La cerca del tresor, un videojoc competitiu en Unity.

David Santos Pascual

Grau d'Enginyeria Informàtica EPSEVG

Resum

S'ha dissenyat i implementat un videojoc competitiu (una persona contra una altra), utilitzant Unity com a eina per al seu desenvolupament.

S'han utilitzat un conjunt de patrons de disseny per al desenvolupament, tenint en compte la seva re-usabilitat i flexibilitat.

El videojoc consta amb un cicle de vida complet, generació procedural, rebre inputs tant de teclat com de gamepad, controlar les accions i els estats dels personatges en funció dels inputs rebuts, ofereix una mecànica prou divertida i components atraients per al jugador.

1. Introducció

La indústria dels videojocs ha experimentat acusat creixement en els últims anys, degut a l'evolució computacional, capacitat de processament, gràfics més realistes i l'estreta relació entre pel·lícules i videojocs. En conseqüència als consumidors els hi és més familiar, i reconeixent els títols més ràpidament.

Segons un informe, que inclou informació aportada per els estudis de desenvolupament espanyols, mitjançant una enquesta realitzada el maig de 2015, la xifra de facturació situa el sector del videojoc en Espanya entre les principals industries nacionals de continguts digitals. Segons les xifres del informe, en 2014 es van crear 70 noves empreses en aquesta indústria, amb un número total que ja supera les 400, i col·loca a Espanya en primer lloc a la llista d'Europa en número d'estudis de desenvolupament operatius.[1]

Donat el considerable creixement en aquesta indústria i el meu interès per als videojocs, vaig decidir endinsar-me en aquest món. Volia conèixer quines oportunitats oferia, que podia aprendre de nou i com podria adaptar els meus coneixements adquirits durant la carrera per aprofitar-los al màxim en el desenvolupament de videojocs.

A més a més, volia crear un producte propi i estimular la creativitat, no només dissenyant l'estructura tècnica, sinó també dissenyant el concepte del videojoc.

2. Objectius

L'objectiu principal del projecte és realitzar un videojoc 2D competitiu, divertit i funcional, utilitzant les eines existents en el mercat.

A continuació es citen els principals objectius derivats de la creació d'una *alpha* del videojoc:

- Escollir i aprendre a utilitzar la eina adient per al desenvolupament.
- Controlar els diferents estats durant la seva execució (cicle de vida).
- Rebre els diferents inputs dels jugadors i realitzar les accions corresponents.
- Permetre enviar inputs a través de teclat o *gamepad* amb una configuració de controls còmode.
- Generació procedural del nivell en cada partida.

- Aprendre i conèixer com es treballa en el àmbit del desenvolupament de videojocs.
- Aprendre nous patrons de programació i aplicar els ja coneguts.

3. Tecnologies

Donada les necessitats del projecte, es van haver d'investigar els diferents *game engines* del mercat per tal de seleccionar el més adient.

3.1. Necessitats

Característiques mínimes a tenir en compte per el *game engine*:

- **Usabilitat**: El seu aprenentatge i desenvolupament ha de ser relativament senzill. Interfície gràfica amigable.
- **Funcionalitat**: Ha de proporcionar les funcions necessàries per desenvolupar un videojoc en 2D. El seu *workflow* ha d'estar el màxim adaptat per l'edició 2D.
- **Documentació**: Ha d'existir el contingut i exemples suficients per aprendre el seu ús.
- **Preu**: El menor possible.

3.2. Elecció

Unreal Engine 4 es massa verd i encara no està lliure de d'errors. És necessari una gran potencia a nivell de hardware cosa que implicaria invertir en nou maquinari. A més a més no està enfocat per al desenvolupament de videojocs en 2D.

CryEngine no proporciona cap llicència gratuïta. No té suport total per el desenvolupament 2D.

Game Maker Studio no conté un gestor de textures en el model gratuït. El llenguatge que utilitza per *scripting* es un de propi i no es interessant aprendre un llenguatge que només es pot utilitzar en el seu motor.

La versió graulita que ofereix Construct2 prescindeix de característiques útils en termes de desenvolupament general i el llenguatge que utilitza és *Javascript*, que personalment prefereixo evitar.

