PRÀCTICA OPCIONAL

Disseny d'unes Gameplay Foundations

MONOGAME

Motors de Jocs

17 de març de 2023

Carles Rigat Almansa

TecnoCampus Escola Superior Politècnica

Centre adscrit a la

upf. Universitat Pompeu Fabra *Barcelona*

Índex

Per començar, es comprova la instal·lació de l'SDK de .NET, i es segueixen els passos de l'enunciat de la pràctica per a crear un primer projecte de Monogame:

1) Instalar Monogame content builder editor. Executar en cmd el comandament. El pas c es opcional.
   a) `dotnet tool install --global dotnet-mgcb-editor`
   b) `mgcb-editor --register`
   c) `dotnet new --install MonoGame.Templates.CSharp`
2) Executar: `dotnet new <TemplateID> -o <ProjectName>` veure per templates ids: Monogame .Net CLI
3) Canviar el directori del cmd amb el comandament:
   a) cd <ProjectName>
4) Instalar Monogame Framework al directori:
   a) `dotnet add package MonoGame.Framework.<PlatformId>`
5) Testejar que la solució estigui bé executant el programa:
   a) `dotnet run Program.cs`

Un cop acabat, es procedeix a seguir tutorials per a executar un primer joc a Monogame, abans de decidir la *gameplay foundation* a implementar:

https://docs.monogame.net/articles/getting_started/0_getting_started.html

https://medium.com/learning-c-by-developing-games/getting-started-with-c-monogame-in-vs-code-2c26c7f198c2

Amb MonoGame funcionant, es crea un projecte de prova, s'hi entra a dins i s'hi instal·la el propi MonoGame:

```
PS D:\TECNOCAMPUS\4_QUART\2n T\MOTORS DE JOCS\PRACTIQUES\POpcional\repo\p0_monogame_gameplay_foundations\po_crigat_monogame> dotnet new mgdesktopgl -o ProjecteProva
The template "MonoGame Cross-Platform Desktop Application (OpenGL)" was created successfully.

An update for template package 'MonoGame.Templates.CSharp::3.8.0.1641' is available.
To update the package use:
   dotnet new install MonoGame.Templates.CSharp::3.8.1.303

PS D:\TECNOCAMPUS\4_QUART\2n T\MOTORS DE JOCS\PRACTIQUES\POpcional\repo\p0_monogame_gameplay_foundations\po_crigat_monogame> cd ProjecteProva
PS D:\TECNOCAMPUS\4_QUART\2n T\MOTORS DE JOCS\PRACTIQUES\POpcional\repo\p0_monogame_gameplay_foundations\po_crigat_monogame\ProjecteProva> dotnet add package MonoGame.Framework.DesktopGL --version 3.8.0.1641
>>
  Determining projects to restore...
  Writing C:\Users\carle\AppData\Local\Temp\tmpBB3E.tmp
info : X.509 certificate chain validation will use the default trust store selected by .NET.
info : X.509 certificate chain validation will use the default trust store selected by .NET.
info : Adding PackageReference for package 'MonoGame.Framework.DesktopGL' into project 'D:\TECNOCAMPUS\4_QUART\2n T\MOTORS DE JOCS\PRACTIQUES\POpcional\repo\p0_monogame_gameplay_foundations\po_crigat_monogame\ProjecteProva\ProjecteProva.csproj'.
```

1. Definició del gènere objectiu de les eines

En primer lloc, s'escull una Gameplay Foundation a implementar, d'entre les següents:

1. *Món de joc*
2. *Models d'objecte en runtime*
3. *Manipulació dels objectes de joc*
4. *Esdeveniments i missatges*
5. *Scripting*

En aquest punt, s'escull tractar la Gameplay Foundation dels ***Esdeveniments i missatges***.

1.1 Identificació de les necessitats prioritàries

Cal trobar una forma de comunicar les diverses classes del projecte per a notificarles de canvis en el sistema i proveïr-los la informació necessària per a realitzar certes accions.

A més, a ser possible, la solució hauria de ser ràpida (per exemple, coneguda pel compilador i que així es possibiliti una precàrrega) i flexible (per tal de reutilitzar codi o evitar de refer manualment parts del sistema cada cop que en calgui una ampliació no prevista).

