TecnoCampus

Centres universitaris adscrits a la

upf. Universitat Pompeu Fabra Barcelona

Grau en Disseny i Producció de Videojocs

Desenvolupament d'una eina per a la producció de jocs

RPG a Unity

Memòria

Joan Ortiga Balcells

Tutor: Jorge Arnal Montoya

Memòria final

Curs 2021-2022

Agraïments

Agraeixo enormement al meu tutor que m'ha pogut solucionar dubtes i encarrilar amb el meu TFG.

Agraeixo a la meva família, i els meus amics, en especial a l'Àlex i el Marc, que m'han acompanyat durant un tram important del TFG.

Finalment, moltes gràcies a la meva parella Aylin per ajudar-me, tenir paciència amb mi, i aconseguir mantenir les meves ganes i motivació durant tot el projecte.

Resum

Aquest projecte explora la implementació d'una eina per desenvolupar videojocs del gènere RPG de forma ràpida, fàcil i efectiva dins el motor gràfic Unity utilitzant C# i la tecnologia UI Toolkit d'Unity.

Resumen

Este proyecto explora la implementación de una herramienta para desarrollar videojuegos del género RPG de forma rápida, fácil y efectiva para el motor gráfico Unity utilizando C# y la tecnologia UI Toolkit de Unity.

Abstract

This project explores the implementation of a tool for developing videogames of RPG genre in a quickly, easily, and effectively way in the Unity game engine using C# and Unity's UI Toolkit technology.

Índex

Índex d'il·lustracions

Índex de taules

Glossari

Hardcode: Posar informació al codi directament, fent complicat canviar el valor per l'usuari (Cambridge, 2011).

Plugin: Programa informàtic que fa funcionar un altre programa més ràpid i/o tenir més característiques (Cambridge, 2022).

Backend: Part d'un programa informàtic que gestiona la informació (Cambridge, 2022).

Frontend: Part d'un programa informàtic que és vist i utilitzat directament per l'usuari (Cambridge, 2022).

Asset: Un asset és qualsevol element que s'utilitzi en un projecte de desenvolupament. Per exemple: un model 3D, una textura, un so, o una taula de dades (Unity, 2022).

Serialization: La serialització és el procés pel qual les dades es transformen en un format que es pot guardar i recuperar més tard (Unity, 2022).

Serialized Object: Un serialized object és un objecte d'una classe que conté tota la informació i la serialitza automàticament (Unity, 2022).

Script: Una peça de codi que et permet crear o modificar un comportament del programa (Unity, 2022).

Runtime: En temps d'execució. Quan el software s'està executant, es diu que està en runtime.

Indie: De baix pressupost.

1. Introducció

Les eines en l'àmbit de desenvolupament dels videojocs han guanyat especial rellevància en els últims anys on les necessitats de desenvolupament són més àmplies i diverses. Els motors gràfics no poden abastir per si mateixos tots els aspectes que molts desenvolupadors necessiten, i les eines poden cobrir aquestes necessitats.

Aquestes necessitats poden ser vàries, des d'ampliar funcionalitats, incrementar la velocitat en el flux de treball, millorar l'eficiència i el rendiment, entre d'altres.

Les eines poden ser publicades i opcionalment venudes en fòrums, o en botigues especialitzades, com la botiga oficial d'Unity anomenada Unity Asset Store, on es poden trobar quasi 10.000 eines (Unity, 2022).

Molts desenvolupadors produeixen eines pel desenvolupament d'un videojoc i després venen aquestes eines per aconseguir més rèdit econòmic; empreses més grans, però, les mantenen privades pel seu propi desenvolupament, com pot ser Ubisoft (David Lightbown, 2021), Obsidian Entertainment, Bethesda, Guerrilla Games, etcètera.

Aquest projecte ataca l'àmbit de les eines per desenvolupar videojocs del gènere RPG (role playing games) amb Unity, on només existeixen 2 eines amb elevat preu i que estan molt enfocades als seus propis sistemes de joc, no donant tanta independència a l'eina.

2. Marc teòric

El projecte es basa en el desenvolupament d'una eina per al motor gràfic Unity que faciliti el desenvolupament de jocs RPG. És per això que el marc teòric es divideix en 2 àrees principals. La primera en la qual es parla principalment de les eines en el desenvolupament de videojocs i Unity, i la segona que parla dels videojocs RPG.

2.1 Unity

Unity és una eina de software categoritzada com a motor gràfic, que s'utilitza àmpliament a la indústria dels videojocs. Tal com expressen a la seva pàgina web "Unity és molt més que la millor plataforma de desenvolupament en temps real del món, també és un ecosistema sòlid dissenyat per facilitar el teu èxit." (Unity, 2022)[1].

En paraules de David Helgason, fundador i actual CEO d'Unity, "Unity és un conjunt d'eines utilitzades per a construir videojocs, i és la tecnologia la que s'encarrega d'executar els gràfics, l'àudio, les físiques, les interaccions, el multijugador." (Craighead, Burke, & Murphy, 2008).

El llançament inicial del motor gràfic data del 2005, on es va publicar la primera versió. Anys després el motor ha evolucionat incorporant tota mena de tecnologies.

Unity permet l'ús a tothom, oferint el motor de forma gratuïta fins a un cert volum de guanys monetaris. Permet fer videojocs tant en 2 com en 3 dimensions.

Unity està escrit en llenguatge C++ (Bjarne Stroustrup, 1979), encara que el llenguatge de programació que l'usuari fa servir és C# (Microsoft, 2000) per programar la lògica del joc.

El motor permet l'exportació fins a més de 10 plataformes incloent algunes com PC, MAC, Linux, iOS, tvOS, WebGl, etcètera.

[1] "Unity is so much more than the world's best real-time development platform – it's also a robust ecosystem designed to enable your success." (Traducció de l'autor)

2.1.1 Motor gràfic

Un motor gràfic és una plataforma que s'encarrega de realitzar funcions comunes com el renderitzat, càlcul de físiques, input, etcètera, perquè els desenvolupadors (artistes, dissenyadors, programadors i altres), es puguin centrar en el desenvolupament del joc (Paul, Goon, & Bhattacharya, 2012).

Unity (Unity, 2021) defineix motor gràfic com "un software que ofereix diverses eines i característiques per poder fer videojocs de forma professional i eficient".

Els motors gràfics atorguen abstracció de la plataforma, permeten la publicació en diverses plataformes sense modificar gairebé el codi del videojoc (Paul, Goon, & Bhattacharya, 2012).

Els motors de videojocs sovint s'enfoquen en un gènere específic per poder oferir més funcionalitats i optimització. Alguns d'aquests gèneres són: Massive Multiplayer Online (mon massiu multijugador en línia), First Person Shooter (acció en primera persona) o Visual Novels (novel·les visuals), d'entre altres (Paul, Goon, & Bhattacharya, 2012).

Un dels primers motors gràfics sabuts és Ultima Underworld (Blue Sky Prod., 1989), que es va utilitzar per crear un joc que rep el mateix nom, "Ultima Underworld". Més endavant, i ja un dels més famosos, és Doom Engine (ID software, 1993), que tot i que no podia renderitzar gràfics 3D, tenia capacitat de representar objectes, personatges i el mapa amb l'ús d'imatges en 2D. Tant el motor i l'art usat aconseguien crear una il·lusió òptica que et trobaves en un món 3D. (Paul, Goon, & Bhattacharya, 2012)

Alguns exemples de motors gràfics populars actualment són Unreal Engine (Epic Games, 1998), Unity (Unity, 2005), Godot, Frosbite (Electronic Arts, 2008), Creation Engine (Bethesda Game Studios, 2011), etcètera.

L'ús de motors gràfics s'ha anat popularitzant al llarg dels anys a causa de les facilitats que li concedeixen al desenvolupador, com ja hem pogut llegir abans.

En un article de GameDeveloper.com (Doucet & Pecorella, 2021) s'exposa l'increment de l'ús dels motors gràfics en el sector dels videojocs utilitzant com a base de dades Steam (Valve Corp., 2003). Steam és una plataforma de distribució de videojocs digitals, on els usuaris poden tenir la seva pròpia biblioteca de videojocs, crear comunitats, interaccionar amb altres usuaris, etcètera (Valve Corp., 2003). Steam és la plataforma més gran del món de venda de videojocs digitals amb més de 50.000 videojocs publicats, i arribant al pic màxim de 28 milions d'usuaris connectats simultàniament, formant un total de més de 120 milions d'usuaris actius mensuals.

2011	2021
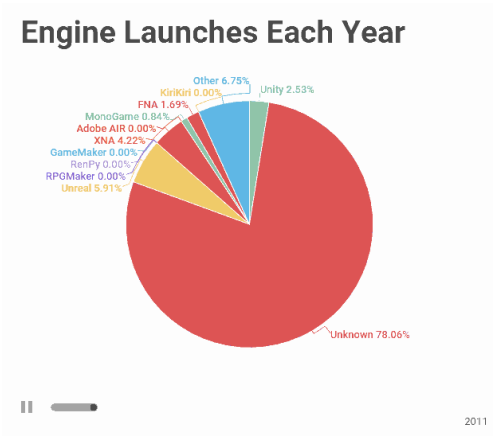	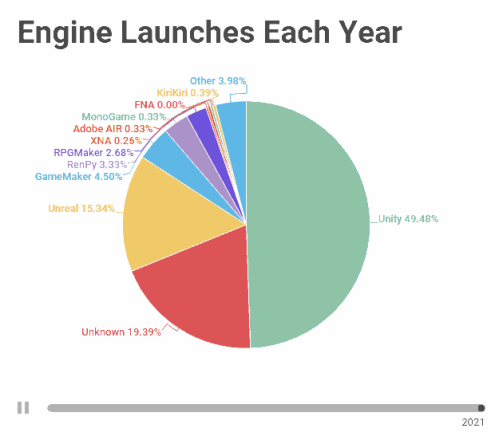
Figura 1 Publicació de jocs i motor gràfic que utilitzen cada any el 2011. Font: Doucet, L., Pecorella A., 2021	Figura 2 Publicació de jocs i motor gràfic que utilitzen cada any el 2021. Font: Doucet, L., Pecorella A., 2021
Com es pot veure a la Figura *1*, en el 2011 l'ús de motors gràfics era molt reduït. S'utilitzaven motors gràfics de molt baix nivell com XNA o FNA, i només en pocs casos s'utilitzaven motors més avançats com Unity o Unreal Engine (encara en un estat molt inicial).	A la Figura *2*, l'ús dels motors gràfics ha canviat radicalment respecte al 2011, i en el 2021, es va reportar que només tenint en compte Unity, el 50% dels videojocs publicats utilitzaven aquest motor gràfic. Un canvi del +1955%, obviant altres motors de gran pes com Unreal Engine que ocupa la 3a posició amb 15,34% (també un augment significatiu del +249%).

Taula 1 Comparativa de l'ús de motors gràfics en els jocs publicats entre l'any 2011 i l'any 2021. Font: Elaboració pròpia.

2.1.2 Llenguatge C#

Unity utilitza el llenguatge C# (Microsoft, 2000) com a llenguatge de scripting, per tant, els usuaris programen utilitzant C# (Unity, 2022).

Segons Microsoft (Microsoft, 2022) C# és un llenguatge de programació modern orientat a objectes. C# és una evolució del llenguatge C (Dennis Ritchie, 1972). C# té les següents característiques:

- Garbage collection automàtica: fa referència al procés de recollir la memòria del programa que ja no està sent emprada i alliberar-la de manera automàtica, per tal que l'usuari no ho hagi de fer.

- Funcions Lambda: són una característica que et permet crear funcions anònimes en el programa.

- Tipat fort: assegura que una variable té definit un tipus i un nom i que no es podrà canviar.

- Tipus per referència i tipus per valor, definits per l'usuari: L'usuari és qui els defineix. Si són per referència vol dir que la variable apunta a un altre espai de memòria què és la que guarda la informació. Si són per valor vol dir que la variable és la que guarda la informació.

Aquestes són algunes d'un llistat llarg de característiques que defineixen el llenguatge. C# s'executa sobre la plataforma .NET, un sistema d'execució virtual anomenat CRL i un conjunt de biblioteques de classes. CLR és una implementació feta per Microsoft que prové de CLI, un estàndard internacional (Microsoft, 2022).

El projecte de C# va començar el desembre de 1998, però no va ser fins a l'any 2002 que va ser finalment publicat (Nagel, 2018). Anders Hejlsberg va ser el dissenyador principal de C# i un desenvolupador clau en el desenvolupament de .NET (Microsoft, 2001). Va entrar a la companyia el 1996 després del desenvolupament d'altres llenguatges, i el 1999 el van ascendir al màxim nivell dins de Microsoft (Microsoft, 2001).

2.1.2.1 OOP, Object Oriented Programming

C# es fonamenta en la programació orientada a objectes, també popularment coneguda com a OOP (Microsoft, 2022).

Segons Chambers (Chambers, 2014) la programació orientada a objectes té 4 idees principals:

1. Tot el que es computa es diu que és un "objecte".
2. L'eina principal en programar són les definicions de classes, que contenen propietats. Aquestes propietats són alhora, objectes definits com una classe.
3. Una classe pot heretar d'una altra classe (anomenada superclasse), de tal manera, l'objecte d'aquesta classe, serà alhora un objecte de la superclasse.
4. Per utilitzar els objectes, es defineixen mètodes, els quals et permeten modificar els atributs dels objectes.

La programació orientada a objectes aporta beneficis, tals com fer el codi més ràpid i fàcil de mantenir i controlar la complexitat en programes grans (Hemmendinger, 2022).

Simula (Nygaard, K., Dahi, J., 1962) va ser el primer llenguatge a implementar OOP. Smaltalk (Kay, Alan, 1980), va ser molt rellevant i va influenciar els llenguatges actuals (Hemmendinger, 2022).

2.2 Eines pel desenvolupament dels videojocs

Una eina serveix per millorar el flux de treball pels programadors, artistes i dissenyadors de nivells (Unity, 2019).

En paraules de Brett Taylor (Taylor, 2020), una eina és un codi o sistema que estalvia temps a un dissenyador o programador, optimitzant el procés.

Segons Katherine Neil (Neil, 2019), les eines de disseny de jocs ajuden a solucionar problemes de disseny sense haver de construir experiències jugables per provar el valor de les teves idees. Ajuden a crear eficientment informació ordenada i pots veure fàcilment el teu progrés. Exemples d'eines per dissenyadors de jocs són: obtenció de mètriques, prototipatge ràpid, diagrames de flux, modificar elements del joc...

A més, Neil afegeix que les eines ajuden amb el sistema de disseny i balanç, disseny narratiu, disseny del progrés, disseny del nivell i missions i disseny de contingut procedimental. A continuació, exposa algunes eines com Aritcy: Draft (Arcity Software, 2014), que serveix de base de dades visuals per poder visualitzar les línies de la història, personatges, i variables de la història. A més, permet exportació directa a motors gràfics com Unity o Unreal Engine, The sentinent Sketchbook (Liapis, 2013), amb la funcionalitat de crear mapes i obtenir resultats de la jugabilitat en temps real mitjançant intel·ligència artificial...

A vegades, les eines, no tan sols serveixen per no perdre tant de temps, sinó també per mantenir el ritme i la motivació intactes durant el temps. Un exemple d'eines per a dissenyadors són eines que permetin afegir contingut al joc, i un exemple d'eines per a programadors és afegir llibreries que permetin programar de forma més ràpida i fàcil (Taylor, 2020).

Segons Armstrong (Armstrong & Ewing, 2019), invertir temps en eines, els va permetre crear un joc amb producció AAA amb un abast indie i els va ajudar a suportar els constants canvis del desenvolupament.

2.2.1 Creació d'eines

David Lightbown (2019) exposa les 3 preguntes principals que s'han d'autoplantejar a l'hora de realitzar una eina.

1. La primera pregunta que es planteja és "Quin és el problema que els usuaris volen solucionar?". Això permet identificar quins són els objectius de l'eina. No es pot donar per entès que els usuaris objectius de la teva eina sàpiguen com fer eines, però si és interessant sempre prestar atenció a les diferents característiques que et poden demanar.

2. La segona pregunta és "Què es pot aprendre d'altres eines que intenten solucionar els mateixos problemes que l'eina que es vol desenvolupar?". Això ajuda a veure el diferent ventall de possibilitats a l'hora d'afrontar els reptes que et plantegi l'eina.

3. Finalment, la tercera pregunta és "Com l'usuari utilitza les eines i com encaixen amb les altres eines que tinguis?". Això permet avançar-se als problemes que poden sorgir i el flux de treball que els usuaris faran servir per saber les diverses necessitats. A més, ajuda a definir elements comuns entre les eines que facilitin als desenvolupadors a l'hora d'adaptar-se a l'eina i treballar amb ella.

Un mètode per fer certes eines més senzilles és inserir codi de forma hardcoded per provar la funcionalitat, i un cop sabent si funciona en l'àmbit d'experiència d'usuari, ja es pot programar l'eina de manera completa. D'aquesta forma s'evita dur a terme l'eina en profunditat i perdre temps si el resultat no és satisfactori. (Lightbown, 2021)

A l'hora de crear les eines s'ha de tenir en compte, que al principi del desenvolupament, el joc no està definit al 100%, per tant, les eines i els sistemes han de ser prou flexibles per adaptar-se als canvis que succeeixen al llarg del desenvolupament (Armstrong & Ewing, 2019).

2.2.2 Quan fer una eina?

El primer pas a l'hora de desenvolupar una eina és desenvolupar les funcionalitats bàsiques per poder tenir una idea més acurada de si realment necessites l'eina i quines característiques ha de tenir. També hi ha la possibilitat d'elaborar eines a partir de detectar patrons, com repetir una acció moltes vegades o esperar càrregues (Taylor, 2020).

Elaborar una eina específica pot ser molt útil per augmentar al màxim l'eficiència, però com més específica, més ràpid es quedarà obsoleta. Tot i això, és recomanable fer eines específiques, ja que fer eines generals que puguin adaptar-se a tot, també comportarà més temps de desenvolupament (Taylor, 2020).

Unes bones preguntes a fer-se per saber si val la pena desenvolupar una eina són: com seria de molest desenvolupar la meva feina sense l'eina, quant de temps es tardaria a desenvolupar la feina, i si el resultat final realment faria estalviar temps (Taylor, 2020).

La funció principal de les eines és estalviar temps, per tant, si s'estima que acabar l'eina que s'està realitzant comportarà més temps del que s'estalviarà, val la pena parar de desenvolupar-la (Taylor, 2020). A més a més s'ha de tenir en compte que les eines tindran errors, que poden dificultar el desenvolupament segons el context del projecte i reduir l'eficiència d'aquesta, o provocar que s'allargui el seu desenvolupament (Taylor, 2020).

2.2.3 Tècniques d'implementació

Vesa Paakkanen (Paakkanen, 2021), desenvolupador de Remedy (Remedy Enterteinment, 1995) defensa que sempre s'hauria de prioritzar un disseny de components / plugins sobre un disseny monolític. El disseny monolític els hi causava sovint problemes en modificar un element de l'eina, causava problemes en altres elements. A més, afegeix que és bona idea que s'injecti el comportament de les diverses funcionalitats als diferents panells, i no que s'hagi d'obrir un nou panell per cada element que es vol modificar. Paakkanen destaca la importància de

separar la interfície, el comportament i les dades per evitar errors i dificultat afegida a l'hora de desenvolupar noves característiques.

Jeff Stewart (Stewart, 2020) explica que quan es crea una eina, no s'ha de tenir present ni l'eina ni el motor gràfic, sinó el flux de treball que l'usuari final vol tenir. A més recomana incloure certs elements bàsics que faran millor l'eina:

- Mostrar resultats abans d'hora en buscar un element en un explorador.
- Afegir dreceres amb el teclat i poder reassignar-les.
- Avisar a l'usuari que existeixen les dreceres.
- Ajudar a l'usuari amb icones.
- Intentar que no només existeixin les eines, sinó que també siguin fàcils d'usar.
- Intentar només donar a l'usuari la informació necessària, evitant la sobreexposició a opcions i elements que no són necessaris per a la tasca actual.

2.2.4 Implementació d'eines a Unity

Unity és conscient que els usuaris utilitzen el motor per fer les seves pròpies eines que serveixin a altres usuaris o en el seu propi desenvolupament. És per això que Unity concedeix eines a l'usuari per crear-les. Des del desenvolupament de les mateixes amb llibreries per fer interfície d'usuari, com poden ser UI Toolkit o IMGUI, el backend, amb els ScriptableObjects o classes com EditorGUIUtility, fins a la publicació de les mateixes a la seva pròpia botiga d'assets, sota l'apartat de "Tools" (Unity, 2019).

