

PC



ARMANSION

un joc de realitat augmentada



12+

Hugo Ramos Montesinos

Tutor: Jordi Vendrell

Curs 2021-2022

2^º Batxillerat Científico-Tecnológico

Agraïments

Agraeixo al meu tutor Jordi Vendrell per l'ajuda i guia proporcionada per a fer aquest treball, a més de comptar amb l'ajuda del meu germà Mario Ramos, per a desenvolupar la part més pràctica.

Abstract

Este proyecto se basa en explicar que es y como funciona la Realidad Aumentada, añadiendo su historia, evolución con el paso de los años y usos. Como objetivo principal, para acabar de remarcar con un ejemplo práctico todo lo que puede ofrecer esta tecnología, el proyecto trata sobre la creación de un videojuego multiplataforma desarrollado tanto para ordenadores como para dispositivos móviles, haciendo uso de la Realidad Aumentada.

Para conseguir el objetivo planteado, se han empleado aplicaciones que ofrecen soporte a lenguajes de programación, como Visual Studio, motor gráfico, Unity, y por último diseño de escenarios, Tiled.

Como conclusión, se puede afirmar que la Realidad Aumentada aún es una tecnología reciente que debe evolucionar para brindar a la sociedad actual ideas y posibilidades revolucionarias. Además, a otra conclusión que se ha llegado, es la dificultad y esfuerzo que supone crear un videojuego desde cero, sin ningún tipo de experiencia previa.

This project is based on explaining what Augmented Reality is and how it works, adding its history, evolution over the years and uses. As a main objective, to finally highlight with a practical example all that this technology can offer, the project is about the creation of a multiplatform video game developed for both computers and mobile devices, making use of Augmented Reality.

To achieve this goal, applications have been used that offer support for programming languages, such as Visual Studio, graphic engine, Unity, and finally scenario design, Tiled.

In conclusion, it can be asserted that Augmented Reality is still a recent technology that must evolve to provide today's society with revolutionary ideas and possibilities. In addition, another conclusion that has been reached is the difficulty and effort involved in creating a video game from scratch, without any previous experience.

Índex

Motivacions (Introducció)	1
Part Teòrica	3
Què és la Realitat Augmentada?	3
2.1 Definició: RA	3
2.2 Característiques i Nivells: RA	4
2.3 Diferències: RA i RV	7
Personatges Històrics: RA	8
Frank L. Baum	9
Louis B. Rosenberg	9
Morton Heilig	10
Usos: RA	11
Educació	11
Vehicles	11
Moda	12
Sector Immobiliari	12
Decoració	13
Arquitectura i Construcció	13
Logística	14
Videojocs i Oci	14
Exemple: Pokémon GO!	14
Part Pràctica	16
Objectiu	16
Aplicacions: ARMansion	17
Motor Gràfic: Unity	17
Codi: Visual Studio	20
Mapa: Tiled	22

Desenvolupament: ARMansion	24
Apartats Generals	24
Mapes	24
UI - User Interface (Interfície d'Usuari)	27
Apartats Específics	31
Jugador i Moviment (animacions)	31
Interaccions amb el Mapa	34
Proves i Final	36
Conclusions	51
Bibliografia	52
Annexos	57

1. Motivacions (Introducció)

Des que tinc consciència he sentit una forta atracció pel sector informàtic i les seves variants, com els videojocs, codi, creació de pàgines web, aplicacions mòbils, etc. Durant la major part de la meva infància, observava com el meu germà jugava a videojocs i a vegades me'n deixava provar alguns, creant en mi una curiositat i gran admiració per tot aquest món digital. Avui dia conservo tot això, a més, que és un dels entreteniments més importants i agradables que tinc, ja que hi dedico la major part del meu temps lliure en ells.

A mesura que anava creixent, la meva curiositat augmentava en altres àmbits dins del món de la informàtica. Ja no sols em centrava en els videojocs, sinó que m'informava via fòrums, vídeos, pàgines webs, sobre *hardware* i *software*. Al principi, em centrava en *software*, ja sigui instal·lant els meus primers programes, descarregant els meus primers jocs, restaurant de fàbrica el meu ordinador, etc. Amb tot això, em van sorgir dubtes que avui dia m'agradaria poder respondre.

El dubte que em va sorgir va ser: Quines possibilitats oferia la **interacció entre hardware i software?**

Allà va ser quan vaig decidir buscar un nou enllaç entre *hardware* i *software* i va sorgir la Realitat Augmentada. Aquesta tecnologia fa interactuar al *hardware* i al *software* de manera única. Per a resoldre aquesta qüestió requereixo d'una recerca en profunditat més enllà d'una consulta en la Wikipedia.

Per aquest motiu he triat aquest tema, per a fer un treball de cerca que sigui atractiu i interessant, a més de per a resoldre aquest dubte, que, des de fa temps, com apassionat de la informàtica, he tingut.

D'altra banda, aquest treball em serveix, a través del desenvolupament d'un cas pràctic, per a aprendre a programar des de gairebé zero i preparar-me per al que realment vull fer en un futur, que consisteix en ser programador.

L'exemple pràctic que vull desenvolupar és un videojoc, o, més ben dit, una aplicació d'entreteniment: **Un Scape Room, amb elements de Realitat Augmentada.**

El llenguatge de programació que he pensat utilitzar per a desenvolupar el *Scape Room* és C#. A més de comptar completament amb l'ajuda del motor gràfic de Unity, ja que aquest compta amb una pàgina web, on es poden trobar diversos tutorials i facilitats per a un programador inexpert.

En resum, aquest treball intenta resoldre les següents qüestions:

- **Explicar en què consisteix la Realitat Augmentada i la seva història.**
- **Desenvolupar un videojoc. Això comporta interactuar amb un llenguatge de programació, un motor gràfic i aprendre a utilitzar-lo.**

Aquest treball tracta sobre la Realitat Augmentada utilitzada en una aplicació d'entreteniment. Tot això em servirà per a fer una intensa cerca sobre la història d'aquesta nova tecnologia i sobre com ha anat evolucionant. A més d'aprendre a desenvolupar una aplicació i adaptar els meus nous coneixements sobre Realitat Augmentada per a poder utilitzar-la en noves aplicacions que desitgi realitzar, en conseqüència, ser capaç d'adaptar els meus nous coneixements en futurs projectes per a obtenir el desitjat.

Part Teòrica

2. Què és la Realitat Augmentada?

2.1 Definició: RA

La Realitat Augmentada és una imatge, objectes o dades renderitzades per qualsevol dispositiu, ja sigui per ordinador, telèfon intel·ligent, tauleta...(a través de la càmera) que estan sobreposades sobre el món real. Una altra manera de dir-ho, a més de ser una relació que existeix entre hardware i software a temps real, és la relació entre el món físic (real) un món digital.

Aquesta definició permet extreure diverses definicions de què és la Realitat Augmentada en diferents àmbits:

- Realitat Augmentada en el Reconeixement: D'aquest tipus de Realitat Augmentada utilitza marcadors que han de reconèixer elements per a veure un objecte 3D en el nostre dispositiu.
- Realitat Augmentada en la Localització: Aquest tipus de Realitat Augmentada mostra objectes en el nostre dispositiu depenent del lloc físic en el qual ens trobem en el moment, usant l'entorn.
- Realitat Augmentada en Superposició: Aquest tipus de Realitat Augmentada reemplaça parcialment o totalment, la visió real d'un objecte per a mostrar una versió augmentada d'aquest.
- Realitat Augmentada en la Projecció: Aquesta projecta una imatge en elements i espais físics del món real. És possible crear projeccions interactives, és a dir, hologrames.

2.2 Característiques i Nivells: RA

Les característiques principals que conformen la Realitat Augmentada són les següents:

- La Realitat Augmentada és una tecnologia que superposa elements虚拟 a la imatge real, és a dir, és capaç de combinar el món material o físic amb allò virtual.
- Una altra característica important que ens permet la Realitat Augmentada, és la d'interaccionar amb ella a temps real, és a dir, que les accions que realitza l'usuari creen una reacció sobre la recreació de la realitat que es visualitza.
- La imatge virtual pot adquirir capacitats i proporcionalitats físiques de l'entorn.

Dins de la Realitat Augmentada, existeixen diferents nivells segons la relació que tinguin amb la realitat, és a dir, aquesta anirà relacionada segons el *hardware* que tingui a la seva disposició. Els nivells són els següents:

- Nivell 0 - *Physical World Hyperlinking* (Enllaçat amb el món físic). Aquest és el nivell més bàsic de Realitat Augmentada, ja que consisteix a escanejar codis QR o de barres, per a enllaçar-los a un altre contingut. Un exemple actual és poder escanejar un codi QR en qualsevol local, per exemple un restaurant, i obtenir un enllaç que et mostri la carta.



Imatge 1 - Codi QR. Exemple d'un restaurant

- Nivell 1 - *Maker Based AR* (Realitat Augmentada amb marcadors). S'empren marcadors per reconèixer patrons en 2D o 3D sense complexitat, és a dir, dibuixos esquemàtics o figures. En paraules més simples, consisteix a renderitzar imatges o objectes a través de codis QR o imatges de referència. Un exemple d'això:



imatge 2 - Codi QR 3D. Exemple genèric

- Nivell 2 - *AR without markers* (Realitat Augmentada sense marcadors). En aquest cas, en compte d'utilitzar marcadors, s'utilitzen brúixoles digitals (GPS o ubicació del dispositiu) per reconèixer la localització de l'usuari i així projectar imatges virtuals. Un exemple d'això, pot ser un simple filtre d'Instagram, que renderitza un PNG a temps real, sense necessitat d'un marcador, com podria ser un codi QR.



imatge 3 - Realitat Augmentada sense marcador, filtre d'Instagram

- Nivell 3 - *Augmented vision* (Visió Augmentada). S'utilitzen dispositius d'alta tecnologia per a oferir una experiència totalment immersiva i fusionada amb la realitat. Un exemple són les ulleres de Visió Augmentada desenvolupades (prototip) per la marca de vehicles MINI. Aquestes ulleres compten amb un sistema interconnectat, que ofereix funcions interactives personalitzades a través de la Realitat Augmentada.



Imatge 4 - Prototip ulleres RA desenvolupat per MINI



Imatge 5 - Prototip visual o ús de les ulleres RA desenvolupades per MINI

2.3 Diferències: RA i RV

La Realitat Virtual (VR) i la Realitat Augmentada (AR) són tecnologies recents, conegudes per la seva experiència enriquidora, que reuneix un món virtual amb el real. Encara que són fàcils de confondre, existeixen algunes diferències significatives.

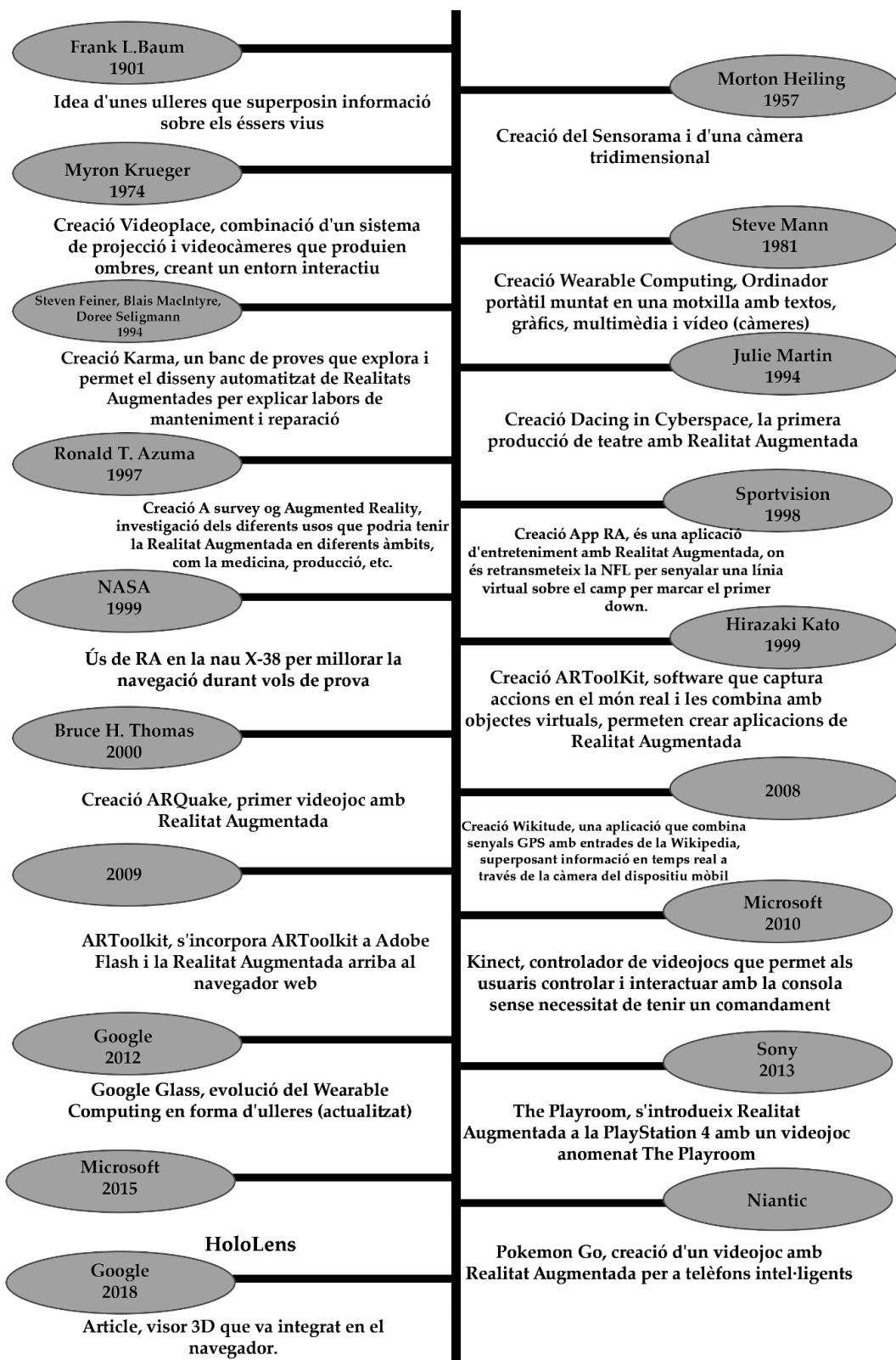
Realitat Augmentada (AR): És la relació entre el món digital i físic, consisteix a projectar imatges o objectes renderitzats virtualment sobre el món real a través d'un dispositiu.

Realitat Virtual (VR): Utilitza dispositius amb sensors (ordinadors, guants, ulleres...) per a produir una simulació totalment immersiva en el món digital. Aquestes simulacions poden crear gairebé qualsevol lloc visual o imaginable per a l'usuari.

Les distincions entre la Realitat Augmentada i la Realitat Virtual es redueixen als dispositius que requereixen i a la mateixa experiència com a tal:

- La Realitat Augmentada utilitza l'entorn real (món real) mentre que la Realitat Virtual és completament virtual.
- Els usuaris de Realitat Augmentada poden controlar la seva presència en el món real, mentre que en la Realitat Virtual són controlats pel sistema.
- La Realitat Augmentada no requereix d'un gran equip, és a dir, un equip d'alta gamma per poder utilitzar-se, mentre que la Realitat Virtual requereix d'equips d'alta gamma, és a dir, un bon ordinador, ulleres de Realitat Virtual, etc.

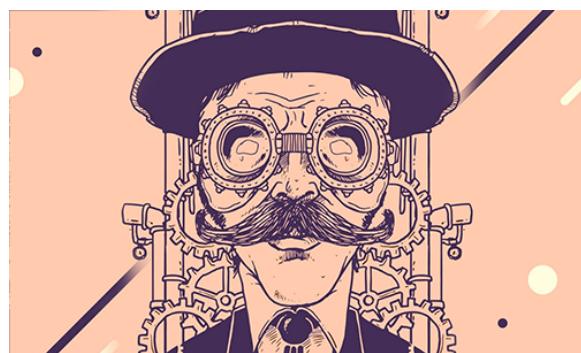
3. Personatges Històrics: RA



Diferents personatges al llarg de la història van donar el seu punt de vista a la Realitat Augmentada o van utilitzar les seves idees per a donar-li vida a aquesta mateixa.

Frank L. Baum

Frank Lyman Baum (Chittenango, Nova York, els EUA) 1856-1919, escriptor estatunidenc de llibres infantils. Va conèixer l'èxit comercial amb el seu llibre Father Goose al qual va seguir la història més popular del seu catàleg, El meravellós *mago de Oz*. En 1901 va donar conèixer la seva idea d'unes ulleres electròniques que poguessin superposar informació sobre els éssers vius. Les va anomenar *Character Maker*.



imatge 6 - Dibuix Frank L.B.

Louis B. Rosenberg

Louis B. Rosenberg (24 maig de 1969) tecnòleg, inventor, emprenedor, escriptor i director executiu i científic de l'empresa Unanimous AI. El seu doctorat en la Universitat de Stanford va desembocar en el sistema d'accessos virtuals que va utilitzar la Força Aèria dels EUA. Això va desembocar en el primer sistema de Realitat Augmentada immersiva construït en 1992. Aquest projectava la visió d'uns braços robòtics sobre els braços de l'usuari.

imatge 7 - Louis B. Primer sistema de
Realitat Augmentada immersiva



Morton Heilig

Morton Leonard Heilig (desembre 22, 1926 - maig 14, 1997) va ser un home que va establir les bases del que Tom Caudell definiria com a Realitat Augmentada en la dècada dels noranta. Era un cinematògraf, filòsof i inventor, que va aplicar, conjuntament al seu equip, una experiència immersiva a través d'una màquina anomenada Sensorama en 1957.

El Sensorama és un dispositiu que compta amb un aspecte semblant a una màquina recreativa. En la part superior es pot albirar un visor per a introduir el cap. El visor utilitzava una pantalla estereoscòpica, a color, que reproduïa imatges tridimensionals donant la sensació a l'usuari d'estar dins de la imatge. L'aparell també comptava amb ventiladors, emissors d'olors i un sistema de so en estèreo per a crear una sensació encara més realista.

Per a poder prendre imatges idònies per al Sensorama, Morton Heiling va dissenyar una càmera amb la qual gravava pel·lícules amb sensació de tres dimensions. Constava d'una cambra frontal amb altres dos laterals per a crear imatges perifèriques.

El Sensorama també comptava amb un panell de control que permetia interactuar amb la pel·lícula, creant aquesta sensació immersiva.