Finalment el *motor engine* escollit és Unity3D. Proporciona suport total per desenvolupar jocs en 2D. Existeix una immensa quantitat de contingut i documentació disponible. Disposa de llicència gratuïta amb totes les característiques del motor disponibles. El llenguatge que utilitza per *scripting* és C#, interessant en aprendre ja que pot ser utilitzat altres àmbits de desenvolupament.

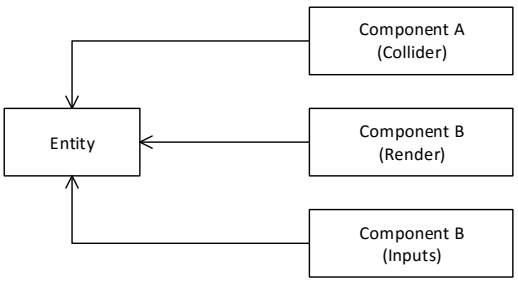
4. Unity3D

Unity és un motor de videojocs desenvolupat per Unity Technologies. Utilitzat per creat videojocs per PC, consoles, dispositius mòbils i pàgines web. El seu primer llançament va ser en el 2005.

4.1. Arquitectura del software

Unity està basat en el concepte *Entity Framework*[2]. El concepte *Entity Framework* està inspirat en el patró de disseny *Entity-Component*. Una entitat conté múltiples dominis. Per mantenir els dominis aïllats, el codi de cada un es col·locat en la seva pròpia classe de component. L'entitat es reduïda a un simple contenidor de components[3].

En Unity, les entitats s'anomenen *GameObjects* i els components estan basats en components propis d'Unity o en implementacions (herència de la classe base *MonoBehaviours*) d'scripts anomenats *Script Component*.



Il·lustració del patró Entity-Component

Totes les entitats són instanciades en una escena. La escena representa el món virtual del videojoc. No totes les entitats que apareixen en una escena són visualitzades per el jugador quan executa el videojoc. Es necessari afegir una entitat amb un

component de càmera i configurar-lo per a que renderitzi la zona de l'escena desitjada.

4.1.1. Funcions d'event

Els scripts d'Unity no són com la idea tradicional de programar en que el codi s'executa contínuament dins d'un bucle fins que completa la tasca. En comptes d'això, Unity passa el control als scripts de forma intermitent cridant determinades funcions que hi ha declarades en ell. Un cop la funció ha acabat la seva execució, el control es retorna a Unity, en resposta a *events* que han succeït durant el *gameplay*.

En cada actualització de *frame (tick del videojoc)* totes funcions d'*events* són cridades per a cada *script*.

5. Disseny del videojoc

El disseny del videojoc consta de les característiques bàsiques, mecàniques, objectius i elements que tindrà el joc. S'explica informació referent els controls, càmera i interfície d'usuari.

5.1. Perspectiva general

El títol del videojoc és: *Super Cute Miners*. Es tracta d'un joc 2D, d'acció, en que el seu objectiu principal consisteix en trobar el tresor abans que el contrincant. El temps aproximat del videojoc són entre 5 i 10 minuts.

5.2. Modes de joc

L'únic mode de joc que oferirà el videojoc és multi-jugador local. Dos jugadors en la mateixa màquina, els quals podran utilitzar el mateix teclat, dos *gamepads* diferents o un teclat i un *gamepad*.

5.3. Mecànica del joc

L'objectiu del jugador és trobar el tresor abans que el contrincant. El tresor està enterrat sota terra.

Excavant també es poden trobar metalls. Aquests són intercanviables en una botiga situada a la superfície del mapa. A canvi dels metalls es poden obtenir millores que facilitaran la tasca de trobar el tresor (reforç positiu al jugador per sortir a l'exterior i deixar de picar).

5.4. Mapa

El mapa es genera aleatòriament a partir de blocs en cada partida.

Per tal de dificultar la cerca del tresor, sota terra hi ha una "boira" la qual no permet als jugadors veure més enllà d'un radi d'un bloc de distància.

Per a que el jugador pugui tornar a la superfície, es generen unes escales automàticament en el la posició dels blocs destruïts en vertical (a partir d'una *y* distància del terra excavat).