Els desenvolupadors de Firewatch (Campo Santo, 2015), joc creat amb Unity, anomenen els avantatges de fer eines dins d'un motor gràfic (Armstrong & Ewing, 2019):

- Són barates de fer, ja que no has de perdre temps en tasques com obtenció d'inputs, dibuixar la interfície...
- És molt flexible, perquè no has de crear capes que separin el motor de l'eina, fàcil d'afegir contingut.
- Funciona per defecte en temps d'execució.

Els desavantatges, segons Ewing són (Armstrong & Ewing, 2019):

- Són desagradables a la vista per treballar còmodament.
- Són complexes d'utilitzar.
- Canvien constantment segons les necessitats del desenvolupament.

En paraules de la mateixa documentació d'Unity, "Unity et permet ampliar l'editor amb les teves pròpies finestres i inspectors i pots definir com les propietats es mostren a l'inspector amb "custom property drawers"" (Unity, 2021).

2.2.4.1 Inspector

L'inspector et permet veure i editar propietats i opcions dels diferents elements d'Unity, com poden ser els GameObjects (objectes dins del món del joc), diferents assets com textures, materials o models 3D, components, i la mateixa configuració del motor, d'entre altres. (Unity, 2021)

Unity per defecte et genera un inspector pels diferents components i scripts de l'usuari, però et permet crear inspectors personalitzats (Unity, 2021).

Crear inspectors personalitzats pot atorgar avantatges tals com:

- Donar una visió de les teves propietats més amigable a l'usuari.
- Organitzar i agrupar propietats.

- Ensenyar o amagar propietats i elements segons les opcions escollides per l'usuari.

- Donar informació concreta per cada propietat.

Crear inspectors personalitzats és més fàcil amb la tecnologia UI Toolkit comparada amb IMGUI (ambdues tecnologies cobertes al 3.2.5.4), ja que UI Toolkit té avantatges com vinculació de la informació automàtica, o la possibilitat de desfer una acció. (Unity, 2021)

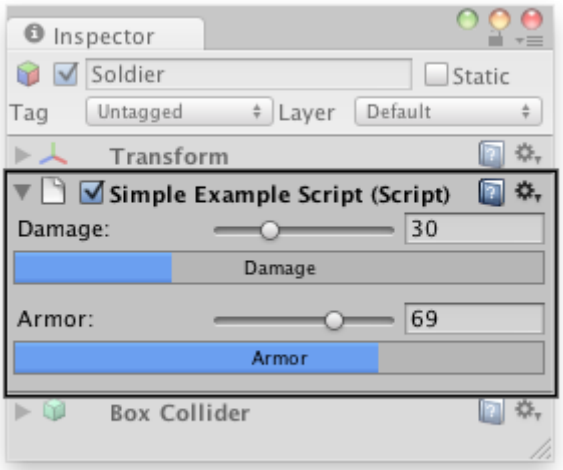

Figura 3 Exemple d'inspector personalitzat. Font: Unity

2.2.4.2 Finestres d'editor

Es poden crear tantes finestres personalitzades com es vulgui i funcionen com qualsevol finestra d'Unity. Ajuden a donar interfícies d'usuari a sistemes del videojoc.

Figura 4 Exemple de finestra personalitzada. Font: Unity

2.2.4.3 Property drawers

Els "property drawers" permeten personalitzar mitjançant atributs des dels scripts l'aparença a l'inspector.

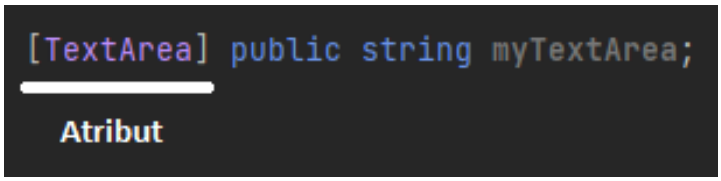

Figura 5 Exemple d'ús d'un atribut al codi. Font: Elaboració pròpia

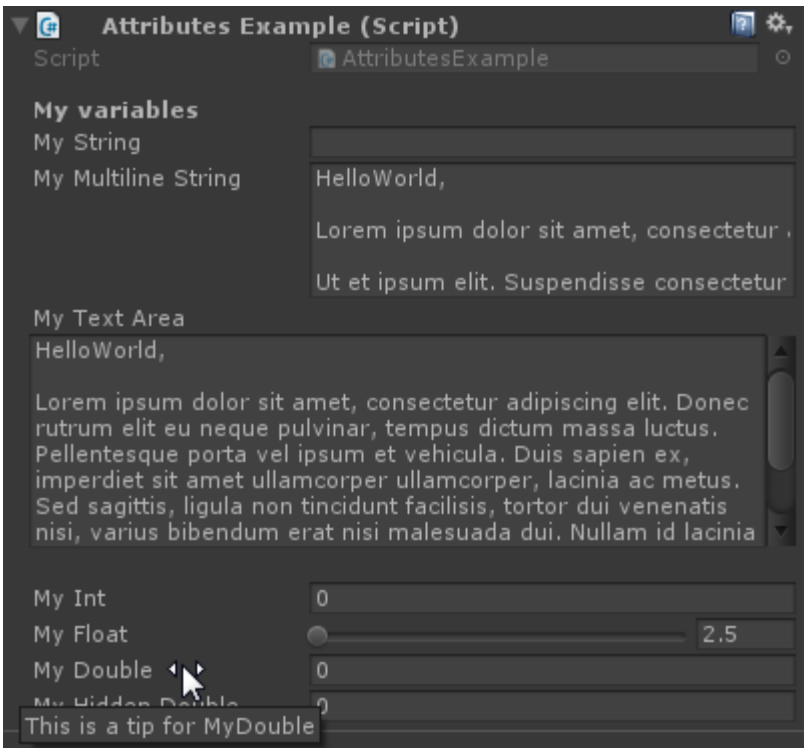

Figura 6 Exemple d'ús dels property drawer. Font: Elaboració pròpia.

A la Figura 6, podem veure com una variable de tipus string (cadena de text) per defecte només ocupa una línia. Gràcies a afegir-li l'atribut de tipus property drawer, [TextArea], es crea un espai més gran on escriure, i que es pot redimensionar.

2.2.4.4 IMGUI i UIToolkit

IMGUI és una biblioteca d'interfície gràfica de codi obert escrita en C++. El mateix creador ho defineix com un "sistema ràpid, portable, independent del sistema de render i que no té dependències de tercers" (Cornut, 2014).

Especialment, IMGUI està pensat per integracions en motors gràfics, aplicacions 3D en temps real i aplicacions en pantalla completa o incrustades (Cornut, 2014). És per això que Unity va escollir inicialment utilitzar IMGUI per tota la seva interfície del motor gràfic, incloent-hi la renderització de la interfície en temps real dins del joc. Poc després Unity va crear Unity UI. Unity UI està pensat per la renderització en temps real dins del joc, però no pot ser usada com a interfície visual del motor (Unity, 2021).

Per acabar, durant l'any 2019, Unity va anunciar UI Builder, actualment conegut com UI Toolkit. UI Toolkit ve a unificar tant la interfície de l'editor, com la interfície en temps real dins del joc (Unity, 2019), i a la vegada permetre que no sol siguin els programadors els que puguin utilitzar la tecnologia, sinó que també rols com els dissenyadors d'interfície o artistes tècnics puguin utilitzar-ho i conviure-hi. (Unity, 2021)

Type of user	UI Toolkit	Unity UI (uGUI)	IMGUI
Programmer	✓	✓	✓
Technical Artist	Partial	✓	✗
UI Designer	✓	Partial	✗

Figura 7 Taula de possibilitat d'ús segons tipus de desenvolupador. Font: Unity, 2022

Centrant-nos en la part de fer editors personalitzats, UI Toolkit permet iteracions més ràpides gràcies a la seva construcció visual (en lloc de per codi) i el sistema d'estils (similar a la programació web) que no té IMGUI. A més a més, UI Toolkit està amb actiu desenvolupament, comparat amb IMGUI, i té promeses d'incloure elements que no estan a IMGUI com Grid Views, Graph view, Data bindings, nested prefabs, taules bidimensionals, integració amb el nou sistema d'input, esdeveniments serialitzats, renderitzat de shaders, animacions i debugging, d'entre altres (Unity, 2021).

Per tant, UI Toolkit està pensat per substituir tots els sistemes de UI d'Unity, i de fet, la major part de components del motor gràfic ja utilitzen UI Toolkit per defecte (Unity, 2021).

2.2.4.5 Unity Style Sheets (USS)

Per estilitzar UI Toolkit, s'utilitza Unity Style Sheet (també conegut com a USS), un concepte semblant al CSS (CSS Working Group, 1996) de web. Amb aquest sistema, pots crear aspectes predefinits pels diferents elements o classes, i que aquests aspectes predefinits opcionalment funcionin respecte a unes variables definides o bé nombres concrets (Unity, 2022).

Les variables són molt útils per definir, per exemple, paletes de colors que la teva interfície farà servir, i si en algun moment modifiques algun dels colors tots els elements, se'n veuen reflectits (Unity, 2022).

L'únic desavantatge de les variables a l'USS és que només funcionen per codi, i ara per ara, no es poden visualitzar a l'editor d'Unity de UI Toolkit (Unity, 2022).

2.2.4.6 Binding i esdeveniments

Per connectar la interfície de UI Toolkit amb les dades Unity ofereix principalment 2 opcions: El binding i el sistema d'esdeveniments.

El binding és un sistema d'Unity per vincular el valor d'una variable a propietats de la interfície d'usuari. Aquest sistema substitueix al sistema d'esdeveniments. El binding només funciona a l'editor. (Unity, SerializedObject data binding, 2022)

Els avantatges del binding són:

- Suporta el fer/desfer.
- Implementació ràpida: des de l'editor podem definir a quina variable afecta l'element visual. Després, només hem de vincular la finestra de l'editor a un serializableObject i automàticament totes les variables es vinculen als elements visuals.

Els desavantatges del binding són:

- Només funciona per variables que siguin serialitzables, com valors primitius del C# (int, float, bool), classes scriptable object, d'entre altres.
- Els serialized objects són difícils de treballar, per com funciona el sistema de serialització a Unity.
- El binding i, per tant, la serialització basa gran part del sistema en el nom que tingui la variable de forma hardcoded. Això causa que si el nom de la

variable canvia, es pot trencar el funcionament sense que el desenvolupador se n'adoni.

- No permet tanta personalització com el sistema d'esdeveniments.

El sistema d'esdeveniments de UI Toolkit és molt similar al sistema d'HTML. Quan un esdeveniment succeeix, s'envia a través dels elements fins a arribar a l'arrel.

Per qualsevol element visual de la finestra de l'editor, pots subscriure un o més esdeveniments diferents, com per exemple el clic del ratolí, canviar de mida de l'element o que es polsi alguna tecla.

L'esdeveniment que ens interessa principalment, però, és el de "ValueChanged". Aquest esdeveniment és el que s'usa per registrar el canvi del valor a la interfície. Això serveix directament per vincular el valor de la interfície a la variable.

Els avantatges del sistema d'esdeveniments són:

- Alta personalització: quan el valor canvia, es pot decidir si actualitzar-lo o no, o afegir comportaments que no actuarien normalment.
- El sistema no utilitza codi "hardcoded", per tant, no hi ha problema si en un futur es canvia el codi, noms de variables, etcètera, perquè el programador i l'usuari podran veure amb facilitat l'error.

Els desavantatges del sistema d'esdeveniments són:

- Per cada camp a la interfície has de registrar l'esdeveniment, el qual causa que hagis d'anar manualment un per un assignant els esdeveniments i valors.
- No suporta el sistema de fer/desfer.

2.2.4.7 Unity Asset Store

Unity Asset Store és la botiga d'Unity que permet comercialitzar assets (models 3D, textures, animacions, eines...) (Unity, 2022).

La botiga té un apartat específic per comercialitzar eines, actualment amb més de 9000 eines publicades i representen el 13% dels assets disponibles (Unity, 2022).

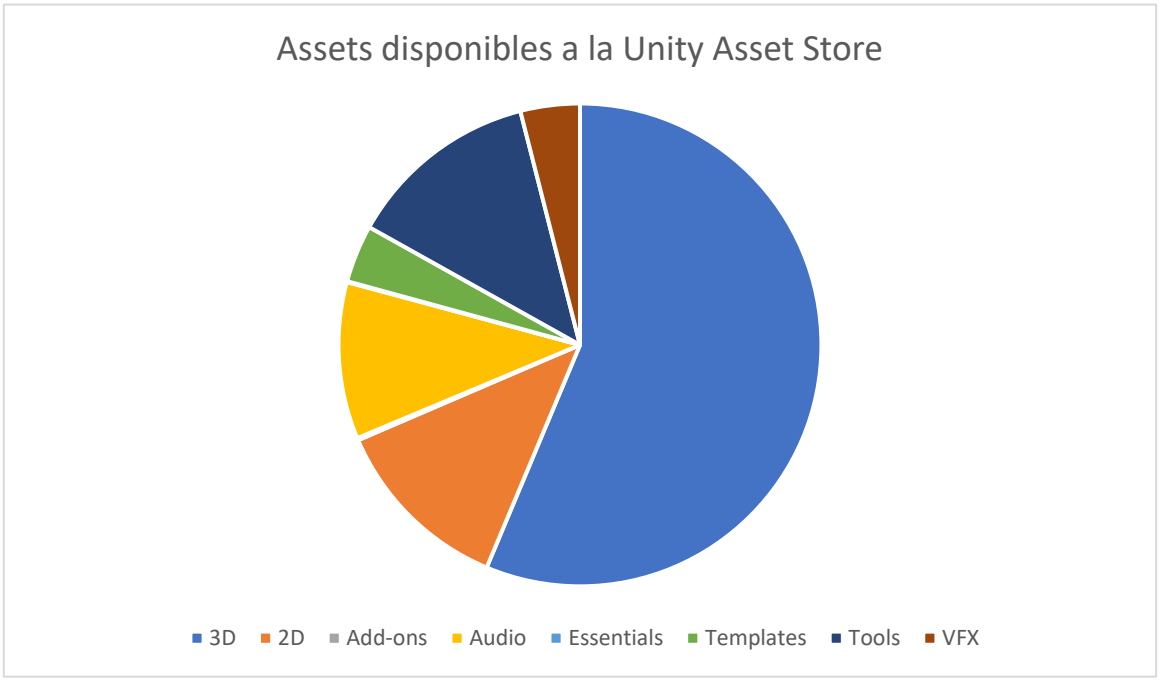

Figura 8 Gràfic d'assets publicats a la Unity Asset Store. Font: Elaboració pròpia. (Dades de assetstore.unity.com)

L'Unity Asset Store ha tingut una mitja de 4 milions de visites cada mes durant els mesos gener, febrer i març del 2022 (SimilarWeb, 2022).

2.3 Videojocs RPG

2.3.1 Que són els videojocs RPG?

Les sigles RPG provenen de "Role-Playing Game", o en català, "joc de rol". En un principi, es podria pensar que tots els videojocs obtenen la categoria RPG, ja que, el jugador al jugar un videojoc està executant un rol, però el gènere RPG s'utilitza per definir unes característiques més concretes (Kratochvíl, 2014).

Un videojoc de rol és un gènere de videojocs digitals on el jugador avança una missió principal, i opcionalment, missions secundàries, amb les quals els personatges o grup de personatges que controla el jugador, guanyen experiència, la qual permet millorar atributs i habilitats de cada personatge. Els videojocs RPG tenen la seva base en el clàssic "Dungeons and Dragons" (TSR, Inc., 1974) (Hosch, 2019).

2.3.2 Història dels jocs RPG

El primer videojoc RPG és "Dungeon" (Digital Equipment Corporation, 1975), una adaptació de Dungeons and Dragons que mancava de llicència per part de TSR (Hosch, 2019). A la revista Computer Gaming World, Daglow, el creador del joc Dungeon, va dir que "a mitjans dels setanta, tenia un videojoc RPG plenament funcional amb combat a distància i cos a cos, línia de visió, mapa aleatori i una IA discreta" (Daglow, 1988).

Cal destacar que els primers jocs RPG mantenien estructures molt similars a Dungeons and Dragons, amb mons de fantasia fonamentats en goblins, dracs, elfs, trols, nans... El primer joc comercial RPG va ser Ultima (Origin Systems,1980), seguit de Wizardy (Sir-Tech Software, 1981). Els dos mantenien les bases de Dungeons and Dragons, i van ser publicats inicialment per la plataforma d'ordinador Apple 2 (Apple, 1977). Es van publicar diverses seqüeles de Wizardy i Ultima durant les 2 següents dècades. (Hosch, 2019)

El 1986 Square Enix va publicar els populars Dragon Quest (Square Enix, 1986) i només un any després, el 1987, Final Fantasy (Square Enix, 1987). El 1995 Nintendo va revolucionar el mercat amb Pokemon (Nintendo, 1995), la saga de videojocs més exitosa tenint en compte les vendes i incloent marxandatge. (Hosch, 2019)

2.3.3 Aspectes i característiques generals dels jocs RPG

Segons Adams (Adams, 2014), un joc de rol segueix les següents característiques:

- Història i ambientació: El món sovint està basat en la fantasia o ciència-ficció, el qual ajuda al fet que els jugadors puguin fer coses que no poden a la vida real.

- Exploració i missions: L'exploració d'un món és part important dels RPG. Sovint, la història utilitza l'exploració com a eina per avançar en les missions. Normalment, els jugadors han de complir una consecució de missions per arribar al final del joc i de la història. Molts jocs sovint també permeten allunyar-te de la rama de missions principals per fer-ne de caràcter secundari a canvi de recompenses.

- Ítems i inventari: Els jugadors tenen un inventari on s'acumulen els diferents objectes que van obtenint en recórrer el món, tals com armes, armadures, roba, o qualsevol mena de botí. A vegades, s'implementa a l'inventari un sistema de límit per pes o per espais, per afegir dificultat.

- Experiència i nivells: Els jocs de rol usen l'experiència i els nivells com a símbol de progressió. En aconseguir certs punts d'experiència puges de nivell. Normalment, s'assoleix experiència completant missions i derrotant enemics. Pujar nivells et facilita desbloquejar noves habilitats, armes, màgies, etcètera, o bé augmentar els teus atributs, els quals et permeten enfrontar-te a reptes més difícils, que et possibilitaran augmentar més el nivell. Aquest bucle és fonamental pels RPG.

- Habilitats i accions de personatge: Les accions que faci el personatge dins del joc seran un èxit o un error segons els atributs del personatge. Habitualment es permet la creació i personalització del personatge a l'inici del joc, on es pot decidir l'estètica visual a més a més dels atributs inicials.

- Combat: El combat és una part essencial dels RPG, a vegades per torns, o a vegades en temps real, el personatge o grup de personatges combatran amb diferents enemics per avançar a través de les missions i del món.

A més Frans Mayra (Mäyrä, 2017) assenyala que els diàlegs són un aspecte essencial en els RPG, ja que són el suport que situa al jugador en la història i el món, una part important del gènere.

3. Referents

En aquest apartat es vol exposar els diversos referents i precursors en els quals es basa aquest projecte, primer repassant algunes eines enfocades al gènere RPG, al motor gràfic d'Unity, seguit d'altres eines en altres motors gràfics, finalitzant amb un exemple d'eina d'una empresa privada.

3.1 RPG Builder (Unity)

RPG Builder és una eina per fer videojocs de gènere RPG a Unity sense programar, desenvolupada per Blink (Blink, 2022). RPG Builder té un cost de 156,33€ (04/2022) a l'Asset Store d'Unity, i s'imposa vers les altres eines similars amb 1521 favorits d'usuaris, i amb 205 puntuacions amb mitjana de 5 estrelles de 5.

L'eina està dividida en 31 mòduls (alguns independents), i inclou diverses animacions de personatges, assets, centenars de tutorials i un controlador per al jugador.

Alguns d'aquests mòduls són: habilitats, NPC, característiques, ítems, races, classes, encanteris, arbres de talent, missions, diàlegs, tasques, etcètera.

L'eina, principalment, té una única finestra com a editor, on es modifiquen la majoria de funcionalitats i mòduls que té.

Un complement que li dona un valor afegit és la inclusió d'animacions i assets que facilitin el prototipatge o acabat del joc, mitjançant models 3D, textures, etcètera. Això ha estat gràcies al fet que el desenvolupador ha establert un tracte amb diferents desenvolupadors d'assets de l'Asset Store, per incloure'ls a la seva eina per defecte. Alguns d'aquests desenvolupadors són: Polytope (Polytope Studio, 2018), Malbers (Malbers Animation, 2015), Gabriel Aguiar (Gabriel Aguiar Prod, 2017) i Poneti (Poneti, 2018), d'entre altres.