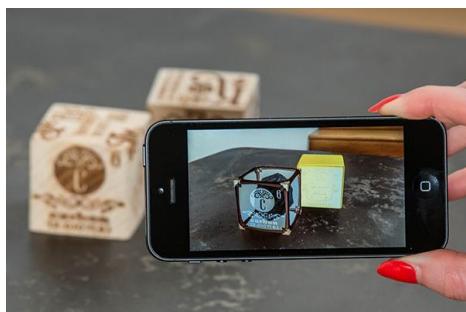


imatge 8 - Màquina de Morton Heiling (Sensorama)

4. Usos: RA

Educació

La Realitat Augmentada es pot aplicar a l'educació, és a dir per a l'ensenyament. Un exemple d'això, és una aplicació anomenada Elements 4D, que permet reconèixer els marcadors (proporcionats pels desenvolupadors) en unes figures per a superposar imatges sobre els elements químics de la taula periòdica.



Imatge 9 - RA aplicada a l'educació. APP amb elements químics amb RA

Vehicles

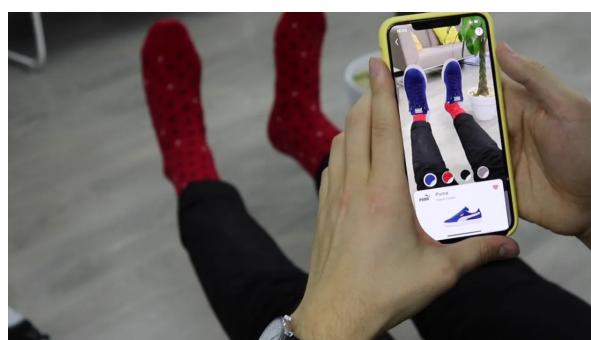
Existeixen algunes aplicacions (en desenvolupament) o objectes (GPS) que proporcionen dades de navegació a temps real, com a guies tridimensionals de direcció, punts d'interès, carrils, semàfors, etc. Aquestes es basen en una imatge real (el que es veu al seu voltant). Un exemple és el GPS AVIC-VH09CS, que inclou un sistema implantat, que permet tot l'esmentat anteriorment. Aquest es connecta de manera inalàmbrica en la part del mirall central o quadre de comandament del vehicle, i comença a captar informació per mostrar-la a través d'una pantalla.



Imatge 10 - RA aplicada als Vehicles. GPS amb un sistema de RA implementat

Moda

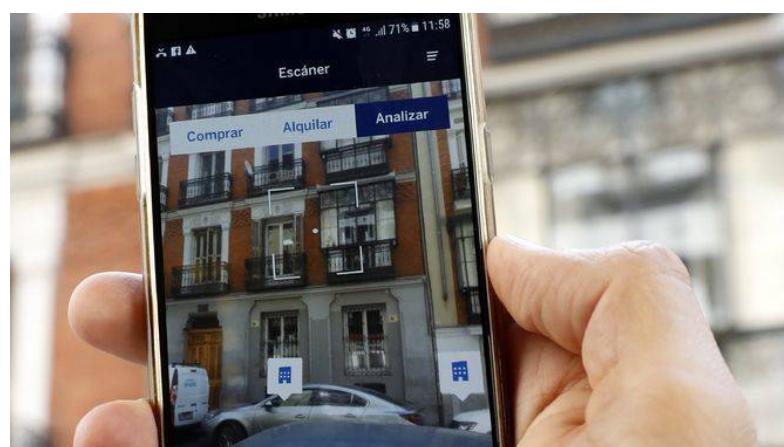
La Realitat Augmentada, ens permet obtenir informació virtual a l'apuntar amb el mòbil a l'etiqueta d'una peça o a un catàleg. A més, existeix la possibilitat de provar-se roba virtualment, enfocant el nostre cos amb la càmera del dispositiu, i deixant que l'aplicació superposi en temps real, com quedaría la peça. Un exemple és l'aplicació Wanna Kicks, que permet provar-se sabates gràcies a models 3D, proporcionats per les grans marques, com Nike o Adidas. De moment només compta amb models limitats, perquè l'aplicació es troba en desenvolupament.



imatge 11 - RA aplicada a la Moda. App de sabates amb RA

Sector Immobiliari

La Realitat Augmentada pot ser útil en el sector immobiliari. Una aplicació que demostra això, és BBVA Valora View. Aquesta permet a l'usuari visualitzar informació a temps real, simplement enfocant amb el dispositiu a l'habitatge. A més permet realitzar una estimació d'hipoteques o lloguers.



imatge 12 - RA aplicada a Immobiliària. App d'habitacions amb RA

Decoració

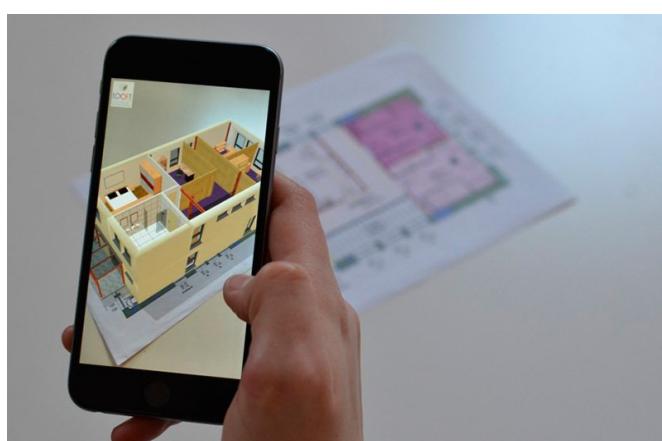
La Realitat Augmentada té la seva funció en la decoració. Un exemple d'això, és una aplicació desenvolupada per IKEA anomenada IKEA Place, que permet escanejar i reconèixer dimensions dels seus productes tipus moble i veure'ls (superposar-los) de manera artificial sobre la nostra realitat, i així saber si és el tipus de producte que es desitja i com es veuria en obtenir-ho.



imatge 13 - RA aplicada a Immobiliària. App d'habitatges amb RA

Arquitectura i Construcció

La Realitat Augmentada té un gran paper en l'arquitectura i construcció. Un exemple d'això, és Augment 3D, una aplicació que permet visualitzar un model 3D completament personalitzat, a través d'enfocar, amb el dispositiu, el pla de l'estructura.



imatge 14 - RA aplicada a Immobiliària. App d'habitatges amb RA

Logística

La Realitat Augmentada té la seva funció dins de la Logística. Aquesta permet agilitzar qualsevol procés i ajudar als operaris amb la gestió dels magatzems. Els ajuda amb el procés de cerca i selecció d'articles en un magatzem per a preparar o emmagatzemar vehicles. Normalment, cada empresa compta amb el seu propi sistema, per tant, no és possible trobar una aplicació com a tal, de cara al públic. Un exemple (amb ulleres de Realitat Augmentada):



Imatge 15 - RA aplicada a Immobiliària. App d'habitatges amb RA

Videojocs i Oci

La Realitat Augmentada té un gran ús en l'apartat de l'entreteniment. Un dels exemples més destacats, és Pokémon GO!

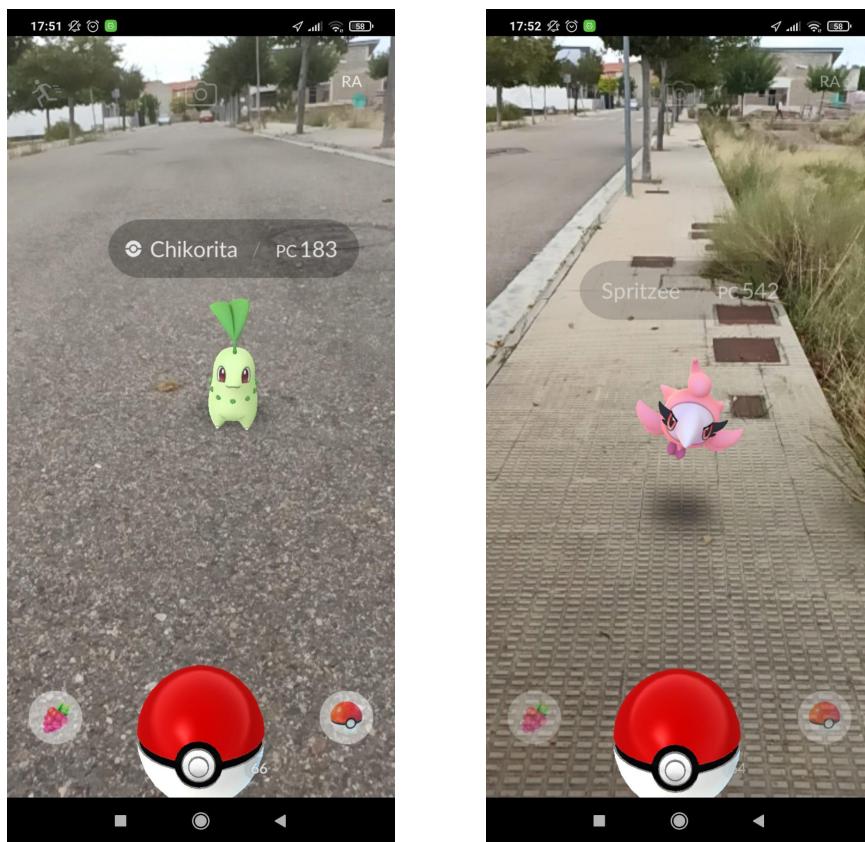
Exemple: Pokémon GO!

Un clar exemple de com podem usar la realitat augmentada és el famós Pokémon GO!, videojoc dissenyat per Nintendo que va usar aquesta tecnologia per a contentar a tots els fans de la saga, que durant anys van somiar amb poder veure Pokémoms en el món real.

El joc és senzill, et crees el teu propi personatge i gràcies a la detecció d'ubicació de tots els dispositius mòbils, l'aplicació pot situar-se en el mapa, utilitzant com a base, el Google Maps. A mesura que camines en el món real, el teu personatge segueix el mateix rumb i velocitat que tu.

Aquí és on entra la Realitat Augmentada, en detectar un Pokémon en el radar i pulsar sobre ell, aquest crea una imatge artificial sobre el fons real que pot veure el nostre telèfon. Utilitzarem la pantalla del mòbil per llançar la Pokeball i atrapar així al Pokémon. La localització i la grandària del Pokémon són importants en aquest aspecte, així que respecten les seves mesures i la zona on estigui. El tipus voladors no tocaran el sòl (Referència IMG 2) i els petits respectaran la seva grandària (Referència IMG 1).

Gràcies a la Realitat Augmentada podem gaudir d'un videojoc que barreja una cosa intangible i irreal, amb el nostre món, usant el paisatge que veiem enfront dels nostres ulls.



imatges 16/17 - RA aplicada a Videojocs. App d'entreteniment (videojoc) amb RA. Exemple Pokémon GO!

Part Pràctica

5. Objectiu

La finalitat d'aquest projecte pràctic consisteix a demostrar com funciona la Realitat Augmentada, mitjançant un mitjà d'entreteniment, és a dir, un videojoc. Així, és més senzill d'explicar per a qualsevol mena d'edat, ja que els videojocs són lineals i t'obliguen a seguir un camí i aprendre les tècniques per a arribar al final.

La idea principal del videojoc, es basa completament en un *Scape Room*, on el jugador, haurà de passar per diferents sales, amb diferents puzzles, per poder avançar. La gràcia d'aquest projecte, és incloure elements de Realitat Augmentada, és a dir, combinar "hardware" (realitat) amb "software" (fictici) al mateix temps.

Amb tot això, el que obtinc és un coneixement sobre C#, sobre com funciona un videojoc internament, disseny i creació de mapes (Tiled) i finalment el funcionament d'un motor gràfic (Unity) i com són capaços de combinar-se entre ells.



6. Aplicacions: ARMansion

Per a desenvolupar ARMansion, he utilitzat el motor de videojocs Unity, que és programa en C# mitjançant Visual Studio. A més, per a la creació de mapes he fet servir Tiled. A continuació es detallen en profunditat.

Motor Gràfic: Unity

Unity és un motor gràfic que s'utilitza per a fer videojocs, creat per Unity Technologies. Les característiques principals que conformen Unity són que es pot utilitzar juntament amb Blender, 3DS Max, Adobe Photoshop, Adobe Fireworks... A més, el motor gràfic fa ús de OpenGL, OpenGL ES, Direct3D i interfícies propietàries. Té suport de mapatge de relleu, reflexos, paralaje, oclusió ambiental en espai de pantalla,ombres dinàmiques, render de textures i efectes de post-processament de pantalla completa. Gràcies a la incorporació del llenguatge ShaderLab per a la creació de “shaders”, permet que pugui incloure múltiples variants i una especificació declarativa, la qual permet que pugui detectar la millor variant per a la targeta gràfica, i en cas de no ser compatible recórrer a uns altres que, malgrat que sacrifici rendiment, permeti la seva compatibilitat.

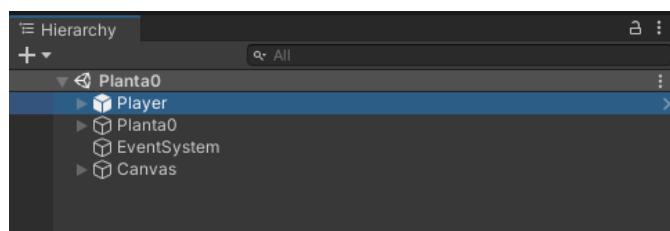
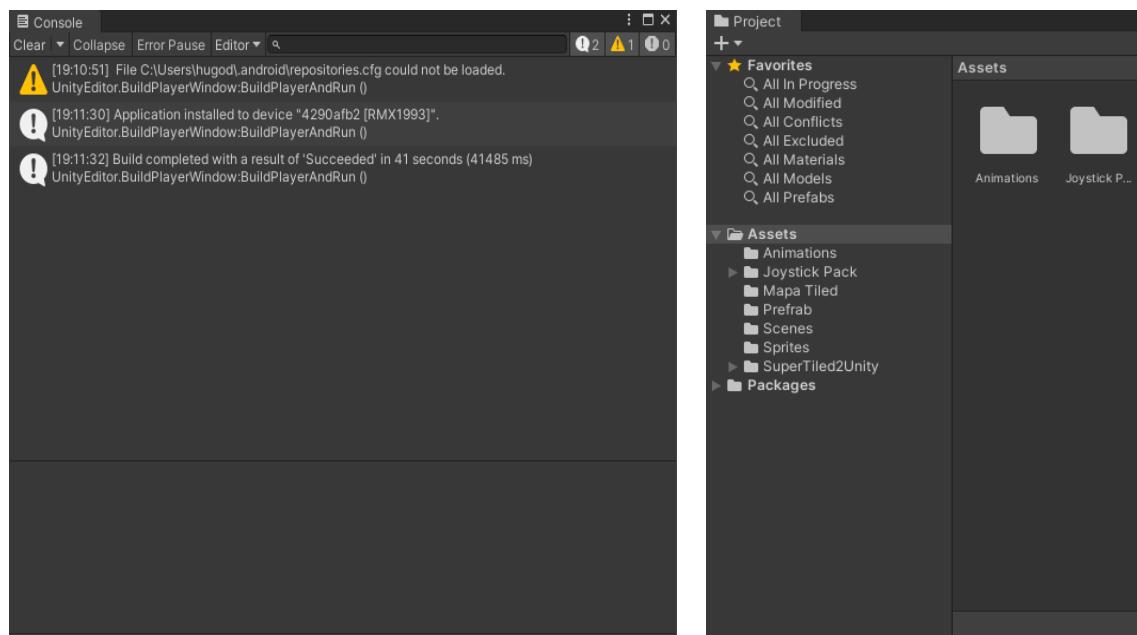
A més, compta amb suport integrat per a Nvidia, utilitzant el seu motor de físiques PhysX, el qual ofereix capes de col·lisió.

A través de Mico, recorre al scripting, per poder crear diferents “scripts” per als elements que es desitgin incloure en el projecte, ja sigui moviment, col·lisió, etc. Per a desenvolupar el scripting, es recorre al llenguatge de programació C#. A més Unity, compta amb Mecanim, un sistema per a poder dur a terme animacions amb moviment fluid i natural, amb una interfície clara i senzilla.

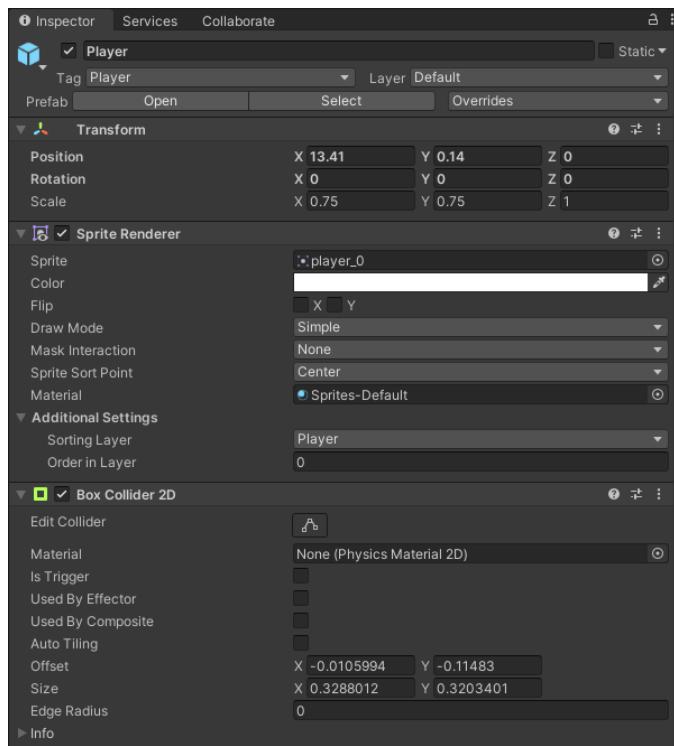
Finalment Unity, compta amb una pàgina web, on es pot trobar diferents recursos, com per exemple, “assets” (Assets Store) per a poder fer un projecte desitjat, a més de gran quantitat de guies, escrites i audiovisuals i biblioteca de sons. La interfície de Unity, està composta de la següent manera:

Unity compta amb una consola, on es mostren tots els errors commesos a l'hora de desenvolupar el projecte o durant la programació dels scripts. D'altra banda, també té un menú ben detallat, on es mostren tots els elements (animacions, escenes, llibreries, etc.) que l'usuari ha afegit al projecte i poder interactuar amb ells a temps real.

