



UNIVERSITAT POLITÈCNICA DE CATALUNYA

Escola Politècnica Superior d'Enginyeria de Manresa

Sistema remot de visió estereoscòpica.

9 d'octubre de 2020

treball de fi de grau que presenta

JOSEP FENOY BERMUDEZ

en compliment dels requisits per assolir el

GRAU D'ENGINYERIA EN SISTEMES TIC

Direcció: Sebastià Vila Marta

Aquesta obra està subjecta a una llicència Attribution-NonCommercial-ShareAlike 3.0 Spain de Creative Commons. Per veure'n una còpia, visiteu <https://creativecommons.org/licenses/by-nc-sa/3.0/es/deed.ca> o envieu una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

A Néstor, per tot el seu suport i tota la inspiració
que sempre m'ha donat des que era un nen.

Agraïments

Voldria agrair el suport que he tingut en tot moment tant per part de la meva família com dels meus amics. Sense ells segurament no hauria pogut seguir endavant de la manera que ho he fet aquests mesos.

També vull agrair al meu tutor Sebastià Vila pels seus consells i ajuda per a poder avançar quan he estat parat sense saber què fer.

Finalment, estic molt agraït als professors que m'han ajudat a aprendre els últims quatre anys els coneixements necessaris per a poder afrontar un projecte com aquest.

Resum

Aquest treball tracta de desenvolupar una aplicació de visió estereoscòpica per sistema remot per a un dispositiu de realitat virtual, que permeti a l'usuari sentir una sensació d'immersió a l'hora d'utilitzar-la. Per a aconseguir-ho, s'avalua la viabilitat de múltiples solucions per a l'aplicació, tot fent ús de llibreries i eines de codi obert que actualment es troben en desenvolupament.

Adicionalment, s'ha realitzat una cerca i estudi sobre una varietat de conceptes i tecnologies considerades necessàries per a la resolució dels reptes que s'esperen trobar.

Abstract

The goal of this project is the development of a remote stereoscopic vision system for a virtual reality headset, which would provide the user with a fully immersive experience. To achieve this goal, the viability of several potential solutions is evaluated, utilizing open-source tools and libraries that are still under development.

In support of this development, research was done on a variety of methods and technologies deemed necessary to address the expected challenges.

Índex

Resum	i
Abstract	i
1 Introducció	1
1.1 Context	1
1.1.1 La indústria de la realitat virtual	1
1.2 Possibles limitacions	2
1.2.1 Limitacions de Hardware	2
1.2.2 Limitacions de Software	2
1.2.3 Limitacions de sistema operatiu	3
1.3 Objectius	3
1.4 Resultats	3
1.5 Organització de la memòria	4
2 Conceptes essencials	5
2.1 Estereoscòpia	5
2.2 La realitat virtual	9
2.2.1 L'evolució de la realitat virtual	11
2.3 Renderització	14
2.3.1 Graphics pipeline	15
2.4 Transcodificació	16
2.4.1 Bitrate	17
2.4.2 Compressió de vídeo	18
2.5 Càmeres IP	19
2.6 RTSP	20
2.7 WebRTC	20
2.8 Unity	21
2.9 ADB	23
3 Plantejament del projecte	25
3.1 Possibles mètodes d'implementació	25
3.1.1 Valoracions de càmeres	25
3.1.2 Valoracions d'ulleres de realitat virtual	27
3.1.3 Valoracions de targetes gràfiques	27
4 Implementació del projecte	29
4.1 Unity Render Streaming	31

5	Conclusions	35
6	Treball futur	37
	Bibliografia	39

1 Introducció

1.1 Context

El simple fet de trucar a algú mitjançant una aplicació, sigui des d'un terminal mòbil o un ordinador i, a més, poder escollir si volem no només parlar-nos sinó veure'ns alhora, és una acció que lentament ha arrelat a la nostra societat com a una activitat més del nostre dia a dia. Plataformes com ara *YouTube*, *Instagram* o *Twitch* ens ofereixen la possibilitat de retransmetre vídeo en directe per a ser visualitzat per altres usuaris, arribant aquests a ser desenes de milions diàriament.

Ens trobem a un punt de no retorn com a societat, no podem assolir com seria tornar enrere tecnològicament i desfer-nos de tota l'electrònica que ens envolta i facilita la vida, i és encara més difícil quan veiem a diari nous avenços que fa uns anys eren pràcticament impossibles. Un exemple d'aquests serien les ulleres de realitat virtual, que després d'uns anys difícils on no s'estava segur sobre la viabilitat d'aquests aparells els trobem actualment cada cop més presents, arribant fins i tot a museus que han dissenyat aplicacions per a fer visites virtuals.

1.1.1 La indústria de la realitat virtual

En aquest treball descobrirem com funciona i com es defineix el concepte de la realitat virtual, així com els mètodes que s'han anat descobrint i han servit com a precursors dels aparells amb els quals podem disposar avui en dia. Veurem així el creixement d'aquest sector, començant pels primers visors d'imatges en tres dimensions, passant per la famosa *espasa de Dàmocles* i finalment arribant als sistemes actuals com el que utilitzarem per al nostre projecte.

Veurem també que, tot i portar gairebé tants anys en desenvolupament com altres tecnologies, aquesta es pot dir que encara es troba a les beceroles, ja que trobar informació i documentació sobre com desenvolupar aplicacions per a aquests dispositius és certament difícil. L'estat actual de la indústria és una carrera entre les actuals empreses més potents del mercat. Al podi trobem a *HTC*, *Oculus (Facebook)* i *Sony*. Quan parlem de facilitats per a desenvolupar aplicacions o videojocs per a aquestes plataformes, podríem dir que *Sony* es tracta de l'empresa que més inconvenients posa al respecte. Necessites parlar amb la companyia i adquirir un *devkit* per a poder treballar correctament, cosa que impossibilita fer-ho a un gran percentatge de persones, al tenir aquest un preu d'uns, aproximadament, 2.500€.

Respecte a les altres dues marques, desenvolupar aplicacions per a les seves ulleres té més facilitats que per a les de *Sony*, especialment *Oculus*, que ofereix un paquet oficial pel programa *Unity* per a poder treballar amb molta més facilitat. Així i tot, és

certament difícil trobar documentació oficial, i la comunitat de creadors treballa molt freqüentment amb fòrums per a poder-se ajudar mútuament.

A l'apartat 2.2 La realitat virtual veurem de què tracta la mateixa, tot analitzant possibles problemes que sempre l'han caracteritzat. Tanmateix, descobrirem el funcionament de les ulleres de realitat virtual i estudiarem els fenòmens que ens provoquen una sensació d'immersió en utilitzar un aparell d'aquesta índole.

1.2 Possibles limitacions

Les limitacions en aquest projecte seran quelcom que haurem de tenir present en tot moment. Es tracta d'un desenvolupament sobre una plataforma que encara es troba en una fase de canvi constant que, a més, té necessitat d'un hardware potent. Podríem dir que les limitacions que tindrem s'agrupen en tres grups.

1.2.1 Limitacions de Hardware

Creiem que el hardware (que també anomenarem pel seu acrònim: HW) pot ser que ens limiti més que el software. El motiu és principalment econòmic, ja que sabem que hi ha certs components necessaris per al desenvolupament del projecte com ara una targeta gràfica potent, o unes ulleres de realitat virtual. Aquests components no tenen preus precisament baixos, el que ens portarà a fer una cerca analitzant quins són els més assequibles.

El que farem llavors en aquest projecte és treballar amb tots els components que necessitem, però ho farem amb uns de gamma més baixa que els ideals per a poder permetre'ns adquirir-los. Al capítol 3 Implementació del projecte, el que farem serà valorar per a cada component quin model ens és assequible d'adquirir i de treballar amb ell. A més, també parlarem sobre quins components serien els idonis a l'hora de realitzar professionalment aquest mateix projecte.

1.2.2 Limitacions de Software

Opinar sobre les possibles limitacions que podem tenir amb el software (també conegut per l'acrònim: SW) previ inici del treball és molt difícil, ja que, quan realment trobarem limitacions aquí serà un cop haguem entrat en profunditat. Com hem dit, treballarem amb *Unity* com a plataforma de desenvolupament de l'aplicació. Ho farem ja que té uns paquets oficials de la marca *Oculus* que ens permetran desenvolupar l'aplicació per a les ulleres. *Unity*, però, és un programa complex que actualment té diferents versions que s'han anat acumulant amb el pas dels anys, sent la seva primera versió el juny de 2005 i la seva última versió estable a l'agost d'aquest mateix any.

L'aplicació que volem desenvolupar no és una aplicació que haurem de programar completament des de zero, a causa del grau de dificultat i del temps necessari per a realitzar-la, sinó que treballarem sobre altres projectes ja existents com a base per al nostre muntatge. És important considerar el fet que, ja que treballarem sobre ells, les limitacions a les quals s'enfronten també les tindrem nosaltres.

1.2.3 Limitacions de sistema operatiu

Per últim, també ens trobem limitats pel sistema operatiu (d'ara endavant també anomenat *SO*). Si bé és cert que amb el transcurs dels anys Unity cada cop més ha millorat la seva compatibilitat amb sistemes operatius com *Linux* o *Mac OS*, encara existeixen massa problemes que dificulten el desenvolupament d'aplicacions per a aquestes plataformes, especialment quan parlem de realitat virtual. I no només Unity pot donar problemes, sinó que els altres projectes dels quals depenem per a realitzar el nostre depenen també del sistema operatiu Windows, i no han estat encara (o en cas d'alguns directament no tenen plans de fer-ho) adaptats als altres SOs.

1.3 Objectius

L'objectiu principal del treball és arribar a desenvolupar una aplicació que tracti de, com bé diu el títol, un sistema remot de visió estereoscòpica. Aquesta visió estereoscòpica serà primerament simulada, amb una única càmera enviant el vídeo. En cas d'èxit s'implementarà una segona càmera per a tenir una visió estereoscòpica real. A més, volem que aquest sistema sigui visualitzat per unes ulleres de realitat virtual, per tal d'aconseguir una sensació d'immersió a l'hora de col·locar-se-les.

Per a arribar a aquest punt haurem de realitzar certs passos previs que també podem considerar com a objectius del treball, sent aquests els següents:

- 1) Estudi dels coneixements clau pel desenvolupament.
- 2) Anàlisi dels possibles mètodes d'implementació.
- 3) Cerca, valoració i compra del hardware necessari.
- 4) Cerca i instal·lació del software necessari.
- 5) Realització del projecte.

Veiem llavors que, tot i tenir com a objectiu final l'aplicació, gran part del projecte es basa en arribar a entendre què és el que està passant al seu darrere. Des del funcionament de la compressió de vídeo, que fa possible el fet de poder retransmetre a temps real o el seu emmagatzematge, fins als motius pels quals veiem en tres dimensions quan ens posem les ulleres.

1.4 Resultats

El resultat del treball el podem definir com a una aplicació escalable a altres sistemes de realitat virtual. Degut a problemes d'incompatibilitat amb els dispositius adquirits, no hem pogut realitzar el muntatge ideal, ja que aquest depenia tant del correcte funcionament de les ulleres de realitat virtual com de la compatibilitat de les càmeres amb el programari utilitzat.

L'aplicació realitzada, però, està dissenyada de manera que, en cas de disposar de dispositius més potents, es podria adaptar per al seu funcionament, aconseguint així l'objectiu principal d'una visió estereoscòpica mitjançant dues càmeres, com podem veure a l'apartat 6.

1.5 Organització de la memòria

La memòria podria considerar-se essencialment dividida en quatre capítols. El primer és un capítol introductori, que ens servirà per a aprendre i comprendre els conceptes que necessitarem a l'hora d'afrontar aquest projecte. Estudiarem des del concepte de l'estereoscòpia fins al funcionament d'una *IP Cam*, assolint la informació necessària de cadascun per a poder seguir endavant.

Al següent capítol veurem el plantejament del projecte. En aquest, analitzarem com el podríem implementar, valorarem els components que necessitarem i prendrem decisions a l'hora d'adquirir-los. No obstant això, com que probablement no siguin els més òptims, parlarem també d'aquells que serien més adequats per a la correcta realització d'aquest projecte a l'últim capítol.

En tercer lloc, veurem la implementació del projecte, la qual podríem considerar dividida en dues parts. Per una banda, parlarem de quina és la configuració necessària per a vincular l'ordinador amb el dispositiu, juntament amb el primer gran inconvenient que ens va fer haver de canviar la manera d'enfocar el problema. Per l'altra banda, expliquem com vam fer el projecte, juntament amb la descripció d'un nou software amb el que treballar, la configuració actual de l'ordinador per a poder-lo executar, i els resultats obtinguts.

Finalment, a l'últim bloc tenim les conclusions i treball futur, on analitzarem els resultats del treball respecte els objectius inicials, i exposarem les diferents opcions disponibles per a la millora del treball.

2 Conceptes essencials

Hem parlat de retransmissions de vídeo que poden ser visualitzades per milions de persones i d'ulleres que et permeten introduir-te, en certa manera, a un altre món. Tot i que és cert que estem acostumats a viure amb aquests avenços tecnològics, la pregunta que ens podem fer és: Sabem realment què ho fa possible?

Prèvia realització del treball pràctic, estudiarem un seguit de temes que es troben relacionats amb el mateix i que ens donaran la resposta a l'anterior pregunta.

