D-BOOKER

II. DÉVELOPPER POUR ANDROID & iOS

CRÉEZ DES JEUX DE A À Z AVEC UNITY®

(version 5)

Anthony Cardinale

Créez des jeux de A à Z avec Unity

II. Développer pour Android & iOS

par
Anthony Cardinale

D-BOOKER
éditions

Aperçu général de l'ouvrage

Le livre que vous avez sous les yeux constitue un module autonome de l'ouvrage plus global, intitulé *Créez des jeux de A à Z avec Unity*. Chaque module vise à traiter un aspect de Unity. Le dessin ci-dessous vous donne un aperçu des thèmes traités dans l'un ou l'autre. Les chiffres indiquent le numéro de chapitre correspondant.

Créez des jeux de A à Z avec Unity - II. Développer pour Android & iOS
par Anthony Cardinale

ISBN (PDF) : 978-2-8227-0375-8

Publié par les Éditions D-BookeR, 5 rue Delouvain, 75019 Paris
www.d-booker.fr
contact@d-booker.fr

Ce module fait partie de l'ouvrage global intitulé : Créez des jeux de A à Z avec Unity
ISBN (HTML) : 978-2-8227-0342-0

Les exemples (téléchargeables ou non), sauf indication contraire, sont propriété des auteurs.

Mise en page : générée sous Calenco avec des XSLT développées par la société NeoDoc (www.neodoc.biz)
Image de couverture : © Unity Technologies – Reproduite avec leur aimable autorisation. Le nom Unity, son logo, ainsi que tous les noms de produits ou images s'y rapportant sont des marques déposées et relèvent de la propriété unique de Unity Technologies. Ce livre a été écrit en toute indépendance, il n'est issu d'aucun partenariat avec Unity Technologies ou l'une de ses filiales, et son contenu ne relève que de la responsabilité de son auteur et de l'éditeur.

Date de publication : 25/06/2015
Édition : 1
Version : 1

Table des matières

À propos de l'auteur

Anthony Cardinale

Développeur passionné de jeux vidéo, Anthony utilise Unity depuis 2008. À cette date, le logiciel commence tout juste à se faire connaître en France, et il compte parmi les premiers à l'adopter. Auteur de plus d'une trentaine de jeux publiés sur Google Play, l'App Store ou Windows Phone Store, il s'est peu à peu spécialisé dans les applications à destination des mobiles et tablettes, après avoir également développé pour le web et les PC.

En parallèle, Anthony maintient différents sites d'e-learning. Il a réalisé à ce jour plus de 200 heures de formation vidéo sur la création de jeux avec Unity, diffusées via son site internet Formation facile.

Préface

par Mathieu Muller

Ingénieur d'application chez Unity Technologies

Unity, au-delà d'être un outil complet de création de médias interactifs 2D et 3D multi-plateformes, de sa communauté de plusieurs centaines de milliers d'utilisateurs ou de son Asset Store, me semble tout à fait remarquable à deux égards. Le premier est sa polyvalence, tant en genres qu'en types d'utilisateurs. Il suffit de consulter la galerie des contenus "made with unity" et de s'intéresser à leurs auteurs pour s'apercevoir d'une part que tous les styles s'y retrouvent, du jeu de plateforme 2D à l'œuvre d'art numérique interactive en passant par le configurateur industriel 3D, d'autre part, que leurs créateurs peuvent être jeunes étudiants ou bien vétérans de l'industrie du jeu vidéo, hommes ou femmes, artistes ou développeurs. Le second constat est encore plus surprenant : il arrive régulièrement qu'un passant interpelle un employé Unity qui porte un t-shirt de l'entreprise pour le remercier. Pour quelle raison ? Car cet outil lui a ouvert de nouvelles perspectives, permit de réaliser son rêve de faire son propre jeu ou bien de rendre ses journées professionnelles ou ses temps libres plus créatifs.

Mais aussi accessible soit-il, tout outil a besoin d'être maîtrisé. Tout comme la guitare ou le piano, les premières notes donneront vite satisfaction, mais seul le travail permettra de créer les plus riches compositions.

Unity propose un manuel complet, des tutoriaux, des webinars hebdomadaires, des forums alimentés par des centaines de milliers d'utilisateurs. Alors pourquoi un livre ? La première raison d'être de cet ouvrage est que l'essentiel de ce matériel n'est disponible qu'en anglais. La seconde et la plus romantique est que chacun a son média préféré d'apprentissage ou bien, pour aller plus loin encore, que le bon média au bon moment peut changer un bout de sa vie. Un livre, même technique, est bien plus qu'un manuel utilisateur, un tutoriel, un webinar ou un forum. C'est une histoire, un compagnon que l'on emporte dans le train ou sur un banc, qui nous remplit d'idées et d'émotions et les connecte à notre environnement lorsqu'on lève la tête du bouquin.

L'histoire c'est Anthony Cardinale qui la raconte. Anthony a consacré des milliers d'heures de sa vie à faire partager sa passion pour le jeu. Tournez cette page et suivez-le. Bienvenue dans ce nouveau monde, et bon voyage à la découverte et la maîtrise de l'infini des possibles des médias interactifs.

Paris, le 07 juin 2015

Introduction

Les jeux pour mobiles sont de plus en plus présents dans notre vie. Si vous souhaitez développer des jeux, vous ne pouvez pas passer à côté de ce marché.

L'apparition des smartphones en 2007 ont conduit les développeurs à proposer une nouvelle façon de se divertir avec des jeux pour téléphones. Aujourd'hui, les ventes d'applications mobiles ont largement dépassé les ventes de jeux pour consoles portables. En quelques chiffres, ce sont plus d'un million d'applications qui sont en ligne et pour l'année 2014 ces applications ont été téléchargées plus de 130 milliards de fois (en 2017 ce sera 250 milliards de téléchargements). Cela représente plus de 30 milliards de dollars de revenus. En France, ce ne sont pas moins de 27 millions de mobinautes qui utilisent régulièrement des applications mobiles.

Beaucoup de jeux sont aujourd'hui développés avec le logiciel Unity. Grands et petits studios l'utilisent pour le développement de leurs jeux mobiles car cet outil est très puissant, simple d'utilisation et permet d'exporter ses applications vers les différents systèmes d'exploitation sans avoir besoin de les développer plusieurs fois. De plus, Unity est gratuit même si vous publiez des jeux commerciaux. Plus de la moitié des jeux mobiles sont développés avec Unity. Gameloft, Microsoft, Ubisoft, EA, Disney... tous utilisent Unity pour certains de leurs jeux. Alors pourquoi pas vous ?

J'utilise Unity depuis quelques années. Aujourd'hui je ne développe presque que des jeux pour mobiles. En effet, le marché des applications mobiles est beaucoup plus accessible que celui des consoles. Les licences développeurs sont très abordables (25€ pour une licence développeur Android à vie) et Unity est gratuit. Tout le monde peut se lancer dans le développement de jeux grâce aux nouveaux outils et aux nouvelles plateformes.

Dans ce livre, je vais vous montrer comment développer efficacement des jeux mobiles en me basant sur mon expérience. Au fil des chapitres et des connaissances acquises, nous réaliserons de A à Z, jusqu'à sa publication sur Google Play, un jeu de plateforme 3D. Je vous montrerai comment optimiser vos jeux et comment adapter le gameplay aux écrans tactiles des mobiles.

Le chiffre d'affaires généré par ces applications et jeux mobiles est faramineux mais la concurrence est devenue très rude et il est de plus en plus difficile de se faire une place sur les stores. Pour vous donner un ordre d'idée, ce sont plus de 10 000 nouvelles applications qui sont publiées chaque jour. Figurer dans le TOP 50 voire le TOP 100 n'est pas facile et si votre application n'est pas dans le classement des meilleures appli-

cations, elle ne sera pas téléchargée. Dans ce livre, je vous donnerai des astuces qui vous permettront de maximiser vos chances et d'être le plus visible possible.

Les utilisateurs sont principalement répartis entre les systèmes Android et iOS. Ces deux plateformes représentent plus de 95 % du marché, ce sont donc pour ces plateformes que nous développerons nos applications. Je vous expliquerai comment changer de plateforme (Android/iOS) et exporter vers ces deux plateformes.

Je vous montrerai également comment monétiser vos applications et comment suivre l'évolution des téléchargements et des revenus qu'elles génèrent. Nous verrons enfin comment mettre en avant votre jeu sur les différentes plateformes de vente afin d'augmenter vos chances de décoller.

1. Ce dont vous avez besoin

Ce module suppose que vous avez déjà utilisé Unity et que vous connaissez les bases de son utilisation et de la création de scripts avec C#. Une connaissance du logiciel, même sommaire est requise. Si vous ne l'avez pas, je vous invite à lire le premier module Votre premier jeu PC.

Pour suivre ce livre, vous avez besoin de télécharger et d'installer le logiciel Unity sur votre machine. Vous devez également télécharger les sources des exemples sur mon site ou celui de l'éditeur. Nous allons développer une application Android. Nos exemples seront donc basés sur le système Android (plus accessible et désormais plus répandu qu'iOS avec 60 % du marché). Je ferai également un comparatif avec iOS et je vous donnerai les astuces pour adapter votre projet pour l'export vers iPhone ou iPad.

Enfin, vous devez disposer d'un téléphone ou d'une tablette tactile pour tester votre application.

Pour toute remarque, suggestion, question, vous pouvez me contacter via mon site web http://anthony-cardinale.fr.

2. Présentation des différents chapitres

Ce livre va vous permettre d'approfondir votre connaissance de Unity en vue de concevoir entièrement des jeux pour mobiles. Nous allons développer un jeu de plateformes 3D utilisant l'accéléromètre ainsi que l'écran tactile du mobile.

Chapitre 1 - Outils de développement mobile

Dans ce chapitre, nous allons mettre en place l'environnement de développement et passer en revue les différents outils requis pour le développement Android et iOS.

Chapitre 2 - Importation des ressources

Nous allons préparer le projet et importer les ressources dont nous aurons besoin pour développer notre jeu. Pour cela, nous établirons un mini cahier des charges.

Chapitre 3 - Compiler et tester

Nous allons voir comment configurer les outils qui vont nous permettre de compiler et de tester le jeu, au fur et à mesure de son développement, en direct sur notre téléphone ou notre tablette.

Chapitre 4 - Contrôles spécifiques au téléphone

Dans ce chapitre, je vais vous montrer comment gérer l'écran tactile de votre téléphone et utiliser l'accéléromètre pour pouvoir mettre en place le gameplay de notre jeu.

Chapitre 5 - Les niveaux

Vous apprendrez à concevoir des niveaux (level design) adaptés et optimisés pour les appareils mobiles. Nous créerons quelques niveaux avec une difficulté progressive pour rendre le jeu plus attractif.

Chapitre 6 - Créer ses propres modèles 3D

La modélisation 3D est une étape incontournable pendant le développement d'un jeu. C'est pourquoi je vous montrerai quelques astuces très utiles qui vous aideront à modéliser des objets 3D optimisés pour les mobiles.

Chapitre 7 - Animations et effets

Il est important de rendre vos jeux vivants. C'est ce que vous allez apprendre à faire dans ce chapitre en recourant à des systèmes de particules personnalisés et en créant vous-même des animations.

Chapitre 8 - Gestion des collisions

Nous allons mettre en place un système de collisions qui va permettre de ramasser des objets, d'ouvrir des portes ou de perdre si le joueur touche un objet mortel.

Chapitre 9 - Fixer des objectifs

Afin de donner de l'intérêt à notre jeu, nous allons mettre en place des missions ou objectifs que le joueur devra réaliser pour pouvoir passer au niveau suivant.

Chapitre 10 - Optimiser les scènes

L'optimisation d'un jeu doit se faire en amont. C'est pourquoi nous allons apprendre à optimiser nos scènes afin d'éviter tout ralentissement/bug lorsque le jeu sera publié.

Chapitre 11 - L'interface utilisateur

L'interface utilisateur (UI) correspond au menu principal ou au menu pause de votre jeu. Nous allons découvrir comment créer des menus pour nos jeux ainsi qu'une barre de vie.

Chapitre 12 - Sauvegarder des informations

Dans ce chapitre, vous allez apprendre à gérer les sauvegardes du joueur. Vous verrez comment stocker en mémoire des données comme les points du joueur, les niveaux débloqués, etc.

Chapitre 13 - Publier son jeu

Nous allons publier une V1 de notre application. Cette publication permettra d'obtenir une URL et un code d'identification indispensables pour mettre en place des achats intégrés ou des publicités.

Chapitre 14 - Monétiser son jeu

Plus de 90 % des jeux sont gratuits. Une manière de les monétiser est de diffuser de la publicité à l'intérieur. Cela vous rapportera de l'argent si le joueur clique sur la publicité. Nous allons apprendre à mettre en place des annonces dans un jeu.

Chapitre 15 - Les services de jeux

Afin de rendre votre jeu intéressant et de mettre les joueurs en compétition, nous allons voir comment intégrer un système de succès ainsi qu'un classement mondial.

Chapitre 16 - Améliorer son application

Il est important d'analyser vos joueurs, de suivre les revenus générés et de proposer des mises à jour de votre jeu. Dans ce dernier chapitre, nous verrons comment se passe la maintenance d'un jeu et sa mise en avant sur le store.

3. Accès aux vidéos

Ce module contient quelques vidéos.

Dans la version en ligne, celles-ci sont intégrées à votre page, et votre navigateur ira chercher de lui-même le format qu'il supporte.

Dans les versions téléchargeables (EPUB, PDF), un clic sur l'image vous redirigera vers la vidéo en ligne au format MP4. Si votre navigateur par défaut ne supporte pas nativement ce format, modifiez à la main l'extension du fichier dans l'URL en remplaçant `.mp4` par `.webm`. Et n'oubliez pas d'allumer vos haut-parleurs !

Note › Avant de cliquer, assurez-vous que le pointeur de votre souris s'est changé en main.

Vous pouvez aussi consulter directement la galerie en ligne des vidéos.

4. URL raccourcies

Dans un souci de lisibilité, et pour pouvoir les maintenir à jour, nous avons pris le parti de remplacer toutes les adresses internet par ce qu'on appelle des URL raccourcies. Une fois que vous avez accédé à la page cible, nous vous invitons à l'enregistrer avec un marque-page si vous souhaitez y revenir fréquemment. Vous disposerez alors du lien direct. Si celui-ci se périme, n'hésitez pas à repasser par l'URL raccourcie. Si cette dernière aussi échoue, vous pouvez nous le signaler !

1

Outils de développement mobile

Dans ce premier chapitre, nous allons mettre en place l'environnement de développement requis pour créer des applications mobiles. Cela implique d'installer, en sus de Unity, plusieurs outils indispensables, étroitement liés aux plateformes ciblées. Leur configuration n'étant pas toujours aisée, nous accompagnerons nos explications de quelques captures vidéo commentées.

1.1. Préparation du matériel de test

Le développeur d'applications mobiles doit être équipé. Si vous voulez en faire votre métier, il faudra vous munir de différents smartphones et tablettes tournant sous plusieurs systèmes. C'est important pour tester vos applications sur tous les types d'appareils et d'écrans. Pour suivre ce livre, un simple téléphone ou une tablette Android suffira. Cela vous permettra de tester le jeu que nous allons développer et détecter d'éventuels soucis. Il vous faudra cependant un téléphone assez récent (écran large, accéléromètre, et de préférence une version d'Android supérieure à la 4.0.

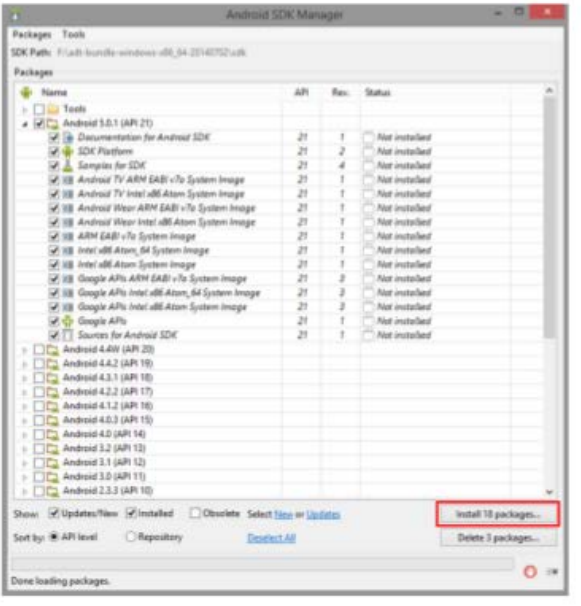
Vous devez activer le mode développeur sur votre mobile. Ce n'est pas très compliqué mais la procédure varie d'un appareil à l'autre. Je vous recommande donc de vous rendre sur Google et de rechercher "Activer mode développeur pour ..." (spécifiez votre modèle d'appareil). Une fois le mode développeur activé, vous serez en mesure de tester vos applications sur votre téléphone. Vous devez également autoriser dans les paramètres de sécurité les applications provenant de sources inconnues pour tester les applications non publiées. Enfin vous devez vous rendre dans le menu développeur et activer le débogage USB.