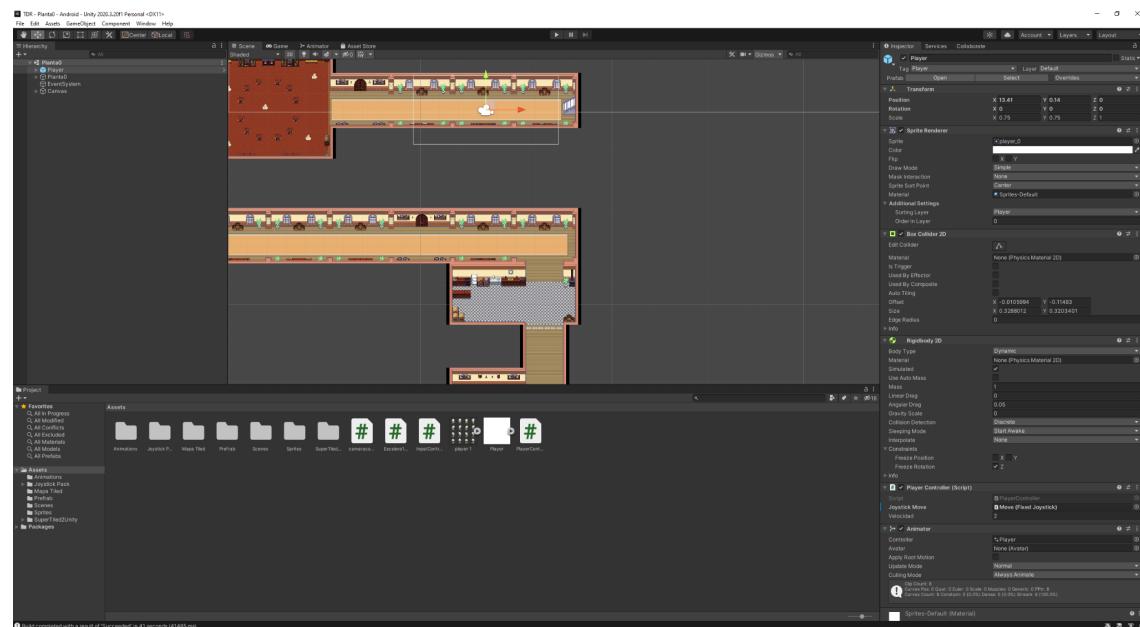
A més, compta amb una barra d'eines, amb diferents opcions com per exemple: pestanyes, per guardar, exportar, editar, ajuda, etc.



A la dreta de l'aplicació, es pot trobar un menú on es pot editar tot l'escrit en el codi del Script, fent referències, a més de poder editar les físiques com la gravetat, "colliders" (col·lisió de l'element seleccionat), aspecte de l'element, posició, etc.



Vista general de l'aplicació:

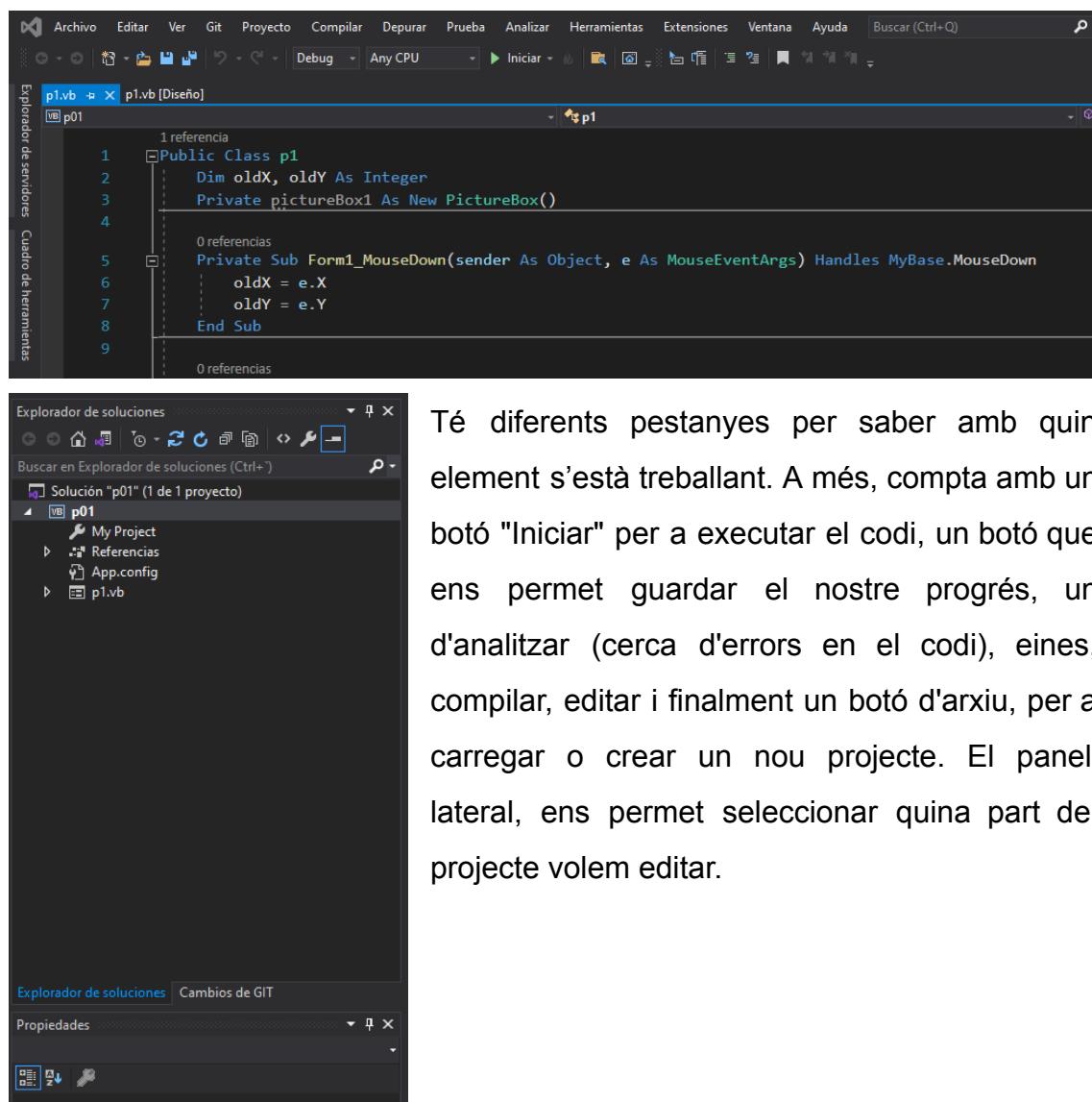


Codi: Visual Studio

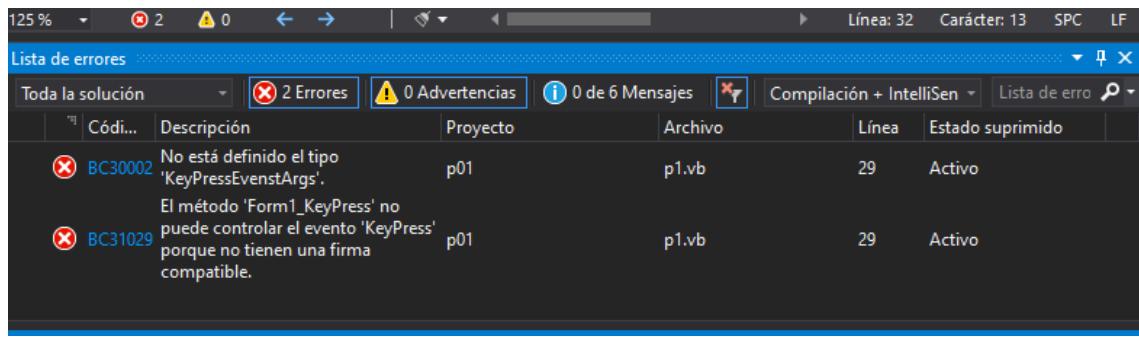
Microsoft Visual Studio és una aplicació que permet treballar amb múltiples llenguatges de programació (multiplataforma) com ara C, C++, C#, Visual Basic .NET, Java, Python, etc. Permet als desenvolupadors crear aplicacions / llocs web, compatibles amb la plataforma .NET

Visual Studio, s'ha anat actualitzant i modernitzant, per a afegir més llenguatges de programació, i anar actualitzant els que ja coexistien.

Visual Studio, compta amb una interfície molt senzilla, que ens permet treballar de forma organitzada.

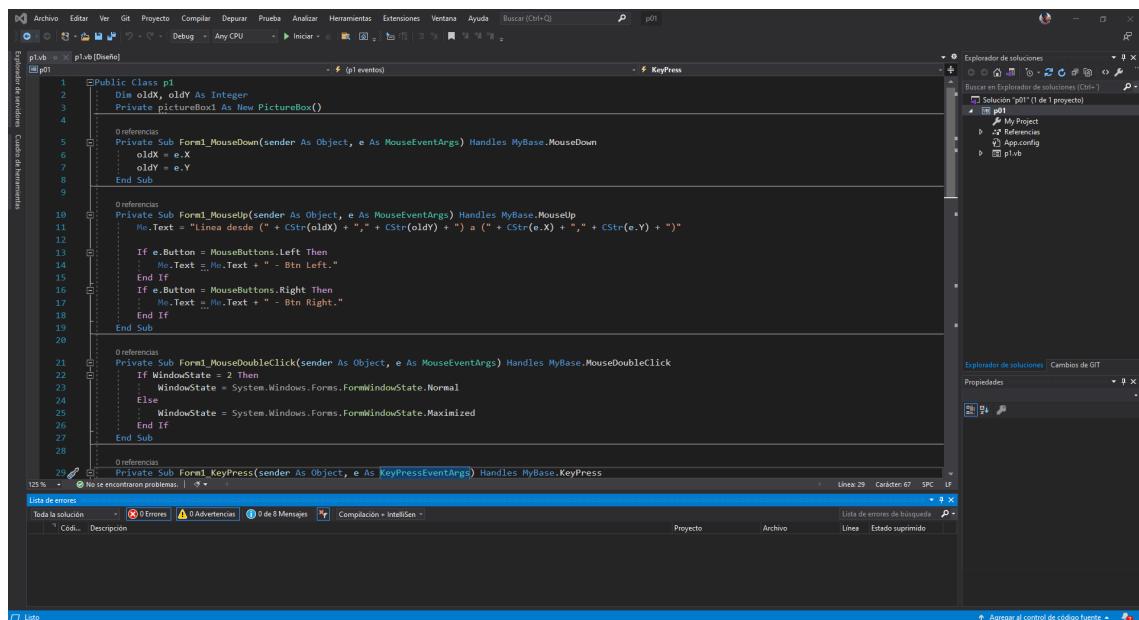


Té diferents pestanyes per saber amb quin element s'està treballant. A més, compta amb un botó "Iniciar" per a executar el codi, un botó que ens permet guardar el nostre progrés, un d'analitzar (cerca d'errors en el codi), eines, compilar, editar i finalment un botó d'arxiu, per a carregar o crear un nou projecte. El panell lateral, ens permet seleccionar quina part del projecte volem editar.



En la part inferior, mostra una llista d'errors o advertiments que no permeten que el codi funcioni correctament. Mostra la línia, l'error i la seva referència.

Vista general de l'aplicació:



Mapa: Tiled

Tiled és una aplicació que s'utilitza per a crear i editar mapes. S'adapta molt bé a l'hora de crear jocs 2D, ja que solen ser de plataformes o RPG's, que no necessiten gran detall en el seu escenari.

Els escenaris es creen a partir de colocar "sprites" en un cert mode, que solen ser retallats d'un "spritesheet", els quals són la unió de diferents píxels de colors, que donen forma a objectes, mobles i detalls. Aquests píxels són anomenats "tiles". Tiled suporta mapes ortogonals i isomètrics. A més, permet generar fitxers per a poder usar-los en qualsevol mena de motor gràfic, un exemple és Unity.



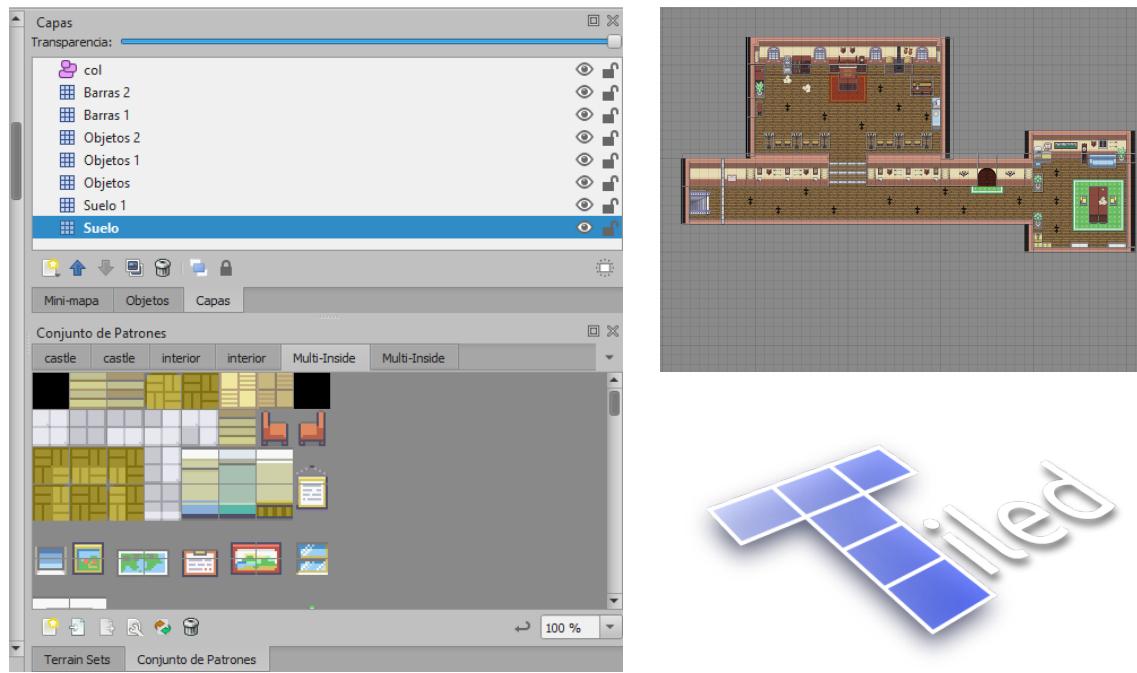
En poques paraules, Tiled ofereix una experiència senzilla i còmoda per a desenvolupar escenaris. Interfície:



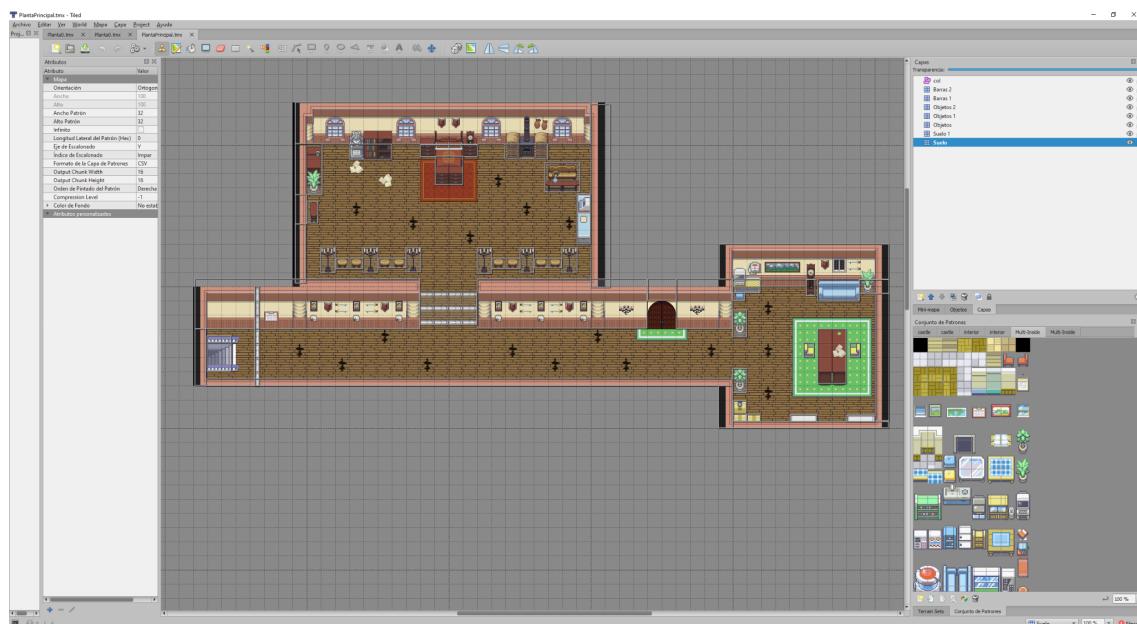
En la part superior del programa, es poden trobar les típiques opcions per afegir arxius o crear nous, editar el contingut, canviar la visualització de la interfície, treballar per capes, etc. A més, compta amb pestanyes, que permeten treballar amb diferents mapes alhora. Finalment, compta amb eines que permeten fer un mapa, ja sigui esborrador, farcit, estampat, etc.

En la part dreta del programa, es pot visualitzar una secció que ens permet separar les textures "tiles" de capa, per a oferir una major organització. D'altra banda, en la part inferior, se situa el conjunt de patrons, on se situen tots els

“tiles” que vol utilitzar l’usuari per a realitzar el seu mapa (aquests, poden ser descarregats d’internet. Es troben en diverses pàgines webs, com per exemple “OpenGameArt” que és una biblioteca de recursos de llicència gratuïta per a videojocs.