2.1 Estereoscòpia

Per començar, hem d'esclarir que quan parlem de l'estereoscòpia estem parlant d'una tècnica, la qual té com a objectiu poder simular una sensació de profunditat sobre una imatge aprofitant el fenomen de l'estereòpsia. Varis intel·lectuals d'èpoques passades com poden ser Leonardo da Vinci i Euclides tractaren d'estudiar com funcionava la nostra manera de veure el món, la visió binocular. No obstant això, va ser amb Charles Wheatstone (1802-1875) quan es va construir el primer visor. Wheatstone va definir el fenomen de l'estereòpsia la següent manera:

«(...)The mind perceives an object of three dimensions by means of the two dissimilar pictures projected by it on the two retinae.(...)»

Traduïda, ens diu que la nostra ment és capaç de percebre que els objectes tenen tres dimensions gràcies al fet que aquests mateixos objectes projecten dues imatges diferents sobre cadascuna de les dues retines dels nostres ulls. Per a més informació sobre la seva obra, consultar [Wad02].

Es pot arribar a confondre, però en essència les imatges estereoscòpiques, també anomenades imatges estereogràfiques o imatges 3D, no tracten del mateix que una imatge amb perspectiva. Amb la perspectiva podem arribar a crear un efecte de profunditat, però no deixem de tenir una única imatge on el que veiem és el que interpretem. En canvi, aquestes tècniques tracten de generar una sensació aparentment real d'estar veient un objecte, escenari o punt que físicament no existeix a través d'una imatge que realment genera el nostre cervell, per tant, el que estem veient és una combinació d'imatges que interpretem com a una de sola.

Una imatge estereogràfica requereix dues fotografies que simulen el que estaria veient cadascun dels nostres ulls. Per a aconseguir-ho s'assigna a cada ull una de les dues imatges, que estan fetes des de gairebé el mateix punt de vista, però difereixen per una petita distància (de la mateixa manera que els nostres ulls) que anomenem disparitat

binocular. Els nostres ulls tenen una visió diferent del món entre ells, el que fa que, quan observem un punt amb cada ull el veiem desplaçat juntament amb l'escenari que el rodeja, ja que tenen dues línies de visió diferents. Aquesta distància s'anomena paral·laxi, i es mesura amb l'angle d'inclinació que existeix entre aquestes dues línies imaginàries. El nostre cervell és capaç d'interpretar aquestes dues imatges creant-nos una de sola i, en conseqüència, també ens augmenta el nostre camp visual respecte al que tindriem amb un sol ull.

La nostra vista funciona rebent els senyals lumínics generats pel reflex de la llum sobre l'escenari on ens trobem, però per a poder interpretar bé aquests senyals, els ulls han d'anar adaptant-se de forma constant, realitzant dos processos d'adaptació. En primer lloc, els ulls realitzen la rotació necessària de manera sincronitzada que els permet convergir sobre un mateix punt, moviment que s'anomena vergència òptica. En segon lloc, tenim l'acomodació visual, que tracta en certa forma de «regular la potència» amb la qual el nostre ull enfoca el seu objectiu. Concretament, el nostre cervell és capaç de calcular la distància entre el punt a observar i els nostres ulls, tot adaptant ràpidament una petita lent situada a l'interior de l'ull anomenada cristal·lí, que veu modificats el seu gruix i curvatura fins arribar a enfocar correctament el punt.

Hem d'afegir que treballar amb aquestes tècniques ens pot arribar a causar problemes físics si no són implementades de forma correcta, ja que, com hem vist, es basen en part en el moviment dels nostres ulls, i podem arribar a forçar-los fins a un punt on les molèsties comencin a aparèixer. Aquests problemes poden variar en funció de la persona, i poden sorgir com a combinació de múltiples factors, com han demostrat els estudis que s'han fet sobre aquest camp que podem veure a [MGS98]. D'entre aquests factors, destaquem els següents:

- *Edat.* S'ha demostrat que hi ha diferència en la percepció tridimensional segons l'edat. Concretament, la gent d'edat més avançada és menys sensible a l'hora de percebre profunditats i curvatures. I no només això, sinó que amb més edat també som més propensos a tenir problemes com per exemple la rivalitat binocular. Aquesta ens impedeix crear la imatge única a la qual estem acostumats, presentant dues imatges diferents alhora. D'aquestes dues, una domina i l'altra és reprimida alternant de forma periòdica. A més, la disparitat binocular en nens és més petita que en adults, motiu pel qual poden tenir problemes i molèsties en visualitzar escenes en 3D.
- *Gènere.* Hi ha estudis que han tractat de demostrar si tant els homes com les dones tenim la mateixa habilitat respecte a la percepció espacial. Els resultats decanten la balança cap al sexe masculí, significant que és menys possible tenir problemes com per exemple un mareig. Tanmateix, van efectuar-se als participants dels estudis unes enquestes que en certa manera *justifiquen* els resultats. I és que sembla que aquesta diferència entre els sexes ve definida pel fet que, especialment des de petits, els participants masculins tenien en comú que van participar en més activitats que ajuden a desenvolupar les habilitats de percepció espacial que les participants femenines. Per tant, aquest estudi s'haurà de renovar amb el pas dels

anys, ja que és molt possible que amb els canvis actuals i futurs de la societat respecte a la qual hi havia anys enrere, les noves generacions no tinguin aquesta diferència.

- *Experiència prèvia amb visualització tridimensional.* El nostre cervell sembla ser capaç d'adaptar-se a visualitzar contingut en tres dimensions amb el temps. El primer cop que s'utilitzen unes ulleres de realitat virtual ha de ser relativament breu, per a evitar experimentar marejos i fatiga, però si les utilitzem freqüentment podem arribar a adaptar-nos fins a poder estar sessions més i més llargues.

Respecte als problemes que ens podem trobar, poden ser classificats en diferents grups. A continuació parlarem breument de cadascun d'ells juntament amb petits exemples de problemes que hi pertanyen. A [WM11] podem trobar tota la informació sobre aquests.

- *Profunditat.* Aquests problemes venen relacionats amb el fet que el nostre cervell està acostumat al món real, on enfoca la vista sobre un objecte o punt que és allà físicament. En un exemple d'un sistema de visió estereoscòpica sobre una pantalla, els objectes no existeixen físicament, sinó que es troben projectats sobre una pantalla que és més a prop dels nostres ulls a la qual realment es trobaria aquest objecte o punt. Aquesta situació no és natural per al cervell, ja que la distància real a la qual trobaríem l'objecte no coincideix amb la que percebem. Per aquest motiu, els usuaris poden sentir mals de cap o incomoditats, especialment els primers cops. A la figura 2.1 podem veure una representació de l'exemple exposat.

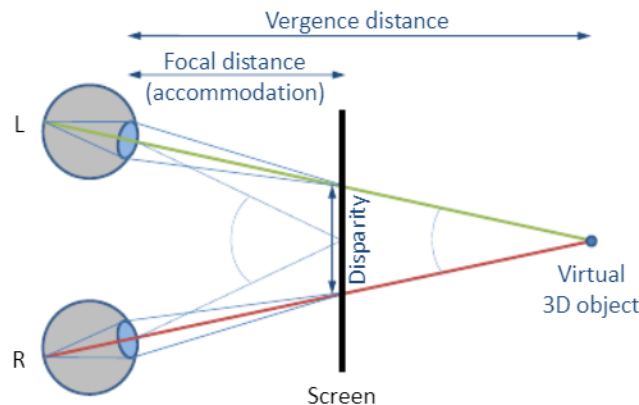


Figura 2.1: Esquema de visualització d'una imatge estereoscòpica.

- *Multivisió.* Quan treballem amb aquestes tècniques, hem d'assegurar-nos que cada ull està veient exactament la imatge que volem. Per aquest motiu, es prenen en consideració una llarga sèrie d'aspectes, dels quals en comentarem tres d'ells a continuació. Primer de tot, la construcció de la màquina ha d'assegurar que tant la part de l'ull dret com l'esquerre són idèntiques en els sentits de: sensors òptics, velocitat d'oclusió, gamma, apertura... En segon lloc, s'ha d'assegurar que

l'usuari no experimentarà una dessincronització de vídeo entre els dos ulls, ja que fins i tot una petita quantitat de fotogrames poden causar molèsties. En tercer i últim lloc, en cas de treballar utilitzant la xarxa, hem de tenir en consideració possibles propagacions d'errors, especialment si treballem amb una retransmissió independent per a cada ull.

- *Subtítols.* Afegir subtítols sobre un vídeo, imatge o videojoc que es trobi en format de dues dimensions no té un impacte sobre la percepció que té l'usuari sobre aquest contingut. Malauradament, no és el cas quan ens trobem en un entorn tridimensional. De fet, és un procés complex on s'han de processar adequadament aquestes línies de text perquè no afectin la sensació d'immersió, ja que si aquests es mantinguessin estàtics mentre la càmera es troba en moviment, l'usuari podria arribar a perdre la sensació de profunditat de l'escenari.
- *Visualització.* Aquests vénen generalment de dues fonts. La primera pot ser present a tots els sistemes estereoscòpics, la diafonia. En definició, és que part del vídeo o de les imatges que s'haurien de projectar únicament sobre un dels dos canals, també pertorba l'altre. Un exemple serien unes ulleres de visió 3D a través de filtres de colors, on aquests filtres no poden separar correctament els espectres emesos per la pantalla. La segona font és present també en una gran part dels aparells, però aquesta pot variar en funció de cadascun. El motiu és que cada sistema té una manera òptima de ser utilitzat, i la trobem en un angle concret de visualització que pot variar depenent del visor.

Per últim, parlarem dels problemes que poden venir lligats al fet de no gaudir d'una visió binocular. Com a éssers humans disposem d'una visió binocular, i gràcies a ella la nostra forma de percebre el món és essencialment diferent a com ho fan els altres animals que tenen visió monocular. Tanmateix, existeix un gran nombre de persones que per desgràcia no posseeixen una correcta visió binocular per diversos motius, i el fet de no disposar d'ella provoca certes conseqüències, entre elles:

- *Aprenentatge amb més dificultats.* Comptar amb únicament un ull per a llegir, escriure o dibuixar pot arribar a exigir un gran esforç visual. Pot suposar un gran problema sobretot en infants, que a més a més encara es troben desenvolupant el sentit de la vista.
- *Percepció de l'espai i la distància distorsionades.* El fet de no comptar amb els dos ulls causa una manca de visió tridimensional. Per aquest motiu, la percepció de la distància serà una tasca complexa.
- *Disminució de l'agudesesa visual.* L'agudesesa visual és un paràmetre que ens permet avaluar la nostra salut ocular. Quan no utilitzem un dels dos ulls, aquesta es pot veure afectada negativament, podent arribar a repercutir a la nostra vida diària de manera important ja que, per exemple, per a conduir un cotxe necessitem tenir aquest paràmetre dins d'uns barems.

2.2 La realitat virtual

Com bé hem parlat abans, la història de la realitat virtual (també anomenada VR de l'anglès *Virtual Reality*) té uns inicis difícils de definir, i és que tot i haver tingut un desenvolupament certament significatiu en els últims anys amb l'aparició de les ulleres de realitat virtual comercialitzades, els prototips de màquines creades per a aconseguir el mateix objectiu porten existint desenes d'anys. Però abans de parlar dels aparells que ens permeten experimentar la realitat virtual, sabem realment de què tracta aquesta?

La podríem definir essencialment de la següent manera: Es tracta d'un espai tridimensional simulat per un ordinador on l'usuari és capaç d'explorar-lo i, en certs casos, interactuar amb els elements que s'hi troben allà. Per a poder tenir aquestes capacitats, l'ordinador ha de comptar amb un software i un hardware que siguin capaços de respondre als moviments de la persona que es troba utilitzant-lo, endinsant-la així en un espai alternatiu.

És important tenir clara aquesta definició, ja que a més de la realitat virtual, actualment existeixen dos conceptes més que, tot i tenir un nom molt similar, tenen certes diferències que els separen en diferents grups. Aquestes són la manera d'interactuar amb l'entorn i l'experiència de l'usuari. [Kov]

- 1) *Realitat Virtual*. Com hem esmentat abans, aquesta transporta a l'usuari a un nou entorn simulat completament diferent, que pot anar des d'estar a un submarí sota el mar, al terrat d'un edifici, o a punt de caure per una muntanya russa. L'usuari s'endinsa en aquest entorn ignorant l'espai físic sobre el que realment es troba i, de fet, és recomanable que aquesta experiència es realitzi a un lloc espaiós sense objectes ni persones a prop amb les que es pugui interactuar. D'aquesta forma, es pot evitar tant perdre la sensació d'immersió (al no rebre cap estímul del món real) com trencar quelcom al tenir la vista inhabilitada.
- 2) *Realitat Augmentada*. També coneguda com a AR (de l'anglès *Augmented Reality*), utilitza una càmera per a capturar l'entorn real on es troba l'usuari. Sobre aquest entorn, projecta models tridimensionals detectant la posició relativa de la càmera mitjançant sensors i algorismes. El mètode de visualització és una pantalla com pot ser la del nostre mòbil, pel que no necessitem cap tipus de material addicional com en el cas de la realitat virtual. Un bon exemple de realitat augmentada és un joc que va fer-se famós globalment, *Pokémon GO*, que utilitza aquest sistema per a projectar aquestes criatures imaginàries sobre el nostre món real.
- 3) *Realitat Mixta*. També coneguda com a MR (de l'anglès *Mixed Reality*), es pot considerar una combinació de les dues anteriors, ja que els objectes virtuals es poden vincular amb els elements del nostre entorn real. La seva aplicació és enfocada cap a formacions, simuladors, millora de la indústria, simulacions per a cirurgians... De fet, la realitat augmentada es pot considerar com a una categoria de la realitat mixta on els objectes projectats no tenen cap tipus d'interacció amb el món real.

