**SILENTIUM**

**-MEMÒRIA-**

Héctor Lara García,

Marc Sánchez López,

Eric Morales Martínez

**ABSTRACT**

Silentium is a Survival Horror where the player will have to explore an isolated map to solve puzzles without making any type of noise because three enemies are listening, and if they hear, they will try to kill. All this with a limited inventory to test the player’s ability to manage resources.

Inside this document, you will find all the information regarding the development of our game, such as the motivation behind it, the technologies we used, how the development was, and which difficulties there were and how we overcame them. Regarding the organization, you will also find how we organized and distributed the tasks of the project at first versus at the end. Finally, we will make an overall conclusion about the whole project and what we would change or do in another way if we had to repeat it.

ÍNDEX

[1. INTRODUCCIO 1](#_Toc199001781)

[1.1. IDEA, MOTIVACIONS 1](#_Toc199001782)

[1.2. PITCH 1](#_Toc199001783)

[2. JUSTIFICACIÓ 2](#_Toc199001784)

[2.1 PROPOSTA DE VALOR 2](#_Toc199001785)

[2.2. MODEL DE NEGOCI 2](#_Toc199001786)

[2.3. DAFO 2](#_Toc199001787)

[3. TECNOLOGIES I ESTAT DE L’ART 4](#_Toc199001788)

[3.1. TECHNOLOGY STACK 4](#_Toc199001789)

[3.2. MOTOR I LLIBRERIES 4](#_Toc199001790)

[3.2.1. UNITY 4](#_Toc199001791)

[3.2.2. VFX GRAPH 4](#_Toc199001792)

[3.2.3. SHADER GRAPH 4](#_Toc199001793)

[3.2.4. BARRACUDA 4](#_Toc199001794)

[3.2.5. PROBUILDER 4](#_Toc199001795)

[3.3. IDES I LLENGUATGES DE PROGRAMACIÓ 4](#_Toc199001796)

[3.3.1. VISUAL STUDIO CODE 4](#_Toc199001797)

[3.3.2. C# 4](#_Toc199001798)

[3.3.3. JSON 4](#_Toc199001799)

[3.4. CÒPIES DE SEGURETAT 4](#_Toc199001800)

[3.4.1. GITHUB 4](#_Toc199001801)

[3.4.2. GOOGLE DRIVE 4](#_Toc199001802)

[3.5. ART 4](#_Toc199001803)

[3.5.1. BLENDER 5](#_Toc199001804)

[3.5.2. CHATGPT 5](#_Toc199001805)

[3.5.3. PAINT 5](#_Toc199001806)

[3.5.4. UNITY ASSET STORE 5](#_Toc199001807)

[3.5.5. MIXAMO 5](#_Toc199001808)

[4. DESENVOLUPAMENT DEL PROJECTE 6](#_Toc199001809)

[4.1. BACKLOG 6](#_Toc199001810)

[4.2. SPRINTS 7](#_Toc199001811)

[4.2.1. SPRINT 1 7](#_Toc199001812)

[4.2.**2.** SPRINT 2 9](#_Toc199001813)

[4.2.3. SPRINT 3 9](#_Toc199001814)

[4.2.4. SPRINT 4 9](#_Toc199001815)

[4.2.5 PROGRÉS PER SPRINT 9](#_Toc199001816)

[4.2.6 TASQUES INACABADES 9](#_Toc199001817)

[5. IMPLEMENTACIÓ 10](#_Toc199001818)

[5.1. CODI 10](#_Toc199001819)

[5.1.1. SISTEMA DE SO 10](#_Toc199001820)

[5.1.2. ENEMICS 12](#_Toc199001821)

[5.1.2. INTERACCIÓ DEL PLAYER AMB EL MÓN 12](#_Toc199001822)

[5.1.3. PUZZLE AMB IA 12](#_Toc199001823)

[5.2. DADES 12](#_Toc199001824)

[5.2.1. OBJECTES 12](#_Toc199001825)

[5.2.2. INVENTARI 12](#_Toc199001826)

[5.2.3. DESAT 12](#_Toc199001827)

[5.3. INTERFÍCIES 12](#_Toc199001828)

[5.3.1. SHADERS 13](#_Toc199001829)

