador, per molt simple o antic que sigui, podrà treballar amb vídeos d’alta qualitat i realitzar composicions extraordinàries. En resum, es vol crear una aplicació multimèdia amb la que poder gestionar fluxos de vídeo d’entrada i de sortida, combinant-los i gestionant-los d’una manera senzilla, eficaç i ràpida. Durant el transcurs d’aquest treball, s’anirà veient que hi ha moltes possibilitats per dur a terme l’objectiu, es plantejaran totes elles, es compararan, i s’escollirà la que més s’adeqüi a les necessitats.

Aquest projecte ha estat realitzat a l’empresa Watchity S.L., on he estat treballant com a becari, i a la vegada ho he combinat amb el meu treball final de grau. Ja que es tracta d’una idea molt interessant i innovadora, que em va cridar molt l’atenció des de el primer moment. En aquesta empresa ja disposen d’una eina al núvol per tractar vídeo, porten anys treballant amb ella i han comprovat que la demanda per aquest tipus d’eines és real, pel que volen evolucionar el sistema i donar més possibilitats d’edició. Actualment tenen una aplicació basada en una eina anomenada Snowmix que té les seves limitacions. La quantitat de vídeos amb la que es pot treballar és limitat, estàs lligat a un programari que no controles i els efectes que es poden aplicar sobre els vídeos són massa senzills. S’analitzarà l’aplicació actual, es compararà i es decidiran uns requisits indispensables pel seu

successor. Aquesta és una de les raons per la qual es van plantejar contractar un becari que refés el sistema des de un altre punt de vista totalment diferent.

En el mercat existeixen algunes altres propostes. Com a exemples de competència tenim vMix o Wirecast, com a software instal·lable, que fan vídeo mixing professional, pel que es requereixen uns coneixements tècnics sobre el món audiovisual. Per altre banda tindríem StreamYard o Restream que fan vídeo mixing al núvol, però llavors es queden bastant limitats, no es permet gaire control sobre l'escena, no tenen gaires efectes que donin un toc elegant i es nota a simple vista que s'està usant una eina que no és professional. I és aquí on Watchity vol incidir com a una proposta diferencial, creant una eina potent al núvol totalment remota i a la vegada, potent.

Alguns exemples de les empreses/entitats que ja han treballat anteriorment amb Watchity són: *el Correo Gallego, l'Oréal, Generalitat de Catalunya, Honda...*

2 Part Teòrica

2.1 Explicació del projecte

2.1.1 Projecte Actual

Arquitectura Global

El primer que s'ha d'explicar, és la arquitectura del sistema actual de l'empresa Watchity, abans de començar amb el nou. Tenen a disposició dels clients, diferents eines de vídeo, les més importants són:

- **Cut & Share**, serveix per crear petits vídeos fàcilment, a partir d'altres més llargs, i poder-los compartir d'una manera senzilla i còmode. [1]
- Per últim tenim el **Mixer** que és en el que ens centrarem al transcurs d'aquest projecte. Aquest disposa d'un sistema de *Live Video Mixing* que es compon del frontend, i el backend. [3]
- **Control Room**, és una eina que serveix per controlar fluxos de vídeo i distribuir-los a diferents xarxes socials. [1]

En el Mixer trobem una aplicació controlada per navegador que té totes les opcions bàsiques per fer una edició en viu dels vídeos. Però, té algunes limitacions, com poden ser: un màxim de 15 capes de vídeos o imatges, no té cap tipus de compatibilitat amb elements 3D, una resolució de 720p al output, no disposa de control per guardar/carregar, limitació de potència per fluxos de streaming simultanis, poques possibilitats de transicions, no disposa de cap eina per afegir efectes ni de vídeo, ni de so (per exemple, chroma key o reverb, respectivament), impossibilitat per treballar a resolucions de 4K o més.



Figure 1: Esquema de la producció Watchity

A partir d'això, es genera un streaming de sortida en WebRTC que es pot visualitzar al navegador de l'usuari. A la vegada, es genera un streaming RTMP, que s'emet automàticament a totes les xarxes socials que l'usuari desitgi (Facebook, Youtube, Instagram, etc.)

En quant al backend, s'ocupa de realitzar les tasques demanades des de el frontend explicat anteriorment. A més de controlar les diferents configuracions com poden ser la transcodificació dels vídeos, configuració dels vídeos...

2.1.2 Requeriments del nou Mixer

Degut a les limitacions del sistema actual, es proposa una sèrie de requisits que el nou Mixer haurà de tenir, assegurant-nos que compleixi amb les expectatives i sigui útil per dur a terme aquesta “evolució”. És possible que el producte final acabi tenint més funcionalitats de les que s'esmenten a continuació, però sí que reflexa les més bàsiques.

- Poder injectar un **flux de streaming**
- Poder injectar un vídeo **preenregistrat (local)**
- Tot tipus d'elements s'han de poder inserir amb **posicions i paràmetres inicials**
- Injecció d'**imatges** amb transparència, a més de permetre el canal **alpha**
- Serà necessari que es pugui definir **escenes**, les quals també pugui emmagatzemar les posicions dels elements dins de cada una i totes les seves propietats.
- **Mute/un-mute** de cada font d'entrada
- Control de volum de cada font d'entrada
- Mixing de les fonts no silenciades
- Control de volum de la sortida mesclada
- Mute/un-mute de la sortida mesclada
- Possibilitat de tenir fins a **6 fonts** de streaming **simultàniament**

- **Chroma-keying** a les fonts de vídeo (streams o local)
- Els flux d'entrada poden tenir **diferents resolucions**
- Extreure el render final via streaming (**rtp** o similar)
- Poder definir la resolució i framerate del render final (720p/25fps, 720p/30fps, 1080p/30fps, etc.)
- Poder canviar d'un element a un altre tant per **tall** com per **dissolve**
- Possibilitat de treballar fins a **4K**

Encara que hi han alguns elements que primera vista poden semblar que es poden reutilitzar, no és així, ja que la idea es canviar totalment el sistema des de la base, utilitzant un altre tipus de software totalment diferent, veient que l'actual està molt limitat. Per tant, s'ha decidit no utilitzar res de l'anterior projecte i crear-ho tot des de zero.

2.2 Eines Gràfiques

A continuació es presentaran les eines gràfiques que podrien servir per aconseguir l'objectiu. Totes elles són ben coneudes, pel que ens assegurem una estabilitat i suport del programari. A més de tenir en totes una constant evolució, que podria ajudar a fer créixer el projecte a mida que va passant el temps, per evitar que es quedi estancat.

2.2.1 Headless Chrome

Vanilla

La primera opció que es va proposar va ser utilitzar Headless Chrome. A partir de la versió 59 de Chrome (actualment està per la versió 90), és possible executar-lo de manera *headless*, el que significa que es pot utilitzar des de terminal, o més interessant, des de un servidor sense cap tipus de UI. Es va crear pensant en que podria ser molt útil, per fer testos automatitzats de webs o aplicacions online.

Per executar-lo és molt fàcil, només cal afegir uns quants paràmetres d'inici al Chrome. Els paràmetres d'exemple són:

```
chrome \
--headless \                      # Runs Chrome in headless mode.
--disable-gpu \                    # Temporarily needed if running on Windows.
--remote-debugging-port=9222 \
https://www.chromestatus.com # URL to open. Defaults to about:blank.
```

Listing 1: Config Headless Chrome

Encara que està pensat per fer automatitzacions com he comentat abans, es pot aprofitar la potència de Chrome per altres tasques, com poden ser la reproducció de vídeos, crides a APIs, entre altres.

Buscant algun exemple per internet que aprofités Headless Chrome per la reproducció de vídeo, ens vam trobar amb uns quants que havien pensat el mateix.

En un article [4] trobem que van aconseguir fer streaming a Facebook i Youtube Live amb uns resultats bastant satisfactoris. A partir d'una web (vídeo + àudio), ho convertien a un flux de sortida RTMP. Comenta que el més complicat va ser aconseguir que el vídeo i l'àudio estiguessin sincronitzats. Però finalment ho va aconseguir gràcies al controlador d'àudio anomenat PulseAudio [5].

Headless Chrome està pensat per ser utilitzat amb alguna eina d'automatització, i així ho recomanen des de el propi blog oficial de Google Developers [6]. Degut a que no disposem de cap tipus de UI per debugar o interactuar, utilitzarem una eina externa. Al blog de Google ens parlen de Puppeteer, Selenium, WebDriver i ChromeDriver. Totes elles tenen el mateix objectiu, automatitzar tasques al navegador. Podríem dir que la més desenvolupada i consolidada, és Selenium, però Puppeteer també s'ha guanyat molt de renom en els últims temps, per tant, haurem de fer un estudi per veure els pros i contres de cada una i decidir amb quina d'aquestes ens quedarem.

Selenium

Selenium és una eina gratuïta i open-source [7], que serveix per crear tasques automatitzades en un navegador o aplicació per validar funcionalitats. Es poden utilitzar scripts tant de C#, Python, JavaScript, PHP, Scala... Realment Selenium com a tal és una Suite que es compon d'un grup d'eines; el seu propi IDE, Grid i WebDriver. Nosaltres estem interessats en l'últim, per tant és el que estudiarem.

Selenium WebDriver pot controlar un navegador tal i com ho faria un usuari, ja sigui de manera local o remota en un servidor.

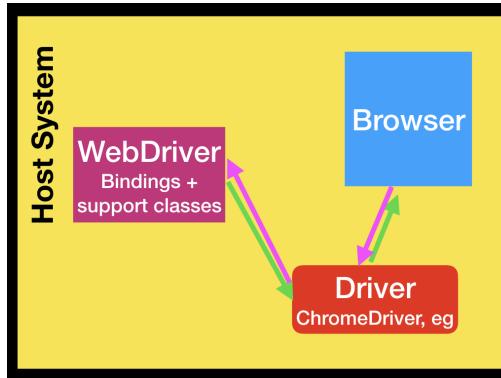


Figure 2: Selenium Driver Graph

Com podem veure a la figura 2, WebDriver es comunica amb el navegador a través d'un driver. Aquest driver és específic de cada navegador. El que ens interessa és el de Chrome, per tant sabem que utilitzarem el ChromeDriver.

Els navegadors que suporta són: Chrome, Chromium, Firefox, Internet Explorer, Opera i Safari. A la seva documentació [8] no comenten en cap lloc compatibilitat amb Headless Chrome, però degut a que està basat en el propi navegador, i no fa cap canvi brusc en el seu funcionament, podem assegurar que també serà compatible. Ho podem corroborar amb un article on ho experimenta [9], en el que expliquen que només hem d'indicar el driver corresponent a Chrome, i escriure a les opcions, el flag «`--headless`».

Puppeteer

Per altre banda tenim un altre eina bastant més nova, anomenada Puppeteer [10]. Les funcionalitats principals venen a ser molt semblants a Selenium, controlar totes aquelles coses que podries fer manualment al navegador, fer-ho de manera automàtica. A més de funcionalitats molt útils com fer captures de pantalla i crear PDFs.

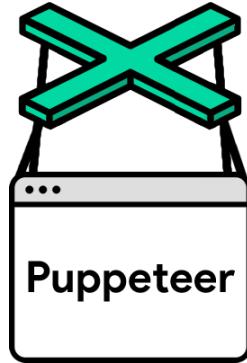


Figure 3: Puppeteer Logo

És open-source i està sota la llicència d'Apache 2.0 [11] pel que es pot utilitzar i comercialitzar de manera totalment lliure tot aquell contingut que tingui les seves eines.

És molt fàcil d'instal·lar, ja que només cal instal·lar un paquet npm i per utilitzar-lo es fa tot amb javascript mitjançant les comandes que disposen a la seva documentació. A més de tot això, la major avantatge que té és que està pensat per ser utilitzat amb Headless Chrome, dóna moltes facilitats per realitzar totes les tasques sense estar veient cap tipus de UI, i a la vegada poder debugar de manera còmode i senzilla.

Un petit codi d'exemple que ens mostren a la seva documentació per comprovar com de fàcil és utilitzar-lo. El que fem és afegir Puppeteer al nostre script de js amb require, iniciem el navegador en mode headless, obrim el link «<https://example.com>», fem una captura i tanquem el navegador.

```
const puppeteer = require('puppeteer');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('https://example.com');
  await page.screenshot({ path: 'example.png' });

  await browser.close();
})();
```

Listing 2: Puppeteer Get Started Example

2.2.2 WebGL Headless

WebGL és una API escrita en Javascript utilitzada per la renderització de gràfics en 3D al navegador. Està basat en la Open Graphics Library (OpenGL), pel que només necessitarem que sigui compatible amb aquest [12], sense cap altre dependència que podria dificultar la compatibilitat. Hem de tenir en compte que encara que el navegador sigui compatible, per tota la renderització 3D, per molt mínima que sigui, necessitarem una targeta gràfica. Això pot ser un problema ja que es vol executar tot en un model al núvol, al donar estabilitat i professionalitat. La majoria de gent que utilitza servidors té unes idees bastant diferents al que es vol fer aquí, i potser necessiten potència de CPU o molta RAM, però no GPU, pel que potser és complicat trobar una màquina remota amb la potència suficient que es necessita.

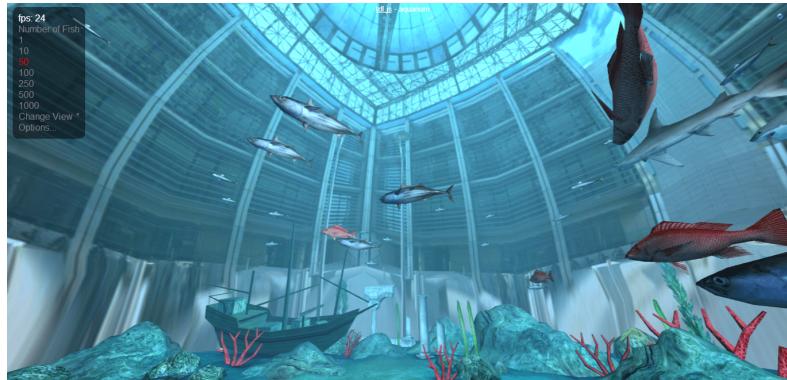


Figure 4: Exemple WebGL

Per una banda aquesta solució és molt atractiva ja que es faria tot el sistema des de 0 sense utilitzar cap tipus de dependència, però a la vegada també depens de les polítiques o limitacions de potència del navegador. Mai serà capaç d'utilitzar el 100% de la màquina i no és comparable amb una aplicació standalone.

2.2.3 Unreal Engine 4

Un dels motors gràfics més utilitzats dels últims temps és Unreal Engine. Està creat per Epic Games, programat en C++ i amb una potència sorpre-

nent que fa que sigui el més utilitzats en projectes foto realistes, o inclús, en projectes d'arquitectura. Sobretot a partir de 2019, on es van ajuntar amb l'empresa Quixel, dedicada entre d'altres coses, a crear «megascans». És una tecnologia que disposa textures amb grans resolucions i qualitats sorprenents, però a la vegada molt eficients pel desenvolupament de videojocs. Així aconsegueix un renderitzat molt realista a temps real, i destacant entre els altres motors competidors.



Figure 5: Captura Quixel Demo Megascan a UE4 (Youtube)

Durant el transcurs del 2021 es llençarà la versió d'Unreal Engine 5, on han demostrat que poden inclús superar-se i crear uns entorns sorprenents ajuntant la tecnologia que ja havien estat desenvolupant de Quixel, amb una nova anomenada Lumen, que controlarà totes les llums i farà ús del Ray-Tracing a temps real, també dit RTX a les targetes gràfiques de Nvidia. Amb totes aquestes novetats i evolucions tecnològiques que han anat fent en els últims anys, és lòtic que s'hagin guanyat entre la comunitat el nom a millor motor gràfic en quant a qualitat visual [13]. Però com bé sabem, això no és l'únic que importa.

L'Unreal disposa d'un sistema de programació anomenat Blueprints, és una metodologia de programació totalment visual i basada en nodes. Bàsicament són petites capces que internament tenen trossos de codi en C++, els quals adjuntem entrades i sortides.

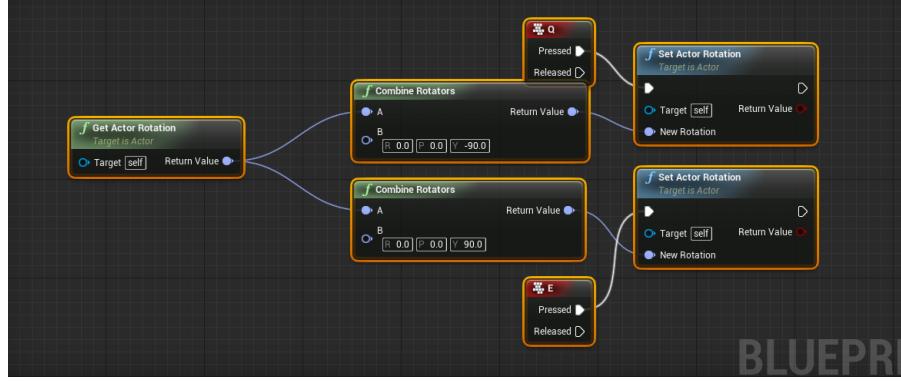


Figure 6: Rotació Programada en Blueprint UE4

Això pot ser molt còmode i és un punt a favor per la gent que comença a desenvolupar videojocs, si estàs acostumat a utilitzar-ho o no es tenen els coneixements necessaris de programació i vols fer scripts amb una complexitat baixa-mitjana. Però pel contrari si es volen fer scripts més complexes pot ser que ens trobem amb certs inconvenients i poden quedar scripts amb milers de nodes i que sigui molt difícil d'interpretar o debugar.

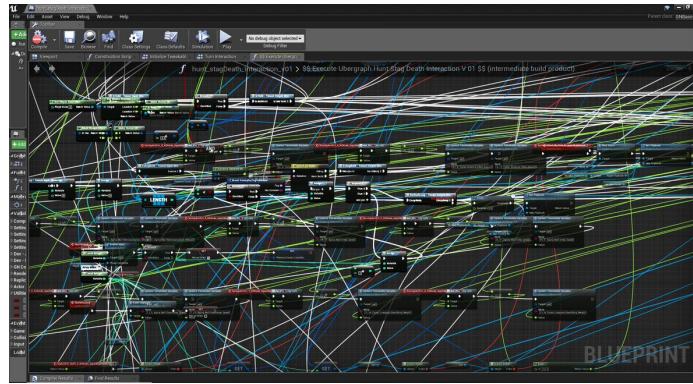


Figure 7: Exemple Blueprint Caòtic UE4

Altres inconvenients poden ser que les idees que es tenen surtin fora de la norma, que els nodes estiguin massa limitats o que simplement no t'agradi, per això també et donen la opció a crear un script buit i fer-ho tot amb C++. I llavors veiem que la documentació que es pot trobar és molt inferior a la dels blueprints, i possiblement sigui més complicat i més llarg de fer que els seus competidors, com poden ser Unity o Godot.

2.2.4 Unity

El motor de videojocs més famós i més utilitzat des de els seus orígens és sense dubte Unity [14]. Va ser creat al 2005 a Copenhaguen per 3 amics, David Helgason, Joachim Ante i Nicholas Francis.

És el motor que suporta més plataformes actualment [15], com Windows, Mac OS, Linux, iOS, Android, WebGL, Playstation, Xbox, Nintendo Switch, Stadia i moltes altres més. Això és un punt molt a favor, ja que necessitem desenvolupar l'aplicació per un servidor Linux, i Unity és el motor més professional dels que donen suport. Altres motors com Unreal Engine, Game Maker o Cry Engine no accepten el sistema operatiu Linux com a opció de compilació, o funcionen amb bastants problemes.

La programació dels scripts es fa amb llenguatge C# el qual forma part de la plataforma .NET propietària de Microsoft. És un llenguatge derivat de C/C++ i és dels més utilitzats juntament amb Java. Si ens fixem en el rànquing de plataforma TIOBE [16], que és una web que s'encarrega d'ordenar els llenguatges de programació de més utilitzat a menys, podem veure que en el top 5, tenim els 3 llenguatges de C, els quals tenen una base molt semblant. Després tindríem Java i Python, tots dos són llenguatges d'alt nivell molt senzills i que faciliten molt el flux de programació.