Un cop hem entès què són i els motius pels quals es diferencien entre elles, focalitzarem la nostra atenció en entendre el funcionament de la VR.

Per començar hem de saber que, per a poder aconseguir el nivell de sensació d'immersió que busca la realitat virtual, l'usuari ha d'estar privat de qualsevol estímul visual provinent del món real. Per a fer-ho, els aparells de realitat virtual s'encarreguen de cegar-lo completament posant davant dels seus ulls una pantalla (o dues, una per a cada ull). Entre aquesta pantalla i els ulls es troben unes lents d'enfocament automàtic, que s'ajusten de manera constant d'acord amb els moviments oculars que l'usuari realitza. [Anu18]

En l'actualitat el mercat disposa d'un gran ventall de models d'ulleres. Podem trobar models que són capaços de funcionar de manera independent perquè compten amb processadors incorporats i, en certa forma, es poden considerar com a un mòbil dedicat a funcionar únicament per a les ulleres. Trobem també models molt econòmics on s'acobla el mòbil i únicament fan la funció de visor, ja que tot el processament és a càrrec del telèfon. De fet, aquest últim model és interessant pel fet que no suposa d'uns grans costos, però en essència és igual als convencionals. I és que la gran majoria disposen d'un cable HDMI amb el qual es connecten a un ordinador, i és aquest qui s'encarrega de renderitzar i enviar les imatges a les ulleres. La renderització és el procés amb el qual es genera una imatge mitjançant un model sigui 2D o 3D. (Parlarem amb més profunditat a l'apartat 2.3 Renderitzat)

Addicionalment, per a poder aconseguir una bona sensació d'immersió, hi ha certs requisits que s'han de complir. Primerament, la velocitat d'imatges per segon que la nostra GPU ha de ser capaç de processar. Aquesta és de 60 Fotogrames Per Segon (FPS).

En segon lloc, tenim la freqüència de refresc de la pantalla que projecta aquestes imatges sobre els ulls de l'usuari. Aquest valor significa el nombre d'imatges que mostra la pantalla per segon. Generalment, aquest és un valor que va dels 50Hz als 144Hz, però per a la realitat virtual les pantalles tenen un valor mínim de 60Hz, arribant a 90Hz en els models més cars. És molt important que aquests dos valors es trobin sincronitzats i constants. En cas de no fer-ho, hi ha riscos de mareig, mal de cap i, si l'exposició és massa llarga, és possible que l'usuari arribi a tenir nàusees i, fins i tot, pot arribar al vòmit. Si el refresc de la pantalla és massa lent, podem notar també que hi ha un retard entre els nostres moviments i els que veiem a la pantalla. Per a evitar-ho, el temps de resposta ha de ser de menys de vint mil·lisegons.

En tercer lloc, trobem el camp de visió que es presenta a la persona. El seu valor es dona en graus, i el podem definir com la quantitat d'escenari observable que tenim davant. Aquest, de la mateixa manera que ho fa la freqüència de refresc, varia en funció al model alhora que augmentant en funció a l'augment de preu, variant el seu valor des de 90 fins a 110 graus.

En quart i últim lloc, tenim la resolució. Tot i que una resolució no sigui totalment perfecte pot arribar a ser una molèstia si és molt baixa. Per aquest motiu s'ha de procurar treballar amb resolucions adients a la que té la pantalla del nostre dispositiu.

A part d'aquests requisits fonamentals, hi ha elements addicionals dissenyats per a millorar expressament l'experiència amb aquests dispositius, com poden ser:

- *Seguiment de la vista.* Aquesta és una característica amb la qual disposen les ulleres amb preus més elevats. Per a fer aquest seguiment, incorporen uns infrarojos que s'encarreguen de seguir els moviments de l'ull, permetent així al dispositiu adaptar millor l'entorn virtual.
- *Seguiment dels moviments del cap.* Aquest seguiment s'ha de realitzar de manera precisa i ràpida per tal que ocorrin simultàniament al món on es troba l'individu. Per a aconseguir-ho, s'assignen uns eixos als moviments que pot realitzar el nostre cap, i s'incorporen al dispositiu sensors com ara un acceleròmetre o un giroscopi.
- *Seguiment dels moviments de tot el cos.* Quan es té un sistema capaç de fer seguiments tant del nostre cap com de tot el nostre cos, podem dir que ha assolit el que es coneix com a sis graus de llibertat. Els graus de llibertat són les maneres en què un objecte es pot moure en un espai tridimensional. El nostre cos es pot moure cap endavant i enrere, amunt i abaix i de dreta a esquerra. Quan ho combinem amb els moviments que podem fer amb el cap, veiem que podem afegir una rotació sobre cadascun dels eixos, arribant així al màxim de sis graus de llibertat. [Goo18]
- *Efectes sonors.* Aquests poden arribar a millorar molt l'experiència, especialment si es poden utilitzar auriculars que t'aïllin de qualsevol soroll extern. D'aquesta manera, tampoc es poden rebre estímuls auditius del món real, i si els efectes sonors van perfectament sincronitzats amb allò que la persona està experimentant visualment, es pot arribar en major grau a experimentar una realitat alternativa més completa.

2.2.1 L'evolució de la realitat virtual

A continuació, veurem un breu resum dels inicis i creixement amb el pas dels anys de la realitat virtual fins a arribar als dispositius actuals. Tots ells han estat dissenyats d'acord amb els principis de l'estereoscòpia i l'estereòpsia que hem estudiat prèviament. A la figura 2.2 podem veure un resum ràpid amb els elements més importants a destacar des de 1935 fins a 2018, dels que en comentarem una selecció. A [Poe19] i [Coo17] podem trobar una explicació més extensa de tots els avenços amb el pas dels anys, incloent-hi d'alguns no tan rellevants que no trobem a la figura.

Quan parlem dels inicis de la realitat virtual podem dir que tenim dos referents històrics, Morton Heilig i Ivan Sutherland. Heilig va crear per primer cop el que podem considerar la primera màquina de realitat virtual l'any 1957 (encara que no va ser patentada fins al 1962). Aquesta, però, difereix en gran escala del concepte que tenim avui dia, i és que no es tractava d'un dispositiu que es col·locava sobre el cap, sinó que era tota una cabina.

No va arribar gaire lluny, doncs era realment un projecte molt ambiciós al no voler només submergir a l'usuari en una experiència visual, sinó que a més de les imatges tridimensionals, també s'ajudava d'estímuls auditius, olors, vibracions, i fins i tot una cadira que et permetia girar. Ell mateix va ser també qui, l'any 1960, va desenvolupar el primer dispositiu de realitat virtual en forma d'ulleres que s'agafaven al nostre cap, dispositius

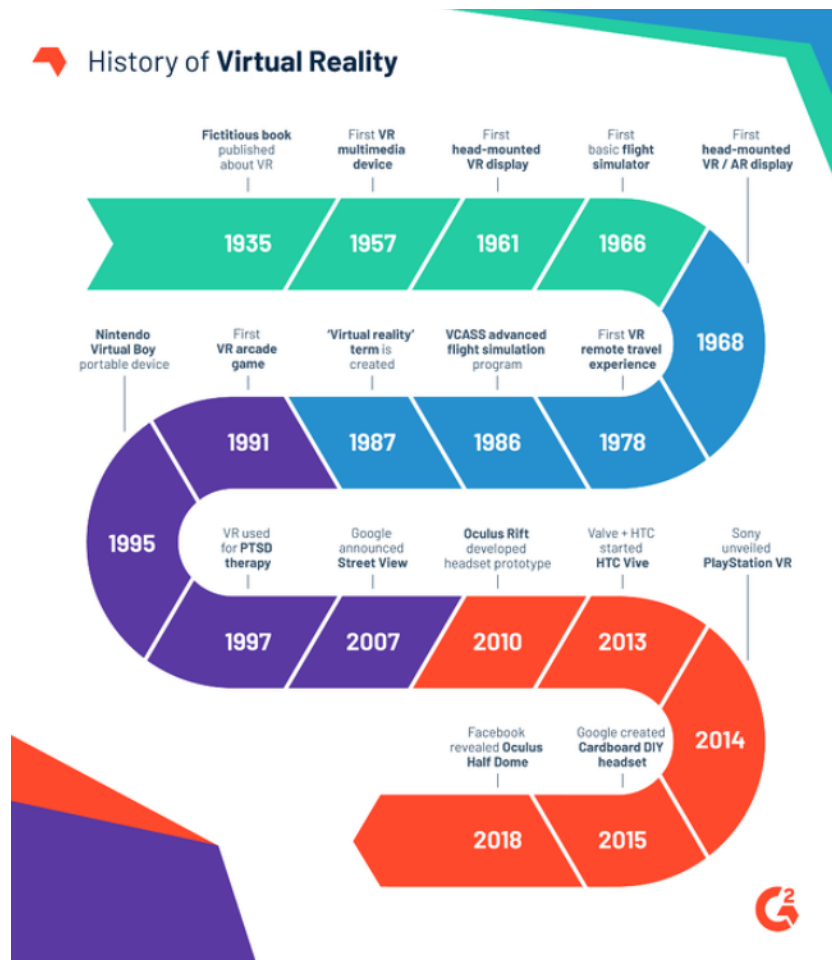


Figura 2.2: La realitat virtual amb el pas dels anys. Font: [Poe19]

coneguts com a HMD (*Head Mounted Display*). Un any després, el 1961, l'empresa *Philco Corporation* va crear el primer HMD capaç de fer seguiment dels moviments del cap de l'usuari.

Al 1965, Sutherland va presentar públicament un nou concepte al que va anomenar *The absolute display*. En les seves paraules:

«The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming such a display could literally be the Wonderland into which Alice walked.»

El que volia era exposar el seu concepte del què seria la realitat virtual en uns anys, on tindriem un ordinador capaç de controlar totalment el món on ens trobem, significant

això que podríem interactuar amb la matèria d'aquest i viure una experiència propera a la coneguda pel·lícula *Alícia en terra de meravelles*. De fet, aquesta definició ha servit com la base que ha portat a la indústria fins on la podem trobar ara.

Tres anys més tard, amb l'ajuda del seu estudiant Bob Sproull, va presentar el que seria el primer prototip d'HMD de realitat virtual i realitat augmentada, realitzat a la Universitat Harvard. Aquest era un artifici de grans dimensions sospès del sostre i connectat a un ordinador, i a causa del seu pes i la impossibilitat d'utilitzar-lo sense que estigués penjat, va rebre el nom de *l'Espasa de Dàmocles*. A la figura 2.3 podem veure les grans dimensions del dispositiu, així com podem veure al fons de la fotografia l'ordinador al qual estava connectat, també molt voluminós. A [Sut68] podem trobar la publicació original del projecte d'Ivan.

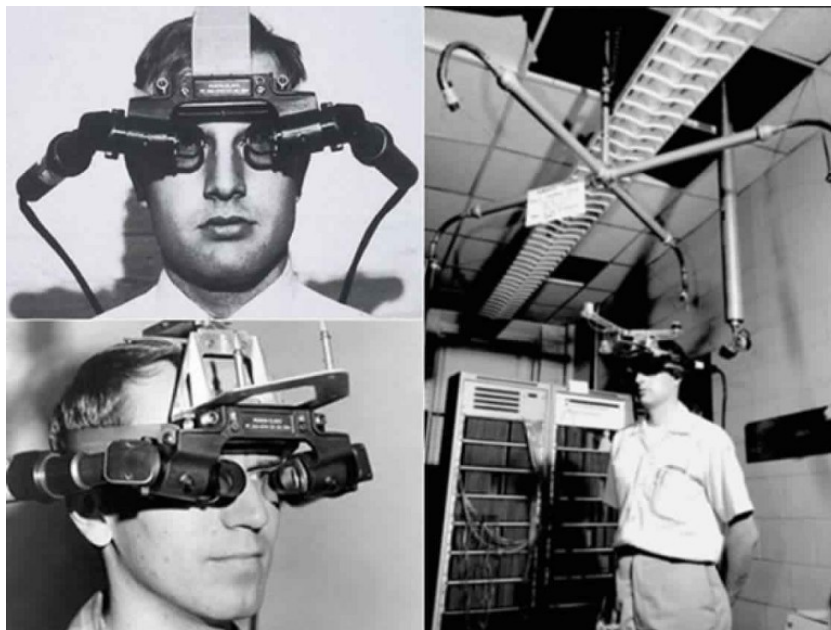


Figura 2.3: L'espasa de Dàmocles

Als següents anys es va seguir treballant sobre aquest camp, amb avenços com ara el simulador anomenat *The Super Cockpit*, dissenyat per a l'entrenament de pilots d'aviació. No va ser, però, fins al 1987, quan va néixer el nom de "Realitat Virtual" en mans d'un científic anomenat John Lanier. Lanier també va contribuir activament al desenvolupament d'aparells per a la VR, creant la primera companyia que va presentar aquests productes al mercat.

