République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

MÉMOIRE DE MASTER PROFESSIONNEL

En
Informatique

Option
Génie Logiciel

Thème

Conception et réalisation d'un jeu mobile 3D sous le moteur de jeux Unity 3D

Présenté par : M^le^. Yasmine SAYAD

Évalué par :

Examinateur	**M. ATMANI M.**	**Maître de Conf. B**	**U. A/Mira Béjaia.**
Examinateur	**M. DJEBARI N.**	**Maître de Conf. B**	**U. A/Mira Béjaia.**
Encadreur	**M. SIDER A.**	**Maître de Conf. A**	**U. A/Mira Béjaia.**

Béjaia, Septembre 2020.

⁂ ⁂ ⁂ *Remerciements* ⁂ ⁂ ⁂

Louanges à Allah, le Miséricordieux, sans Lui rien de tout cela n'aurait pu être.

Je tiens à exprimer ma sincère reconnaissance et mes vifs remerciements à tous ceux qui ont contribué de prés ou de loin à l'élaboration de ce travail et avant tout ma famille qui n'a jamais cessé de m'encourager.

En tout premier lieu, je tiens à remercier grandement mon encadrant de l'université de Béjaia je cite **M. Abderrahmane SIDER** *d'avoir accepté de m'assister dans la réalisation de ce mémoire et de m'avoir donné l'occasion de m'épanouir dans le domaine ...*

Ensuite aux membres de jury qui ont eu l'amabilité d'accepter d'évaluer ce travail. Qu'ils trouvent ici l'expression de ma reconnaissance.

Enfin, j'adresse mes plus sincères remerciements à tous mes proches et amis(es) qui m'ont toujours soutenu et encouragé au cours de la réalisation de ce mémoire.

⁂ *Dédicaces* ⁂

A mes très chers parents qui ont toujours étaient là pour moi, et qui m'ont donné un magnifique modèle de labeur et de persévérance ;

A mes très chers frères ;

Et à M. BENHADDAD pour son soutien, son aide et ses encouragements ;

Je vous dois ce que je suis aujourd'hui grâce à votre amour, à votre patience et vos innombrables sacrifices.

Que ce modeste travail, soit pour vous une petite compensation et reconnaissance envers ce que vous avez fait d'incroyable pour moi.

Que Allah, le tout puissant, vous préserve et vous procure santé et longue vie afin que je puisse à mon tour vous combler.

M^le ***Yasmine SAYAD***

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

2D : Deux Dimensions.
3D : Trois Dimensions.
3DS : 3 Dimensional Screen.
3 DSMAX : Troisième Dimension Studio Max.
APK : Android Package Kit.
C# : C sharp.
EDI : Environnement de Développement Integré.
FBX : FilmBox.
FPS : First Person Shoot.
GB : GigaByte.
HTML : Hyper Text Markup Language.
iOS : iPhone Operating System.
ipa : iOS App Store Package.
JDK : Java Developement Kit.
MMO : Massivly Multiplayer Online.
Obj : Object.
OSX : Operating System eXtension.
PC : Personal Computer.
RAM : Random Access Memory.
RPG : Role Playing Game.
SDK : Software Package Kit.
TV : Television.
UML : Unified Modeling Language.
UP : Unified Process.
VR : Virtual Reality.
WebGL : Web Graphics Library.
WiiU : Wireless Interactive Interface Ultra

Introduction générale

Un jeu vidéo est un monde virtuel qui permet de transporter le joueur dans un autre univers. Aujourd'hui, les jeux vidéo représentent l'une des industries culturelles les plus dynamiques au monde, en 2019 elle a produit un chiffre d'affaire mondial de 120 milliards de dollars. Le segment mobile représente 45% du marché mondial des jeux.[1]

Ce travail consistera à concevoir, à réaliser et à tester un jeu mobile de réflexion pour les deux plateformes Android et iOS (iPhone Operating System) que les utilisateurs peuvent installer sur leurs téléphones et leurs tablettes afin de jouer à un jeu de reconstruction d'un puzzle d'un objet en 3D (Trois Dimensions).

Le but de ce travail est de faire connaître les monuments, leurs localisations et leurs histoires.

Pour ce qui est de la démarche de développement, notre choix s'est porté sur le processus unifié, en effet le processus unifié est un processus de développement logiciel adapté à tout type de projet.

Le langage de modélisation utilisé est UML (Unified Modeling Language), qui est une partie intégrante de la démarche UP (Unified Process). Ses diagrammes sont largement utilisés dans chaque étape de ce processus de développement.

Pour implémenter ce projet nous avons opté pour Unity 3D qui est l'un des moteurs de jeux les plus puissants sur le marché et les plus complets pour la réalisation des jeux vidéo multiplateformes, et pour concevoir nos scripts nous avons opté pour C# (C sharp) qui est un langage trés similaire au java .

Notre mémoire est subdivisé en trois principaux chapitres :

Le premier chapitre intitulé " **Unity et développement des jeux vidéo mobiles** consistera à donner une vue globale sur le développement des jeux mobiles sous le moteur de jeux Unity 3D.

Dans le deuxième chapitre "**Analyses des besoins et conception**", nous présenterons le processus de développement et le langage de modélisation utilisé, ensuite nous analyserons les besoins du future application et nous terminerons par présenter les différents diagrammes de conception de notre application.

Le dernier chapitre "**Réalisation**", sera consacré à l'implémentation de notre jeu, nous commencerons par présenter les différents outils qui ont servi au développement

de notre application, ensuite nous expliquerons le déroulement du jeu illustré par des captures de ses interfaces.

Enfin, nous clôturons notre mémoire par une conclusion dans laquelle nous résumerons notre solution et proposerons quelques perspectives futures.

Chapitre 1

Unity et développement des jeux vidéo mobiles

1.1 Introduction

Ce chapitre aborde des notions sur le développement des jeux mobiles sous Unity, nous commençons par des généralités sur les jeux vidéo, ensuite une étude comparative entre les deux moteurs de jeux Unreal Engine et Unity 3D sera effectuée afin de choisir le moteur le plus adapté à notre projet, nous définissons par la suite Unity et ses concepts clés. Enfin une partie sera dédiée pour présenter les deux plateformes iOS et Android ainsi que leurs boutiques respectives App store et Play Store.

1.2 Jeux vidéo

1.2.1 Histoire des jeux vidéo

Ralph BAER est l'inventeur de la première console de jeux vidéo, en 1951 dans la société Loral Electronics aux Etats-Unis, il eut l'idée d'intégrer un jeu au téléviseur. En 1966 en travaillant sur un téléviseur, il voulait créer une boîte de jeux que tout consommateur pourrait brancher à son téléviseur. Avec son collègue Bob Trembley ils ont conçu le prototype du premier jeu de l'histoire "Chase Game" permettant d'afficher un rectangle blanc à l'écran et de le déplacer verticalement et horizontalement. Chase Game se joue à deux joueurs uniquement, le but de chaque joueur est de pourchasser l'autre (Figure 1.1)[2].

FIGURE 1.1 – Chase Game

1.2.2 Types de jeux vidéo

Les principaux genres de jeux vidéo sont :[3]

- **Jeux d'aventure :** c'est un jeu dans lequel le joueur explore l'environnement et résout des énigmes.
- **Jeux de plate-forme :** le personnage est contrôlé par le joueur, il saute d'une plate-forme à une autre pour gagner des bonus.
- **Jeux de rôle RPG** (**R**ole **P**laying **G**ame) : le joueur incarne un ou plusieurs personnages aventuriers qui lui font vivre des aventures au travers de l'histoire.
- **Jeux de simulation :** ce sont des jeux qui permettent de représenter les comportements des engins d'une manière plus réaliste en prenant en considération les lois de la physique.
- **FPS** (**F**irst **P**erson **S**hoot) : ce type de jeux est basé sur le combat et le joueur voit l'action à travers les yeux du personnage.
- **Jeux de course :** dans les jeux de courses le joueur est placé au volant d'un véhicule.
- **Jeux de gestion :** dans ce type de jeux le joueur gère une ville ou une entreprise, ...
- **MMO** (**M**assivly **M**ultiplayer **O**nline) : sont des jeux de rôle multi-joueurs en ligne.
- **Jeux de sport :** simulation d'un sport.
- **Jeux de réflexion :** les jeux de réflexion sont basés sur la logique et la réflexion comme les jeux de puzzle.

1.2.3 Grandes entreprises de jeux

La figure ci-dessous (Figure 1.2) représente le classement des éditeurs de jeux vidéo selon le chiffre d'affaire en 2018 [4].

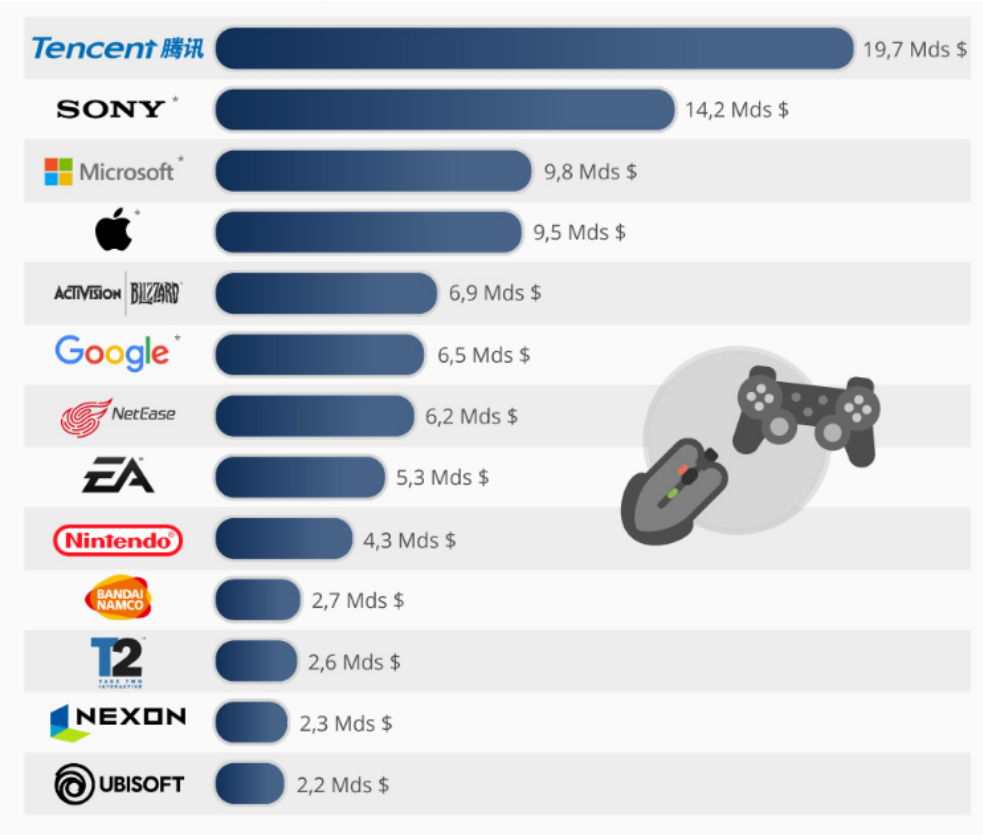


FIGURE 1.2 – Les leaders de l'industrie des jeux vidéo

1.2.4 Création des jeux vidéo

Le développement d'un jeu vidéo se fait généralement par une équipe composée de :[5]

- **Un concepteur de jeu :** il est chargé de la conception du jeu, il réalise le cahier des charges en collectant et en synthétisant les idées des différents membres de l'équipe. Dans la phase production le concepteur du jeu coordonne les actions réalisées par les différents membres du groupe, son but est de rendre le jeu plus interactif.
- **Un directeur artistique :** dans la création des jeux vidéo c'est celui qui définit l'apparence visuelle du jeu (le choix des couleurs, le logo ...), il répartit les tâches entre les infographistes du groupe et supervise leur travail.
- **Un développeur informatique :** conçoit des programmes informatiques en analysant les besoins exprimés dans le cahier des charges.
- **Un concepteur de niveaux :** est le créateur des différents niveaux du jeu, il travaille sous la responsabilité du concepteur du jeu.
- **Un infographiste/graphiste multimédia :** sa mission principale est de définir l'esthétique générale du jeu.

- **Un modeleur numérique ou modeleur 3D :** est celui qui crée les objets du jeu tels que les modèles 3D (voiture, immeuble..).
- **Un testeur :** c'est celui qui détecte les anomalies (bugs).
- **Un concepteur des sons :** il a pour mission de concevoir et de créer des sons et des musiques adaptés à l'environnement du jeu.

1.3 Étude comparative de Unity 3D et Unreal

Cette partie sera dédiée à une étude comparative entre les deux moteurs de jeux les plus populaires sur le marché de développement des jeux vidéo, soient **Unity 3D** et **UNREAL** (Figure 1.3), pour ainsi justifier notre choix par rapport à la plate-forme de développement appropriée pour la réalisation de notre projet. Basé sur des résultats de *benchmarking* [8][9][10], nous avons sélectionné les aspects les plus pertinent pour notre comparaison.

FIGURE 1.3 – Logos de Unreal Engine et Unity3D

1.3.1 Technique

Les deux moteurs prennent en charge la réalisation des jeux en 2D (Deux Dimensions) et 3D. Malgré les performances du rendu graphique chez le moteur de jeux Unity 3D mais reste que Unreal Engine offre des visuels de haute qualité.

1.3.2 Facilité d'utilisation

Grâce à son interface intuitive, Unity permet la facilité et la rapidité de la création des jeux tandis qu'Unreal Engine est plus bien complexe.

1.3.3 Développement

Unity utilise C# ou Javascript indépendamment de la plateforme finale, le langage sera interprété par Unity. Unreal Engine utilise C++, et contrairement à Unity le code est compilé. Unreal possède Blueprint un système de script visuel qui permet à l'utilisateur de créer des jeux sans taper une ligne de code.

1.3.4 Modélisation

La modélisation 3D des objets complexes se fait avec des logiciels de modélisation, Unreal Engine est compatible avec Blender, Google sketchup, Maya et 3DSMax (Trois Dimensions Studio Max), en plus de ces derniers Unity est compatible avec Cinema4D, Cheetah3D, Modo et Lightwave.

1.3.5 Communauté

La communauté francophone est bien plus à l'aise avec Unity qu'avec son concurrent, on retrouve de ce fait un large éventail de tutoriels, de forums ainsi que des ouvrages édités sur Unity 3D.

1.3.6 Plateformes compatibles

Le tableau suivant représente les différentes plateformes compatibles avec Unity et Unreal Engine :(Table 1.1)

Plateforme	**Unity**	**Unreal Engine**
Ordinateur	Windows, OS X (Operating System Extension), Linux, Windows Store Apps.	Windows, OS X, Linux.
Console	Playstation 4, Xbox One, SteamOS,WiiU, Xbox360, PsVista, 3DS (3 Dimensinal Screen).	Playstation 4, Xbox One, SteamOS,WiiU (Wireless Interactive Interface Ultra) .
Réalité virtuelle	Occulus Rift, PlayStationVR (Virtual Reality), Google Cardboard, SteamVR, Gear VR, Microsoft Hololens.	Occulus Rift.
Mobile	iOS, Android, Windows Phone, Tizen.	iOS, Android.
Navigateur	WebGL (Web Graphics Library).	HTML5 (Hyper Text Markup Language).
Téléviseurs	AndroidTV (Television), Samsung Smart TV, tvOS.	/

TABLE 1.1 – Tableau récapitulatif des plateformes compatibles avec Unity et Unreal Engine.