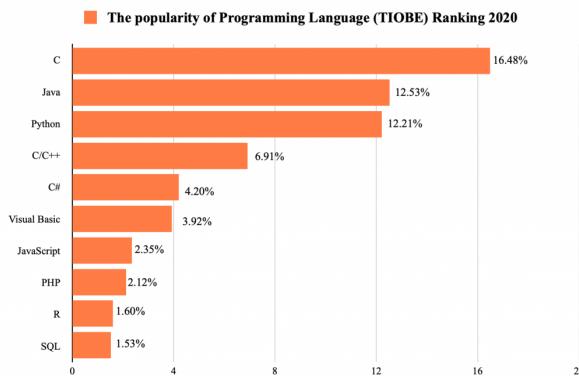


Figure 8: Llenguatges de programació més utilitzats al 2020

A més de que Unity utilitzi C# que ja és un llenguatge d'alt nivell de per si, s'utilitza la API MonoBehaviour [17], que estalvia molta feina i permet fer

tasques que a priori són complicades o llargues, en pocs minuts i en poques línies.

MonoBehaviour és la classe base d'on parteixen tots els scripts de Unity. Només cal indicar que el nou script s'estén de la API.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

}
```

Listing 3: Unity Code Basic Template

En les últimes versions de Unity, han apostat molt per competir directament amb Unreal Engine, en un dels punts que tenien més diferenciats, l'atractiu visual hiperrealista.

Quan Unity va començar, com que era relativament fàcil d'utilitzar i gratis, van aparèixer una quantitat immensa de jocs de mòbil creats amb aquest motor, pel que es va guanyar una fama errònia, de que aquest motor només servia per jocs petits, amb uns gràfics tirant a simples i sense molts efectes.

Per tant, al 2019 van llençar amb la seva última actualització, una nova funció anomenada HDRP (High Definition Render Pipeline) [18], la qual millorava tot el sistema del render del motor, aconseguint uns resultats que van sorprendre a tota la comunitat, i que no tenien res a envejar amb Unreal Engine.



Figure 9: Captura Book of the Dead Unity Demo HDRP (Youtube)

A més, podem trobar que el videojoc més realista que s'ha creat mai en quant al gènere de FPS (First Person Shooter), van decidir crear-lo amb el motor Unity, enllloc d'Unreal, i van aconseguir uns resultats molt bons, fins al punt que està considerat el joc amb millors gràfics, creat amb aquest motor. Es tracta del videojoc *Escape from Tarkov*.



Figure 10: Captura Escape from Tarkov Pre-Alpha

2.3 Tractament dels fluxes de video

2.3.1 Formats Compatibles

L'aplicació haurà de rebre tot tipus de vídeos, de tots els formats possibles, entenent formats el conjunt de còdecs i contenidors. Primer s'ha de fer un

anàlisi dels formats que existeixen i com de populars són.

Còdecs

Hi ha molts tipus de còdecs, però és cert que alguns s'utilitzen molt poc, o ja s'han quedat antiquats.

- H.261: Aquest còdec va ser el primer que es va tornar popular, degut a que era el que donava millors resultats en el seu moment. Però evidentment, ha passat molt temps i la tecnologia ha evolucionat suficient com per deixar aquest còdec antiquat.
- H.263: Va ser el primer que còdec realment eficient, però a la vegada demanava molts recursos de l'ordinador per codificar un vídeo, i a la vegada, per reproduir-lo necessitarem una potència relativament alta.
- H.264: És el més popular a l'actualitat de manera professional, també conegut com MPEG-4 AVC, compta amb una compressió molt alta amb una pèrdua de qualitat imperceptible. A més de ser molt polivalent al acceptar molts tipus de bit rate d'entrada.
- H.265: També anomenat HEVC, és el successor del H.264. A vegades pot donar uns resultats pitjors al seu predecessor, però sobretot pels vídeos en 4K obtenim uns molts millors resultats ja que comprimeix molt més obtenint uns resultats de qualitat molt semblants. En els vídeos 4K es pot obtenir fins a un 64% de reducció de bit rate en comparació amb H.264.
- MPEG-1: És un còdec bastant antic que ja no s'utilitza, però va ser molt utilitzat per discs de dades CD-ROM i inclou compressió tant de vídeo com d'àudio.
- MPEG-2: Dona bons resultats amb el DVD, però requereix pagar llicència per poder-lo utilitzar.
- MPEG-4: S'utilitza molt per contingut de vídeos a la xarxa, ja que té una gran compressió de vídeo i àudio, però la qualitat pot baixar al utilitzar baixos nivells de bit rate.

- DivX: Comunament utilitzat per comprimir pel·lícules DVD i famós per impulsar la pirateria al voltant de l'any 2000 al permetre comprimir fins a 7 pel·lícules en un sol disc DVD [19].

Contenidors

- AVI: Cada cop és menys utilitzat, però a l'anterioritat va ser molt popular per la gran compatibilitat de la que disposava en diferents sistemes operatius.
- MOV: A diferència de l'AVI, aquest còdec creat per Apple, era molt poc o gens compatible amb altres sistemes operatius, pel que només s'utilitzava amb els seus dispositius.
- MPG: Són un conjunt de formats, i alguns d'ells han arribat a ser els més famosos, sobretot el MPG-4, també anomenat MP4. Que té molta qualitat, comparable al MOV, i actualment és pràcticament impossible trobar algun ordinador que no sigui capaç de reproduir-lo.
- WMV: Windows Media Video és un format propietari de Windows, i necessitarem reproduir-lo amb el Windows Media Player per aconseguir els millors resultats d'aquest.
- MKV: Acrònim de Matroska, és un format molt utilitzat per emmagatzemar més d'un arxiu en un sol. Per exemple podem trobar un arxiu MKV que contingui més d'una pista d'àudio (per diferents idiomes), alguns arxius de subtítols o inclús diferents arxius de vídeo en un sol.
- FLV: El format de vídeo de Flash Player, que és molt compatible amb tots els ordinadors, però cada cop s'està quedant més antiquat.

Els reproductors més famosos, com són VLC, MPV, o MPlayer, accepten tots els tipus de còdecs i contenidors, pel que serà necessari i pràcticament requisit que l'aplicació també els accepti tots per a que no es quedí enrere. Ja sigui creant un reproductor propi, o un basat en llibreries de tercers.

Imatges

En quant a les imatges, també trobem molts formats diferents i s'haurà de tenir en compte per decidir si es podrà donar suport a tots ells, o pel contrari s'haurà de prescindir d'algun. Com a mínim hauria d'acceptar els més famosos, que són:

- BMP: Creat per l'empresa Microsoft, és l'estàndard del sistema operatiu Windows.
- JPEG: Un dels formats més utilitzat degut a la seva gran compressió, mantenint una qualitat molt acceptable.
- GIF: El GIF és un format diferent a la resta, ja que consta d'un conjunt d'imatges que es reproduueixen de manera continua, formant un petit vídeo. Això si, la seva qualitat queda molt limitada.
- PNG: Disposa d'una gran compressió mantenint la qualitat, com podria ser la del JPEG, però, amb la gran avantatge de que disposem de l'atribut alpha, el que permet tenir imatges amb transparències.

Àudio

Els vídeos tindran el seu propi àudio, i encara que no està pensat com a requisit que es pugui reproduir àudio de manera independent, com podria ser una música de fons o una veu en off. Els formats que s'haurien de suportar per poder tenir un mínim de compatibilitat són:

- WAV: Desenvolupat per Microsoft i un dels més utilitzats, és compatible amb pràcticament tots els còdecs. Accepten arxius d'una mida gran, fins a 4GB, i poden estar molt comprimits o poc.
- MPEG: Un dels formats més famosos per ser un estàndard en àudio i sobretot, en la música, podem trobar per exemple els còdecs MP2 (més utilitzat per aplicacions de broadcast) o el MP3.

2.3.2 Protocols de Streaming

En quant a la sortida del vídeo/àudio resultant, s'haurà de decidir quin protocol s'utilitza, ja que es voldrà una sortida en streaming. El vídeo en streaming, o retransmissió en directe, és la emissió d'un flux de dades a temps real, o amb el menor retard possible, a un destí. Per exemple, quan mirem la

televisió, tenint en compte que és per internet i no per antena, estem rebent un flux de dades que arriba des de uns servidors centrals de la cadena en particular. Es van començar a fer experiments de televisió fa molts anys, però la majoria van ser un fracàs degut al gran ample de banda que demanaven els vídeos. Per tant el que va sorgir primer van ser els serveis de VOD (Video on Demand). No s'ha de confondre el VOD amb el Streaming ja que son dos conceptes totalment diferents i que no utilitzen les mateixes tecnologies. Un exemple de servei VOD que va triomfar és la plataforma Youtube. Aquesta tecnologia va anar avançant fins que van millorar les connexions a internet i això va provocar que per fi el Streaming fos possible. L'objectiu del directe, és tenir un retard mínim, amb una qualitat bona. Actualment el mínim de qualitat per una plataforma professional és la resolució 1920x1080. La resolució 4K dona molt bona qualitat, però requereix una quantitat molt alta d'ample de banda i actualment grans plataformes de streaming com Youtube o Twitch no ho permeten com a opció. Encara i això serà un repte a tenir en compte i seria un gran punt a favor respecte a la competència.

Els protocols de streaming són els encarregats de posar les «normes» en com es transmetrà el flux de dades, a més de contemplar els errors i minimitzar-los en tot el possible.

Un protocol no té res a veure amb un còdec o un format, sinó que és complementari a aquests i només s'ocupa de la transmissió.

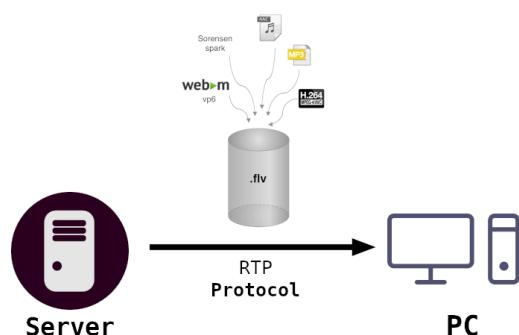


Figure 11: Diagrama Protocol-Format-Códec

Tots els protocols hauran d'utilitzar un dels dos mètodes de transmissió que existeixen, que són TCP (Transmission Control Protocol) o bé, UDP (User Datagram Protocol). La diferència més important i notòria entre aque-

sts, és que el TCP, requereix d'una resposta del client, forçant a que tots els paquets arribin, el que fa una connexió més segura, però més lenta. Quan TCP no aconsegueix enviar un paquet correctament, repeteix el procés fins a aconseguir-ho. Per altre banda, UDP ignora això, simplement envia els paquets, i el receptor ja s'ocuparà de rebre-ho correctament. Això provoca que a vegades es puguin percebre fragments en el vídeo i pèrdua de qualitat, però és molt més ràpid i té menys retard.

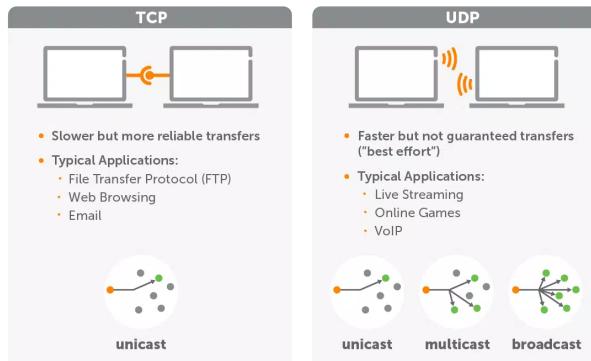


Figure 12: Comparació TCP-UDP

Tornant als protocols de live streaming, hi ha molts per escollir, pel que es comentaran els més importants.

- RTMP (Real Time Messaging Protocol): Protocol TCP. És propietat d'Adobe, pel que era molt utilitzat a les webs que disposaven d'Adobe Flash Player. No obstant, s'ha quedat bastant obsolet i presenta problemes de compatibilitat amb còdecs moderns i la seguretat és baixa. Per altre banda, els resultats que dona són de bona qualitat i té molt suport amb les diferents plataformes i software. Els còdecs més recomanats per aquest protocol són H.264 pel vídeo, i AAC per l'àudio.
- WebRTC: Protocol TCP i UDP. Utilitzat per la gran majoria de desenvolupadors web, degut a la seva gran compatibilitat amb tots els navegadors moderns: Chrome, Firefox, Opera, Safari, Edge... És de gran qualitat i suporta els còdecs VP8 i VP9. Per l'àudio, tenim que WebRTC suporta el còdec Opus, gairebé el més utilitzat per vídeos en directe. A més, amb la avantatja de ser open-source, el projecte està en una constant evolució, i pròximament suportarà el còdec H.265, i

més important encara el AV1, que promet una gran millora en tots els aspectes [21]. Té el millor resultat de latència, podent arribar a valors menors a 1 segon.

- FTL (Faster Than Light): Protocol UDP. Creat per l'empresa Microsoft, dedicat a la plataforma ja extinta Mixer [22]. Una proposta que no va sortir del tot bé, i que pretenia competir directament amb Twitch [23]. Aquest protocol té com a objectiu ser el més ràpid, com indica el seu nom, per poder comunicar-te amb els visualitzadors del directe d'una manera pràcticament instantània. Tenint en compte que Twitch utilitza RTMP, i ho converteix a HLS [24] (molt més lent que la proposta de Mixer), no era una mala idea. Encara i això, Twitch ja era massa gran i estàndard com per poder competir.
- SRT (Secure Reliable Transport): Protocol UDP. També es un protocol open-source, al igual que WebRTC, i és considerat com a l'evolució de RTMP. És de molta qualitat i bastant més estable als anteriors. Els objectius d'aquest projecte són crear un protocol que transmeti vídeos sense soroll al senyal (jitter) i que eviti la pèrdua de paquets.

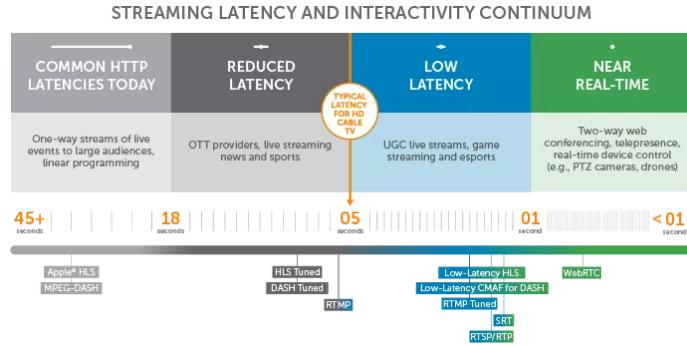


Figure 13: Gràfic Latències Protocols de Streaming

Resumint, no hi ha protocols més bons o més dolents, depenen el projecte que es vulgui fer, serà més adequat un o altre. Pel projecte que s'ha proposat en aquest treball, es necessitarà sobretot un protocol amb poc retard.

	RTMP	WebRTC	FTL	SRT
Pros	Estabilitat, gran compatibilitat i qualitat	Open-source, còdecs actuals i molt baixa latència	Qualitat adaptativa i molt ràpid	Molta qualitat, estabilitat i compatibilitat de còdecs
Contres	Molta latència, més insegur que altres opcions, còdecs antics	Constantment en desenvolupament i pot ser inestable	Qualitat inferior a la resta i poca compatibilitat	Està en desenvolupament, no hi ha gaire documentació
Video Còdecs	H.264	VP8, VP9, H.264 (H.265 i AV1 futur proper)	H.264	-
Àudio Còdecs	AAC	Opus	Opus	-
Latència	3-30 segons	1 segon o menys	1 segon o menys (menor a WebRTC)	1 segon o menys

Table 1: Taula resum dels protocols esmentats

2.3.3 Llibreries de processament

La gran majoria d'aplicacions que utilitzem a diari que disposen de reproductors multimèdia, o qualsevol tipus de processament de vídeo/àudio, utilitzen llibreries externes per fer aquest procés més senzill. Hi ha varies opcions de llibreries de processament de contingut multimèdia. Totes elles tenen un desenvolupament complex, és per això que hi ha poca rivalitat, i totes disposen d'una API per utilitzar-les. Les més utilitzades i conegudes, són dues que es comentaran a continuació:

FFMPEG

És la llibreria de processament de vídeo/àudio més gran que existeix, i és utilitzada per la gran majoria de desenvolupadors.



Figure 14: Logo FFmpeg

Concretament, el que defineix FFmpeg, és un conjunt de software lliure (framework), que tal i com diu a la seva documentació [25], les funcionalitats que permet són:

- Descodificar: és l'acció de convertir un flux de dades en un contingut multimèdia visible i amb sentit per a un receptor.
- Codificar: és l'acció contraria a descodificar, convertir un contingut multimèdia a un flux de dades, ja sigui per una transmissió més còmode, comprimir, o el que es desitgi.
- Transcodificar: significa convertir un contingut multimèdia d'un còdec a un altre. Per exemple, ens podem trobar amb un contingut en H.265, que el volem reproduir en un ordinador relativament antic, que no accepta aquest còdec, pel que haurem de transcodificar-lo a un altre que l'ordinador si que «entengui» com podria ser el H.264.
- Multiplexar: si tenim vídeo i àudio per separat i ho volem ajuntar, el que haurem de fer es un mux, és a dir, ajuntar dos o més fluxos d'entrada, en un de sortida.
- Desmultiplexar: és el contrari de multiplexar, tenint un sol flux d'entrada, extraiem dos o més.
- Stream: si enlloc de voler generar un fitxer de sortida, el que volem és directament crear un streaming, utilitzant el protocol més apropiat, ho podem fer tot des de FFmpeg i sense utilitzar cap eina externa.

- Filtres: per si totes les funcionalitats anteriors fossin poques, també permeten aplicar filters als àudios o als vídeos, el que ho fa molt còmode per poder automatitzar tasques des de terminal, sense haver d'utilitzar editors de vídeo o àudio. En quant a filters d'àudio disposem de 105 efectes, entre ells els més senzills com el volum, compressors... I en quant al vídeo, disposem de 264 efectes, com per exemple, fer un crop, escalar, rotar... [26]
- Reproduir: FFmpeg també inclou una eina anomenada FFplay, que permet reproduir tot tipus d'arxius sense problemes. Normalment s'utilitza com a eina de test.

Es va desenvolupar principalment per GNU/Linux, però està disponible per tot tipus de plataformes com Windows i MacOS. S'utilitza a través d'un terminal, però també es pot integrar en tot tipus d'aplicacions per utilitzar-lo internament, com fa Audacity, Youtube, Chrome, OBS Studio, i molts altres grans projectes.

```
ffmpeg -i input -map 0 -c:v libx264 -crf 18 -c:a copy output.mkv
```

Listing 4: Exemple Transcodificació d'un Arxiu a H.264

La instal·lació d'aquesta eina és molt senzilla, sobretot si s'utilitza Linux, o concretament Ubuntu, es pot aconseguir introduint un parell de línies al terminal.

```
sudo apt-get update
sudo apt-get install ffmpeg
```

Listing 5: Instalació FFmpeg Ubuntu

També és molt utilitzat per la compressió de vídeos de manera automàtica, per exemple, si es puja un vídeo a «Twitter», no es pujarà el vídeo original, ja que el pes de l'arxiu seria excessiu per emmagatzemar-ho al seus servidors, sinó que es fa una copia reduïda del vídeo, i aquest és el que es guarda.

LibVLC

Per altre banda trobem un altre gran framework, l'únic capaç de competir amb FFmpeg, i és LibVLC. Està desenvolupat per la organització VideoLAN, on tenen com a objectiu, des de la seva fundació, desenvolupar aplicacions open-source gratuïtes. Van començar el projecte a una universitat de França l'any 1996, i va començar a ser open-source a partir de l'any 2001, però, a l'any 2009 van decidir seguir amb el projecte de manera independent i trencar tots els vincles amb l'escola [27]. Ara mateix tenen desenvolupadors per 40 països diferents, i animen a la comunitat a contribuir i recolzar el projecte ja sigui programant, traduint, donant components i material, o amb alguna aportació econòmica [28].

El major projecte creat per aquesta organització és el mundialment conegut VLC, un reproductor multimèdia basat en les seves pròpies llibreries LibVLC.



Figure 15: Captura de la primera versió de VLC

LibVLC no està tant pensat per utilitzar-lo des de terminal com podria ser FFmpeg, i encara que també es pot integrar a qualsevol projecte, és més complicat de instal·lar i utilitzar [29]. A continuació es mostra el codi base o plantilla del que partim per poder utilitzar libVLC mitjançant un script en C++.