[5.3.2. EFECTES DE POST-PROCESSAT 13](#_Toc199001830)

[5.3.3. PARTÍCULES 13](#_Toc199001831)

[5.3.4. WAVE EFFECT 13](#_Toc199001832)

[5.3.4. MODEL VISTA-CONTROLADOR 13](#_Toc199001833)

[6. TESTING I RESULTATS 14](#_Toc199001834)

[6.1. TESTING 14](#_Toc199001835)

[6.2. RESULTATS DE L’APLICACIÓ 14](#_Toc199001836)

[6.3. OBJECTIUS 14](#_Toc199001837)

[6.3.1. OBJECTIUS ASSOLITS 14](#_Toc199001838)

[6.3.2. OBECTIUS NO ASSOLITS 14](#_Toc199001839)

[7. CONCLUSIONS 15](#_Toc199001840)

[7.1. DIFICULTATS 15](#_Toc199001841)

[7.2. MILLORES PROPOSADES 15](#_Toc199001842)

[7.3. VALORACIÓ FINAL 15](#_Toc199001843)

[8. BIBLIOGRAFIA 16](#_Toc199001844)

[8.1. WEBS I VÍDEOS CONSULTATS 16](#_Toc199001845)

[8.2. CRÈDITS 16](#_Toc199001846)

[8.2.1. MODELS I TEXTURES 16](#_Toc199001847)

[8.2.2. MÚSICA I SFX 16](#_Toc199001848)

[9. ANNEXOS 17](#_Toc199001849)

# 1. INTRODUCCIO

## 1.1. IDEA, MOTIVACIONS

Nosaltres teníem clar que volíem fer un joc de por, ja que a un integrant del nostre grup li feia molta il·lusió. Teníem una mecànica clara des de pràcticament l’inici: el so, volíem que els enemics fossin sensibles al so que fes el jugador i que reaccionessin a aquest. Vam estar pensant i no trobàvem res que fos adequat per aquest projecte més enllà del so, ja que la majoria de les nostres idees no eren molt interessants mecànicament. Per aquest motiu vam decidir que mantindríem el gènere de terror i la mecànica del so, però canviant de gènere al Survival Horror perquè era molt més interessant.

Una vegada ja teníem una idea clara de quin gènere volíem vam començar a pensar la història, els enemics, les diferents mecàniques...

## 1.2. PITCH

És un joc de por enfocat a fer puzles mentre escapes de 3 monstres amb característiques diferents. El teu objectiu és resoldre els puzles mentre t’amagues, dispares i fuges. La part important del joc és el sistema de so que hi ha incorporat a cada objecte i enemics. Pots distreure als enemics llençant objectes a on vulguis que hi vagin, però has de vigilar perquè cada acció del jugador pot fer un soroll més gran que una altra. Si fas massa soroll t’has de donar per mort perquè vindran tots a l’hora. Es recompensa el saber quan utilitzar els objectes (cures, bales, etc.) i administrar l’inventari, ja que aquest és molt limitat.

# 2. JUSTIFICACIÓ

## 2.1 PROPOSTA DE VALOR

El que ens separa dels diferents jocs de por és que oferim una perspectiva del Survival Horror que no és molt habitual: en comptes que el joc sigui en 3a persona, el nostre és en 1a persona per oferir una experiència més immersiva. També fem servir un sistema de so per tal que el jugador sigui molt previngut amb el que fa. Si mirem tots els jocs Survival Horror veurem que molt pocs o pràcticament cap implementen aquestes característiques i creiem que això és el que pot fer el nostre producte més atractiu que la resta.

Una altra cosa que ens diferencia del gènere és que tenim tres enemics, els quals no pots matar (només pots noquejar-los) i que tenen un comportament diferent l’un de l’altre. Creiem que aquesta característica fa que el nostre joc tingui més game juice, ja que el jugador ha d’actuar d’una manera diferent depenent del monstre, la qual cosa pot atraure jugadors cansats dels mateixos enemics que sol haver-hi als Survival Horror.