Un cop aquest sector va arribar al públic, altres empreses van començar a desenvolupar els seus propis dispositius durant els anys noranta. Van sorgir locals amb màquines recreatives (també conegudes com a arcades), així com una nova temàtica als cinemes, la qual va suposar una novetat al introduir a l'espectador dins un nou món prèviament desconegut, amb pel·lícules com per exemple *Matrix*. Una altra aplicació que hem trobat molt interessant amb VR, és la d'un simulador que et portava a Vietnam. El seu objectiu

era el de poder tractar els pacients exmilitars amb traumes, posant-los a situacions com ara dins un helicòpter. Aquesta és una aplicació que considerem molt interessant, i que avui en dia podria aplicar-se per al benefici de moltes persones.

Arribem així a l'actualitat, on fa deu anys del llançament del primer prototip que coneixem avui dia com a *Oculus Rift*. Aquest va ser dissenyat per Palmer Luckey, qui dos anys més tard va llançar una campanya a la plataforma *KickStarter* per a subvencionar el seu projecte i dedicar-se a ell completament.

2014 va ser un any molt important per a la indústria, ja que per una banda, la companyia *Facebook* va comprar *Oculus*, el que significava que el pressupost per al desenvolupament seria molt més elevat. Per altra banda, va haver-hi més companyies que també van presentar els seus projectes.

Google, per la seva part, volia apropar aquesta tecnologia al públic que no disposava de suficients diners per a comprar no només l'HMD d'*Oculus*, sinó també un ordinador suficientment potent per a executar correctament les aplicacions sobre aquest. Així va néixer *Google Cardboard*, que és un *kit* que porta el necessari per a construir un visor de realitat virtual de manera casolana. En aquest visor s'hi situa el mòbil, i sobre aquest podem tant veure vídeos com obrir aplicacions que ens divideixen la pantalla en dos, aconseguint així visualitzar una imatge estereoscòpica.

Alhora, *Sony* presenta el que ara coneixem com a *PlayStation VR*, tot i que no és fins a 2016 que surt oficialment al mercat. Aquesta és una altra alternativa a les *Oculus* que funciona amb una consola com a ordinador, fet pel que a tothom que ja disposés d'una se li obre una possibilitat de poder gaudir d'aquesta tecnologia.

En l'actualitat, *Oculus* ha continuat llançant nous models al mercat de diferents preus i característiques, i altres companyies com ara *HTC*, que va presentar el seu model *HTC Vibe* el 2015, s'estan afegint també a la iniciativa. Si tot segueix així, en un futur podrem disposar d'un ventall de possibilitats més gran que l'actual, amb un mercat on podrem trobar més varietat de preus i uns dispositius més eficients.

2.3 Renderització

Aquesta és una paraula que hem incorporat al nostre idioma de l'anglès *rendering*, que en certa forma podem interpretar com a *representació*. En general, és un terme relatiu a la computació gràfica, i tracta del procés de modelat d'una imatge amb l'ajuda d'un *software*. Sota aquest terme podem trobar diferents subapartats, com per exemple el renderitzat d'arquitectura, però en aquest treball ens centrarem en el renderitzat 3D.

Unity defineix el renderitzat 3D de la següent manera: [Unic]

«El renderitzat 3D és el procés de producció d'una imatge basada en dades tridimensionals emmagatzemades al teu ordinador. També és considerat un procés creatiu, de la mateixa manera que ho són la fotografia o la cinematografia, ja que fa ús de la llum i finalment produeix imatges.»

Aquestes *dades tridimensionals* no són més que la informació sobre l'escena 3D que

estem renderitzant: Polígons, punt de vista, il·luminació, i materials.

L'ordinador és capaç de processar totes aquestes dades, calculant, per exemple, la dimensió i la forma de l'ombra d'un objecte a l'espai tenint en compte la posició i intensitat de la/les fonts de llum que tinguem.

Depenent de la complexitat de l'escena i dels models que hi trobem en ella, un renderitzat pot trigar des de pocs segons fins a hores o dies. En cas de renderitzar una animació, hem de considerar que per a cada segon haurem de realitzar el procés de renderització un cop per a cada fotograma.

Degut a que aquest procés té un cost molt elevat al nostre processador, també anomenat CPU (*Central Processing Unit*), els ordinadors solen incorporar una targeta gràfica, també anomenada GPU (*Graphics Processing Unit*) que està especialitzada en realitzar tots aquests càlculs. Aquesta funciona paral·lelament a la CPU, i la seva principal diferència és que la CPU sol tenir un nombre de nuclis de processament relativament reduït, que li permeten realitzar càlculs seqüencials i processos individualment de manera molt efectiva. Una GPU, en canvi, disposa d'una quantitat més elevada de nuclis (tot i que més reduïts) que poden realitzar de forma més efectiva més d'una tasca simultàniament.

Per aquests motius, podem distingir dos tipus de renderitzat:

- 1) *Prerenderització*. Aquest és el nom que rep el procés de renderització realitzat per la CPU. Malgrat que una GPU sigui capaç de realitzar molts més càlculs al ser capaç d'executar-los paral·lelament, un renderitzat amb CPU sempre és més fidel a l'original. Per aquest motiu, si no estem treballant a temps real, aquesta pot ser una millor opció. De fet, les pel·lícules o certes parts de videojocs com ara els escenaris que trobem al fons es renderitzen amb CPU. És possible també realitzar tècniques més complexes que donen resultats encara més realistes. A més, existeixen tècniques que permeten dividir el treball entre diverses CPUs, accelerant així la velocitat i obtenint el mateix resultat en menys temps.
- 2) *Renderització a temps real*. Aquest és el nom que rep el procés de renderització realitzat per la GPU. Aquest tipus de renderització s'utilitza de manera principal als videojocs, on tenim un personatge movent-se per un espai de manera constant, el que repercuteix en tenir la font de llum canviant de posició constantment, veure els objectes des d'un altre punt de vista... Degut a la velocitat necessària per a fer aquests processos constants, es justifica la pèrdua de qualitat del renderitzat enfront la de la prerenderització, on no tenim límit de temps.

A [Poo] podem trobar un article comparatiu sobre les diferències entre els dos renderitzats, i més informació sobre quines són les tècniques que es poden utilitzar amb la CPU a [Den].

2.3.1 Graphics pipeline

Finalment introduïrem un nou terme relatiu al renderitzat 3D, i és la *Graphics Pipeline*, també coneguda com a canonada de renderitzat. Aquest terme es refereix a tots els passos que es realitzen a la renderització amb targeta gràfica. Depenent de l'estàndard, aquesta

pot variar lleugerament, però essencialment el que fa és: ubica els models que tenim a l'espai, aplica la il·luminació sobre aquests, es selecciona una posició que representa la de l'observador i es *mapeja* des d'aquest punt de vista l'escenari tridimensional sobre un pla.

Fet això, es pot descartar la informació sobre les parts que no es troben a la vista. El resultat serà el mateix, però en cas d'eliminar-les evitem que els següents passos que veurem també s'apliquin sobre elles, incrementant la velocitat de renderitzat.

A continuació s'aplica la texturització sobre els models. Per a entendre aquest concepte veiem un exemple: Tenim un escenari on veiem una taula marró situada en una petita sala amb parets blaves, i volem renderitzar aquesta escena. Fins que no s'arriba al pas de la texturització, els altres passos del renderitzat es fan considerant tant les parets de la sala com la pròpia taula com a únicament formes geomètriques simples, i és en aquest pas on s'apliquen els colors, o textures, sobre aquestes. En fer-ho, aquestes formes geomètriques definides a l'espai prenen color i es defineixen, tot tenint en compte també la informació sobre com els hi arriba la llum, que s'ha calculat en un pas previ.

Per últim lloc es trama la imatge, és a dir, es converteix en un conjunt de píxels que coneixem com a imatge rasteritzada o bitmap, i aquesta imatge és la que podem mostrar finalment per pantalla.

2.4 Transcodificació

Podem definir la transcodificació com el procés que apliquem sobre un fitxer de vídeo ja codificat. Aquest és descodificat (tot i que no completament, sinó que es deixa en un pas intermedi), i més tard tornat a codificar de nou. Aquest procés s'aplica quan volem desplaçar d'un lloc a un altre el fitxer en qüestió, per exemple, a l'enviar-lo a un destinatari, o per a canviar el format de compressió en què es troba el vídeo.

Per a realitzar aquest procés requerim un còdec. El seu nom prové de la combinació de les síl·labes de les paraules angleses *encoder* i *decoder*, que signifiquen codificador i descodificador respectivament. Un còdec pot ser qualsevol mena de SW o HW que pugui realitzar les dues tasques esmentades prèviament, que en essència volen dir comprimir el vídeo perquè aquest ocupi molt menys espai i es pugui transportar i emmagatzemar millor, i descomprimir-lo per a poder-lo visualitzar.

Podem trobar diferents estàndards de compressió de vídeo depenent del tipus de còdec utilitzat. De fet, també s'utilitza aquest terme per a definir l'estàndard de compressió que utilitza un codificador o descodificador de vídeo.

Existeix una àmplia varietat de còdecs en l'actualitat, els motius són deguts a què cadascun d'ells dona millors resultats en una àrea concreta, a més d'haver-se creat de nous que essencialment són versions millorades d'altres ja existents. Entre aquests, podem distingir entre dos tipus, els còdecs amb pèrdua (*lossy*) i sense pèrdua (*lossless*). La principal diferència entre ells és bàsicament que els *lossless* comprimeixen en menor mesura els arxius, permetent així una recuperació més fàcil de l'arxiu original a canvi d'un pes de fitxer més elevat, i s'utilitzen principalment quan es vol assegurar la seva possible reconstrucció exacta.

A continuació, en comentarem tres d'ells (podem trobar més informació i diferents còdecs a [Nik]):

- 1) *MPEG-2*. Aquest és el pioner quan parlem de la codificació de vídeo per a DVDs o la televisió digital. Encara es segueix utilitzant per a casos específics, però actualment ja té 30 anys, el que fa que es quedi enrere respecte als seus successors respecte a l'algoritme de compressió.
- 2) *H.264 / AVC (Advanced Video Coding)*. En l'actualitat, s'estima que aquest còdec és l'utilitzat per la majoria de vídeos que circulen per la xarxa. S'ha convertit en l'estàndard de compressió per als vídeos en alta qualitat, arribant a suportar fins i tot el 4K gràcies a les millores que s'han incorporat amb el pas dels anys. És un estàndard molt flexible que permet la seva aplicació en tota classe d'aplicacions i sistemes. Malgrat això, el següent còdec és una millor opció quan parlem de retransmissions en qualitat molt elevada.
- 3) *H.265 / HEVC (High Efficiency Video Coding)*. Es tracta d'una versió millorada de l'anterior còdec, al ser capaç de comprimir els vídeos fins al doble que ho fa l'H.264, el que vol dir que podem enviar la meitat de bits i tenir la mateixa qualitat. És per aquest motiu que s'utilitza per a les tecnologies 4K i Blu-Ray. Per a aconseguir-ho, emprà una de les tècniques que veurem més endavant, basada en la predicció.

Aquests dos últims algoritmes tenen capacitat de reduir les dimensions d'un fitxer fins a mil vegades. És per aquest motiu que als últims anys hem vist un gran creixement de plataformes de retransmissió de vídeo, com ara *Twitch*, ja que gràcies a aquests avenços és possible gaudir d'aquests serveis tot i no tenir una connexió potent, alhora que permet als espectadors veure pràcticament a temps real el contingut desitjat.

2.4.1 Bitrate

Entenem com a *bitrate* (o taxa de bits) el nombre de bits per segon que processem, i com podem suposar, com més alt sigui aquest valor, més informació serem capaços de rebre. Aquest és un factor molt important, ja que en cas de no tenir un valor suficientment alt podem perdre part de les dades pel camí.

Cada segon a un vídeo està representat per una quantitat determinada de fotogrames (generalment 24), i aquests tenen la seva informació representada per un nombre de bits. Per tant, si el nostre *bitrate* no és suficientment alt, no serà possible llegir tota la informació. En passar això es genera el que anomenem redundància, és a dir, com que no es té la informació respecte a tots els píxels, s'agrupen diversos amb la mateixa informació, perdent així qualitat al vídeo.

És interessant comentar que en l'actualitat existeix el que es coneix com a retransmissió amb *bitrate* adaptatiu. Aquesta és una tècnica que ha sorgit gràcies a les tècniques de compressió de vídeo que veurem a continuació, i permet ajustar la qualitat d'una retransmissió d'acord amb la capacitat de la CPU i l'amplada de banda del que es disposa al moment. D'aquesta manera, en cas de no tenir suficient capacitat en un determinat moment, es veurà afectada la qualitat del vídeo però s'evitarà un tall.

2.4.2 Compressió de vídeo

Com hem pogut veure, la compressió de vídeo és essencial. A continuació veurem un exemple amb uns petits càlculs per a veure quant pesaria un arxiu sense comprimir. Per a realitzar-los, ens hem ajudat de la pàgina [For], que ens permet calcular el pes que tindriem per fotograma en un vídeo de les característiques especificades. En aquest cas, un vídeo amb resolució 1920x1080 gravat a 24FPS.

Video filesize calculator

Dimensions: 1920 x 1080

Framerate: 24fps (cinema)

Time duration: 5 seconds

Estimate video size!