L'eina no compta amb un sistema d'inventari, però a continuació s'analitza el sistema d'ítems i el sistema de diàlegs, ja que són els mòduls objectius en els quals s'enfoca la part pràctica del projecte:

3.1.1 Mòdul d'Ítems

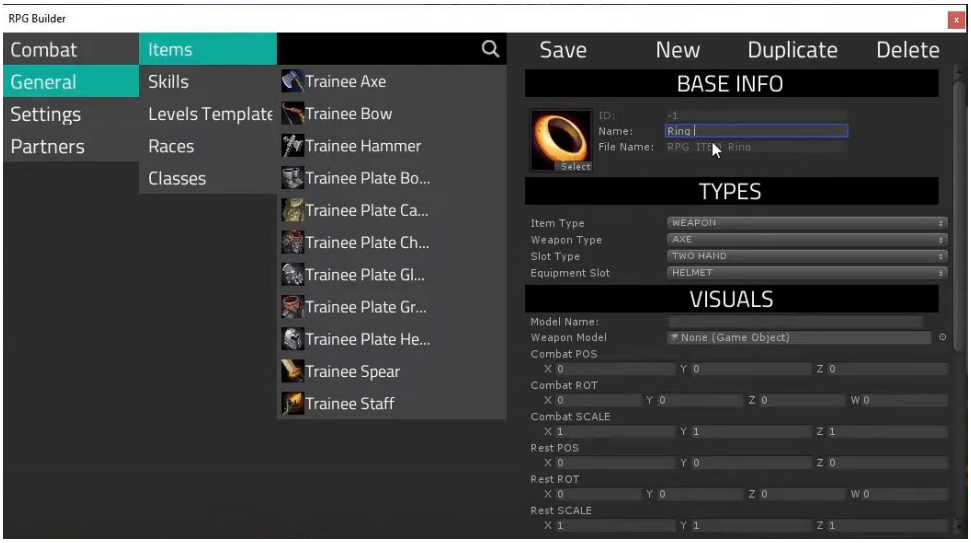

Figura 9 Mòdul d'ítems de RPG Builder. Font: (Blink, 2022)

Com es pot veure a la Figura 9, l'editor consisteix en 2 parts, la part esquerra, que compta amb un menú i submenú d'opcions per triar quin dels mòduls vols modificar, i un cop seleccionat el mòdul d'ítems dins el submenú de general, a la part dreta, apareix el mòdul d'ítems, mostrant-ne un llistat i un inspector de dades per modificar l'ítem en qüestió. La llista de personalització de l'ítem conté moltes propietats:

- Generals: Nom, nom que s'ensenya al joc, descripció.
- Preu: tipus de moneda i preu.
- Inventari: quantitat que pots tenir d'aquest ítem a l'inventari.

- Tipus: Qualitat de l'ítem (Es poden definir diferents qualitats en un altre menú), tipus d'ítem (Per exemple arma, armadura, menjar…), tipus d'arma (Es poden definir diferents tipus d'arma en un altre menú), la posició que ocupa l'ítem a l'inventari del personatge i el tipus d'armadura (Es poden definir diferents tipus d'armadura en un altre menú).

- Encantaments: encantament que té l'ítem i si aquest es consumeix en utilitzar-lo.

- Botí: opció per fer aparèixer físicament l'ítem, model 3D i duració dins del món.

- Estats: estat que altera (per exemple bonificació del mal, bonificació d'armadura, etcètera), quantitat d'alteració i si la quantitat d'alteració és en format de percentatge.

- Estats aleatoris: El mateix que en l'apartat anterior, però amb possibilitat d'aleatorització.

- Requisits per utilitzar l'objecte: classe, nivell, habilitats desbloquejades, posseir un altre ítem, ser d'una raça, estat d'una missió i haver matat un NPC en concret.

Totes aquestes característiques de personalització permeten una alta personalització amb facilitat.

Ja que la intenció de l'eina és que l'usuari no hagi de programar, es troba el primer desavantatge al no tenir accés clar al codi de la base de dades dels ítems (tampoc hi ha documentació que et pugui ajudar), amb el qual augmenta de forma pronunciada la dificultat de fer modificacions al software i personalitzar-lo amb característiques i necessitats concretes del projecte.

3.1.2 Mòdul de diàlegs:

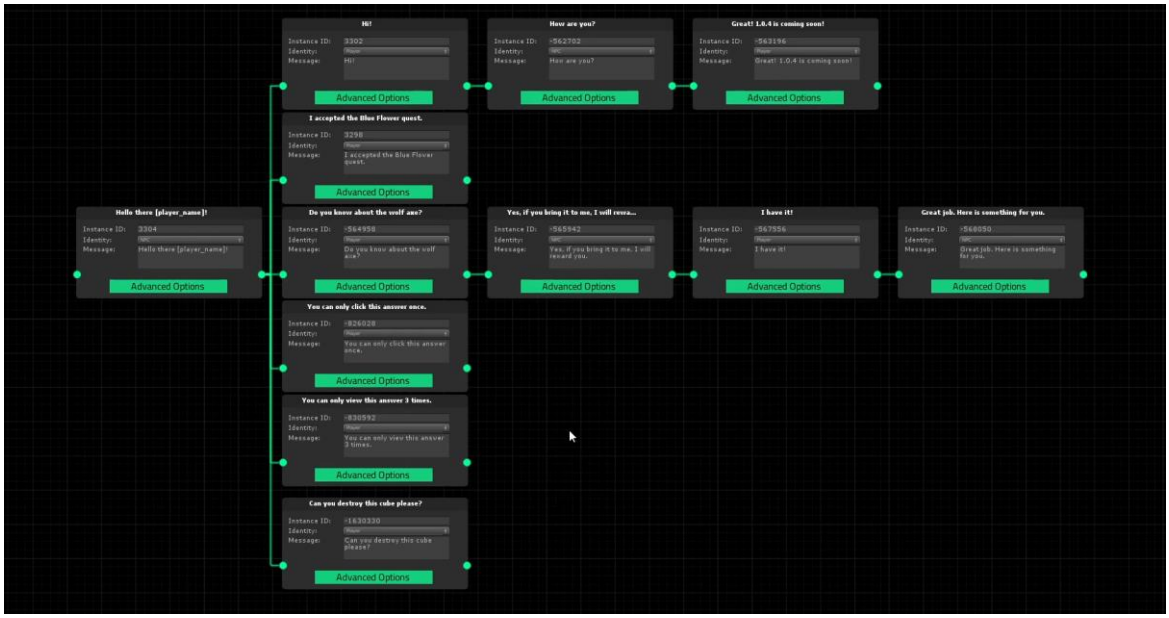

Figura 10 Mòdul de diàlegs de RPG Builder. Font: (Blink, 2022)

El mòdul de diàlegs de RPG Builder es basa en un sistema de nodes on visualment uneixes els diversos blocs per decidir la lògica dels diàlegs.

A més a més de decidir quin personatge parla i les línies de diàleg, hi ha un panell d'opcions extra que permet una major personalització amb opcions de tenir:

- Requeriments: necessitats per desbloquejar aquesta línia de diàleg. Poden ser tenir un ítem en concret a l'inventari, ser d'una classe en concret, i moltes opcions més.

- Accions: accions que s'activen en el món, personatge, etcètera, com per exemple pujar de nivell al jugador, donar-li un ítem, guanyar monedes o fer que apareguin enemics, d'entre altres.

- Opcions: afegir una imatge com a icona del personatge que manté el diàleg, fer el diàleg amb temps límit, etcètera.

Aquest diagrama de nodes es guarda com un arxiu, que s'utilitza des d'un altre menú per assignar-li una ID amb la qual obtenir referències, descripció i la possibilitat d'afegir un node final per concloure el diàleg.

Per construir el diàleg es crea primer un node que no tindrà cap connexió a l'entrada, i a partir d'aquí es connecten el resta de nodes. Si un node té més d'un node connectat a la sortida, el programa detecta que són opcions de diàleg. Si el node només té un node connectat de sortida, el programa detecta que és un diàleg normal.

A més a més, RPG Builder permet crear unes claus globals, que després pots col·locar dins els diàlegs. Aquestes claus globals les pots actualitzar amb dades i el diàleg les introduirà.

Per altra banda, el sistema permet especificar un node de sortida que sempre estarà actiu quan se li mostrin opcions de diàleg al jugador.

Figura 11 Exemple de diàleg generat a RPG Builder. Font: (Blink, 2022)

3.2 RPG Editor: ORK Framework (Unity)

ORK Framework és una eina per fer videojocs de gènere RPG a Unity, desenvolupada per Gaming is Love (Gaming is Love, 2018). ORK Framework té actualment un cost de 89,33€ (04/2022) a l'Asset Store d'Unity.

L'eina promet poder crear un joc RPG complet sense haver de programar una sola línia de codi.

ORK Framework inclou tota mena d'aspectes necessaris d'un videojoc RPG: sistema de combat, sistema d'esdeveniments, habilitats, inventari, equipament, sistema de missions, faccions, guardat del joc, bestiari, localització, etcètera.

ORK Framework, a diferència de RPG Builder, sí que permet més accés al codi de l'eina, cosa que permet personalitzar més en profunditat el funcionament de totes les característiques que inclou.

Una virtut d'ORK Framework és la gran i profunda documentació que es pot trobar a la seva pàgina web.

3.2.1 Mòdul de sistema d'inventari

El sistema d'inventari permet una gran personalització. Algunes de les característiques són: definir inventaris comuns o individuals per grups de jugadors, apilar els ítems de forma automàtica, deixar anar ítems, diverses pestanyes per elements de diversos tipus, d'entre altres.

El llistat de propietats de l'ítem té un nombre molt elevat, si es compara amb RPG Builder. Compta amb gairebé 100 propietats amb les quals personalitzar els ítems.

AssetSelection< ItemTypeAsset >	**itemType** = new AssetSelection<ItemTypeAsset>()
bool	**ownDamageType** = false
AssetSelection< DamageTypeAsset >	**damageType**
bool	**hidden** = false
bool	**dropable** = false
bool	**stealable** = false
float	**sympathyChange** = 0
AssetSource< MakinomSchematicAsset >	**initSchematicAsset** = new AssetSource<MakinomSchematicAsset>()
bool	**overrideInventoryAddType** = false
InventoryAddType	**inventoryAddType** = InventoryAddType.Add
bool	**ownInventorySpace** = false
InventorySpace	**inventorySpace**
bool	**ownQuantityLimit** = false
InventoryQuantityLimit	**quantityLimit**
bool	**ownStackLimit** = false
InventoryStackLimit	**stackLimit**
MenuTargetSelection	**menuTargetSelection** = new MenuTargetSelection()
PriceSettings	**price** = new PriceSettings()
bool	**ownPrefab** = false
ItemPrefabSettings	**prefabSettings**
KeySchematicSetting []	**customSchematic** = new KeySchematicSetting[0]
bool	**ownShortcutUI** = false
UIShortcutSettings	**shortcutUI**
TypePrefabViewPortrait []	**portrait** = new TypePrefabViewPortrait[0]
bool	**ownNumberFormat** = false
AdvancedNumberFormat	**quantityFormat**
bool	**ownBattleInfoText** = false
BattleInfo	**battleInfoText**
bool	**ownNotifications** = false
InventoryNotification	**addedNotification**
InventoryNotification	**removedNotification**
bool	**ownConsoleAddAction** = false
ConsoleTextActionPreview	**consoleAddAction**
bool	**ownConsoleAction** = false
ConsoleTextActionPreview	**consoleAction**
bool	**ownConsoleCast** = false
ConsoleTextActionPreview	**consoleCast**
bool	**ownConsoleCastCancel** = false

Figura 12 Algunes de les propietats que es poden trobar dins d'un ítem. Font: (Gaming is Love e.U., 2022)

A més a més, ORK Framework inclou un sistema complet d'inventari, amb opcions de (Gaming is Love e.U., 2022):

- Compartir inventari amb altres personatges del grup o inventari individual.
- Com els ítems s'agrupen a l'inventari.
- Quan espai ocupen i límits d'espai.
- Limitar certa quantitat d'ítems a l'inventari.
- Desbloquejar receptes o nou contingut en obtenir un ítem.

- Funciones per tirar ítems a terra.
- Notificacions de l'estat o d'esdeveniments dels mateixos ítems.
- Funcionalitats per definir l'inventari del jugador.

Per altra banda, l'eina disposa d'utilitats per definir el comportament dels ítems tirats a terra, com es mostren aquests a la pantalla, i com es mostren els menús d'inventari, definir aspectes especials per certs grups d'ítems, com ara ús de partícules, augments d'altres habilitats, receptes per fabricar els ítems, etcètera.

Per tant, és un sistema molt complet que acumula tota mena de característiques que permeten un desenvolupament més ràpid.

3.2.2 Mòdul de diàlegs

ORK Framework no compta amb un sistema de diàlegs incorporats, un aspecte negatiu tenint en compte que els diàlegs són una part essencial de qualsevol joc RPG (Mäyrä, 2017).

3.3 Rpg Core (Unreal Engine)

Rpg Core s'anuncia com una eina per construir amb facilitat les característiques bàsiques dels videojocs del gènere RPG a Unreal Engine 5. Desenvolupat per StoryAge, Rpg Core té actualment un cost de 88,06€ (06/2022) a la botiga "Unreal Engine Marketplace" d'Unreal Engine (StoryAge, 2022).

Les seves característiques principals són (StoryAge, 2022):

- Sistema de vista topdown i controlador de personatges del gènere RTS.
- Sistema de combat cos a cos, a distància, atacs en àrea, habilitats, etcètera.
- Sistema d'habilitats i atributs com vida o mana.
- Utilitats com temps de recuperació d'habilitats i esdeveniments.
- Sistema bàsic d'inventari i ítems.
- Alguns exemples d'efectes visuals.

Com es pot visualitzar, l'eina s'enfoca molt en el combat i no tant amb altres aspectes del gènere RPG.

3.4 Eines utilitzades a Firewatch (Camposanto, 2016)

Camposanto va construir diferents eines per desenvolupar el videojoc Firewatch (Armstrong & Ewing, 2019), fet amb el motor gràfic Unity.

La primera eina que van construir inicialment era un sistema d'esdeveniments genèric, amb el qual en qualsevol moment podien actualitzar l'estat del joc i obtenir-ne l'estat (Armstrong & Ewing, 2019).

Per altra banda, van utilitzar l'eina anomenada PlayMaker (Armstrong & Ewing, 2019). Playmaker és una eina de programació visual enfocada a usuaris que no saben programar, animant als usuaris a crear més ràpida i eficientment. Ofereix una estructura d'estats, accions i esdeveniments per construir comportaments ràpidament (Hutong Games LLC, 2011).

Per Camposanto, disposar d'aquestes eines ajudava al fet que qualsevol dissenyador o creador de contingut del joc pogués crear contingut amb facilitat sense haver de donar feina als programadors (Armstrong & Ewing, 2019).

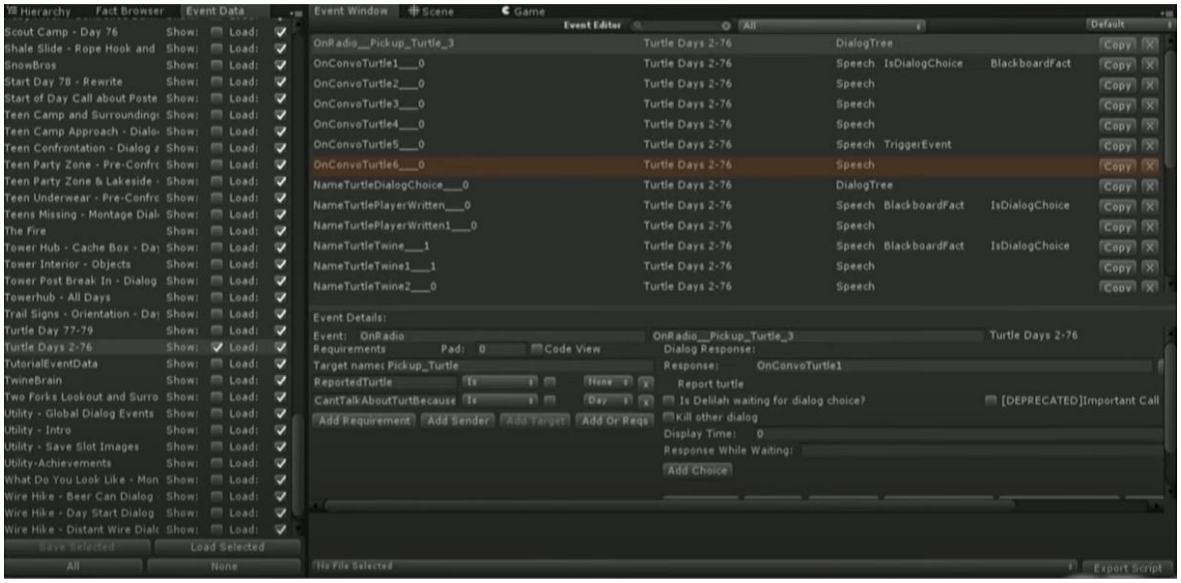

Figura 13 Captura de l'eina d'editor d'esdeveniments construït amb IMGUI. Font: (Armstrong & Ewing, 2019)

El sistema de diàlegs utilitzat al joc, estava basat en la seva eina d'esdeveniments. Quan finalitzava una línia de diàleg, es cridava la següent línia de diàleg amb un esdeveniment, que s'encarregava d'executar el clip d'àudio i mostrar el text en pantalla (Armstrong & Ewing, 2019).

Camposanto va crear una eina per gestionar els diàlegs de forma externa a Unity basada en la web anomenada Magpie. Els objectius de l'eina eren els següents (Armstrong & Ewing, 2019):

- Gestionar les línies de progrés.
- Conservar versions anteriors.
- Permetre als escriptors utilitzar les seves eines de text preferides.
- Integració a Unity fàcilment.
- Permetre la localització.

A més, en ser una eina web aportava l'avantatge de poder accedir des de qualsevol dispositiu i part del món (Armstrong & Ewing, 2019).

3.5 Elden Ring

Per a l'exemple de videojoc RPG s'ha agafat el millor valorat per la crítica i el públic en el rànquing dels millors videojocs de tots els temps categoritzats com a jocs RPG (Metacritic, 2022).

Elden Ring (From Software, 2022) es defineix com un joc de rol d'ambientació fantàstica. En el joc, el jugador ha de descobrir els misteris del poder del Cercle d'Elden. El jugador s'enfrontarà a criatures terribles i adversaris, i coneixerà personatges amb les seves pròpies motivacions per ajudar-te o complicar-te el camí (Bandai Namco, 2022).

Al joc es pot veure com s'utilitzen ítems, amb diferents propietats, icones, descripcions, etcètera...

Figura 14 Mostra dels ítems al videojoc Elden Ring. Font: Polygon, 2022

I també l'ús de diàlegs al llarg del joc:

Figura 15 Mostra d'ús de diàlegs al videojoc Elden Ring. Font: GameRant, 2022

4. Objectius

4.1 Objectius principals

- Desenvolupar una eina per al motor de videojocs Unity que permeti el desenvolupament de videojocs del gènere RPG de forma més fàcil i eficaç.
- Afegir el mòdul “Sistema d’inventari” a l’eina que permeti definir i controlar els diferents ítems del joc, i l’inventari del jugador.
- Afegir el mòdul “Sistema de diàlegs” a l’eina que permeti definir i executar diàlegs entre personatges.

4.2 Objectius secundaris

- Permetre a l’usuari final de l’eina tenir una fàcil modificació i inserció d’elements.
- L’eina ha de ser còmoda i fàcil d’utilitzar, tant per usuaris avançats com per nous usuaris.
- Publicar l’eina a l’Unity Asset Store.

5. Disseny metodològic i cronograma

Per a la planificació del projecte s'ha utilitzat l'eina de Trello, usada freqüentment per planificar projectes. S'han definit 2 taulers de tasques. El primer per a la mateixa memòria, i el segon, per a l'eina d'Unity a desenvolupar. En cada un d'ells s'han definit 4 columnes:

- Backlog: Llista de possibles tasques i característiques a implementar (si existeix el temps necessari, però no són 100% necessàries).
- ToDo: Llista de tasques i característiques que han d'estar implementades obligatòriament per poder fer funcionar l'eina i tenir acabada la memòria.
- Doing: Llista de tasques que s'estan treballant en cada moment.
- Done: Llista de tasques que ja s'han realitzat, i estan finalitzades.

Totes aquestes característiques i tasques tenen un nom i una descripció que determinen quina és la feina a fer. A més a més, tenen la data límit indicant quan han d'estar finalitzades.