1.2. Préparation au développement Android

Pour pouvoir compiler votre jeu et le transférer sur votre mobile, vous aurez besoin du kit de développement Android, qui lui-même requiert le JDK (Java Development Kit) pour fonctionner. Téléchargez le JDK sur le site d'Oracle et installez-le sur votre ordinateur. Une fois terminé, rendez-vous sur le site des développeurs Android pour récupérer le kit de développement pour Android : vous pouvez télécharger soit le SDK seul, soit l'environnement de développement complet si vous souhaitez développer d'une manière générale des applications Android avec le langage Java.

Pour connecter votre ordinateur à votre téléphone, il vous faudra également le package Google USB Driver. Téléchargez-le depuis la page Google USB Driver ou passez directement par l'Android SDK Manager, qui centralise le téléchargement des ressources complémentaires. Vous pouvez d'ailleurs y télécharger aussi les fichiers permettant de développer pour plusieurs versions d'Android ainsi que divers plug-ins.

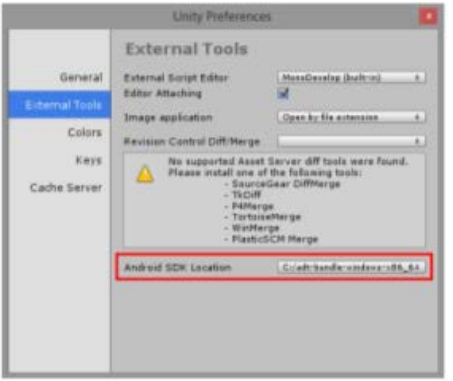
Figure 1.1 : Installation des dépendances supplémentaires



Pour télécharger et installer des ressources, il vous suffit de les cocher et de cliquer sur le bouton INSTALL PACKAGES.

Enfin, vous devez indiquer à Unity où se trouve le dossier SDK d'Android pour que les applications puissent être compilées par son intermédiaire. Pour cela, ouvrez le menu EDIT/PREFERENCES et spécifiez l'emplacement du SDK à l'onglet EXTERNAL TOOLS.

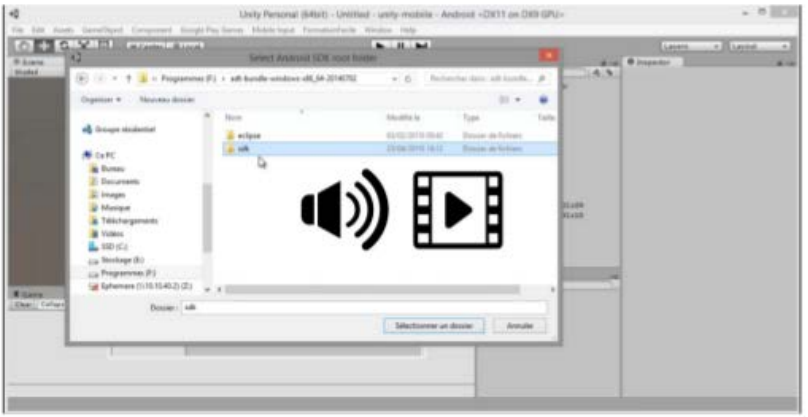
Figure 1.2 : Ajout du chemin du SDK aux préférences de Unity



***Note** : Vous pouvez retrouver toutes les informations de configuration dans la documentation en ligne de Unity.*

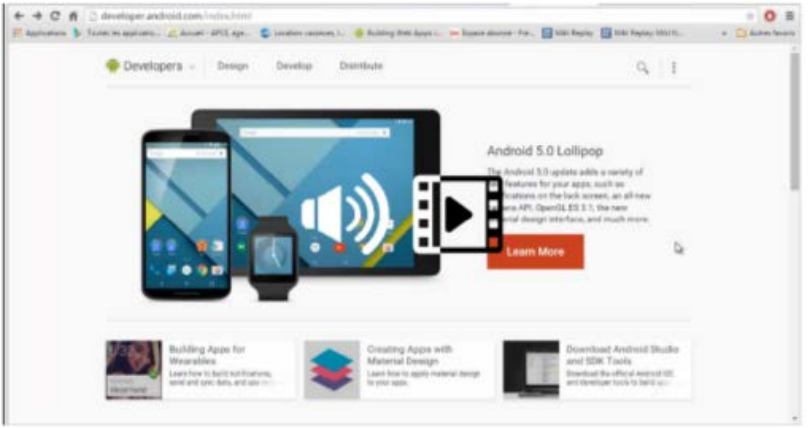
Si vous avez eu un peu de mal à reproduire les manipulations précédemment décrites, je vous invite à visualiser le tutoriel vidéo suivant.

Figure 1.3 : Configuration de l'environnement de développement Android (vidéo)



Pour publier une application sur Google Play (Android), vous devez vous enregistrer en tant que développeur Google. Cela coûte 25 € pour une licence à vie. Pour obtenir votre licence, vous devez vous rendre sur le site officiel des développeurs Android et obtenir un accès à la console développeur. L'obtention de votre licence développeur est assez simple et rapide.

Figure 1.4 : Tour d'horizon de l'espace développeur Google (vidéo)



1.3. Préparation au développement iOS

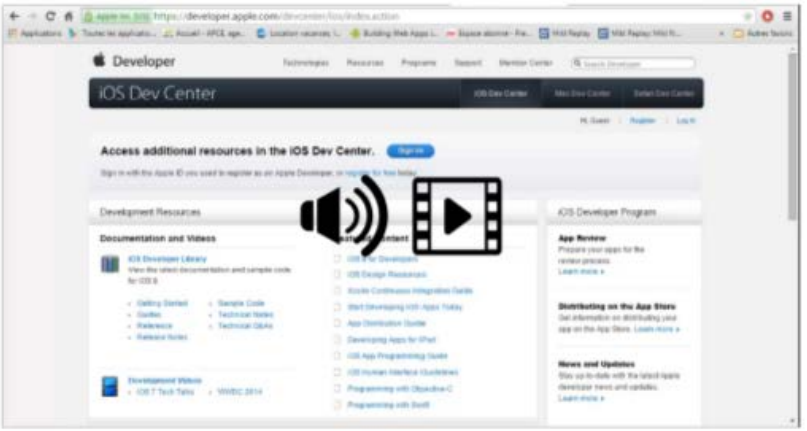
Si vous êtes sous Mac OS et que vous souhaitez développer des applications iOS, vous aurez à télécharger le logiciel Xcode sur l'App Store et souscrire au programme développeur iOS sur le site officiel des développeurs iOS. Ce programme vous coûtera 100 € à l'année. Il permet de tester vos applications sur vos appareils, de publier des applications sur le store et de suivre vos statistiques. Sans ce programme, vous ne pourrez pas développer d'applications iOS. Je vous montrerai par la suite comment enregistrer votre appareil pour installer vos applications et les tester.

Pour développer des jeux iPhone

Vous devez impérativement développer sous Mac OS si vous souhaitez développer des applications iPhone ou iPad. Il est impossible de compiler une application iOS si vous êtes sous Windows. Apple oblige les développeurs à passer par leurs systèmes.

La vidéo suivante passe en revue les différents outils dont vous aurez besoin pour développez des jeux iOS.

Figure 1.5 : Outils de développement iOS (vidéo)



1.4. Combien ça coûte ?

Pour développer des jeux Android, vous n'aurez que 25 € à débourser, ce qui est très correct. Pour développer des jeux iOS il faudra débourser 100 € par an, ce qui est un peu plus cher mais qui peut être vite rentabilisé si vous obtenez plus de 100 000 téléchargements. La version gratuite de Unity vous permettra de développer vos jeux sans soucis donc vous n'avez rien de plus à acheter. Bien sûr, si vous souhaitez acheter des modèles 3D, des sons ou des scripts, il faudra investir un peu d'argent pour financer votre jeu.

Vous avez désormais tous les outils nécessaires au développement d'applications mobiles avec Unity. Pour les développeurs Apple, la procédure est un peu particulière car il est impossible de compiler directement une application iPhone. Apple nous oblige à passer par Xcode pour finaliser la compilation et pour envoyer l'application sur iTunes. Nous y reviendrons plus tard.

Dans le prochain chapitre nous allons importer les ressources indispensables au développement de jeux mobiles, nous verrons comment les utiliser et comment les tester.

Dans ce chapitre, vous avez :

- considéré les enjeux du marché des jeux mobiles ;
- téléchargé et installé tous les outils de développement ;
- configuré votre téléphone pour pouvoir tester vos applications ;
- obtenu une licence développeur Android.

2

Importation des ressources

Comme vous le savez, pour développer un jeu nous avons besoin de ressources. Les ressources peuvent être des éléments visuels comme des personnages 3D mais aussi des sons ou des textures. Lorsque vous développez un jeu vidéo, vous devez toujours faire très attention à utiliser des ressources optimisées pour que votre jeu soit fluide quelle que soit la plateforme. Cette règle est encore plus importante lorsque vous développez des jeux mobiles, en effet, les mobiles sont beaucoup moins puissants que les ordinateurs et il faudra utiliser uniquement des ressources très optimisées. Les ressources optimisées sont :

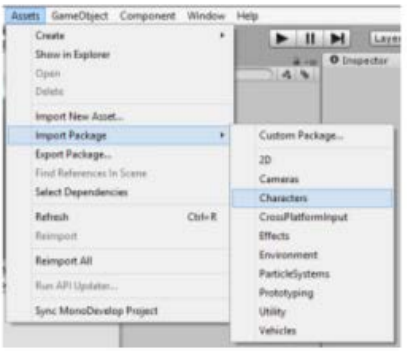
- des modèles 3D *low poly* ;
- des textures compressées ;
- des fichiers au format optimal (JPG, OGG, FBX, etc.).

Notre objectif sera de créer un jeu qui tournera au minimum en 30 images par seconde (60 dans l'idéal) et qui ne dépassera pas les 50Mb une fois compilé. Il devra tourner sur les appareils Android récents mais aussi être compatibles avec le maximum d'appareils plus anciens. Voyons maintenant les ressources dont nous allons avoir besoin pour nous lancer.

2.1. Les assets pour mobiles

La première chose que vous devez importer lorsque vous souhaitez créer un jeu à destination des smartphones et tablettes, ce sont des assets standards optimisés et adaptés aux terminaux mobiles. Les différents packages proposés par Unity sont désormais tous compatibles avec les mobiles depuis la version 5. Ils permettent de gérer l'écran tactile, de gérer l'accéléromètre et de mettre en place un gameplay avec un personnage contrôlable par des joysticks virtuels. Pour importer ces packages vous devez vous rendre dans le menu ASSETS/IMPORT PACKAGES/....

Figure 2.1 : Importation des ressources



Ces ressources proposées par le logiciel sont gratuites même pour vos projets commerciaux. Les nouvelles ressources sont compatibles à la fois avec les ordinateurs, les consoles et les mobiles. Unity met à notre disposition un script permettant d'adapter votre jeu en fonction de l'appareil sur lequel il est installé. En d'autres mots, si vous créez un jeu PC, le personnage sera contrôlable avec le clavier ; si vous créez un jeu mobile, il sera contrôlable avec l'écran tactile. Cela se fait automatiquement. Dans un premier temps, je vais vous présenter l'ensemble des ressources puis nous verrons lesquelles vous aurez à importer pour notre projet d'exemple.

- Le package `2D` contient des ressources permettant de créer des jeux 2D. Vous trouverez une scène d'exemple avec un petit robot que vous pouvez contrôler au clavier ou via l'écran tactile d'un smartphone. Le gameplay proposé est simple, vous pouvez marcher, tourner ou sauter. Le personnage est entièrement animé et prêt à être utilisé.
- Le package `Cameras` contient des scripts et prefabs qui donnent la possibilité d'utiliser des caméras préconfigurées. Par exemple, une caméra qui va suivre le joueur.
- Le package `Characters` contient trois types de personnages préconfigurés. Un personnage en FPS (vue à la première personne), un personnage en vue à la troisième personne et une balle contrôlable avec l'accéléromètre.
- Le package `CrossPatformInput` contient l'outil permettant de rendre compatible vos jeux avec toutes les plateformes.
- Le package `Effects` contient des scripts permettant d'appliquer des effets visuels à votre caméra, par exemple rendre l'arrière-plan flou ou modifier la couleur.

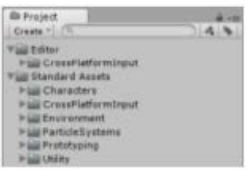
- Le package `Environment` contient des textures et ressources permettant de créer des décors comme de l'eau, des arbres, de l'herbe ou des montagnes.
- Le package `ParticleSystems` contient des systèmes de particules permettant de créer des effets visuels comme un feu, de la fumée, de l'eau, une tempête de sable, un feu d'artifice, etc.
- Le package `Prototyping` contient quelques modèles 3D comme des sols, des cubes, des escaliers ou maisons qui vont vous permettre de prototyper vos niveaux avec de simples modèles 3D.
- Le package `Utility` contient divers scripts qui effectuent des tâches souvent requises. Vous trouverez par exemple un compteur d'images par seconde, des scripts d'activation/désactivation d'objets, des scripts permettant de gérer le mouvement, etc.
- Le package `Vehicles` contient des véhicules préconfigurés comme une voiture ou un avion que vous pouvez utiliser si vous souhaitez développer un jeu de course ou de simulation.

Pour notre exemple, nous n'aurons pas besoin de tous ces packages. Nous allons développer un jeu d'aventure/plateforme dans lequel le joueur contrôlera une balle qui pourra rouler pour se déplacer afin de ramasser des objets et ouvrir des portes. Nous aurons donc besoin des packages suivants :

- `Characters` pour la création du personnage ;
- `Environment` pour le décor ;
- `ParticleSystems` pour les effets ;
- `Prototyping` pour le sol et les murs ;
- `Utility` juste au cas où.

Si vous avez importé ces packs, vous devriez avoir ceci sous les yeux :

Figure 2.2 : Les assets dont nous avons besoin

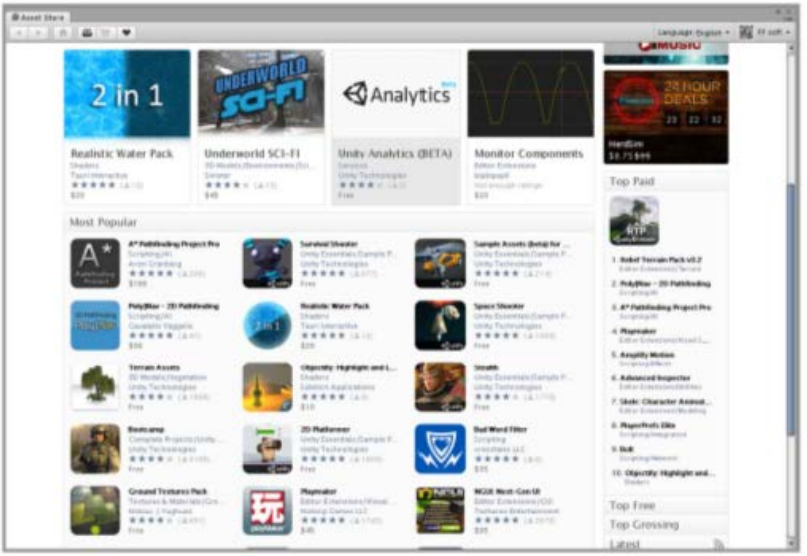


Note : Toutes ces ressources sont celles de la version 5 de Unity. Dans la version précédente, nous avions un package nommé STANDARD ASSETS (MOBILE) qui contenait des scènes d'exemples spécialement destinées au développement de jeux mobiles. Celles-ci ont été supprimées de la nouvelle version. Si vous souhaitez examiner ce qu'elles contenaient, vous pouvez télécharger le package sur mon site.

2.2. Les ressources externes

Pour que vous puissiez suivre ce livre sans difficulté, nous allons essayer de n'utiliser que des ressources de base fournies avec le logiciel. Cependant, vous aurez besoin de trouver des textures afin d'ajouter de la couleur à votre jeu. En effet, nous allons voir comment colorer le sol, les murs ou les objets des scènes 3D. Pour télécharger des textures, il existe des sites proposant des ressources gratuites comme opengameart ou encore l'Asset Store officiel de Unity.