Frame file size

- **Megapixels: 2.1 MP (2,073,600 pixels)**
width in pixels x height in pixels
- **Aspect ratio: 1.78**
width in pixels / height in pixels
- **Uncompressed monochrome frame (Bayer masked): 3.11 MB (2.1 MP x 12 bit/pixel)**

Figura 2.4: Càlcul de pes per a cada fotograma sense compressió d'un vídeo

A la figura 2.4 podem veure que el pes per a cada fotograma serà de 3.11MB, i tenim un vídeo de 5 segons amb 24 fotogrames per a cada segon. Per tant:

$$3,11 \cdot 24 \cdot 5 = 373,2$$

Com podem veure, un pes de 373,2MB per a un vídeo de 5 segons amb una qualitat com a la que ens trobem acostumats, és un pes que avui en dia mai ens podríem imaginar. Si fem el mateix càlcul considerant una pel·lícula de dues hores:

$$3,11 \cdot 24 \cdot 60 \cdot 120 = 537.408$$

Tindríem un pes de més de mig Terabyte per a un únic vídeo, el que seria l'equivalent a més de la meitat de la capacitat actual de molts discs durs, el que és totalment inviable.

Tècniques de compressió de vídeo

Ara que sabem els avantatges que ens aporta la compressió de vídeo, veurem dues de les tècniques més importants, entre elles, la tècnica de predicció que utilitza el còdec H.265 [Rom18]:

- *Redimensionament de la imatge.* Aquesta és una tècnica que ha afavorit molt el desenvolupament del *bitrate* adaptatiu, i com bé diu el seu nom, el que es fa és alterar la resolució del vídeo. Quan parlem d'una resolució de vídeo volem dir el nombre total de píxels que representen cada fotograma, per exemple, una resolució molt comú és la de 1920x1080. Com veiem, aquest valor es dona en funció al total de píxels en amplada i altura, pel què si realitzem el càlcul podem trobar el total, en aquest cas, de 2.073.600 píxels. Si redimensionem aquesta imatge a una altra resolució més baixa, tindrem un menor número de píxels, i, per tant, necessitarem menys bits per a poder-la representar. S'ha de tenir cura, ja que amb aquest mètode podem tenir petits defectes a la imatge i que no es representi correctament en algunes parts.
- *Inter-frame prediction.* Abans hem parlat sobre com el còdec H.265 era capaç de reduir les dimensions d'un vídeo gràcies a tècniques de predicció. Per a entendre com funcionen, veurem un exemple i parlarem sobre els tres tipus de frame que podem tenir. A un vídeo com el que hem parlat abans, tenim 24 fotogrames representant cada segon. Si ens imaginem que aquest vídeo ha estat gravat a una platja, i en ell veiem la sorra, el mar i el cel, ens podem adonar que totes tres parts tenen una cosa en comú, i és que, en general, es mantenen estàtiques. A la sorra hi podem trobar persones caminant, però quan no hi hagi ningú, aquesta estarà quieta (suposant que no hi hagi gaire vent, és clar), la mar romandrà igual a excepció de la riba en arribar les onades, i els moviments dels núvols que hi pugui haver al cel no serà suficient com per a variar 24 cops en un segon. Per tant, ens podem estalviar donar informació sobre totes aquestes parts estàtiques, centrant-nos així sobre la part de la pantalla on hi hagi moviment. És aquí on trobem els tres tipus de fotogrames: En primer lloc trobem els *I-Frames*, o *Keyframes*, que són fotogrames amb la totalitat de la seva informació, i especialment són necessaris en l'inici del vídeo i a canvis d'escena, moments on es necessita tota la informació del que es mostrarà per pantalla per primer cop. En segon lloc tenim els *P-Frames*, o *Predictive frames*, fotogrames que només contenen una part de la imatge. Per a crear-los, es mira el frame anterior i es compara quines parts són les mateixes per a evitar haver de codificar aquesta informació, necessitant així menys bits per a representar-los. En tercer i últim lloc, tenim als *B-Frames* o *Bi-directional predictive frames*. Funcionen de la mateixa manera que ho fan els *P-Frames*, però aquests també son capaços de mirar el seu següent fotograma a més de l'anterior. D'aquesta manera, es pot reduir encara més la quantitat de bits per a representar-los, ja que poden descartar encara més píxels a codificar. Aquests últims, però, necessiten d'un *encoder* especialitzat per a poder-los crear.

2.5 Càmeres IP

Les càmeres IP (de l'anglès *Internet Protocol*) són un tipus concret de càmera relativament nou. Aquestes reben el seu nom gràcies a la seva capacitat d'accedir a internet de manera independent a un ordinador, ja que consten amb un petit microordinador

personal i es connecten a l'alimentació. El seu funcionament respecte a la captura de vídeo és idèntic al d'una càmera normal, però aquesta té incorporat tot el necessari per a poder comprimir i transmetre el vídeo (també té la seva pròpia adreça IP) tant per ethernet com per WiFi o USB.

Aquestes càmeres van sorgir principalment amb una intenció industrial, però degut a les característiques que ofereixen s'han popularitzat cada cop més com a una forma de protegir la nostra llar. Essencialment es divideixen en dos grups:

- *Centralitzades.* Generalment s'adquireixen més d'una alhora, i es connecten a un sistema de videovigilància amb un programa que rep el nom NVR, de l'anglès *Network Video Recorder*. Aquest programa és capaç d'emmagatzemar i reproduir totes les senyals de vídeo de les càmeres connectades, però li han d'arribar codificades ja que no té capacitat de fer-ho.
- *Decentralitzades.* Aquestes càmeres tenen una interfície pròpia, alhora que tenen també capacitat d'emmagatzematge pels seus propis vídeos. És comú també que tinguin una petita ranura per a poder-hi afegir una targeta de memòria que ens permeti expandir la seva capacitat total.

Podem trobar més informació a [Sec] i [Inc16]

2.6 RTSP

RTSP prové de l'anglès *Real Time Streaming Protocol*. Estem parlant d'un dels molts protocols que podem trobar actualment a la xarxa. Concretament, aquest protocol és utilitzat pel client, i s'encarrega de controlar el flux de dades d'àudio i vídeo entre ell i el servidor. Per a aconseguir-ho, combina els protocols TCP, per a parlar amb el servidor, i UDP, per a l'entrega dels arxius multimèdia que s'han demanat.

No obstant això, no és a través de la connexió RTSP per on s'envien aquestes dades, sinó que s'utilitza per a enviar comandes al servidor, tot especificant quin contingut es vol o donant ordres. En certa forma, podem fer una analogia amb un comandament a distància d'un televisor, sent aquest últim el servidor, i el client la persona que utilitza el comandament.

És gràcies a tots els avantatges que comporta aquest protocol, que la gran majoria d'*IP Cams* l'utilitzen per a enviar la seva informació. Per a més informació sobre aquest protocol, podem consultar el llibre [Mat03].

2.7 WebRTC

WebRTC és un projecte de codi obert que reb el seu nom de l'anglès *Real Time Communication*). Aquest té com a objectiu proporcionar una forma de comunicació a temps real entre qualsevol dispositiu capaç d'accedir a la xarxa, permetent que es comuniquin entre ells mitjançant una sèrie de protocols estandarditzats.

Anys enrere, aquestes comunicacions eren molt complexes i requerien tecnologies de vídeo i àudio amb preus elevats, però actualment disposem d'elles a tots els navegadors moderns. En certa forma, WebRTC és un conjunt de moltes interfícies de programació d'aplicacions (també conegudes com a APIs), que ofereixen des de petites aplicacions, com ara rebre l'àudio d'un micròfon, fins a connexions *Peer-to-Peer* entre nodes.

A la seva web oficial, [Goo], podem trobar una sèrie de petites mostres d'APIs, així com el codi font de totes elles al repositori de *GitHub*.

2.8 Unity

En essència, *Unity* és un motor de videojocs que permet el seu desenvolupament per a diferents plataformes, des de videoconsoles fins a ordinadors o dispositius mòbils. Consta amb un entorn integrat de desenvolupament (també conegut com a IDE, de l'anglès *Integrated Development Environment*), que ens permet veure allò que estem creant, a més de poder fer tots els tests necessaris gràcies a la seva funció de *Play*, que reproduïx el videojoc des d'aquest propi entorn. Addicionalment, té integrat un editor de codi, però també et permet utilitzar d'altres. (Si es realitza la configuració prèviament, és clar)

En l'actualitat és reconegut globalment, ja que ofereix una grandíssima quantitat d'avantatges als desenvolupadors. A més, és totalment gratuït en cas d'utilitzar-lo individualment, excepte si els ingressos que es generen superen els 100.000 dòlars americans anuals. Aquest límit difereix en cas de ser una empresa, que pot utilitzar el programa de franc mentre no superin el límit de 200.000 dòlars. En cas de superar els ingressos esmentats prèviament, s'ha d'utilitzar la versió *Unity Pro* (amb un cost de 1800 dòlars anuals per a cada llicència), o la versió *Unity Enterprise* (200 dòlars mensuals per llicència, però amb un mínim de 20 contractades), ja que aquestes no tenen un límit d'ingressos. [Unia]

Tot seguit, veurem un llistat dels avantatges que ens ofereix aquest motor, els quals fan de *Unity* una molt bona opció qualitat-preu per a tot tipus de desenvolupadors i programadors. [Pol13]

- Permet treballar d'una manera molt còmoda alhora que intuïtiva, amb un sistema de *Drag and Drop* incorporat. A *Unity*, tenim per una part els *assets* (o *Game assets*), terme que fa referència a qualsevol recurs que sigui emprat en un videojoc, ja sigui una imatge, un model 3D, un fitxer d'àudio.... Per l'altra tenim el codi escrit. Per a poder vincular aquest codi, que ens diu el que volem que faci l'*asset*, amb aquest mateix, ho podem fer arrossegant el fitxer on tenim l'*script* sobre el propi asset. De la mateixa manera, també permet fer-ho a la inversa. Per a entendre-ho millor, podem veure l'exemple que trobem a [Pol13]: Suposem que tenim un *script* que fa que un personatge canti, un objecte personatge (que pot ser per exemple un model 3D), i un altre objecte instrument. El programa ens permet simplement arrossegar aquest *script* sobre el personatge per a vincular-los, i podem arrossegar l'instrument sobre aquest codi que hem assignat per a indicar quin és l'instrument a tocar. Aquestes pràctiques són molt útils en petits o

mitjans projectes, però hem de tenir cura en cas d'estar treballant en un de gran, on es recomana treballar amb mètodes menys orientatius (tot i que es pot seguir utilitzant aquest mètode per a *debugging* o fer petites proves còmodament).

- *Unity* tracta a tots els seus elements com a components. La definició donada per Margaret Rouse [Rou] ens pot ajudar a entendre millor què vol dir un component en l'àmbit de la programació orientada a objectes:

«(...) A component is a reusable program building block that can be combined with other components in the same or other computers in a distributed network to form an application.»

Si la traduïm, ens diu que un component és un bloc de codi reutilitzable que, combinat amb d'altres, arriba a formar una aplicació completa. Això aporta una gran llibertat a l'hora de treballar, ja que independentment de la quantitat d'*assets* que afegim a un objecte, l'essència d'aquest mai variarà. Aquesta metodologia ha estat triada per sobre de l'heretatge (que és un altre mètode emprat) per a facilitar l'ús del programa a tots els públics. A [Low15] podem veure un article elaborat sobre l'heretatge enfront de la composició.

- Està dissenyat de manera que permet treballar còmodament amb altres aplicacions que solen ser utilitzades en aquest sector, com per exemple, programes de modelatge 3D com ara *Cinema 4D*, *Maya* o *Blender*. Quan tenim importat un model 3D de qualsevol d'aquests programes sobre el nostre projecte, podem editar-lo amb tranquil·litat ja que, en el moment de guardar-lo, *Unity* el re-importa instantàniament i actualitza el projecte amb els canvis realitzats.
- Quan estem treballant sobre el projecte, és important saber que l'entorn de desenvolupament té una característica molt interessant. Com bé hem dit abans, tenim un botó de *play* que ens permet provar el videojoc (o aplicació en el nostre cas), en qualsevol moment. Quan entrem en aquest mode, podem canviar l'estat de qualsevol variable per a poder fer les proves que necessitem, ja que en sortir, tots els canvis es reverteixen i desapareixen, tornant totes les variables al seu estat previ.

Els creadors han desenvolupat també una eina molt útil per a poder treballar amb diferents projectes i versions. El nom d'aquesta és *Unity Hub*, i permet gestionar múltiples instal·lacions de diferents versions de l'editor de *Unity*. Actualment la versió més actualitzada és la 2020, però amb aquesta eina podem treballar amb quina vulguem. D'aquesta manera, podem tenir diferents projectes realitzats amb una versió diferent cadascun sense preocupar-nos, ja que no es pot donar una situació on actualitzem l'editor i deixem de poder treballar sobre el nostre projecte. (Podem trobar tota la informació respecte la seva instal·lació a [Unib])

Veiem finalment una última característica molt important sobre *Unity*, l'anomenada *asset store*. És una botiga virtual on podem trobar tant *assets* com components sencers

que poden aportar una solució a un problema. Els articles que es poden trobar tant poden ser de franc com tenir un preu establert, tot depèn de la decisió de la persona o equip que l'ha creat.

2.9 ADB

Android Debug Bridge, o ADB, és una eina emprada per a les comunicacions entre diferents dispositius, com en el nostre cas, l'ordinador i les ulleres de realitat virtual. Com podem veure a la seva documentació oficial [dev20], el programa consta de tres components:

- *Client*. Executat a la màquina on desenvolupem el treball, i encarregat d'enviar les comandes cap al dispositiu.
- *Daemon (o servei)*. S'encarrega d'executar les comandes al dispositiu, i és executat en segon pla.
- *Servidor*. S'executa en segon pla a la màquina on desenvolupem el treball, serveix per a administrar la comunicació client-*daemon*.