El treball aplicat pràctic consisteix en la programació i disseny d'una eina per fer videojocs RPG, amb el següent software:

- Unity: Motor gràfic base per al qual es programarà l'eina.
- C#: Llenguatge de programació necessari per programar a Unity.
- Rider: Software que serveix d'entorn de desenvolupament per programar.
- UI Toolkit: Tecnologia base que utilitza Unity per dibuixar interfícies a l'editor i al joc.
- IMGUI: Tecnologia que implementa Unity per dibuixar interfícies a l'editor.
- Adobe Illustrator: Software de disseny artístic per realitzar divers art per l'eina.

- Trello: Software per a la planificació del projecte.
- Github: Software de control de versions.

Per al desenvolupament del projecte s'utilitzarà el software de control de versions Git amb Github. D'aquesta manera, es poden tenir diverses branques per cada element que vas desenvolupant, de tal manera que sempre es tingui una versió estable i segura del projecte. A més a més, evita pèrdues en cas d'avaria de l'ordinador on s'està duent a terme el projecte.

S'han definit les següents branques de desenvolupament:

- Release: només compta amb les versions finals i estables de l'eina: RPGCreator.
- Prerelease: és la que s'actualitza i uneix les diferents branques "Stable" dels diferents mòduls cada cop que una d'aquestes s'actualitza.
- Stable_DialogueModule: un cop s'ha generat una versió estable del mòdul de diàlegs, es guarda aquí.
- Development_DialogueModule s'empra per desenvolupar el mòdul de diàlegs activament.
- Stable_ItemModule: un cop s'ha generat una versió estable del mòdul d'ítems, es guarda aquí.
- Development_ItemModule: s'utilitza per treballar en el mòdul de l'eina d'ítems activament.

Amb aquest sistema de branques, s'eviten pèrdues i s'assegura un desenvolupament segur.

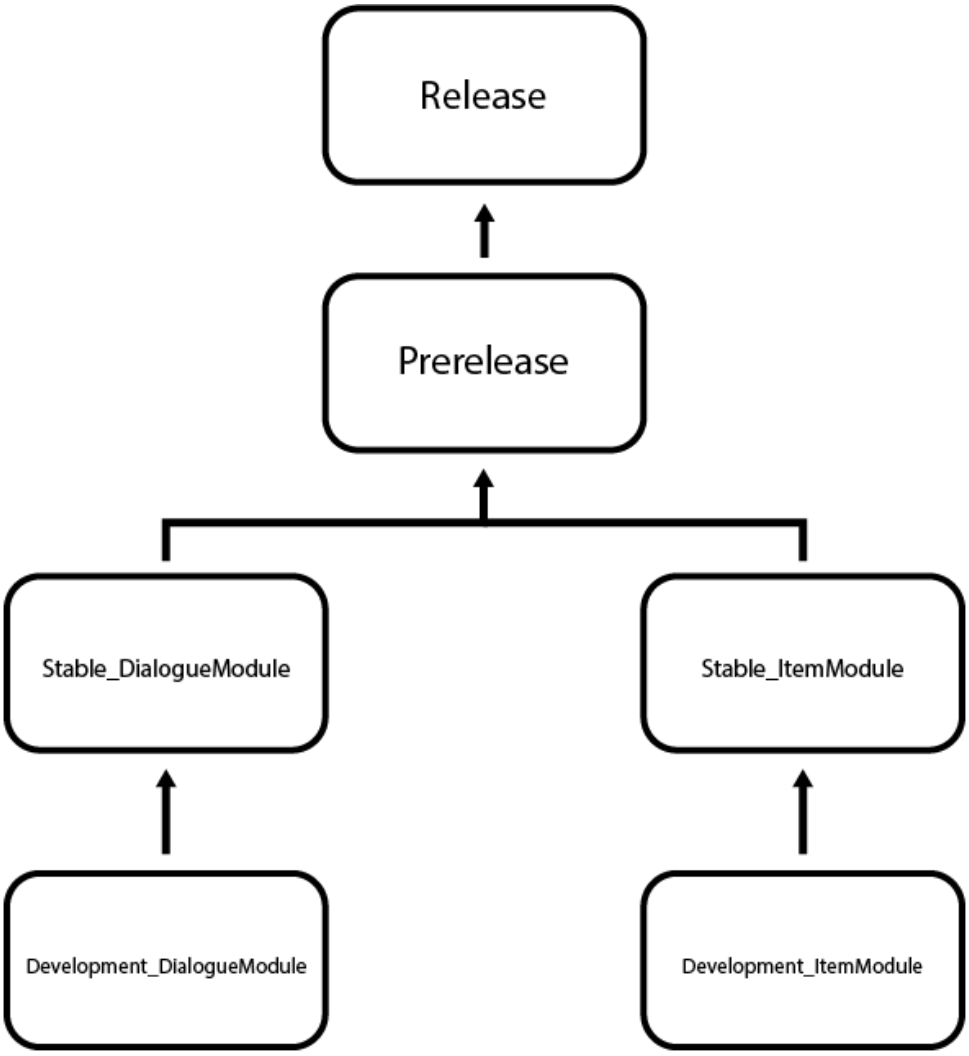

Figura 16 Representació visual de la jerarquia de branques. Font: Elaboració pròpia

5.1 Cronograma

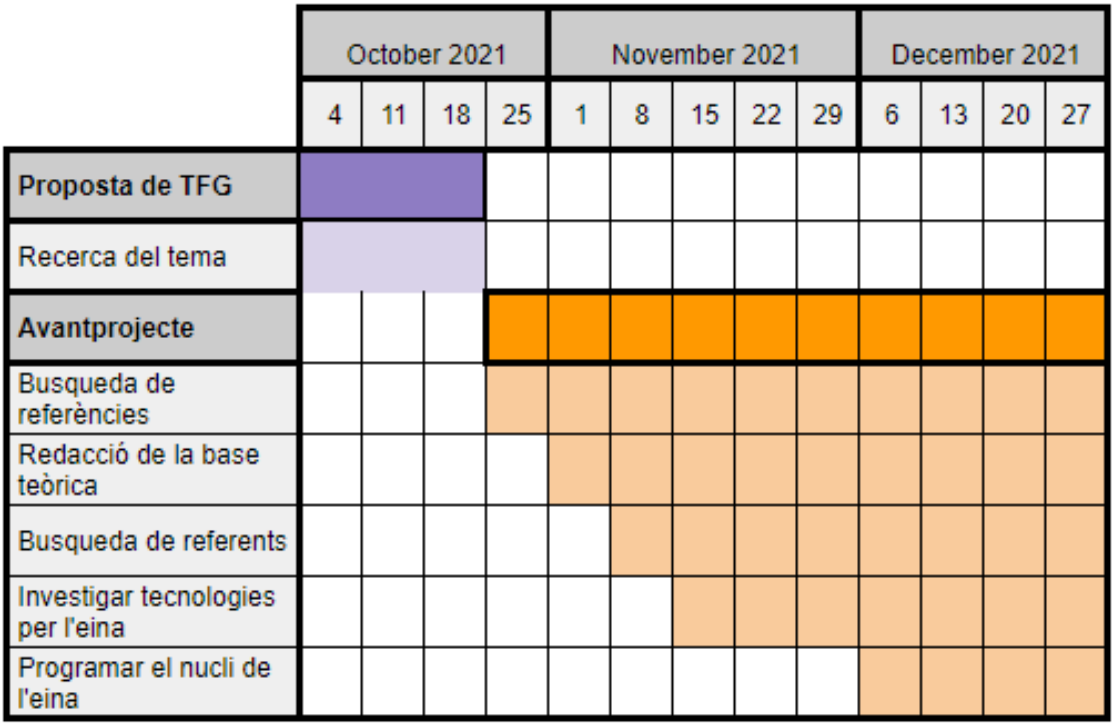

Figura 17 Cronograma del TFG 1/3. Font: Elaboració pròpia.

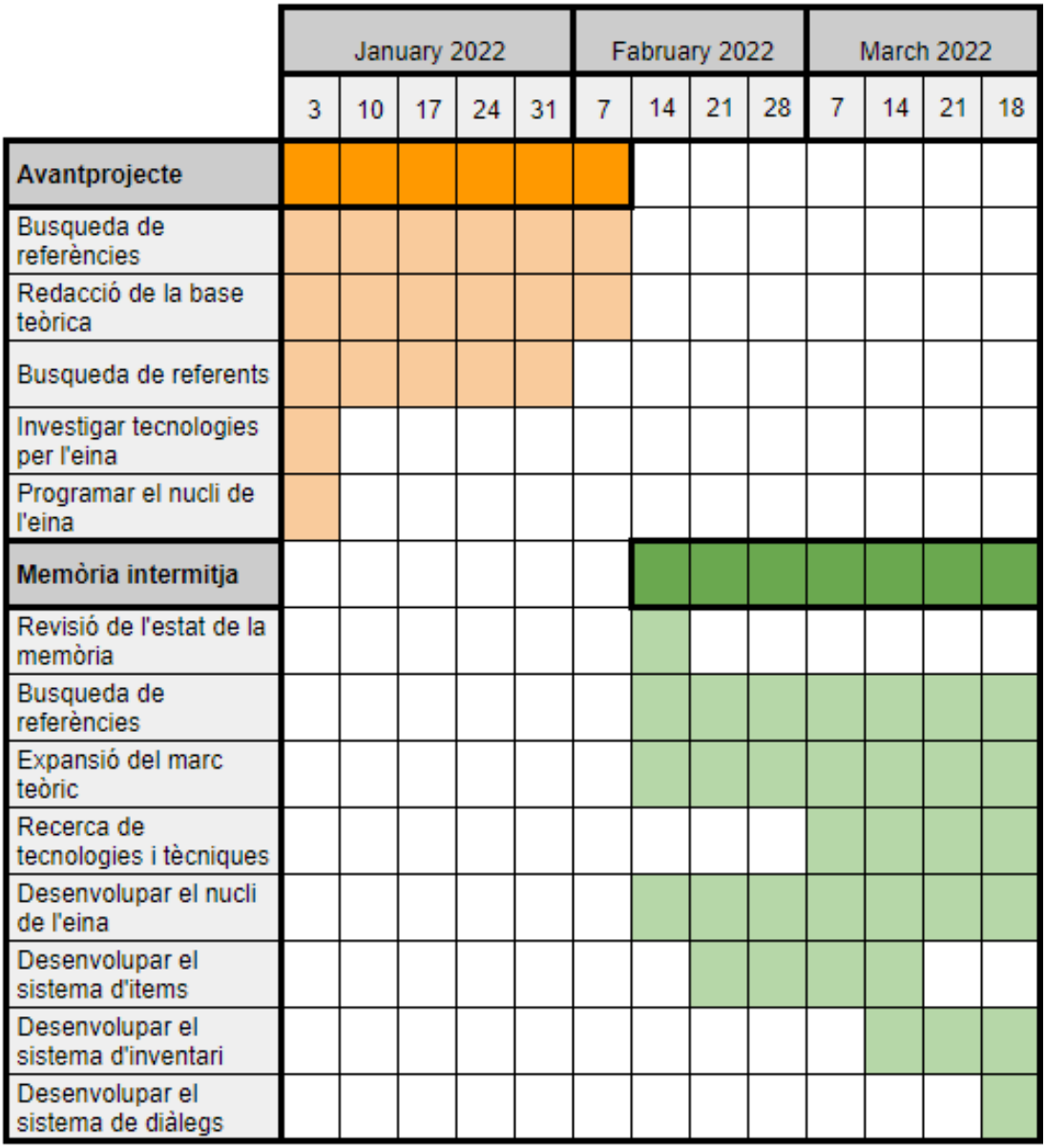

Figura 18 Cronograma del TFG 2/3. Font: Elaboració pròpia.

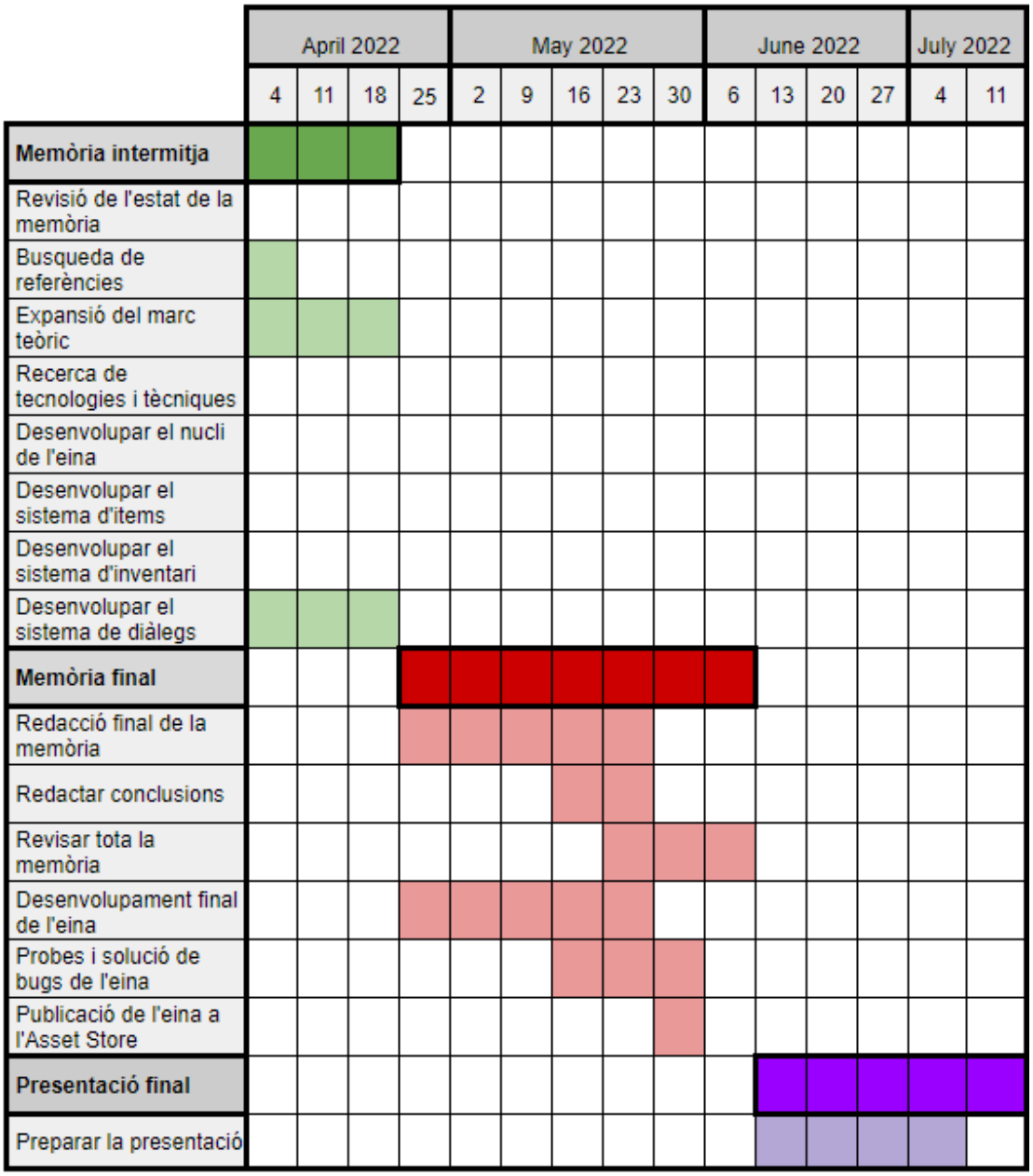

Figura 19 Cronograma del TFG 3/3. Font: Elaboració pròpia.

6. Desenvolupament i resultats

En aquest apartat s'explicarà el desenvolupament de l'eina d'Unity, des de la planificació, fins a la publicació i els resultats finals.

6.1 Desenvolupament

Per explicar el desenvolupament s'han de tenir en compte els objectius del projecte.

Com ja s'ha indicat anteriorment, el principal propòsit és desenvolupar una eina pel motor gràfic Unity que faciliti el desenvolupament de jocs RPG. Per això s'han definit 2 objectius principals: desenvolupar un sistema d'ítems i desenvolupar un sistema de diàlegs. A més a més, s'han afegit com a metes secundàries: característiques tècniques per facilitar a l'usuari final l'ús i ampliació de l'eina i la publicació de l'eina a l'Asset Store d'Unity.

Vistos els objectius es pot explicar el desenvolupament: en plantejar l'eina es defineixen 3 parts.

La primera part, anomenada CoreModule, on se situarà tot allò que pugui ser utilitzat per més d'un mòdul. Un exemple d'això és un "Logger", que permeti imprimir a la consola d'Unity els diferents errors que vagin succeint en el projecte. Aquest també podria tenir utilitats per funcions de càrrega d'assets, guardar icones comunes, etcètera.

La segona part, anomenada DialogueModule, on estarà tot el relacionat amb el sistema de diàlegs.

La tercera part, anomenada ItemModule, que inclourà tot el relacionat amb el sistema d'ítems.

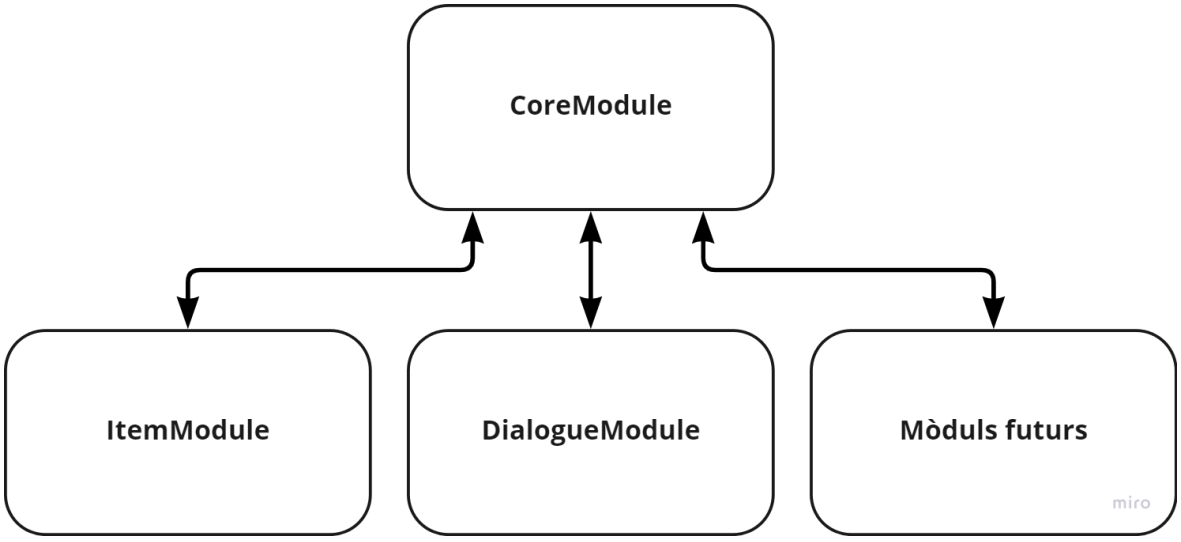

Figura 20 Divisió de mòduls de l'eina. Font: Elaboració pròpia.

Es divideix així per evitar crear dependències entre mòduls que puguin donar problemes i permetre a l'usuari només instal·lar els mòduls que ell necessiti. Tot el que pugui ser utilitzat de forma comuna, es posarà al CoreModule, de tal manera que s'eviti repetir el codi en els diferents mòduls.

6.1.1 Primera versió del mòdul de diàlegs

El primer que s'ha desenvolupat ha estat una primera iteració del mòdul de diàlegs.

Aquesta versió ha servit diversos propòsits. Un d'ells i el principal era aprendre en profunditat el sistema de UI Toolkit que implementa Unity pel desenvolupament d'interfícies. L'altre era estudiar molt més com estructurar l'eina internament per poder processar totes les dades, poder crear el sistema de guardat d'aquestes, i aconseguir crear una eina sense errors.

Errors principals detectats en aquesta primera versió:

- Incloure codi de l'editor en scripts que han de ser utilitzats a la build del joc: això provoca que s'hagi de reescriure gairebé per complet el codi, ja que s'ha estructurat tot el mòdul d'una forma que queda totalment descartada.
- Donar massa pes a un sol script de l'editor: quan es fan eines per a l'editor a Unity és molt fàcil acabar donant molt de pes a un sol script, per la manera

en què es programa amb UI Toolkit i IMGUI; tot i això, es pot dividir i, per tant, fer codi més llegible, flexible i fàcil de mantenir.

