Universitat de Girona
Escola Politècnica Superior

Grau en Enginyeria Informàtica

PROJECTE FINAL DE GRAU

Disseny i desenvolupament d'un videojoc "rogue-like" d'acció en temps real

Autor:
Albert Falgàs Ribas

Tutors:
Gustavo Ariel Patow
Sergio Gonzalo Besuievsky Glikberg

RESUM

Convocatòria:
Setembre 2023

Departament :
Departament d'informàtica, matemàtica aplicada i estadística

1 Introducció

El procés de creació d'un videojoc consisteix en la suma i coordinació de múltiples disciplines molt diferents entre elles. Això fa que siguin productes complexos els quals solen requerir un equip gran i divers per poder produir un producte a l'altura dels estàndards de qualitat i extensió que avui dia existeixen en la indústria.

En els darrers anys hem pogut observar, però, el naixement de diversos projectes independents o "indies", els quals sense el suport d'una gran empresa i amb un petit equip de persones o inclús una sola, han esdevingut obres de referència d'una enorme qualitat i originalitat.

Aquests jocs sovint destaquen per l'ús de tècniques de generació procedimentals, les quals fan recaure sobre el desenvolupador la càrrega més gran de feina, ja que aquest pot expandir la duració i atractiu del joc, delegant la combinació aleatòria de diferents components a algoritmes. El fet de reduir la dependència d'àrees alienes a la programació fa que aquests tipus de projectes siguin ideals per a petits equips que volen entrar dins la indústria, amb propostes que destacaran per la jugabilitat, però que no tenen per què decaure en temes d'atractiu artístic, ja que aquests no han de ser produïts de manera massiva.

El propòsit d'aquest projecte és desenvolupar un primer prototip de videojoc que faci ús de tècniques de generació per procediments per tal d'expandir-ne la duració i jugabilitat. Això s'aconseguirà, concretament amb la generació de nivells que consistiran en sales interconnectades entre si de manera aleatòria, amb un inici i un final. Tot això dins d'un marc de videojoc de perspectiva superior de tipus "rogue-like" de trets, inspirat en gran manera amb els videojocs *Enter the Gungeon* i *The binding of Isaac*.

2 Objectius

Aquest projecte se centra en els següents objectius:

- Estudi del motor *Unity* i del llenguatge C#.
- Estudi, disseny i implementació de mecanismes que expandeixin la jugabilitat minimitzant la creació de nous elements específics per aquest propòsit (models d'enemics, trampes, armes, terrenys...).
- Generació de nivells aleatoris dividits en sales interconnectades.
- Sistema de moviment i interacció del jugador amb l'entorn.
- Implementació d'enemics amb tècniques d'intel·ligència artificial.
- Estudi i creació de "shaders".
- Desenvolupament de menús.
- Desenvolupament de la interfície de joc.

3 Procés de desenvolupament del PFG

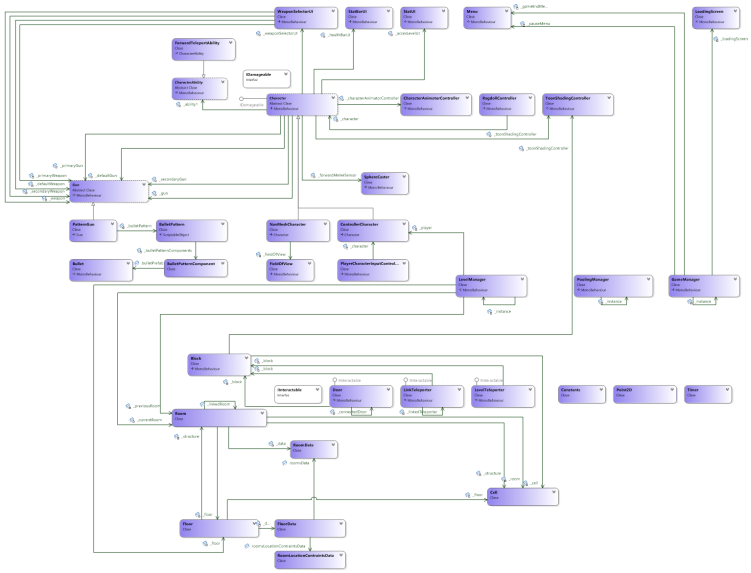

FIGURA 1: Diagrama de les classes implementades en el projecte.

3.1 Generació del pis

Un pis es genera a través de les indicacions d'un fitxer *JSON* del qual podem distingir-ne 3 parts: la informació general, la informació de les habitacions que el componen i les restriccions sobre com han de col·locar-se. Havent carregat aquesta informació, podem començar amb la generació del pis.

3.2 Personatge

Els personatges s'han implementat com una composició de components originals de *Unity* i propis amb un component gestor derivat de *Character*, *ControllerCharacter* o *NavMeshCharacter*, que s'encarrega de mantenir el seu estat i de servir de interfície per a controlar tota la seva lògica d'accions. Una gestió important que aquest component realitza es l'associació de les animacions del personatge segons l'arma i acció utilitzada.