Aquest programa sol comunicar-se per via USB, però ofereix també alternatives que permeten utilitzar una xarxa sense fils. En el nostre cas treballarem amb cable, i hem de tenir en consideració també que prèviament haurem d'haver configurat les ulleres, activant a les opcions de desenvolupadors la depuració per cable USB.

3 Plantejament del projecte

Quan ens enfrontem a un projecte amb un alt grau de dificultat com ara és aquest, les decisions preses des d'un inici poden definir el desenvolupament del mateix completament. Per aquest motiu, valorarem un ventall de possibilitats en un inici i escollirem un primer camí, però en tot moment és important considerar altres opcions. De fet, hem de recordar sempre que mai ens hem d'aferrar a una única opció, ja que al començar a implementar-la podem veure que té problemes que porten a una inviabilitat de forma inevitable.

3.1 Possibles mètodes d'implementació

Per a poder valorar aquest projecte, el primer que farem serà analitzar-lo des d'un punt de vista primerament superficial i anirem entrant en profunditat a mesura que avancem. Si ens parem a analitzar, per exemple, quin material necessitarem en essència per a dur a terme el projecte, podem destacar certs elements clau:

- Una (o més) càmeres.
- Un ordinador amb una targeta gràfica.
- Ulleres de realitat virtual.

Veiem que haurem de fer una cerca i valoració sobre els models de càmera que ens poden servir, ja que podem tenir vies diferents depenent de la càmera que escollim, i repetir el procés per a les ulleres i la targeta gràfica.

3.1.1 Valoracions de càmeres

Escollir la càmera (o model de càmeres) que utilitzarem pot arribar a complicar-se, per tant el que farem serà fer-nos a nosaltres mateixos les següents preguntes per a intentar orientar-nos:

- 1) Volem una càmera amb cable o una sense fils?
- 2) Si és sense fils, de quina manera funciona?
- 3) Quina resolució i FPS té la càmera?
- 4) Té un preu assolible?

El primer que hem de decidir és com volem que funcioni la càmera. Un funcionament per cable com el d'una *webcam* ens pot ser molt còmode a l'hora de fer funcionar la càmera i rebre les imatges a l'ordinador, però aquesta comoditat ve lligada a certes limitacions.

La primera d'elles és que, evidentment, podrem únicament desplaçar la càmera en un radi igual a la longitud del seu cable, per tant no es podrien dur a terme certs muntatges que es poden arribar a construir (com els que veurem més endavant a l'apartat 6 Treball futur), si es treballa sense tenir de forma obligatòria un cable connectat a un ordinador. La segona és el fet que la càmera està lligada a un PC per a poder funcionar i hauríem de buscar la manera d'enviar aquesta senyal de vídeo a les ulleres.

Aquesta limitació pot ser eliminada en escollir com a càmera una *IP Cam*. Com vam veure a l'apartat 2.5 *IP Cams*, aquestes poden funcionar de manera independent al disposar d'un petit ordinador incorporat, i tenen una adreça IP assignada que ens permet un accés a la seva retransmissió fàcilment. Per aquest motiu, la càmera que més s'adapta a les nostres necessitats és una *IP Cam*. A més, en escollir-la hem resolt també el segon dubte, ja que sabem com funcionen per a rebre el seu senyal, el que ens porta ara a fer una cerca sobre els diferents models que trobem al mercat d'*IP Cams* per a veure quina se'ns pot ajustar per al nostre treball.

A més a més, disposem d'una *webcam* previ començament del projecte, pel què en cas de tenir problemes amb l'adquirida, tenim una alternativa.

Model de càmera escollit

En el procés d'escollir el model d'*IP Cam* se'ns ha obert també una altra possibilitat, i és utilitzar una càmera amb lent d'ull de peix que ens doni una visió total de la sala a la qual estem, mentre que les *IP Cams* ens permeten control sobre elles, fent que girin cap a on vulguem imitant en certa manera el moviment que faríem amb el cap i els ulls.

Aquest moviment podria ser vinculat al comandament remot de les ulleres de realitat virtual, d'aquesta manera podríem controlar-la i d'aquesta manera veure tota un àrea sense moure'ns del lloc. El motiu pel qual no es pot connectar al moviment de gir del cap de l'usuari és degut a què aquestes càmeres tenen uns motors que permeten fer els girs, i aquests motors tenen una potència concreta. A més, estan programats per a girar a una velocitat en concret, i com vam veure a l'apartat 2.2 La realitat virtual, necessitem tenir els nostres moviments sincronitzats amb el que estem veient.

Hem trobat diferents models que ens podrien arribar a funcionar per al nostre projecte, i entre aquests veurem breument els tres que hem considerat més interessants (entre ells, el model escollit, és clar).

- 1) *Reolink RLC-423*. Es tracta d'una càmera amb possibilitat de gir horitzontal de 360° i vertical de 90°. Permet fer zoom, que seria interessant de tenir a l'aplicació, té una qualitat de vídeo de 2560x1920 i té una simple instal·lació. Preu: 276.99€
- 2) *Tapo c200*. Aquesta càmera també consta d'un gir horitzontal de 360°, però té un gir vertical que arriba a 114°. Té una qualitat de vídeo de 1080p, i ve amb una aplicació que permet un fàcil control i visualització de la mateixa. Preu: 39.99€

- 3) *Vstarcam 360° HD 1536P*. Per últim tenim un model de càmera amb una lent d'ull de peix. Disposa d'un camp de visió de 360 graus i una correcció de deformitat per a veure les imatges de manera més natural. Preu: 59,99€

Finalment, tot i què no és la millor càmera per les seves especificacions, la Tapo c200 tenia un preu de 29.99€ en el moment de la decisió, i degut a les despeses que també comporten les ulleres de realitat virtual, hem decidit que aquesta seria la càmera amb què treballarem. Sabem que té certes limitacions, com ara la seva resolució, però és la que ens podem permetre.

3.1.2 Valoracions d'ulleres de realitat virtual

Juntament amb tot el que sabem sobre com funciona aquest aparell, sabem també que es tracta d'un dispositiu amb un preu força elevat, que pot arribar a superar el miler d'euros en alguns dels seus models (per exemple el model *HTC Vive PRO*). Per sort, el nostre projecte no té una part d'aplicació a les ulleres que requereixi un gran processador, cosa que ens permet poder utilitzar altres models més econòmics.

Com ja vam veure a l'apartat 2.2, generalment els models requereixen anar connectats a un ordinador, però com en el cas de les *IP Cams*, també n'hi ha que poden funcionar independentment.

El model de la marca *Oculus*, les *Oculus Go*, se'ns adapte perfectament a les nostres necessitats, ja que no només ens permet un grau més de llibertat gràcies a no tenir cap cable, sinó que també tenen un preu *relativament* assequible (especialment en compararlo amb les altres) a canvi de no tenir unes grans prestacions com d'altres models. Però com ja hem dit, a priori compleix les nostres necessitats, i la pròpia casa que les fabrica té a disposició per a qualsevol un paquet a *Unity* que permet poder treballar amb elles. El seu preu és de: 219€

3.1.3 Valoracions de targetes gràfiques

Encara que la nostra aplicació final funcionés únicament amb les ulleres i la càmera que ja hem adquirit, seguiríem necessitant una targeta gràfica per a desenvolupar el projecte. A diferència dels anteriors components, però, ja disposem d'una targeta gràfica que ens permetrà poder desenvolupar el projecte.

Un cop més, veiem com el cost pot suposar impediments i també limitacions als projectes, ja que tot i que la nostra gràfica compleix els requisits que es necessiten per a poder-lo realitzar com a tal no té suport per als *B Frames*, que permeten agilitzar molt més la transcodificació de les dades de la càmera a l'ordinador.

Tot i tenir aquest petit defecte, però, es tracta d'una targeta gràfica de gamma alta de la qual ja disposàvem prèviament al començament del projecte i, per tant, tenint en compte que també estem tenint limitacions que provenen d'altres fonts (com per exemple la càmera), i que els preus de les targetes que sí que tenen aquesta característica és molt elevat, seguirem el projecte treballant amb l'actual.

La pàgina oficial d'*Nvidia* ens ofereix una taula que ens diu les característiques relatives al codificat i descodificat de vídeo de les seves targetes [NVI].

Les targetes gràfiques d'aquesta marca tenen un codificador de vídeo desenvolupat per ells mateixos anomenat *Nvidia NVENC*, el qual s'encarrega de realitzar aquesta tasca (que és prou intensa), fet a partir del qual s'aconsegueix que no l'hagi d'executar la CPU. Podem veure les especificacions de la nostra, que la trobem situada a la quarta posició de la Figura 3.1. (*GeForce 1060*).

NVENC Support Matrix

BOARD	FAMILY	CHIP	Desktop/ Mobile/ Server	# OF CHIPS	# OF NVENC /CHIP	Total # of NVENC	Max # of concurrent sessions	H.264 [AVCHD] YUV 4:2:0	H.264 [AVCHD] YUV 4:4:4	H.264 [AVCHD] Lossless	H.265 [HEVC] 4K YUV 4:2:0	H.265 [HEVC] 4K Lossless	H.265 [HEVC] 4K 8k	HEVC B Frame support
GeForce														
GeForce GT 1030	Pascal	GP108	D	1	0	0	0	NO	NO	NO	NO	NO	NO	NO
GeForce GTX 1050 / 1050 Ti	Pascal	GP107	D/M	1	1	1	3	YES	YES	YES	YES	YES	YES	NO
GeForce GTX 1050 / 1050 Ti	Pascal	GP106	D/M	1	1	1	3	YES	YES	YES	YES	YES	YES	NO
GeForce GTX 1060	Pascal	GP106	D/M	1	1	1	3	YES	YES	YES	YES	YES	YES	NO
GeForce GTX 1060	Pascal	GP104	D/M	1	1	1	3	YES	YES	YES	YES	YES	YES	NO
GeForce GTX 1070M / 1080M	Pascal	GP104B	M	1	2	2	3	YES	YES	YES	YES	YES	YES	NO
GeForce GTX 1070 / 1070Ti	Pascal	GP104	D/M	1	2	2	3	YES	YES	YES	YES	YES	YES	NO
GeForce GTX 1080	Pascal	GP104	D/M	1	2	2	3	YES	YES	YES	YES	YES	YES	NO
GeForce GTX 1080 Ti	Pascal	GP102	D	1	2	2	3	YES	YES	YES	YES	YES	YES	NO
GeForce GTX Titan X / Titan Xp	Pascal	GP102	D	1	2	2	3	YES	YES	YES	YES	YES	YES	NO
Titan V	Volta	GV100	D	1	3	3	3	YES	YES	YES	YES	YES	YES	NO
GeForce GTX 1650	Turing*	TU117	D/M	1	1*	1*	3	YES	YES	YES	YES	YES	YES	NO
GeForce GTX 1660 Ti / 1660	Turing	TU116	D/M	1	1	1	3	YES	YES	YES	YES	YES	YES	YES
GeForce RTX 2060 / 2070	Turing	TU106	D/M	1	1	1	3	YES	YES	YES	YES	YES	YES	YES
GeForce RTX 2080	Turing	TU104	D/M	1	1	1	3	YES	YES	YES	YES	YES	YES	YES
GeForce RTX 2080 Ti	Turing	TU102	D	1	1	1	3	YES	YES	YES	YES	YES	YES	YES
Titan RTX	Turing	TU102	D	1	1	1	3	YES	YES	YES	YES	YES	YES	YES

Figura 3.1: Taula d'especificacions de cada model de gràfica amb la tecnologia *NVENC*

Com podem veure, el suport als *B Frames* no està present fins a les últimes files tot i que aquestes no es troben ordenades en funció del seu cost. Com a exemples podem veure la *GeForce 1080 Ti*, que té un preu proper als 800€ i la 1660 aproximadament 200-250€. (Pot diferir en funció del nombre de ventiladors, fabricant...)

Això té com a explicació el canvi generacional entre targetes. A excepció de la 1650, els models que veiem sense suport dels fotogrames B pertanyen a la generació anterior a les que sí que poden, que són la generació actual. (També podem veure que hi ha hagut un canvi a l'arquitectura, de la Pascal a la Turing)

Cal comentar també que les targetes gràfiques de la família *GeForce* no poden realitzar simultàniament més de dos codificacions de vídeo, mentre que altres targetes professionals són capaces de suportar fins a 21.

Per tant, treballarem amb la targeta gràfica *Nvidia GeForce GTX 1060*, ja que ens compleix una gran quantitat de requisits, a més de poder evitar així un altre desemborsament considerablement alt que no podríem fer actualment.

4 Implementació del projecte

El primer que hem realitzat ha estat una instal·lació completa del programa *Unity*. Aquest, però, no l'hem instal·lat directament, sinó que hem utilitzat l'eina de la qual hem parlat a l'apartat 2.8, el *Unity Hub*. Com ja hem dit abans, aquesta ens permetrà tenir descarregats i poder utilitzar en qualsevol moment la versió de l'editor que necessitem. Addicionalment, també ens permet escollir quins paquets volem tenir instal·lats al nostre editor, permetent a l'usuari personalitzar el seu entorn amb únicament el que necessiti.