```
#include <stdio.h>
#include <stdlib.h>
#include <vlc/vlc.h>

int main(int argc, char* argv[])
{
```

```

libvlc_instance_t * inst;
libvlc_media_player_t *mp;
libvlc_media_t *m;

/* Load the VLC engine */
inst = libvlc_new (0, NULL);

/* Create a new item */
m = libvlc_media_new_location (inst, "http://mycool.movie.com/test.mov");
//m = libvlc_media_new_path (inst, "/path/to/test.mov");

/* Create a media player playing environement */
mp = libvlc_media_player_new_from_media (m);

/* No need to keep the media now */
libvlc_media_release (m);

#if 0
    /* This is a non working code that show how to hooks into a window,
     * if we have a window around */
    libvlc_media_player_set_xwindow (mp, xid);
    /* or on windows */
    libvlc_media_player_set_hwnd (mp, hwnd);
    /* or on mac os */
    libvlc_media_player_set_nsobject (mp, view);
#endif

/* play the media_player */
libvlc_media_player_play (mp);

sleep (10); /* Let it play a bit */

/* Stop playing */
libvlc_media_player_stop (mp);

/* Free the media_player */
libvlc_media_player_release (mp);

libvlc_release (inst);

return 0;
}

```

Listing 6: Codi base d'exemple de libVLC

3 Proposta tècnica

3.1 Elecció de l'eina i justificació

Per poder decidir amb criteri i assegurar-se de que es pren la decisió correcte sobre quina és l'eina que s'ha de utilitzar, es farà una prova amb cada una de les eines esmentades (Chrome Headless, Unreal Engine 4 i Unity).

3.1.1 Chrome Headless

És molt útil per tasques automatitzades, no es necessita cap tipus de software extra, com a molt es poden necessitar llibreries de nodejs pels scripts de javascripts, però en tot cas, quedaria tot empaquetat en un sol projecte. Al utilitzar el navegador de Chrome, això ens dona moltes avantatges en quant a simplicitat de les tasques, però també ens limita la potència, compatibilitat de còdecs i formats, pel que mai serà exactament igual que si executessim les mateixes tasques de manera nativa al ordinador.

Si busquem projectes creats amb aquesta eina que s'assemblin a l'objectiu que volem aconseguir, trobem una aplicació creada per «Sebastian Pereyro», de la web Empirical [30]. Headless Chrome està disponible a partir de la versió 59, actualment van per la 91, pel que no seria cap problema utilitzar aquesta eina. Es poden carregar pàgines web i executar tests o tasques, a més de generar PDFs i fer captures de pantalla repetidament. A més comenta que es pot compartir de manera directa el contingut «Screencast content», això si, fent múltiples captures de pantalla. L'experiment que es va proposar va ser de capturar àudio i vídeo de manera automàtica, tant d'una web pròpia com d'una web externa i transmetre el contingut a Facebook, Youtube, Twitch...

El que utilitza com a eines, són Node.js, el navegador Headless Chrome en qüestió, Pulseaudio que és el motor de vídeo que utilitzarà per la captura, i FFmpeg que farà la transmissió final del flux de Headless Chrome cap a les destinacions pertinents. Tot això ho vol fer utilitzant un servidor que s'ocupa de totes les tasques. Tenint una aplicació client, en aquest cas farà ús de Postman [31], es comunica amb el servidor central que tindrà tota la programació, agafa el contingut multimèdia d'un altre web, i l'envia als servidors multimèdia.

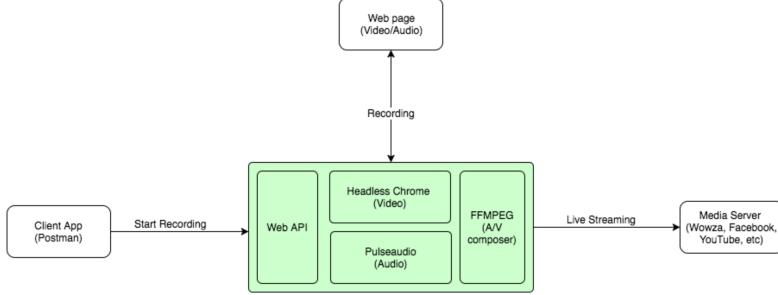


Figure 16: Gràfic Experiment Headless Chrome

```
// Init remote interface to chrome port
function initRemoteInterface(chrome){
  const port = chrome.port;
  logger.log("Initialize Remote Interface on port: " + port);
  return ChromeRemoteInterface({port: port});
}

//Load Page
async function loadPage(url){
  await Page.navigate({url: url});
}

//Start screencast
function startCapturingFrames(){
  logger.log("Starting capturing screen frames...");
  return Page.startScreencast({
    format: "jpeg",
    quality: 100
  })
}
```

Listing 7: Experiment Headless Chrome Empirical

En el codi que ens proporciona 7, podem comprovar com fa la captura de la pantalla. Es tracta de una captura total de la finestra del navegador. Per a fer-ho, utilitza una funció integrada al Chrome, anomenada `startScreencast`. Pels paràmetres de configuració que necessita i veient la documentació de Chrome DevTools [32], podem deduir que es tracta un petit script que envia la comanda de fer captura de pantalla, en el format que desitgem (en el cas d'exemple és en jpeg), i junta aquests frames per aconseguir tenir un vídeo. En quant a l'àudio, ens comenta que encara que era un dels seus reptes, no està suportat per Chrome, pel que no es podrà capturar de manera nativa. La opció més vàlida serà capturar-lo per separat, i més tard ajuntar-ho, el que

farà que hi hagi problemes de sincronització d'àudio/vídeo, un problema molt comú en aquest tipus de projectes. La qualitat dels vídeos tampoc podrà ser molt alta, ja que headless chrome no suporta acceleració per hardware amb GPU, ho veiem a l'exemple 1 amb el flag `--disable-gpu` extret directament de la documentació de Google [6]. Un altre cosa que comenta és que el framerate serà variable, nosaltres podem enviar 30 comandes per segon de que faci la captura de pantalla, però és pràcticament impossible que vagi totalment sincronitzat i faci exactament les 30 captures cada segon, un dels problemes que comporta això, és que s'anirà dessincronitzant poc a poc, cada cop més, l'àudio.

Després de fer totes les captures, o mentre s'estan fent, es pot utilitzar FFmpeg per ajuntar les imatges en una seqüència de vídeo, ajuntar-li l'àudio, i transmetre-ho a on es desitgi, ja sigui en directe o a un fitxer local.

En aquest projecte, no s'ha utilitzat cap eina d'automatització de navegador, sinó que s'ha fet tot amb Headless Chrome Vanilla. Pel que es farà una prova utilitzant totes les eines del projecte, i a més, amb Puppeteer per poder-ho automatitzar tot.

Per la primera prova, es programarà tot en HTML i Javascript. Per una banda tindrem el `index.html`, un arxiu molt senzillet que contindrà un o dos vídeos locals que es reproduiran automàticament a dins un canvas, al obrir l'arxiu.

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="index.css">
  </head>
  <body>

    <div id="element-to-record">
      <video autoplay muted controls class="myvideo" id="myvideo">
        <source src="/localVideos/NightcrawlerTrailer.mp4" type="video/mp4">
      </video>
      <video autoplay muted controls class="myvideo2" id="myvideo2">
        <source src="/localVideos/StalkerTrailer.mp4" type="video/mp4">
      </video>
    </div>

    <canvas id="background-canvas" style="position:absolute; top:-99999999px; left:-999999999px;"></canvas>

  </body>
</html>
```

Listing 8: Test Chrome Headless | index.html

L'aspecte de la web és correcte, tenim un vídeo, concretament, el tràiler de la pel·lícula *Nightcrawler* [33] ocupant tota la pantalla en el fons, i a la cantonada superior esquerra, ocupant un quart de la pantalla tenim el tràiler de la pel·lícula *Stalker* [34].



Figure 17: Resultat index.html

Un dels primers problemes que han sorgit, ha sigut amb la reproducció dels vídeos, degut a que un navegador no està pensat exclusivament per aquestes funcions, les seves polítiques d'ús poden canviar dràsticament d'un dia per l'altre i destrossar un projecte d'aquest estil. Tant és així que a l'any 2017, van introduir una nova política a la versió 71, en la que no estava permès auto reproduir els vídeos automàticament al obrir una web, sinó que el usuari havia de clicar manualment al vídeo. Això si, permetia reproduir-los si estaven sense so. Una política probablement introduïda per evitar el ús de publicitat molesta, però que hauria canviat per complet aquest projecte i hauria fet que deixés de funcionar. Per tant, les opcions que es tenen, és reproduir els vídeos sense àudio (inviable ja que l'àudio és un dels requisits), o utilitzar una eina externa com Puppeteer per fer «trampes» i clicar allà on vulguem fent creure al navegador que ho està fent un usuari qualsevol.

```
const puppeteer = require('puppeteer');
const PuppeteerVideoRecorder = require('puppeteer-video-recorder');
var path = '/Output/';

init_puppeteer();
```

```

var global_browser;
const videosPath = "/Output/";

async function init_puppeteer() {

    //Executem Puppeteer
    global_browser = await puppeteer.launch({headless: true, executablePath: '/usr/bin/google-chrome', ignoreDefaultArgs: ['--mute-audio']});

    check_login()
};

async function check_login()
{
    try {
        const page = await global_browser.newPage();
        const recorder = new PuppeteerVideoRecorder();
        await recorder.init(page, videosPath);
        await page.setViewport({width: 1920, height: 1080});

        await page.goto('http://localhost:3000/', {timeout: 60000})
            .catch(function (error) {
                throw new Error('TimeoutBrows');
            });

        //Cliquem el boto que executra el script per desmutear els videos
        await page.click('#unmute');

        //Executem el script per a fer captures
        await page.click('#startStreaming');

    }
    catch (e) {
        console.log(' LOGIN ERROR -----');
        console.log(e);
    }
}

function delay(time) {
    return new Promise(function(resolve) {
        setTimeout(resolve, time)
    });
}
}

```

Listing 9: Test Reproducció Videos | videoPlayback.js

Amb aquest codi ja tindrem uns scripts que guardaran 30 captures per segon a una carpeta local. A la vegada, podem executar una comanda FFmpeg, per convertir aquestes captures en una seqüència de vídeo, i guardar-ho a un arxiu local. Això ho podrem aconseguir amb aquesta sèrie de comandes a la terminal.

```
//Executem el script de node videoPlayback.js
node videoPlayback.js
```

```
//Iniciem el ffmpeg per llegir totes les captures i convertir-les a un streaming
    rtmp
cd /Output/images/
ffmpeg -r 24 -pattern_type glob -re -i '*.jpg' -f flv rtmp://example.com

//FFplay Stream per reproduir-lo
ffplay rtmp://example.com
```

Listing 10: Comandes FFmpeg

Pel FFmpeg s'utilitzen els flags:

- -r 24: indica el número de frames per segon que tindrà el vídeo resultant.
- -pattern_type glob [36]: indiquem el patró del nom dels arxius per una lectura correcte, en aquest cas ens ajudàvem d'un txt que contenia tots els noms dels arxius, generat automàticament pel javascript.
- -re: indica que el processament s'ha de fer amb una velocitat de 1x, per evitar que processi les imatges més ràpid de 24 FPS i es vegi més ràpid del que hauria.

Aquesta solució no ha donat uns resultats gaire atractius, sobretot pel fet d'estar capturant la pantalla amb screenshots, i sense poder transmetre el àudio. Pel que s'ha provat un altre alternativa, basada en un altre projecte creat per l'usuari de Github FBSamples [37].

En aquesta altre proposta, trobem algunes variants interessants, el més diferent és que utilitza un servidor de websockets com a intermediari.

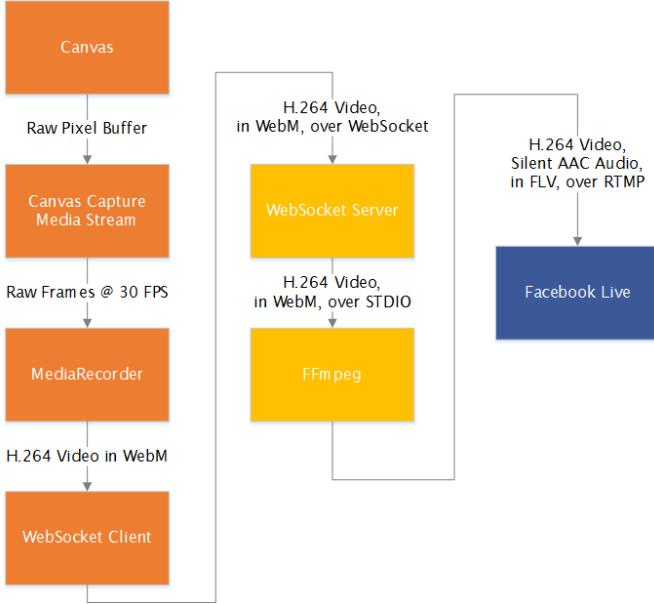


Figure 18: Arquitectura Projecte Canvas Streaming

Això sí, per capturar el canvas, utilitza MediaRecorder, que és el mateix plugin que s'utilitzava en la primera prova. Fent algunes modificacions al projecte de github de prova obtenim resultats amb el html creat anteriorment.

```

//Creacio del servidor websockets
//....
//Al iniciar servidor
wss.on('connection', (ws, req) => {
  // Ensure that the URL starts with '/rtmp/', and extract the target RTMP URL.
  let match;
  match = "rtmp://example.com";

  const rtmpUrl = "rtmp://example.com";
  console.log('Target RTMP URL:', "rtmp://example.com");

  // Launch FFmpeg to handle all appropriate transcoding, muxing, and RTMP
  const ffmpeg = child_process.spawn('ffmpeg', [
    // Facebook requires an audio track, so we create a silent one here.
    // Remove this line, as well as `--shortest`, if you send audio from the
    // browser.
    '-f', 'lavfi', '-i', 'anullsrc',
    // FFmpeg will read input video from STDIN
    '-r', '30',
  ]);
}
  
```

```

'-i', '-',
// Because we're using a generated audio source which never ends,
// specify that we'll stop at end of other input. Remove this line if you
// send audio from the browser.
//'-shortest',

'-f', 'pulse',
'-i', 'alsa_output.output_name.analog-stereo.monitor',
'-async', '1',
'-ac', '2',
// If we're encoding H.264 in-browser, we can set the video codec to 'copy'
// so that we don't waste any CPU and quality with unnecessary transcoding.
// If the browser doesn't support H.264, set the video codec to 'libx264'
// or similar to transcode it to H.264 here on the server.
'-vcodec', 'copy',
// AAC audio is required for Facebook Live. No browser currently supports
// encoding AAC, so we must transcode the audio to AAC here on the server.
'-acodec', 'aac',
// FLV is the container format used in conjunction with RTMP
'-f', 'flv',
// The output RTMP URL.
// For debugging, you could set this to a filename like 'test.flv', and
// play
// the resulting file with VLC.
rtmpUrl
]);
//FFmpeg error and close functions
//...
});

```

Listing 11: Servidor Websockets | server.js

Les avantatges amb aquest canvi és que aconseguim molta més qualitat de imatge, i es captura automàticament el so. Per altre banda, es segueix utilitzant Pulseaudio de manera autònoma, pel que seguirà havent un retard variant de vídeo/àudio.

Les conclusions que es treuen d'aquest experiment és que aquesta eina és interessant per fer algun petit projecte o que no requereixi de so, però no acaba d'encaixar en els requisits que es proposaven.

3.1.2 Unreal Engine 4

Per la prova amb Unreal Engine, s'ha utilitzat una tecnologia molt utilitzada de manera professional a la televisió, i que recentment ha estat integrada a Unreal com a plugin extern. Es tracta del software NDI (Network Device Interface) [38]. Creat per la empresa NewTek i amb una llicència privativa, permet una gran qualitat d'imatge, amb poc retard i una estructura molt estable. Dóna un grau important de professionalitat i compta amb un suport per part de NewTek i un fòrum de comunitat. Les desavantatges són, que al ser una empresa que recolza el software privat i tancat, no té cap mena de suport per Linux, només per Windows i MacOS. Sumant això a que la compatibilitat d'Unreal Engine tampoc és extraordinària, és bastant inabastable aconseguir bons resultats amb Unreal en una màquina Linux. Per tant, les proves que es faran seran des de una màquina Windows per comprovar la potència d'aquesta tecnologia.

Per començar, crearem un streaming de NDI des de VLC, per poder comprovar que funciona correctament tant entrada com sortida de fluxos. Un cop instal·lem les Tools de NDI, el plugin de VLC per poder treballar amb aquest software, s'instal·larà automàticament. Només cal entrar a les opcions de VLC i marcar com a motor de sortida «NewTek NDI Video Output».

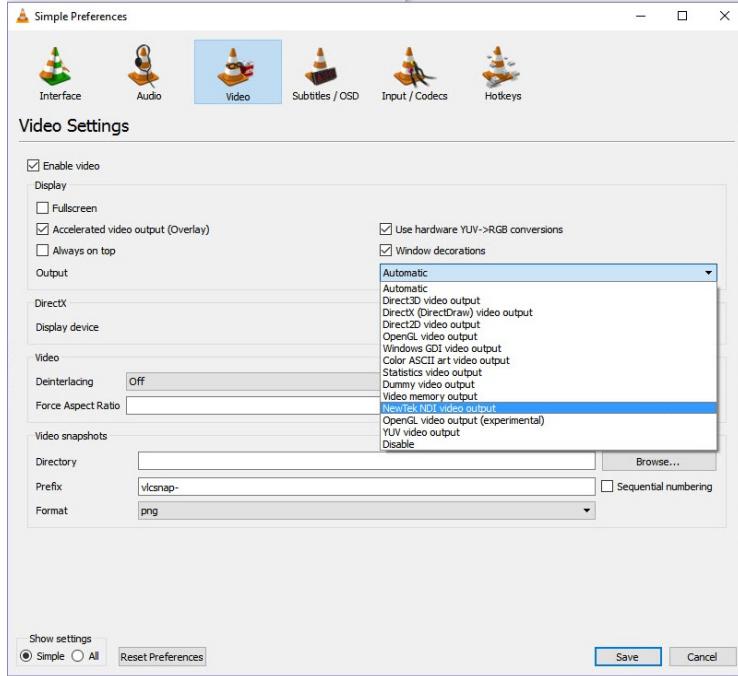


Figure 19: Configuració Sortida NDI a VLC

Encara que per aquesta prova s'hagi fet la sortida amb VLC, es pot fer de moltes altres maneres, com pot ser FFmpeg o una eina pròpia de NDI dedicada a això.

Un cop ja tenim un flux de vídeo creat, l'haurem de rebre a l'Unreal, per fer-ho, hem d'utilitzar el plugin corresponent, i seguir els passos de la seva documentació [39].



Figure 20: Node NDI Receiver

Ara que ja està tot programat, ja comencem a veure alguns resultats d'aquest software. Peraprofitar l'Unreal i provar altres efectes també, s'ha escollit un vídeo amb un chroma per comprovar com es comportava davant un streaming calculant-lo a temps real.

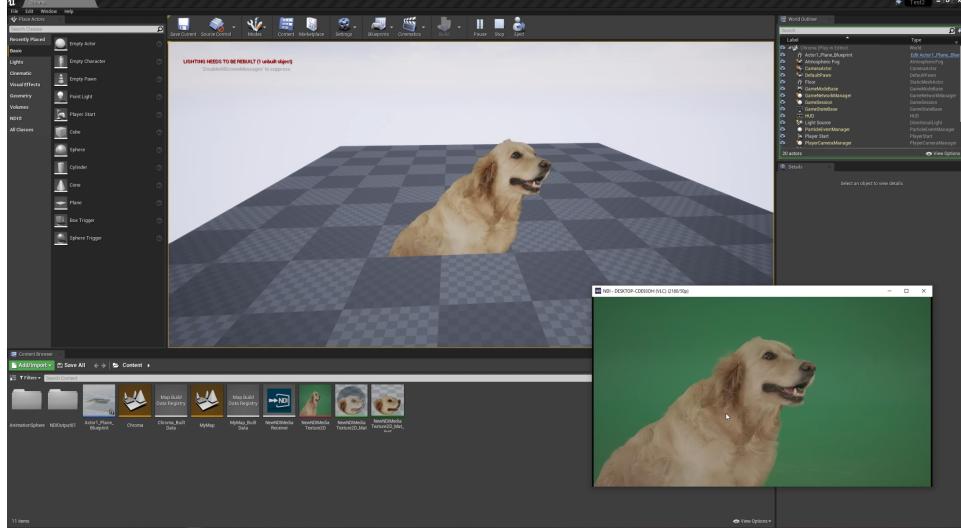


Figure 21: Resultat NDI Input

Els resultats d'aquesta prova han sigut molt bons, la latència del NDI era molt baixa, a més d'una qualitat sorprendent. Ha donat alguns problemes d'estabilitat en algun moment, però s'ha de tenir en compte que el plugin d'Unreal encara és molt nou i és normal que doni algun problema que altre, encara i això, són resultats prou estables i té potencial per poder ser utilitzat com a eina de televisió professional.