Figure 2.3 : L'Asset Store de Unity



Sur l'Asset Store vous retrouverez des milliers de ressources et de scripts utilisables pour vos jeux. Beaucoup de ressources sont payantes mais certaines sont gratuites. Nous n'utiliserons que des ressources gratuites dans ce livre. Les ressources sont classées par catégories, par exemple MODÈLES 3D. Si vous souhaitez télécharger des modèles 3D,

vous devez faire très attention à ce que ces modèles soient *LOW POLY*, ce qui signifie qu'ils sont optimisés pour les mobiles.

Vous pouvez également utiliser vos propres ressources, conçues par vous-même ou par votre équipe, dès lors que vous utilisez des formats compatibles avec Unity. Nous verrons au chapitre Créer ses propres modèles 3D comment créer ses propres ressources afin d'éviter d'en acheter. Pour importer vos propres ressources, vous pouvez simplement les glisser/déposer dans la fenêtre projet.

2.3. Mini cahier des charges

Avant de nous lancer dans le développement, nous allons mettre sur papier les fonctionnalités de notre jeu. C'est une étape importante, par laquelle passe tout développeur, pour mieux s'organiser et anticiper les ressources dont nous aurons besoin. Bien sûr, nous n'allons pas réaliser un vrai cahier des charges de 100 pages mais nous allons décrire quelques fonctionnalités ainsi qu'une petite description du gameplay que nous allons mettre en place.

Nous devons garder en tête que notre jeu sera destiné aux mobiles, nous allons donc mettre en place un système basé sur l'accéléromètre du téléphone ainsi que l'utilisation de l'écran tactile. Comme il n'y a ni clavier ni joystick, le gameplay devra être très simple. Nous allons créer un jeu dans lequel le personnage sera une boule capable de se déplacer lorsque le joueur penchera le téléphone.

Au niveau des fonctionnalités, voici ce que nous allons devoir coder :

- la balle doit pouvoir ramasser des objets lorsqu'elle les touche ;
- la balle doit pouvoir changer de couleur afin d'ouvrir des portes ;
- le joueur doit rejoindre la fin du niveau sans tomber dans le vide pour passer au niveau suivant.

Notre application doit également offrir un certain confort à l'utilisateur et proposer des fonctionnalités essentielles comme par exemple :

- un menu principal intuitif ;
- un jeu optimisé pour tourner sur mobiles ;
- un système de classement mondial (en ligne) ;
- un système de réussites à débloquer (en ligne) ;
- un système de monétisation (publicités).

Il s'agit bien sûr d'une liste non exhaustive des différentes fonctionnalités que nous allons mettre en place. Maintenant que nous savons à quoi nous attendre, nous pouvons nous lancer dans le développement de notre projet !

Dans ce chapitre, nous avons importé quelques ressources qui vont faciliter la création de notre jeu. Vous avons vu :

- ce que sont des ressources optimisées ;
- les nouveaux packages de base de Unity ;
- le rôle d'un cahier des charges.

3

Compiler et tester

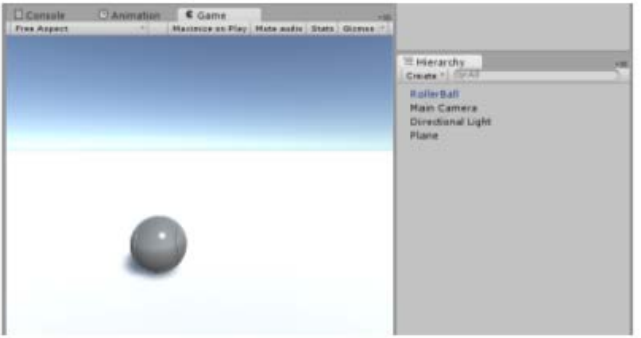
3.1. Préparation de la scène

Lorsque vous développez un jeu, vous devez le tester régulièrement afin de vérifier qu'il corresponde bien à ce qui est attendu lorsqu'il est installé sur un terminal mobile. Vous devez vérifier si les contrôles fonctionnent, si l'affichage est correct et traquer des erreurs visibles uniquement sur smartphone ou tablette. Il est recommandé de tester votre application à chaque fois que vous ajoutez une fonctionnalité majeure.

Il existe deux façons de tester votre application. Vous pouvez passer par l'outil `Unity remote` permettant de tester en direct votre jeu sur un mobile sans avoir à le compiler. Cette façon de procéder étant moins stable et moins fiable, nous ne l'utiliserons pas dans cet ouvrage. La seconde méthode consiste à compiler et installer l'application sur votre mobile. C'est ainsi que nous procéderons quand bien même la manipulation est un peu plus longue.

Nous allons commencer par créer une petite scène pour pouvoir réaliser un premier test. Nous allons simplement créer un sol et une balle contrôlable. Pour cela, cliquez sur GAMEOBJECT/3D OBJECT/PLANE pour créer un sol. Positionnez le sol en coordonnées 0, 0, 0 via l'inspector. Ajoutez ensuite le prefab de la balle à votre scène. Pour cela, faites-la glisser sur la scène à partir du dossier STANDARD ASSETS/CHARACTERS/ROLLERBALL. Vous devriez donc avoir une scène avec un sol et une balle comme sur la Figure 3.1.

Figure 3.1 : Scène de base



Afin de pouvoir vous déplacer dans la scène (faire rouler la balle), vous allez devoir ajouter le prefab `MobileTiltControlRig` à votre scène. Ce prefab permet d'ajouter des contrôles comme la gestion de l'accéléromètre de votre téléphone. Ce prefab se trouve dans le dossier STANDARD ASSETS/CROSSPLATFORMINPUT/PREFABS. Glissez/déposez-le n'importe où sur la scène puis enregistrez cette dernière, par exemple en lui donnant le nom `Jeu`, c'est nécessaire pour pouvoir ensuite compiler l'application.

3.2. Paramétrage du projet

Autre étape indispensable : la configuration du projet. Ouvrez la fenêtre FILE/BUILD SETTINGS et ajoutez la scène dans la liste des scènes à compiler en cliquant sur le bouton ADD CURRENT (voir ❶ sur la Figure 3.2). Cliquez ensuite sur ANDROID ❷ puis sur SWITCH PLATFORM ❸ pour spécifier qu'il s'agit d'un projet Android.

Figure 3.2 : Paramètres du projet

Le changement de plateforme peut être plus ou moins long selon le nombre de ressources que vous avez importées dans le projet. Une fois effectué, accédez aux paramètres de l'application en cliquant sur le bouton PLAYER SETTINGS.

Dans cette nouvelle fenêtre, vous allez renseigner des informations à propos de votre application. Pour commencer, vous allez spécifier le nom du développeur, le nom du jeu et l'icône associée :

Figure 3.3 : Paramètres de l'application

Dans la section RESOLUTION AND PRESENTATION, vous indiquerez si le jeu est en mode portrait ou en mode paysage. Vous pouvez également autoriser la rotation automatique de l'écran. Dans notre cas, il s'agit d'un jeu en mode paysage (LANDSCAPE) :

Figure 3.4 : Orientation de l'écran

Ensuite, dans la section OTHER SETTINGS, vous devez spécifier le BUNDLE IDENTIFIER de la façon suivante : `com.votreNom.NomJeu`.

***Note** : Le BUNDLE IDENTIFIER permet d'identifier de façon unique votre jeu en associant votre nom + nom du jeu. Il est utilisé par les magasins comme Google Play pour générer l'URL de la page de votre jeu par exemple. Vous devez respecter la syntaxe demandée.*

Figure 3.5 : Identification unique de l'application

Enfin, dans les PUBLISHING SETTINGS, vous devez enregistrer votre *keystore* ou en créer une nouvelle. La keystore permet de vous identifier de façon unique et d'associer les applications que vous avez crées à vous-même. Vous pouvez utiliser la même keystore pour toutes vos applications. Nous allons créer une nouvelle clé :

Figure 3.6 : Enregistrement de votre keystore

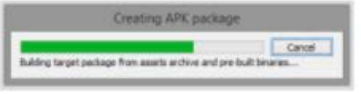
Nous sommes maintenant prêts à compiler notre scène.

3.3. Compilation du projet

Pensez à brancher votre téléphone à votre ordinateur via le câble USB. Si vous avez bien installé tous les outils de développement, que votre téléphone est en mode développeur et que vous avez les bons drivers, vous devriez être en mesure de compiler.

Pour compiler et installer une application sur votre téléphone, cliquez sur le bouton BUILD AND RUN tout en laissant le téléphone branché. Vous devez spécifier un nom pour votre fichier APK. Nous l'appellerons `Jeu.apk`.

Figure 3.7 : Création de l'APK



Une fois l'application compilée, elle est automatiquement installée sur votre mobile et le jeu se lance. Si vous déverrouillez votre téléphone, vous aurez la possibilité de faire rouler la balle sur le sol. Si tout marche normalement, c'est que votre configuration est bonne et que tous vos outils sont correctement configurés. Désormais vous pourrez la mettre à jour chaque fois que vous testerez une nouvelle fonctionnalité du jeu.

Dans ce chapitre, vous avez vu comment :

- préparer une scène ;
- configurer un projet ;
- paramétrer les *player settings* ;
- compiler et installer l'application.

4

Contrôles spécifiques au téléphone

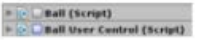
Dans ce chapitre, nous allons apprendre à intégrer l'accéléromètre et l'écran tactile pour contrôler la balle. Vous pourrez ainsi adapter votre gameplay aux téléphones et tablettes. Dans un premier temps, nous allons essayer de coder par nous-mêmes une fonction prenant en charge l'accéléromètre du téléphone. Nous utiliserons la scène d'exemple que nous avons mise en place au chapitre précédent. Puis nous verrons comment gérer l'écran tactile.

Commençons par voir comment programmer nos propres scripts pour coder la gestion du mouvement de la balle. Le prefab de la balle que nous avons choisi contenant déjà ce type de scripts, nous les désactiverons pour tester les nôtres. Pour cela, cliquez sur le prefab `MobileTiltControlRig` qui se trouve dans votre hiérarchie. Une fois le prefab sélectionné, décochez la case se trouvant tout en haut de l'inspector pour le désactiver :

Figure 4.1 : Désactivation d'un GameObject

Désactivez également les deux scripts qui se trouvent sur votre balle :

Figure 4.2 : Désactivation des scripts



Vous allez maintenant pouvoir coder vos propres scripts de mouvement.

4.1. Gérer l'accéléromètre

Maintenant que nous avons désactivé l'objet `MobileTiltControlRig` et les scripts qui s'y trouvent, nous allons pouvoir coder nos propres fonctions à partir de zéro.

Note : Cette section suppose que vous avez acquis les bases de la création d'un script. Si ce n'est pas le cas, reportez-vous au module I. Votre premier jeu PC.

Nous allons créer un nouveau script C# pour faire nos essais. Appelez-le `Mouvement-Balle` et ajoutez-le à la balle en le glissant/déposant sur celle-ci. Comme vous avez désactivé les anciens scripts, seul le vôtre sera pris en compte. Nous allons utiliser la fonction `Input.acceleration` pour détecter l'inclinaison du téléphone et la fonction `AddForce` pour faire avancer la balle. `AddForce` ne fonctionne que si la balle possède un `Rigidbody`. Le script complet pour notre exemple est le suivant :

```
using UnityEngine;
using System.Collections;

public class MouvementBalle : MonoBehaviour {

  void Update () {
    this.gameObject.GetComponent<Rigidbody>().AddForce(new
 Vector3(Input.acceleration.x, 0, Input.acceleration.y) * 1000 *
 Time.deltaTime);
      }
  }
```

Lorsque vous allez pencher votre téléphone, une force va propulser la balle dans la direction de l'inclinaison. Nous devons également paramétrer la caméra afin que celle-ci suive la balle. Pour cela, nous allons utiliser le script préconfiguré `FollowTarget`, qui se trouve dans les assets de base dans le dossier `Utility`. Placez ce script sur votre caméra et précisez les bonnes valeurs (GameObject de la balle et position de la caméra) :

Figure 4.3 : Paramétrage du script caméra

Votre script est prêt. Vous pouvez compiler et tester votre scène.

4.2. Gérer l'écran tactile

Afin de voir comment gérer l'écran tactile, nous allons ajouter la fonction de saut en suivant la documentation de Unity. La documentation nous indique qu'il faut utiliser la méthode `Input.touchCount` pour détecter si le joueur touche l'écran. Si vous suivez l'exemple donné dans la documentation, vous obtiendrez le code suivant :

```
using UnityEngine;
using System.Collections;

public class MouvementBalle : MonoBehaviour {

  void Update () {
    this.gameObject.GetComponent<Rigidbody>().AddForce(new
 Vector3(Input.acceleration.x, 0, Input.acceleration.y) * 1000 *
 Time.deltaTime);

    if (Input.touchCount > 0 && Input.GetTouch(0).phase ==
 TouchPhase.Moved) {
     this.gameObject.GetComponent<Rigidbody>().AddForce(new
 Vector3(0,100,0) * 100 * Time.deltaTime);
    }
  }
}
```

Pour tester votre jeu, vous devez une nouvelle fois le compiler et le lancer sur votre téléphone. Vous pouvez utiliser le raccourci clavier Ctrl+B pour faire une compilation rapide.

Il existe d'autres façons de détecter si le joueur touche l'écran tactile. Vous pouvez par exemple utiliser la fonction `OnMouseUp` pour détecter un clic. Sur mobile, un clic souris est équivalent à un doigt qui touche l'écran tactile. Si vous voulez vérifier cela, vous pouvez modifier le script `MouvementBalle` et ajouter cette nouvelle fonction en dessous de la fonction `Update` :

```
void OnMouseUp()
{
  this.GetComponent<Renderer>().material.color = Color.red;
}
```

Ce code indique que si le joueur touche la balle, nous accédons au `material` de celle-ci pour lui attribuer une nouvelle couleur, en l'occurrence la couleur rouge. Vous pouvez compiler et tester de nouveau l'application. Si vous touchez la balle, vous verrez le résultat suivant :

Figure 4.4 : Modification de la couleur de la balle

Gardez de côté la fonction permettant de changer la couleur de la balle, vous en aurez besoin par la suite pour notre projet. Nous verrons plus en détails la gestion de l'écran lorsque nous aborderons le chapitre L'interface utilisateur. Pour l'heure, je vous invite à désactiver les scripts que nous avons développés et à réactiver les scripts qui se trouvaient sur la balle. Les scripts fournis par Unity sont plus performants et mieux optimisés que ceux que nous avons créés, il est donc préférable d'utiliser ces derniers.

Dans ce chapitre, vous avez vu comment utiliser l'accéléromètre ainsi que l'écran tactile de votre smartphone. Vous pouvez donc :

- faire bouger la balle en penchant votre téléphone ;
- faire sauter la balle en touchant l'écran ;
- changer la couleur de la balle en touchant celle-ci.

5

Les niveaux

Dans ce chapitre, vous allez apprendre à concevoir des niveaux optimisés pour les appareils mobiles. Dans le cadre de notre projet, nous allons créer trois niveaux différents, mais bien sûr, pour un vrai jeu, je vous invite à en créer bien plus.

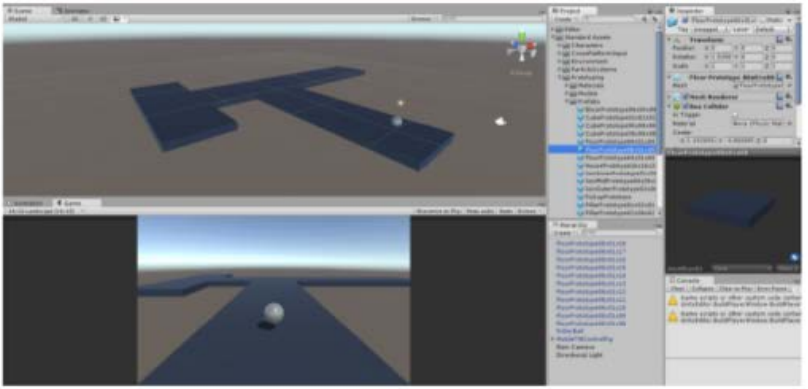
Nous allons repartir de la scène avec laquelle nous avons travaillé jusque-là en supprimant le sol présent qui était un simple PLANE. Vous devriez dont avoir juste une balle qui flotte dans le vide.

5.1. Création du sol

Nous allons commencer par créer le sol sur lequel la balle va pouvoir rouler. Vous devez toujours commencer par la modélisation des principaux éléments pour ensuite placer les objets interactifs (potions, pièces, portes...) pour enfin passer aux détails visuels.