Jugador

El personatge controlat pel jugador es caracteritza per moure's a través de moviments indicats pels controls que faci servir, és per això que utilitza la implementació de *Character* amb *ControllerCharacter*. Això es produeix a través d'una cadena on *PlayerInput* capta els "inputs", *PlayerCharacterInputController* els captura, els processa i els passa a *ContollerCharacter*, i finalment *ControllerCharacter* segons l'accio i l'estat, les solicita al *CharacterController*, la *CharacterHability* o la *Weapon* seleccionada.

Enemics

Els personatges controlats per les IAs es caracteritzen per moure's a través d'un *NavMesh*, essent indispensable per a ells que siguin col·locats sobre un i que addicionalment tinguin un component de tipus*NavMeshAgent* que defineixi les seves característiques sobre aquest. És per això que aquests fan servir la implementació de

Character de *NavMeshCharacter*, ja que aquesta permet controlar-los oferint mètodes de control més adequats perquè la IA pugui treballar amb ells.

És també important destacar que aquests, per a poder detectar el seu entorn tenen un component de tipus *FieldOfView* el qual els proporciona un con de visió de rang determinat i és la manera que tenen de poder detectar el jugador.

Una altra característica rellevant és el component *BehaviourTreeRunner* el qual executa l'arbre de comportament que en determina les seves accions. És important destacar que alguns nodes que conformen aquest arbre treballaran amb les habitacions per tal que aquestes els hi proporcionin posicions del *NavMesh* de l'habitació que ocupen, per tal que així puguin indicar al *NavMeshAgent* quin és el seu destí.

3.3 Armes

Les armes consisteixen en una implementació genèrica i abstracta de *Gun*, que defineix una sèrie de paràmetres, estats (espera, disparant, recarregant, carregant, esperant disparar) i funcions d'interacció (prémer, deixar de prémer, recarregar, cancel·lar procés actual) comunes per a tots els tipus d'armes que s'implementin.

Pel que fa a la instanciació de projectils, una *PatternGun*, treballa amb un *BulletPattern*. Aquest *ScriptableObject* defineix un seguit de patrons radials de bales dels quals, quan l'arma ho sol·licita, s'encarrega d'instanciar cada bala en l'orientació indicada pel patró i la posició del punt de tret de l'arma. Cal dir, que la instanciació de les bales, no es fa de manera directa sinó que es demana una instància en desús de la bala en qüestió al *PoolingManager*.

Per acabar, les bales que el patró utilitza, tenen una implementació molt senzilla, la qual simplement consisteix en, un cop activades, desplaçar-se cap endavant fins a col·lidir amb un objecte, sostraient el mal que fan de la vida del personatge, o fins a exhaurir el temps de vida. Un cop hagin finalitzat, però, en comptes de destruir-se com és habitual, aquestes es desactiven i ho notifiquen al *PoolingManager* per tal que puguin ser reutilitzades.

3.4 Shaders

La implementació dels *shaders* s'ha dut a terme mitjançant l'editor visual de nodes de *Unity*, el *ShaderGraph*.

Els *shaders* que s'han fet amb aquesta eina són el *shader* d'estètica "cartoon" que comparteixen tots els models 3D i el *shader* que s'aplica a les partícules que formen les bales.

Toon Shader

La implementació base consisteix en un *shader* que simplifica les ombres projectades sobre els objectes, i els hi afegeix un contorn brillant ("rim light") i finalment carrega la sortida de color en el canal d'emissió en comptes del canal de color base, aconseguint una estètica menys realista i més pròpia a la del 2D. Aquesta implementació no és pròpia sinó que està extreta íntegrament de la implementació realitzada per *MinionsArt*.

No obstant això, per a poder fer que els objectes apareguin i desapareguin amb efectes especials s'ha hagut de modificar aquest *shader* de manera que inclogués aquest efecte.

Bales

Les bales, pel que fa a gràfics, s'han creat utilitzant el sistema de partícules de *Visual Effect Graph* en combinació amb un *shader* que simula un efecte d'electricitat. Aquesta combinació aconsegueix com a resultat una esfera elèctrica..

3.5 flux de joc

Per tal de gestionar el flux del joc, hem implementat dues classes, el *GameManager* i el *LevelManager*. Tot i la seva naturalesa de gestió, en utilitzar el cicle de vida de *Unity*, aquestes han estat implementades com a components amb un patró de *Singleton* i col·locades en un objecte buit dins l'escena de *Unity*.

GameManager

Es troba implementat com un seguit de mètodes que s'encarreguen d'iniciar la navegació entre les dues escenes (menú principal i joc) i mostrar els menús en la partida (fi de joc i pausa). Aquesta navegació es fa carregant la nova escena de manera asíncrona a través del *SceneManager* mentre es mostra la pantalla de càrrega.

LevelManager

S'encarrega de llegir el fitxer *JSON* de la generació del pis, construir un pis amb aquesta informació utilitzant *Floor* i col·locar el personatge a la primera posició del pis construït. Un cop acabada aquesta inicialització aquesta classe implementa mètodes per a controlar els diversos aspectes de la partida, que es redueixen a les transicions entre habitacions, la qual és la que provoca l'instanciació d'enemics.

4 Resultats

(A) Menú principal (B) Pantalla de càrrega (C) Menú de pausa

(D) Menú de derrota (E) Menú de victòria (F) Generació d'un pis

FIGURA 2: Captures de pantalla dels menús i generació del pis