5.5. Tipus de blocs

En el mapa existeixen diferents tipus de blocs els quals es poden classificar per les seves funcionalitats:

- Defineixen l'entorn i l'estructura bàsica del terreny del mapa: *Herba, Terra, Pedra i Pedra mare*.
- Indiquen si el tresor està a prop o lluny de la localització on s'està excavant: *Pedra dura, Pedra super-dura, Pedra mega-dura i Obsidiana.*
- Els metalls són un tipus de bloc especial perquè és una combinació d'un bloc normal amb un metall. Metalls intercanviables en una botiga situada a la superfície del mapa. A canvi dels metalls es poden obtenir millores: *Coure, Ferro, Plata i Or*.

Cada tipus de bloc té una resistència (vida) la qual determina la dificultat per trencar-lo.

Per simular l'estat d'un bloc quan és malmès, s'aplicarà una textura de "trencament" depenen de la resistència restant del bloc.

5.6. Tresor

Component essencial que defineix l'objectiu principal del joc. El primer jugador que aconsegueix picar el tresor es qui guanya la partida. La partida finalitza amb un guanyador.

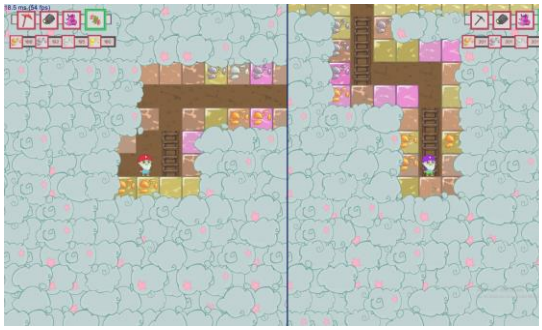
5.7. Jugador

El jugador portarà el control d'un personatge miner el qual pot realitzar diferents accions: saltar, moure's, picar blocs, escalar i comprar en la botiga.

Existeixen uns objectes especials que el miner portarà equipat durant la partida de forma permanent. Aquests objectes proporcionen una sèrie d'habilitats especials que permeten al jugador facilitar la cerca del tresor. Els objectes, i per tant les seves habilitats, es podran millorar durant el transcurs de la partida.

5.8. Càmera

Hi ha una càmera diferent per a cada jugador (*split screen*). Cada càmera segueix al seu jugador corresponent. La càmera mostra una part del mapa corresponent a un radi del jugador.



Pantalla dividida en dues parts (split screen).

6. Disseny tècnic

S'ha creat un conjunt de classes genèriques identificables amb el prefix CGF (*Chloroplast Games Framework*). També he utilitzat i adaptat alguns patrons de disseny als requeriments del projecte: *MVC i State*.

6.1. Patrons de disseny

Els patrons de disseny s'utilitzen per cercar solucions a problemes comuns en el desenvolupament de software.

Per a que una solució sigui considerada un patró de disseny ha de tenir dues característiques:

- Comprovat la seva **efectivitat** resolent problemes similars en ocasions anteriors.
- Ha de ser **reutilitzable**, o sigui, aplicable a diferents problemes de disseny en diferents circumstancies.

Els patrons de disseny pretenen:

- Proporcionar elements reutilitzables en el disseny de sistemes de software.
- Evitar la repetició de cerca de solucions a problemes ja coneguts i solucionats anteriorment.
- Formalitzar un vocabulari comú entre dissenyadors.
- Estandarditzar un mètode de realització del disseny.
- Facilitat en el aprenentatge de noves generacions de dissenyadors, condensant coneixement ja existent.

Els patrons que han estat utilitzats en el projecte són:

- **Singleton**[4]: Garanteix que un objecte només serà instanciat una sola vegada. Accés global.
- **MVC (Model - Vista - Controlador)**[5]**:** Compost per tres components: Model, Vista i Controlador. Té com objectiu separar les dades i la lògica, de la vista i el controlador. La vista i el controlador depenen del model, però ell no depèn de cap dels altres.
- **State**[6]**:** Permet que un objecte modifiqui el seu comportament cada vegada que canvia el seu estat intern. Útil per modificar el comportament d'un objecte en temps d'execució.
- **Observer**[7]**:** Defineix una relació de un a molts, permetent així que quan un dels objectes canvia el seu estat, notifica als demès dependents del canvi.