Pour modéliser le sol, nous utiliserons des ressources fournies par Unity. Dans les assets de base, vous trouverez un dossier nommé PROTOTYPING/PREFABS. Il s'agit de simples cubes texturés qui vont nous servir à mettre en place le sol. Glissez-déposez l'un des prefabs (celui de votre choix) sur votre fenêtre de scène. Placez-le en position 0,0,0. Ensuite, dupliquez le cube que vous avez placé en le sélectionnant d'un clic puis en faisant la combinaison de touches Ctrl+D. Vous pouvez déplacer la copie avec les flèches représentant les trois axes 3D tout en appuyant sur la touche Ctrl de votre clavier afin de déplacer le nouveau cube cran par cran pour travailler avec plus de précision. Dans mon cas, j'ai utilisé environ dix cubes pour construire le niveau suivant :

Figure 5.1 : Création du sol

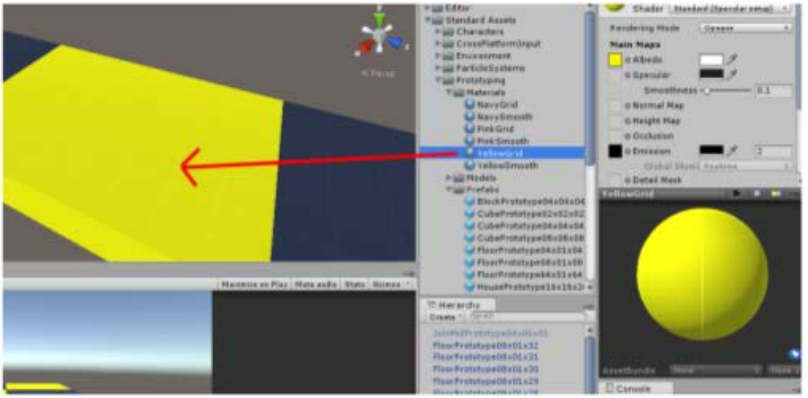


Comme vous pouvez le constater, nous pouvons élaborer un niveau avec quelques cubes. Notez que j'ai utilisé le prefab `FloorPrototype08x01x08`. Le mieux est de ne pas utiliser trop de prefabs différents : en effet, si vous utilisez plusieurs objets 3D et plusieurs textures, les calculs seront plus nombreux et le jeu sera plus lent. Bien sûr, avec les mobiles d'aujourd'hui vous pouvez utiliser une cinquantaine d'objets différents sans soucis.

Le but ici est d'élaborer des petits niveaux dans lesquels le joueur devra se déplacer pour ramasser des objets et pour ouvrir des portes. Il faut que vous augmentiez la difficulté pour chaque nouveau niveau. Par exemple vous pouvez faire des niveaux plus grands, mettre plus d'obstacles ou encore créer de petits labyrinthes. Nous allons créer les portes et les clés par la suite.

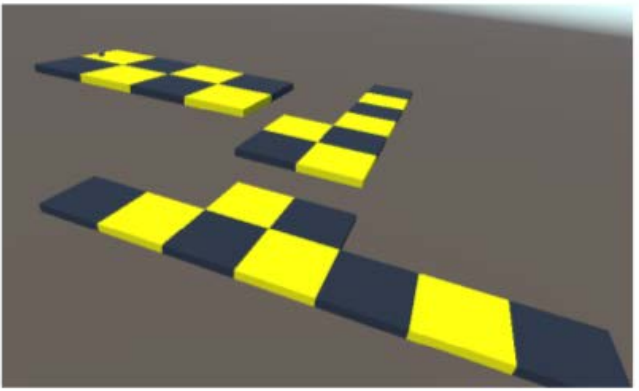
Pour changer le rendu visuel, vous pouvez utiliser les matériaux qui se trouvent dans le dossier MATERIALS. Glissez/déposez le materiau sur un objet afin de changer sa couleur :

Figure 5.2 : Modification de la couleur d'un objet



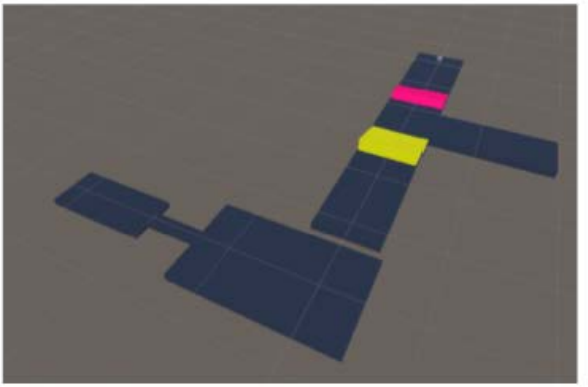
Vous pouvez par exemple utiliser une couleur différente pour votre second niveau comme c'est le cas ci-dessous :

Figure 5.3 : Création du niveau 2



Afin d'augmenter un peu la difficulté, le joueur sera dans l'obligation de sauter pour atteindre les autres plateformes. Si le joueur descend sans avoir récupéré la clé, il sera bloqué en bas par la porte et obligé de recommencer. Voilà une petite astuce pour ajouter de la difficulté. La Figure 5.4 représente le niveau 3 que j'ai également modélisé :

Figure 5.4 : Création du niveau 3



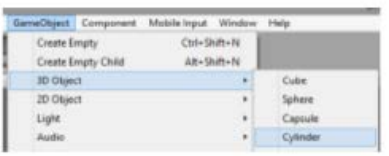
N'hésitez pas à modéliser plusieurs niveaux afin de créer un jeu complet avec une bonne durée de vie. Dans mon cas, je vais me limiter à trois niveaux que vous pouvez télécharger avec les sources fournies avec ce livre.

5.2. Création des objets interactifs

Pour rendre votre jeu intéressant, vous devez ajouter des objets interactifs. Ces objets peuvent être des pièces, des diamants ou toutes autres pierres précieuses. Le joueur doit gagner des points en ramassant ces objets. Pour notre jeu, nous allons créer des pots de peinture que le joueur pourra ramasser. Lorsque celui-ci ramassera un pot de peinture, la balle changera de couleur. Par exemple, si la balle touche un pot de peinture rouge, elle deviendra rouge. Cela va nous permettre de créer des énigmes. En effet, pour ouvrir la porte rouge, le joueur devra ramasser le pot rouge. Ce concept peut être creusé plus avant en ajoutant par exemple le mélange des couleurs primaires.

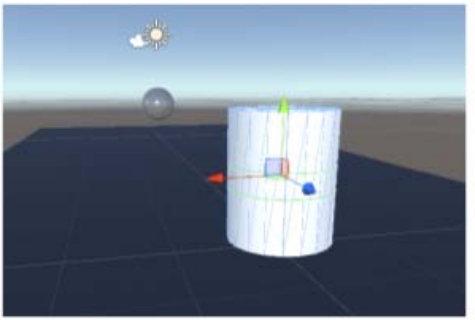
Vous allez commencer par modéliser le pot de peinture. Pour cela, créez un objet cylindrique en sélectionnant le menu GAMEOBJECT/3D OBJECT/CYLINDER :

Figure 5.5 : Création du pot de peinture



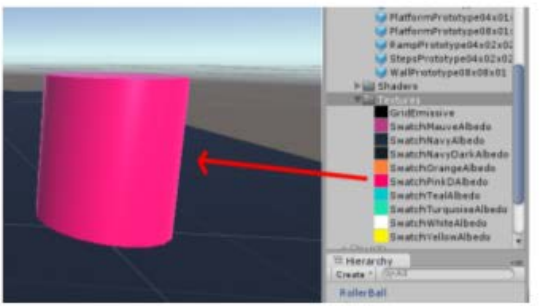
Le pot apparaît sur votre scène. Vous pouvez maintenant changer sa taille et sa position.

Figure 5.6 : Ajustement du pot de peinture



L'objectif est de créer quelques pots de peinture avec une couleur différente. Par exemple un pot bleu, un pot rose et un pot jaune. Vous trouverez des textures de couleur dans le dossier PROTOTYPING/TEXTURES. Appliquez la texture à l'objet en la faisant simplement glisser sur celui-ci :

Figure 5.7 : Ajout de la texture



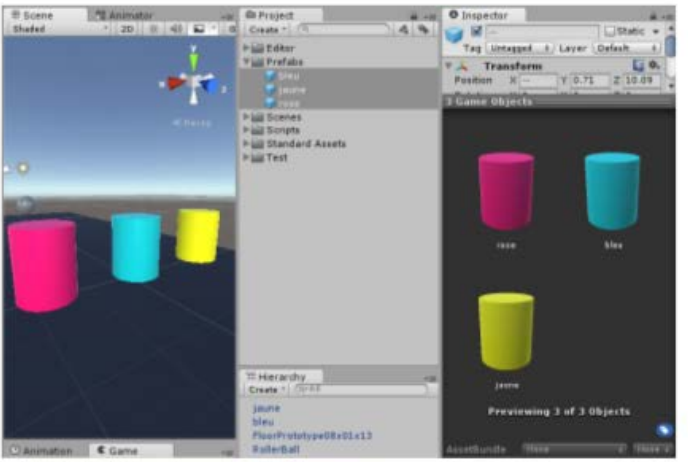
Afin de différencier les pots, nous allons modifier leur nom. Cela va nous permettre de savoir quel pot a été touché par le joueur. Changez donc le nom de votre pot en `rose` :

Figure 5.8 : Modification du nom

Lorsque nous détecterons une collision grâce au code, nous serons en mesure de récupérer le nom de l'objet touché. Si l'objet touché s'appelle rose, nous ferons en sorte de colorer la balle en rose et de faire disparaître le pot qui a été utilisé.

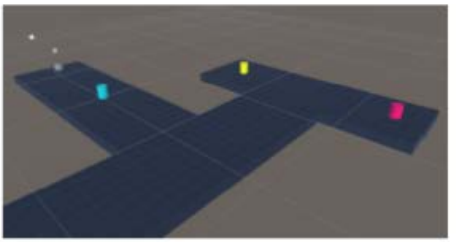
Je vous invite à créer d'autres pots de peinture avec d'autres couleurs. Pensez à changer le nom des pots en "bleu", "jaune", etc. Une fois que vous avez au moins trois pots, créez des prefabs de ces pots afin de pouvoir les réutiliser dans toutes vos scènes.

Figure 5.9 : Création des pots et des prefabs



Vous avez maintenant vos objets interactifs. Vous pouvez placer des pots de peinture un peu partout dans vos scènes afin que le joueur puisse les ramasser. Pensez bien à gêner le joueur en mettant des pots inutiles. En effet, si la porte finale est rose, le joueur devra être rose. Il faut évidemment un pot rose dans la scène mais vous pouvez ajouter des pots jaunes ou bleus afin de ralentir le joueur dans sa progression.

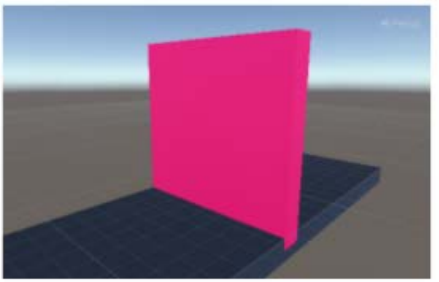
Figure 5.10 : Placement des pots



5.3. Création des portes

Nous allons maintenant nous occuper de créer les portes qui vont empêcher le joueur d'avancer. Le but est de mettre des portes dans vos niveaux pour ralentir le joueur. Une fois que le joueur arrive à la fin du niveau, il peut passer au niveau suivant. Pour créer les portes, nous allons utiliser le même objet que pour le sol. Nous devons juste lui appliquer une rotation pour qu'il soit perpendiculaire au sol :

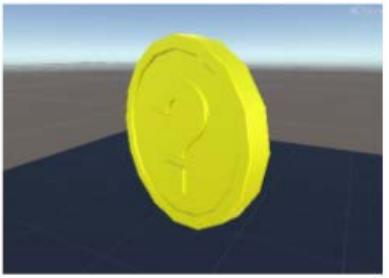
Figure 5.11 : Création de la porte



Ici la porte est rose. Le joueur devra donc être rose pour ouvrir cette porte. Pensez aussi à lui donner un nom pour pouvoir détecter la collision avec celle-ci. Nous pourrons par exemple la nommer `porterose`.

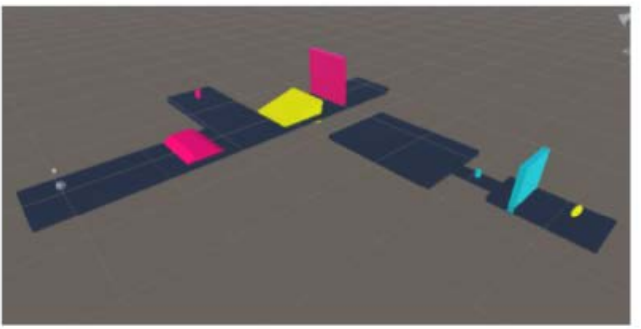
Vous pouvez par la même occasion créer l'objet représentant la fin du niveau. Si le joueur touche l'objet de fin, le niveau se termine et il peut passer au niveau suivant. Dans mon cas, je vais utiliser l'objet `PickupPrototype01x01x01` que vous pouvez retrouver dans le dossier `Models` des ressources de prototypage. J'ai placé l'objet de fin juste derrière la porte :

Figure 5.12 : Objet de fin de niveau



N'oubliez pas non plus de le renommer, nous l'appellerons ici `fin`. Ci-dessous un exemple de niveau complet avec sol, portes, pots et fin :

Figure 5.13 : Niveau complet



Une fois que vos niveaux sont terminés, vous pouvez passer au chapitre suivant. Vous retrouverez mes niveaux dans les sources téléchargeables du livre.

Dans ce chapitre, nous avons vu comment concevoir des niveaux complets avec :

- le sol ;
- les obstacles ;
- les objets interactifs ;
- la fin du niveau.

6

Créer ses propres modèles 3D

Dans le chapitre précédent, vous avez vu comment assembler des éléments pour créer des niveaux. Nous avons créé ensemble plusieurs scènes en utilisant des ressources gratuites livrées avec Unity. Dans ce chapitre, je vais faire un petit détour du côté des logiciels de modélisation 3D. En effet, vous serez probablement amené à créer vos propres objets 3D lorsque vous ne trouverez pas votre bonheur sur le net. Vous aurez donc à passer par un logiciel de modélisation pour créer vos modèles 3D. Je vais ainsi vous montrer quelques astuces qui vont vous permettre de créer des objets 3D simples utilisables dans Unity.

Ce chapitre n'est ni obligatoire ni indispensable pour la suite de ce livre ou de votre projet. Il s'agit d'une partie modélisation qui est importante, mais si vous ne souhaitez pas modéliser vos propres modèles 3D vous pouvez passer directement au chapitre suivant. Sa lecture toutefois vous apportera une meilleure compréhension des objets 3D et enrichira votre culture personnelle.

Afin d'illustrer mon exemple, je vais utiliser le logiciel Cinema 4D. Ce logiciel est payant mais il existe une version gratuite de démo. Vous pouvez aussi utiliser Blender 3D qui est un logiciel entièrement gratuit, les manipulations sont les mêmes. Je vous invite donc à télécharger un logiciel de modélisation 3D. Il en existe plusieurs, et chaque logiciel payant propose une version d'évaluation gratuite.

Ce chapitre sera également l'occasion de vous montrer comment optimiser vos modèles 3D pour le jeu vidéo. Plus votre modèle 3D est réaliste, plus il possède de polygones. Les polygones constituent toutes les faces d'un modèle 3D. Par exemple, un personnage 3D est souvent composé de 3000 polygones. La complexité d'un modèle 3D se mesure donc en polygones (sous Unity, les polygones sont découpés en triangles). Le but ici est de modéliser des objets en *low poly*, ce qui signifie *peu de polygones*. En effet, un mobile aura plus de mal à calculer un objet complexe, nous allons donc nous limiter à quelques polygones par objet.

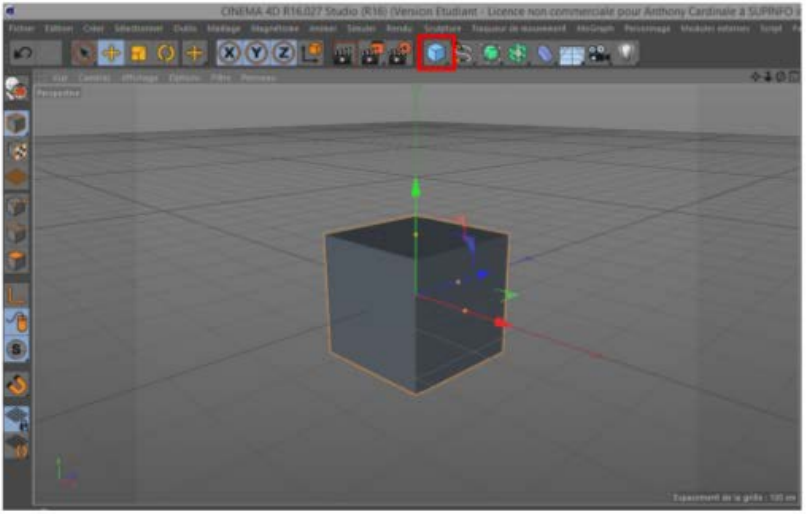
Pour cette petite introduction à la modélisation 3D, nous allons créer un modèle simple qui sera une plateforme pour notre jeu. Vous êtes libre de choisir la forme de votre plateforme et de la modéliser comme vous le souhaitez en suivant la technique que je vais vous présenter.