3.1.3 Unity

Per la prova del Unity, s'ha creat un projecte que contingui diversos elements. Un vídeo de fons, un element 3D amb un vídeo com a texture, un logo 3D i un PNG amb transparència. Per a la reproducció de vídeos s'ha escollit el plugin UMP (Universal Media Player) [40], considerat el reproductor més complert de tota l'Asset Store (plataforma de descàrrega de plugins).



Figure 22: Funcionalitats UMP

A més, és l'únic compatible amb Linux, el sistema operatiu més utilitzat en sistemes al núvol [41]. Pel UMP, haurem de posar un element que controla la reproductor, i definir quines són les malles que han de renderitzar el vídeo. Per aquesta prova es té un controlador que reproduirà el vídeo del fons, i un altre que reproduirà el vídeo de la càpsula, l'altre element 3D. A més, tindrem una animació pel logo 3D que anirà girant, i un altre per la càpsula que s'anirà movent per la pantalla.

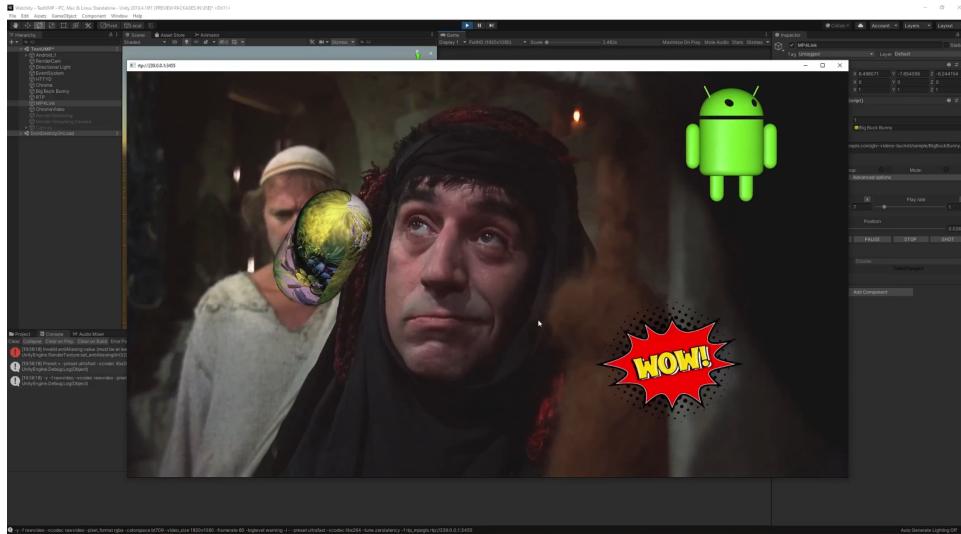


Figure 23: Prova Unity

Els resultats amb Unity han sigut molt satisfactoris, no ha donat cap problema i la reproducció ha sigut la més fluida. El plugin UMP suporta pràcticament tot tipus de formats de vídeos ja que utilitza les llibreries de libVLC. El codi és mitjanament obert pel que es podrà editar fàcilment. La imatge en PNG té la transparència perfectament aplicada, pel que veiem que el Unity suporta els valors d'alpha d'aquest.

Ara que ja s'han fet proves suficients amb totes les eines, es pot decidir sense cap tipus de dubte, que la millor opció per aquest projecte és **Unity**.

3.2 Taula Comparativa Eines Gràfiques

Està clar que cada eina té les seves avantatges i desavantatges, tampoc hi ha una que sigui millor que l'altre, sinó que depèn del projecte que es vulgui fer les solucions seran molt diferents. A continuació es resumeixen els avantatges i desavantatges esmentats anteriorment, partint del punt de vista per aquest projecte.

	Chrome Headless	Unreal Engine 4	Unity
Pros	Simplicitat, alta compatibilitat, no requereix utilitzar plugins	És professional, millor qualitat d'il·luminació	Molta documentació, molts plugins open-source, optimitzat per escenes senzilles, alta compatibilitat
Contres	Pot quedar limitat en quant a potència; efectes de vídeo	Relativament poca documentació, no tants plugins i comunitat com la competència, poca compatibilitat amb Linux	Menys potent que Unreal, eines externes no professionals creades per la comunitat

Table 2: Comparativa subjectiva de les possibles eines

3.3 Elecció de les llibreries a utilitzar

3.3.1 Entrada de fluxos

Decidit Unity com a eina a utilitzar, el següent pas és escollir les llibreries que permetran l'entrada de fluxos. Anteriorment s'han comentat les dues més importants, que són FFmpeg i libVLC. A les proves del Unity s'ha utilitzat un plugin anomenat UMP, el qual ha donat molts bons resultats. S'ha decidit utilitzar-lo ja que fa ús de libVLC, i ens ofereix una gran compatibilitat, a diferència d'altres com pot ser AVPro [42], molt més potent, però amb la desavantatge d'utilitzar llibreries pròpies i no ser compatible amb Linux.

Encara i apuntar tot a UMP, s'ha fet una prova comparativa dels dos plugins, ambdós executant-se en el mateix sistema operatiu (Windows 10) i mateixes condicions, un vídeo molt pesat en resolució 8K.

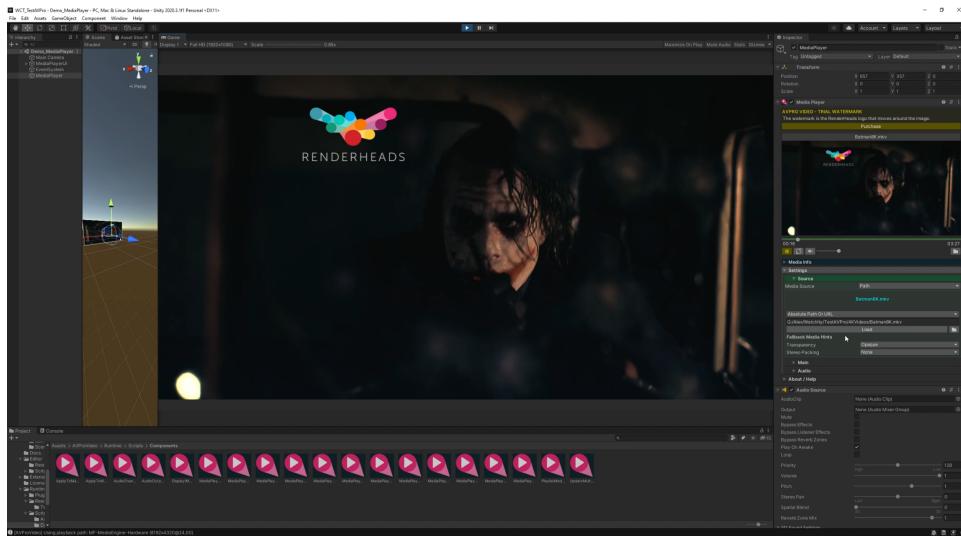


Figure 24: Prova AVPro (Video 8K)

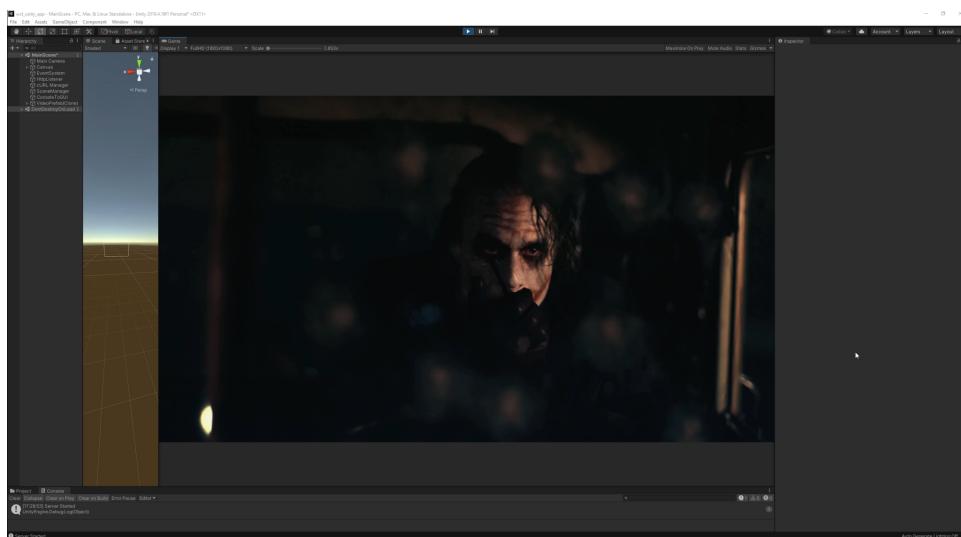


Figure 25: Prova UMP (Video 8K)

Les conclusions són que el AVPro és molt més potent per vídeos en 4K o més, però en vídeos 1080p o menys, el rendiment és molt igualat, i si tenim en compte que UMP accepta molts més formats, tenim un clar guanyador pels objectius d'aquest projecte on resolucions de 4K o més, no

són la prioritat.

3.3.2 Sortida del frame

Per altre banda, la sortida del frame és més complicada, no és comú voler treure per streaming el frame de Unity. La opció més viable sembla utilitzar FFmpeg amb configuracions de piping. És a dir, hauríem de generar les imatges a dins de Unity a una velocitat determinada per aconseguir els FPS (Frames per second) desitjats. I després introduir-ho a FFmpeg amb una comanda utilitzant «pipe» com a input.

Hi ha un projecte de Github [43], desenvolupat per l'usuari Keijiro, molt conegut a la comunitat de Unity per crear una gran varietat de plugins molt útils i innovadors. El que aconsegueix és introduir FFmpeg a Unity, i utilitzar el mètode dels pipeline per gravar un vídeo local. Afortunadament el codi és open-source, pel que el podem modificar per aconseguir que enlloc de generar un fitxer en local, faci un streaming per rtp. Aconseguit això, ens trobem amb diversos problemes. Amb Linux només suporta la API de render «Vulkan» [44]. És una API que està agafant molta força, però és millor evitar-la ja que és encara molt nova, i el plugin de reproducció de vídeo que s'utilitza només suporta OpenGL. A part d'això i més important, l'àudio no surt de cap manera, i s'ha de capturar de manera externa, pel que serà molt complicat sincronitzar-ho amb el vídeo.

Un altre opció és automatitzar unes comandes de FFmpeg per capturar la finestra i generar un streaming de sortida. Com ja ha sigut comentat abans, FFmpeg és l'eina més potent de vídeo i es controla mitjançant comandes al terminal, el que és molt útil per a poder automatitzar les tasques a través de scripts.

3.4 Elecció del hardware i sistema

3.4.1 Plataforma Virtual AWS

L'elecció de la plataforma que contindrà el software no ha sigut gaire complicada. Les opcions eren Azure, AWS (Amazon Web Services) [45] o Google. Es tracten de plataformes Cloud on es poden emmagatzemar i executar tot

tipus d'arxius. Són molt utilitzades ja que només les mega corporacions disposen d'infraestructures de servidors propis, la majoria de desenvolupadors lloguen els servidors a una d'aquestes plataformes. Comparativa entre AWS i Azure [46].

AWS	Azure
Plataforma de computació al núvol sota demanda (Amazon)	Plataforma pública al núvol (Microsoft)
Amigable des de els seus inicis amb el model de codi obert	No tenen bona relació amb la comunitat de codi obert
Té una gran avantatge en quant a ofertes de núvol governamental	Limitat en quant a ofertes per al núvol governamental
Models de preu flexibles	Models de preus més tancats i menys varietat
Segueix donant recolzament per suportar els models de núvols híbrids	Excel·lent per models amb núvols híbrids
Gran varietat de software i sistemes operatius tant Linux com Windows	Està més limitat al sistema operatiu de Windows encara que també té algunes opcions de Linux
El sistema d'emmagatzematge de AWS (EBS) és molt ràpid per sistemes amb big data.	L'emmagatzematge estàndard té problemes amb el big data i per tant s'ha de contractar un server de primera qualitat.
Entorns molt més madurs pel big data	Entorns més verds per big data, encara que Azure està millorant en aquest aspecte
Es pot accedir a les màquines de manera individual	Les màquines estan agrupades en un sol servei al núvol i responen al mateix domini però amb diferents ports
Elastic Compute Cloud (EC2) Es paga per hora utilitzada	Es paguen per minut
S3 Arxius de recuperació a curt termini. Per llargs terminis es pot utilitzar Glacier	Té una opció semblant a S3, però per recuperacions a llarg termini encara no disposa de cap solució
La seguretat es proporciona a través de rols definits per l'usuari administrador	Es defineix un director que serà l'administrador del servidor

Table 3: Comparativa subjectiva de les possibles eines

AWS és la millor opció per hostejar aquest projecte, a més concorda amb l'elecció de l'empresa on s'està fent aquest projecte, Watchity, que tenen tot el seu software i servidors contractats amb aquesta plataforma.

3.4.2 Sistema operatiu

Per escollir el sistema operatiu més adient haurem de veure les avantatges que té cada un dels més populars, que són: Windows, MacOS i Linux. Tenint en compte que el projecte s'emmagatzemarà i s'executarà des de un servidor de AWS, el més adient és tenir-ho amb el sistema operatiu més popular al núvol, Linux. Windows també disposa d'opcions per ser executat en un servidor, però les funcionalitats son infinitament inferiors, a més de ser privat i no permetre la modificació de cap element del sistema.

Per tant, sabent que el sistema operatiu serà Linux, també caldrà escollir la distribució més adient. Les millors opcions són Ubuntu Server, CentOS, SUSE i Arch. Cada un d'ells tenen les seves avantatges i desavantatges. A Watchity, treballen amb CentOS en totes les seves màquines, ja que es una distribució molt sòlida i de les més utilitzades per servidor. La diferència més gran entre aquests dos és que Ubuntu al estar basat en Debian, els paquets són instal·lats en .deb amb el gestor de paquets «apt-get», mentre que a CentOS s'utilitza «yum» o «dnf».

Realment per aquest projecte és indiferent l'elecció d'un o un altre, pel que s'escollirà CentOS simplement per no diferenciar-se a la metodologia de treball de Watchity.



Figure 26: CentOS Logo

4 Part Pràctica

Les eleccions ja han estat preses i és el moment de començar amb el desenvolupament de l'aplicació. Finalment s'ha decidit utilitzar Unity, amb l'eina UMP per a l'entrada de fluxos de vídeo, FFmpeg per la sortida del frame, AWS que serà el servidor que emmagatzemarà i executarà l'aplicació, i com a sistema operatiu s'ha decantat per la distribució de Linux Centos 8.

4.1 Arquitectura del sistema

L'aplicació es compondrà d'un binari executable de Unity, emmagatzemat a un servidor d'AWS. En aquest servidor es rebran crides cURL a través d'un frontend comprensible per l'usuari a qui va dirigit. Les peticions dels usuaris seran processades i «traduïdes» per fer els canvis a l'aplicació de Unity d'una manera que entengui. El servidor estarà emetent constantment la sortida del Unity amb un monitor virtual, que serà enviada per multi cast a la web on l'usuari està interactuant per poder veure els canvis que ha realitzat.

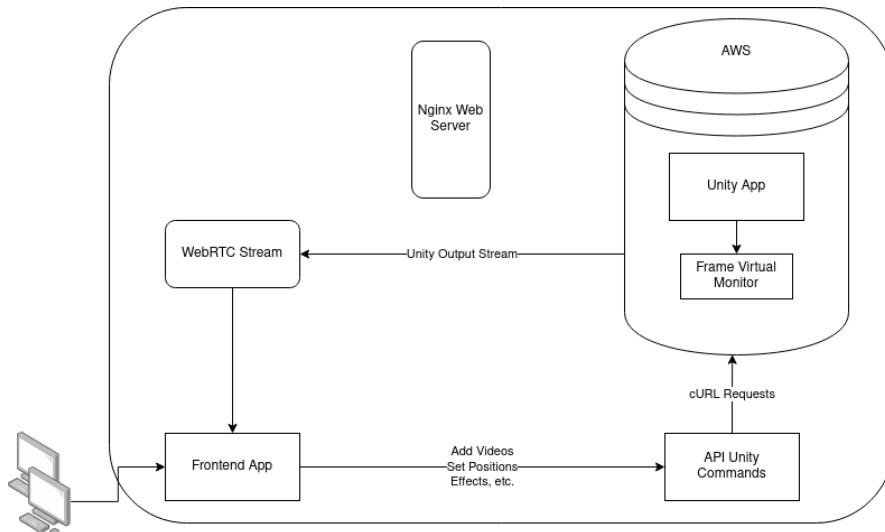


Figure 27: Arquitectura del Projecte

4.2 Possibilitats

4.2.1 Resolució

La resolució de l'aplicació serà per defecte 1920x1080. Això es defineix a la configuració de la targeta gràfica del servidor d'AWS, concretament a l'arxiu /etc/X11/xorg.conf. Aquest, és un arxiu de configuració que utilitza Nvidia i que emmagatzema tot tipus de configuració gràfica, ja sigui resolucions, tipus de targeta, monitors virtuals, refresc, etc. Al tractar-se d'un servidor, les configuracions gràfiques s'han de realitzar manualment, i crear un xorg.conf que s'adeqüi a les necessitats de cada un. X11 és el sistema de finestres més utilitzat de Linux. S'utilitza per donar interície gràfica als sistemes Unix. Encara que no és necessari per a servidors i per tant no venen instal·lats de sèrie, en aquest cas si que es necessitarà, ja que l'aplicació s'ha de poder executar en una finestra, encara que aquesta sigui virtual.

A continuació s'adjunta el fitxer xorg que s'ha realitzat per aquest projecte.

```
Section "Monitor"
    Identifier      "Monitor0"
    Option          "Primary" "true"
EndSection

Section "Monitor"
    Identifier      "Monitor1"
    Option          "LeftOf"   "Monitor0"
EndSection

Section "Device"
    Identifier      "Device0"
    Driver          "nvidia"
    VendorName     "NVIDIA Corporation"
    BoardName       "Tesla T4"
    BusID          "PCI:0:30:0"
    Option "NoFlip" "false"
EndSection

Section "Screen"
    Identifier      "Screen0"
    Device          "Device0"
    Monitor         "Monitor0"
    DefaultDepth    24
    Option          "AllowEmptyInitialConfiguration" "True"
    SubSection      "Display"
        Virtual      1920 1080
        Modes         "1920x1080"
        ViewPort     0 0
        Depth         24
    EndSubSection

```

```

    Option      "MetaModes" "1920x1080_60 +0+0 {ForceFullCompositionPipeline=
        On}"
    Option      "AllowIndirectGLXProtocol" "off"
    Option      "TripleBuffer" "on"
EndSection

```

Listing 12: Configuració Gràfica | xorg.conf

Començant amb la secció de «Monitor», serveix per definir un nou monitor, pel primer cas fem un primari amb el nom Monitor0, i en el segon cas fem un monitor secundari amb el nom Monitor1, situat a l'esquerra del primari. Aquest segon monitor es crea per si de cas en un futur es volgués tenir més d'una aplicació de Unity executant-se a la vegada, encara que la idea es tenir-ne només una en cada màquina AWS de forma ordinària.

La següent secció és el «Device», tracta de la targeta gràfica que es té instal·lada, amb tota la informació necessària, molt important posar el model (Tesla T4), targetes molt potents que a disposició de AWS, i el BusID, en aquest cas el PCI:0:30:0. Si no es posen aquestes dades, podria no funcionar la connexió gràfica amb l'ordinador.