Nous avons noté dans cette partie les différences majeures entre Unity 3D et Unreal Engine. Nous constatons que Unity est le plus adapté à notre projet car il utilise un langage très similaire au java et une interface intuitive ce qui nous facilite la réalisation de notre projet, en outre nous utilisons des modèles 3D existants réalisés par les différents logiciels de modélisation 3D et puisque Unity est compatible avec un nombre important de logiciels de modélisation il nous donne la chance d'utiliser plusieurs modèles. Aussi Unity possède

une grande communauté francophone ce qui explique le choix des débutants francophones vers Unity. Enfin, Unreal requiert des machines à haute performance (8 GB (GigaByte) de RAM(Random Access Memory) minimum ...) ce qui n'est pas à notre disposition.

1.4 Unity 3D

1.4.1 Définition

Unity 3D est un moteur de jeux multiplateforme crée par Unity technologies, il permet la création des jeux 2D et 3D avec le choix du langage de script : C ou javascript.

Il a la particularité de proposer une licence gratuite dite "Personal", elle comprend toutes les fonctionnalités nécessaires pour la création d'un jeu vidéo mais pour un nombre de plateformes limité : Android, iOS, PC (Personal Computer) et les différents navigateurs pour les autres plateformes il faut acheter une licence.

Unity supporte les objets de format FBX (Filmbox) et OBJ (Object) [6].

1.4.2 Principe de la 3D

Dans la 3D les objets sont représentés sur 3 axes (X,Y,Z), l'axe X (l'abscisse) représente le plan horizontal, l'axe Y (l'ordonnée) représente le plan vertical, l'axe Z représente la profondeur.

Pour réaliser une application 3D il est très important de comprendre la notion de l'espace local et de l'espace global (Figure 1.4), dans les univers 3D il y a un point d'origine (0,0,0) qui est un espace global, les valeurs de position et de rotation d'un objet (son espace local il s'agit généralement de son centre) sont relatives à cet origine. Dans les relations père-fils entre les objets (les objets imbriqués), les valeurs de position et de rotation d'un objet fils sont relatives à celles de l'objet parent, donc la position du parent devient la nouvelle origine de l'objet fils [7].

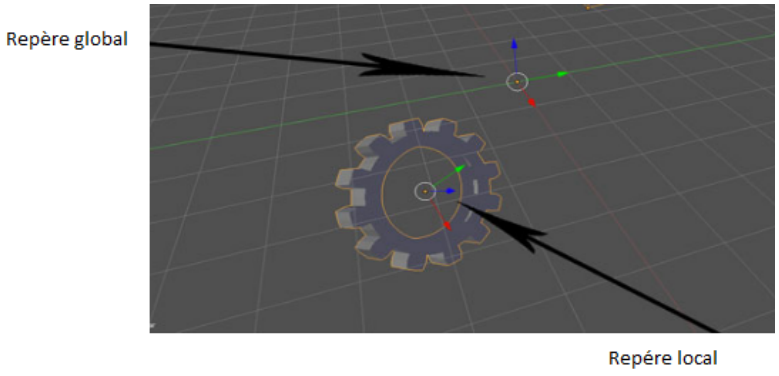


FIGURE 1.4 – Espace local et Espace global

1.4.3 Concepts de Unity

L'utilisation d'un moteur de jeux nécessite la compréhension de ses concepts, ici nous présentons les concepts clés de Unity [7][10].

1.4.3.1 Caméra

Les caméras 3D ont une forme de pyramide et un champ de vision réglable afin de donner un point de vue sur le monde 3D, elles peuvent être animées ou attachées à un objet.
Des effets de lumière et de soleil et d'autres effets peuvent être appliqués à la caméra afin de simuler la vision de l'oeil humain. Dans Unity plusieurs caméras peuvent être utilisées dans une seule scène et contrôlées par un script. Une caméra multi-cibles peut être aussi utilisée, cette caméra est dynamique elle suit le mouvement des objets cibles.

1.4.3.2 Objets (GameObject)

On appelle GameObject toute ressource utilisée dans la scène. Chaque GameObject contient un composant **TRANSFORM**(Figure 1.5) qui indique sa position, sa rotation et son échelle.

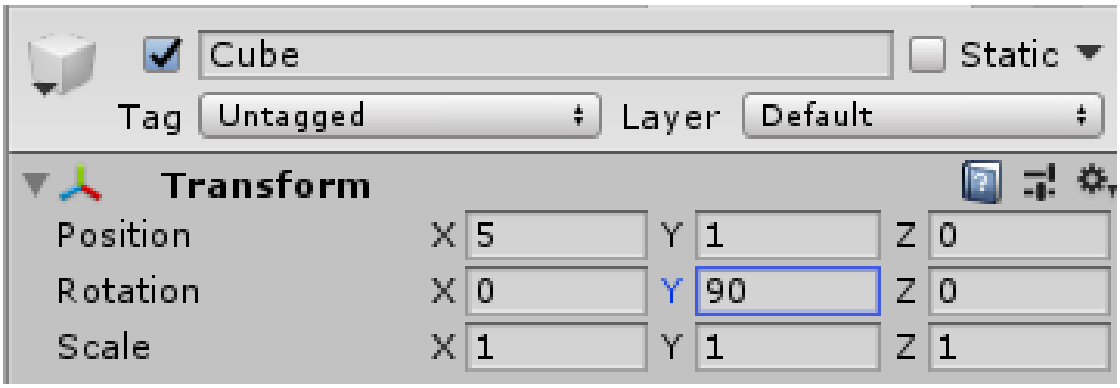


FIGURE 1.5 – Le composant transform

1.4.3.3 Lumière

L'éclairage en Unity se fait par des objets lumineux. Il existe plusieurs types de lumière en Unity :

- **Point lumineux (Point Light)** : Un point lumineux se situe dans un point dans l'espace et émet la lumière dans toutes les directions, l'intensité de la lumière émise se diminue en s'éloignant du point lumineux. Ce type de lumière se comporte comme la lumière dans le monde réel.
- **Projecteur (Spot Light)** : Un projecteur est comme un point lumineux mais il est contraint à un angle et la zone d'illumination est en forme de Cône. Ce type

de lumière illumine une petite zone de la scène, il est utilisé pour simuler les lumières artificielles telles que les lampes de poche .

- **Lumière directionnelle** : Ce type de lumière illumine toute la scène, son intensité ne diminue pas avec la distance et elle crée le même effet que celui de la lumière de soleil.
- **Lumière de zone (Area Light)** : La lumière de zone est émise dans une zone rectangulaire, elle est utilisée pour simuler l'éclairage d'une petite zone d'une manière plus réaliste comme l'éclairage intérieur d'une maison.
- **Matériaux émissifs** : Les matériaux émissifs émettent la lumière sur leur surface, ils sont utilisés pour créer des objets illuminés (éclairés à l'intérieur).
- **Lumière ambiante** : La lumière ambiante ne provient d'aucun objet source et elle est émise dans toute la scène, elle est généralement utilisée pour augmenter la luminosité globale d'une scène.

1.4.3.4 Objets (La physique des corps rigides)

Dans un moteur de jeux un objet n'est pas soumis aux lois de la physique par défaut, pour ce faire il faut lui assigner un composant Rigidbody (Figure 1.6) qui permet de créer un mouvement réaliste, il possède les propriétés suivantes : Masse, gravité, vélocité, friction.

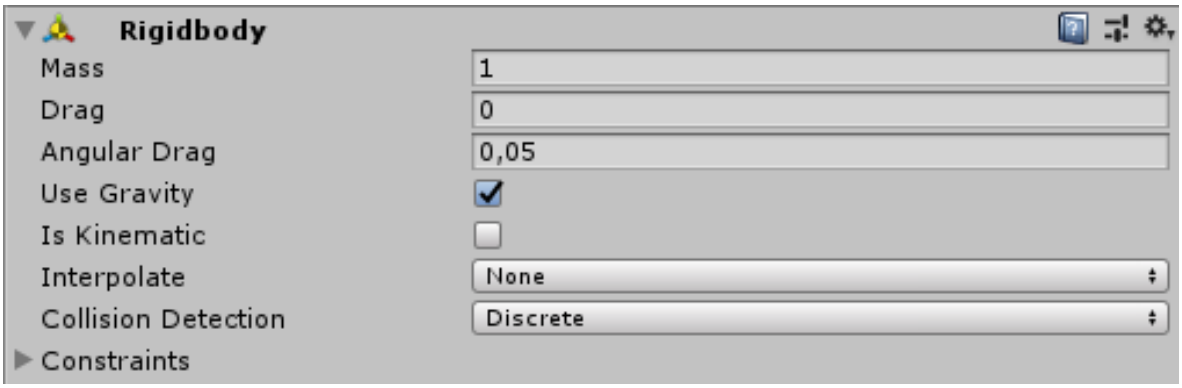


FIGURE 1.6 – Corps rigide

1.4.3.5 Matériaux et Textures

Un matériel sert à définir l'aspect visuel d'un modèle 3D avec une simple couleur ou avec l'utilisation d'une image dite "Texture". Les textures créées dans un logiciel externe comme Photoshop doivent avoir les dimensions suivantes : 128*128, 256*256, 512*512 et 1024*1024.

1.4.3.6 Éléments préfabriqués

Les éléments préfabriqués permettent des stocker un objet, ses composants et sa configuration actuelle pour le réutiliser dans les différentes parties du jeu.

1.4.3.7 Asset store

C'est une boutique en ligne qui met à disposition des ressources utilisables dans des projets commerciaux ou personnels (images, sons, modèles 3D), ces ressources peuvent être payantes ou gratuites.

1.4.3.8 Détection de collisions

La détection de collisions se fait en assignant un composant de collision invisible (collider) à un objet. Le collider entoure l'objet et signale chaque collision avec d'autres colliders. il existe plusieurs types de collider :

- **Mesh collider** qui respecte le maillage de l'objet pour une collision très réaliste mais qui est très coûteux ;
- **Box collider** pour les cubes ;
- **Terrain collider** pour les terrains ;
- **Sphere collider** pour entourer des sphères...

1.4.3.9 Script

Un script en Unity est considéré comme un composant attaché à un objet écrit en C# ou en JavaScript. Tout script Unity écrit en C# doit dériver de la classe **MonoBehaviour** qui permet l'utilisation de plusieurs fonctions notamment :

- **Start() :** Start() est appelée lorsque le script est activé avant que l'une des méthode Update soit appelée la première fois.
- **Update() :** Update() est appelée à chaque Frame.
- **OnCollisionEnter() :** OnCollisionEnter() est appelée lorsque ce Collider ou Rigidbody a commencé à toucher un autre Rigidbody ou Collider.
- **OnApplicationQuit() :** est envoyée à tous les objets du jeu avant de quitter l'application.

1.4.3.10 Balises (Tags)

Les balises en Unity sont des noms qu'on applique aux objets du jeu, elles sont utilisées comme des mots clés pour identifier les objets facilement via le code, un objet peut avoir une seule balise seulement et une balise peut être appliquée à plusieurs objets. Pour récupérer un GameObjet par tag dans le script il suffit d'utiliser la fonction **GameObjet.FindGameObjectWithTag(string tag)**, pour récupérer plusieurs GameObjects identifiés par une seule balise on écrit : **GameObject.FindGameObjectsWithTag(string tag)** (Figure 1.7).

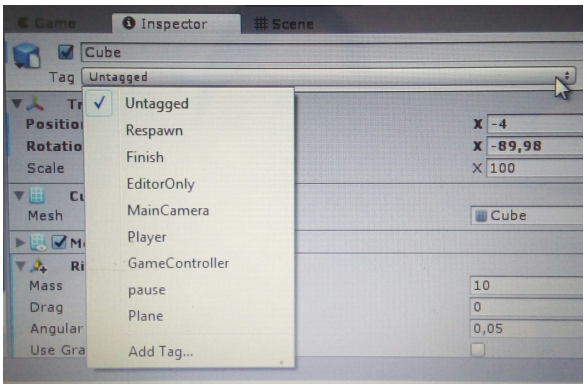


FIGURE 1.7 – Balises (Tags)

1.4.3.11 Calques (Layers)

Les calques sont similaires aux balises, ils sont utilisés pour identifier des objets et aident à indiquer leurs fonctionnalités (ils indiquent les GameObjects qui doivent être ignorés par les collisions ou ceux qui doivent être invisibles par la caméra) , ils sont généralement utilisés pour la détection de collisions (Figure 1.8).

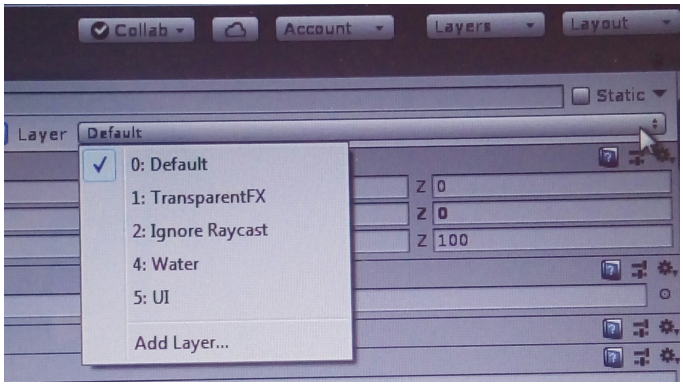


FIGURE 1.8 – Calques (Layers)

1.5 Développement mobile

Unity 3D est l'une des plateformes les plus complètes pour la création de jeux mobiles, pour notre projet nous avons choisi les deux plateformes Android et iOS pour leur popularité.

Dans cette partie nous présentons les deux plateformes Android et iOS, leurs SDK (Software Development Kit) ainsi que leurs boutiques. Nous expliquons aussi les étapes nécessaires pour générer une application iOS et Android et pour la publication sur Play Store et sur App Store.

1.5.1 Plateforme Androïd

1.5.1.1 Définition

Androïd est un système d'exploitation mobile basé sur Linux. Racheté par Google en 2005 et lancé en 2007 [12].

1.5.1.2 Play Store

Play Store (Figure 1.9) est la plateforme de téléchargement des applications pour les appareils mobiles fonctionnant sous Androïd.

FIGURE 1.9 – Logo de Play Store

1.5.1.3 Androïd SDK

Androïd SDK est le kit de développement logiciel qui permet aux développeurs de créer des applications pour les appareils de la plateforme Androïd. Le Androïd SDK est open-source, facile à installer, compatible avec tous les systèmes d'exploitation (Windows, Linux, MacOs) et il contient tous les outils nécessaires pour créer, tester et déboguer des applications Androïd [12].

1.5.1.4 Exporter un projet pour Android

Lors de son installation, Unity propose d'installer un ensemble de modules, parmi eux le module d'export Android, pour l'installer il faut cocher **Android Build Support**, sinon après l'installation d'Unity, dans **File/ Build Settings**, dans la partie **Plateform**, sélectionner **Android**, cliquer sur **Open Download loaded** pour télécharger et installer le module Android, à la fin de l'installation, relancer Unity, aller dans File/ Build Settings, sélectionner Android et cliquer sur **Switch Plateform**.

La deuxième étape consiste à installer le JDK (Java Developement Kit) et le SDK, pour ce faire, allez dans Edit/preferences/ External tools, cliquer sur le bouton **Download** du SDK, une fois téléchargé cliquer sur le bouton **Browse**, Unity le trouve automatiquement. Suivre les mêmes étapes pour installer le JDK (Figure 1.10).