Finalment, en el centre, se situa la quadrícula amb la qual l’usuari pot treballar, és a dir, portar a cap el seu mapa. Vista general de l’aplicació:



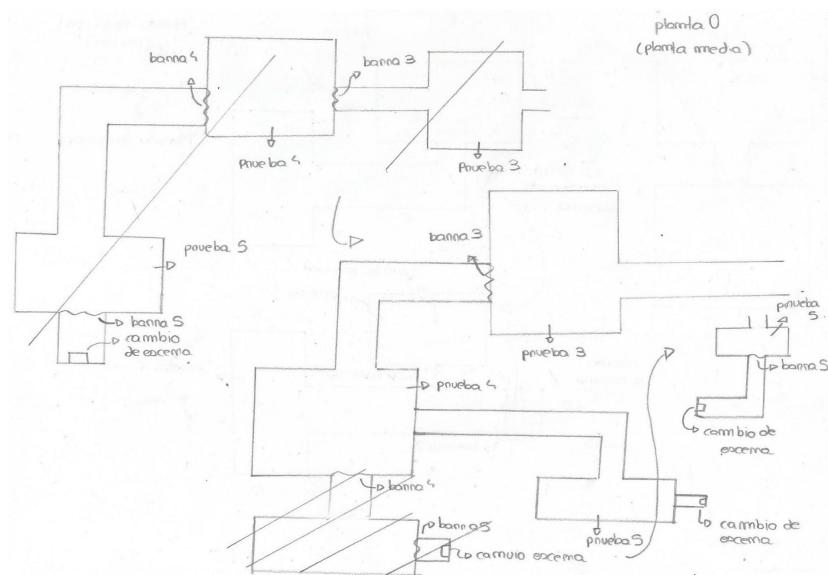
7. Desenvolupament: ARMansion

Per a realitzar una bona explicació sobre com he realitzat el videojoc, el dividiré en dos grans apartats, els quals estaran dividits en apartats més específics.

Apartats Generals

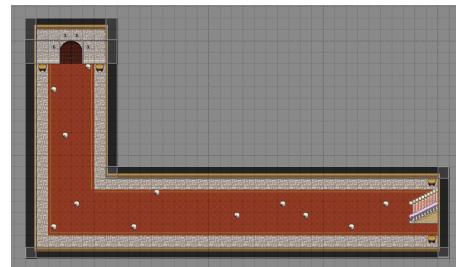
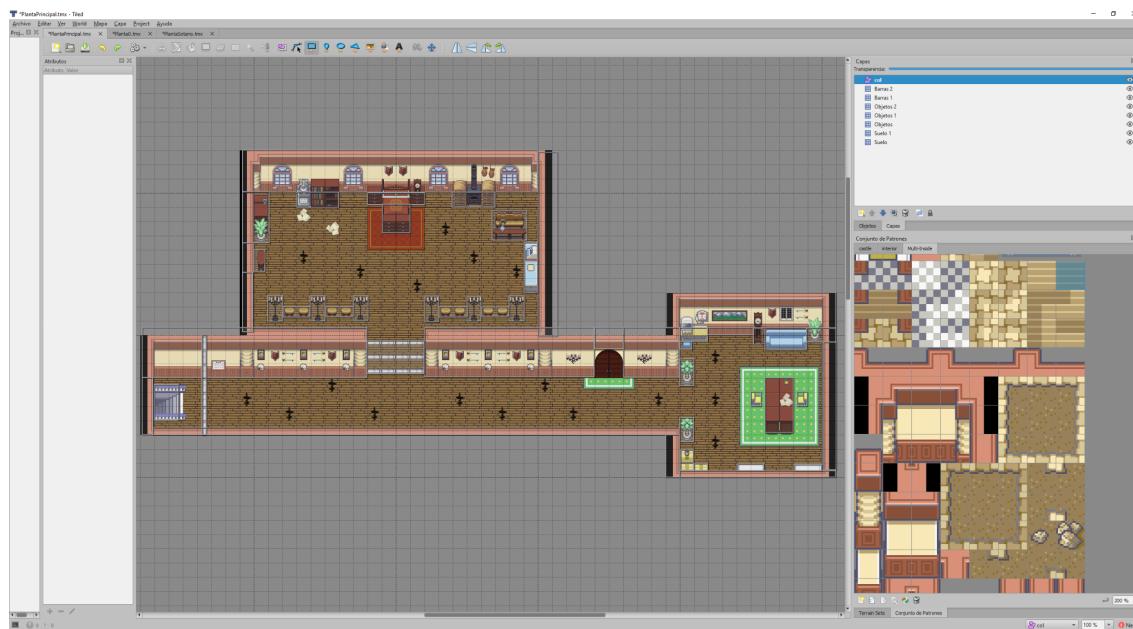
Mapes

Per al disseny dels mapes, primer vaig pensar en la temàtica del joc, i vaig decidir que se situés en una mansió abandonada. Posteriorment, vaig dibuixar a mà alçada alguns esbossos per a fer-me una idea de com seria l'escenari a crear, i així estructurar-lo correctament. En total vaig efectuar tres esbossos, un per cada planta de la mansió.

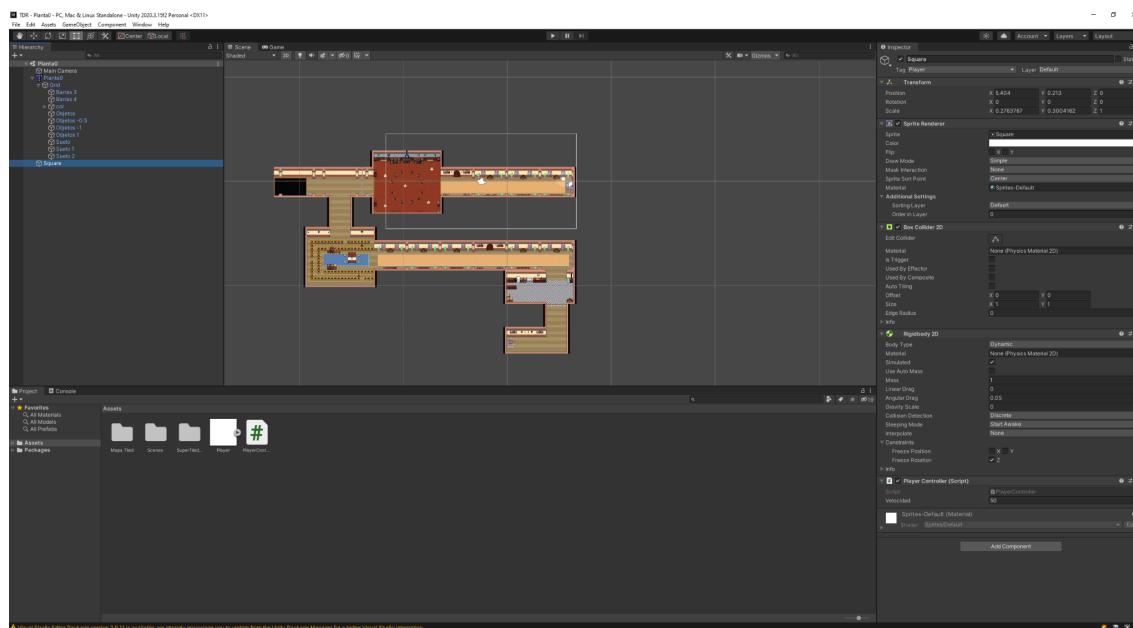
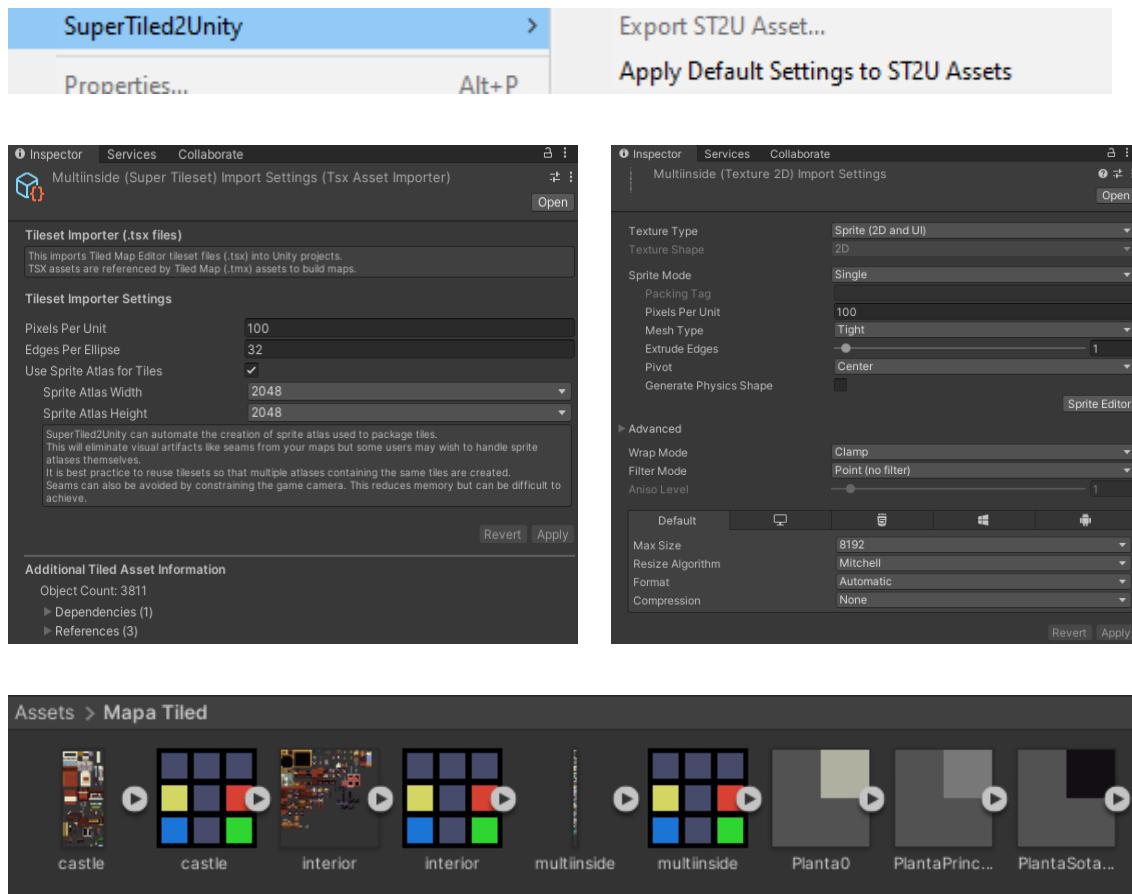


Una vegada decidit el disseny, vaig passar al programa esmentat anteriorment, Tiled. En aquest programa, has d'adjuntar unes imatges que continguin "tiles", en el meu cas vaig utilitzar uns proporcionats per la pàgina web: OpenGameArt. Seleccions el "tile" i ho col·loques sobre el pla. En unir un "tile" amb un altre, pots crear escenaris amb decoracions separats per capes per a establir una profunditat.

En el meu cas, el primer que vaig fer va ser realitzar l'estructura del mapa amb un sol nivell, per a saber sobre quina àrea treballaré. Una vegada feta l'estructura, vaig afegir els murs formant límits, agregant la decoració i il·luminació. Finalment, vaig fer una capa de col·lisions, per a afegir "colliders" que impedeixen que el jugador abandoni el mapa, travessant parets o objectes, i a més creant una sensació de profunditat (diferents altures) en els "tiles".



Una vegada crea els mapes, els vaig importar a Unity mitjançant la llibreria anomenada “SuperTiled2Unity”. Cal destacar que per a importar-los de manera correcta, és necessari afegir també les imatges originals on es troben els "tiles", i ajustar la grandària d'aquests perquè Unity els reconegui correctament.



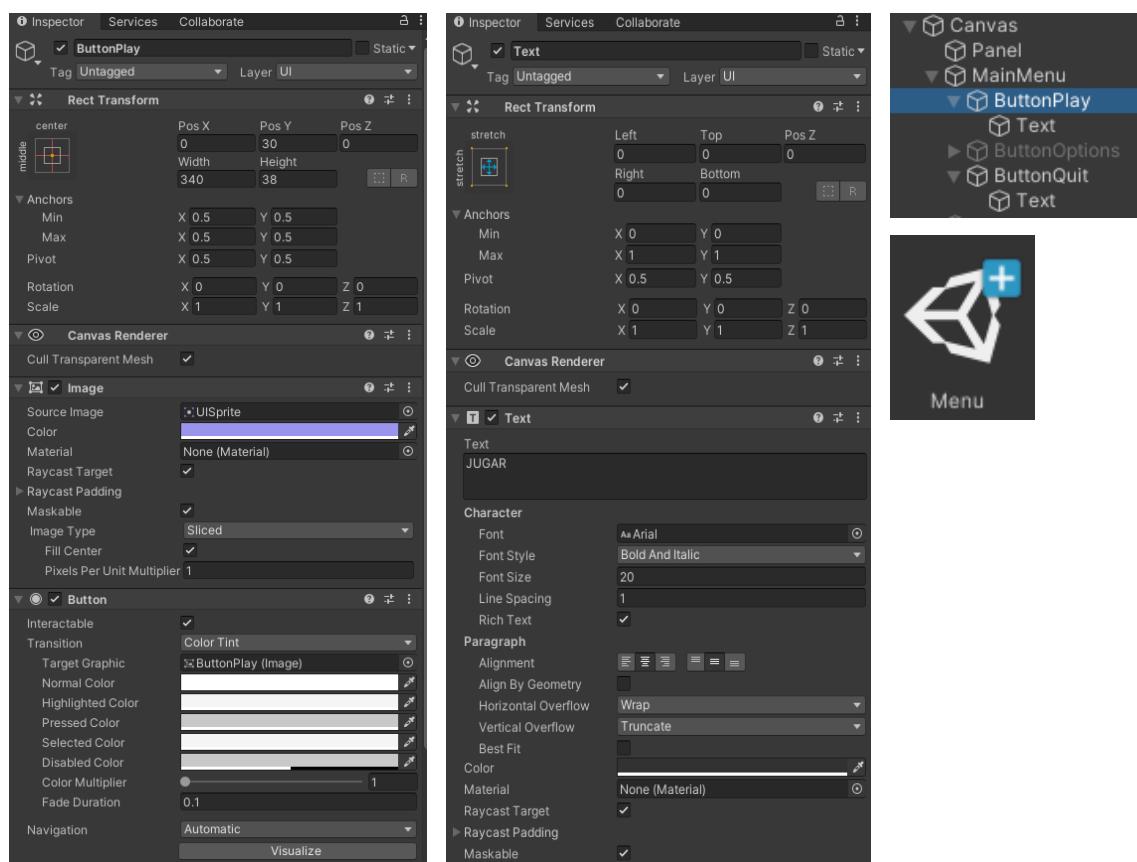
Una vegada fet això, els mapes estaran correctament importats a Unity.

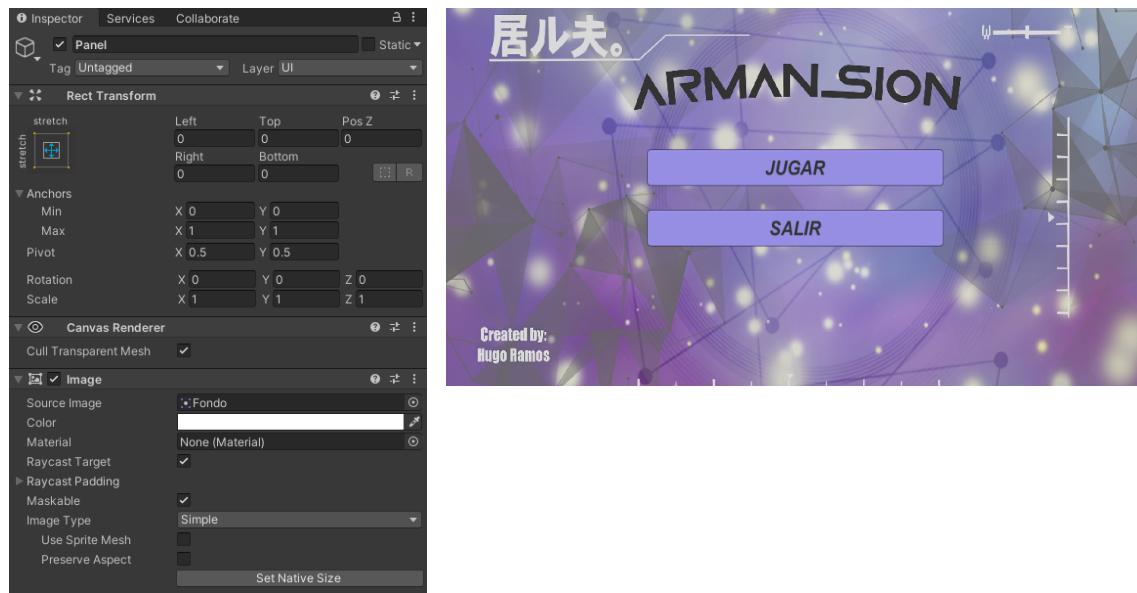
UI - User Interface (Interfície d'Usuari)

La interfície de l'usuari es basa en els menús. Compta amb botons que tenen per objectiu "Jugar", "Salir", "Volver" i "Reanudar" afecten el joc en si, en canvi, el botó "Ayuda" té per objectiu accedir a un apartat que conté pistes per a resoldre les proves del joc.

El joc compta amb dos menús. El menú principal és aquell que s'albira res més iniciar el joc, permetent-nos endinsar-nos dins del joc o abandonar-lo. Per a realitzar aquest menú, vaig utilitzar l'eina "Canvas" de Unity. Aquest ens permet agregar elements d'interfície d'usuari com a textos, botons, llistats, etc.

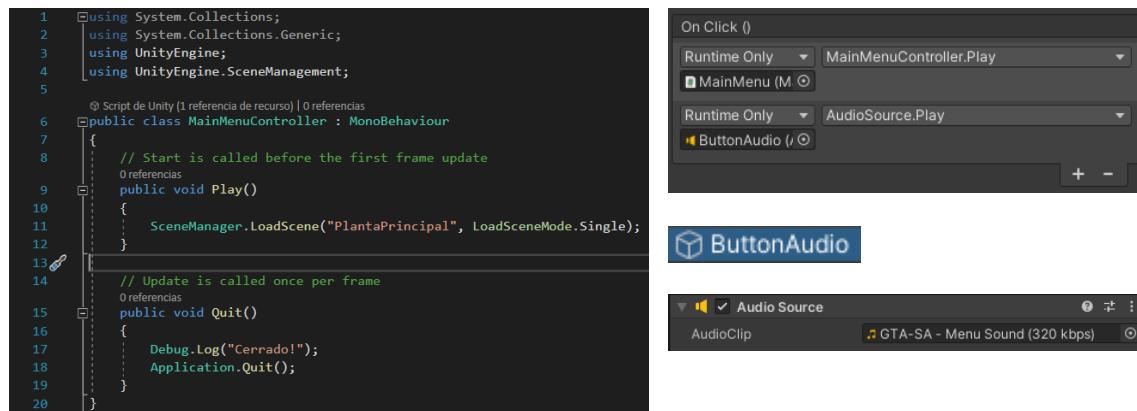
Per a crear el menú principal, primer vaig crear una nova escena anomenada "Menu". Aquesta compta amb un "Canvas", que conté un panell (element de UI que ens permet afegir una imatge). Posteriorment, vaig afegir al "Canvas" una carpeta anomenada "MainMenu" per a afegir els botons de "Jugar" (ButtonPlay) i "Salir" (ButtonQuit).