6.1. L'extrusion

L'extrusion est une technique permettant de créer des polygones en partant d'une face d'un objet. Par exemple, si vous avez un cube et que vous réalisez une extrusion, vous pourrez faire apparaître de nouvelles faces.

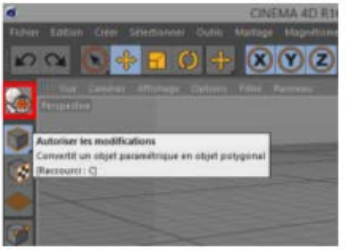
Pour réaliser une extrusion commençons par créer une primitive. Je vous invite à créer un cube 3D si vous voulez faire simple ou à créer un cylindre si vous voulez une forme plus complexe. Dans Cinema 4D, vous pouvez cliquer sur l'icône de cube dans le menu du haut pour faire apparaître un cube à l'écran.

Figure 6.1 : Création d'un cube



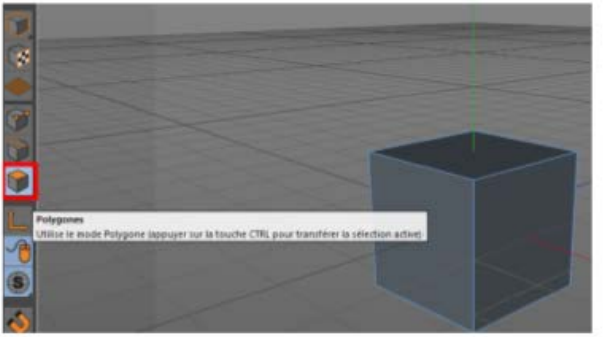
Dans Cinema 4D, pour modifier un objet vous devez le rendre éditable. Pour cela, cliquez sur le bouton permettant d'activer le mode édition.

Figure 6.2 : Activation du mode Édition



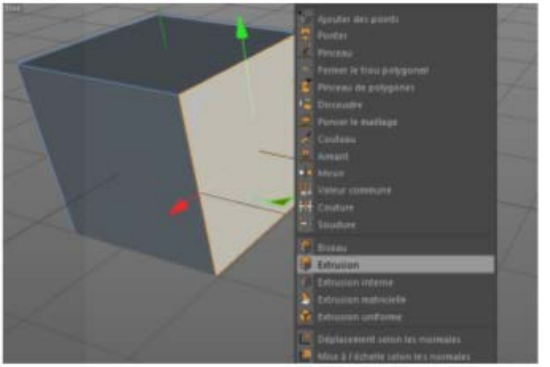
Nous pouvons travailler en mode points, ce qui permet de modifier les points (intersections des arêtes) des modèles 3D, ou travailler avec les arrêtes ou encore avec les polygones. Pour notre part, nous choisissons le mode Polygones.

Figure 6.3 : Activation du mode Polygones



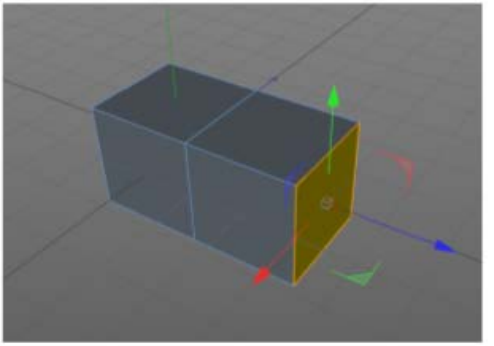
Ce mode va vous permettre de sélectionner un ou plusieurs polygones et d'appliquer des déformations sur ces derniers. Nous pouvons par exemple sélectionner un polygone sur le côté du cube et faire un clic droit/EXTRUSION.

Figure 6.4 : Activation de la fonction Extrusion



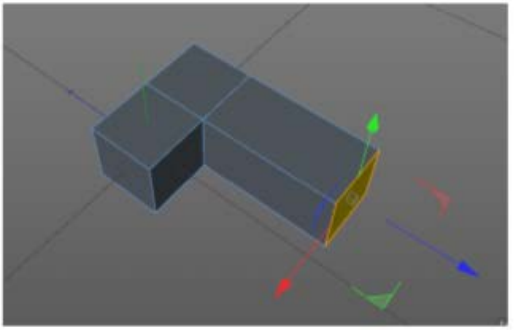
Vous pouvez maintenant utiliser les flèches pour appliquer l'extrusion sur la face sélectionnée et ainsi faire apparaître de nouvelles faces.

Figure 6.5 : Création de faces



Une fois que vous avez réalisé cette première extrusion, vous pouvez sélectionner une autre face et appliquer une nouvelle extrusion par exemple pour créer un angle.

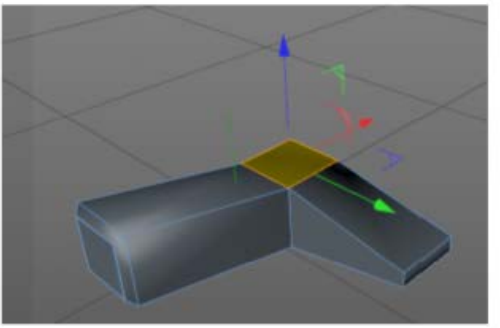
Figure 6.6 : Création d'un angle droit



6.2. Modification du modèle 3D

Vous pouvez facilement déformer un modèle 3D ou une partie de ce modèle en appliquant une translation, une rotation ou une modification de sa taille. Nous allons par exemple essayer de créer une pente. Pour cela, vous devez sélectionner une facette et utiliser les flèches pour la déplacer et la remonter.

Figure 6.7 : Déformation de l'objet

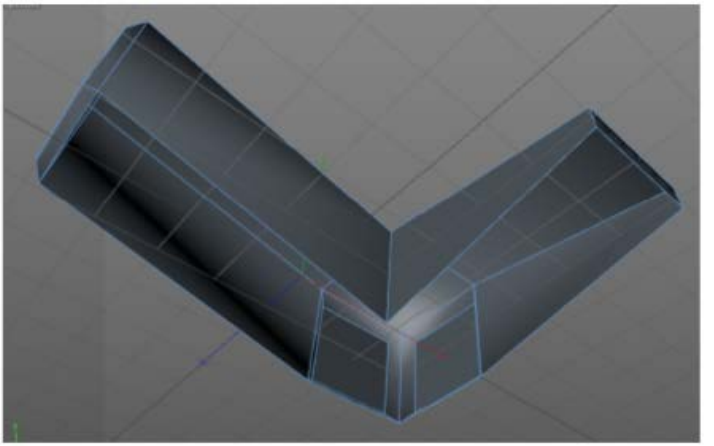


Vous pouvez allonger ou réduire la taille de certaines faces du modèle 3D. Ou encore créer d'autres extrusions afin de pouvoir appliquer de nouvelles déformations à l'objet. Une fois que vous avez terminé, vous pouvez passer à l'optimisation de votre modèle 3D.

6.3. Optimisation du modèle 3D

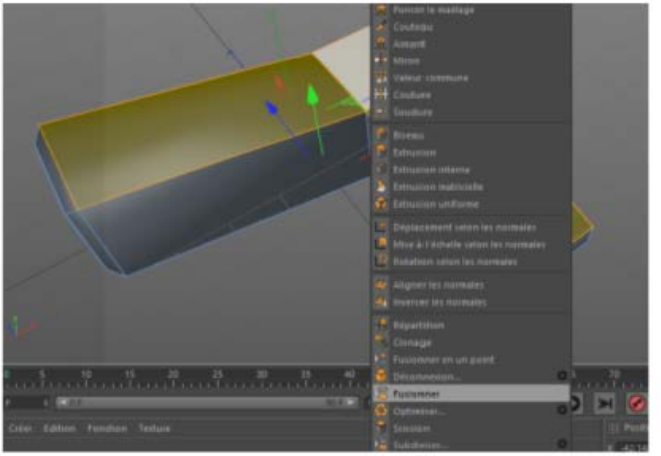
Notre modèle a beau être très simple, nous pouvons l'optimiser. Pour cela, il faut dans un premier temps rechercher les polygones qui ne seront jamais visibles par le joueur. Dans notre cas, la balle roulera sur le dessus de l'objet, elle ne verra donc jamais les polygones se trouvant dessous. Vous pouvez donc sélectionner ces derniers et les supprimer définitivement en appuyant sur la touche Suppr de votre clavier.

Figure 6.8 : Suppression de polygones



Une autre technique d'optimisation consiste à fusionner des polygones se trouvant sur un même plan. Par exemple dans notre cas, nous pouvons fusionner les polygones du dessus en les sélectionnant et en activant la fusion. Dans Cinema 4D, cela se fait par un clic droit/FUSIONNER.

Figure 6.9 : Fusion de polygones

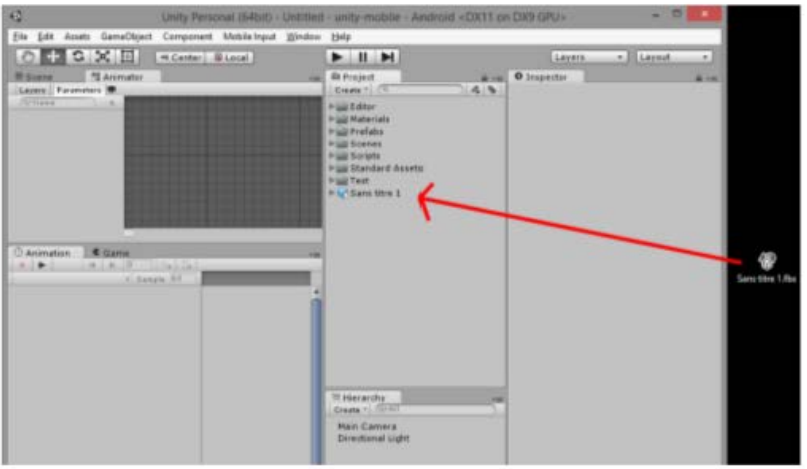


Une fois votre modèle optimisé, vous devez l'exporter au format FBX qui est le format le plus adapté à Unity. Même si d'autres formats sont supportés comme le OBJ, je vous invite à privilégier ce dernier pour utiliser vos modèles 3D dans Unity. La procédure d'export est sensiblement la même quel que soit le logiciel 3D. Allez dans le menu FICHIER/EXPORTER et choisissez le format FBX.

6.4. Importation et utilisation du modèle dans Unity

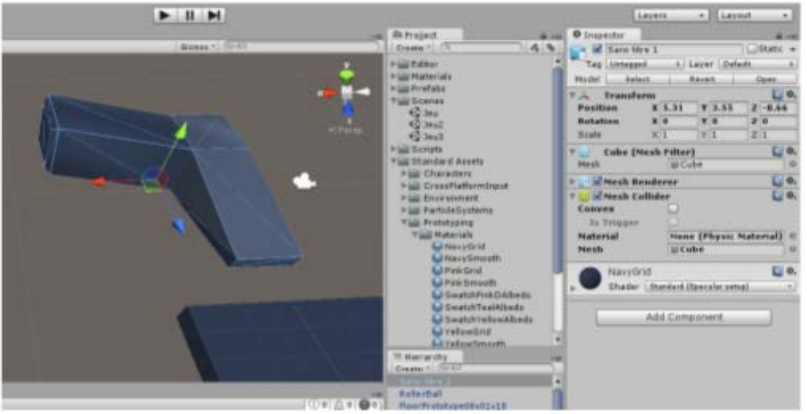
Vous vous demandez maintenant comment vous allez pouvoir utiliser cet objet 3D dans Unity. La réponse est simple : faites-le glisser depuis votre ordinateur vers la fenêtre de projet. Vous pouvez maintenant l'utiliser comme n'importe quel autre objet.

Figure 6.10 : Importation du modèle 3D



Glissez/déposez-le sur la fenêtre de la scène pour le faire apparaître dans votre monde virtuel. Il se comportera comme une primitive de base. Vous avez la possibilité de lui appliquer n'importe quelle texture, de lui ajouter un `mesh collider` pour activer la collision ou de lui ajouter vos scripts.

Figure 6.11 : Ajout de composants au modèle



Il n'est pas possible de modéliser des objets directement dans Unity, c'est pourquoi nous sommes passés par un autre logiciel. Cela nous a permis de voir comment on pouvait travailler en complément avec un logiciel tiers. Cette méthode vous permet de créer des modèles optimisés pour vos jeux mobiles. De plus, si vous voulez créer un jeu unique, il est toujours préférable de réaliser vos propres modèles, plutôt que d'utiliser des ressources glanées sur le net. Je vous invite à modéliser les pots de peinture avec un logiciel 3D pour vous entraîner, même si cela n'est pas obligatoire pour la suite du projet.

Dans ce chapitre, vous avez vu :

- comment réaliser un objet simple avec un logiciel de modélisation ;
- l'importance de l'optimisation des modèles ;
- comment utiliser votre modèle dans Unity.

7

Animations et effets

Pour rendre nos niveaux plus vivants, nous allons les animer. Cette étape est très importante car elle permet de faire en sorte que le joueur reste éveillé, que ses yeux soient toujours attirés vers de nouveaux éléments en mouvement.

Il existe plusieurs techniques permettant d'introduire du mouvement dans les scènes. Dans ce chapitre, nous allons vous présenter les deux principales méthodes pour créer des animations avec Unity : par un script ou via son outil d'animation intégré. Un autre moyen est d'ajouter un système de particules. Nous apprendrons ici à créer nos propres systèmes et au chapitre suivant vous verrez comment tirer parti des particules pour renforcer une collision.

7.1. Animer un objet par script

Vous pouvez animer un objet à l'aide d'un script C#. Pour notre exemple, nous allons animer la pièce jaune qui marque la fin du niveau. Pour l'instant, cette pièce est statique, nous voulons la faire tourner sur elle-même.

Commencez par créer un nouveau script C# que vous appellerez `rotationPiece`. Ouvrez ce script pour pouvoir écrire le code. Nous allons utiliser la fonction `Rotate` du composant Transform. Cette fonction permet de faire tourner un objet en spécifiant un sens et une vitesse.