A la secció «Screen» es defineix la configuració de pantalla. Molt diferent a la configuració de monitor. En aquest bloc es posarà un nom identificador «Screen0», el dispositiu gràfic que s'utilitza «Device0» (referència a la GPU Nvidia Tesla T4), el monitor que renderitza aquesta pantalla «Monitor0» la profunditat de color, que són 24. Ja que utilitzarem el estàndard, corresponent al True Color 24-bit, que utilitza 8-bits per cada color primari R, G i B. Cada cop s'està utilitzant més el Deep Color que utilitza 10-bits per canal, més conegut com a HDR (també existeix de 12-bits o més) [47] en els televisors més actuals, si en volguéssim fer ús i poder utilitzar contingut que disposi d'aquesta tecnologia hauríem de posar un 30 en el valor de DefaultDepth. Però ara per ara, no interessa utilitzar HDR i no és un objectiu proposat que l'aplicació ho permeti.

Seguit d'això es crea una subsecció que contindrà unes dades fixades de com ha de ser la pantalla, amb una resolució de 1920x1080 i 24 de profunditat.

Per últim, i de manera opcional, es poden definir unes opcions de comportament de la pantalla, el que s'ha utilitzat són les opcions de ForceFullCompositionPipeline i TripleBuffer per evitar bugs gràfics en els que la pantalla pot quedar tallada verticalment (un bug bastant comú en sistemes Linux amb gràfiques d'Nvidia).



Figure 28: Exemple Bug Gràfic VSync

Vist això, queda provat que es pot definir qualsevol resolució al sistema, només caldrà canviar l'arxiu de configuració xorg.conf i posar la que es desitgi. Es podria posar una resolució de 4K o 720p, això si, lògicament com més alta sigui la resolució, més latència es tindrà a la sortida.

4.2.2 Compatibilitat fluxes d'entrada

Per l'entrada de vídeos s'utilitza el plugin UMP de l'asset store de Unity, que a la vegada aquest utilitza internament les llibreries libVLC, pel que es permetrà qualsevol format que VLC accepti de manera nativa. Segons la pròpia pàgina de Videolan, creadors de libVLC els formats acceptats són:

Vídeo Còdecs

- MPEG-1/2
- DivX® (1/2/3/4/5/6)
- MPEG-4 ASP
- XviD
- 3ivX D4
- H.261

- H.263 / H.263i, H.264 / MPEG-4 AVC
- Cinepak
- Theora
- Dirac / VC-2
- MJPEG (A/B)
- WMV 1/2
- WMV 3 / WMV-9 / VC-1
- Sorenson 1/3
- DV
- On2 VP3/VP5/VP6
- Indeo Video v3 (IV32)
- Real Video (1/2/3/4)

Àudio

- MPEG Layer 1/2
- MP3 - MPEG Layer 3
- AAC - MPEG-4 part3
- Vorbis
- AC3 - A/52
- E-AC-3
- MLP / TrueHD>3
- DTS
- WMA 1/2
- WMA 3

- FLAC
- ALAC
- Speex
- Musepack / MPC
- ATRAC 3
- Wavpack
- Mod
- TrueAudio
- APE
- Real Audio
- Alaw/μlaw
- AMR (3GPP)
- MIDI
- LPCM
- ADPCM
- QCELP
- DV Audio
- QDM2/QDMC
- MACE

Subtítols

- DVD
- Text files (MicroDVD, SubRIP, SubViewer, SSA1-5, SAMI, VPlayer)
- Closed captions

- Vobsub
- Universal Subtitle Format (USF)
- SVCD / CVD
- DVB
- OGM
- CMML
- Kate

Fluxos d'entrada

- UDP/RTP Unicast
- UDP/RTP Multicast
- HTTP / FTP
- MMS
- TCP/RTP Unicast
- DCCP/RTP Unicast
- Arxius Locals
- DVD Video
- Video CD / VCD
- SVCD
- Audio CD (no DTS-CD)
- DVB (Satellite, Digital TV, Cable TV)
- MPEG encoder
- Video acquisition

Formats d'entrada

- MPEG (ES,PS,TS,PVA,MP3)
- AVI
- ASF / WMV / WMA
- MP4 / MOV / 3GP
- OGG / OGM / Annodex
- Matroska (MKV)
- Real
- WAV (including DTS)
- Raw Audio: DTS
- AAC
- AC3/A52
- Raw DV
- FLAC
- FLV (Flash)
- MXF
- Nut
- Standard MIDI / SMF
- CreativeTM Voice

En resum, tots els formats estandarditzats son perfectament compatibles amb l'aplicació i hauria de ser bastant complicat trobar-se amb un vídeo incompatible per la reproducció.

4.2.3 Control de l'app

El control de l'aplicació es farà mitjançant comandes cURL, que contindran text pla amb les instruccions que haurà de fer el Unity, per exemple «addVideo», «destroyBackground», etc. La metodologia per utilitzar cada funcionalitat serà semblant entre elles per aconseguir un ús fluït i senzill d'entendre. Els elements estaran separats en Vídeos, Imatges, Textos i Background. Exceptuant el background, la resta seran un conjunt d'arrays ordenats per números, tindrem el vídeo número 1, número 3, i no caldrà que tinguin un ordre seqüencial, a més de crear arrays dinàmics per permetre una longitud infinita.

La metodologia de la comanda a seguir serà:

```
function=elementNumber%attributes~optionalSettings
```

Exemple:

```
addVideo=1%home/user/videoTest.mp4~1
```

Seguint d'esquerra a dreta l'exemple que s'ha posat tenim a la banda esquerra de l'igual, la funció que haurà de fer, en aquest cas addVideo, és a dir, afegir un nou vídeo a l'escena actual. Després de l'igual tenim tots els atributs i configuració, en aquest cas posem el vídeo a la posició 1 (no posició espacial, sinó de l'array), utilitzem el separador % per introduir l'atribut principal que pràcticament totes les funcionalitats tindran, per aquest exemple serà la ruta on es troba el vídeo que es vol reproduir.

4.2.4 Compatibilitat audio

Per altre banda tenim el sistema d'àudio, en el que un usuari pot posar una cançó, veu en off, o el que vulgui de fons. Aquí no s'utilitza libVLC, sinó la llibreria interna d'àudio de la que disposa Unity, pel que serà compatible únicament amb àudio WAV i MPEG, que ha de ser més que suficient tenint en compte que no és un objectiu principal ni molt demandat, i es compleix un suport dels estàndards d'àudio. Això sí, s'accepten fitxers tant locals com al núvol, descarregant la informació d'aquests, i emmagatzemant-la en un array d'AudioClips.

```

//Classe de Unity util per descarregar arxius
UnityWebRequest www = new UnityWebRequest();

//Nomes s'accepten arxius WAV o MPEG, en cas de ser un altre, donara error i no
//es descarregara
if(Path.GetExtension(info[1]).Contains("wav")){
    www = UnityWebRequestMultimedia.GetAudioClip(info[1],AudioType.WAV);
}
else{
    www = UnityWebRequestMultimedia.GetAudioClip(info[1],AudioType.MPEG);
}

//Esperem a que es descarregui el fitxer d'audio
yield return www.SendWebRequest();

//Creem un nou audio al array
audiosArray.Add(nAudio, newAudio);

//Guardem el contigut de la descarrega al Clip
audiosArray[nAudio].clip = DownloadHandlerAudioClip.GetContent(www);

```

Listing 13: Àudio | addAudio.cs

4.2.5 Transicions

Es permeten 3 tipus de transicions en vídeos, per tall, crossfade i fade. En la de tall, passem d'un vídeo a un altre d'una manera brusca. El crossfade significa canviar d'un vídeo a un altre fent que un va desapareixent mentre el segon apareix, pel que no passem per un pla en negre i pot quedar més dinàmic segons el moment. En el fade en canvi, el primer vídeo es va difuminant fins a passar a un pla en negre, i un cop s'ha arribat al negre, el segon apareix de manera gradual.

Per fer ús d'aquestes funcionalitats disposarem de 2 comandes:

`dissolve=videoNumber%link_localPath~transitionTime`

Aquesta crida servirà per fer un canvi d'un vídeo a un altre, amb un efecte de fade. El atribut transitionTime que va després del símbol ~ és opcional, i per defecte el valor serà 0. Aquest valor indica els segons que trigarà en desaparèixer un vídeo, és a dir que la transició completa serà el doble del valor que s'introduceixi.

```
cross=videoNumber%link_localPath~transitionTime
```

En canvi si posem el mode «cross», es farà un efecte de crossfade amb una transició dinàmica. El valor de transitionTime funciona exactament igual que l'anterior, per tant, el temps total de la transició també serà el doble del valor introduït.

El tercer tipus de transició que s'havia comentat era el de tall, com és bastant innecessari crear una nova funció per aquesta transició tan senzilla, el que s'ha de fer és utilitzar qualsevol de les dues però amb un valor de transitionTime de 0, que es pot declarar explícitament o com ja s'ha comentat abans, no posar-lo i l'agafarà igualment com a valor per defecte.

Addicionalment, totes les transicions incorporen un script per tal de que l'àudio baixi de manera gradual conforme desapareguin els vídeos i no talli de manera brusca.

4.2.6 Animacions

A la majoria de comandes visuals de l'aplicació (vídeos, imatges, textos...) es permet un paràmetre opcional que definirà el que trigarà en arribar a l'objectiu demanat. Per exemple si volem moure i escalar un vídeo, podem posar al final de la comanda el símbol ~ i indicar un valor en segons de quant trigarà en fer l'animació. Aquests efectes tenen una animació anomenada EaseOutExpo, el que significa que segueixen una corba exponencial en el transcurs de l'animació. Per defecte seria lineal, però quedaria molt poc dinàmic i poc professional. Pel que s'ha optat per aquesta alternativa ja que es considera la més adequada pel tipus de clients i és molt atractiu. Tot això s'ha aconseguit gràcies a l'ajuda de un plugin molt famós de Unity anomenat LeanTween [48], desenvolupat per l'usuari «Dented Pixel».



Figure 29: Logo Dented Pixel

Es un «tweening engine» que permet fer càlculs intermedis per passar d'un valor a un altre, amb corbes d'animació totalment customitzables. Un exemple d'ús que es mostra a la seva documentació.

```
LeanTween.moveToObject(gameObject, 1f, 1f).setEase(LeanTweenType.easeInQuad).setDelay(1f);
```

4.2.7 Transformacions

Posicions i escala

És molt probable que el modificador les posicions i escales dels vídeos serà de les funcionalitats més utilitzades i és un requisit indispensable. Aquest bloc es divideix en 3 comandes diferents:

Position

Es defineix un valor X i un valor Y compresos entre 0 i 1. Entenent com a 0,0 el punt superior esquerra de la pantalla. Es fa una conversió interna per passar dels valors de tant per 1, a un valor que representi la posició dins del motor de Unity.

```
string[] coordinates = splitInfo[1].Split(',');
float[] coordNums = new float[2];

//Parse coordinates float
for(int i = 0; i < coordinates.Length; i++)
{
    coordNums[i] = float.Parse(coordinates[i], CultureInfo.InvariantCulture.
        NumberFormat);
}

int videoN = int.Parse(splitInfo[0], CultureInfo.InvariantCulture.NumberFormat);

//Define position values
```

```

GameObject videoTran = umpPlayers[videoN].transform.parent.gameObject;
Vector3 initPos = videoTran.transform.position;
Vector3 desiredPos = new Vector3(map(coordNums[0], 0, 1, -_CAMERALIMITS.x,
    _CAMERALIMITS.x), map(coordNums[1], 0, 1, _CAMERALIMITS.y, -_CAMERALIMITS.y),
    videoTran.transform.position.z);

//Video LookAt Camera
videoTran.transform.rotation = Quaternion.Euler(90, 180, 0);

//Move Animation
LeanTween.value(gameObject, initPos, desiredPos, tweenTime).setOnUpdate((Vector3
    val) =>
{
    videoTran.transform.position = val;
}).setEaseOutExpo();

```

Listing 14: Codi Posició | positionVideo.cs

Per tant; la seva utilització és aquesta:

```

position=n_video%posx,posy #Positions (0, 1)

Optional:
~setTransitionTime #Default 0, set Transition Time in seconds

Ex:
position=0%0.5,0.5
position=0%0.5,0.5~5 #5 Second to get the desired parameters
curl —data "position=1%0.5,0.8~2" http://localhost:4444/
curl —data "position=1%0.5,0.5~2" http://localhost:4444/

```

Listing 15: Documentació Posició | positionReadme.md

Scale

Es defineix un sol valor en el rang de 0 a 1 que indicarà l'escala (mida) desitjada. Només té un valor ja que no es contempla tenir vídeos deformats o de resolucions estranyes que no siguin amb un aspect ratio de 16:9.

La programació d'aquest bloc és bastant més senzilla ja que la conversió es directe, el Unity utilitza la mateixa mesura, valors de 0 a 1. Pel que un valor de 0.5 serà el 50% de la mida original.

```

float finalScale = float.Parse(splitInfo[1], CultureInfo.InvariantCulture.
    NumberFormat);
int videoN = int.Parse(splitInfo[0], CultureInfo.InvariantCulture.NumberFormat);
GameObject videoTran = umpPlayers[videoN].transform.parent.gameObject;
Vector3 initScale = videoTran.transform.localScale;
Vector3 desiredScale = new Vector3(_LIMITSCALE.x * finalScale, _LIMITSCALE.y *
    finalScale, _LIMITSCALE.z * finalScale);
videoTran.transform.rotation = Quaternion.Euler(90, 180, 0);

//Scale
LeanTween.value(gameObject, initScale, desiredScale, tweenTime).setOnUpdate((
    Vector3 val) =>

```

```
{  
    videoTran.transform.localScale = val;  
}).setEaseOutExpo();
```

Listing 16: Codi Scale | scaleVideo.cs

La seva documentació corresponent és bastant semblant a la de posició.

```
scale=n_video%scale #Scale(0, 1)  
  
Optional:  
~setTransitionTime #Default 0, set Transition Time in seconds  
  
Ex:  
scale=0%0.4  
scale=0%0.4~5 #5 Second to get the desired parameters  
curl —data "scale=1%0.3~2" http://localhost:4444/  
curl —data "scale=1%1~2" http://localhost:4444/
```

Listing 17: Documentació Scale | scaleReadme.md

Poscale

Poscale és una comanda que fusiona les dues anteriors per poder fer les dues coses de cop, posicionar i escalar, el que permet poder sincronitzar més fàcilment l'animació de totes dues, a més d'un ús més còmode.

```
poscale=n_video%posx,posy,scale #Positions (0, 1) Scale(0, 1)  
  
Optional:  
~setTransitionTime #Default 0, set Transition Time in seconds  
  
Ex:  
poscale=0%0.5,0.5,0.4  
poscale=0%0.5,0.5,0.4~5 #5 Second to get the desired parameters  
curl —data "poscale=1%0.5,0.8,0.3~2" http://localhost:4444/  
curl —data "poscale=1%0.5,0.5,1~2" http://localhost:4444/
```

Listing 18: Documentació Poscale | poscaleReadme.md

Colors i transparències

També es permet modificar la transparència del vídeo, per a fer-ho s'ha de modificar l'alpha del material del Unity. Aprofitant que es programa això, també es permetrà canviar el color del material en cas de desitjar-ho.

Colors

Aquesta funció tenyirà d'un color el vídeo, per exemple si li indiquem un color vermell pur, es dibuixarà tot en escala de vermells. Es té en compte que potser no serà una funcionalitat molt utilitzada, però per algun cas específic pot ser bastant útil.

```
color=n_video%r,g,b #Colors (0, 1)

Ex:
color=0%1,1,1 #White
color=0%0,0,0 #Black
curl —data "color=1%1,0,0" http://localhost:4444/
```

Listing 19: Documentació Colors | colorsReadme.md

Transparències

Es permet canviar l'alpha dels vídeos per poder tenir-ne d'alguns superposats sobre altres. Això és molt útil i era un rels requisits indispensables, sobretot en imatges, del que es parlarà més endavant. Per aquesta funció també es permet crear una animació, que dóna un aspecte molt dinàmic i professional.



Figure 30: Mostra de transparència

Per utilitzar-ho només cal seguir la següent nomenclatura:

```
opacity=n_video%a #Alpha (0, 1)

Optional:
~setAnimationTime #Default 0, set Transition Time in seconds

Ex:
opacity=0%0      #Transparent
opacity=0%1      #Opaque
curl —data "opacity=1%0.3" http://localhost:4444/
curl —data "opacity=0%1~2" http://localhost:4444/
```

Listing 20: Documentació Transparències | alphaReadme.md

Encara que aquest efecte es pot aconseguir simplement modificant el valor alpha del material principal, pels objectes de vídeo, s'utilitza un shader propi anomenat com a Smart Material, que controlarà tot tipus d'efectes que se li demanin. És per això que només calen un parell de línies per aplicar-lo.

```
Renderer render = umpPlayers[nVideo].GetComponentInParent<Renderer>();  
//Apliquem un valor a l'atribut _Opacity del Shader Smart Material  
render.material.SetFloat("_Opacity", val);
```

Listing 21: Codi Alpha | alphaVideo.cs

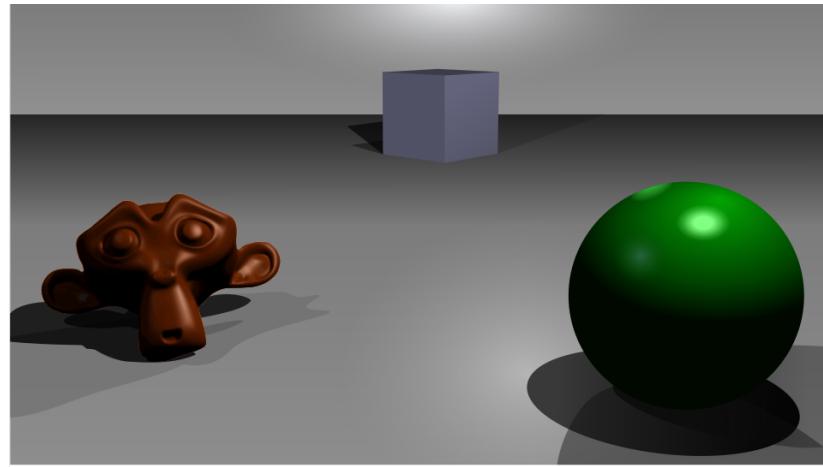
4.2.8 Layer Sorting

Layer Sorting o si traduïm literalment, ordenació de capes, és la funcionalitat que ens permet controlar les posicions en el eix de la Z, és a dir, el vector normal del pla de la càmera. Per aconseguir-ho, no s'ha fet només utilitzant la posició espacial, sinó que també s'ha fet ús de la tècnica de modificar el Z Buffer. En gràfics per ordinador, el z-buffering és el que s'encarrega de gestionar quins son els objectes que es renderitzen davant de la resta. La targeta gràfica fa un càlcul segons les posicions dels objectes i genera la imatge generada.

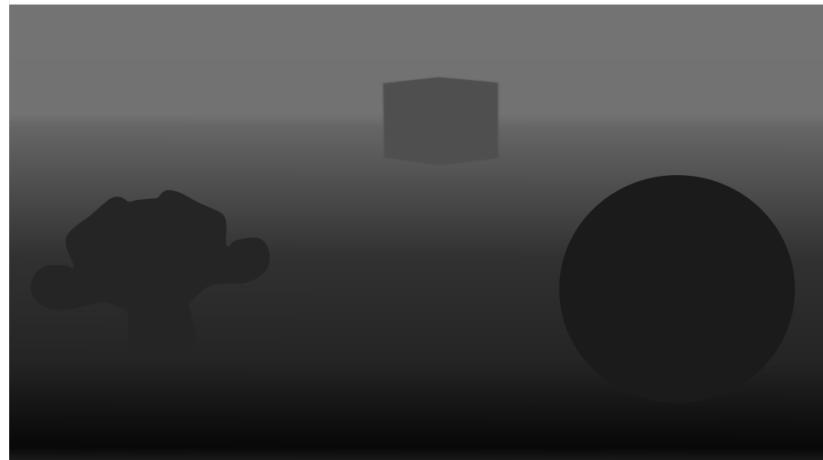


Figure 31: Representació Z-Buffer

En aquesta figura podem veure una representació d'aquest efecte, com més blanc sigui l'objecte, més prioritat tindrà a renderitzar-se, i per tant, més a sobre estarà de la resta d'objectes.