El codi del menu esta a `MainMenuController`. Conté dues funcions públiques, una funció de `(Play())` i una altra de `(Quit())`. Una funció pública ens permet referenciar-la des d'un objecte, en aquest cas els botons, des de Unity. La funció `(Play())`, fa un canvi d'escena a "PlantaPrincipal", que és l'escena inicial del joc. Mentre que la funció `(Quit())`, tanca l'aplicació.

Els botons disposen d'un esdeveniment `(onClick())`, al qual se li poden vincular funcions que s'executarán quan aquest esdeveniment s'emeti. Això fa que el "script" (`MainMenuController`), executi la funció `(Play())`. Per al botó de Sortir, és exactament el mateix, només que es dirigirà a la funció `(Quit())`.



A més, vaig afegir a l'esdeveniment efectués un so quan es premés el botó "Jugar", afegint-li un component d'àudio "Audio Source" i reproduueixi el so.

D'altra banda, el joc compta amb un menú de pausa. Aquest menú està format per tres botons, "Continuar", "Ayuda" i "Salir del Juego". El procés de disseny és exactament el mateix que el del menú principal, només que aquests botons estan agrupats en una carpeta diferent anomenada "PauseMenu".

El codi (PauseMenuController) efectuat en aquest menú és una mica més complex que en (MainMenuController). Per a començar, vaig designar les variables. La primera es tracta d'una variable de tipus "bool" (pot donar dos valors: true (cert) o false (fals)) pública.

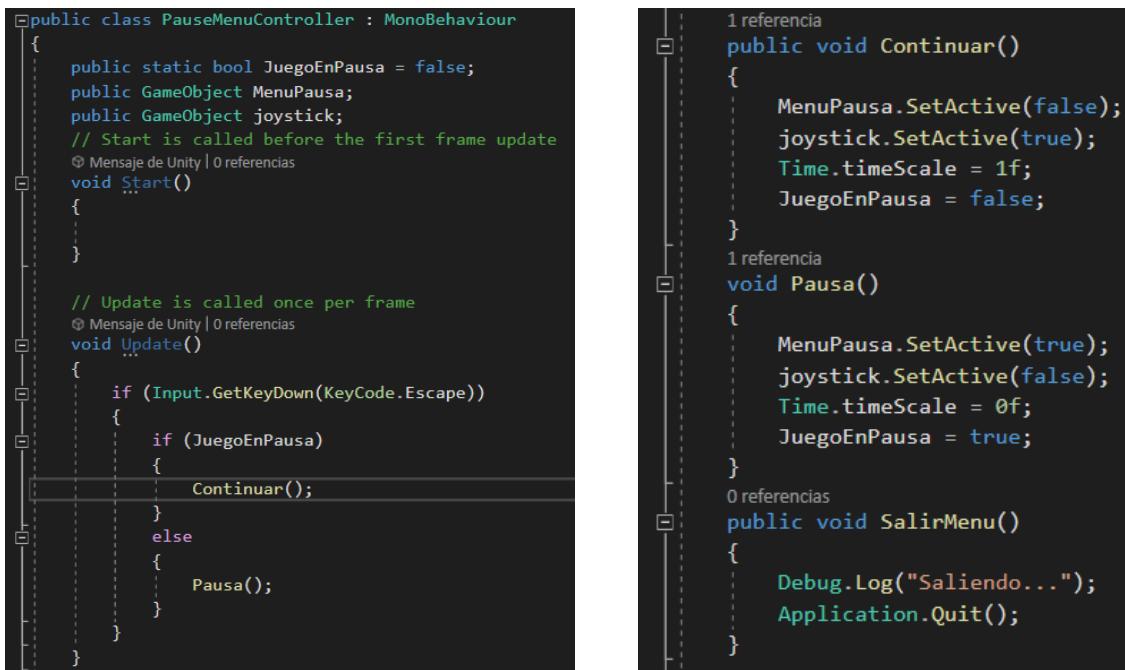
D'una banda, cal referenciar en el codi, que existeixen dos objectes dins de Unity, de tipus "GameObject". En el meu cas els vaig anomenar "MenuPausa" i "joystick". Per a començar, el codi lanza en la funció (void Update()) (el codi revisa de manera constant si s'està executant l'ordre escrita) si la tecla "esc" es prem, en cas que sigui així, que comprovi si s'estan realitzant les següents opcions amb un condicional. Si la variable (JuegoEnPausa = false), el joc continua, però si es prem la tecla "esc", vol dir que la variable (JuegoEnPausa = true) i, per tant, el joc s'aturarà.

Quan el joc s'atura, executa la funció (Pausa()), la qual activa el menú de pausa (MenuPausa.SetActive(true)), el "joystick" es desactiva (joystick.SetActive(false)) i el temps es congela en l'estat actual (Time.timeScale = 0f). Mentre que quan no està pausat executa la funció (Continuar()) i, per tant, el menú de pausa està desactivat (MenuPausa.SetActive(false)), el "joystick" (joystick.SetActive(true)) s'activa i el temps es descongela (Time.timeScale = 1f).

Com anteriorment, cal referenciar a Unity que objecte és "MenuPausa" i "joystick". L'objecte "PauseMenu" és "MenuPausa" i l'objecte "Move" és "joystick".

Una vegada fet això, simplement he creat un esdeveniment (onClick()) en el botó de "Continuar", perquè quan es premi, executi en el Script (PauseMenuController) la funció de (Continuar()). Per al botó de "Salir del Juego" busca en el mateix "script" la funció de (SalirMenu()).

Finalment, en el botó "Ayuda", vaig configurar un altre esdeveniment (onClick()) que funciona d'una manera distinta, ja que posseeix un apartat diferent anomenat "PausaMenuAyuda", on s'inclou una imatge amb les pistes per a realitzar les proves i un botó anomenat "Volver". Aquest actua com un "bool". Si es prem el botó "Ayuda", desactiva "PauseMenu" (false) i activa "PausaMenuAyuda" (true). El botó "Volver" executa la funció contrària.



```

public class PauseMenuController : MonoBehaviour
{
    public static bool JuegoEnPausa = false;
    public GameObject MenuPausa;
    public GameObject joystick;
    // Start is called before the first frame update
    void Start()
    {
    }

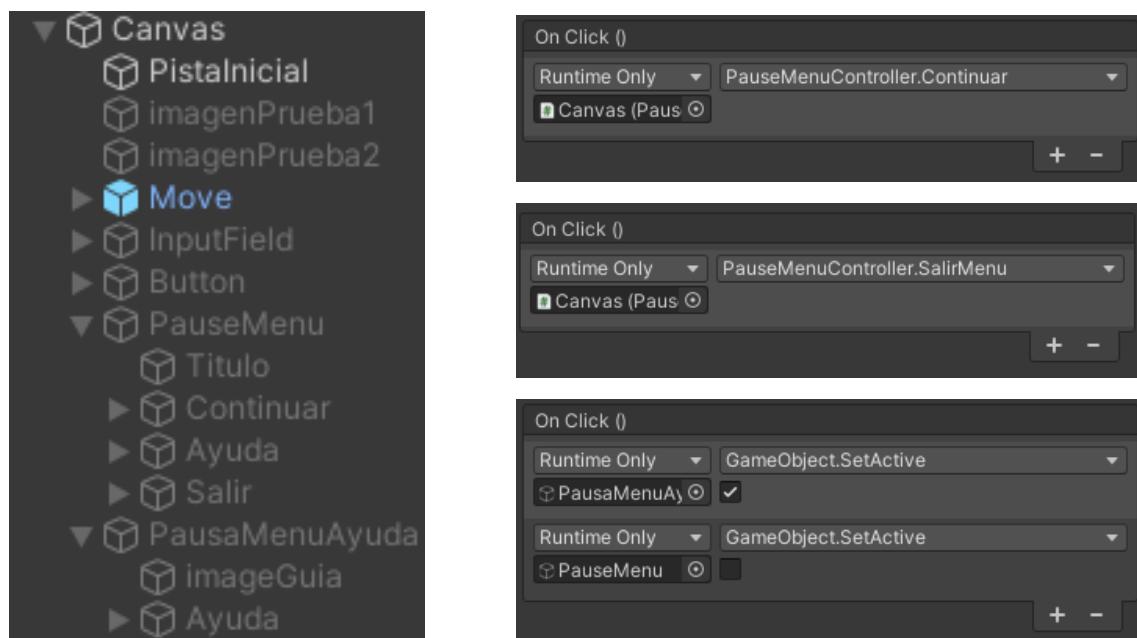
    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (JuegoEnPausa)
            {
                Continuar();
            }
            else
            {
                Pausa();
            }
        }
    }
}

1 referencia
public void Continuar()
{
    MenuPausa.SetActive(false);
    joystick.SetActive(true);
    Time.timeScale = 1f;
    JuegoEnPausa = false;
}

1 referencia
void Pausa()
{
    MenuPausa.SetActive(true);
    joystick.SetActive(false);
    Time.timeScale = 0f;
    JuegoEnPausa = true;
}

0 referencias
public void SalirMenu()
{
    Debug.Log("Saliendo...");
    Application.Quit();
}

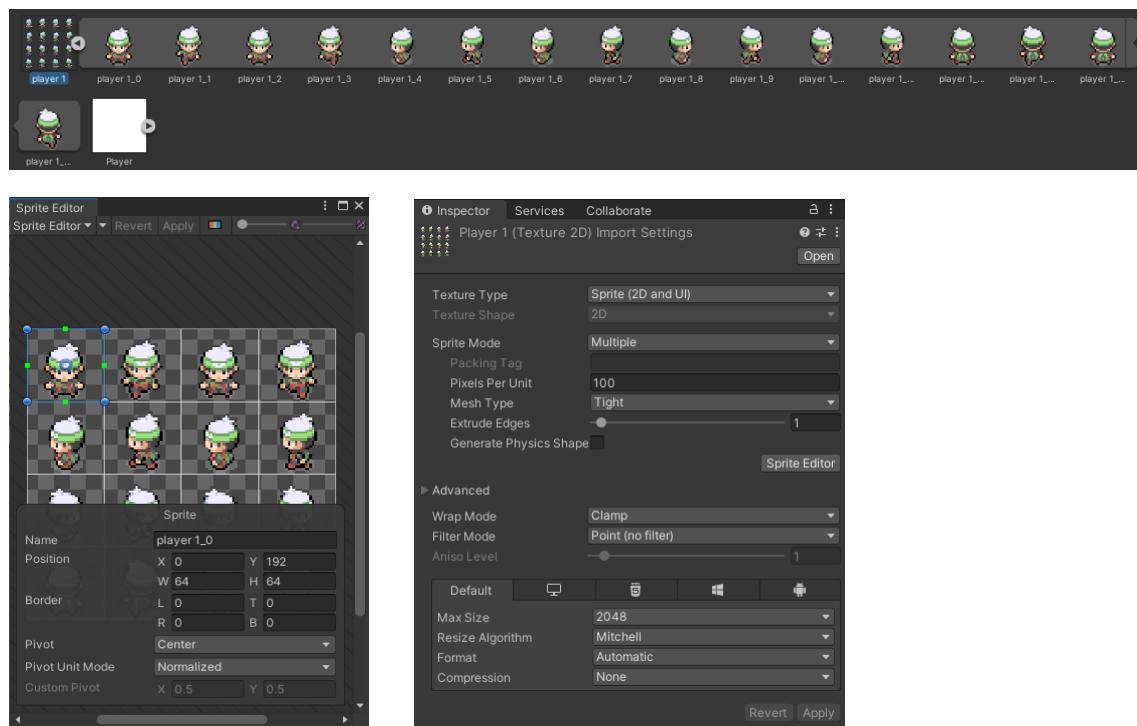
```



Apartats Específics

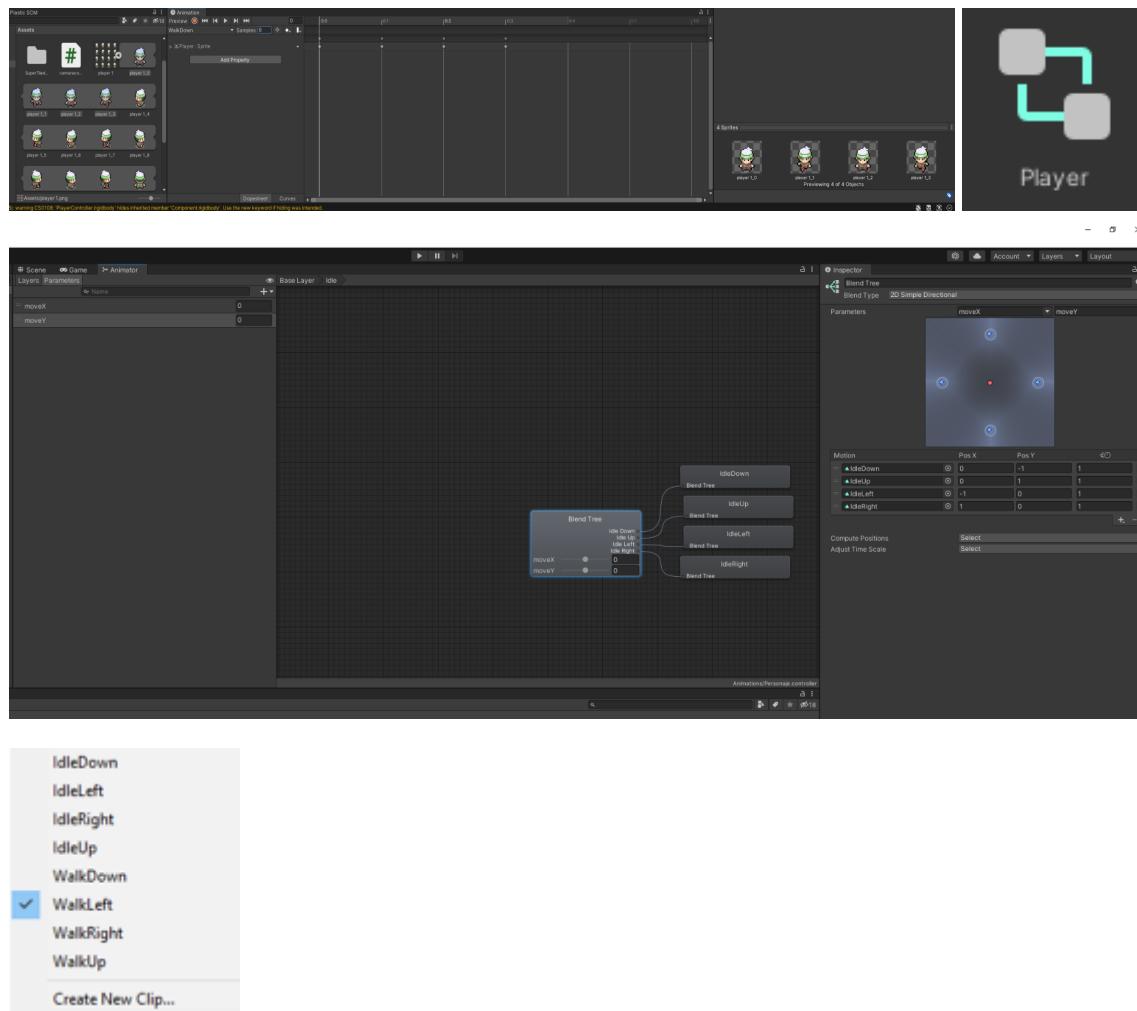
Jugador i Moviment (animacions)

El jugador es tracta d'un "sprite", és a dir, un model 2D basat en una imatge. Posseeix un arxiu ("spritesheet") on es troben totes les imatges necessàries per a realitzar les animacions. Primer de tot, cal dividir el "spritesheet" en "sprites". Per a això s'ha de posar el mode del "sprite" en "Multiple", amb una eina que Unity posseeix. El vaig dividir en 64 píxels, quedant-me un total de 16 "sprites".



Una vegada fets els "sprites", vaig realitzar les animacions de moviment del jugador. Vaig fer ús d'un recurs proporcionat per Unity que permet agrupar diverses imatges per segon, és a dir, diversos fotogrames ("sample"). En el meu cas, vaig emprar un "sample" de sis segons. Per a cada animació, comptava amb quatre imatges que simulen el moviment del jugador en totes les direccions. En total vaig fer cinc animacions per al personatge, quatre animacions són per al moviment, i una altra per a quan està quiet. Una vegada realitzades les animacions, Unity crea un arxiu on s'agrupen totes les animacions. Dins d'aquest arxiu, vaig fer dos paràmetres, un d'ells per al moviment en l'eix horitzontal (moveX) i un altre per a l'eix vertical (moveY).

Aquests paràmetres compten amb un "Blend Tree" en 2D, que permet afegir quatre animacions. Una vegada fet això, vaig ajustar els valors referenciant-me en un eix de coordenades amb (-X,+X,-Y,+Y).



Per al moviment del jugador, vaig voler utilitzar un “joystick”. Per a això, em vaig descarregar una llibreria, anomenada "Joystick Pack". Aquesta permet afegir al "Canva" un "joystick" analògic.

Una vegada efectuats tots els preparatius, és a dir, animacions i UI "joystick", li vaig afegir components al "sprite" del jugador. Un d'ells es diu "Rigidbody2D" que permet que el jugador col·lisioni amb elements del mapa. Posteriorment, vaig fer un "script" per al jugador anomenat (PlayerController). Dins d'aquest "script", vaig realitzar diferents variables, les quals són: "jostick", anomenada (joystickMove), "sprite" jugador (rb), (Vector2), assignant-li que sigui un valor d'entrada "input", velocitat i finalment (Animator), una variable per a l'animació.