***Note** : Le composant Transform est celui qui nous donne les informations de position, de taille et de rotation d'un objet dans l'espace 3D. Nous pouvons agir sur ce composant par script pour modifier les propriétés précitées.*

Voici ce qu'il faut écrire pour faire tourner la pièce :

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {

    public int vitesse = 10;

    void Update() {
```

```
            transform.Rotate(Vector3.right * Time.deltaTime * vitesse);
        }
    }
```

Ce script permet de faire tourner la pièce à l'infini à la vitesse que vous avez spécifiée. Pour le tester, placez-le sur la pièce ou n'importe quel autre objet que vous souhaitez animer de la sorte. Vous pouvez vous amuser à faire tourner certaines parties du sol afin de perturber le joueur et le faire tomber si vous le souhaitez. Si vous lancez le jeu, vous verrez que tous les objets qui possèdent ce script seront en mouvement.

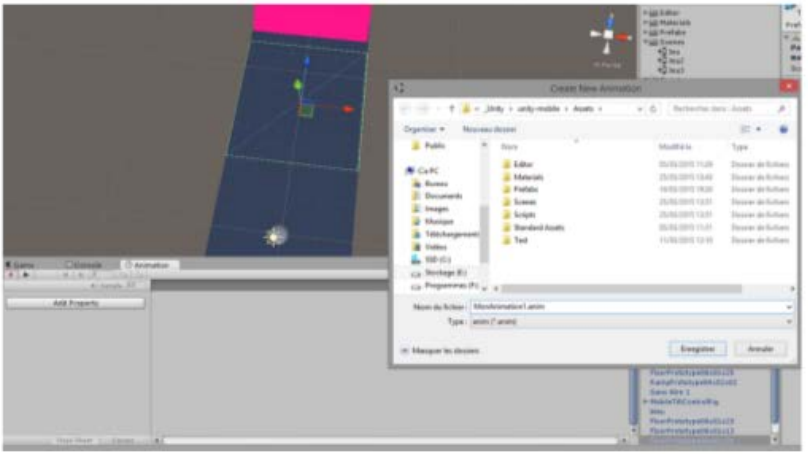
Il existe d'autres fonctions C# permettant d'appliquer un mouvement. La documentation vous aidera à les retrouver. Voyons maintenant comment utiliser l'outil d'animation intégré à Unity.

7.2. Animer avec l'outil d'animation

Unity intègre un outil puissant permettant de créer des animations de toutes sortes en quelques clics. Vous pouvez agir sur la taille, la position, la couleur, la rotation et toute sorte d'autres propriétés de l'objet. Cet outil se trouve dans le menu WINDOWS/ANIMATION. Une nouvelle fenêtre apparaît alors et vous permet de créer une animation et de l'appliquer à l'objet sélectionné.

Nous allons par exemple animer une case du sol en lui appliquant un mouvement de gauche à droite. Elle se décalera d'une unité sur le côté puis reprendra sa place. Ainsi le sol sera en mouvement et le joueur pourra tomber dans le vide s'il ne fait pas attention. Commencez donc par choisir la case que vous souhaitez animer. Ensuite, cliquez sur le cercle rouge de la fenêtre animation. Cela aura pour effet de lancer l'enregistrement de l'animation.

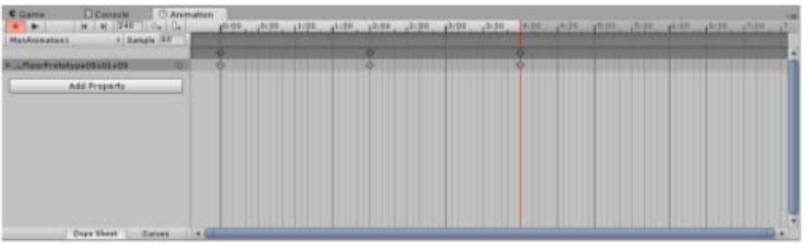
Figure 7.1 : Création d'une nouvelle animation



Unity crée un fichier qui contiendra toutes les informations de l'animation que vous allez créer, et vous demandera de lui attribuer un nom.

Tant que l'enregistrement est actif, vos actions seront sauvegardées dans la timeline. En d'autres mots, vous pouvez spécifier des caractéristiques (taille, position, couleur, rotation...) à un temps donné. Par exemple, si vous vous positionnez sur la timeline à la deuxième seconde et que vous déplacez la case d'une unité, de nouvelles clés seront enregistrées dans la timeline (voir Figure 7.2).

Figure 7.2 : Création des clés d'animation



***Note** : Les clés sont représentées par de petits losanges rouges. Lorsque vous créez une animation, les clés enregistrent toutes les modifications que vous apportez à un objet comme par exemple sa taille ou sa position. Le fichier animation créé sera capable de créer une transition entre chaque clé pour animer votre objet.*

Cela vous permet de dire où doit se trouver la case et à quel moment. Je vous propose de faire un mouvement de va-et-vient de gauche à droite pendant une durée totale de 4 ou 5 secondes. Pour cela, vous pouvez vous positionner à la seconde 2, décaler la case d'un cran et à la seconde 4, vous remettez la case à sa place initiale (la position qu'elle avait à la seconde 0). Cela va permettre d'obtenir une animation qui tournera en boucle.

Figure 7.3 : Animation finale

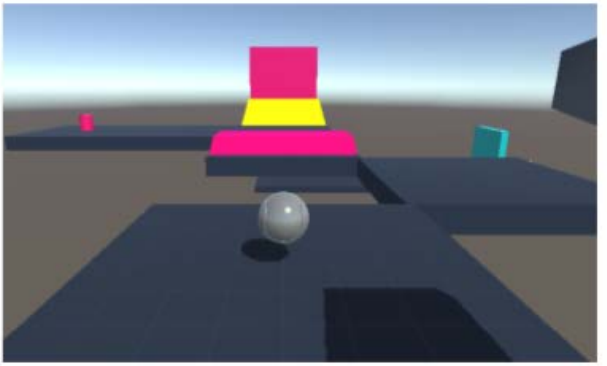
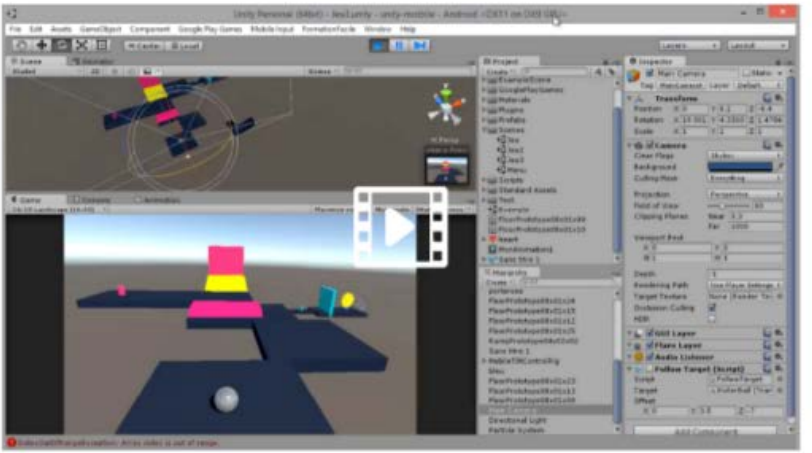


Figure 7.4 : Aperçu vidéo des animations du niveau



Pour terminer l'enregistrement, cliquez de nouveau sur le rond rouge. Votre animation sera automatiquement appliquée à l'objet qui était sélectionné pendant sa création. Vous pouvez aussi faire glisser le fichier de l'animation sur l'objet à animer. Voici ce qui devrait apparaître dans l'inspector :

Figure 7.5 : Animation dans l'inspector

Vous pouvez lancer le jeu et voir l'animation se jouer. Désormais la difficulté a augmenté et le joueur peut tomber dans le vide s'il ne fait pas attention.

Vous pouvez créer d'autres animations pour d'autres objets si vous le souhaitez. Cette étape n'est pas primordiale pour le fonctionnement du jeu mais est conseillée pour le rendre plus professionnel et moins statique.

7.3. Les particules

Les systèmes de particules vous permettent de créer des effets spéciaux comme par exemple de la fumée, du feu, des lucioles, des feuilles qui tombent, de la pluie, etc. Unity propose dans ses packs des systèmes prêts à être utilisés et c'est ce que nous utiliserons au chapitre Gérer les collisions. Cependant, je vais vous montrer comment créer vos propres particules en quelques minutes grâce à Unity.

Dans notre jeu par exemple, nous allons placer un système de particules à la fin du niveau pour que le joueur puisse repérer l'endroit où il doit se rendre. Nous pouvons créer des particules brillantes qui montent au ciel pour qu'elles soient visibles de loin.

Pour créer un système de particules, sélectionnez le menu GAMEOBJECT/PARTICLE SYSTEM. Un système de particules apparaît alors sur votre scène. Vous pouvez le déplacer et le positionner où vous voulez. Si vous regardez l'inspector, vous y verrez toutes sortes d'options vous permettant de personnaliser le système. En voici quelques-unes :

- DURATION vous permet de définir la durée de vie des particules.
- LOOPING permet de spécifier si le système doit tourner en boucle ou non.

- Les sections COLOR OVER LIFETIME et SIZE OVER LIFETIME vous permettent de définir une couleur et une taille en fonction du temps. Vous pouvez dire qu'au début la particule est rouge et grande et à la fin verte et petite. La transition se fera automatiquement et progressivement.

Vous pouvez modifier beaucoup de choses comme la taille, la rotation, le comportement des particules ainsi que leur appliquer des textures.

Voilà par exemple ce que vous pouvez obtenir après quelques retouches :

Figure 7.6 : Particules

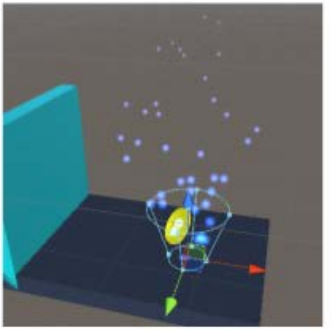
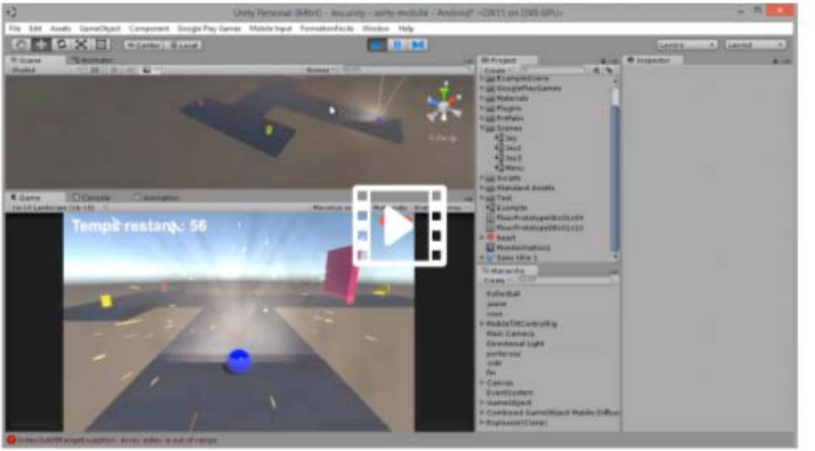


Figure 7.7 : Aperçu vidéo des particules



Nous verrons au chapitre suivant comment peuvent être utilisées ces particules dans vos jeux.

Dans ce chapitre, vous avez vu comment :

- créer des animations par script ;
- créer des animations avec l'outil d'animation ;
- appliquer une animation à un objet ;
- créer un système de particules.

8

Gérer les collisions

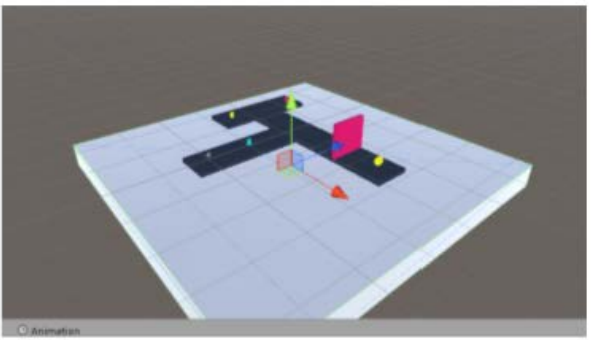
Les collisions sont indispensables dans tous les jeux. En effet, dans notre cas, si la balle tombe dans le vide, elle tombera à l'infini. De même, si la balle touche un pot de peinture ou la pièce de fin rien ne se passe. Cela est dû au fait que nous n'avons pas mis en place de système de collisions.

Les collisions vont nous permettre de savoir si on a touché un pot et ainsi de le ramasser. Si on tombe dans le vide, nous allons détecter la chute et relancer le niveau. Voilà à quoi peut servir un système de collisions.

8.1. Détecter la chute dans le vide

Pour savoir si le joueur tombe dans le vide, nous allons utiliser les collisions et une petite ruse. Vous allez créer un très grand cube 3D que vous allez placer en dessous du décor comme sur la Figure 8.1.

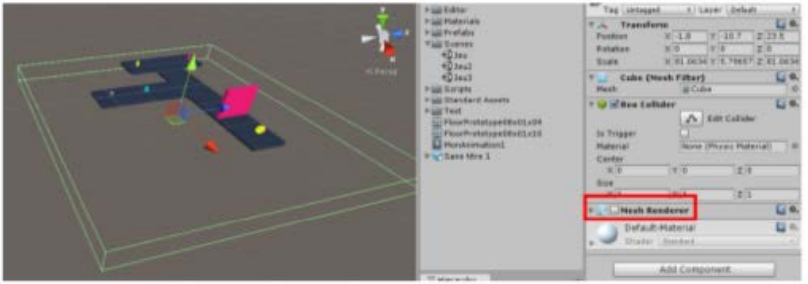
Figure 8.1 : Cube de collision



Nous utiliserons ce cube pour détecter si la balle est tombée dans le vide. En effet, nous le saurons dès lors qu'elle sera entrée en collision avec lui. Nous pourrons alors relancer le niveau depuis le début. Pour que le niveau reste propre, notre cube doit être invisible. Pour cela, nous décocherons l'option MESH RENDERER du cube.

Note › Le composant Mesh Renderer permet de rendre un objet visible. C'est ce composant qui fait qu'un objet apparaîtra à l'écran. S'il est désactivé, l'objet sera là mais il sera invisible.

Figure 8.2 : Désactivation du renderer



Ainsi le test de collision sera invisible pour le joueur, il aura simplement l'impression de perdre et de recommencer le niveau. Comme pour les pots de peinture vous devez donner un nom au cube, par exemple nous l'appellerons "vide".

Maintenant nous allons voir comment détecter les collisions avec un script. Créez un nouveau script que vous appellerez `CheckCollisions`. Pour détecter les collisions, il existe une fonction standard : `OnCollisionEnter`. Cette fonction s'utilise de la façon suivante :

```
void OnCollisionEnter(Collision obj)
{
  if(obj.gameObject.name = "porte")
  {
    // ...
  }
}
```

Note › La fonction `OnCollisionEnter` ressemble beaucoup à `OnTriggerEnter`. Leur utilisation est la même et la seule différence réside dans le fait que `OnCollisionEnter` se déclenche lorsqu'il y a une collision avec un objet solide et `OnTriggerEnter` lorsque l'on "rentre" dans un objet non solide (avec l'option IS TRIGGER activée).

`obj` correspond à l'objet touché. Le test `if` nous permet de savoir si le nom de l'objet est égal au mot spécifié entre guillemets. Nous allons utiliser la fonction `LoadLevel` si on touche `"vide"`. Voilà comment procéder :

```
void OnCollisionEnter(Collision obj)
{
  if(obj.gameObject.name = "vide")
  {
    Application.LoadLevel(Application.loadedLevelName);
```

```
    }
}
```

Si vous placez le script sur la balle et que vous tombez dans le vide, le niveau se relancera.

Attention : Pour que le changement de niveau fonctionne, il faut ajouter toutes les scènes de votre jeu dans le BUILD SETTINGS (FILE/BUILD SETTINGS).

Note : Pour plus d'informations sur les conditions et leur emploi dans un script de collision, se reporter aux chapitres Premiers scripts C# et Interagir avec l'environnement du module I. Votre premier jeu PC.

8.2. Ramassage des pots de peinture

Nous allons maintenant passer aux pots de peinture. Les pots sont déjà configurés et il ne nous reste plus qu'à ajouter une nouvelle condition dans notre fonction précédente. Nous allons créer trois conditions pour le pot rouge, le bleu et le jaune.