## 2.2. MODEL DE NEGOCI

Silentiumserà un joc de pagament on es tindrà accés a tot el contingut del joc amb aquesta única compra de 4,99 € amb futures rebaixes podria baixar fins a 1,99 €. No tenim plans de treure cap DLC ni res per l’estil, ja que creiem que per la temàtica i la història del nostre joc no tindria molt de sentit.

Per a vendre el nostre joc hem decidit fer servir dues plataformes: Steam i GOG. Steam perquè és on hi ha més jugadors a PC actualment i GOG perquè ens agrada molt que no tingui DRM, cosa amb la qual estem molt en contra.

## 2.3. DAFO

Com a debilitats, el fet que sigui un joc Survival Horror ja pot fer que bastants jugadors estiguin reticents a comprar-ho de primeres. Un altre problema de pertànyer a aquest gènere és que és molt nínxol i el nostre públic objectiu no és tan elevat com sí que ho seria un FPS o similars. A més, com que el nostre joc no té una producció AAA o AA, sinó que és més ben petita, pot fer que perdem alguns compradors potencials.

Per aquest últim motiu ens podem sentir amenaçats, ja que és possible que ens comparin amb altres jocs de la indústria i veure que no val la pena comprar el nostre joc si poden comprar un AAA, AA o simplement un joc amb millors gràfics. Aquest problema s’accentua més per una de les nostres plataformes de venda: Steam. A Steam surten molts cada dia i que el nostre humil joc destaqui entre tots aquests pot ser molt complicat per la quantitat d’opcions que hi ha al mercat. A més, el Survival Horror s’associïa a un estil molt específic en quant a gameplay (càmera fixa, 3a persona...) i això pot fer que un cert públic fan d’aquest gènere no acabi d’associar-nos amb aquest. A més, al mercat ja hi ha jocs del mateix gènere amb un aspecte visual més atractiu que el nostre com per exemple Signalis.

La nostra fortalesa és que oferim una perspectiva única del Survival Horrorbarrejant-lo amb terror clàssic. Això ens ofereix una gran oportunitat, ja que podem atraure a jugadors que no els agrada el gènere, per tant, la quantitat de jugadors objectiuscreix. Distribuir-ho a GOG també ens augmenta aquest públic objectiu perquè no és una plataforma on hi hagi tants jocs com a Steam i és més fàcil destacar.

Com a oportunitats, els jocs de terror i puzles són molt populars i el fet que el preu del joc sigui tan baix ens afavoreix. A més, com hem mencionat anteriorment, la nostra única perspectiva del Surival Horrorpot atraure un públic més ampli.

# 3. TECNOLOGIES I ESTAT DE L’ART

## 3.1. TECHNOLOGY STACK

## 3.2. MOTOR I LLIBRERIES

### 3.2.1. UNITY

### 3.2.2. VFX GRAPH

### 3.2.3. SHADER GRAPH

### 3.2.4. BARRACUDA

### 3.2.5. PROBUILDER

## 3.3. IDES I LLENGUATGES DE PROGRAMACIÓ

### 3.3.1. VISUAL STUDIO CODE

### 3.3.2. RIDER

### 3.3.3. C#

### 3.3.4. JSON

## 3.4. CÒPIES DE SEGURETAT

### 3.4.1. GITHUB

### 3.4.2. GOOGLE DRIVE

## 3.5. ART

### 3.5.1. BLENDER

### 3.5.2. CHATGPT

### 3.5.3. PAINT

### 3.5.4. UNITY ASSET STORE

### 3.5.5. MIXAMO

## 3.6. ALTRES

### 3.6.1. TEACHABLE MACHINE

# 4. DESENVOLUPAMENT DEL PROJECTE

## 4.1. BACKLOG

Nosaltres vam voler repartir les tasques principals entre els Sprint 1 i 2 per tal que tota la jugabilitat essencial estigués acabada per al Sprint 3. Se li va donar prioritat sobretot als puzles i als enemics, ja que són el cor del nostre joc i se’ls hi havia de prendre atenció.



Al backlog final es pot apreciar que el volum de tasques ha crescut exponencialment, ja que a mesura que s’ha anat treballant en el projecte han sorgit noves tasques que no haviem tingut en compte o que simplement són conseqüència d’una altra.