Cal esmentar, que les variables del "sprite" jugador (rb) i (Animator) són privades, això vol dir que no permet referenciar-les des de Unity, sinó des del codi. El codi inclou dues funcions, (void Start()), que dicta que ha de fer el codi en iniciar-se, i el (void Update()) que mira fotograma a fotograma si es compleix el que hi ha escrit en el seu interior. En el (void Start()) el codi fa una crida, per a obtenir els objectes (rb) i (Animator). Una vegada fet això, realitza la funció (void Update()). Dins d'aquesta podem trobar dues variables de tipus "float" (variable numeral) igualades a la funció del "jostick", això vol dir que en fer un desplaçament del "joystick" horitzontal o vertical, el codi aconsegueix un número i amb aquest, crea una nova variable, anomenada (rb.velocity) multiplicant tots dos números (tant l'horitzontal com el vertical) per la velocitat (variable velocitat). Una vegada obté el valor de (rb.velocity), comprova si aquesta magnitud és superior a zero, amb una variable de tipus "bool" anomenada (isMoving). Si la variable = true, perquè el valor de (rb.velocity) és superior a zero, assigna els paràmetres (moveX, moveY) on es troben les animacions per al moviment horitzontal i vertical.

```

public Joystick joystickMove;
private Rigidbody2D rb;
Vector2 input;
public float velocidad;
private Animator animator;
private bool isMoving;
CharacterController controller;

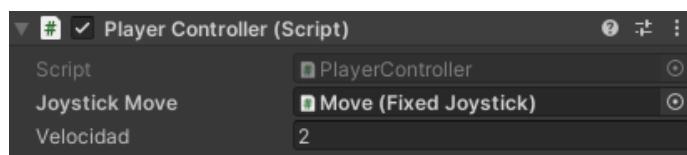
void Start()
{
    rb = GetComponent<Rigidbody2D>();
    animator = GetComponent<Animator>();

}

void Update()
{
    float moveHorizontal = joystickMove.Horizontal;
    float moveVertical = joystickMove.Vertical;
    rb.velocity = new Vector2(moveHorizontal * velocidad, moveVertical * velocidad);
    animator.SetBool("isMoving", rb.velocity.sqrMagnitude > 0);
    animator.SetFloat("moveX", moveHorizontal);
    animator.SetFloat("moveY", moveVertical);
}

```

Una vegada explicat tot el codi, cal referenciar que objecte és el Joystick, i assignar-li un valor a velocitat.



Interaccions amb el Mapa

Les interaccions amb el mapa estan agrupades en el canvi de plantes i ajudes que he anat desplegant perquè el jugador pugui completar les proves.

Escales

Els canvis d'escena els he realitzat mitjançant "colliders" amb "scripts". He assignat un "collider" a cada escala. Aquest compta amb un "script" anomenat (EscaleraXController). El codi és el següent:

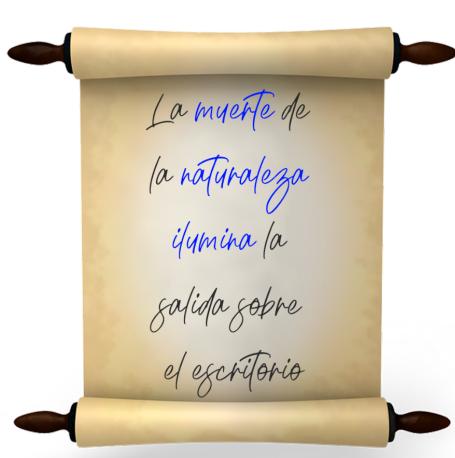
```
void OnTriggerEnter2D(Collider2D other) {
    //Debug.Log("hola");
    if (other.tag == "Player") {
        PlayerPrefs.SetFloat("PlayerX", 17);
        PlayerPrefs.SetFloat("PlayerY", 0.25f);
        SceneManager.LoadScene("Planta0", LoadSceneMode.Single);
    }
}
```

Unity posseeix una funció que quan un "collider" col·lisioni amb un objecte o "sprite", executi el que hi ha escrit a l'interior de la funció (void OnTriggerEnter2D(Collider2D other)). Perquè funcioni, el "collider" ha de ser un "trigger", és a dir que es pugui travessar. El codi expressa un condicional, en el qual si el "collider" de l'escala detecta el jugador ha col·lidit amb ell (if (other.tag == "Player")), executi un canvi d'escena (SceneManager.LoadScene("Planta0", LoadSceneMode.Single)), en la posició designada, en l'eix horitzontal i el vertical (PlayerPrefs.SetFloat("PlayerX", 17)) // (PlayerPrefs.SetFloat("PlayerY", 0.25f)). Aquest codi s'aplica per a totes les escales del mapa, però amb modificacions en la posició i escena que ha de carregar.

Calaveres

Per al sistema d'ajuda, és a dir, petites pistes que he anat col·locant pel mapa, he fet ús d'un model amb forma de calavera, canviant-li el color per blau, perquè sigui un color cridaner. El sistema és semblant al de les escales, però amb afegits i modificacions. Per a les pistes en les diferents proves, he creat una imatge en Photoshop amb enigmes escrits, perquè el jugador pensi que ha de fer. Per a adjuntar una imatge a la UI, he afegit al "Canva" un element de tipus "Image".

Una vegada afegida la imatge, he creat un "script" en les calaveres (CalaveraXController). El codi és el següent:



```
public GameObject imagenPrueba1;
// Mensaje de Unity | 0 referencias
void OnTriggerEnter2D(Collider2D other)
{
    if (other.tag == "Player")
    {
        imagenPrueba1.SetActive(true);
    }
}
// Mensaje de Unity | 0 referencias
void OnTriggerExit2D(Collider2D other)
{
    if (other.tag == "Player")
    {
        imagenPrueba1.SetActive(false);
    }
}
```

Primer de tot, el codi busca una variable de tipus "GameObject" pública, per a poder referenciar la imatge que es mostrarà. Posteriorment, executa la funció que quan el "collider" de la calavera detecta que el jugador ha col·lidit amb ell, activa la imatge corresponent (imagenPrueba1.SetActive(true)). En canvi, quan el "collider" de la calavera detecta que el jugador no està col·lidint amb ell, executa la funció (void (OnTriggerExit2D(Collider2D other))) desactiva la imatge corresponent (imagenPrueba1.SetActive(false)).

Aquest mateix codi s'aplica a totes les calaveres, només que canviant la referència de la imatge per la desitjada.

Proves i Final

El videojoc compta amb un total de cinc proves diferents. Per a explicar que he realitzat en cada prova, és a dir, com l'he organitzat i programat, em detindré a explicar-la una per una.

Prova 1 - Contrasenya "AR" amb Vuforia

L'objectiu de la primera prova és que el jugador explori l'entorn i resolgui un enigma. La prova se situa en l'inici del joc, on el jugador ha de trobar un codi situat a l'esquena d'una estàtua AR i després introduir-lo en un ordinador per a així poder avançar cap a la següent prova.



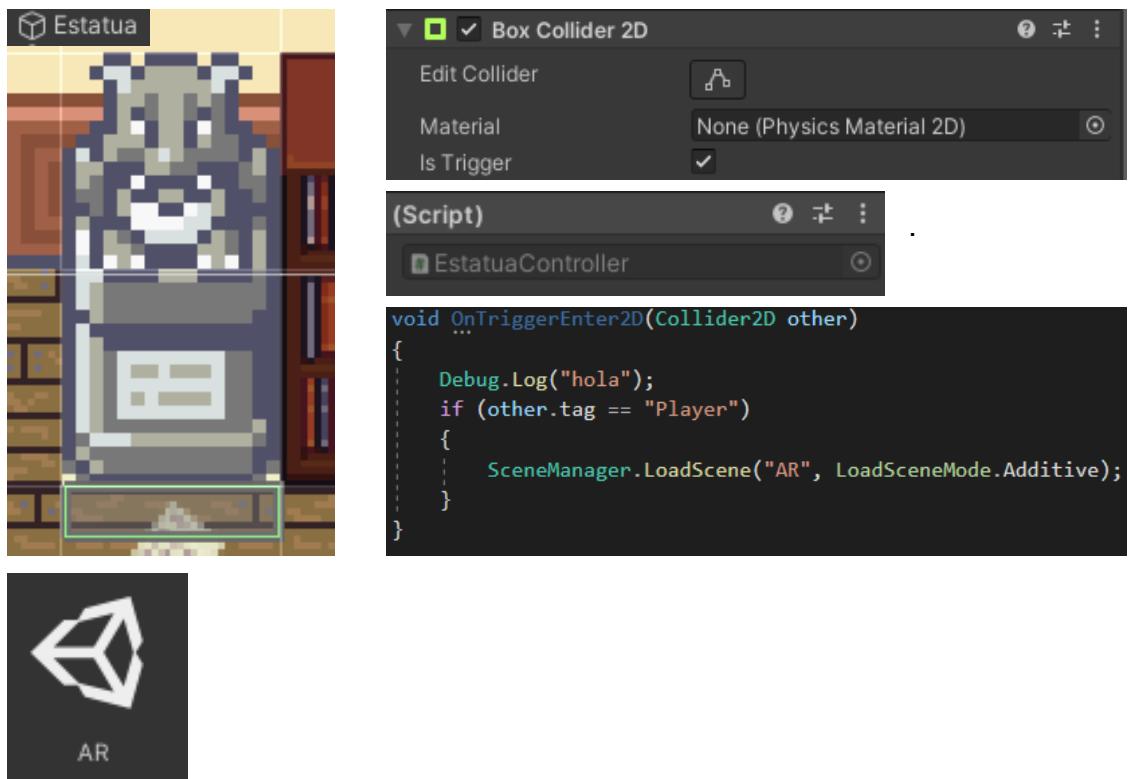
A l'inici de la prova, el jugador observarà un cartell. Per a això, vaig decidir inserir en el "Canva" de l'escena un element de tipus "Image".

Posteriorment, vaig fer un "script" general per a tota la planta, anomenat (PlantaPrincipalController).

Dins del "script", vaig crear una variable de tipus "GameObject" pública, anomenada (textolinicio), per a poder referenciar la imatge. Una vegada fet això, dins de la funció (public void Start()), vaig escriure una nova funció de tipus "Coroutine" (StartCoroutine(Fucion())). Aquesta funció es dirigirà a una altra funció (IEnumerator), que el que farà és activar el text d'inici (textolinicio.SetActive(true)), esperar cinc segons (yield return new WaitForSeconds(5)) i desactivar el text d'inici (textolinicio.SetActive(false)).

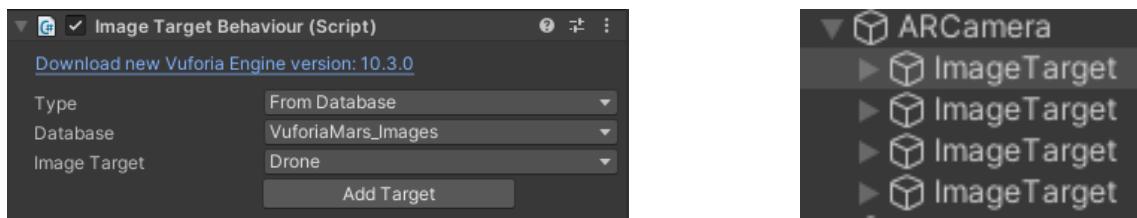


La part principal de la prova és la Realitat Augmentada en l'estàtua. Per a això, vaig posar un "collider" com "trigger" prop de l'estàtua, anomenat "estatua". Una vegada fet això, vaig realitzar un "script" per a aquest "collider", anomenat (EstatuaController) que et dirigeix a una nova escena anomenada "AR", on està situat el sistema de Realitat Augmentada. A diferència d'altres canvis d'escena aquest superposa l'escena damunt de l'actual. D'aquesta manera no és necessari carregar i descarregar l'escena principal. (SceneManager.LoadScene("AR", LoadSceneMode.Additive)).

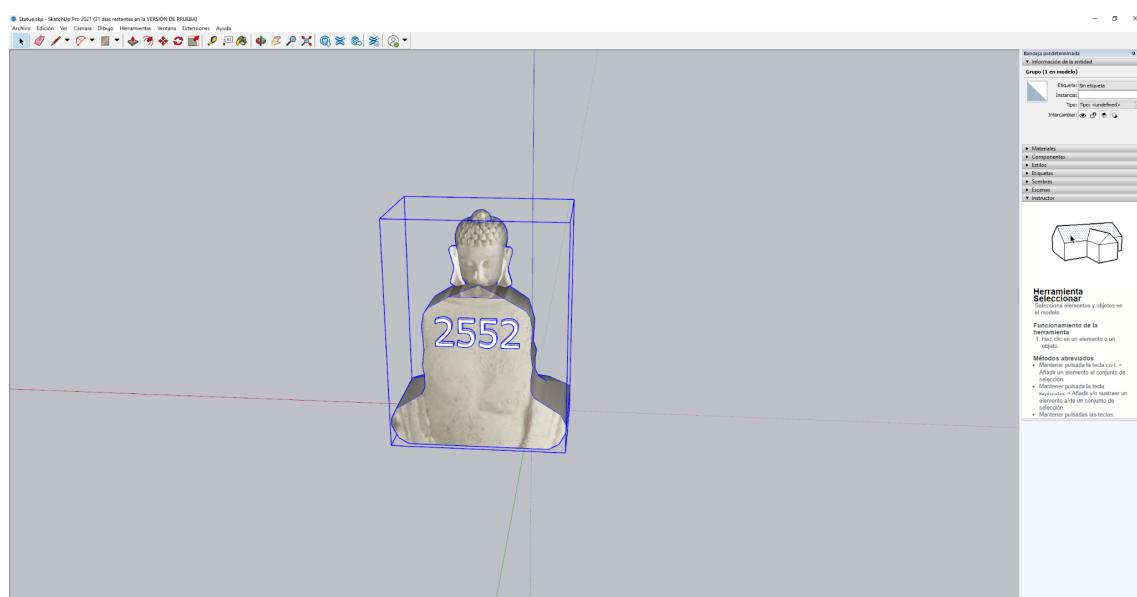


Dins de l'escena "AR" vaig situar tots els elements necessaris per a poder executar Realitat Augmentada dins de Unity. Per a això, vaig necessitar una biblioteca externa anomenada "Vuforia". Aquesta llibreria permet detectar imatges a través de la cambra, per a posteriorment renderitzar un objecte damunt d'aquestes.

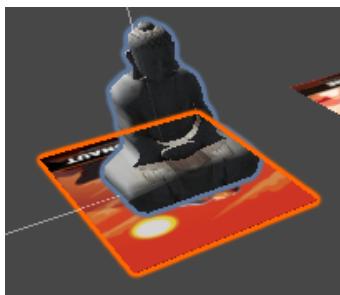
Una vegada introduït per primera vegada "Vuforia" en Unity, es crea una càmera anomenada "ARCamera", la qual representarà la càmera del nostre dispositiu. Dins d'aquesta, es troben els "ImageTarget", que corresponen a les imatges que haurà de reconèixer la "ARCamera". Vaig utilitzar quatre "ImageTargets", un d'ells està destinat a aquesta prova i els altres per a la prova posterior. Les imatges que la "ARCamera" ha de detectar, són de la base de dades proporcionada per "Vuforia". Cada "ImageTarget" ha de tenir una imatge diferent.



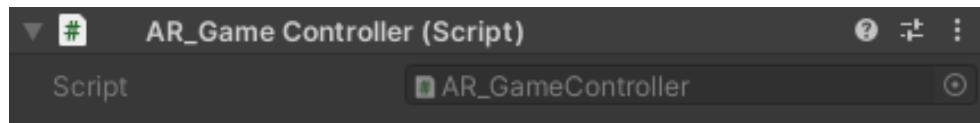
Els objectes renderitzats per a totes dues proves, els he desenvolupat a través d'una aplicació de disseny 3D, anomenada SketchUp. Vaig afegir el codi (2552) a l'estàtua. Posteriorment, els vaig exportar a Unity sense necessitat de cap llibreria.



Una vegada exportats a Unity, vaig escalar els objectes renderitzats, per a situar-los damunt de les imatges. L'estàtua té com a referència la imatge "Astronaut".



Una vegada col·locada l'estàtua damunt de la imatge, vaig realitzar un "script" principal que controlarà tota l'escena "AR", anomenat (ARCameraController).



El codi que conté el "script" és el següent:

```
public class AR_GameController : MonoBehaviour {
    private bool targetInScope;
    0 referencias
    public void TargetLost() {
        targetInScope = false;
        StartCoroutine(Funcion());
    }
    0 referencias
    public void TargetFound() {
        targetInScope = true;
        StopCoroutine(Funcion());
    }
    2 referencias
    IEnumerator Funcion() {
        yield return new WaitForSeconds(3);
        if (targetInScope == false) {
            //SceneManager.LoadScene("PlantaPrincipal", LoadSceneMode.Single);
            var lastSceneIndex = SceneManager.sceneCount;
            SceneManager.UnloadSceneAsync(lastSceneIndex);
        }
    }
}
```

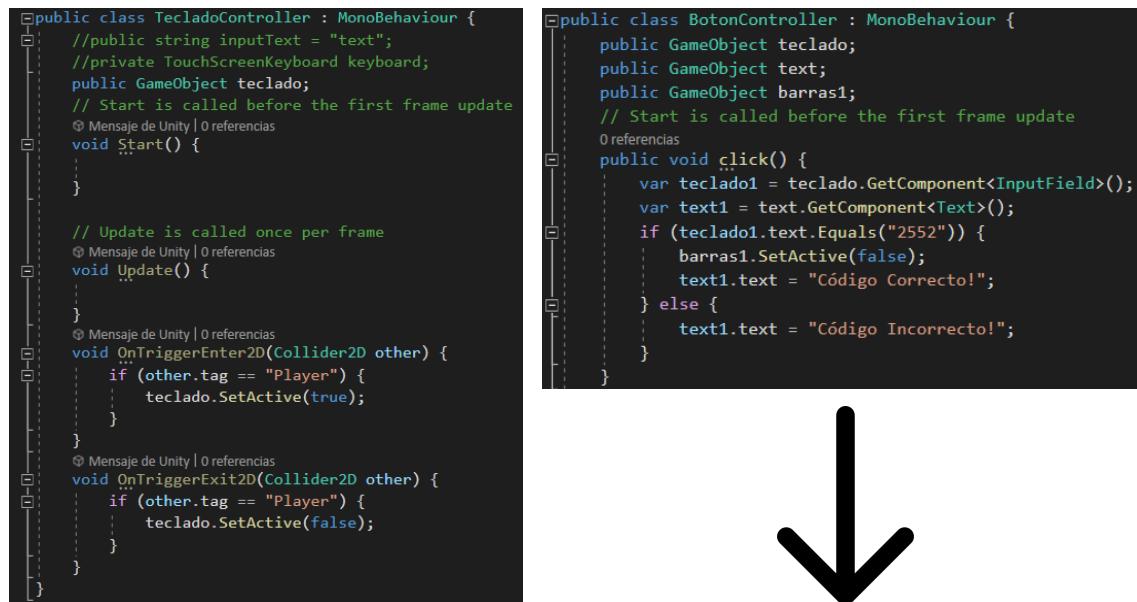
Primer creu una variable de tipus "bool" (targetInScope) pública, que serveix per a verificar si la imatge es mostra en càmera. Posteriorment, vaig efectuar dues funcions per a quan la imatge de referència es mostri en càmera o es perdi. La funció (public void TargetLost()) conté en el seu interior el següent: amb un condicional comprova si la variable (targetInScope) és falsa, en cas que sigui així, comença la funció (IEnumerator), si passen tres segons i la cambra no ha estat capaç de trobar la imatge, torna a carregar l'escena anterior, és a dir, l'escena "PlantaPrincipal". D'altra banda, si torna a recuperar la imatge, o no l'ha perdut, és a dir, que la variable (targetInScope) és veritable, no executi la funció (IEnumerator). La variable (lastScencelIndex) serveix per a fer un recompte de totes les escenes que hi ha en el projecte.