```
void OnCollisionEnter(Collision obj)
{
  if(obj.gameObject.name == "vide")
  {
    Application.LoadLevel(Application.loadedLevelName);
  }
  else if(obj.gameObject.name == "rouge")
  {

  }
  else if(obj.gameObject.name == "bleu")
  {

  }
  else if(obj.gameObject.name == "jaune")
  {

  }
}
```

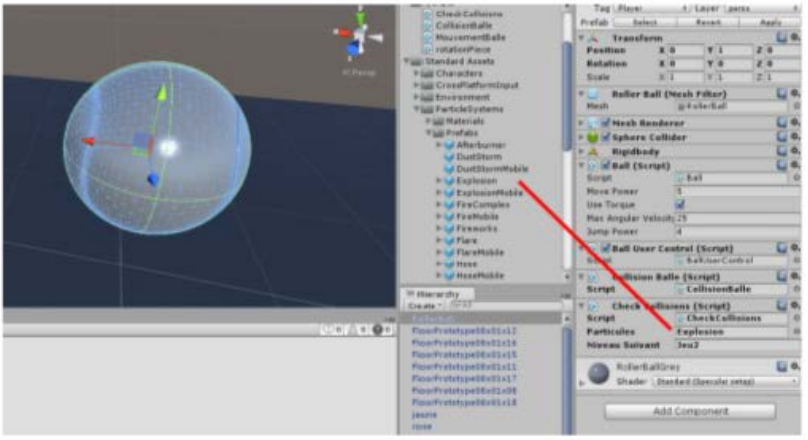
C'est de cette façon que vous allez pouvoir savoir quel type de pot a été touché. Il faut maintenant faire disparaître le pot, faire apparaître des particules pour montrer qu'il a été ramassé et colorer la balle de la couleur du pot.

La fonction `Destroy` permet de détruire un objet, `Instantiate` permet de faire apparaître un objet (dans notre cas les particules) et `Renderer.material` permet de charger la couleur de la balle. Je vais vous montrer un exemple complet pour le pot rouge. Nous allons avoir besoin de créer deux variables en haut de notre script. Une variable de type GameObject qui nous permettra de stocker le prefab des particules et un `string` nous permettant de stocker la couleur de la balle :

```
public GameObject particules;
public static string couleur = "blanche";

void OnCollisionEnter(Collision obj)
{
  // ...
  if(obj.gameObject.name == "rouge")
  {
    //Faire apparaître les particules
    Instantiate(particules, obj.transform.position,
 Quaternion.identity);
    //Colorer la balle et stocker la couleur
    this.gameObject.GetComponent<Renderer>().material.color = Color.red;
    couleur = obj.gameObject.name;
    //Détruire le pot
    Destroy(obj.gameObject);
  }
  // ...
}
```

Ce code doit être repris pour le pot bleu et le pot jaune. Il faudra juste changer la couleur de la balle en `blue` et `yellow`. Pensez aussi à faire glisser un prefab de particules dans la variable `particules` qui est visible dans l'inspector. Vous retrouverez des particules dans les packages de base de Unity. Vous pouvez prendre une explosion, des étoiles, des couleurs ou ce que vous voulez.

Figure 8.3 : Ajout des particules



Testez le jeu et essayez de ramasser un pot de peinture pour visualiser l'effet.

Figure 8.4 : Test du script

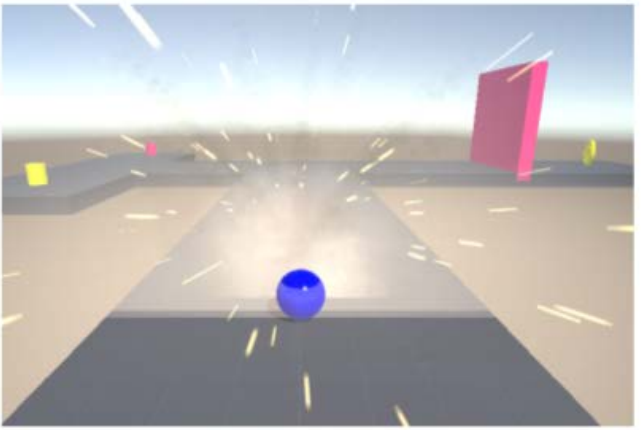
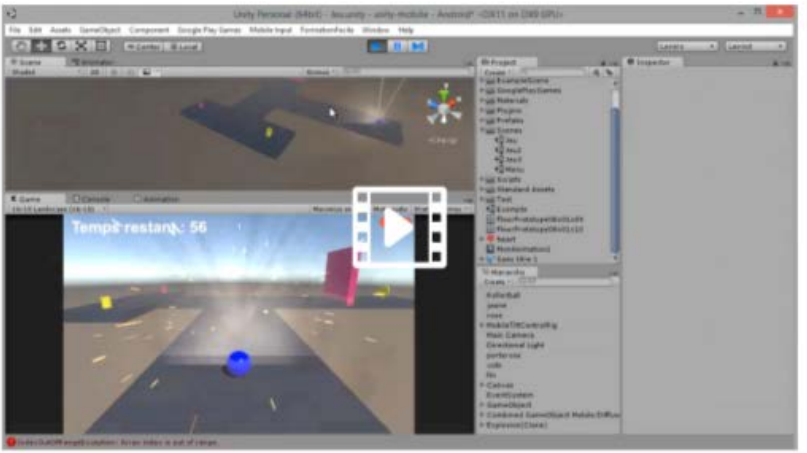


Figure 8.5 : Aperçu vidéo du ramassage des pots



Avant de terminer ce chapitre, nous allons ajouter une dernière condition à notre script pour savoir si on touche l'objet ayant pour nom `fin`. Si le joueur touche la fin, on charge le niveau suivant. Voilà le code complet du script de collisions :

```
using UnityEngine;
```

```
using System.Collections;

public class CheckCollisions : MonoBehaviour {

  public GameObject particules;
  public static string couleur = "blanche";
  public string niveauSuivant;

  void OnCollisionEnter(Collision obj)
  {
    if(obj.gameObject.name == "vide")
    {
      Application.LoadLevel(Application.loadedLevelName);
    }
    else if(obj.gameObject.name == "rouge")
    {
      //Faire apparaître les particules
      Instantiate(particules, obj.transform.position,
 Quaternion.identity);
      //Colorer la balle et stocker la couleur
      this.gameObject.GetComponent<Renderer>().material.color =
 Color.red;
      couleur = obj.gameObject.name;
      //Détruire le pot
      Destroy(obj.gameObject);
    }
    else if(obj.gameObject.name == "bleu")
    {
      //Faire apparaître les particules
      Instantiate(particules, obj.transform.position,
 Quaternion.identity);
      //Colorer la balle et stocker la couleur
      this.gameObject.GetComponent<Renderer>().material.color =
 Color.blue;
      couleur = obj.gameObject.name;
      //Détruire le pot
      Destroy(obj.gameObject);
    }
    else if(obj.gameObject.name == "jaune")
    {
      //Faire apparaître les particules
      Instantiate(particules, obj.transform.position,
 Quaternion.identity);
      //Colorer la balle et stocker la couleur
      this.gameObject.GetComponent<Renderer>().material.color =
 Color.yellow;
      couleur = obj.gameObject.name;
      //Détruire le pot
      Destroy(obj.gameObject);
    }
    else if(obj.gameObject.name == "fin")
    {
      Application.LoadLevel(niveauSuivant);
    }
  }
```

```
}
```

Ce script gère la collision avec le cube, les trois types de pots et l'objet de fin. Lorsque la balle change de couleur nous pouvons connaître sa couleur actuelle avec la variable `couleur`. Nous utiliserons cette variable dans le chapitre suivant dans lequel je vous montrerai comment savoir si l'objectif fixé est réalisé ou pas. Par exemple est-ce que la balle est rouge et est-ce que le joueur a trois diamants ? Nous utiliserons également les collisions pour cela. Si tout fonctionne, je vous invite à passer au prochain chapitre.

Dans ce chapitre, vous avez appris à :

- gérer les collisions ;
- ramasser un objet ;
- instancier des particules ;
- charger un niveau ;
- changer la couleur de la balle.

9

Fixer des objectifs

Les objectifs vont nous permettre de mettre en place une ou plusieurs conditions de victoire. Le joueur ne pourra gagner que s'il a rempli tous les objectifs qui ont été fixés. Par exemple, dans notre cas, l'objectif est de toucher la pièce de fin. Pour cela, il faut que la balle soit de la couleur de la porte finale. Vous pouvez également ajouter d'autres objectifs comme par exemple ramasser trois cristaux ou obtenir un score de 500. Afin de rester simple, nous allons juste mettre en place le système de couleurs et un timer pour forcer le joueur à atteindre l'objectif en moins d'une minute.

9.1. Sommes-nous de la bonne couleur ?

Au chapitre précédent, nous avons créé un script permettant de gérer les collisions. Nous sommes en mesure de savoir quel est le dernier pot de peinture qui a été touché et ainsi de savoir si la couleur de la balle permet d'ouvrir une porte ou non. Nous allons commencer par configurer la porte du premier niveau. Cette porte est rose et s'appelle `porterose`. Vous savez maintenant comment identifier l'objet touché :

```
void OnCollisionEnter(Collision obj)
{
  if(obj.gameObject.name = "porterose")
  {
    // ...
  }
}
```

Ces quelques lignes nous permettent de savoir si la porte a été touchée. Nous savons aussi de quelle couleur est la balle grâce à la variable `couleur` qui se trouve dans le script `CheckCollisions`. Si la balle est rose et que la porte est aussi rose, nous pouvons l'ouvrir ; sinon le joueur ne peut pas continuer dans sa progression.

```
void OnCollisionEnter(Collision obj)
{
  if(obj.gameObject.name == "porterose")
  {
    if(couleur == "rose")
    {
```

```
        Destroy(obj.gameObject);
      }
    }
}
```

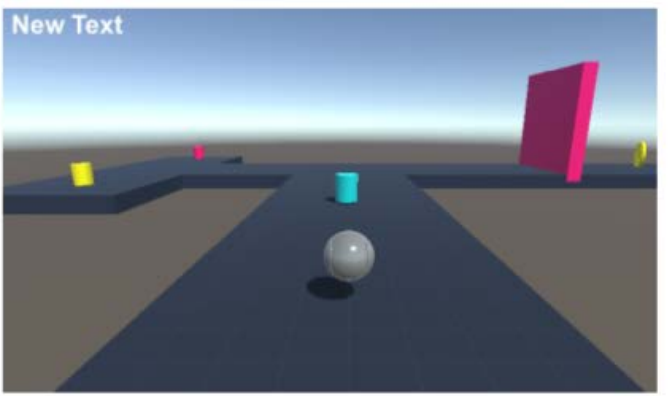
Ce script vous permettra de n'ouvrir la porte que si le joueur est de la bonne couleur. En d'autres mots, si l'objectif est atteint, la porte s'ouvre, sinon rien ne se passe. Vous pouvez de nouveau tester le jeu pour vérifier le bon fonctionnement du système.

9.2. Ajout d'un timer

Le compte à rebours va mettre un peu de pression sur les épaules du joueur. Comme il doit terminer le niveau avant la fin du timer, il va se précipiter et fera plus d'erreurs. Cela ajoute de la difficulté et vous offre l'opportunité de mettre en place un système de scores (quel est le joueur le plus rapide ?).

Pour commencer, nous allons préparer la zone de texte où sera affiché sur l'écran le temps restant. Nous ajoutons à la scène un objet Texte à l'aide du menu GAMEOBJECT/UI/TEXT. Dans l'inspector, vous pouvez accéder à ses propriétés et ainsi modifier sa taille, sa police, sa couleur ou sa position. Voilà à quoi ressemble mon texte après configuration :

Figure 9.1 : Création du texte



Et voilà la configuration que j'ai appliquée dans l'inspector :

Figure 9.2 : Configuration du texte

***Note** › Ici nous utilisons le nouveau système d'UI qui nous permet de créer beaucoup d'éléments d'interface comme des textes ou des boutons et de faire en sorte que ces éléments puissent s'adapter automatiquement à tous les écrans. Le système d'UI est la meilleure façon de créer des éléments pour l'interface utilisateur (barre de vie, score, menu...).*

Nous allons maintenant mettre en place le timer à l'aide d'un script. Créez un nouveau script appelé `Timer` et placez-le sur l'objet texte. Dans le script, créez une variable `myTimer` qui sera un `float` et qui correspondra au temps restant. Vous devez aussi créer une autre variable `finTimer` qui cette fois sera un `bool` et permettra de savoir si le timer est fini ou non. Pour décompter le temps écoulé, vous utiliserez l'expression `myTimer -= Time.deltaTime;`. Il s'agit d'une forme abrégée de `myTimer = myTimer - Time.deltaTime;` `Time.deltaTime` correspond au temps écoulé depuis le début du niveau (pour plus d'informations sur les variables temps, se reporter à la documentation officielle). Ce qui nous donne le script suivant :

```
using UnityEngine;
using System.Collections;

public class Timer : MonoBehaviour {
```

```
    public float myTimer = 60.0f;
    public static bool finTimer = false;

    void Update()
    {
      if(myTimer > 0){
        myTimer -= Time.deltaTime;
      }
      if(myTimer <= 0){
        finTimer = true;
      }
    }
}
```

Ce code est fonctionnel et modifie la valeur de `finTimer` quand le décompte est fini. Le joueur a alors perdu et ne peut plus atteindre la fin. Si vous revenez dans le script de collisions, vous pourrez accéder à la valeur de la variable `finTimer` à l'aide de l'expression `Timer.finTimer`. Les variables `public static` sont accessibles de partout si vous les appelez en faisant précéder leur nom par le nom du script où elles sont déclarées. Nous allons donc modifier le script de collisions au niveau de la condition permettant de détecter si le joueur touche la fin. Désormais le changement de niveau ne se fera que si le timer n'est pas terminé :

```
//...
else if(obj.gameObject.name == "fin")
{
  if(Timer.finTimer == false)
  {
    Application.LoadLevel(niveauSuivant);
  }
}
//...
```

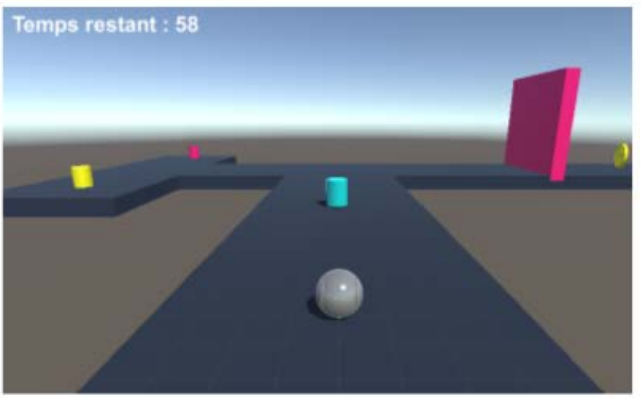
La dernière étape consiste à afficher le temps restant à l'écran pour que le joueur puisse voir le décompte. Pour pouvoir modifier le texte selon la valeur de la variable `myTimer`, nous devons ajouter une nouvelle directive au début du script `Timer` : `using UnityEngine.UI`. C'est ce qui nous permet d'utiliser l'expression `this.gameObject.GetComponent<Text>().text = "MonTexte";` (ce qui signifie "le texte du composant Text de cet objet") pour modifier le texte. Nous allons donc modifier le script `Timer` de la façon suivante :

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class Timer : MonoBehaviour {
```

```
    public float myTimer = 60.0f;
    public static bool finTimer = false;

    void Update()
    {
      if(myTimer > 0){
        myTimer -= Time.deltaTime;
        this.gameObject.GetComponent<Text>().text = "Temps restant :
  "+(int)myTimer;
      }
      if(myTimer <= 0){
        finTimer = true;
        this.gameObject.GetComponent<Text>().text = "Terminé";
      }
    }
  }
```

Et voilà le résultat dans votre jeu :

Figure 9.3 : Affichage du timer



Vous savez maintenant comment mettre en place des conditions de victoire et de défaite. Vous pouvez créer un système de scores avec le même principe.

Dans ce chapitre, vous avez appris à :

- détecter si un objectif est atteint ;
- créer un timer ;
- afficher du texte à l'écran.

10

Optimiser les scènes

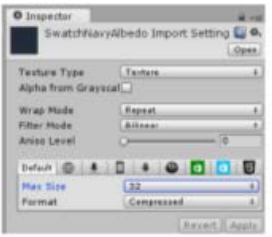
L'optimisation est une étape très importante qui vise à faire en sorte que votre jeu tourne sans ralentissements sur le plus grand nombre d'appareils possibles. Dans ce chapitre, je vais vous montrer quelques astuces d'optimisation qui sont essentielles pour améliorer les performances de votre jeu.

10.1. Optimisation des ressources

La première étape d'optimisation consiste à optimiser les ressources que vous utilisez. Cela implique de n'utiliser que des modèles 3D *low poly*, des textures compressées (PNG ou JPG) de petite taille (maximum 512×512 px) et d'utiliser un site comme http://tinypng.com pour réduire la taille de vos images. Pour les sons et musiques, je vous invite à utiliser le format OGG ou MP3 afin de réduire leur taille et d'améliorer la vitesse de chargement de votre jeu.

Au bas mot, ces techniques vont vous permettre de diminuer par deux le poids total de votre jeu et ainsi de diminuer par deux les temps de chargement. Si vous voulez optimiser encore un peu plus vos images, vous pouvez passer par l'inspector de Unity et choisir un bon format de compression pour les textures :

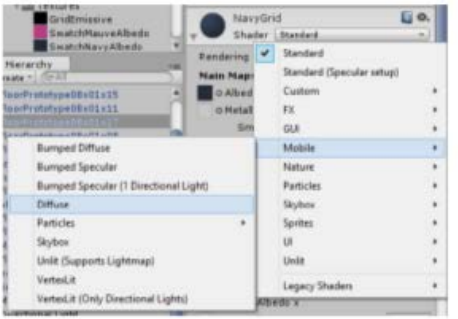
Figure 10.1 : *Compression des textures*



Les shaders contribuent également à alourdir vos scènes. En effet, les shaders standards sont beaucoup plus complexes à calculer que les shaders optimisés (par exemple les

shaders pour mobiles). Les shaders standards prennent en compte la brillance de l'objet, la luminosité, la réflexion de la lumière, l'aspect de la matière, etc. Les shaders optimisés sont très simples et ne demandent que très peu de calculs, c'est pourquoi je vous invite à les privilégier dès lors que vous développez pour des smartphones. Pour changer de shader, cliquez sur le menu déroulant du SHADER de l'objet et sélectionnez-en un se trouvant dans la rubrique MOBILE.