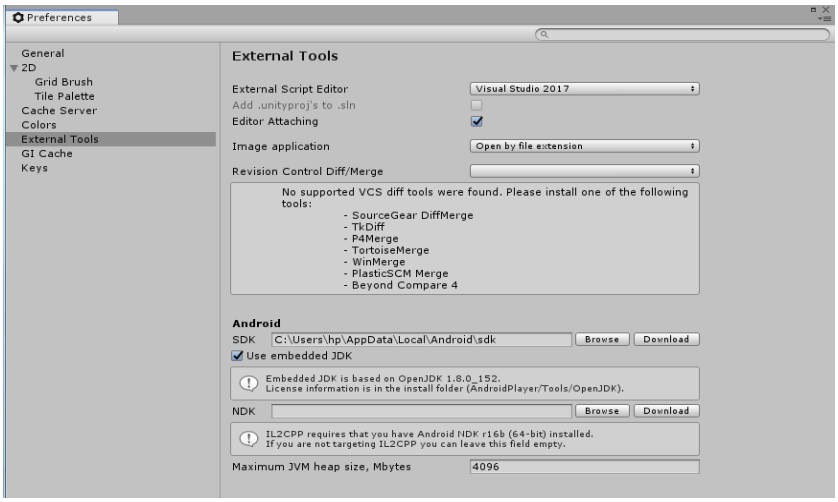


FIGURE 1.10 – Installation du SDK et du JDK

Dans File/ Build Settings/ Player Settings, spécifier le nom l'application, le nom de la société et l'icône par défaut (Figure 1.11).

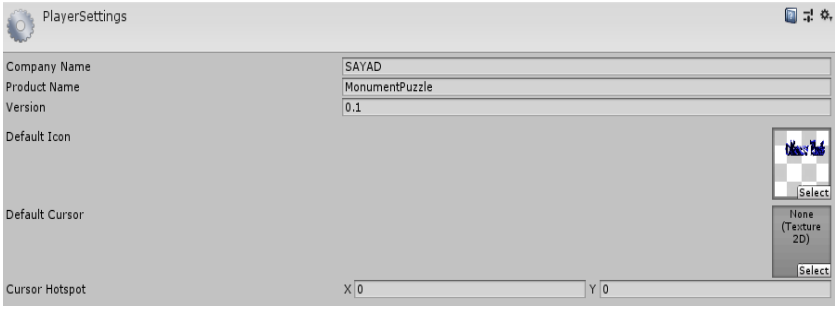


FIGURE 1.11 – Paramètres d'export d'un projet pour Android

Aller dans File/ Build Settings/ Player Settings, dans la partie **Settings for Android**, cliquer sur **Other Settings** et spécifier le nom du package (Figure 1.12).

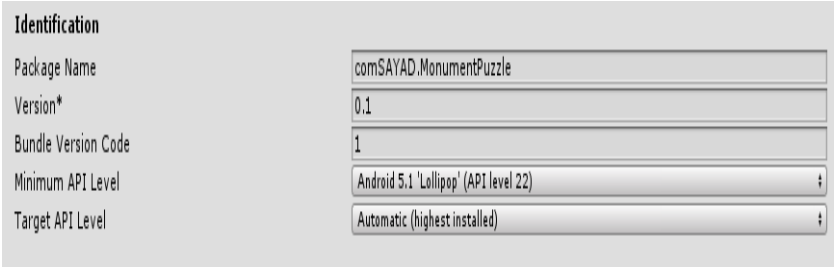


FIGURE 1.12 – Spécification du nom du package

Une fois tout est bien est installé, aller dans File/ Build Settings et ajouter toutes les scènes de l'application par ordre, en ouvrant la scène concernée et en cliquant sur **Add Open Scenes** (Figure 1.13).

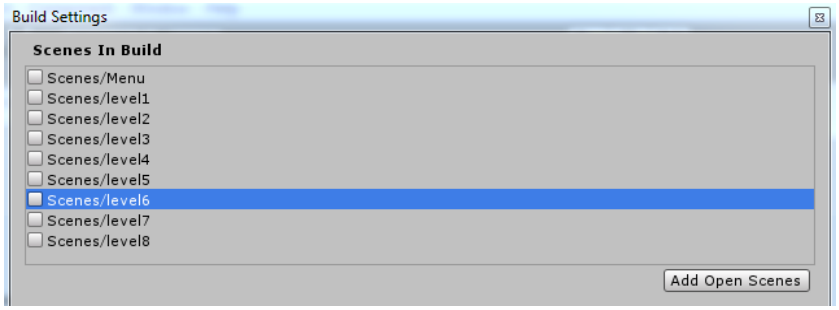


FIGURE 1.13 – Ajout des scènes à exporter

Enfin, dans File/ Build Settings, cocher toutes les scènes ajoutées et cliquer sur **Build**, spécifier le nom de l'APK (Android Package Kit) et son emplacement, puis cliquer sur **Enregistrer**. L'application est prête à être installée sur un appareil Android.

1.5.1.5 Publier une application sur Play Store

Pour publier une application sur Play Store il est nécessaire de suivre les étapes suivantes :

- La première étape consiste à créer un compte développeur sur Google Play, les frais d'enregistrement sur ce dernier sont de 25 $ valable à vie. Si l'application propose une version payante il est obligatoire de configurer le profil de paiement afin de contrôler les paiements et accéder aux rapports de ventes.
- La deuxième étape est de créer une nouvelle application sur Google Play en choisissant une langue par défaut et en lui ajoutant un titre.
- Après avoir créé l'application, il est obligatoire de fournir toutes les informations qui permettront aux utilisateurs de trouver l'application à travers leurs recherches.
- Une fois toutes les informations sont renseignées, il est temps de télécharger l'APK de l'application et de remplir le questionnaire d'évaluation.
- Décider si l'application sera gratuite ou payante et définir son prix.
- Publier l'application et attendre quelques jours pour qu'elle soit validée.

1.5.2 Plateforme iOS

1.5.2.1 Définition

iOS est un système d'exploitation mobile développé par Apple pour les smartphones iPhone, les iPod Touch, les tablettes tactiles iPad et l'Apple TV [11].

1.5.2.2 iOS SDK

iOS SDK est le kit de développement logiciel publié par Apple depuis 2008, il fournit aux développeurs toutes les ressources et les outils nécessaires pour créer, tester, déboguer et distribuer leurs applications iOS [11].

1.5.2.3 App Store

App Store (Figure 1.14) est la plateforme de téléchargement des applications pour les appareils mobiles fonctionnant sous iOS [11].

FIGURE 1.14 – App Store

1.5.2.4 Exporter un projet pour iOS

Pour exporter un projet pour iOS il est indispensable d'avoir le module iOS installé sur Unity, pour ce faire il faut suivre les mêmes étapes que celles d'installation du module Android (voir installation du module Android 1.5.1.4).

Une fois le module est bien installé, aller dans File/ Build Settings, ajouter les scènes que nous voulons exporter, ensuite, cliquer sur build, nous obtiendrons un projet Xcode (Figure 1.15).

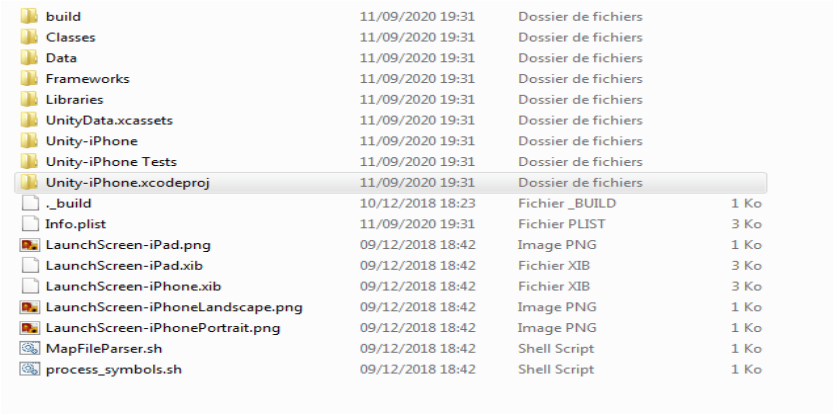


FIGURE 1.15 – Exporter un projet Xcode

La deuxième étape consiste à installer le logiciel **Xcode** sur un Mac (Figure 1.16) qui est un EDI (Environnement de développement intégré) gratuit créé par Apple, il permet de compiler le code pour obtenir des applications iOS, macOS ou watchOS .

Finalement, sur un Macbook, ouvrrir le projet .xcodeproject et faire Build / Archive sur Xcode, le .ipa (iOS App Store Package) sera généré et prêt à être installé sur un appareil ayant un système iOS.

FIGURE 1.16 – Xcode

1.5.2.5 Publier une application sur App Store

Pour publier une application sur App Store il est indispensable de :

- Créer un compte Apple Developer Program et payer 99 $ par an, pour pouvoir accéder aux ressources développeur et aux fonctionnalités d'Apple, parmi ces fonctionnalités on trouve la plateforme web **Itunes Connect** (Figure 1.17) qui permet le test et le paramétrage d'une application iOS afin de la soumettre à apple et de la commercialiser .

FIGURE 1.17 – Itunes Connect

- Avoir une machine MacOS ou créer une machine virtuelle MacOS.
- Avoir le logiciel Xcode installé.
- Après avoir s'inscrit sur Apple Developer Program, il faut lancer Xcode pour s'authentifier et créer deux certificats, un certificat de développement et un certificat de distribution.
- **Le certificat de développement** utilisé lors des tests, il est optionnel pour le déploiement.

- **Le certificat de distribution** est obligatoire pour publier une application sur App Store, il est utilisé pour signer l'application pour la distribution.
- créer un profil d'approvisionnement pour l'application.
- Créer la fiche App Store sur iTunes connect et renseigner toutes les informations et le paramétrage de l'application.
- Publier l'application, il faut attendre quelques heures pour qu'elle soit traitée par Apple.

1.6 Conclusion

Dans ce chapitre, nous avons présenté les notions fondamentales qui nous ont semblé nécessaires pour la réalisation de notre projet. Grâce à notre étude comparative sur ce thème, nous avons pu trancher sur le choix du moteur de jeux adéquat pour le développement de notre jeux en garantissant son opérabilité selon nos besoins.

Le chapitre suivant sera consacré à la spécification des besoins et à la conception de notre projet.

Chapitre 2

Analyses des besoins et conception

2.1 Introduction

Dans ce chapitre nous allons présenter les objectifs, les besoins attendus ainsi que la conception de notre application.

Nous allons commencer par une présentation du processus de développement unifié UP et une définition du langage de modélisation unifié UML.

Ensuite nous allons identifier les différentes fonctionnalités de la future application en spécifiant les besoins fonctionnels et les besoins non fonctionnels, puis nous allons analyser les différents cas d'utilisation.

La dernière partie est consacrée à la conception de notre application, nous commençons par un diagramme d'activités global qui représente toutes les fonctionnalités de la future application ensuite nous allons présenter les différents diagrammes de séquence qui représentent les interactions entre l'acteur et le système et nous terminons par le diagramme de classes.

2.2 Processus unifié

Un processus de développement logiciel contient une séquence d'étapes afin de produire un système logiciel de qualité ou évoluer un système existant, tout en répondant aux besoins et aux exigences des utilisateurs et en prenant en considération le temps et les coûts prévisibles.

Dans notre projet nous avons opté pour le **processus unifié** qui est un processus de développement itératif adapté à tout type de projet. Il couvre toutes les activités de la conception du projet jusqu'à la livraison du produit [13].

2.2.1 Caractéristiques du processus unifié

Les principales caractéristiques du processus unifié sont :

- **UP est piloté par les cas d'utilisation** : Le but de créer un logiciel est de satisfaire les utilisateurs et ceci en étudiant leurs besoins. UP permet

la spécification et l'analyse des besoins fonctionnels des utilisateurs, ces besoins sont illustrés par des cas d'utilisation qui représentent toutes les fonctionnalités du logiciel.
- **UP est centré sur l'architecture** : UP donne une vue sur l'architecture du système à réaliser, cette architecture émerge des besoins fonctionnels exprimés par les utilisateurs et des besoins non fonctionnels (ergonomie, performance ...).
- **UP est itératif et incrémental** : UP est un ensemble d'itérations où un ensemble d'opérations est répété et à chaque fin d'itération une version exécutable du logiciel est produite ce qui permet l'élimination des problèmes imprévus, la prise en compte des besoins des utilisateurs et l'accélération du rythme de développement.
  Incrémental signifie que à chaque itération une version améliorée du système sera créée.

2.2.2 Les étapes du processus UP

UP répète un ensemble d'opérations en 4 étapes :
- **Analyses des besoins** : Cette phase donne une vue globale sur le produit fini, elle permet d'identifier les acteurs, de déterminer les besoins fonctionnels et non fonctionnels du produit.
- **Élaboration** : L'élaboration permet de concevoir une architecture du système en réalisant un modèle initial de cas d'utilisation pour couvrir les besoins fonctionnels et pour planifier la phase de construction, et en identifiant les risques, les coûts, le calendrier et le budget.
- **Construction** : Dans cette phase il faut implémenter tous les cas d'utilisation et fournir une version exécutable du système.
- **Transition** : Dans cette phase les utilisateurs essayent le produit pour détecter les anomalies à corriger et pour vérifier si les services offerts par le système répondent à leurs exigences [13].

2.3 UML

UML est un langage de modélisation, il permet la représentation graphique du logiciel à développer en représentant les besoins des utilisateurs par des diagrammes. Ces diagrammes sont largement utilisés dans chaque étape du processus unifié [14].

2.4 Spécification et analyse des besoins

Notre application est un jeu de puzzle en 3D dont chaque niveau représente un monument, une statue ou une église, dans le but de faire connaître les monuments et leurs histoires.

L'utilisateur déplace et pivote des pièces en 3D pour construire un monument

en une durée déterminée, il pourra aussi consulter l'image du monument pour l'aider à le construire et cela en respectant la durée de consultation.

2.4.1 Les besoins fonctionnels

Afin de satisfaire les utilisateurs, notre jeu doit satisfaire les besoins fondamentaux ci-dessous :

- Dans le menu principal, l'utilisateur peut accéder aux paramètres des sons et des musiques, quitter le jeu, jouer ou continuer le jeu s'il l'a déjà commencé.
- Chaque niveau contient un chronomètre.
- L'utilisateur pivote et déplace les pièces pour construire le puzzle en respectant une durée limitée, cette durée augmente avec la difficulté du niveau. Si cette durée est terminée avant que l'utilisateur puisse finir la partie, il doit recommencer cette partie ou quitter le jeu.
- Si la pièce est mise dans la bonne position le score augmente de 5 points et si elle est bien pivotée le score augmente de 10 points.
- La consultation de l'image du monument construit est possible mais avec une durée limitée qui augmente aussi avec la difficulté du niveau.
- Utiliser un indice pour mettre une pièce choisie aléatoirement dans la position correcte.
- Activer/ Désactiver la musique.
- Activer/ Désactiver les sons.
- Un menu pause qui propose d'aller au menu, de quitter l'application, de recommencer le niveau ou de changer la couleur du monument.
- Si la partie est terminée avec succès une brève histoire du monument s'affiche, l'utilisateur peut la fermer et passer au niveau suivant.

2.4.2 Besoins non fonctionnels

Notre future application doit répondre aux critères suivants :

- **Convivialité** : L'application doit être facile à utiliser, les interfaces doivent être simples et ergonomiques.
- **Rapidité de traitement** : Le temps d'exécution des opérations doit s'approcher du temps réel.
- **Développement de l'application** : L'application sera développée sous Unity 3D en utilisant le langage C#.
- **Sécurité** : L'application doit assurer la sécurité et ses données doivent être protégées.
- **Compatibilité** : Notre application est un jeu mobile fonctionnant sur Android et iOS.