6.1.1.1 Resolent els problemes

1. El primer problema se soluciona de la següent manera: cada mòdul es dividirà en dos, amb un apartat pel Runtime (part jugable del joc) i un apartat per a l'Editor (tot el relacionat amb l'eina i l'editor). A més a més, cada un d'aquests apartats inclourà el seu propi "Assembly Definition".

   Els Assembly Definition són un tipus d'asset a Unity que et permet organitzar el projecte. Qualsevol script creat dins d'una carpeta pertanyent a un Assembly Definition, només pot accedir a altres scripts del mateix Assembly Definition, a no ser que n'inclogui un altre com a referència i, per tant, hi puguis accedir (Unity, 2022).

   Això concedeix diversos avantatges:

   - Pots determinar a quines plataformes va dirigit el codi.
   - Evites dependències entre codi.
   - Retalles temps de recompilació, ja que si els scripts d'un assembly no s'han modificat, no s'inclouen en la recompilació general del projecte.

   Així mateix, Unity no permet incloure scripts que treballin amb l'editor a les builds del joc. En conclusió, és una bona eina per forçar al desenvolupador a no incloure referències a l'editor al codi de Runtime.

2. El segon problema, se soluciona de la següent manera: abans tot el relacionat amb l'editor estava al mateix script de la finestra de l'editor. La nova versió comptarà amb un script que s'encarregui de crear la finestra de l'editor, i a partir d'aquí, cada component de la interfície comptarà amb el seu script propi, de tal manera que se segueix el principi de responsabilitat única, i ajuda al manteniment del codi.

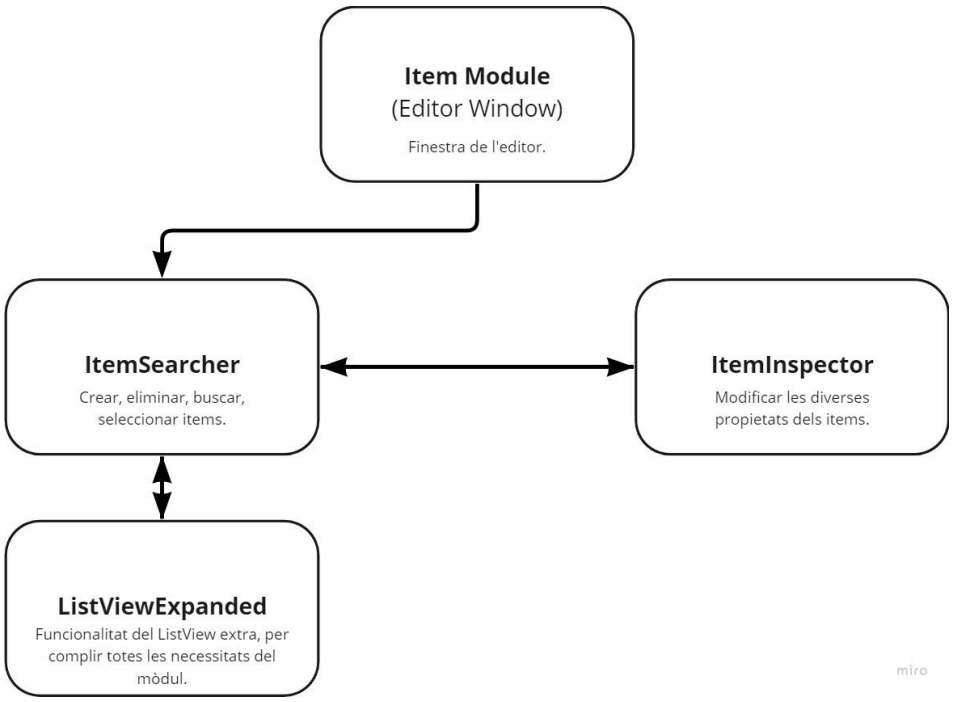

Figura 21 Estructura dels scripts de la interfície de la finestra a l'editor al mòdul d'ítems. Font: Elaboració pròpia.

6.1.2 Mòdul d'ítems

El desenvolupament del mòdul d'ítems ha estat molt més eficaç, atès que s'ha guanyat molta experiència amb la primera versió del mòdul de diàlegs.

6.1.2.1 Taula de característiques

En aquest cas, s'ha determinat un llistat de funcionalitats per a la confecció del mòdul.

Descripció de la característica	***Essencial***
Diferents col·leccions d'ítems, per ajudar al control de versions, organització, i rendiment en obtenir objectes.	SI
Afegir i eliminar ítems.	SI
Clonar ítems.	NO
Inspeccionar ítems.	SI
Propietats essencials: ID, nom, icona, prefab.	SI
Propietats personalitzables.	NO
Propietats extra: descripció.	SI
Configuració de sistema d'inventari.	NO
UI Flexible a canvis.	SI
Permetre a l'usuari ampliar fàcilment les propietats i funcionalitats dels ítems.	SI
Llistat d'ítems reordenable	SI
Suport de fer/desfer	NO

Buscador d'ítems.	SI
Custom Inspector per a l'arxiu de col·leccions d'ítem.	NO
Sistema d'etiquetes personalitzable. Serveix per agrupar i poder buscar al buscador per etiqueta	NO
Sistema econòmic per establir preu de l'objecte	NO

Taula 2 Llista de característiques planejades per al mòdul d'ítems. Font: Elaboració pròpia.

Un cop s'ha disposat de les característiques i funcionalitats clares, s'ha començat el desenvolupament.

El primer que s'ha treballat és la interfície d'usuari. Per suportar la modularitat, s'ha creat un arxiu base .uxml (format en què es guarda i s'edita la interfície d'usuari), seguit de 2 arxius .uxml més. El primer dedicat a la part de creació, eliminació, cerca d'ítems i selecció de la col·lecció d'ítems. Per altra banda, el següent arxiu es dedica a modificar totes les propietats dels ítems.

Aquesta divisió es pot veure de forma més clara a la Figura *20*. Es crea un arxiu d'uxml i un script per cada peça de la interfície.

Unity dona per defecte diferents eines a l'hora de desenvolupar la interfície d'usuari, des de textos, botons, desplegables, a fins i tot corbes i colors, i molts més. Alguns només es poden utilitzar per a l'editor, com en el cas de l'editor de corbes i colors, d'entre altres.

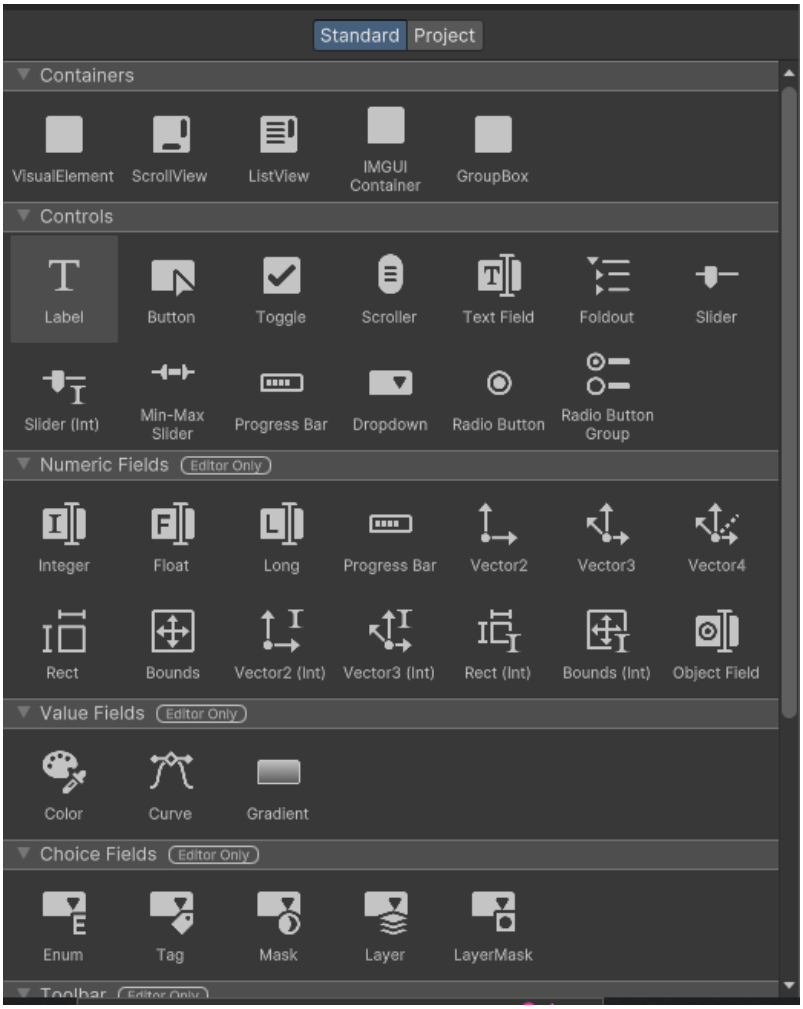

Figura 22 Algunes de les eines que proporciona Unity per modificar les dades des de la interfície. Font: Elaboració pròpia.

Això facilita bastant el desenvolupament, ja que fàcilment pots arrossegar qualsevol d'aquests elements a l'inspector i ja disposen de la funcionalitat incorporada.

No obstant, UI Toolkit compta amb diferents elements que no estan exposats de forma habitual, però es poden forçar a sortir.

Un d'aquests elements, utilitzats en ambdós mòduls, és el "TwoPaneSplitView". Un panell dividit en dues zones que permet redimensionar-les mitjançant una barra vertical o horitzontal. Aquest fet és molt útil quan tens elements que normalment es voldran tenir en una mida reduïda, però potser l'usuari vol expandir per veure amb més detall.

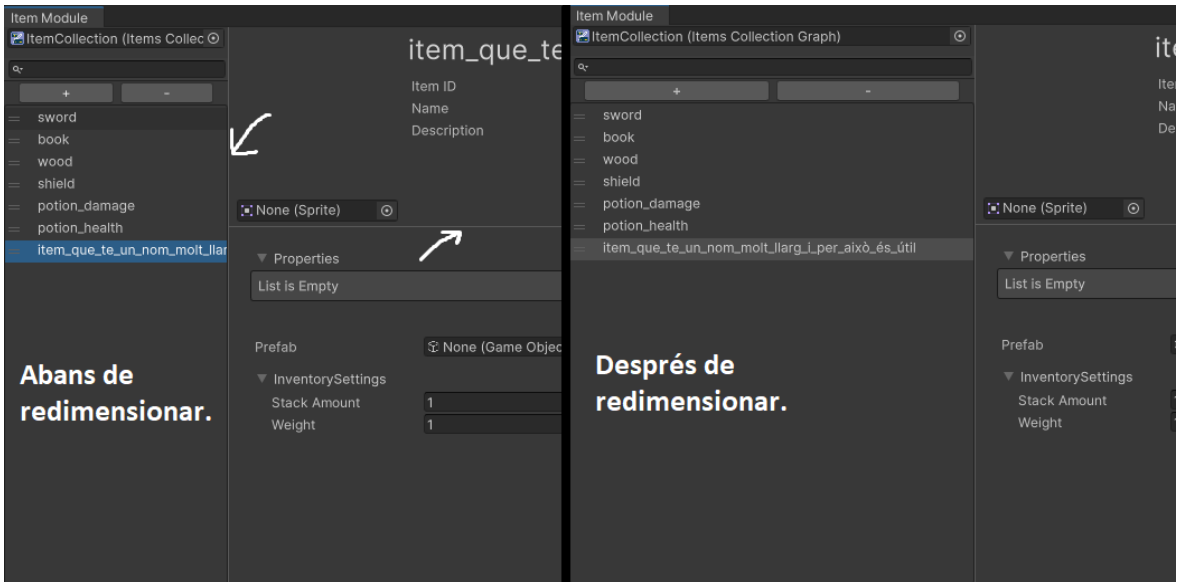

Figura 23 Exemple d'ús del TwoPanelSplitView al mòdul d'ítems. Font: Elaboració pròpia.

Un cop està dissenyada la interfície, les dades encara no estan connectades a aquests diversos camps de l'editor.

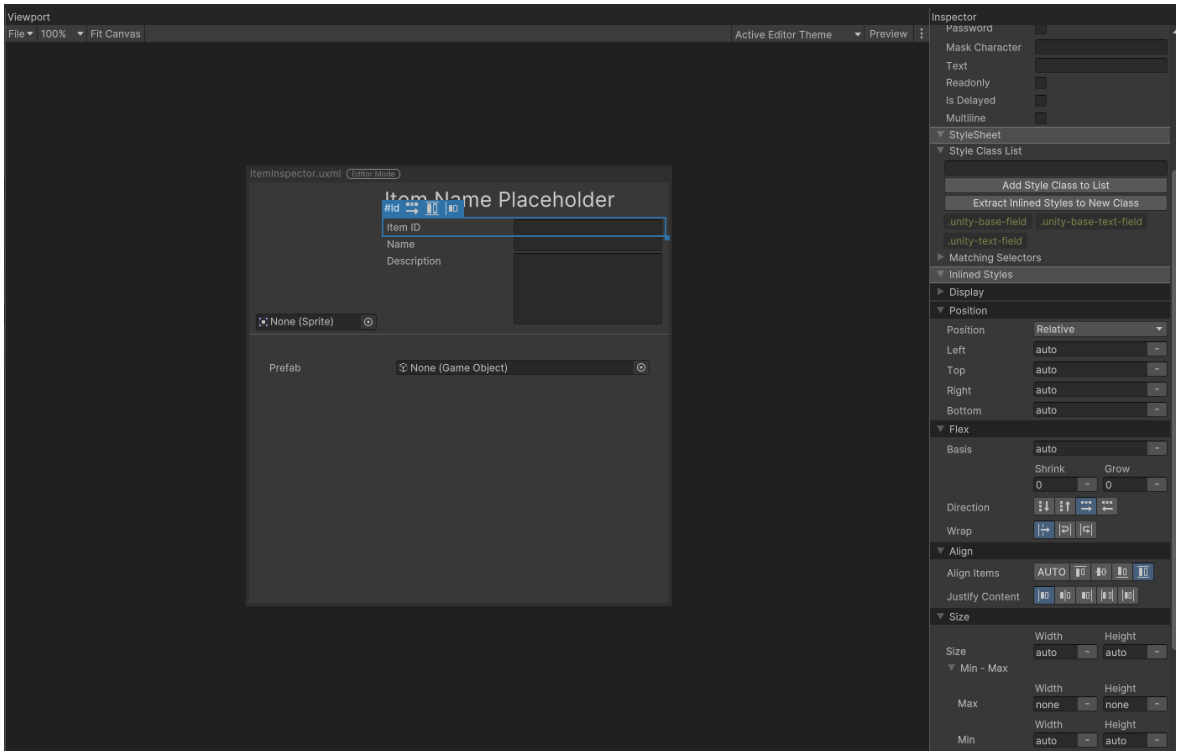

Figura 24 Mostra de part de l'editor de UI Toolkit. Font: Elaboració pròpia.

Per connectar aquestes dades Unity ofereix principalment 2 opcions: el binding i el sistema d'esdeveniments, que ja s'han revisat al marc teòric.

Pel mòdul d'ítems s'ha optat per una solució intermèdia, on s'usa el binding per registrar els canvis de valor (amb aquest mètode es té l'avantatge de tenir funcionalitat de fer/desfer, i, per altra banda, s'utilitza el sistema d'esdeveniments per donar comportament afegit quan el valor es modifica).

En la Figura 25 es pot visualitzar un exemple d'aquesta barreja en el cas de la propietat "Id" dels ítems.

```
//S'obté i es guarda un element de tipus TextField amb nom "Id".
_id = _visualElement.Q<TextField>(name: ItemInspectorUXML.Id);
//S'assigna el bindingPath al nom de la variable Id per fer el binding.
// D'aquesta manera si el nom canvia, s'evita el desavantatge que es modifiqui el
//comportament sense saber-ho. (Això es pot fer perquè la variable és
//pública, no funcionaria amb variables privades).
_id.bindingPath = nameof(Item.id);

//Registrem el TextField Id al esdeveniment ValueChanged, que es crida quan el valor es modifica.
_id.RegisterValueChangedCallback(evt :ChangeEvent<string> =>
{
    //Mitjançant una funció lambda, executem el codi que necessitem.
    _itemModule.UpdateItem(_currentSelectedItem);
    if (evt.newValue == string.Empty)
    {
        _title.text = "Insert an Id";
        return;
    }
    _title.text = evt.newValue;
});
```

Figura 25 Mostra de com s'ha implementat el sistema d'esdeveniments i binding en C#. Font: Elaboració pròpia.

L'única problemàtica detectada en la versió final és la següent:

- Guardar tota la informació en un sol asset pot donar problemes en el desenvolupament quan s'utilitza control de versions i més d'una persona pot estar modificant l'asset.

Si bé és cert que s'ha creat una estructura de col·leccions d'ítems per solucionar-ho, cada ítem es guarda dins de la col·lecció. La finalitat era evitar el fet de tenir problemes en treballar més d'una persona amb els ítems. Però si 2 persones modifiquen la mateixa col·lecció, poden aparèixer problemes.

En el mòdul d'ítems no és un problema tan gran, no obstant això, pel mòdul d'ítems es tindrà en compte per evitar problemes.

6.1.2.2 Implementació

En aquest cas, la implementació ha començat per desenvolupar la part del Runtime. S'ha definit com a nucli central la classe Item.

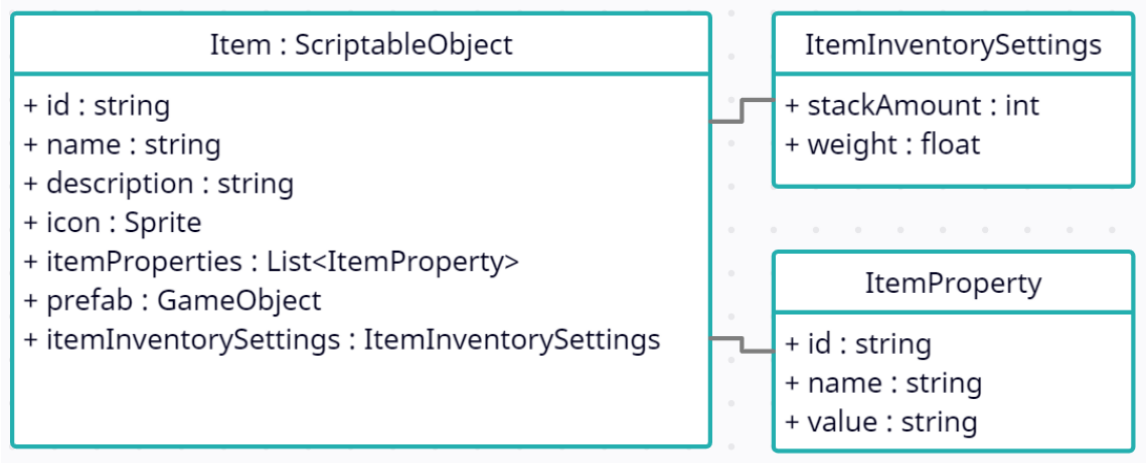

Figura 26 Diagrama UML representant la classe d'Ítem, i altres classes utilitzades. Font: Elaboració pròpia.

Un dels elements a destacar és la variable itemProperties. ItemProperties permet afegir qualsevol tipus de propietat en format de string. Per exemple, si es vol definir la quantitat de vida que una poció de regeneració concedeix es podria definir un element de la següent manera:

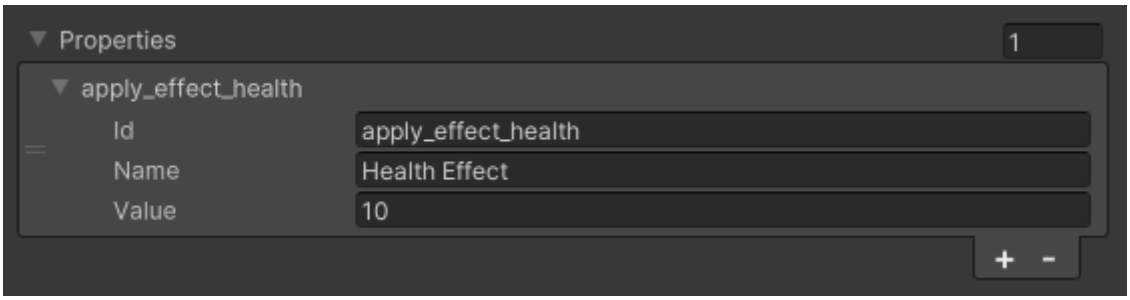

Figura 27 Exemple d'ús de l'Item Property en una poció de curació. Font: Elaboració pròpia.

Després, des d'un altre script es pot agafar la propietat, i segons l'Id usar el valor d'una manera o altra.