1.2 Definició de les eines per a solventar les necessitats

A continuació, es mostren principals solucions observades a classes de teoria són les següents:

"

a) *Podem definir una **funció virtual a una classe comuna** (GameObject).*
- *Quan es detecti l'esdeveniment, es cridarà a la funció de tots els GOs*
- *Els GOs que hagin sobreescrit la implementació, podran realitzar l'acció conseqüent que vulguin.*

Problemes:

- *Lent, el compilador no sap quina implementació es cridarà. Només es coneix en execució.*
- *Poc flexible.*
- *Els GOs no tenen el poder de registrar-se i desregistrar-se a certs esdeveniments.*

b) *Alguns llenguatges com C# ens permeten **vincular funcions dinàmicament** (delegates).*
- *Quan es detecti l'esdeveniment, es cridaran a totes les funcions vinculades/registrades.*
- *Els GOs que s'hagin registrat a l'esdeveniment, podran realitzar l'acció conseqüent que vulguin.*

Problemes:

- *"Lent", el compilador no sap quina implementació es cridarà. Només es coneix en execució.*

c) *Podem **encapsular el missatge** (event, command, message segons motor) en un objecte.*

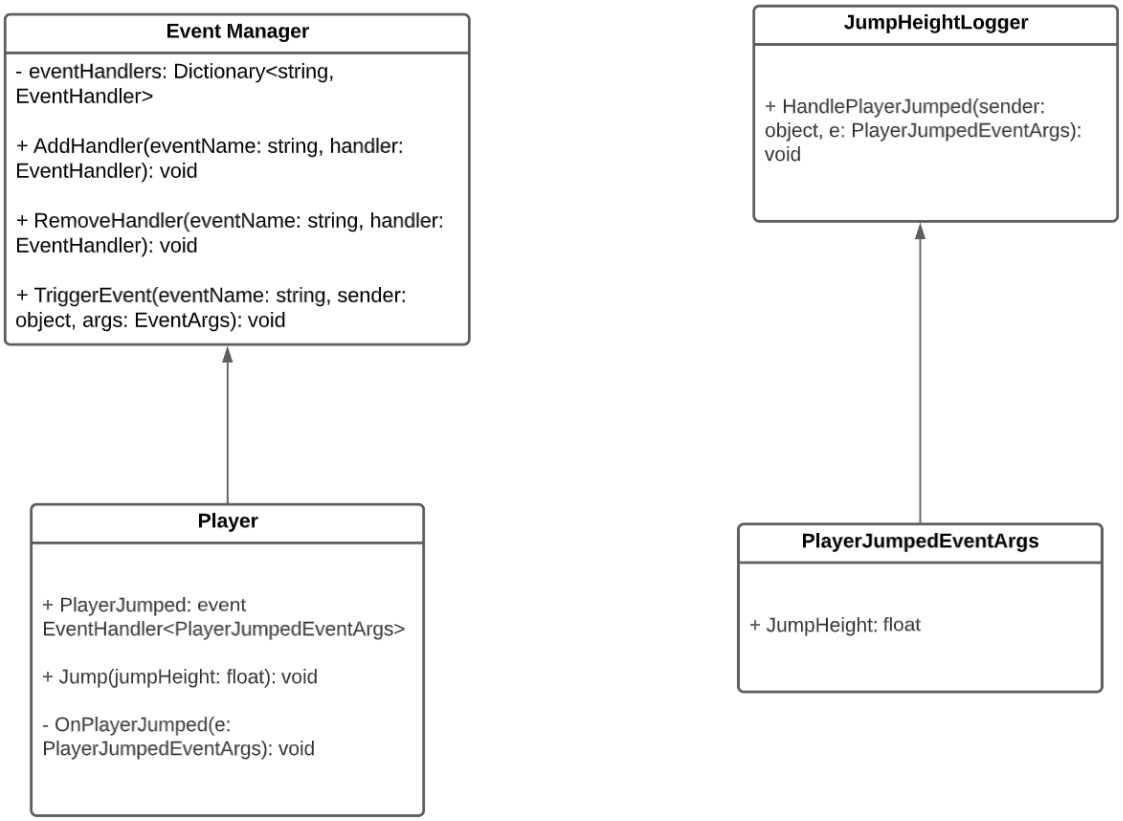
- *Només una funció per a cada GO s'encarregarà de gestionar (handle) tots els esdeveniments.*
- *Faciliten la gestió: cues, ordre, desfer...*
- *Blind forwarding. Un objecte pot derivar-ne la gestió a un altre sense conèixer l'esdeveniment.*
- *Qui emet l'esdeveniment ha de mantenir el llistat de tots els objectes interessats en rebre aquest objecte quan passi l'esdeveniment. "*

(Extret de: Albert Carrillo, Ricard Perea i Ferran Cantariño; Motors de Jocs - Gameplay Foundations - TecnoCampus Mataró, 2023)

Per tant, tot i que MonoGame s'implementa en C# i es podria escollir la segona opció, pels avantatges que comporta la tercera s'escull de tractar d'**encapsular els missatges en objectes**.