A simple three-dimensional scene



Z-buffer representation

Figure 32: Comparació Render i Z-Buffer

Entenen això doncs, i sabent que Unity proporciona un atribut del material anomenat «Render Queue»[49], podrem modificar al nostre gust el Z-Buffer i renderitzar un objecte que estigui molt lluny de la càmera abans que un que estigui més a prop, per exemple.

Per a utilitzar-ho, s'ha definit un sistema de capes, que va de 0 a infinit, entenent que com més gran sigui el número, menys prioritat tindrà a renderitzar-se.

```
layerSort=n_video%Z_coord #Range (0–Inf) Steps 1 by 1 (Int)

Ex:
layerSort=0%0
curl —data "layerSort=0%0" http://localhost:4444/
curl —data "layerSort=1%2" http://localhost:4444/
curl —data "layerSort=0%1" http://localhost:4444/
curl —data "layerSort=1%0" http://localhost:4444/
```

Listing 22: Documentació LayerSort | layerSortReadme.md

Només caldrà indicar quin es el vídeo que es vol canviar de posició, i la capa destí.

La programació al Unity és molt intuïtiva, només caldrà agafar el material del render, i a la propietat renderQueue, assignar-li un valor.

```
render.material.renderQueue = value;
```

4.2.9 Cropping

El efecte de cropping vol dir retallar un vídeo. La forma més simple al retallar seria en forma rectangular. En aquesta aplicació, a més de poder retallar amb forma rectangular, utilitzant la potència i possibilitats de Unity podem retallar amb qualsevol tipus de forma gràcies als shaders. La manera d'utilitzar-ho serà aplicant una màscara que es compondrà de colors blancs (visible) i negres (no visible). Un color gris voldria dir que es veu lleugerament, pel que també ho podríem utilitzar per fer degradats d'opacitat.

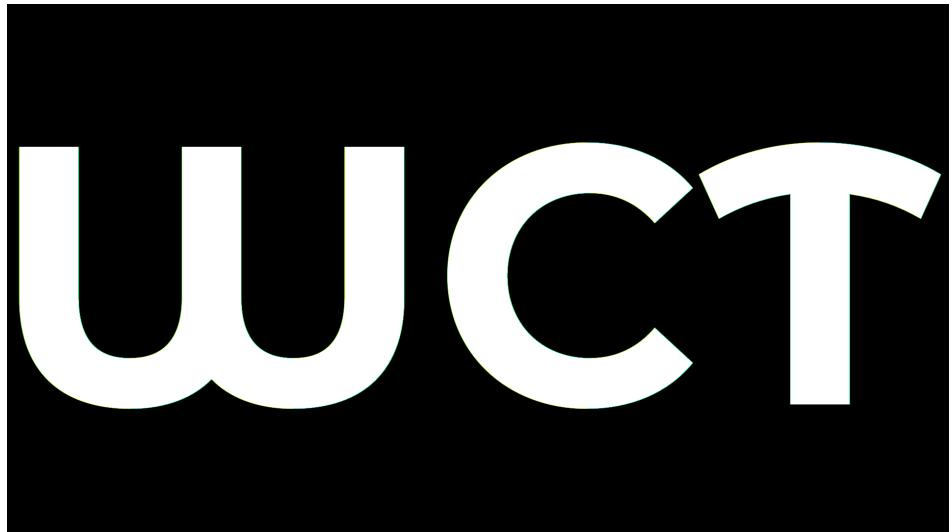


Figure 33: Exemple de màscara de crop

En aquesta figura tenim un exemple de màscara de cropping, aplicarem aquesta imatge al shader que s'ha creat anteriorment i farà l'efecte de crop. Per la mida i posició del crop, utilitzarem les propietats de Tiling i Offset del shader per modificar les UV de la imatge.

Doncs si posem un vídeo qualsevol a sobre d'un altre, i li apliquem aquest efecte amb la mateixa màscara mostrada abans, aconseguirem aquest resultat.

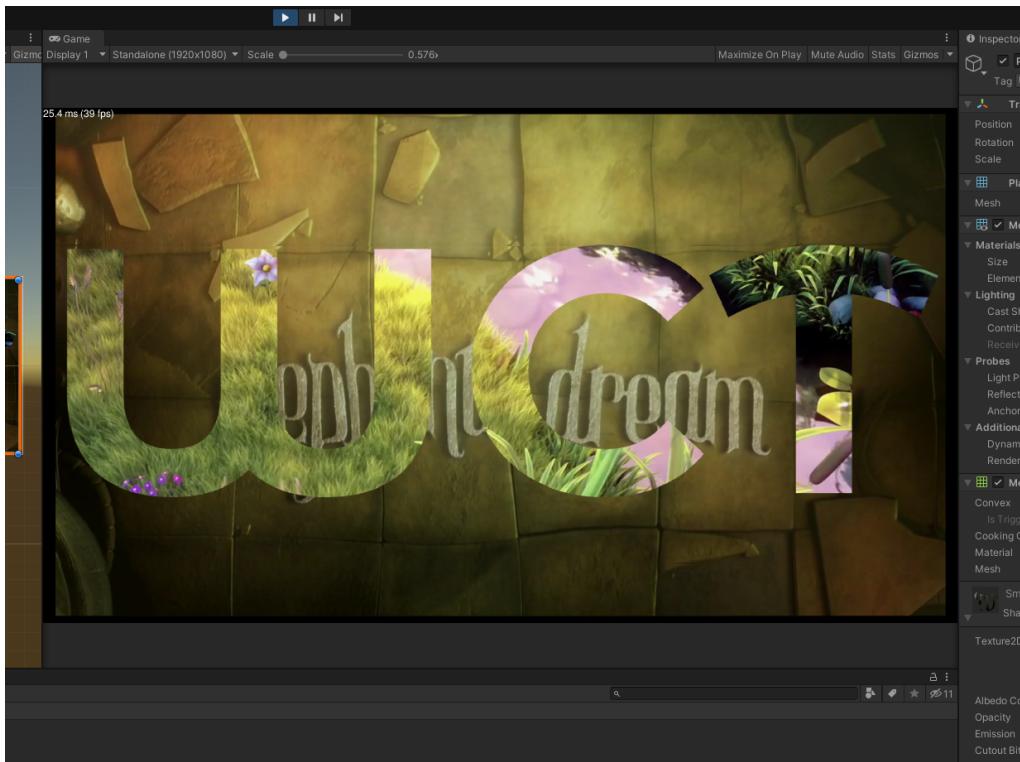


Figure 34: Resultat de Crop amb màscara WCT

Per últim, comentar com s'ha fet la creació del shader. S'ha utilitzat l'eina de Unity ShaderGraph [50], que permet crear-ne sense necessitat de programar, el que ho fa bastant més senzill i intuïtiu. Utilitza un sistema de nodes, bastant inspirat en els blueprints de Unreal, però aquí resulta molt més útil, ja que la programació de shaders és baix nivell i no és gaire fàcil d'aprendre.

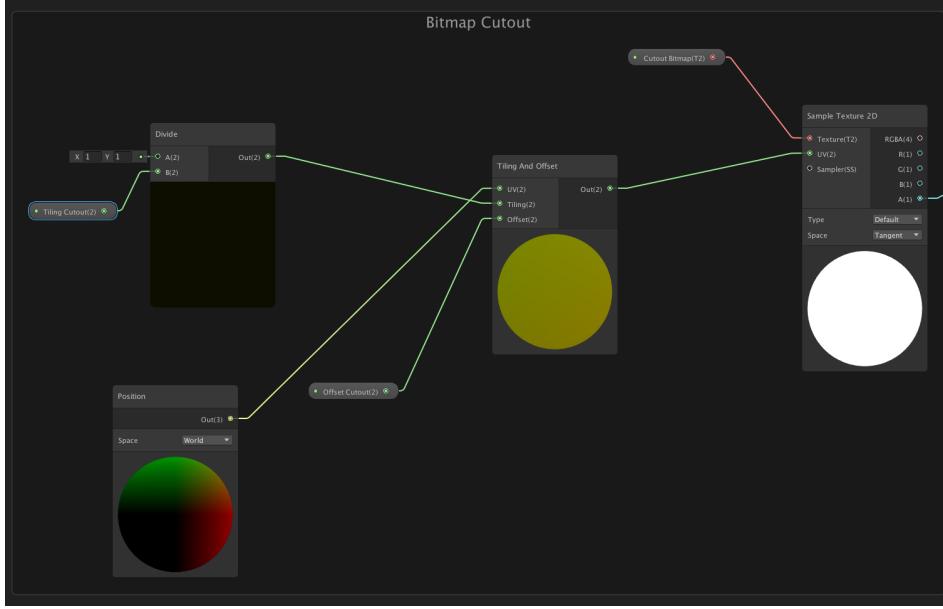


Figure 35: Crop Shader Graph

Així queden els nodes per aconseguir el resultat de un material d'un vídeo amb una màscara. Al centre tenim el node de «Tiling and Offset» que permetrà moure la màscara en el pla. A aquest node li entra a «UV», la sortida del node de posició, que indica respecte a què ho haurem de moure, en aquest cas, respecte la posició global. Entra també el resultat de Tiling, al que se li ha de fer un processament previ, ja que el valor real segueix una corba logarítmica, haurem de dividir 1 entre el valor real. I per últim també un Vector2 que serà el Offset. Tot això ho introduïm al valor de «UV» de la màscara i això creará el desplaçament.

Per últim, només caldrà multiplicar aquest resultat, amb l'Alpha de la textura original (Video), introduir-lo al material PBR resultant, i s'aconseguirà l'efecte.

La documentació corresponent per la seva utilització:

`cutout=n_video%name` #Name of Figure

Available Names:

- None #Remove Cutout Texture
- Rectangle
- RectangleBlur

- Rounded
- Random
- WCT
- Nike
- Gradient

Ex:
 cutout=0%WCT
 cutout=0%Rectangle
 cutout=0%Random
 curl —data "cutout=1%RectangleBlur" http://localhost:4444/
 curl —data "cutout=1%WCT" http://localhost:4444/
 curl —data "cutout=1%Rectangle" http://localhost:4444/

cutScale=n_video%x,y #X,Y Coordinates (0,1)

Ex:
 cutScale=0%0.5,0.5
 curl —data "cutScale=1%0.2,0.6" http://localhost:4444/
 curl —data "cutScale=0%1,1" http://localhost:4444/

cutMove=n_video%x,y #X,Y Coordinates (0,1)

Ex:
 cutMove=0%0.5,0.5 #Default Position
 curl —data "cutMove=1%0.6,0.4" http://localhost:4444/
 curl —data "cutMove=0%0.5,0.5" http://localhost:4444/

Listing 23: Documentació Cropping | cropReadme.md

4.2.10 Outline and Shadows

Un altre efecte que s'ha creat, és l'anomenat «Outline and Shadows». Permet posar un marc d'un color sòlid a un vídeo, o una ombra. En la creació d'aquest efecte, s'ha fet un altre shader des de 0. S'ha seguit la tècnica per fer l'efecte de outline en un sprite d'un videojoc, ja que el crop podrà tenir diferents formes. L'usuari «CodeMonkey», que es dedica a fer tutorials de Unity, en va fer un dedicat a aquest efecte [51], que ha sigut molt útil per crear la base del nostre.



Figure 36: Ombrejat en un vídeo amb Crop

Aquí es pot veure la distribució resultant del shader.

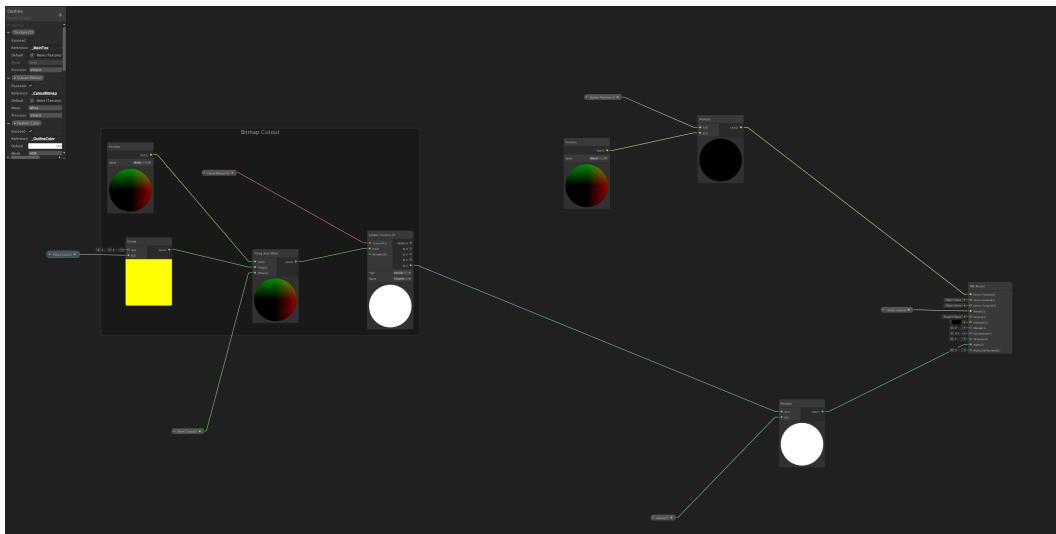


Figure 37: Outline Shader Graph

Per utilitzar-ho hi ha moltes comandes i configuracions ja que és un shader molt complert, però un cop s'entengui el seu ús, pot ser molt útil.

```
#You can pass "Rectangle" or "copy" that would copy the bitmap of parent video
Ex:
shaOut=0%Rectangle
shaOut=0%copy
curl —data "shaOut=1%Rectangle" http://localhost:4444/
curl —data "shaOut=2%copy" http://localhost:4444/
```

```
soColor=n_video%r,g,b #Colors (0, 1)
Ex:
soColor=0%1,1,1 #White
soColor=0%0,0,0 #Black
curl —data "soColor=1%0,1,0" http://localhost:4444/
curl —data "soColor=2%1,0,0" http://localhost:4444/
```

```
soOpacity=n_video%a      #Alpha (0, 1)
Ex:
soOpacity=0%0 #Transparent
soOpacity=0%1 #Opaque
curl —data "soOpacity=1%1" http://localhost:4444/
```

```
soThickness=n_video%thickness #Alpha (0, 5) 1 is default scale
Ex:
soThickness=0%1 #Default Scale
soThickness=0%0.8 #Smaller
soThickness=0%1.3 #Bigger
curl —data "soThickness=1%1.1" http://localhost:4444/
```

```
soMove=n_video%x,y #X,Y Coordinates (0,1)
Ex:
soMove=0%0.5,0.5 #Default Position
curl —data "soMove=0%0.54,0.54" http://localhost:4444/
curl —data "soMove=0%0.6,0.6" http://localhost:4444/
curl —data "soMove=0%0.4,0.4" http://localhost:4444/
```

Listing 24: Documentació Outline | outlineReadme.md

4.2.11 Chroma Key

El chroma és el millor efecte del que disposa l'aplicació, ja que no és gaire fàcil de fer a temps real depèn les eines que s'utilitzin, és per això que la competència no disposa d'aquest efecte. Era un repte molt important aconseguir-ho ja que marcarà la diferència. Per la seva realització, s'ha integrat dins el mateix

shader del Smart Material creat anteriorment, això sí, utilitzant Sub-Shaders. Amb això es poden crear nodes propis que realitzin qualsevol funció. Resultant en un node de «Chroma» amb moltes entrades, que com a sortida té el color resultant del frame del vídeo, i la màscara d'alpha, que es pot combinar amb un Crop sense cap problema.

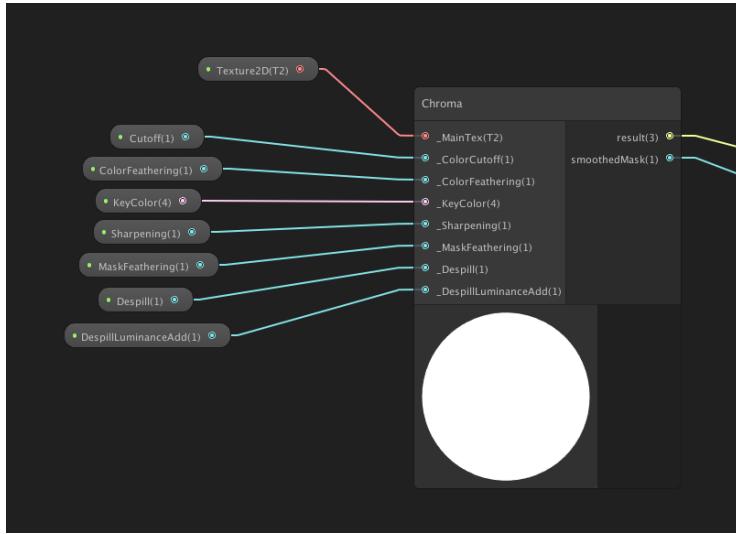


Figure 38: Chroma Shader Graph

Aquest node conté 5 Sub-Graph, és a dir, 5 petits shaders que fan processaments intermedis per aconseguir el resultat final. Processos com poden ser toleràncies, conversions de rgb a un altre format, càlcul de difuminat de contorns, eliminació de restes de color verd (Despill), etc.



Figure 39: Sub Shaders Chroma

Tot això resulta en uns quants atributs que poden ser modificats, que són:

- Key Color: Especifica el color a eliminar (per defecte verd pur $rgb(0,1,0)$).
- Threshold: El rang de color a eliminar partint del Key Color, per si hi ha ombres al color o la il·luminació no està del tot bé.
- Smoothness: Per aplicar un smooth al threshold i no apliqui una opacitat binària.
- Sharpen: El valor indica com d'afilat es vol el contorn.
- Mask Feather: Suavitzat de contorns.
- Despill: Defineix si es volen eliminar les restes de verd o no, i amb quina quantitat, 1 eliminarà tot el verd de la imatge, 0 ho deixarà igual.

El chroma resultant és molt vàlid, i més tenint en compte que s'està processant en temps real.



Figure 40: Chroma Test

4.2.12 Textos

Pels textos s'ha utilitzat el plugin de Unity TextMeshPro, que està inclòs per defecte a les últimes versions, ja que és molt superior al sistema de Textos

que tenia anteriorment. Per defecte apareixen amb una animació de opacitat de 0 a 1, de 0.2 segons. Els paràmetres que es poden modificar són: canviar el text, canviar el pivot (permets ajustar millor Títols amb Subtítols entre altres), canviar la posició, opacitat, i destruir-los. Tot el que té a veure amb el format del text, es farà amb el propi sistema de TextMeshPro, que permet aplicar atributs d'una manera molt semblant a com es faria amb HTML, per exemple, la font, mida, color...