Per altra banda, s'ha creat una classe independent del sistema d'inventari. Això facilita la modificació de l'usuari per sistemes d'inventari específics, ja que la interfície s'adaptarà automàticament.

Per acabar, la classe ItemCollectionGraph s'encarrega de guardar una referència a tots els ítems, i dona funcions per obtenir ítems de la col·lecció.

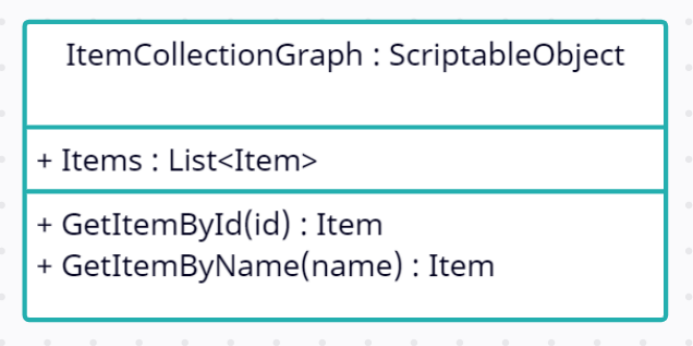

Figura 28 Diagrama UML de la classe ItemCollectionGraph. Font: Elaboració pròpia.

Per a la interfície s'ha establert la següent estructura:

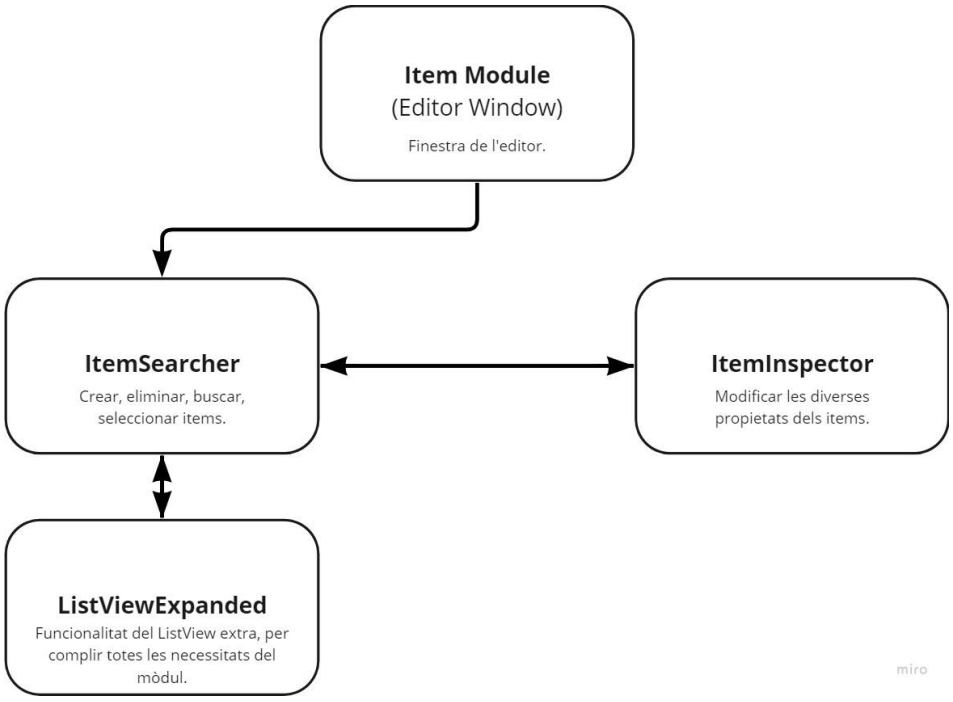

Figura 29 Estructura dels scripts de la interfície de la finestra a l'editor al mòdul d'ítems. Font: Elaboració pròpia.

La classe ItemModule s'encarrega d'inicialitzar la finestra de l'editor. Amb això, la classe ItemSearcher té la funcionalitat de deixar escollir a l'usuari un ítem, i enviar-li a l'ítem inspector per mostrar-lo en pantalla. A més a més, ItemSearcher utilitza la classe ListViewExpanded, una classe creada per administrar la llista d'ítems, i poder afegir filtres segons el buscador.

6.1.3 Mòdul de diàlegs

Amb l'experiència obtinguda en la confecció de la primera versió del mòdul de diàlegs i amb el mòdul d'ítems, s'ha programat un complet i nou sistema de diàlegs.

Per això de nou, s'ha creat una taula de característiques:

Descripció de la característica.	***Essencial***
Panell gràfic on situar els nodes.	SI
Desplaçar-se i seleccionar.	SI
Creació de nodes.	SI
Creació de grups.	SI
Node de diàleg.	SI
Node de diàleg amb opcions.	SI
Sistema de guardat.	SI
Sistema per evitar noms repetits.	SI
Minimapa.	NO
Zoom in/out.	NO
Múltiple selecció.	NO
Sistema de copiar, retallar, enganxar.	NO
Sistema de guardat automàtic.	NO
Suport a nous tipus de nodes (visual).	NO
Suport a nous tipus de nodes (programació).	NO

Múltiples utilitats a l'hora de desconnectar nodes.	NO
Opcions pel sistema de guardat individual/separat.	NO
Elecció de les carpetes de guardat.	NO

Taula 3 Llista de característiques planejades pel mòdul de diàlegs. Font: Elaboració pròpia.

Aquest mòdul s'ha estructurat de la següent manera: per la interfície d'usuari s'han creat 4 scripts principals.

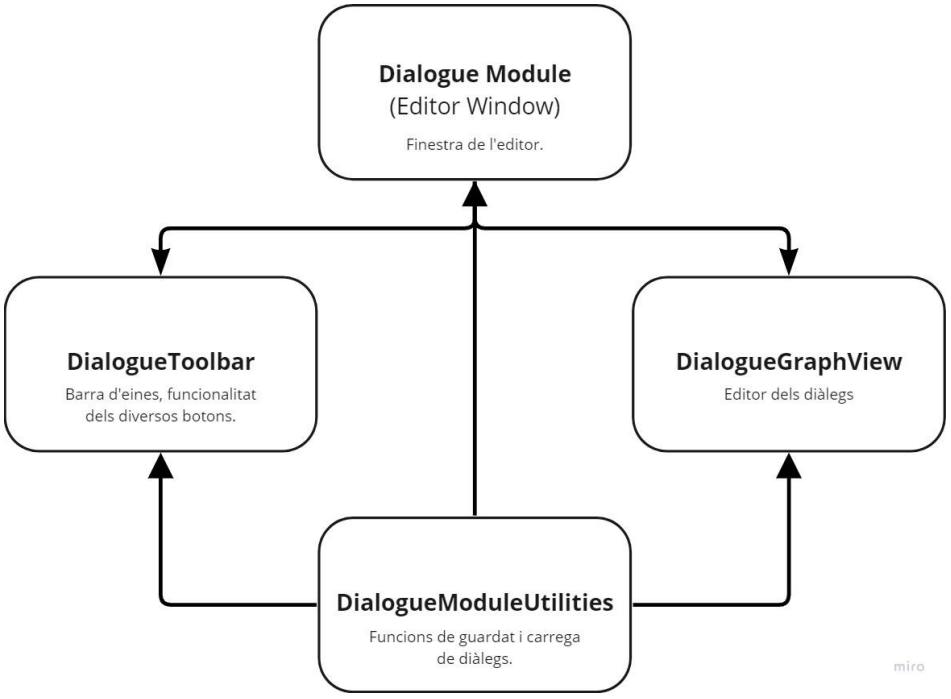

Figura 30 Estructura dels scripts de la interfície de la finestra a l'editor al mòdul de Diàlegs Font: Elaboració pròpia.

6.1.3.1 GraphView

Unity incorpora per defecte un sistema de visualització gràfica, mitjançant el component GraphView. Com en el cas del "TwoPanelSplitView" que hem analitzat abans, s'ha de forçar per codi per poder utilitzar.

```
//Es crea una classe que deriva de l'original GraphView.
11 usages  3 exposing APIs
public class DialogueGraphView : GraphView
{
    //Se sobreescriu la classe UxmlFactory, heretant de la classe UxmlFactory passant
    //la mateixa classe creada i els UxmlTraits originals del GraphView.
    //Els uxml traits són les propietats que l'editor mostrarà per aquest component.
    public new class UxmlFactory : UxmlFactory<DialogueGraphView, GraphView.UxmlTraits>
    {
    }
```

Figura 31 Mostra de com s'exposa un component d'Unity ocult. Font: Elaboració pròpia.

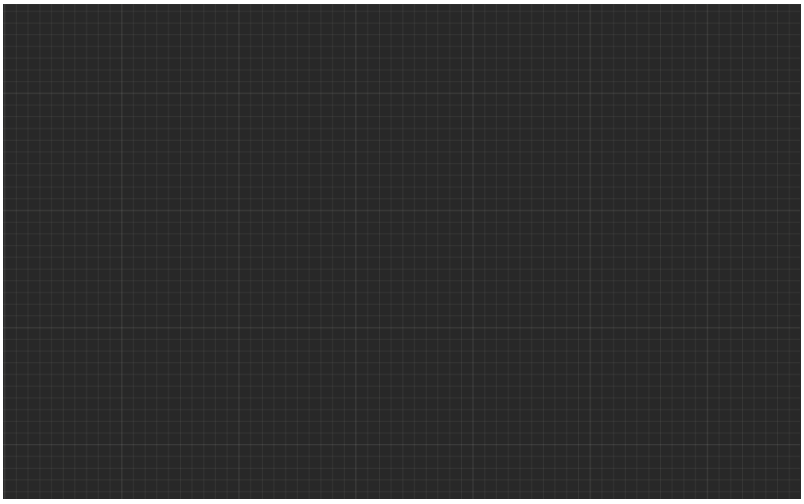

Figura 32 Mostra d'*u*n GraphView creat després d'estilitzar-lo. Font: Elaboració pròpia.

6.1.3.2 USS

Per estilitzar el mòdul de diàlegs, en aquest cas, s'hi ha utilitzat en gran quantitat l'Unity Style Sheet (uss).

```
:root {
    --dm-colors-grid-background: rgb(40, 40, 40);
    --dm-colors-grid-line: rgba(193, 196, 192, 0.1);
    --dm-colors-grid-thick-line: rgba(193, 196, 192, 0.1);
    --dm-metrics-grid-spacing: 25;
}


GridBackground{
    --grid-background-color: var(--dm-colors-grid-background);
    --line-color: var(--dm-colors-grid-line);
    --thick-line-color: var(--dm-colors-grid-thick-line);
    --spacing: var(--dm-metrics-grid-spacing);
}
```

Figura 33 Mostra d'ús de variables i estil amb Unity Style Sheets pel DialogueGraphView. Font: Elaboració pròpia.

6.1.3.3 Implementació

El primer pas per crear el mòdul de diàlegs ha estat crear la interfície. Com s'ha explicat a la Figura *30*, hi ha una classe principal què és la qual hereta d'EditorWindow (classe base per crear finestres a l'editor d'Unity) que rep el nom DialogueModule. Aquesta classe simplement s'encarrega d'afegir els .uxml de la interfície i inicialitzar els diversos components.

Per una banda, el DialogueToolBar, que tan sols s'encarrega del comportament de la barra d'eines que té el mòdul: posar nom a l'arxiu, guardar, carregar, netejar el Graph i mostrar la versió del mòdul.

Per l'altre i com a part fonamental, el GraphView. La classe DialogueGraphView hereta de GraphView i s'encarrega de gestionar tot el relacionat amb el Graph de nodes.

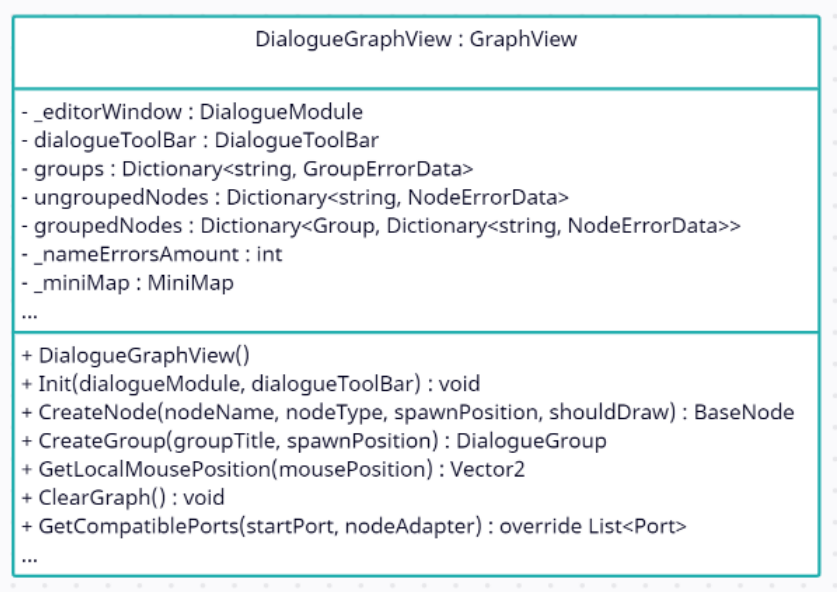

Figura 34 Diagrama UML de la classe DialogueGraphView. Font: Elaboració pròpia.

Com es pot veure a la Figura 34, DialogueGraphView compta amb diferents funcions per inicialitzar: el constructor per inicialitzar variables, afegir minimapa, etcètera, i per altra banda, el mètode "Init" que injecta les referències a les altres parts de la interfície. També compta amb mètodes per crear els nodes, i crear els grups. Això es pot fer des del menú contextual, on apareixen opcions per crear nodes de cada tipus i el grup, o bé des d'un menú de creació d'elements al Graph.

Per tal de suportar la creació de nodes i inserció automàtica al menú contextual o el menú de creació d'elements, s'ha utilitzat la reflexió, que ens permet obtenir objectes de tipus Type segons diferents paràmetres. S'ha creat una funció que rep un Type i retorna tots els Type fills del Type paràmetre. D'aquesta manera si un usuari crea un nou tipus de node, automàticament s'afegirà als menús de creació.

```
public static List<Type> GetInheritedClasses(Type baseType)
{
    return Assembly.GetAssembly(baseType).GetTypes() // Type[]
        .Where(type => type.IsClass && !type.IsAbstract && type.IsSubclassOf(baseType)).ToList(); // List<Type>
}
```

Figura 35 Funció per obtenir classes filla del Type paràmetre. Font: Elaboració pròpia.

El GraphView pot contenir 3 tipus d'elements. Nodes, grups i edges.

Per tant, a l'hora d'implementar els nodes pel diàleg, s'ha d'heretar la classe Node del GraphView.

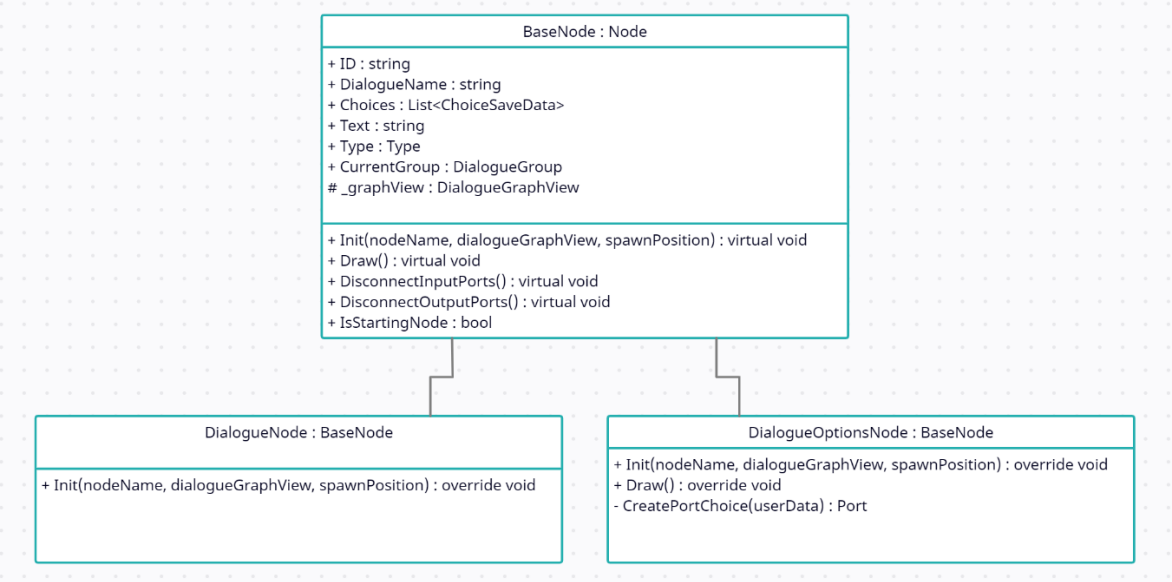

Figura 36 Diagrama UML representant les classes heretant de Node. Font: Elaboració pròpia.

Per poder crear, guardar dades en temps d'execució de l'eina i poder visualitzar el node dins el GraphView, s'ha de crear una classe que hereti de node i dibuixi almenys un element en pantalla. Un cop creada la classe base BaseNode i afegida tota la informació necessària, l'únic que s'ha de fer per crear nodes és heretar la classe BaseNode, i afegir el comportament desitjat. Es pot sobreescriure tant les funcions d'inicialització del node, que el configuren inicialment, com també les funcions de dibuixat, per poder afegir elements visuals al node i personalitzar-los a gust.

Per altra banda, tenim els grups, que han d'heretar de la classe Group. En aquest cas només cal mantenir en una variable la ID del grup, per tal de poder guardar i carregar el diàleg.

Finalment, tenim els edges. Els edges són els elements en forma de fletxes que connecten els diferents nodes. No necessitem crear cap classe pròpia dels edges, ja que quan vulguem crear edges l'únic que s'ha de fer és cridar des del port en el

qual estàs, la funció de connectar i passar per paràmetre el port al qual et vols connectar.

Per guardar els diàlegs s'han creat classes de dades tant per a l'edició a l'editor, com per al runtime. Per una banda, tenim les classes de dades de l'editor:

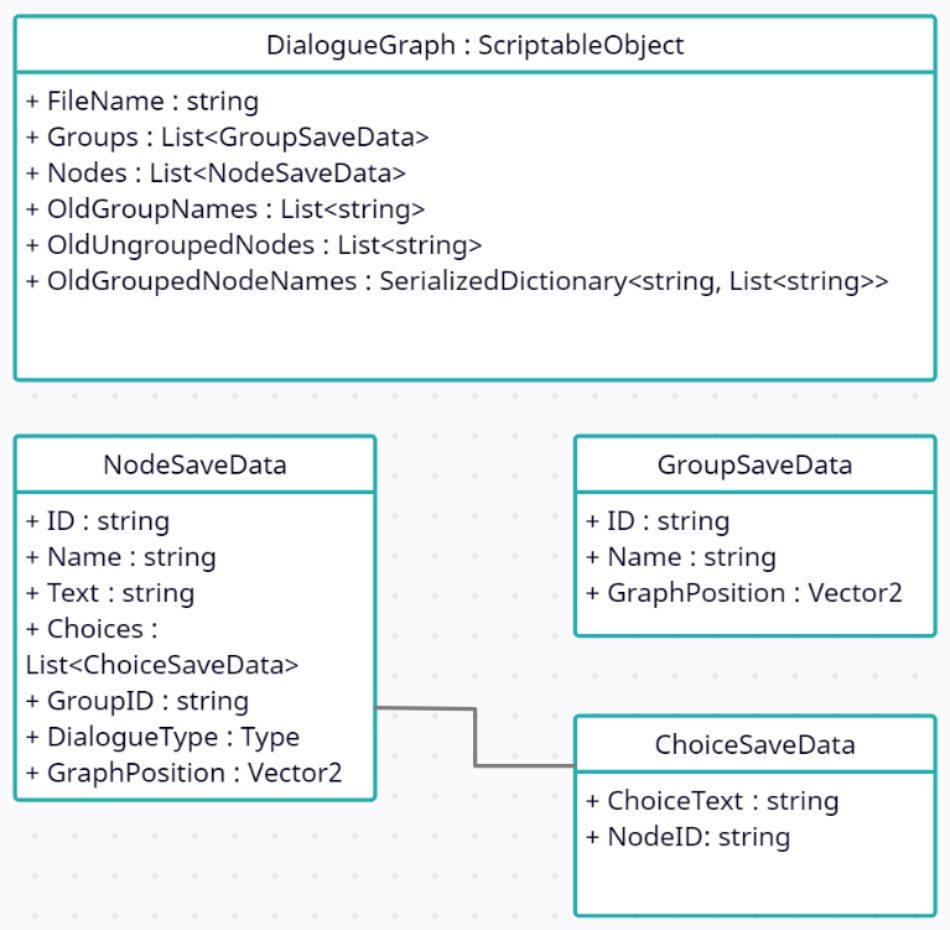

Figura 37 Classes de dades per guardar el diàleg. Font: Elaboració pròpia.