Cal dir que totes aquestes primeres proves i tests, així com veure com funciona tot el procés, les hem realitzat sobre la versió 2018.4.19 de l'editor. El motiu és que la informació que hem trobat sobre com es realitzava era en aquesta, i sent el primer cop que treballem en aquest entorn, hem decidit seguir estrictament els tutorials. De totes maneres, part de la configuració és independent de l'editor que utilitzem, pel que ens serà útil igualment en un futur.

A continuació, hem fet una primera prova de connexió entre el nostre ordinador i les *Oculus Go*. Per a poder fer-la, és necessari activar abans el mode de desenvolupadors al dispositiu, el qual s'activa des de l'aplicació d'*Oculus* al nostre telèfon mòbil. Un cop descarregada, hem de vincular-la al nostre dispositiu i des dels paràmetres de configuració podem activar-lo fàcilment.

A l'ordinador, hem descarregat l'eina *Android Studio*, que és l'IDE oficial per al desenvolupament d'aplicacions a plataforma *Android*. Ho hem fet perquè les ulleres de què disposem utilitzen com a sistema operatiu *Android* (de fet, el seu xip és el processador que trobem a mòbils com ara l'*HTC 10* o l'*LG G5*).

Des del propi *Android Studio*, podem descarregar totes les eines de desenvolupament més actuals per a poder treballar. Un cop les tenim totes instal·lades podem veure el *path* on les tenim, i és necessari que aquests *paths* els guardem, ja que els haurem d'introduir a les variables d'entorn de *Windows*. Concretament, hem de guardar la localització de l'*Android SDK* (Software Development Kit), del *JDK* (Java Development Kit), i de l'*Android NDK*. (Native Development Kit)

Un cop introduïdes, podem dirigir-nos a *Unity*, i a la configuració de la *Build* hem d'assegurar-nos que estem desenvolupant per a l'entorn *Android*. Un cop comprovat, hem afegit, de la mateixa forma que a les variables d'entorn, els *path* que hem guardat prèviament sobre les preferències de *Unity*.

Fet això, podem obrir l'*Asset Store* i descarregar-nos les *Oculus Utilities*. Un cop les tenim, hem d'especificar al nostre projecte que volem treballar sobre realitat virtual, ja que d'aquesta manera, sabrà optimitzar millor l'aplicació creada per al nostre dispositiu. Després, hem de donar un nom de producte i un nom d'empresa a la configuració, així com cal especificar també el sistema operatiu per al qual l'estem creant, en el nostre cas és *Android 8.1 'Oreo'*. També hem de donar-li un nom identificatiu al paquet que es

crearà en fer la *build*. El seu format serà «com.NOMEMPRESA.NOMPRODUCTE».

El següent pas serà dirigir-nos a la pàgina web de *Unity* i identificar-nos amb el nostre usuari i contrasenya. Quan siguem dins del nostre compte, hem de crear el que s'anomena «una nova organització», i dins d'aquesta creem una nova aplicació especificant que serà sobre mòbil, ja que a efectes pràctics es pot dir que estem treballant sobre un. Aquests noms, com podem suposar, seran els mateixos que vam escollir prèviament a la configuració de *Unity*.

A continuació hem trobat el primer problema, i és que hem descarregat els *drivers* de l'ADB personalitzats que ofereix *Oculus*, però aquests no han funcionat, pel que hem trobat una altra versió.

Gràcies a les *Utilities* que hem descarregat, tenim als nostres *Assets* escenes d'exemple. Podem accedir a qualsevol d'elles i agafar i importar a una escena nova l'objecte càmera (anomenat *OVRCameraRig*), que aquest conté. Aquest objecte conté l'*script* de la càmera, on es troba la configuració necessària per a que ens detecti els moviments que fem amb el cap, alhora que ens permet accedir per separat al que seria la pantalla de cada ull. A més, també té tot el necessari per a detectar el comandament a distància amb el que venen les ulleres.

Completats tots aquests passos, podem connectar amb el cable USB el nostre dispositiu a l'ordinador. Si utilitzem la comanda «adb devices» a la terminal, podem veure si aquest s'ha connectat correctament, però tot i estar-ho, no estarà autoritzat per a depurar inicialment. Únicament ens hem de col·locar les ulleres i acceptar aquesta connexió amb el comandament. En fer-ho, podem veure amb la mateixa comanda com sí que està autoritzat.

Finalment, podem fer la nostra primera *build*. Aquest és un procés excessivament lent, en especial quan es realitza per primer cop. En finalitzar, la podem moure a la carpeta on tenim instal·lat l'ADB, i amb la comanda «adb install NOM.apk» podem carregar la *build* sobre el dispositiu, que haurem de buscar a la carpeta *Unknown Sources*.

Acabada aquesta configuració, passem a treballar amb la *IP Cam*. El model que hem escollit té una aplicació pel nostre mòbil que permet una ràpida instal·lació i configuració. En aquesta configuració, hem d'especificar un identificador i una contrasenya per a la retransmissió.

Per a realitzar les primeres proves, cal un software de vídeo al nostre ordinador que sigui capaç de fer connexions RTCP, com per exemple el *VLC Media Player*. Amb aquest software hem pogut connectar-nos a la càmera amb èxit (tot i que s'experimenta un cert retard al senyal).

I és aquí on ens hem trobat amb el primer gran problema que ens ha condicionat el desenvolupament de l'aplicació: *Unity* no és compatible amb el protocol RTSP.

Per aquest motiu, la idea de realitzar l'aplicació a les ulleres i connectar-les a la càmera, tenint així un sistema amb únicament aquests dos dispositius, ha hagut de descartar-se. A més, les *Oculus Go* tenen tot el material de suport als desenvolupadors a la plataforma de *Unity*, pel que tampoc podem fer un canvi sense prèviament adquirir un nou sistema. (Que és totalment inviable amb el nostre pressupost)

Davant d'aquesta situació, se'ns ha ocorregut una possible solució al problema, que

és connectar la càmera al nostre ordinador i retransmetre les imatges a les ulleres. D'aquesta manera, hem buscat de diferents formes la manera en què es pugui connectar la *IP Cam* a l'ordinador, que aquest la detecti com a una càmera més del sistema (com si fos una *webcam*), i poder carregar la imatge a *Unity*.

Després d'investigar alternatives per a connectar-la, i navegar per diferents fòrums, hem vist que aquest problema porta a la comunitat més de cinc anys, i encara no hi ha suport oficial que permeti aquesta connexió. De fet, a la versió 2020.1.7, la versió més recent de *Unity* publicada el dia 1 d'octubre de 2020, encara no hi ha suport del protocol RTCP.

Veient tots aquests inconvenients, i sabent que disposem d'una *webcam* que podem connectar per USB al nostre ordinador, hem decidit optar per treballar amb aquesta opció. Tot i que no és la millor, ja que ens agradaria poder tenir una càmera independent que es pugui instal·lar a on vulguem, ens permetrà treballar i fer un prototip del disseny inicial.

Respecte a la retransmissió, vam trobar més d'una possible opció, però entre aquestes hi havia components a l'*asset store* no disponibles de franc, sinó que amb preus oscil·lant entre els 50 i 150 euros. Finalment, l'opció que més viable ens va semblar va ser la següent.

4.1 Unity Render Streaming

Unity Render Streaming (URS) tracta d'un paquet desenvolupat per un petit grup de treballadors de *Unity* que va ser llançat al mes de setembre de 2019. Ofereix un servei de retransmissió *peer-to-peer* directament des del mateix programa, tot gràcies als avantatges proporcionats per WebRTC. Aquest *streaming* s'activa en el moment de donar-li al *play* a la IDE de *Unity*, i s'envia la informació a temps real de l'escenari 3D que s'està renderitzant, a més de l'àudio. Una característica a destacar és que permet també interactuar amb l'escenari de manera remota, de la mateixa manera que ho podem fer a la IDE, alhora que també podem tenir múltiples connexions de manera simultània.

Hem considerat que aquest és el sistema més adequat a utilitzar, ja que inicialment complim tots els requeriments per al seu ús [Uni19], i en veure les facilitats que s'han afegit amb el pas dels mesos sobre el *software*, creiem que s'adapta al nostre objectiu perfectament. Malgrat això, és un projecte que encara es troba en desenvolupament, i la plataforma on millor es pot treballar és a Windows, ja que les versions per a MacOS i Linux estan en desenvolupament i no són igual d'estables. En el nostre cas, se'ns adapta, ja que és el sistema operatiu sobre el qual estem treballant, però veiem que haguéssim estat limitats en cas d'estar treballant sobre un altre.

Entre les millores afegides des del seu llançament, trobem la possibilitat de retransmetre més d'una càmera (quan parlem de càmera estem parlant de l'objecte càmera del motor), el que ens permetria assignar-ne una per a cada lent del nostre *headset* en treballar en el mode de realitat virtual de *Unity*, juntament amb els paquets de suport que ofereix *Oculus*. Sabem també que ofereix suport per a *Android*, i realment, les nostres ulleres no són més que un dispositiu mòbil *Android* adaptat per a funcionar com a

sistema de realitat virtual.

Per al seu correcte funcionament, s'ha d'instal·lar una sèrie de *softwares* i de complir certes dependències. Cal destacar, però, que la informació oficial no es troba actualitzada, a més de tenir més d'un URL trencada i no especificar certs detalls molt importants.

És per aquest motiu que la instal·lació ha estat complicada, i l'hem aconseguit després de provar diferents combinacions de versions de l'editor de *Unity* i del paquet en qüestió, a més de diferents versions de *drivers* (o controladors) de la targeta gràfica.

A continuació, podem veure una llista amb tot el *software* necessari per al funcionament del programa, les versions utilitzades i les dependències que trobem.

- *Versions de Unity i URS*. La combinació que ha resultat més estable en les proves que hem realitzat ha estat la de *Unity* 2019.2.0.f1 juntament amb la *release* 1.1.1 de *Unity Render Streaming*. En essència, el procés per a aconseguir-ho ha estat afegint una plantilla de projecte a *Unity Hub* amb els paquets d'URS incorporats sobre aquesta, i actualitzant a l'editor tots els paquets necessaris que té aquest com a dependències.
- *Nvidia NVENC*. Aquest és un SDK que conté un còdec de vídeo especial que ofereix Nvidia per a les seves targetes gràfiques. Permet utilitzar el *hardware* de codificació i descodificació de la gràfica des d'altres aplicacions, en el nostre cas, *Unity*. Per a poder-ne fer ús trobem dues opcions d'instal·lació. Per una banda, es poden descarregar els arxius i fer una *build* amb l'ajuda de *CMake* (un conjunt d'eines que permeten empaquetar software). Per l'altra es pot descarregar una eina anomenada *FFMPEG*, que ens permet també accedir al HW de la nostra gràfica d'una manera més senzilla. Primerament vam intentar fer la *build* nosaltres mateixos amb *CMake*, però en veure que no funcionava, vam optar per instal·lar *FFMPEG*, que ens ha funcionat correctament.
- *Controladors de la targeta gràfica*. Nvidia ens especifica que la gràfica ha de tenir els seus controladors actualitzats a la versió 445.87 o superior. En el nostre cas teniem una versió inferior, per tant, vam actualitzar a la recomanada. Un cop testejat el funcionament, vam voler provar d'actualitzar a una versió més nova, la 456.55. Finalment, hem decidit deixar aquesta i no tornar enrere, pel fet que no hi ha hagut cap impacte negatiu sobre el funcionament, i és una versió més nova.
- *Model de targeta gràfica*. Un aspecte a tenir en compte sobre aquest paquet (URS), és que necessita una gràfica potent per al seu funcionament. En el cas de la família de gràfiques *GeForce*, a la qual pertany la nostra, es necessita disposar d'una sèrie 10 com a mínim (sèrie relativament nova i amb preus generalment elevats). Per sort, la nostra forma part d'aquesta com vam veure prèviament, motiu pel qual hem pogut treballar amb aquest paquet correctament.
- *DirectX SDK*. Aquesta és una altra dependència que té el còdec de Nvidia, concretament la *release* de juny de 2010. Es tracta d'una col·lecció de llibreries que ajuden en el desenvolupament de videojocs. S'ha de tenir cura en instal·lar-lo, ja

que si tenim al nostre ordinador el paquet *redistributable* Microsoft Visual C++ 2010, ens causarà un error d'instal·lació. Aquest és un paquet que conté biblioteques i eines per als programes escrits en Visual C++, i es necessita desinstal·lar del sistema per a la correcta instal·lació del SDK de DirectX. Un cop completada, podem tornar a reinstal·lar-lo sense cap problema.

- *CUDA 10.0 Toolkit*. És l'última dependència de la tecnologia NVENC. Es tracta d'un conjunt d'eines que permeten el desenvolupament d'aplicacions que treballen amb el *hardware* de la nostra GPU. Com vam parlar prèviament a la secció 2.3, a una GPU trobem més facilitat per a paral·lelitzar processos, i el procés de codificació implica realitzar les mateixes operacions un gran número de vegades, és per aquest motiu que la codificació de vídeo és més eficient a una GPU que a una CPU.
- *Unity WebRTC*. Aquest és un paquet que no es menciona en cap moment a la guia d'instal·lació, el que fa pensar que ve incorporat en els arxius de l'URS, ja que en la seva pròpia descripció parlen sobre com fa ús d'aquesta tecnologia per a funcionar. Malauradament, no és el cas, motiu pel qual l'hem de descarregar i instal·lar manualment. Concretament, la versió que ens ha servit ha estat la 1.0.1.
- *Node.js*. Es tracta d'un entorn de programació basat en *JavaScript* generalment emprat per a servidors web. Tot i tampoc esmentar-se als passos d'instal·lació de l'URS, el necessitem per al correcte funcionament del mòdul WebRTC i poder fer un *streaming* localment.