2.4.3 Diagramme de cas d'utilisation

Les principales fonctionnalités de notre application sont illustrées par le diagramme de cas d'utilisation suivant (Figure 2.1) :

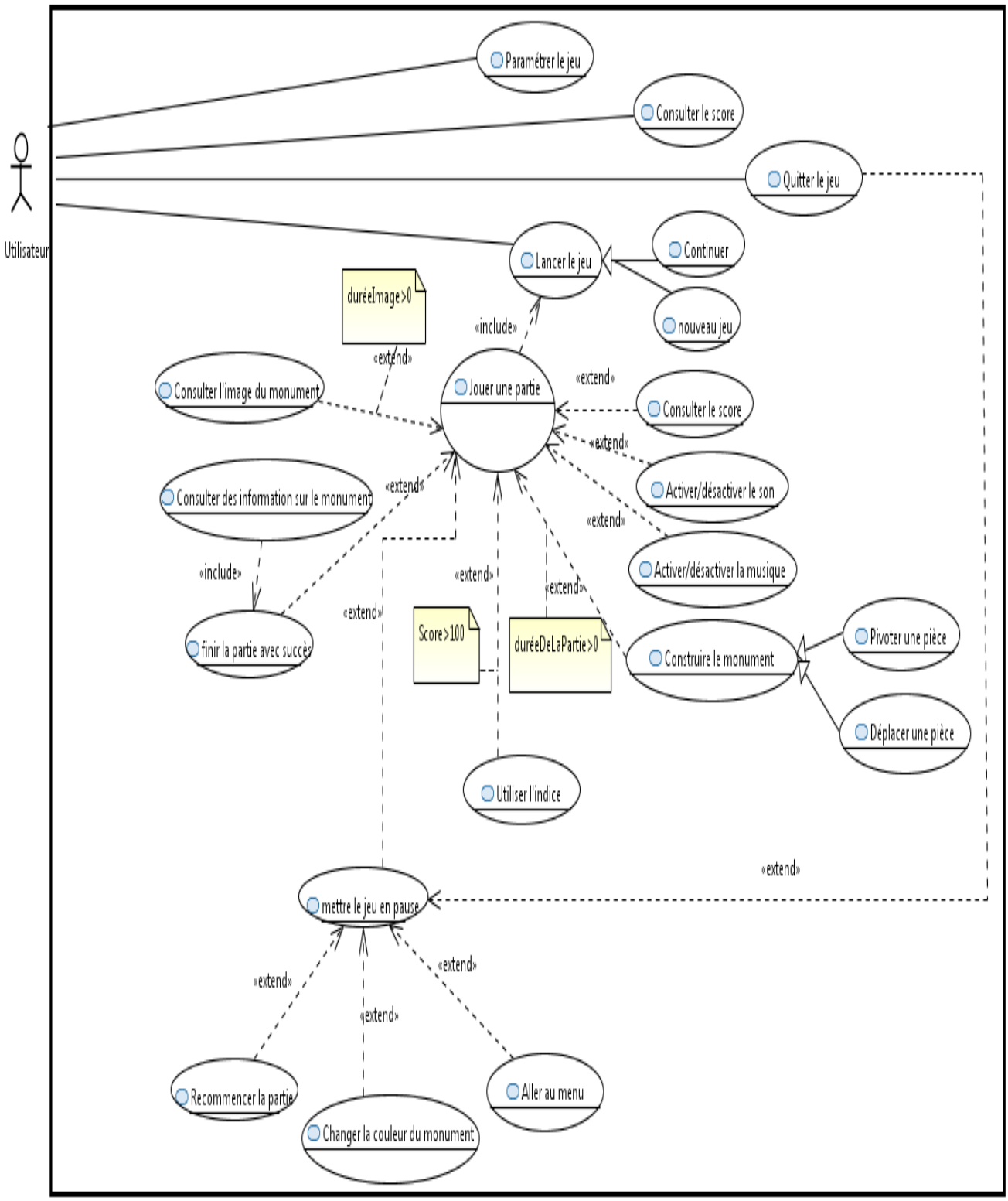


FIGURE 2.1 – Diagramme de cas d'utilisation

Description des différents cas d'utilisation

- **Paramétrer le jeu** : Dans le menu principal l'utilisateur a accès aux paramètres du son et de la musique.
- **Consulter le score** : L'utilisateur peut consulter son score à tout moment dans les différents niveaux du jeu.
- **Quitter le jeu** : L'utilisateur peut quitter le jeu quand il le souhaite.
- **Lancer le jeu** : Pour lancer le jeu il faut choisir soit de jouer à nouveau ou de continuer le jeu.
- **Jouer une partie** : Permet à l'utilisateur de jouer et d'accéder aux différentes fonctionnalités du jeu.
- **Construire le monument** : Permet de glisser, déposer et de pivoter les différentes pièces pour construire le puzzle.
- **Consulter l'image du monument** : Permet à l'utilisateur de consulter l'image du monument avec une contrainte de temps, pour l'aider à le construire.
- **Utiliser l'indice** : Permet aux utilisateurs qui ont un score supérieur à 100 de bénéficier d'une aide permettant de déposer une pièce dans la position correcte.
- **Activer/ Désactiver la musique** : Permet d'activer ou de désactiver la musique du fond.
- **Activer/ Désactiver le son** : Permet d'activer ou de désactiver les différents sons du jeu.
- **Mettre en pause** : Permet de mettre le jeu en pause et d'accéder au menu pause permettant de **recommencer la partie**, d' **aller au menu**, de **changer la couleur du monument** ou de **quiter le jeu**.
- **Finir la partie avec succès** : Si toutes les pièces sont dans les bonnes positions et le puzzle est construit, la partie est finie avec succès et une histoire du monument construit s'affiche à l'écran.

2.5 Conception

Dans cette partie nous détaillons la conception des cas d'utilisation que nous venons d'analyser.

2.5.1 Diagramme d'activités

Le diagramme d'activités global ci-dessous (Figure 2.2) représente toutes les actions de l'application et leurs conditions d'exécution.

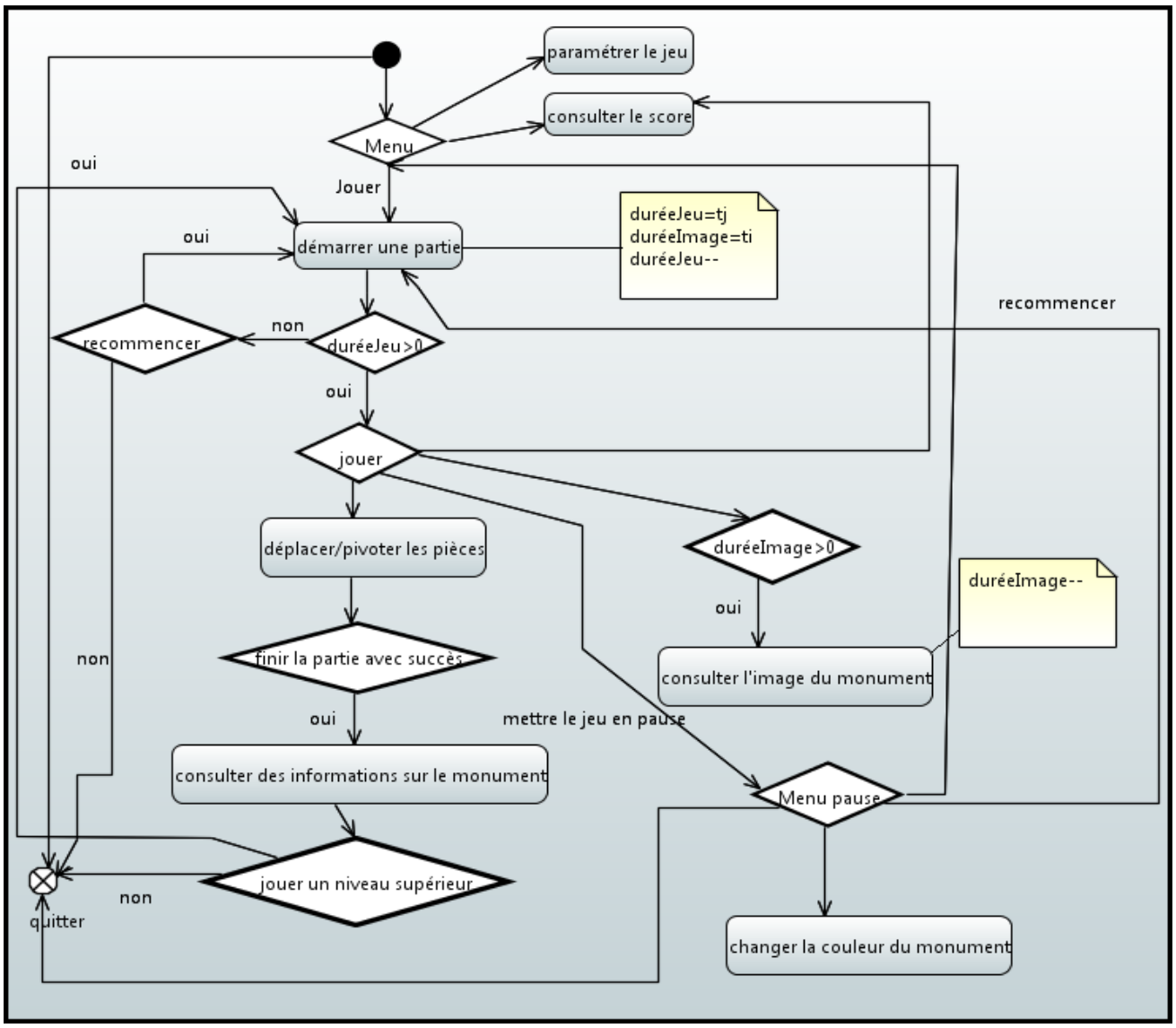


FIGURE 2.2 – Diagramme d'activités

2.5.2 Diagrammes de séquence

Les diagrammes de séquence représentent graphiquement les interactions entre les acteurs et le système selon un ordre chronologique. Dans ce qui suit nous représentons les diagrammes de séquence pour les cas d'utilisation suivant :

- Lancer le jeu.
- Paramétrer le jeu.
- Déplacer une pièce.
- Pivoter une pièce.
- Mettre en pause.
- Mettre en play.

- Consulter l'image du monument.
- Utiliser l'indice.
- Finir une partie avec succès.

- **Diagramme de séquence de cas d'utilisation "Lancer le jeu"**

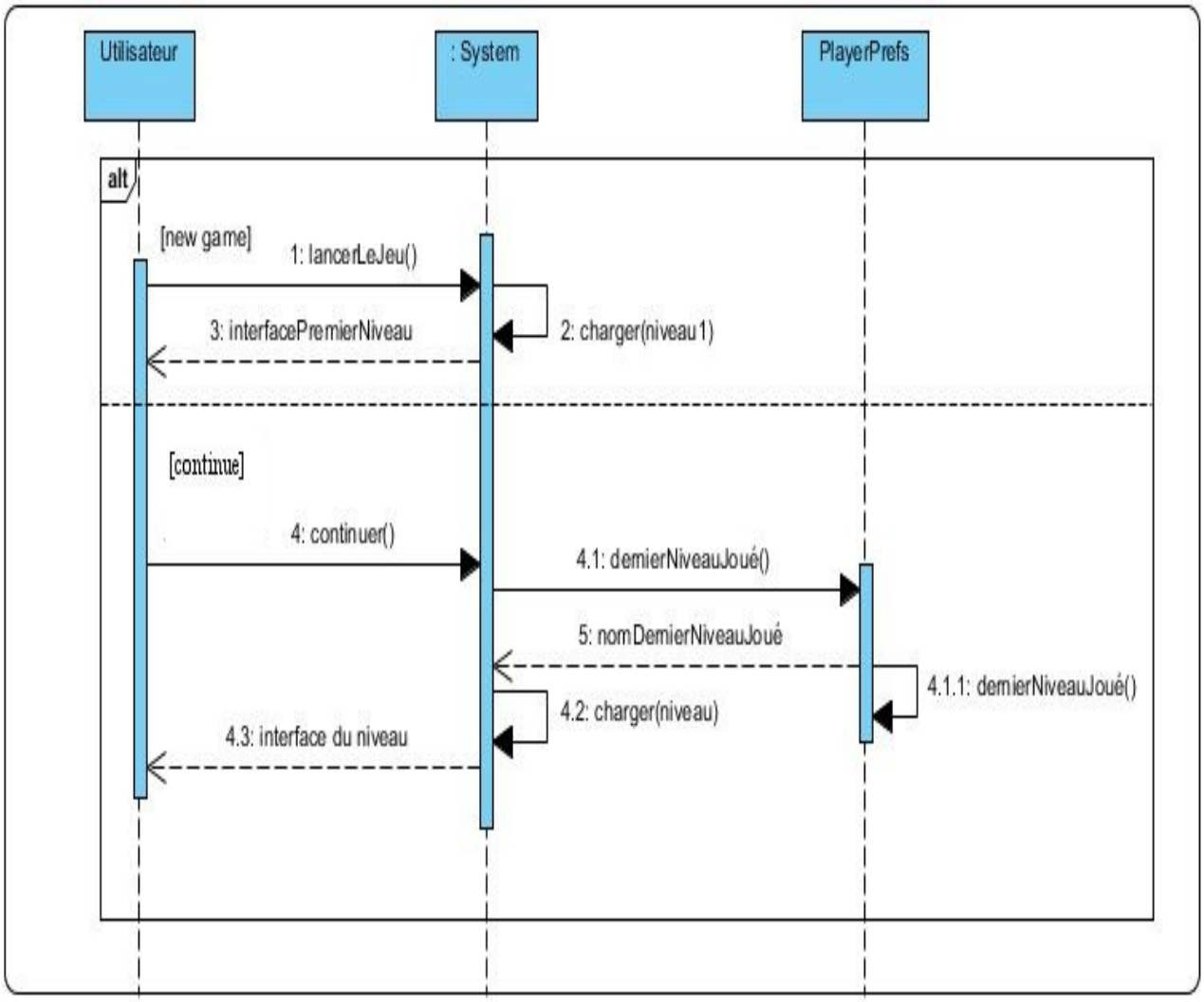


FIGURE 2.3 – Diagramme de séquence de cas d'utilisation "Lancer le jeu"