6.2. Jugador

He decidit aplicar el patró MVC en el jugador per separar el màxim possible la vista (animacions), model (components i dades) i controlador (gestió d'inputs i màquina d'estats). El MVC permet tenir la majoria de parts importants del jugador separades, organitzades, flexibles i fàcilment reutilitzables.

Descripció de les responsabilitats de cada classe:

- **CGFView**: Classe base la qual emmagatzema el component d'Unity encarregat de controlar les animacions.
- **PlayerAnimatorController**: Controla el comportament del component *Animator*, encarregat de reproduir les animacions.
- **CGFModel**: Classe base la qual emmagatzema la vista (*CGFView*) del patró.
- **PlayerModel**: Emmagatzema i controla tots els components que necessita el jugador per realitzar les seves accions, com per exemple: moure's , saltar i picar.
- **CGFInputController**: Classe base sense cap responsabilitat. Reservada per futures ampliacions o canvis.
- **CharacterInputController**: Classe on es defineixen i s'emmagatzema l'estat actual de les accions del jugador.
- **PlayerInputController**: Vincula uns determinats inputs amb les accions del jugador i comprova els seus estats de pulsació.
- **PlayerController**: Encarregat d'actualitzar l'estat actual de la màquina d'estats del jugador a cada *tick* del videojoc. A més a més conté les referencies als diferents components del MVC: *PlayerAnimatorController*, *PlayerModel* i *PlayerInputController*.

6.2.1. Inputs

Unity incorpora el seu propi sistema d'administració d'inputs i mètodes per obtenir els valors necessaris. No obstant, he optat per crear un conjunt de classes les quals fan de pont entre el *Input Manager* d'Unity i el patró MVC desenvolupat pel jugador. Aquest pont aïlla els mètodes específics d'Unity encarregats de la detecció d'inputs. El MVC i la FSM del jugador desconeix l'existència d'aquests mètodes. D'aquesta manera tenim tots els mètodes específics del *Input Manager* aglutinats en una sola classe i no escampats per als diferents estats de la FSM del jugador. Així, un canvi en un input només suposarà el canvi d'un atribut, que a més a més ja sabem on es troba.

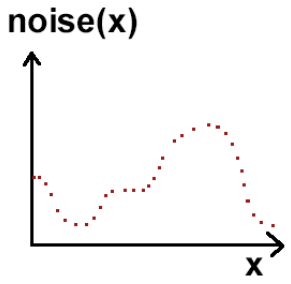
6.2.2. Script Components

Per definir el comportament del jugador, he dissenyat una col·lecció de script components. Cada un d'ells serveix per una funció o acció bàsica diferent. Per al seu ús només cal afegir els components del tipus script a l'entitat desitjada, en aquest cas, el jugador.

7. Implementació

7.1. Generació del terreny

El terreny es generat de forma procedural en cada nova partida.



1D Perlin noise.

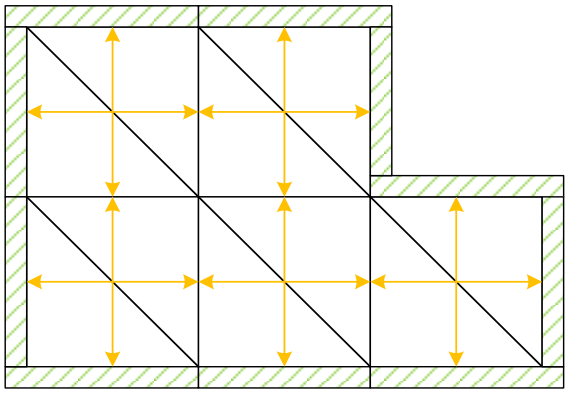
En aquest projecte s'ha utilitzat l'algoritme de "Perlin Noise" per la generació procedural.

Bàsicament la funció de Perlin Noise retorna valors entre 1 i 0, basant-se en els valors d'input (*x,y*). Per obtenir un *output* personalitzat, he implementat un mètode anomenat *GetUnityNoise*.

Primer de tot, el mètode divideix els valors d'input (*x*,*y*) per una escala (*scale*). Com més properes són les coordenades, més similars són els valors que retorna. Multiplica aquest resultat per una magnitud (*mag*). Aquest procés serveix per crear turons, ja que la funció només retorna valors d'entre 1 i 0, es necessari crear valors més elevats.