En acabar de configurar-ho tot, el primer que hem fet ha estat comprovar el funcionament de la retransmissió, per assegurar que tot està correcte. La primera prova l'hem realitzat amb el mateix ordinador sobre el qual estem treballant, amb una escena bàsica d'exemple. Primerament, executem l'aplicació del servidor d'URS, que ens obre una terminal on veiem diverses adreces IP, entre les quals n'hem d'escollir una i especificar-la a l'*asset* de la càmera. Fet això, iniciem l'execució del programa i ens dirigim a l'adreça especificada prèviament al nostre navegador. Un cop carregada la pàgina, veiem un gran botó de *play* al centre del navegador, en donar-li, se'ns connecta correctament al cap d'uns pocs segons.

Un cop comprovat el funcionament, passem a provar altres dispositius, entre ells un telèfon mòbil, un altre ordinador i les *Oculus Go*. Ens hem trobat que no podíem realitzar una connexió per a cap d'aquests dispositius, per tant, hem de trobar quin és el problema.

Finalment, el problema va resultar ser el mateix Windows, o millor dit, el seu *Firewall* (o tallafocs), que no deixava connectar als altres dispositius. Un cop desactivat, tornem a fer les proves un cop més.

I és aquí on ens hem tornat a trobar de nou un problema important: les ulleres no són compatibles amb l'*streaming*. Hem aconseguit la retransmissió en els altres dispositius, però les ulleres no resulten ser compatibles amb la retransmissió d'URS, tot i ser també un dispositiu *Android* i poder-nos connectar amb el mòbil. Això és degut al fet que

el dispositiu no es troba prou actualitzat per a poder treballar amb WebRTC, el que significa que ens és impossible treballar amb elles.

Arribats en aquest punt, el que hem decidit és acabar de configurar l'aplicació de totes formes, ja que així, en cas de poder treballar amb un model de *headset* més actualitzat, que sigui capaç de fer aquesta connexió, ja tindríem una base sobre la qual podríem treballar i poder realitzar l'aplicació al complet.

Com que no es pot fer la connexió amb la *IP Cam* des de *Unity*, hem decidit treballar amb la *webcam* de la que ja disposàvem, que, tot i no tenir una gran resolució, ens serveix per a poder veure si la resta del muntatge seria viable.

Per tant, hem realitzat un petit *script* perquè ens la detectés, i el vídeo rebut l'hem representat com a textura per a fer proves sobre un objecte tridimensional amb forma de pantalla davant l'objecte càmera.

Un cop hem aconseguit tenir simultàniament l'*streaming* amb la càmera funcionant, el que hem vist és que la idea original de transmetre el vídeo de dues càmeres, una per a cada ull, no es podria fer així com així. Els motius que hem trobat són dos: en primer lloc la càmera no fa una sensació de tenir-la realment davant tot i ajustar-la, i en segon lloc, un input des de qualsevol dispositiu connectat et desplaça per l'escena i es desajusta la pantalla.

Per a solucionar aquests dos problemes finalment, el que hem fet és representar la transmissió de vídeo de la *webcam* com si fos part de la interfície d'usuari, configurant-la per a escalar-se correctament ocupant tota la pantalla. D'aquesta forma, tenim realment la retransmissió davant dels nostres ulls, i a més la tindrem sempre tot i que ens desplacem, en ser part de la mateixa interfície.

5 Conclusions

En acabar el treball, podem dir que tot i no haver pogut assolir l'objectiu final, el resultat ha estat satisfactori, ja que els motius pels quals no s'ha pogut arribar han estat fora del nostre abast. Quan vam escollir aquest projecte, teníem una visió molt superficial sobre el món de la realitat virtual, i creïem que era possible realitzar aquest muntatge amb el material adquirit, cosa que hem demostrat inviable.

El món de la realitat virtual viu en un estat de canvi constant, on veiem com en petits intervals de temps, els dispositius s'actualitzen deixant enrere els seus predecessors ràpidament. Hem vist un clar exemple durant el desenvolupament del projecte: aquest mateix estiu es va anunciar l'aturament de producció de les *Oculus Go*, el dispositiu que vam adquirir per a realitzar aquest treball, a més del tancament oficial de la seva *store*, on no es podrà afegir cap més contingut i únicament es podran trobar les aplicacions que hi fossin prèviament.

Com bé vam dir a l'hora de fer la compra, aquest *headset* era l'única opció viable tenint en compte el pressupost del qual disposàvem, però realment creïem que seria prou potent per a la realització de l'aplicació, en ser aquesta essencialment una retransmissió de vídeo. Hem vist, però, que no és el cas, sent impossible la connexió amb l'*streaming* tot i que un altre dispositiu *Android* més actualitzat sí ha estat capaç.

Malgrat aquestes limitacions que ens hem trobat, pensem que l'aplicació que hem desenvolupat demostra que es podria realitzar aquest sistema donats els materials adequats. A més, es tracta d'una aplicació que, en cas d'obtenir aquests materials (càmeres 4K, millors ulleres de realitat virtual i una GPU adequada), està pensada de manera que podria escalar-se.

Respecte a la part teòrica, podem dir que hem assolit els conceptes estudiats. Un exemple que ens pot reafirmar això, és el fet d'adonar-nos que una de les idees que teníem en un inici, contradeïa el mateix principi amb el qual la justificàvem. En un principi, creïem que era possible simular una sensació de visió estereoscòpica tenint una única càmera, però en estudiar realment les tècniques i els fenòmens que generen aquesta visió, vam entendre que era impossible.

En conclusió, creiem que es va cometre un error a l'hora de valorar la dificultat d'aquest TFG inicialment, ja que com hem vist, si volem treballar en l'àrea de la realitat virtual, necessitem disposar dels millors materials possibles per a evitar els problemes amb què ens hem trobat. No només això, sinó que la informació que podem trobar a l'hora de desenvolupar una aplicació per a VR, és principalment via fòrums, on podem o no trobar una solució, a més d'una documentació molt escassa, errònia o inexistent. Malgrat això, també pensem que hem realitzat un bon treball trobant solucions per als problemes amb els quals ens hem enfrontat, alhora que hem après nous conceptes, entès millor els que ja coneixíem, i hem aconseguit finalment una aplicació escalable en un futur.

6 Treball futur

Si enfoquem aquest treball de cara al futur, podem veure que se'ns obren dues possibles opcions. En tots dos casos, però, hi trobem una part en comú, i és la necessitat d'adquisició d'un *hardware* més potent que l'actual.

En cas de tenir un pressupost adequat que ens ho permeti, hauríem de substituir la càmera actual per dues càmeres de qualitat 4K i amb capacitat de retransmissió a 60FPS. La targeta gràfica de la qual disposem ens ha permès realitzar el projecte, però seria ideal disposar d'una targeta professional enfocada al processament de vídeo (com ara la família de targetes *Quadro* d'*Nvidia*). Respecte a les ulleres de realitat virtual, estaria bé disposar d'unes que permetin treballar no només amb un motor, sinó que ofereixin més possibilitats, com ara les *HTC Vive Pro*. D'aquesta manera, disposaríem de més opcions en cas de trobar-nos amb un problema com el que hem tingut amb *Unity* i les *IP Cams*.

Pel que fa a les opcions que hem parlat abans, la primera és la d'escalar el projecte que hem dissenyat. Necessitaríem dues càmeres, preferentment del mateix model, i com hem explicat prèviament, configurariem les dues lents del dispositiu VR perquè cadascuna vegi a través d'una de les càmeres. Aquestes s'haurien de col·locar a una distància que es podria adaptar per a cada usuari, simulant la seva disparitat binocular i obtenint així una sensació d'immersió superior.

La segona opció implica un canvi substancial, i és la de treballar amb el motor *Unreal Engine*. Dispositius com ara els de la marca *HTC* ho permeten, i a *Unreal* tenim el gran avantatge de què és compatible amb el protocol RTSP. Per tant, es podria treballar amb *IP Cams*, cosa que a *Unity* actualment no es pot, i es podria també desenvolupar una aplicació nativa, que de fet, era la idea principal d'aquest projecte.

Això seria probablement una millor implementació que la que ens hem vist obligats a fer, perquè la imatge provinent de les càmeres no hauria de passar per un punt intermedi, en aquest cas, l'ordinador. Això repercuteix en una millor qualitat de vídeo, ja que ens estalviem passar per un còdec de més, així com reduiríem el temps que triga la imatge a arribar de la càmera als nostres ulls.

Bibliografia

- [Anu18] Anurag. *How VR Works? Know The Technology Behind Virtual Reality*. Ang. 2018. URL: <https://www.newgenapps.com/blog/how-vr-works-technology-behind-virtual-reality/>.
- [Coo17] Daniel Cooke. *When was virtual reality invented?* Ang. 2017. URL: <https://www.pebblestudios.co.uk/2017/08/17/when-was-virtual-reality-invented/>.
- [Den] Thomas Denham. *What is Rendering? (For 3D and CG Work)*. Ang. URL: <https://conceptartempire.com/what-is-3d-rendering/>.
- [dev20] Android developers. *Android Debug Bridge (adb)*. Ang. 2020. URL: <https://developer.android.com/studio/command-line/adb>.
- [For] Peter Forret. Ang. URL: <https://toolstud.io/video/filesize.php>.
- [Goo] Google. Ang. URL: <https://webrtc.org/>.
- [Goo18] Google. *Degrees of freedom*. Ang. 2018. URL: <https://developers.google.com/vr/discover/degrees-of-freedom>.
- [Inc16] Mammoth Security Inc. *What is an IP Camera?* Ang. 2016. URL: <https://mammothsecurity.com/what-is-ip-camera/>.
- [Kov] Nadia Kovach. *What is VR and how does it work?* Ang. URL: <https://thinkmobiles.com/blog/what-is-vr/>.
- [Low15] Steven Lowe. *Composition vs. Inheritance: How to Choose?* Ang. 2015. URL: <https://www.thoughtworks.com/insights/blog/composition-vs-inheritance-how-choose>.
- [Mat03] Philip Goldie Matthew Syme. *Optimizing Network Performance with Content Switching: Server, Firewall and Cache Load Balancing: Server, Firewall, and Cache Load Balancing*. Ang. 1a ed. 2003. 288 pàg. ISBN: 0-13-101468-4.
- [MGS98] A. Medina, Helena Gerson i Sheryl Sorby. «Identifying gender differences in the 3D visualization skills of engineering students in Brazil and in the United States». A: (gen. de 1998).
- [Nik] Lina Nikols. *The Beginner's Guide to Video Encoding, Decoding, and Transcoding*. Ang. URL: <https://www.haivision.com/blog/all/the-beginners-guide-to-video-encoding-decoding-and-transcoding/>.
- [NVI] NVIDIA. *Video Encode and Decode GPU Support Matrix*. Ang. URL: <https://developer.nvidia.com/video-encode-decode-gpu-support-matrix>.

- [Poe19] Bridget Poetker. «The Very Real History of Virtual Reality (+A Look Ahead)». A: (set. de 2019).
- [Pol13] Pietro Polsinelli. *Why is Unity so popular for videogame development?* Ang. 2013. URL: <https://designagame.eu/2013/12/unity-popular-videogame-development/>.
- [Poo] Render Pool. *CPU vs. GPU Rendering: Which Is Best for Your Studio Projects?* Ang. URL: <https://renderpool.net/blog/cpu-vs-gpu-rendering/>.
- [Rom18] Anthony Romero. *What is Video Encoding? Codecs and Compression Techniques*. Ang. 2018. URL: <https://blog.video.ibm.com/streaming-video-tips/what-is-video-encoding-codecs-compression-techniques/>.
- [Rou] Margaret Rouse. *Component definition*. Ang. URL: <https://whatis.techtarget.com/definition/component>.
- [Sec] Brick House Security. *Internet Cameras Explained*. Ang. URL: <https://www.brickhousesecurity.com/security-cameras/ip-camera-guide/>.
- [Sut68] Ivan E. Sutherland. «A head-mounted three dimensional display». A: *The University of Utah* (1968). URL: <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.388.2440&rep=rep1&type=pdf>.
- [Unia] Unity. Ang. URL: <https://store.unity.com/compare-plans>.
- [Unib] Unity. Ang. URL: <https://docs.unity3d.com/Manual/GettingStartedInstallingHub.html>.
- [Unic] Unity. *Renderizado en tiempo real en 3D*. Ang. URL: <https://unity3d.com/es/real-time-rendering-3d>.
- [Uni19] Unity. Ang. 2019. URL: <https://github.com/Unity-Technologies/UnityRenderStreaming>.
- [Wad02] Nicholas J Wade. «Charles Wheatstone (1802 – 1875)». A: *Perception* 31.3 (2002). PMID: 11954689, pàg. 265-272. DOI: 10.1068/p3103ed. eprint: <https://doi.org/10.1068/p3103ed>. URL: <https://doi.org/10.1068/p3103ed>.
- [WM11] Stefan Winkler i Dongbo Min. «Stereoscopic image quality compendium». A: *ICICS 2011 - 8th International Conference on Information, Communications and Signal Processing* (des. de 2011). DOI: 10.1109/ICICS.2011.6173571.