- **Diagramme de séquence de cas d'utilisation "Paramétrer le jeu"**

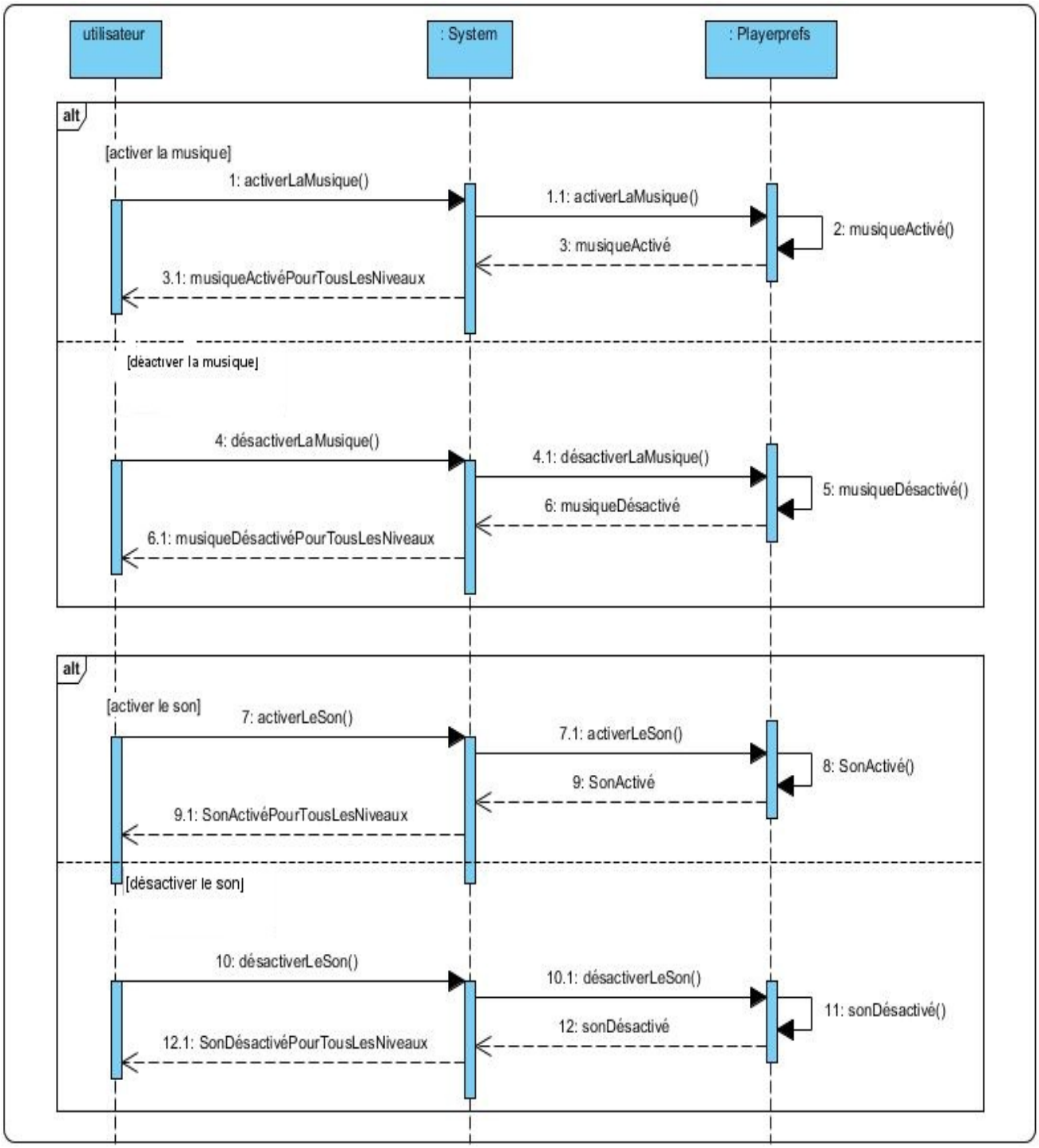


FIGURE 2.4 – Diagramme de séquence de cas d'utilisation "Paramétrer le jeu"

- **Diagramme de séquence de cas d'utilisation "Déplacer une pièce"**

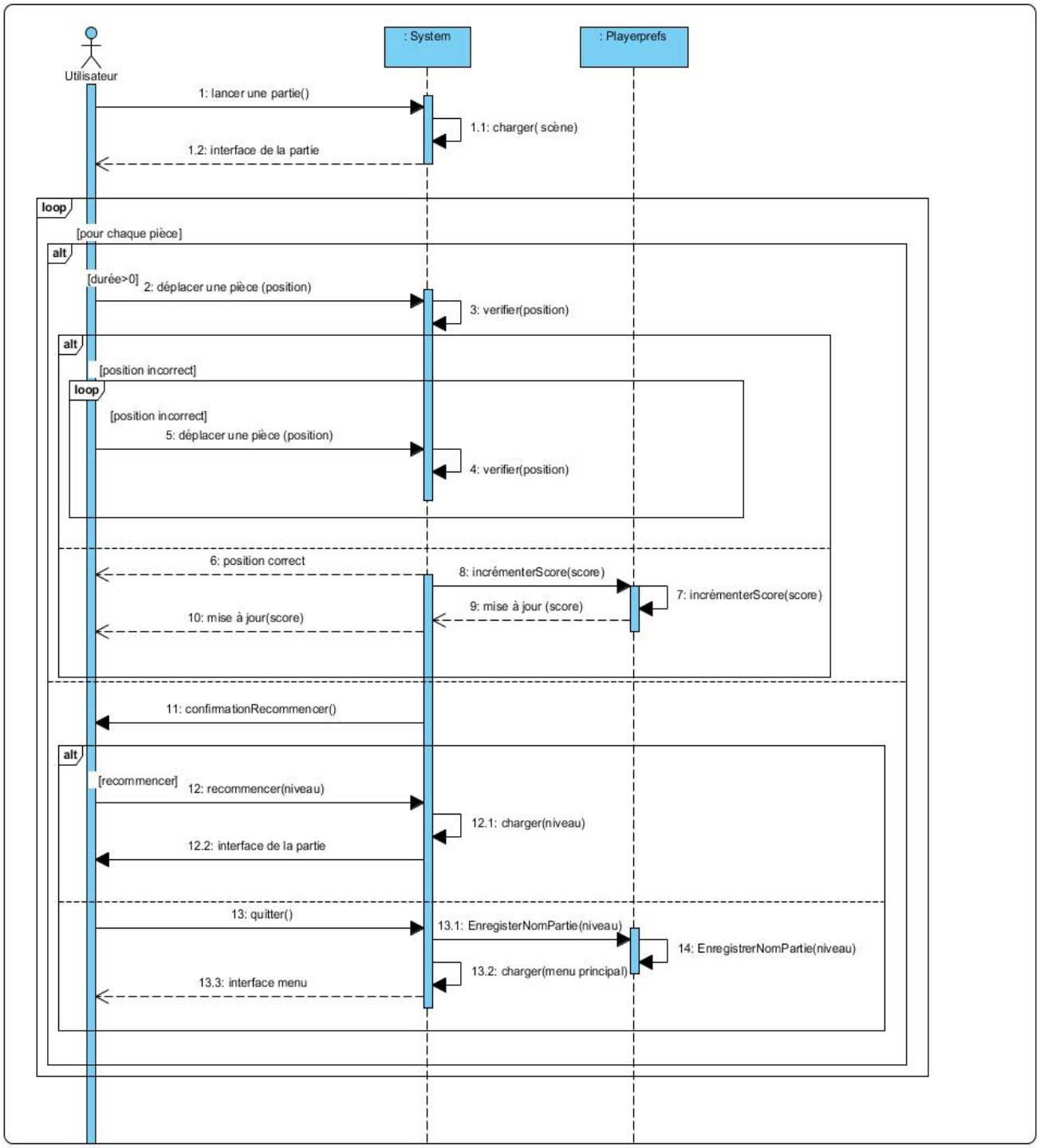


FIGURE 2.5 – Diagramme de séquence de cas d'utilisation "Déplacer une pièce"

- **Diagramme de séquence de cas d'utilisation "Pivoter une pièce"**

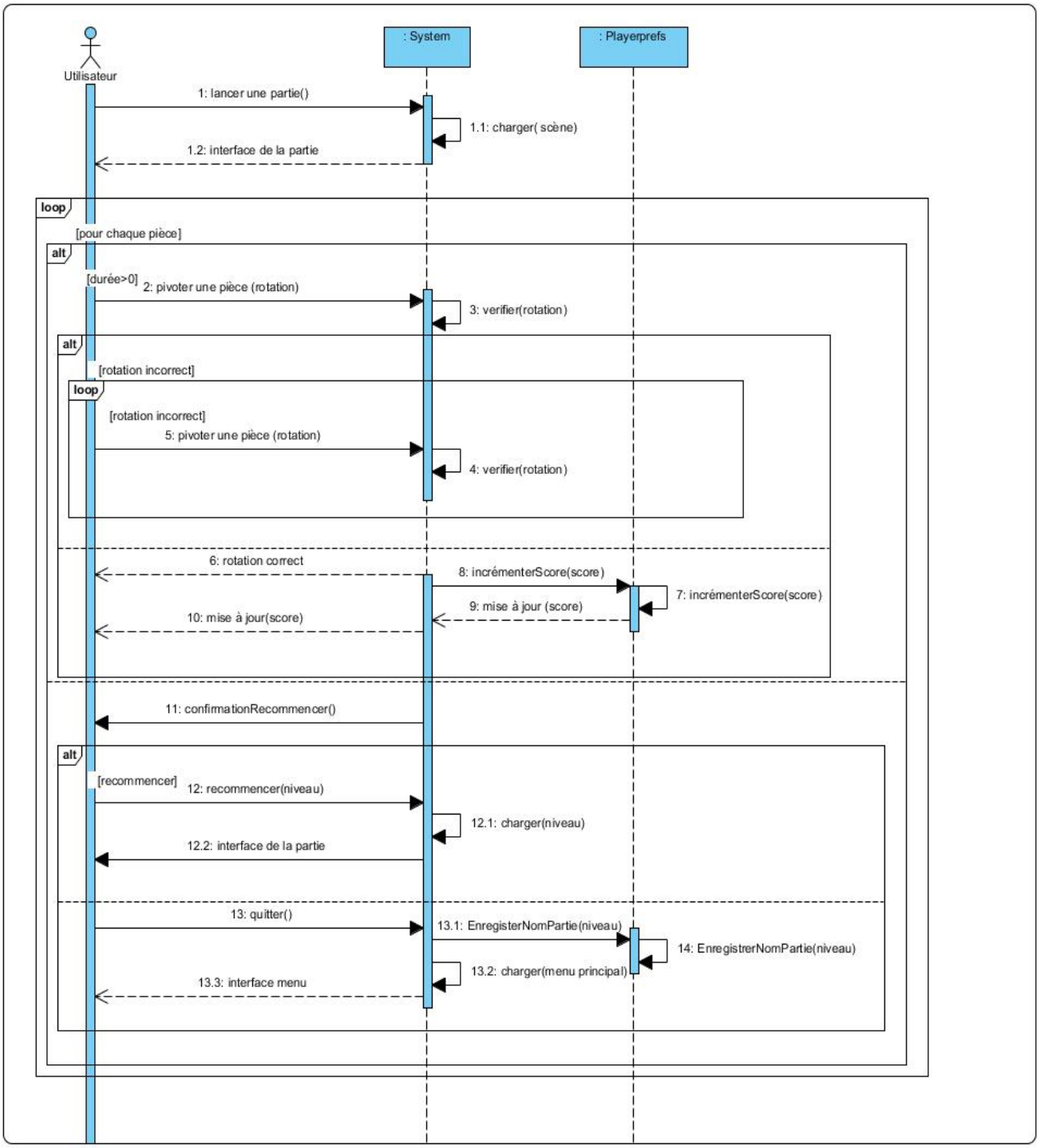


FIGURE 2.6 – Diagramme de séquence de cas d'utilisation "Pivoter une pièce"

- **Diagramme de séquence de cas d'utilisation "Mettre en pause"**

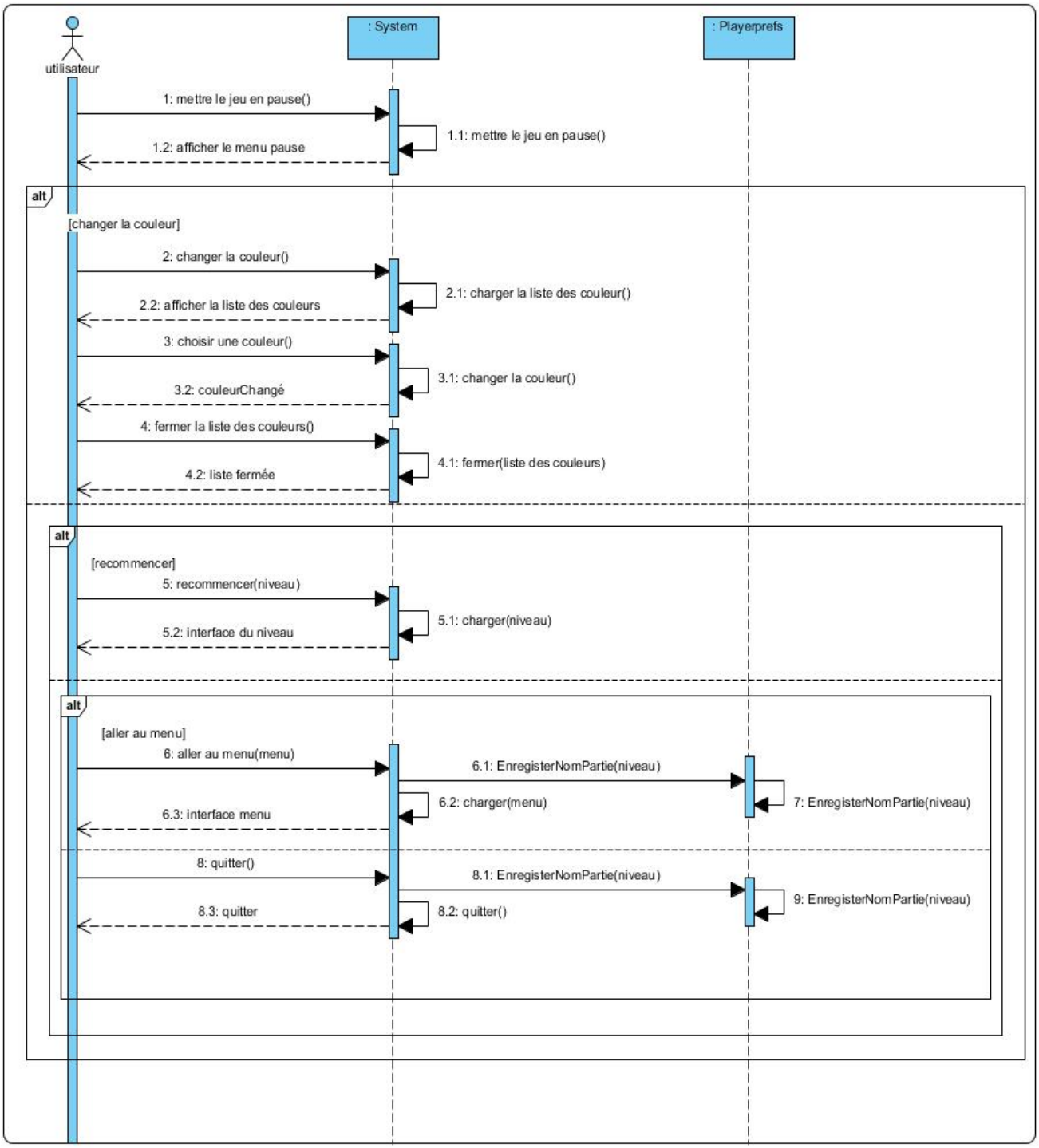


FIGURE 2.7 – Diagramme de séquence de cas d'utilisation "Mettre en pause"