En cas d'haver aconseguit el codi, i voler abandonar l'escena "AR", vaig afegir un botó de "Salir" al "Canvas" de l'escena.



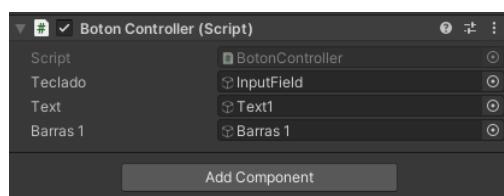
La segona part de la primera prova, consisteix a introduir el codi obtingut en l'estàtua en l'ordinador. Per a això, primer vaig crear un "InputField" en el "Canva" de l'escena, el qual permet escriure un text. A més, també vaig afegir un botó per a confirmar l'escrit. I finalment un text de confirmació. Perquè tot això funcioni, vaig realitzar un "script" (TecladoController) en un "collider" a prop de l'ordinador.

Aquest "script" funciona exactament igual que el de les calaveres, si el "collider" detecta al jugador, s'activa la variable tipus "GameObject" anomenada (teclado). En cas que no ho detecti al jugador, es desactiva. Aquest sistema també compta amb un altre "script" (BotonController) en el botó de confirmació.



Primer de tot, vaig declarar les variables tipus "GameObject", per al teclat, text i barres que no permeten que el jugador surti de la sala. Una vegada fet això, vaig crear una funció pública anomenada (public void clic()) que s'executa quan es prem el botó de confirmació. Dins d'aquesta funció, vaig declarar dues noves variables, per a obtenir el text que introduceix el jugador. Posteriorment, el "script", comprova mitjançant un condicional (if (teclado1.text.Equals("2552"))), si el text que ha introduït el jugador equival al codi que resideix darrere de l'estàtua (2552). Si el text és correcte, desactiva les barres (barras1.SetActive(false)) per a permetre'n accedir a la següent prova i envia un missatge de confirmació, conforme el text introduït és correcte (text1.text = "Codi Correcte!"), en cas de no ser així (else), mostra un missatge que el codi introduït és incorrecte (text1.text = "Codi Incorrecte!").

Per últim, cal referenciar en Unity, que objecte és la variable ("teclado", "text", i "barras1").



Prova 2 - Seqüència d'Objectes "AR"

La segona prova consisteix a realitzar una seqüència d'objectes renderitzats a través de la càmera del dispositiu, amb Realitat Augmentada. Els objectes renderitzats estan creats en SketchUp. Per realitzar la prova, vaig afegir un "collider" en la cadira dreta, prop de l'escriptori. Aquest "collider" posseeix el "script" (SillaController), que executa un canvi d'escena cap a l'escena "AR".

```
void OnTriggerEnter2D(Collider2D other)
{
    Debug.Log("holo");
    if (other.tag == "Player")
    {
        SceneManager.LoadScene("AR", LoadSceneMode.Additive);
    }
}
```

L'escena "AR" funciona exactament igual que la prova anterior, de fet utilitza el mateix "script" (ARGameController), però amb la següent extensió:

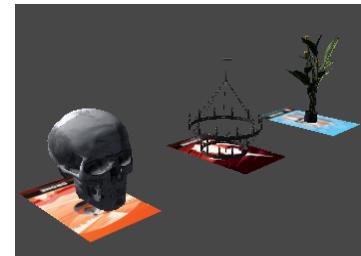
```
public void DetectarCalavera()
{
    if (!PlayerPrefs.HasKey("password")) PlayerPrefs.SetString("password", "");
    var valorActual = PlayerPrefs.GetString("password");
    PlayerPrefs.SetString("password", valorActual + "C");
}

0 referencias
public void DetectarCandelabro()
{
    if (!PlayerPrefs.HasKey("password")) PlayerPrefs.SetString("password", "");
    var valorActual = PlayerPrefs.GetString("password");
    PlayerPrefs.SetString("password", valorActual + "X");
}

0 referencias
public void DetectarPlanta()
{
    if (!PlayerPrefs.HasKey("password")) PlayerPrefs.SetString("password", "");
    var valorActual = PlayerPrefs.GetString("password");
    PlayerPrefs.SetString("password", valorActual + "P");
}
```

He afegit tres funcions més per a cada objecte. La funció pública (DetectarCalavera()) s'executa quan la "ARCamera" troba la imatge referenciada en Unity. Una vegada detectada la imatge, guarda en una variable de tipus "string" (valorActual) un caràcter. (PlayerPrefs.SetString("password", valorActual + "C")). A més compta amb un condicional, que el que fa és establir un valor nul a la variable en iniciar l'aplicació, perquè no es quedí guardat el caràcter sempre. Aquest mateix codi s'aplica per a la funció (DetectarCandelabro()) i (DetectarPlanta()), però simplement el que canvia és el caràcter que es guarda en la variable. Per a la Calavera el caràcter és "C", per al Canelobre el caràcter és "X" i per a la Planta és "P".

Els objectes i les seves referències en les imatges en qüestió són: la Calavera està referenciada amb la imatge "Dron", el Canelobre està referenciat per la imatge "Fisure" i finalment la Planta està referenciada amb la imatge "Oxygen".



Una vegada mostrats els tres objectes renderitzats i guardats els seus caràcters, el valor de la variable (valorActual) es transfereix al "script" (PlantaPrincipalController), on a través de la funció privada (private void Update()) comprova a través d'un condicional si la variable que guardava els caràcters (variableActual) conté els caràcters "CPX", si és així, la condició es compleix i, per tant, desactiva la segona barra (barras2objecto.SetActive(false)) permetent que el jugador pugui accedir la següent planta. Cal esmentar que la segona barra, és una variable pública de "GameObject" que s'ha de referenciar en Unity.

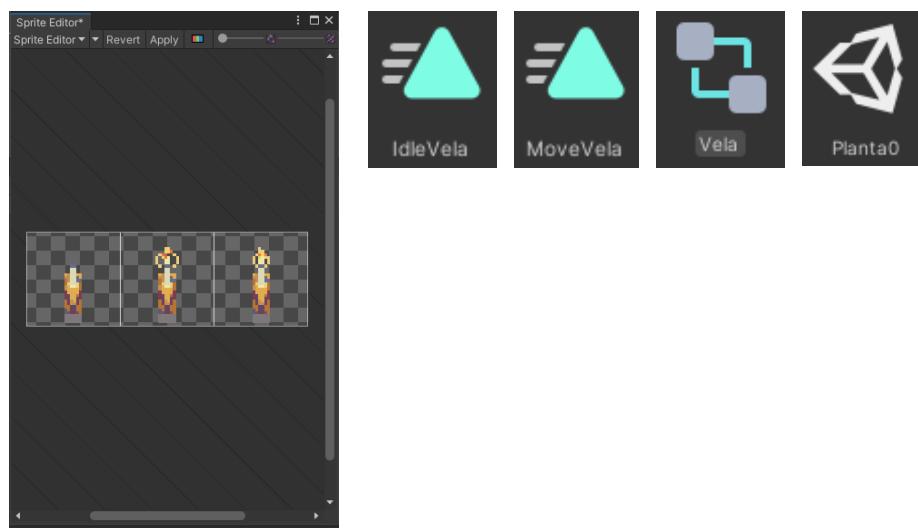
```
private void Update() {
    if (!barras2) {
        if (!PlayerPrefs.HasKey("password")) PlayerPrefs.SetString("password", "");
        var valorActual = PlayerPrefs.GetString("password");
        if (valorActual.Contains("CPX")) {
            barras2 = true;
            barras2objecto.SetActive(false);
        }
    }
}
```

Prova 3 - Veles

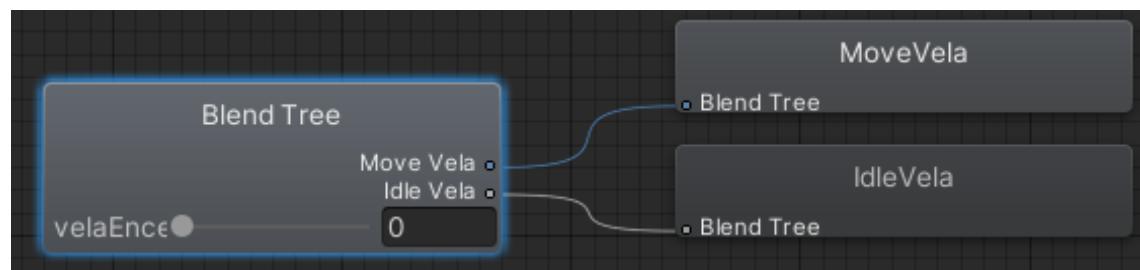
La prova tres consisteix a apagar unes espelmes que es troben damunt d'unes taules, mitjançant una bufada al micròfon del dispositiu. Aquesta prova es troba en l'escena "Planta 0".

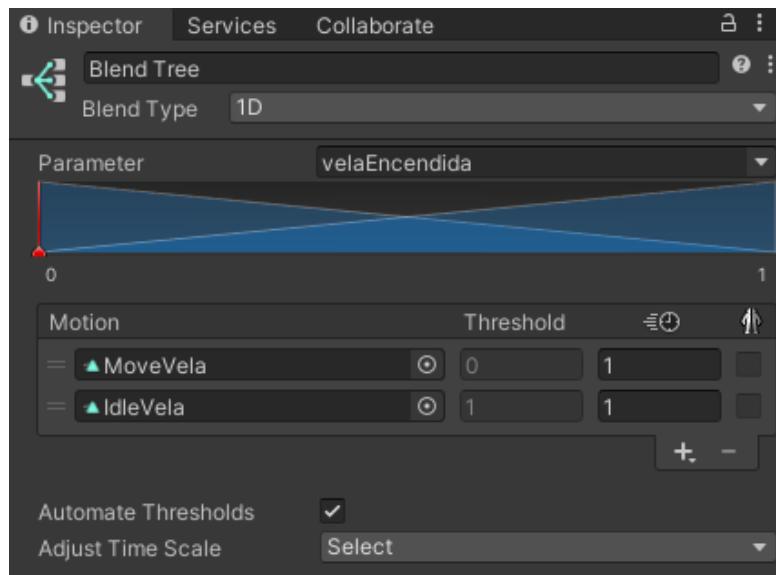
Per a començar, el primer que vaig realitzar va ser el "sprite" per a cada vela. Vaig editar una imatge en Photoshop, amb tres estats per a l'espelma, dos d'aquests són per a l'animació d'encesa i l'altre per a l'animació d'apagat.

Posteriorment, vaig efectuar, igual que el Sprite del jugador, una retallada a la imatge, per a obtenir tres. Una vegada fet això, vaig fer les animacions, creant diferents clips d'animació d'encesa (MoveVela) i apagat (IdleVela). Unity va generar un arxiu anomenat "Vela" on s'agrupen les animacions.



Dins de l'arxiu "Vela", vaig fer un paràmetre anomenat "velaEncendida". En aquest, hi ha un "Blend Tree" 1D, on es troben les animacions d'encesa i apagat. L'animació d'apagat té un valor d'1, i l'animació d'encesa té un valor de 0. El "Blend Tree" habilita el pas entre animacions.





Una vegada realitzada l'animació, vaig realitzar un "script" (GameController) que controla tota l'escena. El codi pertanyent a aquest "script" és:

```

public class GameController : MonoBehaviour {
    public List<GameObject> velas;
    public GameObject player;
    public GameObject barras3;
    int velascontadas = 0;

    void Update() {
        MicLoudness = MicrophoneLevelMax();
        //MicLoudnessinDecibels = MicrophoneLevelMaxDecibels();
        //Debug.Log("MicLoudness: " + MicLoudness);
        //Debug.Log("MicLoudnessinDecibels: " + MicLoudnessinDecibels);

        if (MicLoudness == 1) {
            foreach (GameObject vela in velas) {
                var posicion = player.transform.position - vela.transform.position;
                var animator = vela.GetComponent<Animator>();
                if (posicion.magnitude <= 0.4172759f && animator.GetFloat("velaEncendida") == 0) {
                    animator.SetFloat("velaEncendida", 1);
                    velascontadas += velascontadas + 1;
                }
                if (velascontadas == 5) {
                    Debug.Log("Todas las velas contadas");
                    barras3.SetActive(false);
                }
                //Debug.Log(posicion.magnitude + "/" + MicLoudness + "/" + MicLoudnessinDecibels);
            }
            //Debug.Log("Velastcontadas: " + velascontadas);
        }
    }
}

```

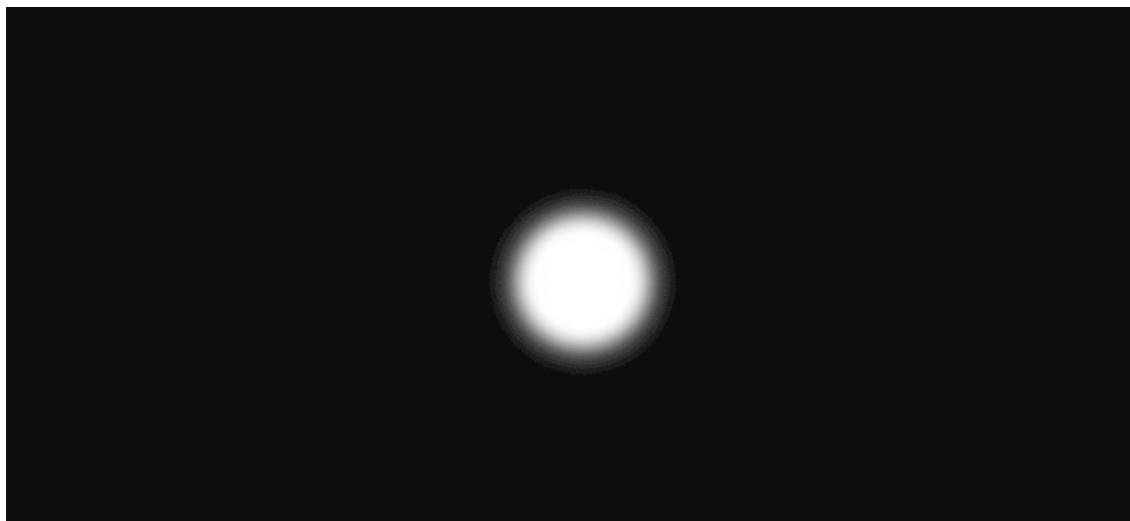
Primer de tot, vaig afegir al codi una llista pública anomenada "veles", perquè agrupés totes les veles pertanyents a la prova (public List veles), posteriorment vaig afegir dues variables públiques de tipus "GameObject" per a referenciar al jugador i a la tercera barrera i finalment una variable de tipus integer (velascontadas) amb un valor de 0.

Seguidament, vaig afegir una biblioteca al codi, perquè Unity em detectés el micròfon del dispositiu (Micro). Dins de la funció (void Update()), revisa constantment el valor rebut pel micròfon. Una vegada obtingut el valor, comprova a través d'un condicional si el valor és igual a 1. En cas de que sigui així, calcula a través d'una nova variable (posicion) la posició que hi ha entre el jugador i la vela, restant-les (var posicion = player.transform.position - vela.transform.position). A més, obté el component d'animació a través d'una nova variable. Conseqüentment, a través d'un altre condicional, comprova si la posició del jugador respecte a l'espelma, és menor o igual a un número establert i si l'animació de l'espelma és 0 (encesa). En cas que es compleixin aquests dos requisits, estableix l'animació de l'espelma a 1 (apagada), i, per tant, aquesta s'apaga, sumant a la variable espelmes comptades +1.

Si la variable veles comptades, suma un total de 5 (if (velascontadas == 5)), la tercera barrera es desactiva (barras3.SetActive(false)), habilitant al jugador la possibilitat de passar a la quarta prova.

Prova 4 - Laberint a les fosques

La quarta prova consisteix a travessar un laberint a les fosques. Per a començar, vaig crear un element de tipus "Image" per a afegir una imatge negra amb un cercle transparent en el centre. La prova es troba en l'escena "Planta 0".



Una vegada afegida la imatge, vaig agregar un "collider" que cobreix tota la sala. A aquest li vaig assignar un nou "script" anomenat (LightRoomController).

El codi és el següent:

```
public class LightController : MonoBehaviour {
    //public bool SensorEnabled = false;
    //public bool HasSensor = false;
    public GameObject imagen;
    public Text DebugText;
    void OnTriggerEnter2D(Collider2D other) {
        if (other.tag == "Player") {
            //SensorEnabled = true;
            imagen.SetActive(true);
        }
    }
    void OnTriggerExit2D(Collider2D other) {
        if (other.tag == "Player") {
            //SensorEnabled = false;
            imagen.SetActive(false);
        }
    }
}
```

De nou, a través d'un condicional, detecta si el jugador col·lideix amb el "collider", i si és així, activa la imatge (imatge.SetActive(true)). En canvi, si el jugador surt del "collider", desactiva la imatge (imatge.SetActive(false)).

Prova 5 - Fletxa

La cinquena prova consisteix a utilitzar el sensor de rotació del dispositiu per a orientar una fletxa situada a la meitat de la sala fins a la cinquena barrera. Es troba en l'escena "Planta 0".