En quant a les tasques, es van decidir abans de començar el projecte i hem seguit aquesta línia amb certs canvis segons les situacions de cadascun. Creiem que la distribució de les tasques podria haver estat millor plantejada, ja que en el cas del enemics, només una persona era l’únic encarregat d’aquests i potser si els haguéssim repartir hauríem anat més ràpid.

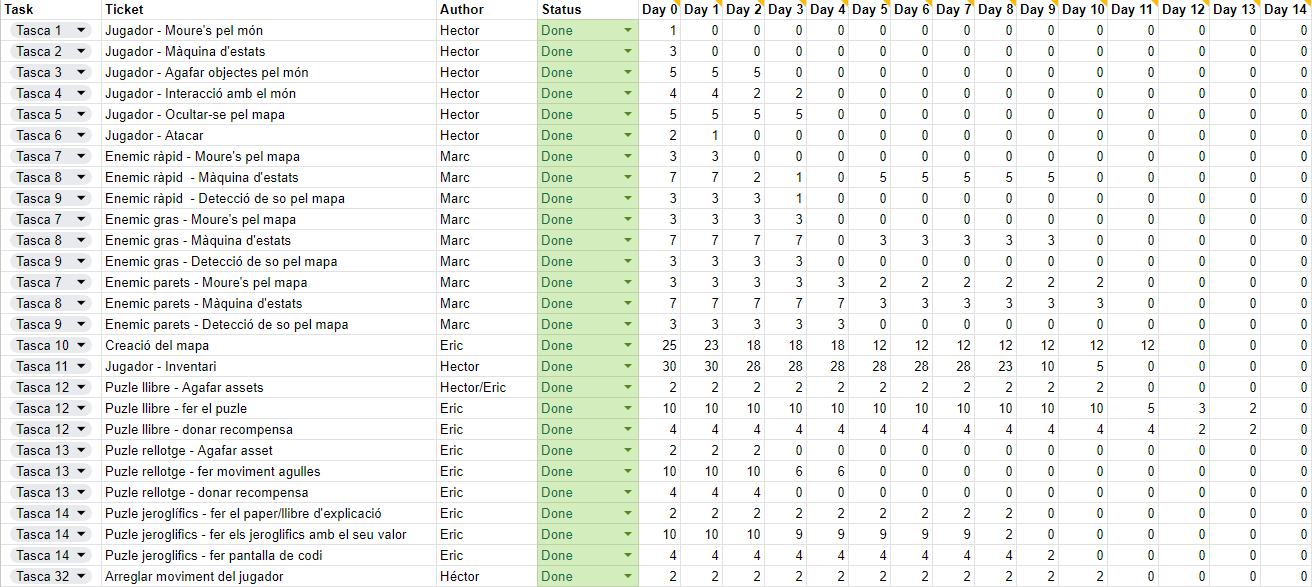




## 4.2. SPRINTS

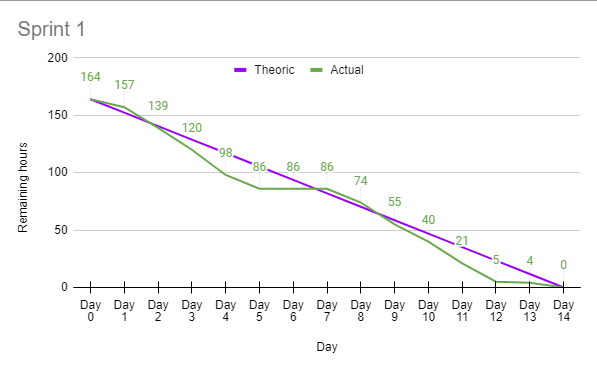
## 4.2.1. SPRINT 1

Durant el primer Sprint el nostre objectiu era fer els monstres (la seva màquina d’estats), el jugador (disparar, moure’s, interactuar amb els diferents elements del mapa), començar alguns puzles i finalment, tot el sistema d’inventari i d’objectes.



En aquest Sprint no es van definir bé les tasques: moltes tasques no especificaven molt bé el que s’havia de fer, sinó que eren generalitzades i no representaven del tot bé el que s’havia de fer en aquella tasca.