- **Diagramme de séquence de cas d'utilisation "Mettre en play"**

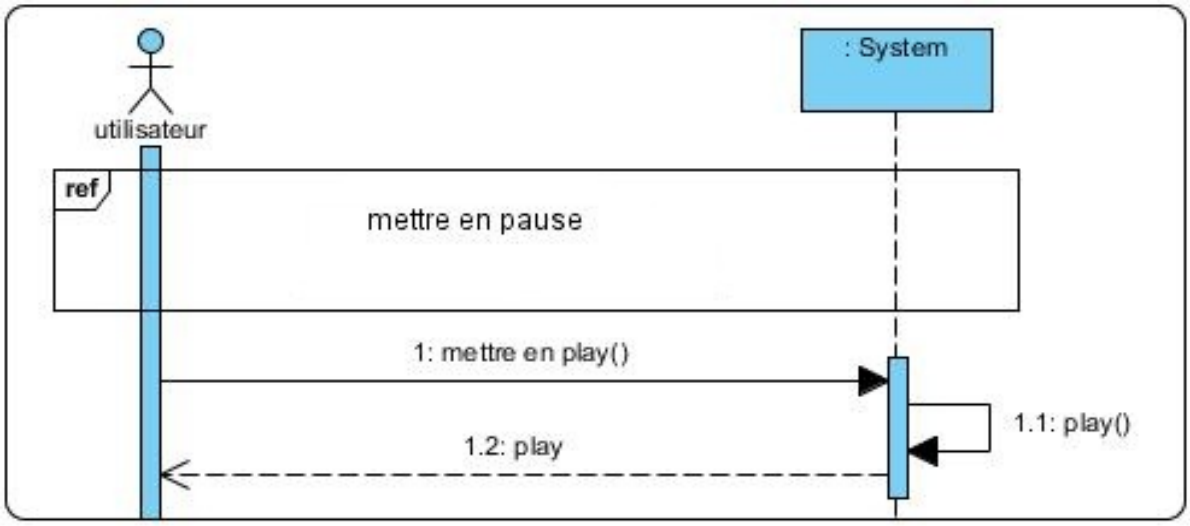


FIGURE 2.8 – Diagramme de séquence de cas d'utilisation "Mettre en play"

- **Diagramme de séquence de cas d'utilisation "Consulter l'image du monument"**

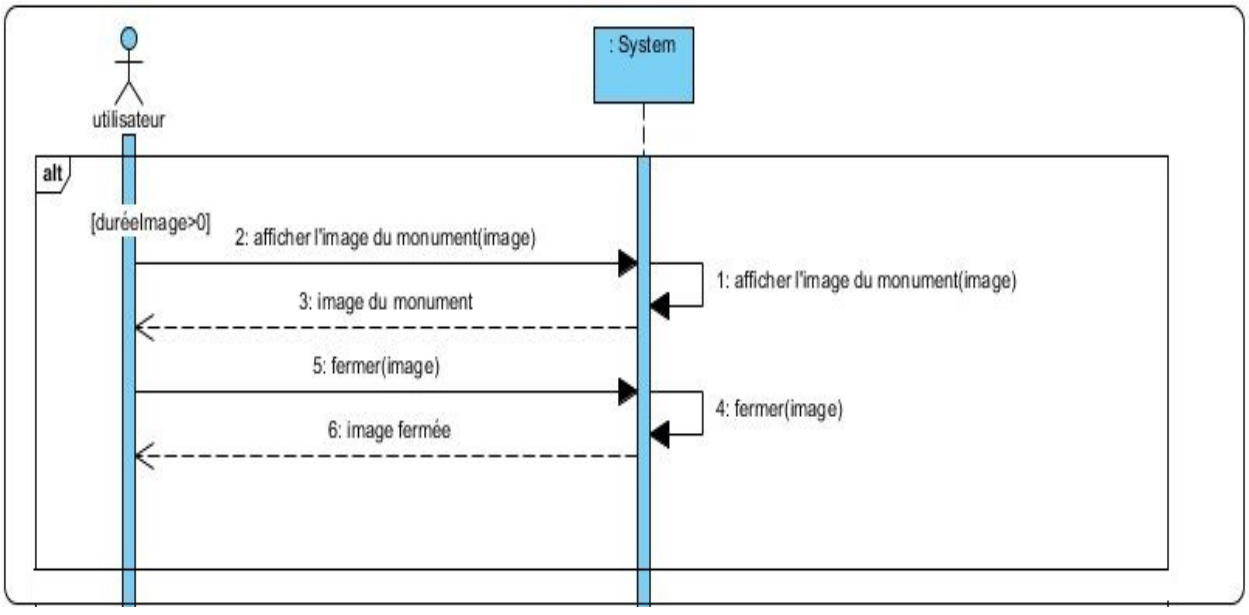


FIGURE 2.9 – Diagramme de séquence de cas d'utilisation "Consulter l'image du monument"

- **Diagramme de séquence de cas d'utilisation "Utiliser l'indice"**

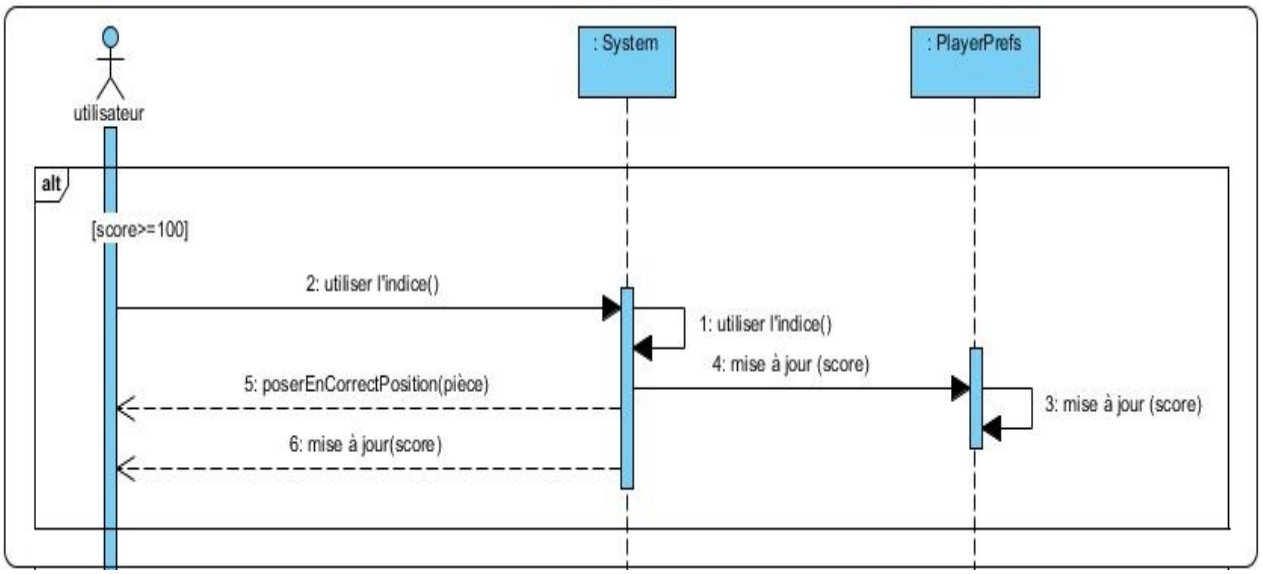


FIGURE 2.10 – Diagramme de séquence de cas d'utilisation "Utiliser l'indice"

- **Diagramme de séquence de cas d'utilisation "Finir une partie avec succès"**

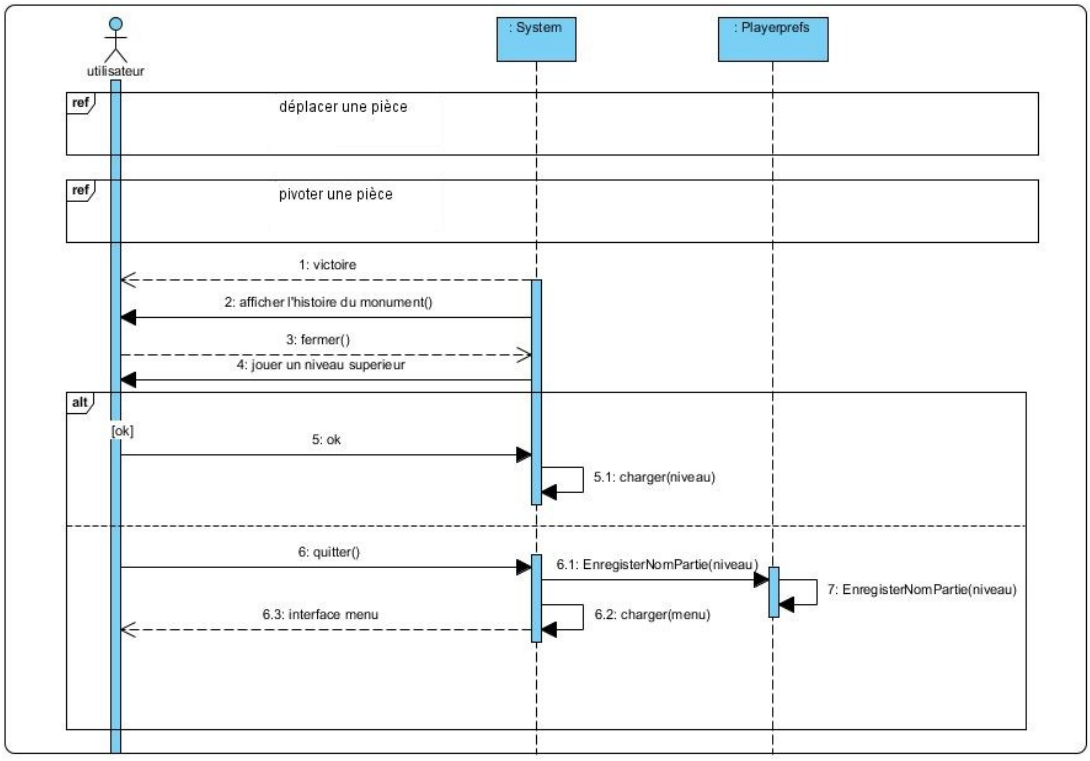


FIGURE 2.11 – Diagramme de séquence de cas d'utilisation "Finir une partie avec succès"

2.5.3 Diagramme de classes

Le diagramme ci-dessous (figure 2.12) représente les différentes classes de notre jeu.

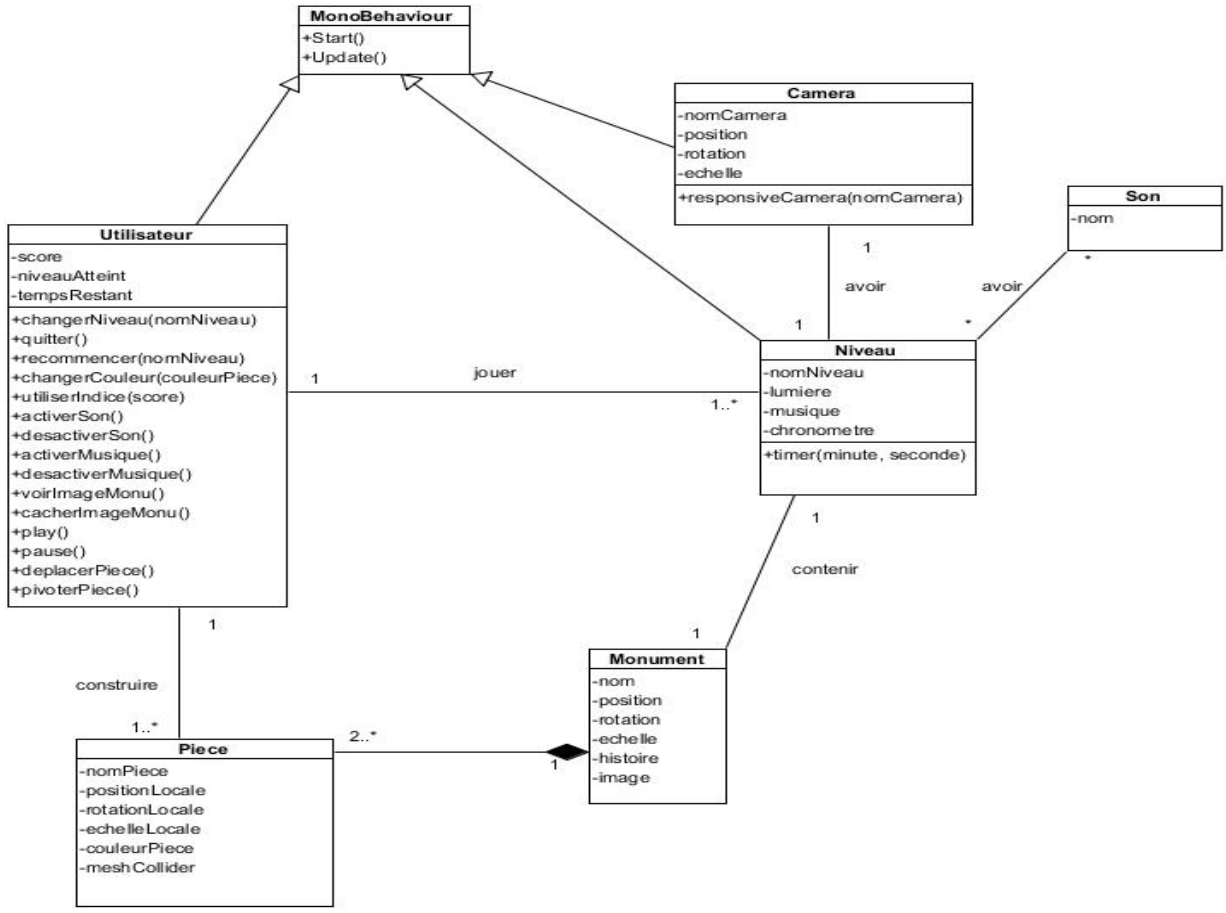


FIGURE 2.12 – Diagramme de classes

2.6 Conclusion

Dans ce chapitre nous avons pu spécifier et analyser les besoins et les différents cas d'utilisation, ensuite nous avons détaillé la conception de notre application.

Dans le chapitre suivant nous entamerons l'étape de réalisation de notre application.

Chapitre 3

Réalisation

3.1 Introduction

Dans ce dernier chapitre, nous allons déterminer tous les outils de développement de notre application, nous allons aussi présenter les grandes étapes qui ont servi à la réalisation de notre application et ses interfaces principales.

3.2 Outils de développement

Pour la réalisation du projet, nous avons utilisé des modèles 3D existants, pour les découper et leurs apporter quelques modifications nous avons utilisé deux logiciels de modélisation 3D (Autodesk 123D Make et Blender). Pour coder notre application nous avons choisi C# pour sa sémantique et sa performance, concernant le moteur de jeux nous avons opté pour Unity 3D et en ce qui concerne les sons et les musiques nous avons utilisé l'existant et nous les avons modifié en utilisant Filmora. Dans cette partie nous allons présenter tous les outils utilisés pour le développement de notre application.

3.2.1 Autodesk 123D Make

Autodesk 123D Make (Figure 3.1) est un logiciel de création et d'édition des modèles 3D et 2D, il permet de découper les modèles 3D [15].

FIGURE 3.1 – Logo d'Autodesk 123D Make.

3.2.2 Blender

Blender (Figure 3.2) est un logiciel de modélisation 3D, nous l'avons utilisé pour modifier des modèles existants à notre convenance [16].

FIGURE 3.2 – Logo de Blender.

3.2.3 Unity

(voir la section 1.4.1)

3.2.4 Environnement de développement

Unity propose **visual studio** (Figure 3.3) créé par Microsoft comme EDI, ce dernier est un EDI complet pour Unity, il permet de créer rapidement des méthodes de scripts Unity et il apporte une expérience de débogage optimale [17].

FIGURE 3.3 – Logo de Visual Studio.

3.2.5 Langage de programmation

Concernant le langage de programmation, nous avons opté pour C# créé par Microsoft en 2002, pour sa syntaxe très ressemblante au Java, sa performance et sa facilité de compréhension.