2. Diagrama UML simplificat de la solució

A continuació es mostra el diagrama UML dissenyat per a la solució:



Explicació simplificada del diagrama:

"

En aquest diagrama, tenim tres classes principals:

1. ***EventManager**: aquesta classe gestiona el sistema d'esdeveniments i proporciona mètodes per afegir i eliminar controladors d'esdeveniments, així com per activar esdeveniments. Conté un diccionari per fer un seguiment dels gestors d'esdeveniments.*

2. ***Player**: aquesta classe representa l'objecte del jugador del joc i genera l'esdeveniment "PlayerJumped" quan el jugador salta. Conté un esdeveniment anomenat PlayerJumped que és un tipus de EventHandler<PlayerJumpedEventArgs>.*

3. ***JumpHeightLogger**: aquesta classe representa un controlador d'esdeveniments que registra l'alçada de salt del jugador a la consola. Conté un mètode anomenat HandlePlayerJumped que incorpora un objecte de tipus object i un objecte d'arguments d'esdeveniment de tipus PlayerJumpedEventArgs.*

*A més, tenim la classe **PlayerJumpedEventArgs**, que representa les dades associades a l'esdeveniment "PlayerJumped". Conté una única propietat JumpHeight de tipus float.*

Les fletxes del diagrama representen les relacions entre les classes. La classe EventManager la utilitza la classe Player per activar esdeveniments i la classe JumpHeightLogger per registrar i anul·lar el registre dels controladors d'esdeveniments. La classe Player genera l'esdeveniment PlayerJumped, que és gestionat per la classe JumpHeightLogger.

" (base de la explicació obtinguda de diverses recerques sobre MonoGame i el funcionament dels EventHandlers)

2.1 Correcció del diagrama en l'estàndard UML

Nota: s'utilitzen els models de l'eina LucidChart per a crear l'UML amb les seves senyals habituals.

2.2 Disseny de la solució

Tot i no requerir de patrons específics, la solució es podria complementar fent ús de diversos patrons de disseny:

"

1. ***Patró Observador**: la classe Player utilitza un esdeveniment per notificar d'altres parts del sistema quan el jugador salta. L'**esdeveniment** és una forma del patró observador, on la classe **Player** és el **subjecte** i la classe **JumpHeightLogger** és un **observador**.*

2. ***Patró Singleton**: la classe EventManager es podria implementar com a singleton per garantir que només hi hagi una instància de la classe al sistema. Això podria ser útil si calgués assegurar-se que només hi ha un gestor d'esdeveniments al sistema en un moment donat.*

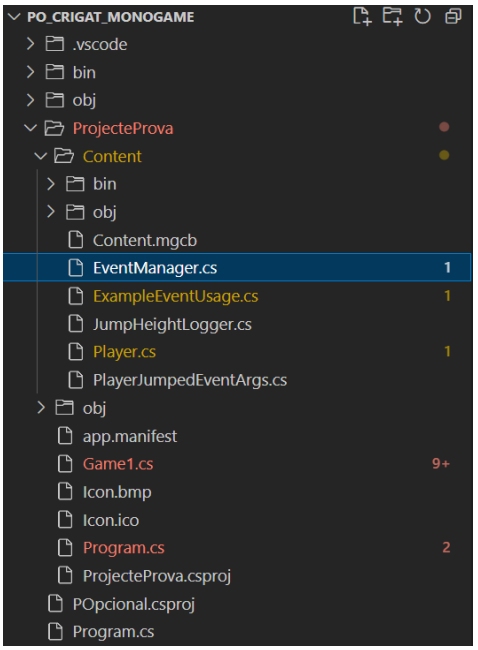
3. ***Patró Command**: si calgués ampliar el sistema d'esdeveniments per permetre la funcionalitat de desfer/refer, es pot utilitzar el patró Command per encapsular els canvis d'estat fets pels controladors d'esdeveniments. Cada controlador d'esdeveniments es pot implementar com una ordre que es pot executar i desfer, permetent desfer una sèrie d'esdeveniments si fos necessari.*