Concretament els atributs o «tags» com diuen a la seva documentació [52] són:

Tags	Efecte
align	Alineació de textos.
alpha, color	Color i Opacitat
b, i	Estils de negreta i cursiva
cspace	Espaiat de textos
font	Font i selecció de materials
ident	Sagnia
line-height	Alçada de la línia
link	Assigna un ID
lowercase, uppercase, smallcaps	Capitalització automàtica
margin	Marges del text
mark	Marcar el text
mspace	Monoespacial
noparse	Evita parsejar els símbols antilambda
nobr	Elimina els espais entre línies
page	Salt de pàgina
pos	Posició horitzontal
size	Mida de la font
space	Espai horitzontal
sprite	Afegir sprites al text
s, u	Ratllat i subratllat
style	Estils personalitzats
sub, sup	Subíndex i superíndex
voffset	Desplaçament de la línia base
width	Amplada del text

Table 4: Atributs de TextMeshPro

La documentació per inserir textos a l'aplicació segueix bastant la nomenclatura dels anteriors.

```
text="n_text%someText"

Optional:
~animationTime #Default 0, set Transition Time in seconds

Ex:
text=This is the first video
```

```

text=This is the first video~1
curl —data "text=1%This is the first video" http://localhost:4444/
curl —data "text=1%This is the first video~1" http://localhost:4444/



---


setText=n_text%someText

Optional:
~animationTime #Default 0, set Transition Time in seconds

Ex:
setText=This is the first video
setText=This is the first video~1
curl —data "setText=1%Text has Changed" http://localhost:4444/
curl —data "setText=1%Text has Changed~1" http://localhost:4444/



---


#Useful for text justification and Centered Titles
pivotText=n_text%pivotX,pivotY #Default is Center 0.5,0.5 Pos 0,0 is Upper Left

Ex:
pivotText=0%0,0
pivotText=0%0.5,0.5
curl —data "pivotText=0%0,0" http://localhost:4444/
curl —data "pivotText=0%0.5,0.5" http://localhost:4444/



---


posText=n_text%posx,posy

Ex:
posText=0%0.5,0.5,1
curl —data "posText=0%0.4,0.5" http://localhost:4444/



---


destroyText=n_text

Optional:
~animationTime #Default 0, set Transition Time in seconds

Ex:
destroyText=0
curl —data "destroyText=0" http://localhost:4444/

```

Listing 25: Documentació Text | textReadme.md

4.2.13 Imatges

El sistema d'imatges funciona amb el component RawImage de Unity UI. No té gaire complicació, els requisits eren poder aplicar imatges de qualsevol format i possibilitat de transparència. Per descarregar la imatge, es comprova si es local o remota, i es descarrega utilitzant la llibreria UnityWe-

bRequest, com ja es feia amb els àudios anteriorment (veure addAudio.cs 13).

```
//Inicialitzar descarregador
UnityWebRequest www = UnityWebRequestTexture.GetTexture(info[1]);
yield return www.SendWebRequest();
//Descarregar Imatge
textureDownload = DownloadHandlerTexture.GetContent(www);
//Assignar imatge i mides
newImage.texture = textureDownload;
var texWidth = textureDownload.width;
var texHeight = textureDownload.height;
newImage.GetComponent<RectTransform>().sizeDelta = new Vector2 (texWidth,
    texHeight);
//Afegir a l'array de imatges
imagesArray.Add(nImage, newImage);
```

Listing 26: Codi Image | addImage.cs

La utilització d'aquesta funció és molt intuïtiva i no requereix de gaire explicació.

```
image="n_image%URL" #Initial Position is Optional

Optional:
#posx,posy,scalex,scaley #Default 0,0,0
~setTransitionTime #Default 0, set Transition Time in seconds

Ex:
image="0%https://www.image.png"
curl --data "image=0%https://www.image.png" http://localhost:4444/
```

```
setImage="n_image%URL"

Ex:
setImage=0%https://www.image.png
curl --data "setImage=0%https://www.image.png" http://localhost:4444/
```

```
poscaleImage=n_video%posx,posy,scalex,scaley #Do not use this for initial
      positions

Ex:
poscaleImage=0%0.5,0.5,1,1
curl --data "poscaleImage=0%0.5,0.5,1,1" http://localhost:4444/
curl --data "poscaleImage=0%0.2,0.3,1,0.5" http://localhost:4444/
curl --data "poscaleImage=0%0.5,0.5,0.2,0.2" http://localhost:4444/
```

```
alphaImage=n_video% #Alpha (0, 1)

Ex:
alphaImage=0%0  #Transparent
```

```

alphaImage=0%1 #Opaque
curl —data "alphaImage=0%1" http://localhost:4444/
curl —data "alphaImage=0%0" http://localhost:4444/
curl —data "alphaImage=0%0.5" http://localhost:4444/



---


destroyImage="n_image"

Optional:
~setTransitionTime #  

    Default 0, set Transition Time in seconds

Ex:
destroyImage=0
curl —data "destroyImage=1" http://localhost:4444/
curl —data "destroyImage=0~3" http://localhost:4444/

```

Listing 27: Documentació Image | imageReadme.md

4.2.14 Scenes Management

L'aplicació permet guardar i carregar escenes personalitzades. Aquestes escenes s'emmagatzem en fitxers binaris que s'entenen millor amb el Unity que els JSON [53]. El «SceneManager» es compon de 2 blocs, el bloc de save, i el de load.

El bloc de save, guarda parseja tots els objectes de l'escena, i els guarda en una classe pròpia anomenada Save Data que conté tots els camps necessaris a omplir per guardar l'escena.

```

[System.Serializable]
public class SaveData
{
    public List<Videos> videos;
    public List<Texts> texts;
    public List<Images> images;
    public List<Audios> audios;
    public string backImagePath;
    public string backVideoPath;
}

```

Listing 28: SaveDataStruct | saveData.cs

Guarda en llistes tots els vídeos, imatges, textos i background. Després formata tota l'estructura en un fitxer binari i el guarda al path per defecte de Unity.

```
BinaryFormatter bf = new BinaryFormatter();
FileStream fileStream = File.Create(Application.persistentDataPath + "/" +
    sceneName);
bf.Serialize(fileStream, saveData);
fileStream.Close();
```

Listing 29: SaveBinary | saveBinary.cs

El bloc de càrrega, fa una mica el procés contrari, carrega el fitxer binari en una nova estructura SaveData. Passant per tots els elements de l'estructura, va executant totes les funcions pertinents per acabar obtenint el mateix resultat.

```
BinaryFormatter bf = new BinaryFormatter();
FileStream fileStream = File.Open(Application.persistentDataPath + "/" +
    sceneName, FileMode.Open);
SaveData saveReceive = bf.Deserialize(fileStream) as SaveData;
fileStream.Close();

//Executa totes les funcions pertinents per cada objecte de SaveData
```

Listing 30: LoadData | loadData.cs

A més, la transició entre aquestes escenes pot ser de diferent tipus. Actualment es permeten 2 estils. El primer seria el més bàsic, es fa a una fosa a negre, es carrega la següent escena, i un cop carregada, es fa visible la següent. L'altre fa un efecte de unes cortines que es tanquen, surt un logo, que pot ser el del client o qualsevol imatge, i un cop carregada la següent escena, s'obre de nou fent visible la nova distribució.

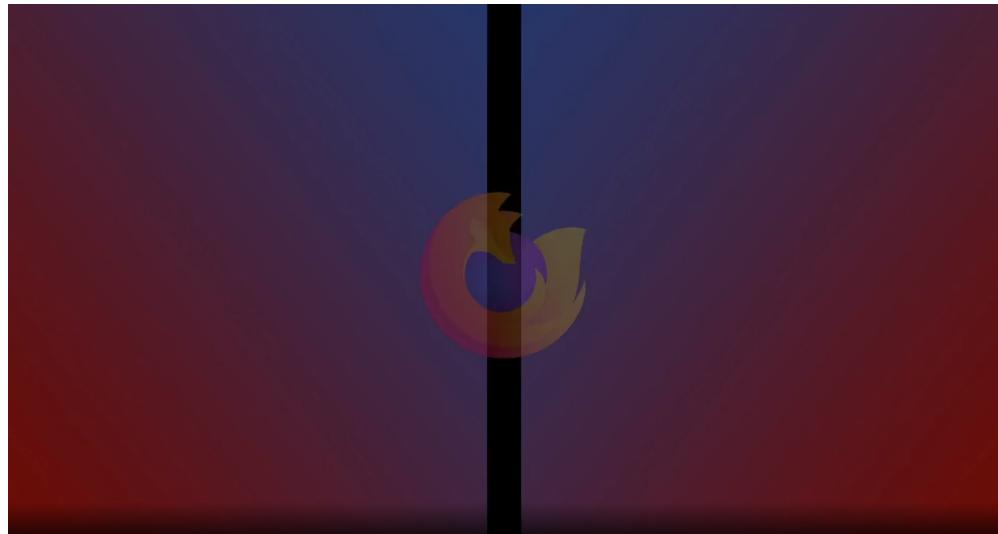


Figure 41: Tancant Escena



Figure 42: Escena Tancada

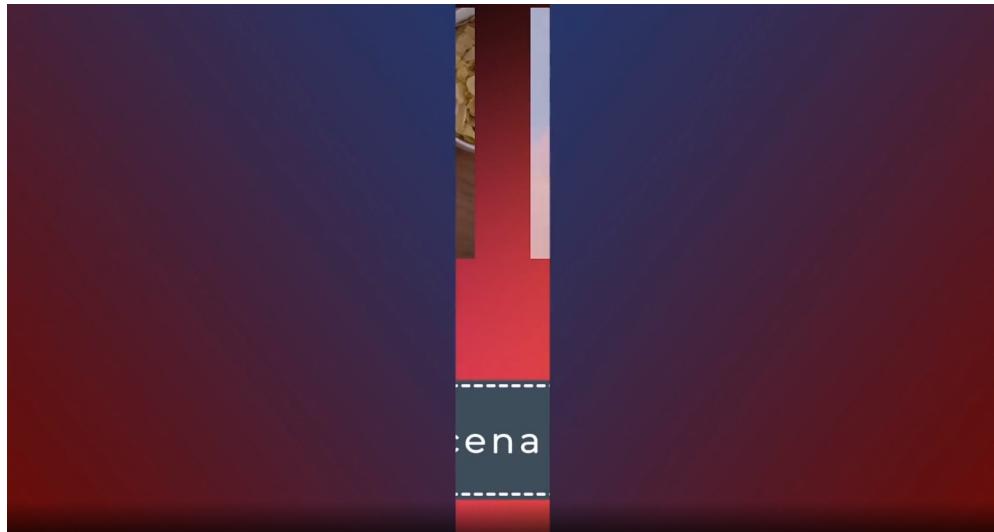


Figure 43: Obrint Escena

4.2.15 Debugging

Aquesta secció està pensada pel desenvolupament de l'aplicació. Ja que quan s'utilitza l'aplicació compilada al servidor no es té una consola que mostri tot el log, s'ha integrat una consola que es mostra a sobre dels vídeos, per poder debuggar més fàcilment. Només cal executar la crida `debug=true` per activar aquesta consola de desenvolupament.

```
curl --data "debug=true" http://localhost:4444/
```

Un cop la tenim activada, tots els errors que passin al Unity aniran sortint per allà. A més també es disposa d'algunes comandes de control per rebre informació. Com pot ser retornar una llista amb totes els vídeos, imatges, textos i escenes disponibles.

```
debug=listVideos
```



Figure 44: Debug Console Test

A més, aquestes funcions de debug també es pot fer que retornin la resposta a través de terminal on fem les crides, si no volem que els vídeos quedin tapats per el debug. Això ho fem indicant la comanda «status» enlloc de «debug».

```
curl --data "status=listVideos" http://localhost:4444/
```

```
alex@gallifrey:~$ curl --data "status=listVideos" http://localhost:4444/
OPENGL NATIVE PLUGIN ERROR: GL_INVALID_OPERATION: Operation illegal in current state
Image Number 1 | Path: https://snoopyimages.com/images/2021/06/07/Pngtreetwitch.png
Image Number 2 | Path: https://snoopyimages.com/images/2021/06/07/Pngtreetwitch.png
Image Number 3 | Path: https://snoopyimages.com/images/2021/06/07/Pngtreetwitch.png
Image Number 4 | Path: https://snoopyimages.com/images/2021/06/07/Pngtreetwitch.png
Video Number 1 | Playing: True | Path: https://assets.mixkit.co/videos/preview/mixkit-discussing-the-project-in-front-of-the-computers-23043-large.mp4 | Volume: 1
Video Number 2 | Playing: True | Path: https://assets.mixkit.co/videos/preview/mixkit-four-scientists-talking-about-research-22999-large.mp4 | Volume: 1
Video Number 3 | Playing: True | Path: https://assets.mixkit.co/videos/preview/mixkit-speaker-talking-to-an-audience-23138-large.mp4 | Volume: 1
Video Number 4 | Playing: True | Path: https://assets.mixkit.co/videos/preview/mixkit-a-woman-answering-a-phone-call-on-a-green-screen-24385-large.mp4 | Volume: 1
-----
alex@gallifrey:~$ curl --data "status=listImages" http://localhost:4444/
Image Number 4 | Path: https://snoopyimages.com/images/2021/06/07/Pngtreetwitch.png
Image Number 2 | Path: https://snoopyimages.com/images/2021/06/07/Pngtreetwitch.png
Image Number 3 | Path: https://snoopyimages.com/images/2021/06/07/Pngtreetwitch.png
Image Number 1 | Path: https://snoopyimages.com/images/2021/06/07/Pngtreetwitch.png
-----
```

Figure 45: Status Response

4.3 Sortida Aplicació

4.3.1 Video

Per la sortida de l'aplicació de Unity, s'utilitzarà FFmpeg, que capturarà un monitor virtual creat anteriorment. Per tant, els passos necessaris a fer son:

Crear un monitor virtual amb Xorg, que carregarà la configuració de l'arxiu xorg.conf

```
sudo Xorg :0
```

El següent pas serà executar el binari de Unity, que contindrà tot el projecte, amb la comanda

```
DISPLAY=:0 /home/wct/Builds/WCT.x86_64 -logFile  
~/Logs/"$(date +"%Y_%m_%d_%I_%M_%p").log"
```

Aquesta comanda executarà l'arxiu WCT.x86_64 en el monitor virtual :0, que és el creat anteriorment amb Xorg :0. A més, generarà un arxiu de log, per poder tenir un seguiment dels problemes que puguin sorgir.

Per últim, s'haurà d'exportar el contingut del monitor a un flux RTMP o el que es desitgi. Hi ha moltes maneres de fer-ho, després de moltes proves, una de les més òptimes és la següent:

```
ffmpeg -y -f x11grab -draw_mouse 0 -s 1920x1080  
-framerate 30 -async 1 -i :0.0+0,0 -f pulse -i default  
-pix_fmt yuv420p -c:v h264 -preset superfast -maxrate  
4000k -bufsize 4000k -c:a aac -b:a 128k -f flv  
rtmp://output.com
```

4.4 Rendiments Màxims i Tests

L'aplicació s'executa a una màquina s3 de AWS. Té 4 cores de CPU i una GPU Nvidia Tesla T4. És de les màquines més senzilles amb GPU, i ja ha donat uns resultats bastant sorprenents, pel que en el cas de necessitar més potència, no hi hauria cap problema en seleccionar una de més potent. Pel

que s'ha aconseguit un dels resultats clau del projecte, escalabilitat. Això permet també oferir una amplia selecció de serveis amb diferents preus, per tal que un usuari que no necessiti tanta potència tingui un preu més baix i aconseguir uns preus atractius si es desitgés.

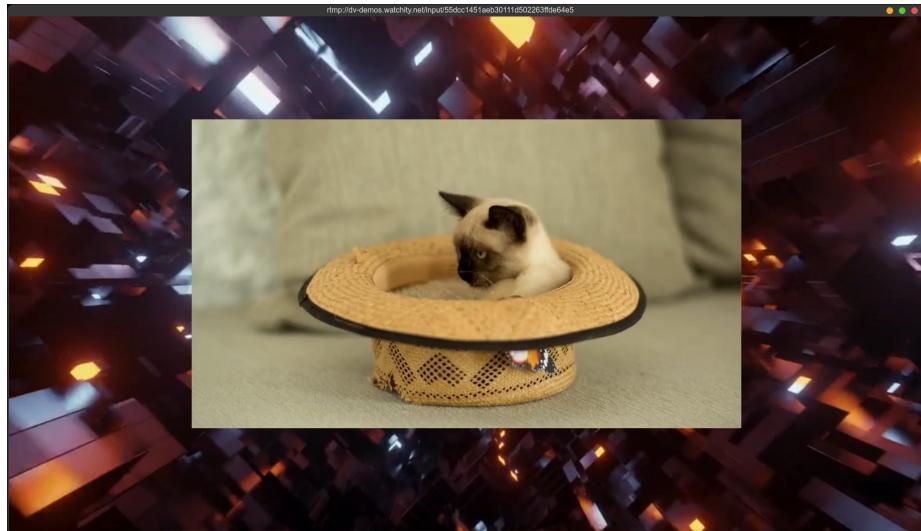


Figure 46: Mostra Resultat Unity

El processador acostuma estar en una càrrega de 60-80%, dependent del contingut que s'estigui reproduint. Tenint en compte que s'està ocupant del Unity i FFmpeg a la vegada. Es podria investigar per tenir la sortida de FFmpeg carregada a la GPU enlloc de tenir-la al processador. Però de moment els resultats ja han sigut òptims, pel que no és un canvi totalment necessari.

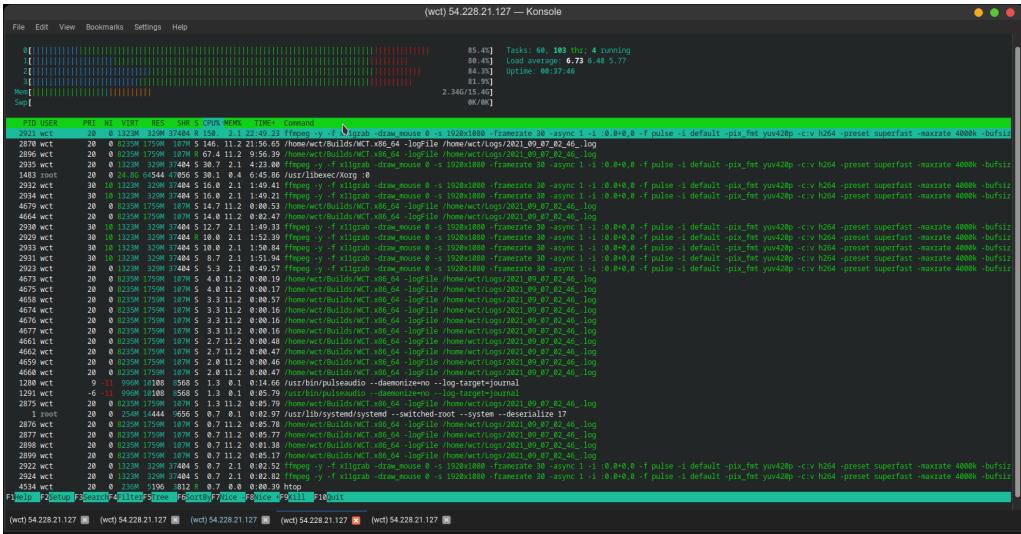


Figure 47: Htop Unity i FFmpeg

La GPU té una càrrega bastant variable, però de VRAM utilitzà molt poca, potser es podria aprofitar tota aquesta VRAM per utilitzar-la en algun altre procés. S'hauria d'investigar i seria un treball d'optimització per si es vol aconseguir una càrrega més equilibrada del servidor. En tot cas com ja s'ha comentat, s'aconsegueixen uns rendiments òptims i es considera més important que no faltin recursos.

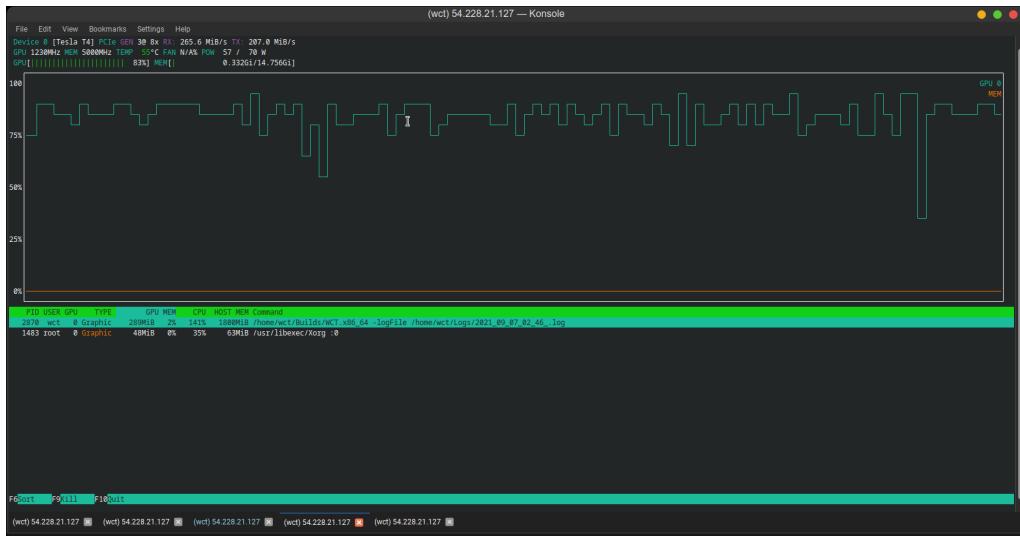


Figure 48: NVTOP Unity i FFmpeg

5 Conclusions

Amb aquest projecte s'ha descobert un món de possibilitats tant per mi, com per Watchity i el món audiovisual. S'han descobert molts aspectes del Unity que semblaven inviables d'ajuntar amb l'arquitectura del Mixer anterior, i que finalment amb molt esforç i estudi, s'ha aconseguit de manera òptima.