3.2.6 Secured PlayerPrefs

Pour stocker les données de notre jeu (Le paramétrage qu'a fait le joueur dans le jeu et son score) nous avons opté pour PlayerPrefs qui est un outil qui permet de stocker des données facilement mais ces données ne sont pas protégées, elles sont accessibles, le joueur peut y accéder facilement et les modifier. Pour les sécuriser nous avons utilisé Secured PlayerPrefs qui est un outil téléchargeable gratuitement sur l'asset store de Unity. Secured PlayerPrefs utilse la fonction de hashage **Salt** qui consiste à concaténer une chaîne de caractères aléatoire à la donnée que l'on veut sécuriser, puis hasher la chaîne ainsi créée.

3.2.7 Filmora

Dans notre jeu nous avons utilisé des musiques et des sons existants et pour enlever les droits d'auteur à ces derniers nous les avons modifié avec le logiciel de réalisation et d'édition vidéo **Filmora** (Figure 3.4) [18].

FIGURE 3.4 – Logo de Filmora.

3.3 Ergonomie

Notre application répond aux critères suivants :

- **Langue utilisée** : La langue utilisée est l'anglais car c'est la plus parlée au monde.
- **Police d'écriture** : Bold.
- **Couleurs** : Les couleurs sont cohérentes, nous avons opté principalement pour le bleu. La personnalisation du jeu est possible en donnant à l'utilisateur la possibilité de changer la couleur du monument.
- **Logo** : Nous avons réalisé un logo significatif (Figure 3.5).

FIGURE 3.5 – Logo du jeu.

- **Interface utilisateur** : Une interface très simple et facile à utiliser.
- **Choix des sons et des musiques** : Les sons sont adaptatifs aux évènements de jeu, et les musiques du fond ont été choisies par rapport au lieu où se trouve le monument.

3.4 Sécurité

Pour sécuriser notre application les données du joueurs comme son score, les paramètres qu'il a choisi sont chiffrées en utilisant Secured PlayerPrefs.

3.5 Passage de la modélisation 3D à Unity 3D

Dans notre projet nous avons travaillé sur des modèles 3D existants des monuments et nous avons utilisé des logiciels de modélisation 3D (Blender et Autodesk 123d) pour les découper et les modifier à notre convenance. Pour ce faire nous avons suivi les étapes suivantes :

Blender :

- Dans **File/import**, choisir l'extension de l'objet à importer : .dae, .abc, .fbx, .bvh, .ply, .obj, .x3d, .wrl, .stl, .svg, .glb ou .gltf .
- Parcourir l'emplacement de l'objet à importer et cliquer sur **import**.
- Effectuer les modifications nécessaires sur l'objet, pour modifier un objet sous Blender il faut accéder au mode édition **Edit mode** en cliquant sur la touche **tabulation**, ensuite cliquer sur la touche **Z** pour passer en représentation **wireframe** qui permet de montrer la structure interne de l'objet.
  Pour sélectionner la pièce à découper de l'objet cliquer sur la touche **B** et sélectionner la partie voulue ensuite cliquer sur la touche **P** et choisir **Selection** pour découper l'objet par sélection, il faut refaire toutes ces étapes à chaque fois que nous voulions découper une pièce.
- Pour terminer, cliquer sur **File/Export**, choisir l'extension .fbx, choisir l'emplacement dans lequel nous allons mettre l'objet, il faut le mettre dans le dossier **Assets** du projet créé sur Unity. Dans notre cas, le projet dans Unity s'appelle **Game** et nous voulons le mettre dans le dossier **level3** de **Assets**. Cliquer sur **Export FBX** (Figure 3.6), le modèle est prêt à utiliser dans Unity.

FIGURE 3.6 – Exporter un modèle Blender pour Unity 3D.

Autodesk :

- Le découpage d'un modèle compliqué sur Blender est très compliqué et prend beaucoup de temps, pour faciliter cette tache nous avons opté pour Autodesk qui permet de le faire facilement.
- Pour importer un modèle 3D dans Autodesk, cliquer sur **Import**, il permet d'importer des modèles ayant l'extension .obj ou .fbx .
- Apporter les modifications nécessaires sur le modèle 3D, pour le découper, aller dans le menu latéral d'Autodesk, dans **Construction technique** choisir **3D Slices**, et voir les paramètres nécessaires pour ajuster la taille des pièces, le modèle sera ainsi découpé en pièces de même taille .
- Pour l'exporter cliquer sur la petite flèche en haut à gauche de l'écran, dans **Export Mesh** choisir l'extension avec laquelle nous voulons exporter le modèle, Autodesk propose seulement .obj et .fbx .
- C'est possible d'exporter directement l'objet sur Unity, en l'exportant sur le dossier **Assets** de Unity mais le modèle découpé est considéré comme un seul objet, nous n'avons pas accès aux sous pièces et ce n'est pas notre but, pour régler ce problème nous proposons d'importer le modèle réalisé avec Autodesk dans Blender et ensuite l'exporter pour Unity.

3.6 Fonctionnalités du jeu

Dans cette partie nous allons expliquer le principe, les règles et les fonctionnalités du jeu.

3.6.1 Principe général

Le jeu développé "**Monument Puzzle**" est un jeu de type casse-tête, il reprend le concept de puzzle qui consiste à construire un objet à partir des pièces.

Le principe général est de construire un monument 3D en déplaçant et en pivotant des pièces 3D tout en respectant la contrainte de temps. Toute pièce bien pivotée augmente le score de 10 points et toute pièce dans la bonne position augmente le score de 5 points, si le score total est égal ou supérieur à 100, l'utilisateur peut utiliser un indice en cliquant sur ce dernier une pièce aléatoire se met dans sa position correcte mais cela coûte 100 points de son score.

Notre jeu comprend huit niveaux de difficultés par rapport au nombre de pièces à construire et à la contrainte de temps. Chaque niveau contient un chronomètre dont la durée augmente avec la difficulté de la partie. Dans chaque niveau l'utilisateur peut consulter l'image du monument construit avec une durée limitée. Á la fin de chaque niveau une brève histoire du monument construit s'affiche à l'écran.

Le but de ce jeu est de faire connaître les monuments et leurs histoires.

3.6.2 Implémentation

Pour réaliser notre jeu nous avons créé pour chaque niveau une scène, et nous avons associé à chaque scène une seule caméra et une lumière directionnelle qui crée le même effet que celui de la lumière de soleil dans la scène.

Pour passer d'un niveau à un autre, il faut importer le package **UnityEngine.SceneManagement**.

```
using UnityEngine.SceneManagement;
```

Ensuite, charger la scène voulu avec l'instruction suivante :

```
SceneManager.LoadScene("level1");
```

level1 est le nom de la scène à charger.
La principale fonctionnalité de ce jeu est de déplacer et de pivoter des pièces 3D pour construire le puzzle, pour ce faire nous avons procédé comme suit :

- Dans chaque niveau, nous avons fixer une pièce, elle sert de bonus à l'utilisateur pour qu'il ne soit pas perdu. Il placera les autres pièces par rapport à cette dernière. Cette solution nous évite d'effectuer beaucoup de traitements pour trouver les positions correctes des autres pièces, si nous fixons une pièce, chacune des autres pièces a une seule position correcte possible.
- Pour déplacer une pièce, nous avons créé une classe qui hérite de la classe **MonoBehaviour**, cette dernière est une classe de base dont dérive chaque script Unity.
- Ensuite dans la méthode Update(), nous avons écrit le code qui permet de détecter le toucher du doigt sur l'écran (Ligne 29-46) et de déplacer la pièce en suivant le mouvement du doigt sur l'écran (Ligne 49-66) si cette dernière n'est pas dans la position correcte (Ligne 56-65).

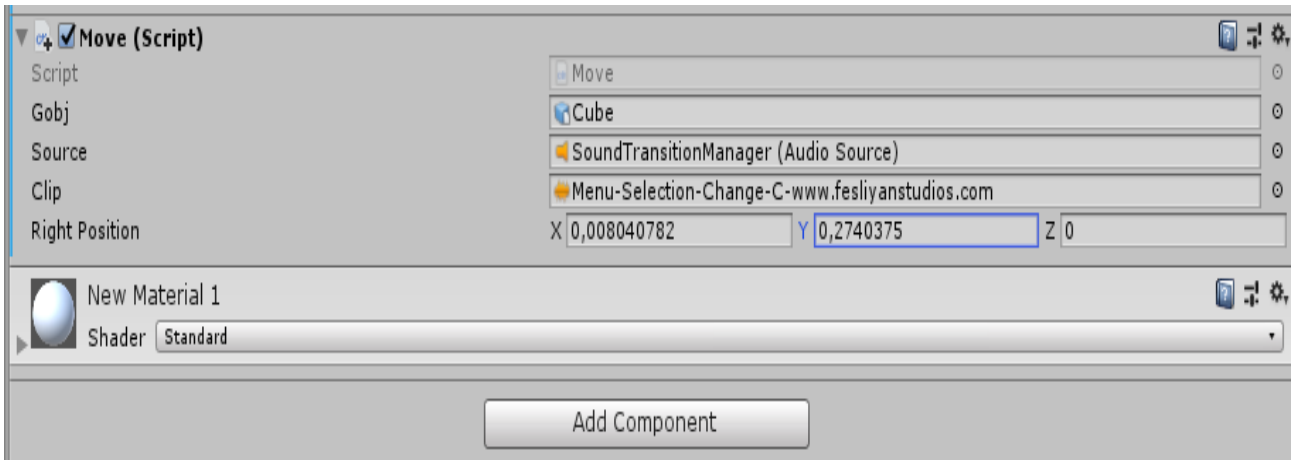
```
28          //Si un contact avec l'ecran a ete detecte
29          if (Input.touchCount > 0)
30          {
31              // Si un doigt a touché l'écran
32              if (Input.GetTouch(0).phase == TouchPhase.Began)
33              {
34
35                  Ray mouseRay = GenerateMouseRay(Input.GetTouch(0).position);
36                  RaycastHit hit;
37                  if (Physics.Raycast(mouseRay.origin, mouseRay.direction, out hit))
38                  {
39                      gobj = hit.transform.gameObject;
40                      objPlane = new Plane(Camera.main.transform.forward * -1, gobj.transform.position);
41                      Ray mRay = Camera.main.ScreenPointToRay(Input.GetTouch(0).position);
42                      float rayDistance;
43                      objPlane.Raycast(mRay, out rayDistance);
44                      m0 = gobj.transform.position - mRay.GetPoint(rayDistance);
45                  }
46              }
47
48              //si un doigt a déplacé sur l'écran
49              else if (Input.GetTouch(0).phase == TouchPhase.Moved && gobj)
50              {
51
52                  Ray mRay = Camera.main.ScreenPointToRay(Input.GetTouch(0).position);
53                  float rayDistance;
54
55                  //Si la pièce n'est pas dans la bonne position déplacer la pièce
56                  if (gobj.transform.localPosition != rightPosition )
57                  {
58                      if (objPlane.Raycast(mRay, out rayDistance))
59                      {
60                          //activer le son
61                          source.PlayOneShot(clip);
62                          //déplacer la pièce
63                          gobj.transform.position = mRay.GetPoint(rayDistance) + m0;
64                      }
65                  }
66              }
67
68              //si un doigt a été levé de l'écran
69              else if (Input.GetTouch(0).phase == TouchPhase.Ended && gobj)
70              {
71                  gobj = null;
72              }
73          }
```

- L'utilisateur ne va pas s'amuser à chercher la position correcte de la pièce, si la pièce est très proche de la position correcte elle se met automatiquement dans la bonne position.

```
distance = Vector3.Distance(gobj.transform.localPosition, rightPoistion);
if (distance < 1f)
{
    gobj.transform.localPosition = rightPoistion;

}
```

- Finalement nous avons assigné le script à chaque pièce.



- Pour pivoter une pièce il faut faire un double clic sur cette dernière, pour cela nous avons initialisé un compteur **counter** à 0 pour calculer le nombre de clic (Ligne 28-46) et nous avons initialisé un chronomètre **clickTimer** à 3 (Ligne 42), si l'utilisateur clique sur la pièce pour la première fois le chronomètre se lance (Ligne 48-51) pour calculer le temps entre le premier clique et le deuxième (Ligne 56-57), Nous avons aussi initialisé **y** à 90, pour pivoter la pièce sur l'axe y si l'utilisateur fait deux clics successifs sur la pièce (Ligne 59-75).

```
26      void Update()
27      {
28          if (Input.touchCount == 1)
29          {
30              //détecter un toucher
31              if (Input.GetTouch(0).phase == TouchPhase.Began)
32              {
33                  Ray mouseRay = GenerateMouseRay(Input.GetTouch(0).position);
34                  RaycastHit hit;
35                  if (Physics.Raycast(mouseRay.origin, mouseRay.direction, out hit))
36                  {
37                      gobj = hit.transform.gameObject;
38                      Ray mRay = Camera.main.ScreenPointToRay(Input.GetTouch(0).position);
39                      float rayDistance;
40                      objPlane.Raycast(mRay, out rayDistance);
41                      //initialiser un timer pour calculer le temps entre le premier toucher et le deuxième
42                      clickTimer = 3;
43                      //compter le nombre de toucher
44                      counter++;
45                  }
46              }
47              // si le nombre de clique sur la pièce est impaire lancer le timer
48              if (counter % 2 != 0)
49              {
50                  StartCoroutine(timer());
51              }
52          }
53      }
```

```
54      IEnumerator timer()
55      {
56          yield return new WaitForSeconds(0.1f);
57          clickTimer = clickTimer - 0.1f;
58          //si le nombre de clique sur la pièce est paire
59          if (counter % 2 == 0)
60          {
61              //si le temps n'est pas fini
62              if (clickTimer >= 0)
63              {
64                  //si la pièce n'est pas dans la rotation correcte
65                  if (gobj.transform.localRotation != Quaternion.Euler((float)-89.98, 180, 0))
66
67                  {
68                      //pivoter la pièce sur l'axe y de 90°
69                      gobj.transform.localRotation = Quaternion.Euler((float)-89.98, (float)y, 0);
70                      //activer le son
71                      source.PlayOneShot(clip);
72                      y = y + 90;
73                  }
74              }
75          }
76
77      }
```

- Enfin nous avons assigné le script à chaque pièce.

3.6.3 Déroulement du jeu

- **Menu principal** :
  Le menu principal est la première interface qui s'affiche au lancement du jeu (Figure 3.7), c'est là où l'utilisateur peut jouer une partie en cliquant sur le bouton "**New Game**" pour jouer à partir du premier niveau, ou en cliquant sur le bouton "**Continue**" pour continuer à jouer s'il a déja commencé. Dans cette interface l'utilisateur peut aussi quitter le jeu en cliquant sur le bouton en haut à droite. Pour accéder aux paramètres de jeu, il suffit de cliquer sur le bouton en haut à gauche et une petite fenêtre apparaîtra, elle contient des cases à cocher pour désactiver/ activer les musiques et les sons de tous les niveaux de jeu, pour fermer cette fenêtre il suffit de la toucher avec le doigt, Pour désactiver la musique du fond et les sons du menu seulement il faut cliquer sur les boutons en haut à droite de l'interface (Figure 3.8).

FIGURE 3.7 – Interface "Menu principal".

FIGURE 3.8 – Interface "Paramètres" .

- **Niveau 1** : Le premier niveau est le niveau le plus simple du jeu (Figure 3.9), il contient une seule pièce à ramasser pour construire **l'arc de triomphe** (Figure 3.10), il est considéré comme un tutoriel pour expliquer à l'utilisateur le fonctionnement du jeu.