Es crea una diferenciació entre el guardat per l'editor i el guardat pel runtime, per evitar posar dades com la ID, o la posició en el graph, a les classes que funcionen dins el joc.

La classe principal és el DialogueGraph, que guarda referències als nodes i als grups. La classe GroupSaveData només s'encarrega de guardar el nom del group i la posició, a més de la ID per poder ser referenciat. La classe NodeSaveData s'encarrega de guardar la informació dels nodes, en especial destacar la variable GroupID, que guarda la ID del grup en el qual pertany. Finalment, la classe

ChoiceSaveData serveix només per guardar la ID del node al qual està connectat, i el text de resposta (si el node és de diàleg, el text estarà buit).

Per altra banda, tenim les classes de dades pel runtime.

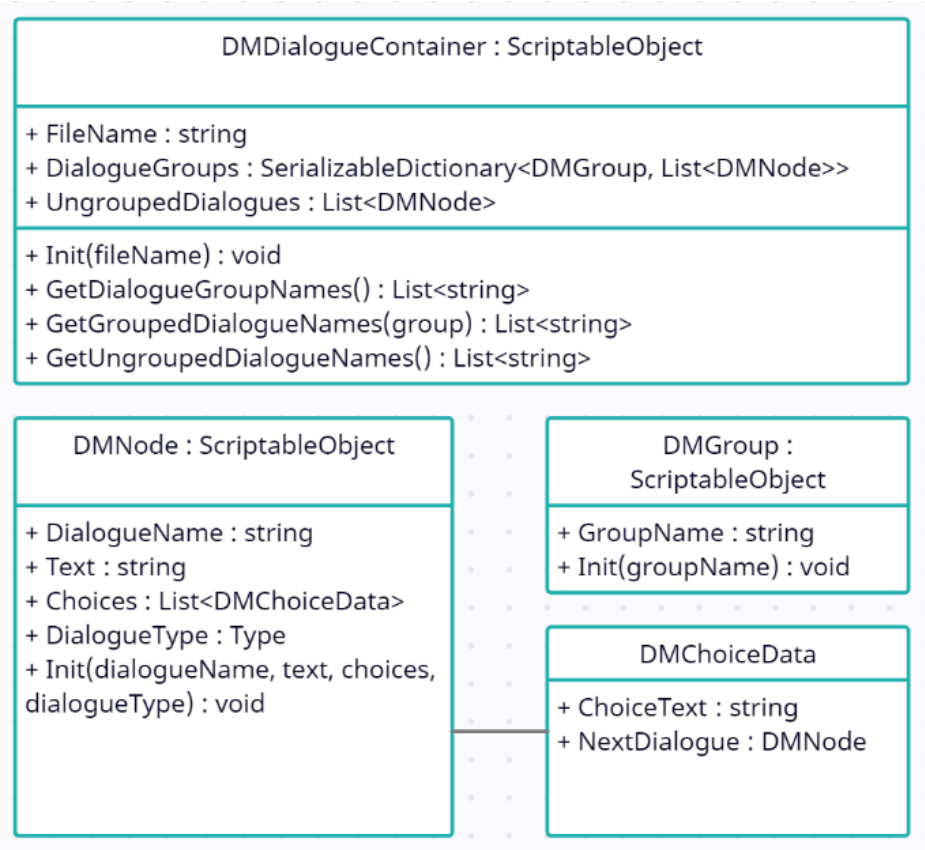

Figura 38 Classes per guardar dades del diàleg pel runtime. Font: Elaboració pròpia.

Eliminem les ID i variables sobrants usades només en l'editor, i canviem les referències d'ID per referències a l'objecte, com per exemple es pot veure en la classe DMChoiceData amb la classe NextDialogue.

6.2 Publicació a l'Unity Asset Store

En aquest apartat es repassarà el procés de publicació a la Unity Asset Store.

Per publicar a l'Unity Asset Store s'ha de crear un compte especial. En crear aquest compte es signen diversos contractes de publicació, permisos, i dades bancàries. Finalment, es pot accedir al panell per publicar a l'Asset Store.

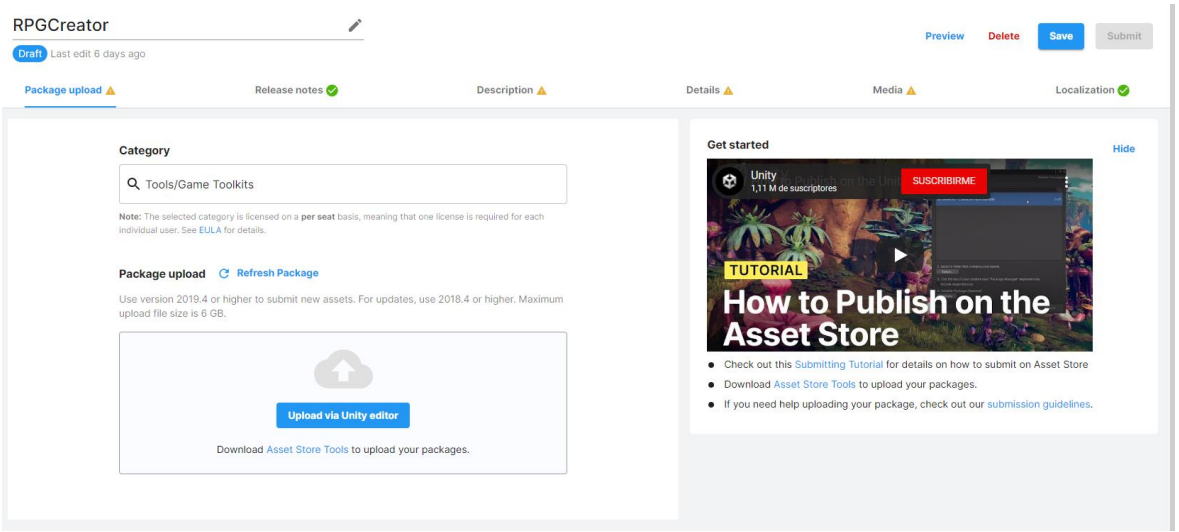

Figura 39 Panell d'edició de la publicació a l'Asset Store. Font: Elaboració pròpia.

Un cop al panell es configuren tots els paràmetres:

- Categoria
- Notes de la versió
- Descripció
- Preu
- Paraules claus
- Imatges de màrqueting.
- Localització (només a Inglés, Xinés, Japonés, Coreà).

Després d'enviar aquesta informació, la revisió tarda 30 dies laborables aproximadament, fins que finalment s'aprova.

Per a les imatges de màrqueting s'ha treballat amb l'artista Aylin Torunlar.

Unity Asset Store demana 4 tipus d'imatge:

- Imatge d'icona.
- Imatge de tipus targeta.
- Imatge de portada.
- Imatge per xarxes socials.

6.3 Resultats

6.3.1 Taula final de característiques del mòdul de diàlegs

A continuació s'exposa la taula final de característiques que s'han implementat.

Descripció de la característica.	***Essencial***	***Implementat***
Panell gràfic on situar els nodes.	SI	SI
Desplaçar-se i seleccionar.	SI	SI
Creació de nodes.	SI	SI
Creació de grups.	SI	SI
Node de diàleg.	SI	SI
Node de diàleg amb opcions.	SI	SI
Sistema de guardat.	SI	SI
Sistema per evitar noms repetits.	SI	SI
Minimapa.	NO	SI
Zoom in/out.	NO	SI
Múltiple selecció.	NO	SI
Sistema de copiar, retallar, enganxar.	NO	SI
Sistema de guardat automàtic.	NO	SI
Suport a nous tipus de nodes (visual).	NO	SI
Suport a nous tipus de nodes (programació).	NO	SI
Múltiples utilitats a l'hora de desconnectar nodes.	NO	SI

Opcions pel sistema de guardat individual/separat.	NO	NO
Elecció de les carpetes de guardat.	NO	NO

Taula 4 Llista final de característiques incloses al mòdul de diàlegs. Font: Elaboració pròpia.

A continuació, es mostren captures de pantalla del resultat final:

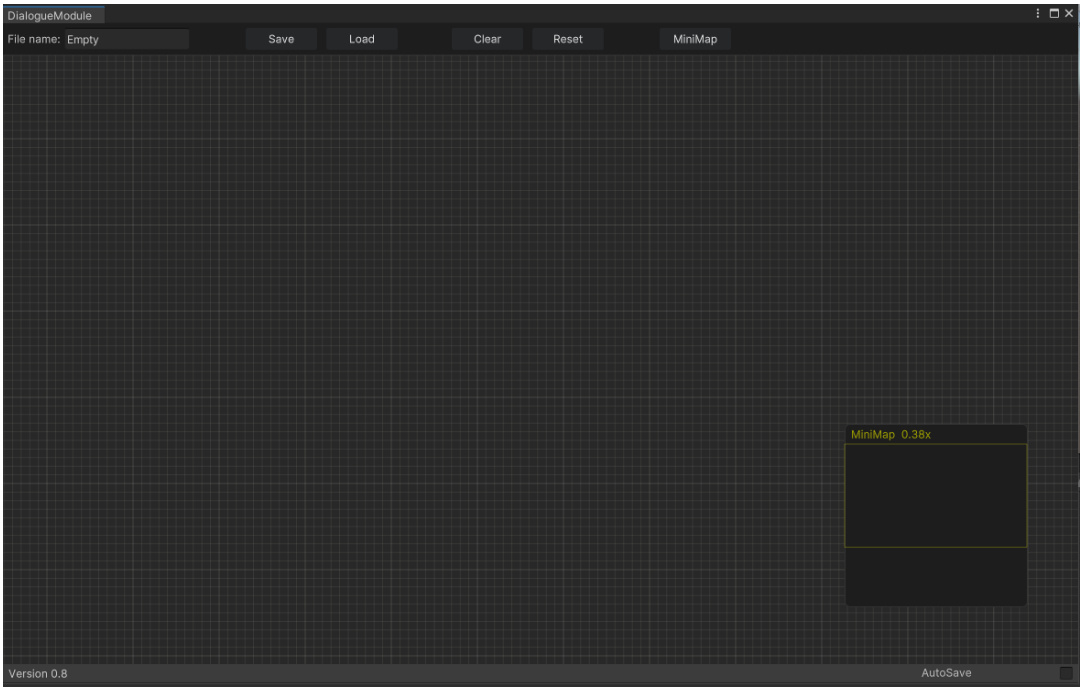

Figura 40 Mostra de la finestra de l'editor del mòdul de diàlegs buida. Font: Elaboració pròpia.

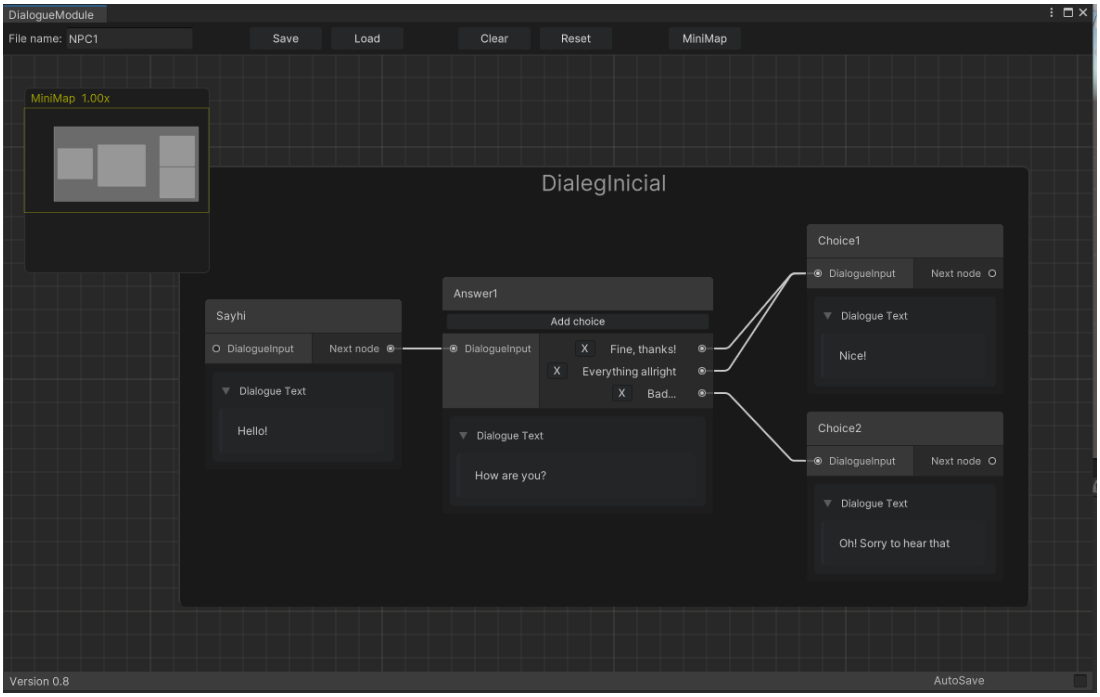

Figura 41 Mostra de l'ús de l'editor del mòdul de diàlegs. Font: Elaboració pròpia.

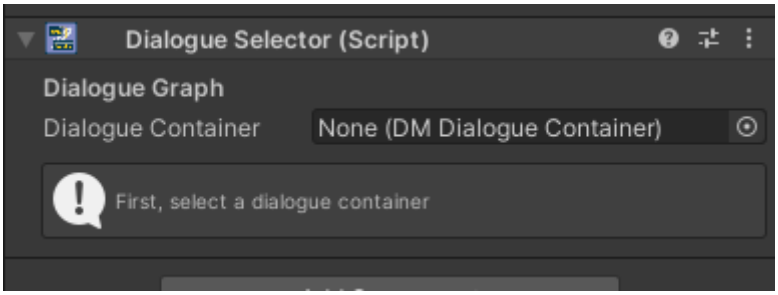

Figura 42 Component per incorporar els diàlegs al joc utilitzant CustomInspector i IMGUI. Font: Elaboració pròpia.

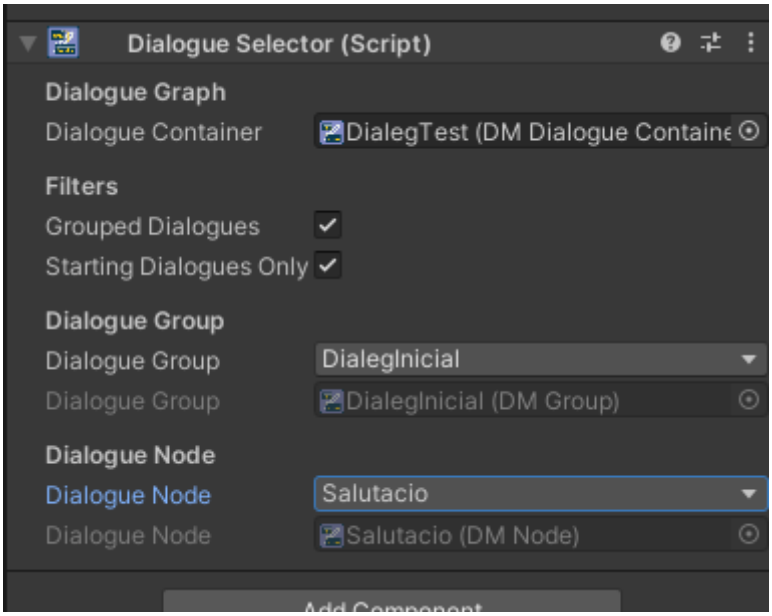

Figura 43 Component per incorporar els diàlegs al joc utilitzant CustomInspector i IMGUI amb diàleg seleccionat. Font: Elaboració pròpia.

6.3.2 Taula final de característiques del mòdul d'ítems

A continuació s'exposa la taula final de característiques que s'han implementat.

Descripció de la característica.	***Essencial***	***Implementat***
Diferents col·leccions d'ítems, per ajudar al control de versions, organització, i rendiment en obtenir objectes.	SI	SI
Afegir i eliminar ítems.	SI	SI
Clonar ítems.	NO	SI
Inspeccionar ítems.	SI	SI
Propietats essencials: ID, nom, icona, prefab.	SI	SI
Propietats personalitzables.	NO	SI

Propietats extra: descripció.	SI	SI
Configuració de sistema d'inventari.	NO	SI*[2]
UI Flexible a canvis.	SI	SI
Permetre a l'usuari ampliar fàcilment les propietats i funcionalitats dels ítems.	SI	SI*[3]
Llistat d'ítems reordenable.	SI	SI
Suport de fer/desfer.	NO	SI*[4]
Buscador d'ítems.	SI	SI
Custom Inspector per l'arxiu de col·leccions d'ítems.	NO	SI
Sistema d'etiquetes personalitzable. Serveix per agrupar i poder buscar al buscador per etiqueta.	NO	NO
Sistema econòmic per establir preu de l'objecte.	NO	NO

Taula 5 Llista final de característiques incloses al mòdul d'ítems. Font: Elaboració pròpia.

Analitzant la taula podem observar que gairebé totes les funcionalitats han estat incorporades, incloent-hi bona part de les opcionals. Per qüestions de temps no s'han pogut implementar algunes de les característiques, però poden ser incloses en el futur més enllà del treball.

[2] S'ha implementat una versió molt bàsica i senzilla de la característica.
[3] Respecte al codi, si és cert, però la UI no s'actualitza automàticament, per tant, s'ha de modificar a mà.
[4] S'ha implementat menys a l'hora de crear/eliminar objectes.

A continuació, es mostren captures de pantalla del resultat final:

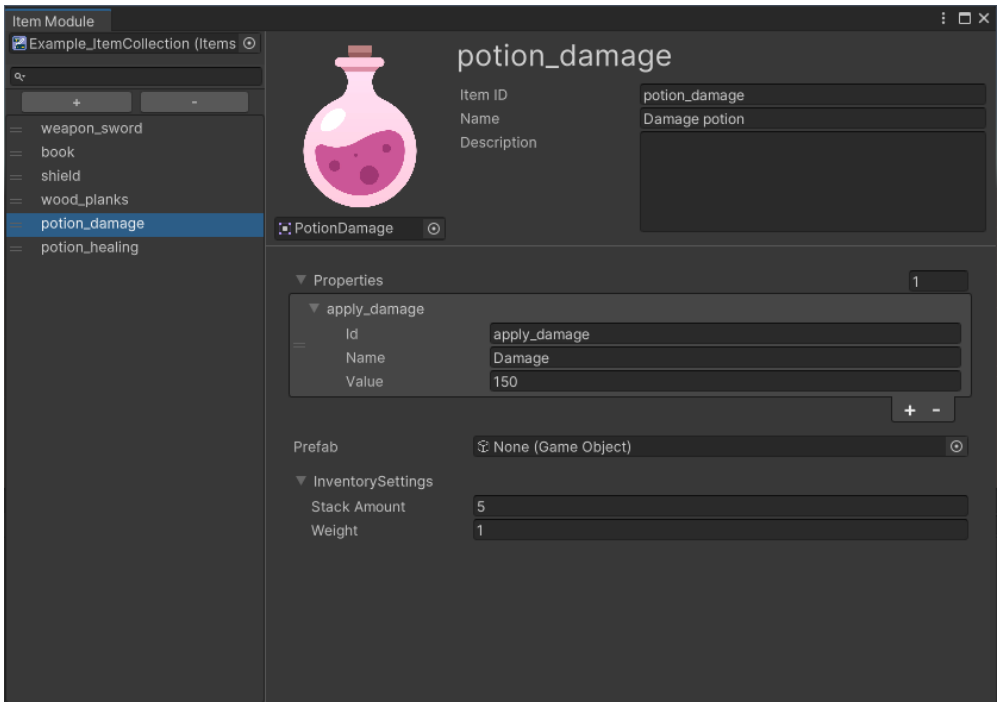

Figura 44 Exemple d'ítem (poció de vida) creat amb el mòdul d'ítems. Font: Elaboració pròpia.