S'ha aconseguit els objectius que es van definir, inclús s'han afegit algunes funcionalitats extres que han anat sorgint en el transcurs del desenvolupament. A més de la satisfacció extra de saber que el projecte tindrà una utilitat real a una empresa.

Veient el potencial que té, s'espera que aquestes tecnologies evolucionin fins a un punt on la televisió o el cine, utilitzi moltes tècniques dels videojocs.

L'aplicació té molts punts forts, gràcies a les animacions aconseguides amb Unity, l'aspecte de les transicions i moviments són molt professionals. Té moltes possibilitats de textos, imatges, compatibilitat de vídeo, que fan que sigui molt diferencial a les altres opcions existents. L'ús és senzill, sobretot gràcies a l'API que s'ha desenvolupat al voltant del projecte, que permetrà calcular totes les posicions i moviments de forma automàtica, per a que si un usuari vol una escena no gaire complicada ho pugui fer amb uns pocs passos. I el que es considera el punt més fort i innovador, s'executa tot al núvol i no es requereix de molts recursos. Permetent que en un futur es pugui utilitzar inclús en dispositius mòbils o tablets. Un altre punt fort és que sigui remot, i la seva escalabilitat, permetent un servei personalitzat per a cada usuari, depenent de la complexitat de les escenes que es desitgi.

Un aspecte dèbil del projecte, són els problemes que s'ha tingut de compatibility entre Unity / Linux / Nvidia Drivers. Unity sí que té un software molt pel sistema operatiu utilitzat, encara que també s'ha d'entendre que no és el seu target principal. Per altre banda, Nvidia proporciona uns drivers que no són open-source, pel que ja trobarem moltes incompatibilitats i errors que només es podran solucionar si la companyia ho permet. Un exemple d'incompatibilitat és que els drivers propietaris de Nvidia no permeten l'ús de VA-API, una de les tecnologies més potents per acceleració de vídeo. Com es comenta no és del tot un impediment, sinó que és un obstacle molest, fàcil de solucionar, però que no depèn dels usuaris, sinó d'una empresa externa com és Nvidia. Pel que pot quedar com una línia de futur si decideixen fer algun canvi en aquest aspecte.

Si en un futur es decidís alliberar el codi del seus drivers, o com a mínim, donar més suport als usuaris de Linux, es podrien aconseguir uns resultats

molt superiors a Windows, a més de totes les avantatges que Linux dona per a servidors.

6 Línies de futur

A mida que s'anava desenvolupat el projecte, amb el descobriment del potencial que tenen aquestes tecnologies, han anat sorgint noves idees per millorar i sobretot, innovar més en el món audiovisual. Alguns d'aquests punts que es realitzaran, o tenen la possibilitat de realitzar-se en un futur són:

- Animacions de textos.
- Labels de textos i banners.
- Playlists de vídeos.
- Macro: possibilitat de emmagatzemar una seqüència de comandes, per reproduir-la més tard automàticament.
- Multi Output Audio: d'un mateix streaming, poder seleccionar entre un idioma o un altre, anant tota la informació pel mateix flux de dades.

El projecte ha motivat molt a tothom que ha estat involucrat, i es seguirà desenvolupant per aconseguir arribar al màxim potencial que es pugui.

7 Referències Imatges

- 1 Watchity. (2019, September 9). Frontend Mixer [Graph]. Frontend Mixer. https://help.watchity.com/hc/article_attachments/360027399494/Tipo_Production.png
- 2 SeleniumHQ. (2021, July 7). Selenium Driver Graph [Graph]. Selenium Driver Graph. https://www.selenium.dev/documentation/images/basic_comms.png?width=400px
- 3 puppeteer. (2018, January 12). Puppeteer Logo [Graph]. Logo. <https://user-images.githubusercontent.com/10379601/29446482-04f7036a-841f-11e7-9872-91d1fc2ea683.png>
- 4 WebGL Example. (2017, March 17). [Illustration]. Aquarium. <https://1stwebdesigner.com/wp-content/uploads/2014/03/aquarium.png>
- 5 Quixel. (2019, March 20). Quixel Megascan [Render]. UE4 Demo. <https://www.youtube.com/watch?v=9fC20NWhx4s>
- 6 Spiris, S. (2014, December 25). Blueprint [Graph]. UE4 Simple Blueprint. <https://answers.unrealengine.com/storage/temp/24917-rotate90degrees.png>
- 7 Neginfinity, N. (2013, January 27). Chaos Blueprint [Graph]. UE4 Blueprint Example of Bad Node Programming. <https://forum.unity.com/proxy.php?image=https%3A%2F%2Fpbs.twimg.com%2Fmedia%2FDeiyJrDUwAAJQJ8.jpg&hash=f36b58c4bd4dc432a0a0ff8c62b453c4>
- 8 Daria, D. (2021, July 12). Ranking Languages 2020 [Graph]. Darly Solutions. <https://darly.solutions/the-most-popular-programming-languages-in-2021/>
- 9 Unity. (2018, January 16). Book of the Dead - Unity Interactive Demo - Teaser. YouTube. https://www.youtube.com/watch?v=DDsRfbfnC_A
- 10 BATTLESTATE GAMES LIMITED. (2015, December 22). Pantalla de Pre-Alfa 19. www.escapefromtarkov.com.

<https://www.escapefromtarkov.com/media/id/29?lang=es>

11 Aakanksha, A. (2016, May 12). Diagrama Protocol-Format-Códec [Graph]. StreamShark. <https://streamshark.io/blog/understanding-codecs-and-formats/>

12 Ruether, T. (2021, March 16). TCP vs UDP [Graph]. Wowza. https://www.wowza.com/wp-content/uploads/Graphic-UDP-Vs-TCP-Diagram_1150x685.webp

13 Ruether, T. (2021a, March 16). Graph Latency Streaming Protocols [Graph]. Wowza. <https://www.wowza.com/wp-content/uploads/latency-continuum-2021-with-protocols-700x300-1.webp>

14 FFmpeg. (2013, April 10). FFmpeg Logo [Logo]. Wikipedia Media. https://upload.wikimedia.org/wikipedia/commons/thumb/5/5f/FFmpeg_Logo_new.svg/1280px-FFmpeg_Logo_new.svg.png

15 Brinkmann, M. (2009, July 7). VLC 1.0 [Photograph]. Ghacks.Net. <https://www.ghacks.net/2009/07/07/vlc-media-player-1-0-released/>

16 Pereyro, S. (2017, October 31). Screencast with Headless Chrome. Google Docs. https://docs.google.com/presentation/d/1b-HvxKmeFYE3qoNDJDHfbgVwEbi0QgmTvpH8w4CK5Tw/edit#slide=id.gc6f90357f_0_0

17 Vicente A. (2021, March 9) Test Headless Chrome Index. Own elaboration.

19 NewTek. (2016, August 31). NDI VLC Output Config [Config]. NewTek. <https://www.newtek.com/blog/tips/vlc-media-player-and-newtek-ndi-vlc-plugin/>

20 NewTek. (2020b, November 20). NDI Receiver Unreal [Graph]. NewTek. <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRMEVIL0BGn1XzKBxR2VKpZowQagdDYYENRmA&usqp=CAU>

- 21 Vicente A. (2021, June 1) Test NDI Receiver. Own elaboration.
- 22 Unity Direction Kit. (2016a, June 4). UMP Functionalities [Graph]. Asset Store. <https://assetstorev1-prd-cdn.unity3d.com/key-image/9f96a930-72d3-4972-a650-f4ea4f6e326b.webp>
- 23 Vicente A. (2021, January 14) Test Unity UMP. Own elaboration.
- 24 Vicente A. (2021, March 25) Test AVPro vs UMP. Own elaboration.
- 25 Vicente A. (2021, March 25) Test UMP vs AVPro. Own elaboration.
- 26 CentOS. (2015, December 12). CentOS [Logo]. Wiki. <https://upload.wikimedia.org/wikipedia/commons/thumb/b/bf/Centos-logo-light.svg/658px-Centos-logo-light.svg.png>
- 27 Vicente A. (2021, January 18) Esquema Arquitectura. Own elaboration.
- 28 Vicente A. (2021, January 18) Exemple Bug Gràfic VSync. Own elaboration (Kingsman 2014).
- 29 Dented Pixel. (2021, January 1). Dented Pixel [Logo]. Twitter. https://pbs.twimg.com/profile_images/1470829398/LogoSquareLarge_400x400.png
- 30 Vicente A. (2021, March 25) Esquema Arquitectura. Own elaboration.
- 31 rzeronte. (2020, January 2). Z-Buffer [Render]. Brakeza. https://brakeza.com/wp-content/uploads/2020/01/z_buffer_penguins_image.png
- 33 Vicente A. (2021, February 18) Crop WCT Màscara. Own elaboration.
- 34 Vicente A. (2021, February 18) Resultat Crop WCT. Own elaboration.
- 35 Vicente A. (2021, February 22) Crop Shader Graph. Own elaboration.

- 36 Vicente A. (2021, February 22) Shadow. Own elaboration.
- 37 Vicente A. (2021, February 22) Shadow. Own elaboration.
- 38 Vicente A. (2021, April 30) Chroma Node Shader Graph. Own elaboration.
- 39 Vicente A. (2021, April 30) Sub Shaders Chroma. Own elaboration.
- 41 Vicente A. (2021, April 19) Tancant Escena. Own elaboration.
- 42 Vicente A. (2021, April 19) Escena Tancada. Own elaboration.
- 43 Vicente A. (2021, April 19) Obrint Escena. Own elaboration.
- 44 Vicente A. (2021, February 17) Debug Console. Own elaboration.
- 45 Vicente A. (2021, February 17) Status Response. Own elaboration.
- 46 Vicente A. (2021, June 05) Resultat Frame. Own elaboration.
- 47 Vicente A. (2021, June 05) Resultat HTOP. Own elaboration.
- 48 Vicente A. (2021, June 05) Resultat NVTOP. Own elaboration.

References

- [1] Watchity. (2019b, September 13). ¿Qué es el Control Room (Live Distribution)? Watchity - Help Center. <https://help.watchity.com/hc/es/articles/360017584293--Qu%C3%A9-es-el-Control-Room-Live-Distribution->
- [2] Watchity. (2020, July 8). Share cuts on Social Networks. Watchity - Help Center. <https://help.watchity.com/hc/en-us/articles/360023619094-Share-cuts-on-Social-Networks>
- [3] Watchity. (2019, September 9). ¿Qué es el Mixer? Watchity - Help Center. <https://help.watchity.com/hc/es/articles/360018866273--Qu%C3%A9-es-el-Mixer->
- [4] Pereyro, S. (2018, February 09). Live streaming with headless chrome - Empirical. Empirical. <https://blog.goempirical.com/how-to-use-headless-chrome-to-screencast-audio-and-video-to-an-rtmp-endpoint-216ccffdde4db>
- [5] FreeDesktop. (2021, January 16). PulseAudio. <https://www.freedesktop.org/wiki/Software/PulseAudio/>
- [6] Google. (2021, February 25). Getting Started with Headless Chrome | Web |. Google Developers. <https://developers.google.com/web/updates/2017/04/headless-chrome>
- [7] SeleniumHQ. (2018, December 19). SeleniumHQ Browser Automation. Selenium. <https://www.selenium.dev/>
- [8] SeleniumHQ. (2021a, July 7). Driver requirements :: Documentation for Selenium. Documentació Selenium. https://www.selenium.dev/documentation/en/webdriver/driver_requirements/
- [9] Smirnov, A. (2020, November 18). How to run a headless Chrome browser in Selenium WebDriver. Medium. <https://itnext.io/how-to-run-a-headless-chrome-browser-in-selenium-webdriver-c5521bc12bf0>
- [10] Google. (2021a, February 11). Puppeteer | Tools for Web Developers |. Google Developers. <https://developers.google.com/web/tools/puppeteer>

- [11] Apache. (2004, January 1). Apache License, Version 2.0. Apache License. <https://www.apache.org/licenses/LICENSE-2.0>
- [12] Mozilla. (2021, May 27). WebGL: 2D and 3D graphics for the web - Web APIs | MDN. WebGL Support by Browser. https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API#browser_compatibility
- [13] Dealessandri, M. (2020, May 14). What is the best game engine: is Unreal Engine right for you? GamesIndustry.Biz. <https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-unreal-engine-4-the-right-game-engine-for-you>
- [14] Eric Peckham, E. P. (2019, October 17). TechCrunch is now a part of Verizon Media. Extra Crunch. <https://techcrunch.com/2019/10/17/how-unity-built-the-worlds-most-popular-game-engine/>
- [15] Technologies, U. (2021, January 1). Multiplatform. Unity. <https://unity.com/features/multiplatform>
- [16] TIOBE. (2021, July 1). index | TIOBE - The Software Quality Company. <https://www.tiobe.com/tiobe-index/>
- [17] Technologies, U. (2021b, January 1). Unity - Scripting API: MonoBehaviour. Unity. <https://docs.unity3d.com/ScriptReference/MonoBehaviour.html>
- [18] Lagarde, S. (2020, February 24). HDRP: Out of Preview in 2019.3. Unity Blog. <https://blog.unity.com/technology/hdrp-out-of-preview-in-2019-3>
- [19] Texto: Luz Fernández. (2002, December 28). La copia de películas se hace fácil en Internet. Cinco Días. https://cincodias.elpais.com/cincodias/2002/12/28/tecnologia/1041322677_850215.html
- [20] Bychok, A. (2021, May 21). Streaming Protocol Comparison: RTMP, WebRTC, FTL, SRT – Restream Blog. Ultimate Live Streaming Hub – Restream Blog. <https://restream.io/blog/streaming-protocols/>
- [21] Fernández, Y. (2019, November 7). Qué es el códec AV1 y cuáles son sus ventajas. Xataka. <https://www.xataka.com/basics/que-codicec-av1-cuales-sus-ventajas>

- [22] Hao, D. T. G. (2019, August 7). Mixer's Faster Than Light streaming protocol explained. Dot Esports. <https://dotesports.com/streaming/news/mixers-faster-than-light-streaming-protocol-explained>
- [23] Rus, C. (2020, June 23). Microsoft cierra Mixer, su alternativa a Twitch en la que tenía en exclusiva a superestrellas de eSports como... Xataka. <https://www.xataka.com/videojuegos/microsoft-cierra-mixer-su-alternativa-a-twitch-que-tenia-exclusiva-a-superestrellas-esports-como-ninja>
- [24] Twitch. (2015, December 18). Twitch Blog | Twitch Engineering: An Introduction and Overview. Twitch Blog. <https://blog.twitch.tv/en/2015/12/18/twitch-engineering-an-introduction-and-overview-a23917b71a25/>
- [25] FFmpeg. (2021, January 1). About FFmpeg. <https://ffmpeg.org/about.html>
- [26] FFmpeg. (2021b, January 1). FFmpeg Filters Documentation. <https://ffmpeg.org/ffmpeg-filters.html>
- [27] VideoLAN. (2021, January 1). Free Software and Open Source video streaming solution for every OS! - VideoLAN. <https://www.videolan.org/videolan/>
- [28] VideoLAN. (2021a, January 1). Contribute to the project - VideoLAN. <https://www.videolan.org/contribute.html>
- [29] VideoLAN. (2019, March 4). libVLC Tutorial - VideoLAN Wiki. WikiVLC. https://wiki.videolan.org/LibVLC_Tutorial/
- [30] Pereyro, S. (2017, October 31). Screencast with Headless Chrome. Google Docs. https://docs.google.com/presentation/d/1b-HvxKmeFYE3qoNDJDHfbgVwEbi0QgmTvpH8w4CK5Tw/edit#slide=id.gc6f90357f_0_0
- [31] Postman. (2021, January 1). Postman | The Collaboration Platform for API Development. <https://www.postman.com/>
- [32] Google. (2021a, January 1). Chrome DevTools Protocol. Page Domain. <https://chromedevtools.github.io/devtools-protocol/tot/Page/#method-startScreencast>

- [33] Filmaffinity. (2014, January 1). *Nightcrawler* (2014). <https://www.filmaffinity.com/es/film779937.html>
- [34] Filmaffinity. (1979, January 1). *Stalker* (1979). <https://www.filmaffinity.com/es/film534365.html>
- [35] Google. (2017, September 13). Autoplay policy in Chrome. Chrome Developers. <https://developer.chrome.com/blog/autoplay/>
- [36] FFmpeg. (2021c, January 1). FFmpeg Formats Documentation. <https://ffmpeg.org/ffmpeg-formats.html#image2-1>
- [37] FBSSamples. (2020, May 8). GitHub - fbsamples/Canvas-Streaming-Example: This project contains example code showing how to go live on Facebook using a element as a source. GitHub. <https://github.com/fbsamples/Canvas-Streaming-Example>
- [38] NewTek. (2021, January 1). About NDI - Network Device Interface. <https://www.ndi.tv/about-ndi/>
- [39] NewTek. (2020, November). NDI Unreal Documentation. NDI Unreal Documentation. http://514f211588de67e4fdcf-437b8dd50f60b69cf0974b538e50585b.r63.cf1.rackcdn.com/Utilities/SDK/NDI_SDK_Unreal_Engine/4.25/NDI%20IO%20Plugin%20for%20Unreal%20Engine%20Quickstart%20Guide.pdf
- [40] Unity Direction Kit. (2016, June 4). UMP (Win, Mac, Linux, WebGL) | Video. Unity Asset Store. <https://assetstore.unity.com/packages/tools/video/ump-win-mac-linux-webgl-49625>
- [41] Verma, A. (2015, August 30). Access denied | fossbytes.com used Cloudflare to restrict access. Fossbytes. <https://fossbytes.com/ubuntu-linux-is-the-most-popular-operating-system-in-cloud/>
- [42] RenderHeadsLTD. (2020, March 12). AVPro Video - Ultra Edition | Video. Unity Asset Store. <https://assetstore.unity.com/packages/tools/video/avpro-video-ultra-edition-184294>
- [43] Keijiro, K. (2019, October 24). GitHub - keijiro/FFmpegOut: Video capture plugin for Unity with FFmpeg. GitHub. <https://github.com/keijiro/FFmpegOut>

- [44] AMD. (2016, February 16). Vulkan. <https://www.amd.com/es/technologies/vulkan>
- [45] Amazon. (2021, January 1). AWS | Cloud Computing - Servicios de informática en la nube. Amazon Web Services, Inc. <https://aws.amazon.com/es/>
- [46] AWS vs Azure-Who is the big winner in the cloud war? (2021, July 27). ProjectPro. <https://www.dezyre.com/article/aws-vs-azure-who-is-the-big-winner-in-the-cloud-war/401>
- [47] López, J. C. (2019, February 20). La biblia del HDR: qué es, cuáles son los estándares actuales y cómo no todo lo que dicen que es HDR lo es en... Xataka. <https://www.xataka.com/televisores/biblia-hdr-que-cuales-estandares-actuales-como-no-todo-que-dicen-que-hdr-efectivamente>
- [48] Unity. (2014, August 11). LeanTween | Animation Tools. Unity Asset Store. <https://assetstore.unity.com/packages/tools/animation/leantween-3595>
- [49] Technologies, U. (2020, January 25). Unity - Scripting API: RenderQueue. Unity3d. <https://docs.unity3d.com/ScriptReference/Rendering.RenderQueue.html>
- [50] Technologies, U. (2021, January 1). Shader Graph | Crea tus shaders visualmente con. Unity. <https://unity.com/es/shader-graph>
- [51] CodeMonkey. (2019, November 24). Code Monkey - Sprite Outline - 2D Shader Graph Tutorial. <https://unitycodemonkey.com/video.php?v=FvQFhkS90nI>
- [52] Digital Native Studios. (2021, January 1). Rich Text, TextMesh Pro Documentation. TextMeshPro. <http://digitalnativestudios.com/textmeshpro/docs/rich-text/#style>
- [53] Harald, H. (2020, January 1). JSON VS Binary. Esoteric Software. <http://es.esotericsoftware.com/forum/JSON-VS-Binary-13339>