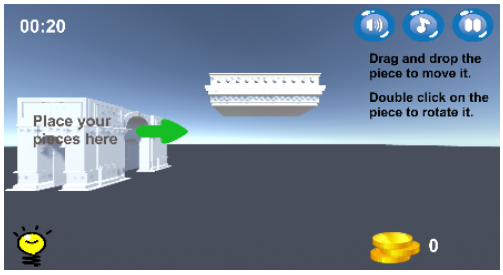


FIGURE 3.9 – Interface "Premier niveau" .

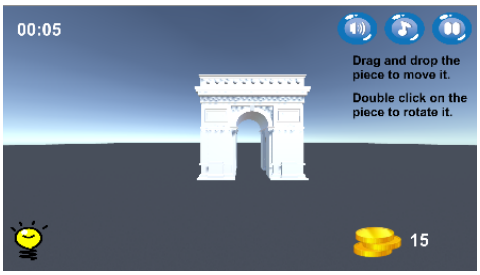


FIGURE 3.10 – Interface "Fin du premier niveau" .

- Ce niveau comprend deux messages indicatifs "**Drag and drop the piece to move it**" et " **Double click on the piece to rotate it**", indiquant à l'utilisateur qu'il doit glisser déposer la pièce pour la faire bouger, et pour la pivoter il doit double cliquer sur elle.
- Au lancement de chaque niveau un message d'indication "**Place your pieces here**" avec une flèche s'affichent pendant 3 secondes indiquant à l'utilisateur où il doit placer ses pièces.
- Le score s'affiche en bas à droite de l'écran.
- Pour consulter l'image du monument construit, il faut cliquer sur l'ampoule en bas à gauche de l'écran et cela pour 5 secondes seulement (Figure 3.11).

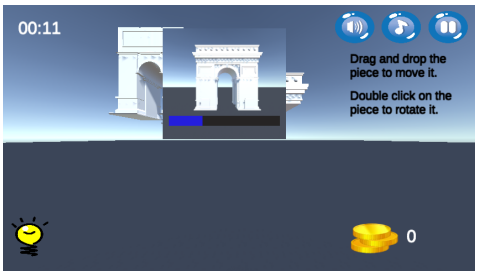


FIGURE 3.11 – Interface "Consulter l'image du monument" .

- Ce niveau doit se faire dans 20 secondes sinon un message s'affiche à l'écran demandant à l'utilisateur s'il veut recommencer le jeu ou le quitter. Le chronomètre s'affiche en haut à gauche de l'écran (Figure 3.12).

FIGURE 3.12 – Interface "Message d'échec" .

- En haut à droite de l'écran, l'utilisateur peut activer/ désactiver la musique et/ou les sons du niveau.
- En cliquant sur le bouton pause en haut à droite de l'écran, un menu pause (Figure 3.13) s'affiche dont l'utilisateur pourra : aller au menu par un clic sur le bouton "**Home**", recommencer la partie en cliquant sur le bouton "**Restart**", changer la couleur du monument en cliquant sur bouton " **Color settings**" (Figure 3.14), (Figure 3.15), et quitter complètement le jeu en cliquant sur le bouton "**Exit**".

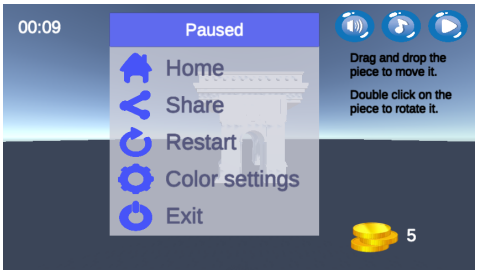


FIGURE 3.13 – Interface "Menu pause" .

FIGURE 3.14 – Interface "Changer la couleur du monument" .

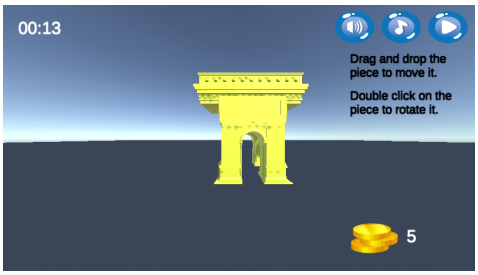


FIGURE 3.15 – Interface "Résultat du Changement de la couleur du monument" .

- Si la partie est terminée avec succès, une fenêtre s'affiche indiquant que la partie est terminée avec succès (Figure 3.16).

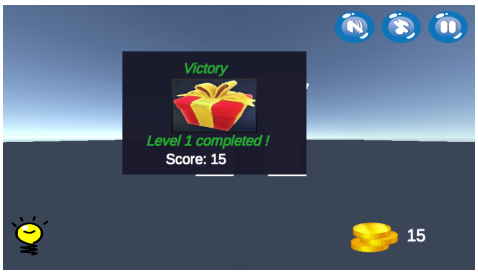


FIGURE 3.16 – Interface "Partie terminée avec succès" .

.

- Á la fin de cette partie une brève histoire du monument s'affiche à l'écran (Figure 3.17).

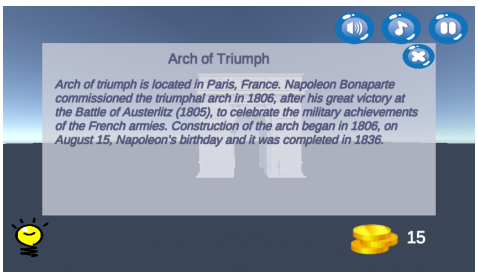


FIGURE 3.17 – Interface "Histoire du monument" .

.

- En fermant la fenêtre de l'histoire qui s'affiche à l'écran, un message apparaît demandant à l'utilisateur s'il veut accéder au niveau supérieur ou bien aller au menu principal (Figure 3.18).

FIGURE 3.18 – Interface "Accéder à un niveau supérieur du jeu" .

.

- **Niveau 2** : Le deuxième niveau (Figure 3.19) comprend 2 pièces pour construire **la statue de la liberté** dans 30 secondes, l'utilisateur peut consulter l'image du monument pour 5 secondes.

FIGURE 3.19 – Interface "Deuxième niveau".

.

- **Niveau 3** : Le troisième niveau (Figure 3.20) comprend 11 pièces pour construire **la tour eiffel** dans une minute, l'utilisateur peut consulter l'image du monument pour 10 secondes.

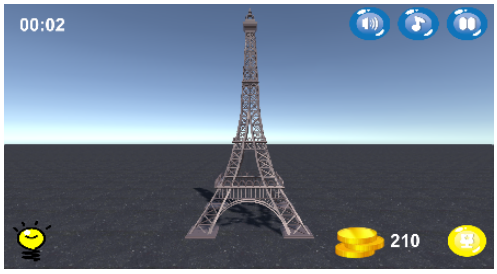


FIGURE 3.20 – Interface "Troisième niveau" .

.

- **Niveau 4** : Le quatrième niveau (Figure 3.21) comprend 24 pièces pour construire l'église **Saint-Fiacre** dans une minute et 30 secondes, l'utilisateur peut consulter l'image du monument pour 10 secondes.

FIGURE 3.21 – Interface "Quatrième niveau" .

.

- **Niveau 5** : Le cinquième niveau (Figure 3.22) comprend 27 pièces pour construire **taj mahal** dans une minute et 30 secondes, l'utilisateur peut consulter l'image du monument pour 10 secondes.

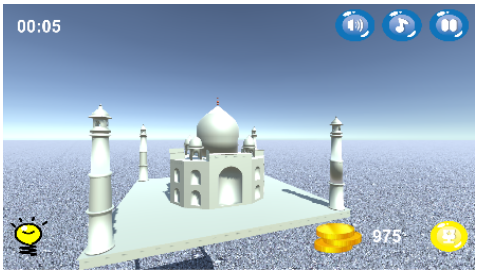


FIGURE 3.22 – Interface "Cinquième niveau".

.

- **Niveau 6** : Le sixième niveau (Figure 3.23) comprend 43 pièces pour construire **la statue de bouddha** dans deux minute et 30 secondes, l'utilisateur peut consulter l'image du monument pour 20 secondes.

FIGURE 3.23 – Interface "Sixième niveau" .

.

- **Niveau 7** : Le septième niveau (Figure 3.24) comprend 44 pièces pour construire **la statue du roi Martin Luther** dans deux minute et 30 secondes, l'utilisateur peut consulter l'image du monument pour 20 secondes. .

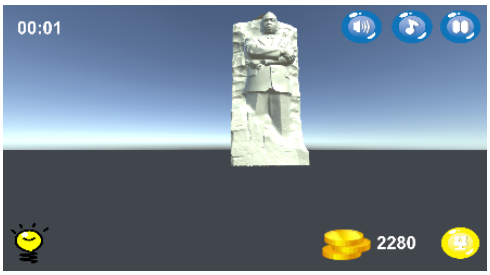


FIGURE 3.24 – Interface "Septième niveau" .

- **Niveau 8** : Le huitième niveau (Figure 3.25) comprend 69 pièces pour construire l'église **notre dame de paris** dans trois minutes, l'utilisateur peut consulter l'image du monument pour 30 secondes.

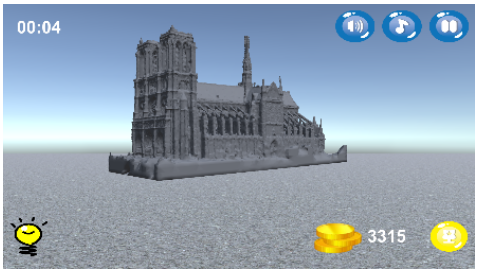


FIGURE 3.25 – Interface "Huitième niveau" .

- Á la fin du jeu une fenêtre apparaîtra indiquant la fin du jeu (Figure 3.26).

FIGURE 3.26 – Interface "Fin du jeu" .

3.7 Conclusion

Dans ce chapitre, nous avons décrit brièvement les outils de développement de notre application, ensuite, nous avons présenté le principe et les fonctionnalités du jeu illustrées par des captures de ses interfaces.

Conclusion générale et perspectives

Notre travail consistait à mettre en oeuvre un jeu mobile de puzzle 3D. Au cours de ce mémoire nous avons présenté des généralités sur le développement de jeux vidéo et les deux plateformes Android et iOS, nous avons aussi expliqué les concepts de Unity 3D. Ensuite, nous avons défini les besoins de l'application, et nous avons analysé les différents cas d'utilisation par les diagrammes UML. Enfin, nous avons défini les différents outils de développement utilisés pour effectuer ce travail et nous avons terminé par présenter les différentes interfaces de notre application.

Ce projet nous a permis d'apprendre énormément de choses dans le développement des jeux vidéo et d'améliorer nos compétences, il représente notre première expérience pour la mise en oeuvre d'un jeu vidéo 3D. Au cours de ce projet nous avons pu apprendre un langage de programmation **C#**, découvrir la 3D, comprendre les concepts du moteur de jeux **Unity 3D** et utiliser pleins d'outils comme **Blender, Autodesk 123D Make, Filmora**.

Comme perspectives, nous pensons que cette application peut être améliorée dans les points suivants :

- Créer des modèles 3D plus ressemblants à la réalité, et intégrer la technologie de la réalité augmentée ou bien la réalité virtuelle. Cela permettra par exemple à l'utilisateur de découvrir et de visiter des monuments sans se déplacer et sans trop dépenser de l'argent.
- Présenter les monuments d'Algérie et les faire connaître au monde.
- Implanter la fonctionnalité **Partager sur les réseaux sociaux**.
- Mettre le jeu en mode multi-joueurs afin de permettre à l'utilisateur de passer un bon moment avec sa famille ou avec ses amis.

Bibliographie

[1] Le marché mondial des jeux vidéo. <https://www.bpifrance.fr/A-la-une/Actualites/Marche-des-Jeux-video-2019-sous-le-signe-de-la-croissance-et-du-mobile-4 2019. [Consulté le 09/04/2020].

[2] L'histoire des jeux vidéo. <http://www.ac-grenoble.fr/college/verney.sallanches/drupal/?q=system/files/Les%20Jeux%20Vid%C3%A9o_0.pdf>. [Consulté le 04/04/2020].

[3] Forsans Emmanuel. Les genres de jeux vidéo sur support physique. <https://www.afjv.com/news/2802_les-genres-de-jeux-video-sur-support-physique.htm>, 2013. [Consulté le 04/04/2020].

[4] Tristan Gaudiaut. Les leaders de l'édition de jeux vidéo. <https://fr.statista.com/infographie/18486/classement-des-editeurs-de-jeux-video-selon-le-chiffre-d-affaires/>, 2019. [Consulté le 04/04/2020].

[5] BENSLIMANE Abdallah. *Portail des jeux éducatifs*. PhD thesis, Abou Bekr Belkaid Tlemcen, 2015.

[6] HOUYEZ Hugo. Qu'est ce que unity 3d ? <https://www.supinfo.com/articles/single/6087-qu-est-ce-que-unity-3d>, 2017. [Consulté le 05/04/2020].

[7] Will Goldstone. *Unity game development essentials*. Packt Publishing Ltd, 2009.

[8] HERRENSCHNEIDER Daniel. Comparaison entre unity 5 et unreal engine 4. <https://www.supinfo.com/articles/single/6087-qu-est-ce-que-unity-3d>, 2016. [Consulté le 08/04/2020].

[9] <https://www.unrealengine.com/en-US/>, 2020. [Consulté le 08/04/2020].

[10] <https://unity.com/>, 2020. [Consulté le 08/04/2020].

[11] <https://fr.wikipedia.org/wiki/IOS>, 2020. [Consulté le 09/04/2020].

[12] Mohamed Anouar DAHDEH. *Conception, développement et intégration d'une application embarqué de téléchargement des applications Androïd : Ftab Stor*. PhD thesis, Université Virtuelle de Tunis, 2011.

[13] G.Picard. *Conduite de projet, Méthode d'analyse et de conception, Processus unifié*. PhD thesis, Ecole Nationale Supérieure des Mines Saint-Étienne, 2008.

[14] Benoît Charroux, Aomar Osmani, and Yann Thierry-Mieg. *UML 2 : pratique de la modélisation*. Pearson Education, 2010.

[15] <https://www.autodesk.com/solutions/123d-apps>. [Consulté le 09/04/2020].

[16] <https://www.blender.org/>. [Consulté le 09/04/2020].

[17] <https://docs.unity3d.com/Manual/30_search.html?q=visual+studio>. [Consulté le 09/04/2020].

[18] <https://filmora.wondershare.com/fr/>. [Consulté le 09/04/2020].

RÉSUMÉ

Monument Puzzle est un jeu mobile 3D de construction de monuments. Le but de ce travail est de faire connaître les célèbres monuments, leurs localisations et leurs histoires. Ce jeu est conçu pour les deux plateformes Android et iOS, en utilisant le moteur de jeux Unity 3D, le langage de programmation C#, des logiciels de modélisation 3D (Blender et Autodesk 123D Make) et le langage UML pour réaliser l'étude conceptuelle du jeu .

Mots clés : Monument Puzzle, Jeu 3D, 3D, Android, iOS, Unity 3D, C#.

ABSTRACT

Monument Puzzle is a 3D mobile game for building monuments. The aim of this work is to make famous monuments, their locations and their stories known. This game is designed for both Android and iOS platforms, using Unity 3D game engine, C# programming language, 3D modeling software (Blender and Autodesk 123D Make) and UML language to perform the study concept of the game.

Key words : Monument Puzzle, 3D game, 3D, Android, iOS, Unity 3D, C#.