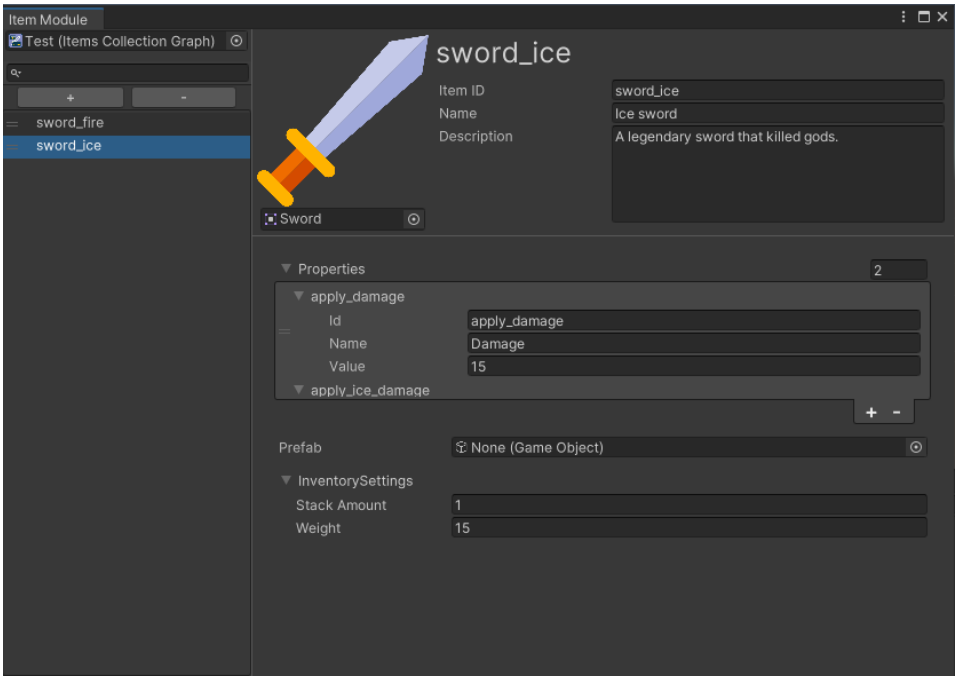

Figura 45 Exemple d'ítem (espasa) creat amb el mòdul d'ítems. Font: Elaboració pròpia.

6.3.3 Màrqueting

El nom escollit per a l'eina és "RPG Creator" i s'ha establert un preu de 25€, que es podrà incrementar a mesura que s'afegeixin mòduls o es millorin els 2 actuals.

A continuació es mostren les imatges de màrqueting finals per a l'Asset Store.

Figura 46 Imatge de tipus icona per a l'Asset Store. Font: Aylin Torunlar

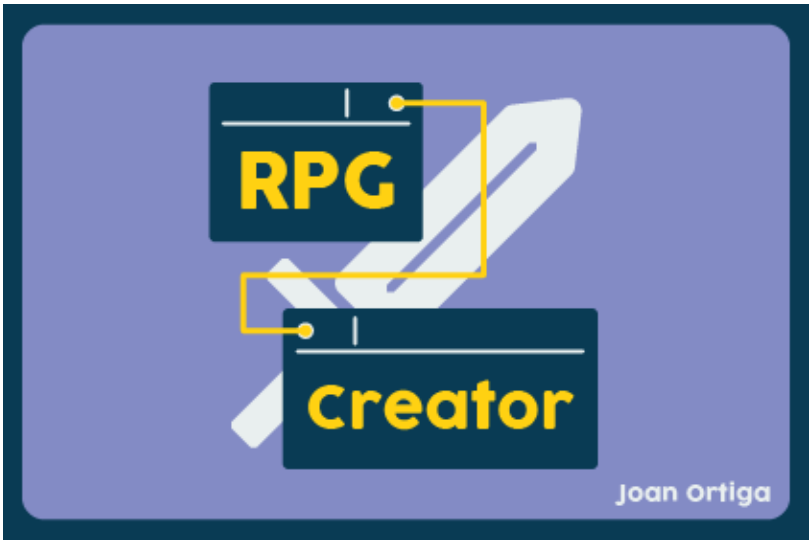

Figura 47 Imatge de tipus *targeta* per a l'Asset Store. Font: Aylin Torunlar

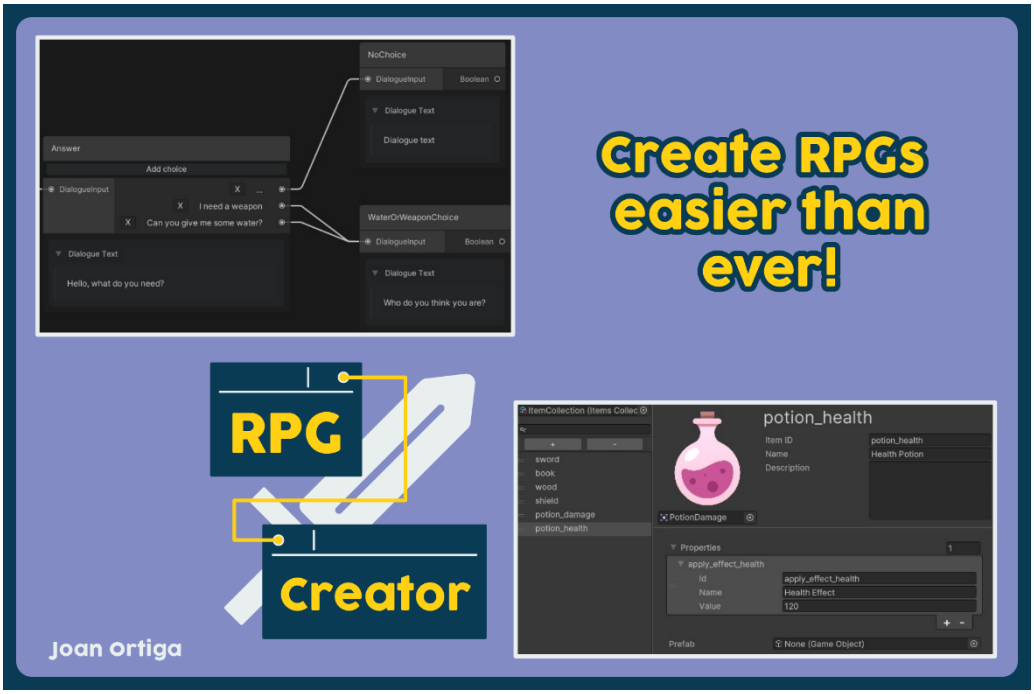

Figura 48 Imatge de tipus portada per a l'Asset Store. Font: Aylin Torunlar

Figura 49 Imatge de les xarxes socials per a l'Asset Store. Font: Aylin Torunlar

7. Conclusions

Com hem pogut comprovar als resultats, la gran part de les característiques que es van planificar han pogut ser incloses en el projecte. Per tant, es pot concloure que l'abast del projecte ha estat ben mesurat, tot i que no a la perfecció. De tota manera, hi ha moltes més característiques que ja no van passar el filtre inicial de ser considerades en el desenvolupament de l'eina per estimació de temps, però que són característiques que aporten valor a l'eina.

L'eina ha complert amb els seus objectius principals, i molts secundaris. Tot i això, elements pel mòdul de diàlegs com localització, personatges i condicions, o pel mòdul d'ítems com esdeveniments i propietats aleatòries, serien grans afegits que per qüestió de temps no s'han pogut implementar, però que són un bon punt de millora.

El cronograma s'ha complert mitjanament, on ha fallat principalment en alguns moments la teoria i en altres moments el desenvolupament de la pràctica. En especial i l'error principal ha estat no disposar de l'eina a temps per a què la revisió de l'Unity Asset Store que tarda 30 dies laborables, quedi aprovada. Això ha impedit afegir informació dels primers dies de la publicació a la memòria. Altrament, la falta del temps ha impedit poder donar l'eina a desenvolupadors de videojocs i més en concret, desenvolupadors d'Unity i videojocs RPG per poder tenir feedback de l'ús de l'eina i comprovar que tot funciona correctament. Haver pogut fer proves amb potencials usuaris de l'eina hauria ajudat a veure possibles millores d'experiència d'usuari i necessitats importants que no han estat incloses.

Respecte al mòdul d'ítems, tot i ser una eina relativament senzilla, aporta molts dels elements necessaris per al desenvolupament d'ítems en un videojoc, donant les eines necessàries al desenvolupador per ampliar les funcionalitats amb facilitat al codi. Visualment, ha quedat acurada, mostrant la informació de forma clara. La inclusió de característiques com poder reordenar el llistat d'ítems, el guardat instantani o el fer/desfer, aporten avantatges respecte a les eines referents com RPG Builder, que no compta amb totes aquestes característiques.

Respecte al mòdul de diàlegs, ha mancat temps per incloure funcionalitats que tot i que no són necessàries per al funcionament del mòdul, sí que afegien un valor rellevant, com per exemple la possibilitat d'escollir la ruta on es guarden els arxius del diàleg. Visualment, és una eina còmoda d'utilitzar i organitzar, i aporta les funcionalitats necessàries per crear diàlegs. Aconsegueix diferenciar-se de RPG Builder amb millores visuals, i característiques com el minimapa, i també es desmarca d'ORK Framework, que no té mòdul de diàlegs.

Al principi del desenvolupament del projecte es va estudiar l'ús de les tecnologies UI Toolkit o IMGUI per fer l'eina. Finalment, es va decidir usar UI Toolkit. Malgrat que al principi el desenvolupament s'ha fet més lentament, ja que és una tecnologia en la qual no s'havia treballat abans i, per tant, no es tenia experiència. Al final el ritme de treball ha estat molt més alt de com hauria estat amb IMGUI, aportant a més els avantatges de suportar futures versions, donar facilitats al desenvolupament i millorar l'aspecte visual de forma ràpida i fàcil.

La principal diferenciació de l'eina amb eines de la competència és que està molt pensada perquè els usuaris puguin personalitzar-la, i afegir funcionalitats. És relativament senzill incorporar novetats, tant a la interfície com el software.

En conclusió, el projecte ha aconseguit els objectius inicials i, malgrat que el desenvolupament de l'eina ha sofert alguns retrassos i no s'hagi pogut publicar a temps a l'Unity Asset Store, té gairebé tot el contingut que es volia.

8. Referències

Adams, E. (2014). *Fundamentals of game design.* Pearson Education.

Arcity Software. (2014). *Articy*. Recollit de https://www.articy.com/en/

Armstrong, W., & Ewing, P. (28 / 6 / 2019). *Do You Copy? Dialog System and Tools in Firewatch*. Recollit de Game Developers Conference - Youtube: https://www.youtube.com/watch?v=wj-2vbiyHnI

Bandai Namco. (21 / 04 / 2022). *Elden Ring*. Recollit de Bandai Namco: https://es.bandainamcoent.eu/elden-ring/elden-ring

Blink. (01 / 06 / 2022). *Items*. Recollit de RPG Builder Doc: https://blink.developerhub.io/v1.0/rpg-builder/items

Cambridge. (2011). *Cambridge Business English Dictionary.* Cambridge University Press. Recollit de Cambridge Dictionary.

Cambridge. (2022). *Backend definition*. Recollit de Cambride Dictionary: https://dictionary.cambridge.org/es/diccionario/ingles/back-end

Cambridge. (2022). *Frontend definition*. Recollit de Cambridge Dictionary: https://dictionary.cambridge.org/es/diccionario/ingles/front-end

Cambridge. (2022). *Plug-in definition*. Recollit de Cambride Dictionary: https://dictionary.cambridge.org/es/diccionario/ingles/plug-in

Cambridge. (2022). *Software definition*. Recollit de Cambridge Dictionary: https://dictionary.cambridge.org/es/diccionario/ingles/software

Chambers, J. M. (2014). Object-Oriented Programming, Functional Programming and R. *Statistical Science*, 167-180. Recollit de Statistical Science: file:///C:/Users/joano/Downloads/13-STS452.pdf

Cornut, O. (11 / 4 / 2014). *Dear ImGui*. Recollit de Github: https://github.com/ocornut/imgui

Craighead, J., Burke, J., & Murphy, R. (September / 2008). Using the unity game engine to develop sarge: a case study. *Proceedings of the 2008 Simulation*

Workshop at the International Conference on Intelligent Robots and Systems.

CSS Working Group. (1996).

Daglow, D. L. (1988). The Changing Role of Computer. *Computer Gaming World*, 18, 42. Recollit de https://www.cgwmuseum.org/galleries/issues/cgw_50.pdf

Dan Sumaili, S. v. (6 / Juny / 2019). *Guerrilla Games*. Recollit de Games Developer Conference - Youtube: https://www.youtube.com/watch?v=KRJkBxKv1VM

Doucet, L., & Pecorella, A. (2 / 9 / 2021). *Game engines on Steam: The definitive breakdown*. Recollit de Game Developer: https://www.gamedeveloper.com/business/game-engines-on-steam-the-definitive-breakdown

Gaming is Love e.U. (11 / 04 / 2022). *Inventory System Overview*. Recollit de Ork Framework: http://orkframework.com/guide/documentation/inventory/inventory-system-overview/

Gaming is Love e.U. (01 / 04 / 2022). *Ork Framework Documentation*. Recollit de Ork Framework: http://orkframework.com/features/

GamingIsLove. (31 / Gener / 2022). Recollit de http://orkframework.com/ork-3-beta/ - https://assetstore.unity.com/packages/tools/game-toolkits/rpg-editor-ork-framework-3-209685

Hemmendinger, D. (3 / 3 / 2022). *Object-oriented programming*. Recollit de Britannica: https://www.britannica.com/technology/object-oriented-programming

Hosch, W. L. (31 / 7 / 2019). *Role-playing video game*. Recollit de Encyclopedia Britannica: https://www.britannica.com/topic/role-playing-video-game

Hutong Games LLC. (2011). *PlayMaker*. Recollit de https://hutonggames.com

Irwin, K. (04 / 01 / 2022). *Valve's Steam Store breaks all-time highs with 28 million users*. Recollit de Input: https://www.inputmag.com/gaming/valves-steam-store-breaks-all-time-highs-with-28-million-users

Kratochvíl, M. (2014). *The Educational Contribution of RPG Video Games:.*

Liapis, A. (2013). *Sentient Sketchbook, Computer-Assisted Game Level Authoring*. Recollit de Sentient Sketchbook: http://www.sentientsketchbook.com/index.php

Lightbown, D. (9 / Abril / 2021). *Ubisoft*. Recollit de Game Developer Conference - Youtube: https://www.youtube.com/watch?v=Zp4tJQJKIPs

Mäyrä, F. (2017). *Dialogue and interaction in role-playing games.* Dialogue across Media. Recollit de https://books.google.com.tr/books?id=9T8DDgAAQBAJ&lpg=PA271&ots=bzWZFgXtnx&dq=dialogues%20role%20playing%20games&lr&hl=es&pg=PA271#v=onepage&q=dialogues%20role%20playing%20games&f=false

Metacritic. (21 / 04 / 2022). *Best Role Playing Games of all Time*. Recollit de Metacritic: https://www.metacritic.com/browse/games/genre/metascore/role-playing/all?view=detailed

Microsoft. (11 / 4 / 2001). *Microsoft's Anders Hejlsberg Receives Prestigious Excellence in Programming Award*. Recollit de news.microsoft.com: https://news.microsoft.com/2001/04/11/microsofts-anders-hejlsberg-receives-prestigious-excellence-in-programming-award/

Microsoft. (18 / 03 / 2022). *A tour of the C# language*. Recollit de docs.microsoft.com: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

Nagel, C. (2018). *Professional C# 7 and. Net Core 2.0.* John Wiley & Sons.

Neil, K. (5 / 7 / 2019). *Game Design Tools: For When Spreadsheets and Flowcharts Aren't Enough*. Recollit de Game Developers Conference - Youtube: https://www.youtube.com/watch?v=XPPtLNkVPWY

Paakkanen, V. (24 / 2 / 2021). *Workflow Driven Tools Design*. Recollit de Game Developers Conference - Youtube: https://www.youtube.com/watch?v=kAfb0yx07Po

Paul, P. S., Goon, S., & Bhattacharya, A. (2012). History and comparative study of modern game engines. *International Journal of Advanced Computed and Mathematical Sciences*, 245-249.

SimilarWeb. (04 / 2022). *Unity Asset Store traffic*. Recollit de similarweb.com: https://www.similarweb.com/website/assetstore.unity.com/#traffic

Stewart, J. (21 / 10 / 2020). *Build Great Tools: Workflow Guidelines from Vicarious Visions*. Recollit de Game Developer Conference - Youtube: https://www.youtube.com/watch?v=XUjAs92UUP4

StoryAge. (06 / 2022). *Rpg Core*. Recollit de Unreal Engine Marketplace: https://www.unrealengine.com/marketplace/en-US/product/rpg-core

Taylor, B. (28 / Abril / 2020). *Playdead's*. Recollit de Game Developer Conference - Youtube: https://www.youtube.com/watch?v=_mbOM06A5sA

Unity. (sense data). Recollit de https://unity.com/

Unity. (04 / 6 / 2019). *Creating basic editor tools*. Recollit de learn.unity.com: https://learn.unity.com/tutorial/creating-basic-editor-tools#

Unity. (27 / 11 / 2019). *UIElements roadmap update*. Recollit de forum.unity.com: https://forum.unity.com/threads/uielements-roadmap-update.784388/

Unity. (2021). *Comparison of UI systems in Unity*. Recollit de docs.unity3d.com: https://docs.unity3d.com/2020.2/Documentation/Manual/UI-system-compare.html

Unity. (2021). *Create a Custom Inspector*. Recollit de Unity docs: https://docs.unity3d.com/Manual/UIE-HowTo-CreateCustomInspector.html

Unity. (2021). *Extending the Editor*. Recollit de docs.unity3d.com: https://docs.unity3d.com/Manual/ExtendingTheEditor.html

Unity. (2021). *The Inspector window*. Recollit de Unity docs: https://docs.unity3d.com/Manual/UsingTheInspector.html

Unity. (2022). *Assembly Definition*. Recollit de Unity Documentation: https://docs.unity3d.com/Manual/ScriptCompilationAssemblyDefinitionFiles.html

Unity. (2022). *Asset Workflow*. Recollit de Unity documentation: https://docs.unity3d.com/Manual/AssetWorkflow.html

Unity. (2022). *Glossary*. Recollit de Unity Documentation: https://docs.unity3d.com/Manual/Glossary.html#Scripts

Unity. (27 / 05 / 2022). *Handle events*. Recollit de docs.unity3d.com: https://docs.unity3d.com/Manual/UIE-Events-Handling.html

Unity. (2022). *Script serialization*. Recollit de Unity Documentation: https://docs.unity3d.com/Manual/script-Serialization.html

Unity. (2022). *Serialized Object*. Recollit de Unity Documentation: https://docs.unity3d.com/ScriptReference/SerializedObject.html

Unity. (27 / 5 / 2022). *SerializedObject data binding*. Recollit de docs.unity3d.com: https://docs.unity3d.com/Manual/UIE-Binding.html

Unity. (27 / 05 / 2022). *Style UI with USS*. Recollit de Unity Documentation: https://docs.unity3d.com/Manual/UIE-USS.html

Unity. (2022). *Unity Asset Store*. Recollit de https://assetstore.unity.com

Unity. (07 / 05 / 2022). *USS custom properties (variables)*. Recollit de Unity Documentation: https://docs.unity3d.com/Manual/UIE-USS-CustomProperties.html

9. Annex

Tots els arxius de l'annex es troben a la següent carpeta de Google Drive:

Ortiga_Balcells_DesenvolupamentEinaRPGUnity_TFG
https:/drive.google.com/drive/folders/15lMRxktiZo8H8295vC_8jY6NODEfMTqd?usp=sharing

9.1 Codi font

El codi font es pot veure instal·lant l'eina dins Unity, o bé al següent enllaç: *Ortiga_Balcells_DesenvolupamentEinaRPGUnity_TFG/CodiFont*

9.2 Instruccions d'ús

Per usar i instal·lar l'eina, es necessita tenir el motor gràfic Unity instal·lat (versió 2021.3 o més recent). Es crea un projecte d'Unity vuit, i un cop dins del projecte, es fa doble clic a l'arxiu RPGCreator.unitypackage

L'instalador es troba al següent enllaç:
Ortiga_Balcells_DesenvolupamentEinaRPGUnity_TFG/RPGCreator.unitypackage

Les instruccions d'ús completes (en anglès) es poden trobar al següent enllaç: *Ortiga_Balcells_DesenvolupamentEinaRPGUnity_TFG/RPGCreatorDocumentation.pdf*

9.3 Vídeo de la simulació final

El vídeo demostratiu es pot trobar al següent enllaç:
Ortiga_Balcells_DesenvolupamentEinaRPGUnity_TFG/Video_Demostratiu.mp4

9.4 Material de tercers

Per l'elaboració de l'eina, s'han utilitzat els següents plugins:

- Unity3D-Class-Type-References (Rotorz Limited, 2018).

  Enllaç: *https://github.com/rotorz/unity3d-class-type-reference*

- SerializableDictionary (Mathieu Le Ber, 2021).

  Enllaç: *https://github.com/azixMcAze/Unity-SerializableDictionary/blob/master/Assets/SerializableDictionary/SerializableDictionary.cs*