Primer, vaig crear un "collider" amb un "script" anomenat (AccelerometerController) sobre la sala, perquè detecti que el jugador està col·lidint amb ell. El codi d'aquest és el següent: Comença declarant dues variables públiques de tipus "bool" per a comprovar si el sensor està activat i si el dispositiu té el sensor de giroscopi, totes dues assignades en fals. A més, vaig declarar una variable de tipus "GameObject" (Arrow) per a referenciar la fletxa.

```
public bool SensorEnabled = false;
public bool HasSensor = false;
public Text DebugText;
public GameObject Arrow;
```

Una vegada declarades les variables, el codi comença executant una funció (void Start()), verificant si el dispositiu té sensor de giroscopi. En cas de tenir-ho, l'activa.

```
void Start() {
    HasSensor = Accelerometer.current != null;
    if (HasSensor) InputSystem.EnableDevice(Accelerometer.current);
    else Debug.Log("Accelerometer sensor isn't detected");
}
```

Quan el sensor està actiu, recorre a la següent funció (void Update()). Aquesta funció revisa a través d'un condicional si es compleix que el dispositiu tingui sensor de giroscopi i aquest activat. En cas en aquest cas, guarda els valors d'acceleració, en una variable anomenada (acceleration) i posteriorment els hi transmet a l'objecte (Arrow), perquè aquest adquiereixi els valors del giroscopi i, per tant, es mogui.

Perquè el sensor s'activi, és necessari que el jugador aquest dins del "collider" establert al principi. Per a això vaig utilitzar la mateixa funció que en la quarta prova. Si el jugador es troba dins del "collider" el sensor s'activa, en cas d'abandonar-lo, el sensor es desactiva. La fletxa és referència per l'editor de Unity.

```
void Update() {
    if (SensorEnabled && HasSensor) {
        var acceleration = Accelerometer.current.acceleration.ReadValue();
        Arrow.transform.Rotate(0, 0, -acceleration.x);
    }
}

void OnTriggerEnter2D(Collider2D other) {
    if (other.tag == "Player") {
        SensorEnabled = true;
    }
}

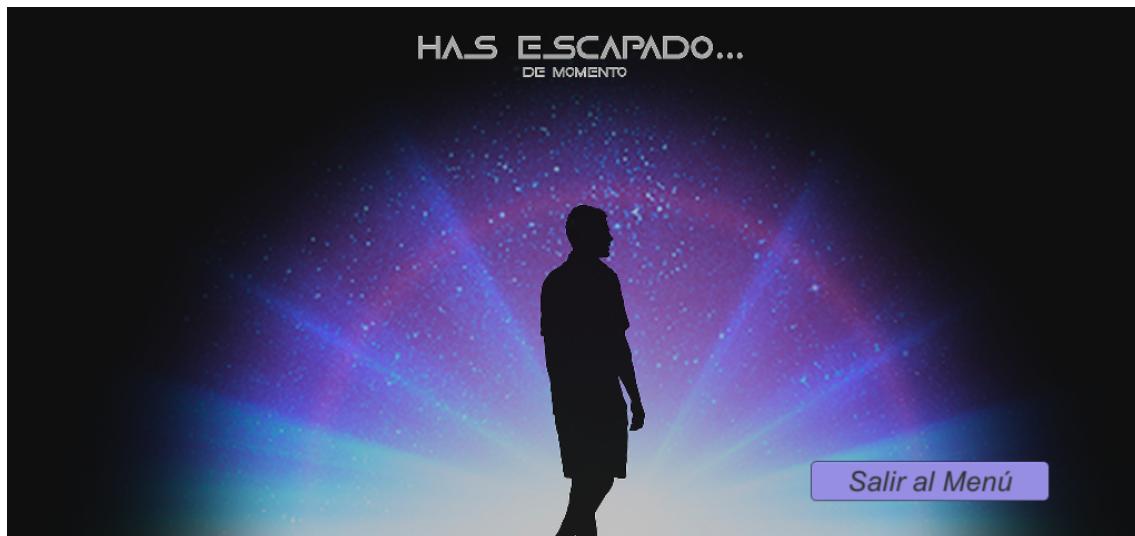
void OnTriggerExit2D(Collider2D other) {
    if (other.tag == "Player") {
        SensorEnabled = false;
    }
}
```

ⓘ Mensaje de Unity | 0 referencias

Final

El final del joc, consisteix a caminar cap a la porta de sortida de la mansió, situada en l'escena "Planta -1". La porta compta amb un "collider", que en establir contacte amb el jugador, a través d'un "script" (PuertaFinal) canvia a una altra escena anomenada "Final".

Aquesta escena inclou un "Canva", amb un element tipus "Image", expressant al jugador que ha aconseguit escapar. A més, compta amb un botó de retorn al menú principal, que, a través d'una funció (onClick()) fa un canvi d'escena a "Menu".



8. Conclusions

Al principi em vaig fixar diversos objectius, els quals eren: descobrir i explicar en què consisteix la Realitat Augmentada, aprendre com ha anat evolucionant a través de la seva història i finalment, desenvolupar un videojoc amb aquesta tecnologia.

Per part del primer objectiu, he aconseguit resoldre aquesta petita curiositat sobre com és capaç de combinar-se "maquinari" i "programari", trobant així una tecnologia relativament recent que permet aquesta combinació, la Realitat Augmentada.

Durant la realització del segon objectiu m'he adonat del recent que és aquesta tecnologia fins i tot, i de tot el que ha aconseguit evolucionar en tan poc temps. Amb el pas dels anys, podrem veure com la societat avançarà enormement gràcies al que podrà oferir la Realitat Augmentada, ja que existeixen diverses idees o prototips que seran molt útils i revolucionàries.

I per part del tercer objectiu, desenvolupar un videojoc amb aquesta tecnologia, ha estat el que més he gaudit. Primer per la satisfacció i curiositat que m'ha generat sobre com és un videojoc internament i com es va formant abans d'arribar a l'usuari final. La part de disseny de nivells em va semblar una de les més laborioses, pel fet de pensar el concepte, l'estètica i tonalitat que li volia donar al projecte. Quant a la part de programació hi ha segons que parts del llenguatge de programació que emprí per a utilitzar el codi que em van resultar una mica complexes com per exemple aplicar la Realitat Augmentada, accedir als sensors del dispositiu que s'està utilitzant, etc. A més, segons que funcions de Unity no em quedaven del tot clar, però vaig aconseguir sortir d'aquests problemes gràcies a la gran quantitat de documentació i ajuda que proporcionen a gent inexperta.

Com a conclusió general d'aquest treball, podria afirmar que he gaudit realitzant-lo i he après moltes coses que em podran servir per a la professió que vull realitzar en un futur.

9. Bibliografia

AdriGM. “*Tiled Map Editor, el editor de mapas libre*. Genbeta.com [Internet]

Cercat 22/07/2021, des de:

<https://www.genbeta.com/desarrollo/tiled-map-editor-el-editor-de-mapas-libre>.

Anònim. “*Augment 3D, la app de realidad aumentada que se escapa del papel*”.

Realidad en Aumento [Internet]. Cercat 14/11/2021, des de:

<https://realidadenaumento.es/augment-3d-app-realidad-aumentada/>.

Anònim. “*C (lenguaje de programación) - Wikipedia, la enciclopedia libre*”.

Es.wikipedia.org [Internet]. Cercat 25/10/2021, des de:

[https://es.wikipedia.org/wiki/C_\(lenguaje_de_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n)).

Anònim. “*C++ - Wikipedia, la enciclopedia libre*”. Es.wikipedia.org [Internet].

Cercat 3/08/2021, des de: <https://es.wikipedia.org/wiki/C%2B%2B>.

Anònim. “*C Sharp - Wikipedia, la enciclopedia libre*”. Es.wikipedia.org [Internet].

Cercat 9/11/2021, des de: https://es.wikipedia.org/wiki/C_Sharp.

Anònim. “*Getting Started with Vuforia Engine and Lumin | VuforiaLibrary*”.

Library.vuforia.com [Internet]. Cercat 29/10/ 2021, des de:

<https://library.vuforia.com/getting-started/getting-started-vuforia-engine-and-lumin>.

Anònim. “*How Do I Import 3D objects (in Collada dae) files into unity - Unity Answers*”. Answers.unity.com [Internet]. Retrieved 25/08/2021, des de:

<https://answers.unity.com/questions/43174/how-do-i-import-3d-objects-in-collada-dae-files-in.html>.

Anònim. “*How to load a scene without closing the current scene - Unity Answers*”. Answers.unity.com [Internet]. Cercat 12/09/2021, des de:

<https://answers.unity.com/questions/1544252/how-to-load-a-scene-without-closing-the-current-sc.html>.

Anònim. “*Infraestructura de lenguaje común - Wikipedia, la enciclopedia libre*”. Es.wikipedia.org [Internet]. Cercat 30/07/2021, des de:

https://es.wikipedia.org/wiki/Infraestructura_de_lenguaje_com%C3%BAn.

Anònim. “*Make pictures into a material - Unity Answers*”. Answers.unity.com

[Internet]. Cercat 29/09/2021, des de:

<https://answers.unity.com/questions/126746/make-pictures-into-a-material.html>.

Anònim. “*Microsoft .NET - Wikipedia, la enciclopedia libre*”. Es.wikipedia.org

[Internet]. Cercat 21/10/2021, des de:

https://es.wikipedia.org/wiki/Microsoft_.NET.

Anònim. “*Microsoft Visual Studio - Wikipedia, la enciclopedia libre*”.

Es.wikipedia.org [Internet]. Cercat 5/08/2021, des de:

https://es.wikipedia.org/wiki/Microsoft_Visual_Studio.

Anònim. “*Lenguaje de alto nivel - Wikipedia, la enciclopedia libre*”.

Es.wikipedia.org [Internet]. Cercat 30/10/2021, des de:

https://es.wikipedia.org/wiki/Lenguaje_de_alto_nivel.

Anònim. “*Lenguaje de bajo nivel - Wikipedia, la enciclopedia libre*”.

Es.wikipedia.org [Internet]. Cercat 30/10/2021, des de:

https://es.wikipedia.org/wiki/Lenguaje_de_bajo_nivel.

Anònim. “*OpenGameArt.org*”. OpenGameArt.org [Internet]. Cercat 25/08/2021,

des de: <https://opengameart.org/>.

Anònim. “*Realidad aumentada en logística. Usos y ventajas - IAT*”. IAT

[Internet]. Cercat 20/10/2021, des de:

<https://iat.es/tecnologias/realidad-aumentada/logistica/>.

Anònim. “*Realitat augmentada*” - Viquipèdia, l'enclopèdia lliure.

Ca.wikipedia.org [Internet]. Cercat 13/06/2021, des de:

https://ca.wikipedia.org/wiki/Realitat_augmentada.

Anònim. “*Realidad aumentada - Wikipedia, la enciclopedia libre*”. Es.wikipedia.org [Internet]. Cercat 25/06/2021, des de:

https://es.wikipedia.org/wiki/Realidad_aumentada.

Augmented Startups. How to Build Augmented Reality Apps with Vuforia 8 in Unity 2020 & Deploy to Android [Video]. Cercat 30/11/2021, des de:

<https://www.youtube.com/watch?v=y7VD7yGwmV4>.

Collado, C. “*Las mejores apps de realidad aumentada: prueba ropa, maquillaje e incluso muebles antes de comprarlos*”. Andro4all [Internet]. Cercat 12/11/2021, des de:

<https://andro4all.com/listas/apps-android/app-probar-ropa-maquillaje-muebles-realidad-aumentada>.

Drechsler, N. “*MINI Augmented Vision: A revolutionary display concept offering enhanced comfort and safety. Exclusive prototype of augmented reality eyewear underlines the innovative flair and creativity of the MINI brand...*”.

BMW Group PressClub [Internet]. Cercat 27/07/2021, des de:

<https://www.press.bmwgroup.com/global/article/detail/T0212042EN/mini-augmented-vision:-a-revolutionary-display-concept-offering-enhanced-comfort-and-safety-exclusive-prototype-of-augmented-reality-eyewear-underlines-the-innovative-flair-and->

Editor, T. “*Tiled Map Editor by Thorbjørn*”. itch.io [Internet]. Cercat 6/07/2021, des de: <https://thorbjorn.itch.io/tiled>.

GameDevTraum. “*¿Cómo funciona OnTriggerEnter en Unity? Ejemplos de aplicación*” [Internet]. Cercat 13/11/2021, des de:

<https://gamedevtraum.com/es/desarrollo-de-videojuegos/tutoriales-y-soluciones-unity/serie-fundamental-unity/como-funciona-ontriggerenter-en-unity-aplicacion-es/>.

Game Dev Experiments. Make A Game Like Pokemon in Unity | #1 - Introduction and Tile Based Movement [Video]. Cercat 13/09/2021, des de: https://www.youtube.com/watch?v=_Pm16a18zy8.

Game Dev Experiments. Make A Game Like Pokemon in Unity | #2 - Player Animations [Video]. Cercat 18/09/2021, des de:

<https://www.youtube.com/watch?v=3h6kZkhza3U>.

Grapsas, "T. *Realidad aumentada: ¿qué es, cómo funciona y para qué sirve?*".

Rock Content - ES [Internet]. Cercat 20/10/2021, des de:

<https://rockcontent.com/es/blog/realidad-aumentada/>.

Peterson, J. "Virtual Reality, Augmented Reality, and Mixed Reality Definitions".

Entmerch.org [Internet]. Cercat 12/08/2021, des de:

<https://www.entmerch.org/digitalema/white-papers/2017-ema-vr-ar-mr-definitio.pdf>.

Innovae, E. "Todo sobre la Realidad Aumentada | Innovae". Innovae [Internet].

Cercat 1/10/2021: <https://www.innovae.eu/la-realidad-aumentada/>.

Polo, R. "Elements 4D, o el mundo de la química en Realidad Aumentada".

WWWhat's new [Internet]. Cercat 2/11/2021, des de:

<https://wwwwhatsnew.com/2014/02/03/elements-4d-o-el-mundo-de-la-quimica-en-realidad-aumentada/>.

Rayos El Dev. Tutorial poner un JOYSTICK en UNITY para móvil - Moviendo un personaje en 3r Persona ROTAR y MOVER [Video]. Cercat 14/09/2021, des de:

<https://www.youtube.com/watch?v=9uEv392wCzw>.

samyam. How to EASILY Build to an Android Phone in Unity [Video]. Cercat

08/10/2021, des de: https://www.youtube.com/watch?v=Nb62z3J4A_A.

Sherry. How To Export SketchUp Models To Unity - Tutorial HD [Video]. Cercat 29/11/2021, des de: https://www.youtube.com/watch?v=DyJ8S_859n0.

SketchUp. 3D Text - Square One [Video]. Cercat 29/11/2021, des de:

<https://www.youtube.com/watch?v=5evLszVYT1g>.

Studios, F. “*Joystick Pack*”. Unity - Assets Store [Internet]. Cercat 7/08/2021, des de:

<https://assetstore.unity.com/packages/tools/input-management/joystick-pack-107631>.

Technologies, U. *Unity - Scripting API*:

“*SceneManagement.SceneManager.LoadSceneAsync*”. Docs.unity3d.com [Internet]. Cercat 29/07/2021, des de:

<https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.LoadSceneAsync.html>.

Univision A Bordo. MINI Augmented Vision Powered by MINI Connected [Video]. Cercat 19/11/2021, des de:

<https://www.youtube.com/watch?v=Dyj0t-GUJhs>.

Xamarin. “*BBVA Valora View, la app de Realidad Aumentada*”. Plain Concepts [Internet]. Cercat 24/09/2021, des de:

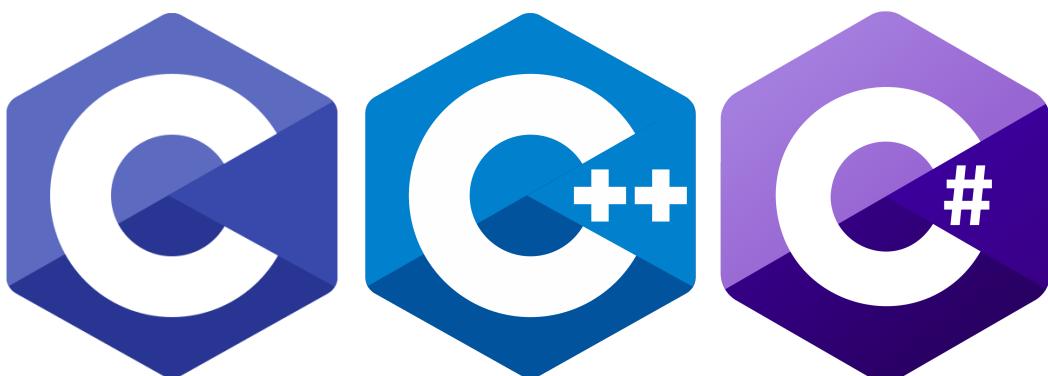
<https://www.plainconcepts.com/es/casestudy/bbva-valora-realidad-aumentada/>.

Annexos

Llenguatge de Programació: C#

C# és un llenguatge de programació desenvolupat i estandarditzat per Microsoft com a part de la seva plataforma .NET

C# és un llenguatge de programació dissenyat per a la infraestructura de llenguatge comú (la principal característica d'aquesta, és poder dur a terme aplicació amb llenguatges d'alt nivell, i poder executar-los en diferents plataformes de *hardware* o *software* sense la necessitat de reescriure o recompilar la base del codi, és a dir, el seu codi base). C# té l'especial característica que la seva sintaxi bàsica deriva de dos llenguatges de programació (C/C++) però amb algunes modificacions (millores).



C / C++

Per a entendre com funciona C#, és necessari comprendre quina finalitat i ús tenien C i C++:

- C: És un llenguatge de programació que té com a finalitat implementar-se en sistemes operatius. El seu codi és versàtil, ja que permet programar estructures d'alt nivell (expressió d'algorismes d'una manera adequada a la capacitat cognitiva humana). Permet entendre el llenguatge màquina (codi binari) i una expressió gairebé oral entre l'estructura del programa i la seva posterior compilació i al seu torn permetre un control de baix nivell (control directe sobre el *hardware*, condicionat pel mateix).
- C++: És una extensió de C, pel fet que afegeix la manipulació d'objectes. Es diu que C++ és un llenguatge multiparadigma, perquè compta amb programació genèrica, estructurada i orientada a objectes.

C#: Característiques

Una vegada clara la base de la qual parteix C#, podem afirmar que C# compta amb les següents característiques:

- Compatibilitat: C# és un llenguatge de programació compatible amb uns altres, com a C, C++, ja que deriva d'ells.
- Actualitzat: C# s'actualitza o renova constantment, per a incorporar noves funcionalitats o millores.
- Senzill / Modern: C# compta amb una gran senzillesa respecte als seus antecessors, ja que elimina elements que no eren necessaris o que simplement han caigut en desús gràcies a la seva constant renovació.
- Assegurança: C# incorpora mecanismes que asseguren que l'accés a dades es portin correctament, evitant així errors de difícil detecció.