7.2. Optimització dels colliders del terreny

He creat un algoritme que generi només *collider* en les parets que estan en contacte amb l'exterior. L'algoritme comprova els quatre veïns del bloc, en cas d'estar en contacte amb un veí sòlid, llavors no genera *collider* per aquesta paret.



Comprovació dels veïns per cada bloc.

8. Conclusions

8.1. Resultats

S'ha creat una *alpha* del videojoc que permet jugar i experimentar l'objectiu principal: excavar i trobar el tresor abans que el contrincant; utilitzant Unity com a eina per al seu desenvolupament.

El videojoc posseeix un cicle de vida complet: menú d'inici, execució partida, finalització de la partida, finalització del joc.

En cada partida el terreny es generat proceduralment.

Els jugadors poden enviar inputs i accionar el seu personatge fent ús tant del teclat com del *gamepad*.

Moltes funcionalitats de les que s'han programat per diferents elements del videojoc, han estat pensades per ser reutilitzables o fàcilment modificables per utilitzar en altres projectes. Gràcies a patrons de disseny tals com: *MVC, State* i *Observer*.

El videojoc es caracteritza per una mecànica prou divertida i uns components que atrauen al jugador.

Un dels majors problemes que he trobat realitzant el projecte ha estat sincronitzar la maquina d'estats de les animacions que proporciona Unity, amb la maquina d'estat que he creat per a la lògica del jugador. La fusió del patró *MVC* amb una maquina d'estats de la lògica del jugador, ha ajudat a bastant en aquest procés i ha facilitat la gestió de les reproducció de les animacions amb el comportament del jugador.

En definitiva, la realització del projecte ha estat bastant satisfactòria, ja que es pot experimentar amb prou fiabilitat l'idea del joc dissenyat i com podria arribar a ser en una versió final.

8.2. Treball futur

Una possible millora en aquest projecte seria implementar el mode multi-jugador on-line. A més a més d'adaptar el videojoc a altres plataformes com *tablets* i així permetre jugar dues persones que es troben en diferents dispositius. També es podria afegir un mode d'un sol jugador amb la incorporació d'una IA per a l'oponent.

És indispensable afegir efectes de so i música de fons, per millorar l'experiència del jugador.

Afegir una pantalla o escena amb les instruccions necessàries per a que els jugadors sàpiguen la llegenda de cada color de bloc i els controls per poder jugar (mini-tutorial).

També seria necessari millorar la qualitat d'alguns gràfics i afegir components nous al joc, com l'obtenció i ús d'objectes usables dins del terreny.

L'efecte de boira de guerra (*fog of war*) que s'utilitza és acceptable, però s'hauria de millorar l'efecte de desaparició. També seria necessari afegir alguns efectes a la partida, com per exemple despendre sorra quan un bloc és destruït, per donar més realisme i vida al videojoc.

Caldria revisar l'optimització d'alguns elements renderitzats, com per exemple el fons de la partida.

Les possibles derivacions d'aquest projecte podrien donar a peu a un producte comerciable, ja que realitzant els canvis descrits podria ser un videojoc amb cara i ulls.

Referencies

[1] EL MUNDO. *La industria española del videojuego aumenta su facturación un 31% en 2015.* [Consulta: 07/10/2015] Disponible a: http://www.elmundo.es/cultura/2015/07/13/55a3c43bca474149588b457e.html

[2] SMITH, Scott. *IoC Container for Unity3D – part 1.* [Consulta: 13/05/2015] Disponible a: http://www.sebaslab.com/ioc-container-for-unity3d-part-1

[3] NYSTROM, Robert. *Component.* [Consulta: 13/05/2015] Disponible a: http://gameprogrammingpatterns.com/component.html

[4] NYSTROM, Robert. *Singleton.* [Consulta: 21/05/2015] Disponible a: http://gameprogrammingpatterns.com/singleton.html

[5] MICROSOFT. M*odel-View-Controller*. [Consulta: 21/05/2015] Disponible a: https://msdn.microsoft.com/en-us/library/ff649643.aspx

[6] NYSTROM, Robert. *State.* [Consulta: 21/05/2015] Disponible a: http://gameprogrammingpatterns.com/state.html

[7] NYSTROM, Robert. *Observer.* [Consulta: 21/05/2015] Disponible a: http://gameprogrammingpatterns.com/observer.html