Pel que fa al càlcul del temps, creiem que el temps que es va destinar als enemics va ser massa poc i s’hauria d’haver destinat més temps del plantejat en un inici, ja que la màquina d’estats de l’enemic ràpid era més complexa del que ens pensàvem.



### 4.2.**2.** SPRINT 2

### 4.2.3. SPRINT 3

### 4.2.4. SPRINT 4

### 4.2.5 PROGRÉS PER SPRINT

### 4.2.6 TASQUES INACABADES

# 5. IMPLEMENTACIÓ

## 5.1. CODI

### 5.1.1. SISTEMA DE SO

El so és la mecànica principal del nostre joc. Cada vegada que el jugador fa alguna acció (disparar, córrer...) produeix un cert nivell de so que els enemics poden sentir i fer que es dirigeixin cap aquella posició.

#### 5.1.1. COM ES PRODUEIX EL SO

Tots els enemics tenen una funció anomenada ListenSound que rep com a paràmetre un Vector3 (la posició de l’origen del so) i el **nivell de so**, és a dir, la quantitat de so. Aquesta funció es cridada pel Player cada vegada que fa una acció que produeix so. Per a fer això el jugador té una funció anomenada MakeNoise que rep com a paràmetre un int (radi) i el nivell de so que produeix aquella acció. Aquesta funció fa un OverlapShere de només la layer dels enemics i per cada enemic trobat dins del radi es crida a la seva funció de ListenSound. Per a poder aconseguir que cada vegada que el Player es mogui, camini, corri o s’ajupi faci so contínuament, es fa una corutina que depèn de l’estat en què estigui aquest fa un nivell de so o un altre. Pel que fa als objectes llençables, quan toquen qualsevol superfície fan el mateix que el Player, però sense corutina. El mateix amb l’acció de disparar.

#### 5.1.2. ATENUACIÓ DEL SO

Per tal de fer el sistema de so més realista i interessant, es va crear una interfície anomenada IAttenuable.

Diagrama

El contenido generado por IA puede ser incorrecto.

Aquesta funció té l’objectiu que totes aquelles superfícies que siguin atenuants l’implementin. El que fa és simplement baixar en 1 el nivell de so que li arriba per paràmetre. En el nostre cas només l’implementa la classe Wall, però es podria escalar i fer atenuant tot el que es desitgi.

Per poder dur a terme el sistema d’atenuació es va haver de fer colliders que no interactuen amb res a totes les parets i tot allò que volíem que fos atenuant.

Imagen que contiene con baldosas, pequeño, lavabo, verde

El contenido generado por IA puede ser incorrecto.

En el nostre cas, com ja hem comentat anteriorment, només hem fet atenuants les parets del mapa.

#### 5.1.3. EL SO I ELS ENEMICS

Tenint el compte la funció de ListenSound dels enemics comentada a l’apartat 5.1.1. COM ES PRODUEIX EL SO, entrarem en detalls sobre el seu funcionament.

Una vegada l’enemic ha rebut un so, el primer que fa es fer un RaycastAll cap a l’origen d’aquest per tal de veure si hi ha algun element atenuant pel camí i si n’hi ha, s’atenua el so. Això, però només és té en compte si la distància entre l’enemic i l’origen del so és més gran que cert llindar, ja que si no se li posen uns valors ja definits.

### 5.1.2. ENEMICS

#### 5.1.2.1. ENEMIC GROS

#### 5.1.2.2. ENEMIC RÀPID

#### 5.1.2.3. ENEMIC CEC

### 5.1.2. INTERACCIÓ DEL PLAYER AMB EL MÓN

La interacció del Player amb el món és una part vital de Silentium, ja que hi ha moltes coses amb les quals s’ha d’interactuar (portes, objectes, puzles...).

Abans d’entrar en detalls comentarem que aquest sistema es va haver de refactoritzar perquè no era eficient i estava molt mal planetejat. Per aquest motiu aquest apartat està dividit en dos: el primer sistema, el segon i l’abans i després del canvi de sistema.