4. ***Patró Factory**: si es tinguessin diversos tipus d'esdeveniments a gestionar de diverses maneres, el patró de fàbrica podria crear diferents tipus de controladors d'esdeveniments en funció del tipus d'esdeveniment que s'estiguin gestionant.*

Això permetria encapsular la creació de gestors d'esdeveniments i facilitar l'addició de nous tipus de gestors d'esdeveniments en el futur.

" *(base de la explicació obtinguda de diverses recerques sobre MonoGame i el funcionament dels EventHandlers)*

3. Implementació de la solució



Es crea una estructura general d'utilitats amb les classes explicades en el diagrama UML i una classe *ExampleEventUsage* que prova la solució.

El codi font de cada classe pot trobar-se al repositori de git (https://github.com/Rigat13/p0_monogame_gameplay_foundations), i s'explica breument en un vídeo, també present com a enllaç en el README del repositori.

4. Testeig de la solució

A la classe *ExampleEventUsage* es prova la solució:

```
2   public class ExampleEventUsage
3   {
4       // Create a delegate
        1 reference
5       public delegate void ExampleEventHandler(object sender, EventArgs e);
6       // Create an event based on that delegate
        2 references
7       public event ExampleEventHandler ExampleEvent;
8       // Create a method to raise the event
        1 reference
9       protected virtual void OnExampleEvent(EventArgs e)
10      {
11          if (ExampleEvent != null)
12              ExampleEvent(this, e);
13      }
14      // Call that method whenever you want to raise the event
        0 references
15      public void RaiseExampleEvent()
16      {
17          OnExampleEvent(EventArgs.Empty);
18      }
19
20
21
        0 references
22      public static void Main(string[] args)
23      {
24          // Example usage
25          EventManager eventManager = new EventManager();
26          Player player = new Player();
27          JumpHeightLogger logger = new JumpHeightLogger();
28
29          // Register the event handler
30          eventManager.AddHandler("PlayerJumped", logger.HandlePlayerJumped);
31
32          // Trigger the event
33          player.Jump(10.0f);
34          eventManager.TriggerEvent("PlayerJumped", player, new PlayerJumpedEventArgs { JumpHeight = 10.0f });
35
36          // Unregister the event handler
37          eventManager.RemoveHandler("PlayerJumped", logger.HandlePlayerJumped);
38      }
39  }
```

En primer lloc, es proven els esdeveniments fent ús de delegats i amb funcions pròpies com el *RaiseExampleEvent*.

En segon lloc, en aquest cas dins d'un main, es fa ús de les classes presents al diagrama UML per tal de crear interaccions entre el sistema. El seu funcionament s'explica breument en el vídeo present al repositori de Git.

5. Conclusió final

MonoGame és un framework flexible que permet als desenvolupadors de jocs de customitzar les bases de les Gameplay Foundations dels seus jocs sense haver de crear un motor des de zero, amb funcionalitats bàsiques ja implementades.

Amb aquest treball s'ha pogut tastar per sobre la manera de treballar que requereix, força més aproximada a la del software convencional en primera instància que els motors tractats a classe (com Defold o Unreal). Això es comença a percebre degut a l'absència d'interfície gràfica, però també, per exemple, de certs sistemes d'esdeveniments ja presents en motors.

Cada projecte té els seus requeriments. Així doncs, pels que vulguin ser més flexibles, i tampoc hagin de "reinventar la roda" al no necessitar de funcionalitats avançades dels motors més coneguts, MonoGame podria ser una bona alternativa.

Per últim, s'ha trobat interessant de tractar la Gameplay Foundation dels Esdeveniments i Missatges, tot i que es reconeix que l'abast del treball s'ha hagut de retallar degut a les poques hores invertides.

S'escollí prioritzar les altres pràctiques i deixar aquesta opcional per a més endavant, fins al punt que no s'ha tingut temps per a fer un projecte complet.

Igualment, es volia entregar quelcom i investigar una pinzellada més de les Gameplay Foundations i del que pot oferir MonoGame.

Moltes gràcies per l'atenció.