#### 5.1.2.1. PRIMER SISTEMA

Durant quasi tota la duració del projecte es va fer servir un sistema d’interacció molt poc eficient i estructurat. El que es feia era que per cada entitat interactuable es feia una layer diferent. A continuació es feia una sèrie molt llarga d’if que mirava de quina layer era allò que el Player està mirant i segons aquesta s’activava un booleà o un altre. A part, s’havia de fer moltes comprovacions si volíem que certes entitats interactuables tinguessin diferents comportaments com les portes per exemple. Aquest booleà servia a l’hora d’interactuar amb allò que s’està mirant, és a dir, a l’hora de fer l’acció, es tornava a fer una sèrie immensa d’if, però que en comptes de mirar la layer, mirava quins booleans estaven activats.

Això suposava un altre problema: el Player era qui s’encarregava de molta de la lògica de la interacció, la qual cosa no creiem que era l’adequat.

#### 5.1.2.2. SEGON SISTEMA

Teníem molt clar que el sistema que havíem muntat estava molt malament fet i ho volíem canviar, així que vam decidir crear una interfície anomenada IInteractuable.

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

Aquesta interfície es va crear amb l’objectiu que totes aquelles entitats que fossin interactuables l’implementessin. D’aquesta manera el Player només hauria de mirar si el que està mirant implementa la interfície i a l’hora d’interactuar només hauria de cridar a la funció pertinent. A més, va facilitar molt la comprovació de si se li havia de posar el shader del qual es parla a l’apartat 5.3.1.1. OUTLINE SHADER, ja que volíem que a certs elements com les portes no se’ls apliqués.

InteractuableItem per exemple representa a tots els objectes que el jugador pot agafar i quan el Player crida la funció d’Interact l’objecte s’encarrega de fer tota la lògica corresponent. D’aquesta manera la lògica està ben repartida entre els scripts corresponents.

#### 5.1.2.3. ABANS I DESPRÉS

En aquest apartat mostrarem les diferències entre els dos sistemes, ja que les diferències son tan grans que veiem interessant mostrar-les.

**Layers:**

Layers utilitzant el primer sistema:

Imagen en blanco y negro

El contenido generado por IA puede ser incorrecto.

Layers utilitzant el segon sistema:

Imagen que contiene Gráfico

El contenido generado por IA puede ser incorrecto.

Com es pot veure, la diferència de quantitat de layers és molt gran i amb el segon sistema tot queda més net i té més sentit que amb el primer sistema.

**Player:**

Pel que fa al script del Player, cal destacar que el codi va passar de tenir 900 línies aproximadament a tenir-ne 700.

### 5.1.3. PUZLE AMB IA

Hi ha un puzle a Silentium que fa servir IA de classificació per tal de reconèixer certs símbols que el jugador ha de fer per tal de resoldre el puzle.

#### 5.1.3.1. FUNCIONAMENT DE LA IA

La nostra IA prové de Teachable Machine, de la qual hem parlat a l’apartat 3.6.1. TEACHABLE MACHINE i en aquest apartat entrarem en detalls del funcionament de la IA i dels diferents punts clau que fan que funcioni.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Com es pot veure, a Teachable Machine hi ha 3 apartats clau per a crear la nostra IA: les classes, la preparació i la vista prèvia. Primer de tot, a l’esquerra, tenim les classes que la nostra IA farà servir per reconèixer els diferents símbols, ja que, com hem comentat anteriorment, la nostra IA és de classificació i intentarà classificar l’input / mostra que li arriba en alguna d’aquestes classes. A la part del centre tenim el següent apartat clau: la preparació. Aquest apartant és molt important per tal que la nostra IA funcioni correctament. Hi ha diferents punts en aquet apartat: les èpoques, la mida del lot i la taxa d’aprenentatge:

* **Èpoques:** La quantitat de vegades que la nostra IA processarà les classes que li posem. És a dir, si li passem 100 imatges de gossos i només 1 època, les “mirarà” només una vegada. Quant més èpoques hi hagi millor, però fins un cert punt, ja que si són masses èpoques pot causar un sobre ajustament. Un sobre ajustament significa que la IA ha recorregut tantes vegades aquelles imatges que se les ha après de memòria en comptes de generalitzar. Per exemple: si a la nostra IA l’entrenem amb 5 fotos de pomes i 5 de plàtans amb 100 èpoques el que succeeix és que el model memoritza perfectament aquelles 10 imatges. Aquestes 10 imatges la reconeixerà perfectament, però si se li mostra una altra poma totalment nova, no la reconeixerà, ja que no ha aprés a generalitzar, només ha memoritzat aquells 10 exemples.
* **Mida del lot:** La quantitat de mostres que s’utilitzen alhora per a ajustar la IA. En el cas de la imatge té com a valor 16, per tant, entrenarà de 16 imatges en 16. Una vegada ha passat per totes les imatges es completa una època. S’ha d’anar amb compte, ja que és molt fàcil que la IA se sobre ajusti, ja que el la mida del lot ha de ser més petita que la quantitat de dades per tal que pugui aprendre i ajustar els pesos correctament[[1]](#footnote-1).
* **Taxa d’aprenentatge:** Controla què tan ràpid aprèn el model, és a dir, quant canvia cada pes en cada ajustament. El problema és que si la taxa és molt alta aprendrà molt ràpid, però es molt possible que no ajusti correctament els pesos i, per tant, la IA no funcioni com s’esperava. Va tan ràpid que realment no està aprenent bé.

L´últim punt clau requereix un exemple i per aquest motiu hem creat una mini IA: Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Com es pot veure a la imatge, la part dreta mostra de la mostra que li hem proporcionat en quina classe la classifica. Aquesta IA té 4 classes d’entre 50 i 60 imatges cadascuna i els valors de preparació que Teachable Machine proporciona per defecte. De primeres ja es pot veure que el símbol de classe 1 l’ha reconegut, però no del tot, tot i que és una de les imatges utilitzades a l’entrenament i hauria de donar 100%. Per poder saber com ha aprés la IA hi ha certs punts que hem de tenir presents:

* **Precisió per classe:** Avaluació que fa la IA reservant el 15% de les imatges proporcionades com a mostra i que ensenyen.   
  Tabla

  El contenido generado por IA puede ser incorrecto.  
  En aquest cas es pot veure que en general la precisió és molt baixa en la meitat de les classes. Això és perquè segurament les èpoques amb les quals ha sigut entrenada són molt poques. Això genera una pregunta: llavors per què la IA ha dubtat tant a l’hora de reconèixer la imatge? Això es pot mirar a la **matriu de confusions.**
* **Matriu de confusions:** Determina per cada classe les que tenen molt percentatge de confusió:

Gráfico, Gráfico de dispersión

El contenido generado por IA puede ser incorrecto.

Es pot apreciar en aquesta imatge que de les 10 mostres de la classe 1 que la IA agafa per avaluar les ha detectat totes correctament. En canvi, de les 8 mostres de la classe 2, 5 les ha identificat com a classe 1. Amb la classe 3 ha succeït una cosa semblant, però pitjor que ens ajuda a entendre el resultat de la precisió per classe: de les 9 mostres, les 9 les ha classificat com a 9. Per aquest motiu la precisió era de 0 a la classe 3. Amb aquests resultats, llavors, veiem que la nostra IA confon molt la classe 2 i la 3 amb la 1 i com que la precisió es tan baixa, el % global baixa i per aquest motiu la nostra IA no arriba al 100% a l’hora de reconèixer la imatge.

#### 5.1.3.2. IMPLEMENTACIÓ A UNITY I INTEGRACIÓ AMB EL PUZLE

Per poder implementar la nostra IA a Unity hem hagut de fer servir un paquet anomenat Barracuda, del qual hem parlat per sobre a l’apartat 3.2.4. BARRACUDA. El problema amb aquest paquet és que hem hagut de fer servir *INTRODUCIR REFERENCIA A LA LIBRERIA*, ja que Barracuda només accepta models amb l’extensió .onnx i Teachable Machine no ofereix aquesta opció per a exporta-lo.

La integració de la IA amb el puzle és el següent: hi ha una càmera que és el que veurà la IA i una altra que és per la qual veu el jugador. Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

La càmera de la IA és ortogràfica i apunta a una RenderTexture. Aquesta RenderTexture és la que fem servir per tal de crear la imatge (textura) que veurà la nostra IA:

Forma, Rectángulo

El contenido generado por IA puede ser incorrecto.  
Això és el que la nostra IA agafa com a mostra per a classificar. En aquest cas com que no és una de les figures correctes ha donat un valor quasi de 1[[2]](#footnote-2) en la 7, la classe que utilitzem per a classificar totes aquelles imatges que no són correctes: Patrón de fondo

El contenido generado por IA puede ser incorrecto.

#### 5.1.3.3. DIFICULTATS TROBADES

La creació d’aquesta IA ha estat tot un repte, ja que no teníem cap tipus de coneixement de com entrenar-la més enllà dels coneixements bàsics que sabíem o que se’ns va proporcionar al propi Teachable Machine. Per aquest motiu vam tenir diversos problemes: el primer problema és que vam començar a entrenar la IA amb moltes poques imatges i per tant no era molt precisa. Una vegada ja vam posar més imatges de mostra el problema era que les figures incorrectes les intentava classificar en alguna de les classes i per aquest motiu una simple línia podia ser classificada com a qualsevol símbol. Per aquest motiu vam haver de crear una classe extra on només hi hagués imatges que voliem que la IA no reconegués com a símbols.

Tot això suposava un problema, ja que per entrenar la IA necessitàvem moltes imatges (100 aprox.) per cada classe, així que ens vam recolzar una mica en ChatGPT per a la creació d’aquestes.

Finalment, per cada nova IA havíem d’ajustar les èpoques i va arribar un punt on no sabíem si la IA no funcionava per les imatges de mostra, per les èpoques, per la conversió a .onnx o per la imatge que estava veient. Això sense tenir en compte que depenent de la quantitat d’imatges i d’èpoques la IA trigava molt entrenar.

Per tots aquests motius el procés de creació de la IA va ser molt lent i tempestuós.

## 5.2. DADES

### 5.2.1. OBJECTES

### 5.2.2. INVENTARI

### 5.2.3. DESAT

## 5.3. INTERFÍCIES

### 5.3.1. SHADERS

#### 5.3.1.1. OUTLINE SHADER

#### 5.3.1.2. GLITCH SHADER

### 5.3.2. EFECTES DE POST-PROCESSAT

### 5.3.3. PARTÍCULES

### 5.3.4. WAVE EFFECT

### 5.3.4. MODEL VISTA-CONTROLADOR

* Parlar de l’inventari i com funciona el sistema d’objectes. Hacer UML del inventario.
* Parlar del sistema de So i com funciona.
* Parlar del Player i com funciona l’interacció amb el món. HACER UML por si se hace lo de interactuable
* Parlar dels enemics i les diferents màquines d’estats. UML de los enemigos.
  + Parlar del navmesh i els links...
* Parlar de com funciona el sistema dels puzles. Explicar cada puzle en específic.
* Parlar de la IA i com funciona la seva implementació. Paraules clau, que es noti que hem investigat.
* UML del desat de la partida i explicar què desem i per què.
* Explicar els shader (outliner i glitch), el sistema de post-processat i el depth de la pistola. Explicar com funciona el sistema de les waves que funcionen segons el so.

# 6. TESTING I RESULTATS

## 6.1. TESTING

## 6.2. RESULTATS DE L’APLICACIÓ

## 6.3. OBJECTIUS

### 6.3.1. OBJECTIUS ASSOLITS

### 6.3.2. OBECTIUS NO ASSOLITS

# 7. CONCLUSIONS

## 7.1. DIFICULTATS

## 7.2. MILLORES PROPOSADES

## 7.3. VALORACIÓ FINAL

# 8. BIBLIOGRAFIA

## 8.1. WEBS I VÍDEOS CONSULTATS

## 8.2. CRÈDITS

### 8.2.1. MODELS I TEXTURES

### 8.2.2. MÚSICA I SFX

# 9. ANNEXOS

1. La IA no veu imatges, sinó que veu números. Pasa aquests números per diverses capes de neurones artificials. Cada neurona té un pes i el model multiplica les dades pels pesos i decideix que X és X. La gràcia és que cada vegada que s’equivoca els pesos s’ajusten. L’ajustament succeeix al final de cada época. [↑](#footnote-ref-1)
2. La IA avalua en un rang de 0 a 1, sent 1 una coincidència exacte i 0 no coincideix en res. [↑](#footnote-ref-2)