Figure 10.2 : Les shaders mobiles

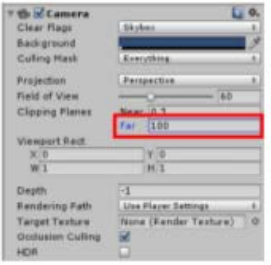


10.2. Optimisation de la distance de vision

Plus la caméra a une distance de vue importante, plus il y aura d'objets à calculer. Pour optimiser vos scènes, vous pouvez diminuer la distance de vision de la caméra afin de ne pas afficher les objets se trouvant loin du joueur et ainsi économiser du temps de calcul.

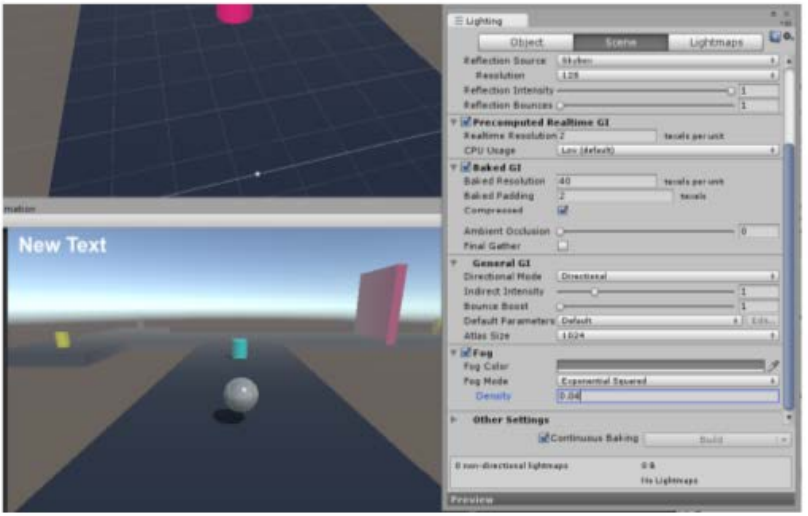
Pour diminuer la distance de vision, cliquez sur la caméra et diminuez la valeur de FAR dans l'inspector :

Figure 10.3 : Distance de vision



Dans notre cas la différence ne sera pas perceptible, mais si vos niveaux sont vastes et qu'il y a beaucoup d'éléments de décor, il est indispensable de diminuer la valeur de cette variable. Notez toutefois que plus cette valeur est basse, moins les niveaux sont beaux visuellement. Vous pouvez y remédier jusqu'à un certain point en en masquant l'arrière-plan avec du brouillard. Pour activer le brouillard, cliquez sur le menu WINDOW/LIGHTING et activez le FOG en bas de la fenêtre.

Figure 10.4 : Activation du brouillard



10.3. Fusion des objets

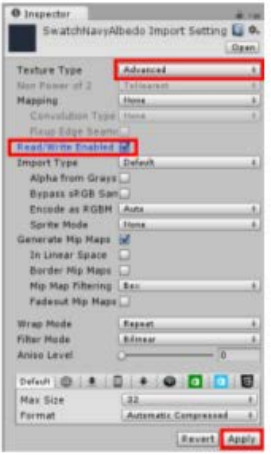
Il est plus facile de calculer la position d'un gros modèle 3D que celle de plusieurs petits modèles 3D. C'est pourquoi, il peut être intéressant en termes de performance, de fusionner plusieurs modèles 3D pour n'en obtenir qu'un seul. C'est ce que vous allez apprendre à faire dans cette section.

Nous allons ici fusionner les cases qui représentent le sol de notre niveau. Attention, il ne faudra pas inclure les parties du sol qui sont animées sinon cela ne fonctionnera pas ! Commencez par créer un GameObject vide dans lequel vous allez faire glisser toutes les parties du sol.

Figure 10.5 : Préparation pour la fusion

Pour pouvoir opérer la fusion également sur les textures et n'en obtenir qu'une seule, vous devez autoriser l'écriture sur la texture du sol. Pour cela, cliquez dessus et dans l'inspector sélectionnez le mode ADVANCED puis cochez READ/WRITE ENABLED comme sur la Figure 10.6.

Figure 10.6 : Activation de l'écriture



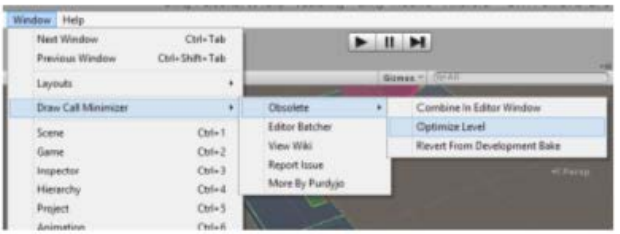
Nous allons maintenant télécharger un petit outil sur l'Asset Store qui va nous permettre d'opérer la fusion. Ouvrez l'Asset Store et recherchez le plug-in gratuit DRAW CALL MINIMIZER :

Figure 10.7 : Téléchargement du plug-in

Téléchargez et importez-le dans Unity. Nous sommes maintenant prêts à fusionner les objets !

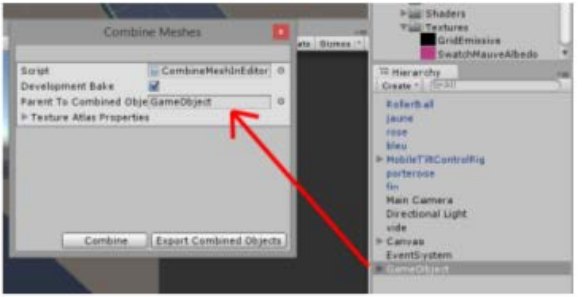
Un nouveau menu est apparu dans l'éditeur. Vous trouverez l'outil de fusion dans WINDOW/DRAW CALL MINIMIZER/OBSOLETE/OPTIMIZE LEVEL.

Figure 10.8 : Utilisation du plug-in



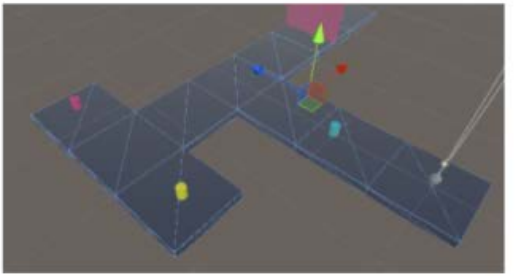
Une nouvelle fenêtre s'ouvre et vous demande de spécifier le GameObject à fusionner. Faites glisser l'objet qui contient toutes les cases du sol que vous souhaitez regrouper.

Figure 10.9 : Affectation du GameObject à fusionner



Puis cliquez sur COMBINE pour valider la fusion, et c'est terminé. Votre sol est maintenant un seul grand objet. Cela va diminuer énormément le nombre d'éléments à calculer. Voilà ce que vous obtenez :

Figure 10.10 : Objet fusionné



Vous devez fusionner le maximum d'objets (sol, murs, décor...). Faites attention à ne jamais fusionner les objets animés ou les objets avec lesquels le joueur peut interagir (pièces, peinture, cristaux...). Toutes ces étapes vont rendre vos jeux beaucoup plus fluides et rapides même sur les vieux appareils mobiles.

Dans ce chapitre, vous avez vu comment optimiser votre jeu via plusieurs techniques comme :

- l'optimisation des ressources ;
- la diminution de la distance de vue ;
- la fusion des textures et des objets.

11

L'interface utilisateur

Quel que soit le jeu, vous aurez besoin de créer un menu principal et d'éventuels menus pause. Par ailleurs, vous pouvez souhaitez afficher un score, les vies du joueur ou d'autres choses. L'interface est constituée de boutons, d'images et de textes qui sont au-dessus du jeu, toujours visibles à l'écran. Pour créer ces éléments, nous nous aidons du système d'UI (*User Interface*) de Unity, accessible depuis le menu GAMEOBJECT/UI.

À la différence des autres objets, les éléments proposés par le menu UI apparaissent toujours au premier plan et s'adaptent automatiquement à la taille des écrans. Il est essentiel de les utiliser pour que vos interfaces, et notamment vos menus, fonctionnent sur tous les appareils avec un rendu optimal. Avant l'arrivée de ce système, il était beaucoup plus fastidieux d'obtenir un résultat correct. Si, à la place d'un Text UI, vous utilisez un 3DText placé devant la caméra pour afficher par exemple un score, le texte pourra dans certaines circonstances disparaître. Supposons que le joueur se colle à un mur, alors il disparaîtra derrière le mur avec lui. Les éléments UI, pour leur part, ne sont pas placés dans le monde 3D, ils sont au premier plan, devant ce que filme la caméra.

Dans ce chapitre, nous allons créer dans un premier temps le menu principal de notre jeu d'exemple. La procédure est très similaire à celle que nous avons pu voir dans la *I. Votre premier jeu PC*, au chapitre pareillement appelé *L'interface utilisateur*. Toutefois nous ne pouvions faire l'impasse sur cette étape indispensable. Puis, dans un second temps, nous créerons une barre de vie très simple pour vous montrer un exemple plus avancé d'utilisation des outils UI proposés par Unity.

11.1. Création d'un menu principal

Commençons par créer le menu principal de notre jeu. Ce menu sera la première chose que le joueur verra à l'écran au lancement de votre jeu, il est donc important de le soigner. Créez une nouvelle scène que vous appellerez `Menu`, ouvrez-la et et ajoutez-la à la liste des scènes du projet dans les BUILD SETTINGS en première position. Nous allons créer un menu très simple avec un titre, un bouton Jouer, un bouton Quitter, un bouton renvoyant vers un site internet. Si vous le souhaitez vous pourrez ajouter un bouton pour les options. Nous avons déjà vu comment créer du texte à l'écran, je vais donc vous

laisser créer le titre du menu. Pour ajouter les boutons, sélectionnez autant de fois que nécessaire l'entrée du menu GAMEOBJECT/UI/BUTTON puis configurez-les.

Figure 11.1 : Création du menu

Vous pouvez ajouter à la scène une image d'arrière-plan et des logos pour décorer un peu votre menu. Pour ma part, je vais conserver un menu simple.

Astuce : Si vous voulez un menu élaboré, aidez-vous pour commencer du menu gratuit mis à disposition par les développeurs de Unity. Vous le trouverez sur l'Asset Store sous l'appellation "Unity Samples: UI".

Ajoutez les éléments de votre choix à votre menu puis passez à la partie programmation des événements.

11.2. Création des scripts du menu

Lorsque l'utilisateur clique sur un bouton, rien ne se passe. Vous devez d'abord créer un script qui va gérer les interactions de l'utilisateur avec le menu. Appelez-le par exemple `ScriptMenu` et ajoutez-le à n'importe quel objet de la scène. Nous allons créer trois fonctions pour notre menu.

La fonction de chargement du premier niveau

```
public void chargerNiveau(string niveau)
{
Application.LoadLevel(niveau);
}
```

Cette fonction est publique et prend en paramètre le nom du niveau à charger.

La fonction qui permet d'accéder à votre site internet

```
public void ouvrirSite()
{
Application.OpenURL("http://anthony-cardinale.fr");
}
```

La fonction qui permet de quitter le jeu

```
public void quitterJeu()
{
Application.Quit();
}
```

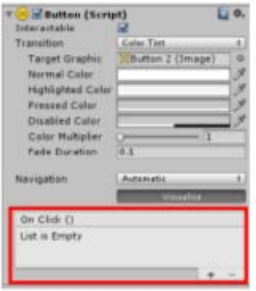
Voilà le script complet :

```
using UnityEngine;
using System.Collections;

public class ScriptMenu : MonoBehaviour {

  public void chargerNiveau(string niveau)
  {
    Application.LoadLevel(niveau);
  }

  public void ouvrirSite()
  {
    Application.OpenURL("http://anthony-cardinale.fr");
  }

  public void quitterJeu()
  {
    Application.Quit();
  }
}
```

11.3. Association des scripts aux boutons

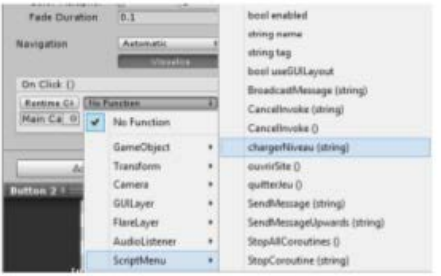
Vous allez maintenant associer le script à chaque bouton pour qu'il puisse être appelé lorsque l'utilisateur cliquera dessus. Sélectionnez un bouton, et accédez à son gestionnaire de clics dans l'inspector.

Figure 11.2 : Gestionnaire de clics d'un bouton



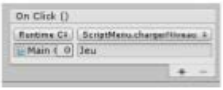
Cliquez sur le signe + pour ajouter une action. Faites glisser l'objet qui contient le script `ScriptMenu` dans l'emplacement et sélectionnez dans le menu déroulant la fonction créée dans le script que vous souhaitez appeler pour le bouton sélectionné. Pour le premier bouton, ce sera `chargerNiveau()`.

Figure 11.3 : Ajout du script



Enfin, spécifiez le paramètre de la fonction (le nom du niveau à charger) dans la case vide :

Figure 11.4 : Configuration du script



C'est terminé et fonctionnel. Vous pouvez tester la scène et cliquer sur le bouton. Lors du clic, le niveau se lance. Répétez l'opération pour les deux autres boutons : cliquez sur le signe + de leur gestionnaire de clics pour ajouter une action, spécifiez le GameObject et la fonction à lancer. Ces fonctions ne prennent pas de paramètres, vous n'avez donc pas besoin de spécifier de valeurs.

Les trois boutons sont maintenant fonctionnels. Vous pouvez utiliser ces techniques pour créer des boutons dans chaque scène de votre jeu, notamment un lien de retour vers le menu principal.

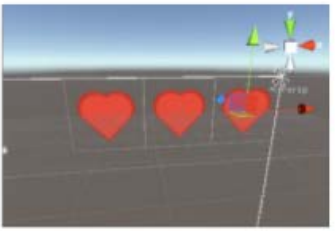
11.4. Ajout d'une barre de vie

Pour aller un peu plus loin dans l'utilisation du système d'UI, nous allons mettre en place une barre de vie très simple qui sera constituée de trois images de cœurs. Le joueur aura trois vies et quand il perdra une vie, un des cœurs va disparaître. Si le joueur perd toutes ses vies, cela pourra déclencher par exemple le chargement d'une scène de `Game Over` par exemple.

Pour garder en mémoire les vies du joueur, je vais recourir à une technique de sauvegarde que nous examinerons plus en détails au chapitre Sauvegarder des informations : les `playerPrefs`. Celles-ci permettent de stocker des variables en mémoire. Je ne m'attarderai donc pas sur leur utilisation, le chapitre suivant leur étant entièrement dédié.

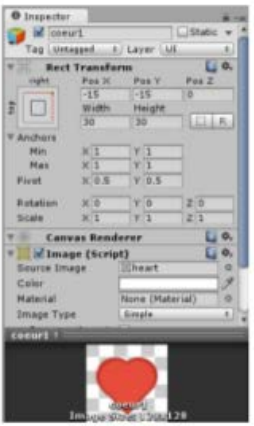
Commencez par créer trois images avec le système d'UI (menu GAMEOBJECT/UI/IMAGE). Donnez-leur une taille de 30×30 px et placez-les en haut à droite de l'écran. Allez ensuite sur le site iconfinder.com pour télécharger une image de cœur et importez-la dans votre projet. Après avoir sélectionné l'image, modifiez son type afin de la définir comme SPRITE (2D AND UI) et de pouvoir associer l'image de cœur aux trois images vides que vous avez créées. SPRITE (2D AND UI) est le seul format compatible avec le système d'UI. Si vos images ne sont pas au bon format, vous ne pourrez pas les appliquer aux carrés blancs. Une fois vos trois images configurées, voici ce que vous obtenez :

Figure 11.5 : Préparation de la barre de vie



Au niveau de l'inspector, nous avons une configuration très simple :

Figure 11.6 : Configuration de l'image de coeur



Nous pouvons maintenant passer au script de la barre de vie. Nous allons utiliser le script `CheckCollisions` qui gère déjà la collision entre la balle et le cube invisible permettant de détecter la chute dans le vide. Nous allons nous servir de cela pour faire perdre des vies à notre balle.

Commencez par ajouter une nouvelle variable `vie` à votre script. Ensuite, lorsque le joueur entre en collision avec l'objet nommé `vide`, diminuez la valeur de `vie` à l'aide de l'expression `vie--;` qui signifie en fait `vie = vie -1;` mais de façon abrégée. Puis vous allez stocker en mémoire la valeur de la variable `vie` avec le système des `PlayerPrefs` tel que nous le verrons au chapitre Sauvegarder des informations. Voilà à quoi ressemble ma fonction de détection de collisions avec l'objet `vide` :

```
void OnCollisionEnter(Collision obj)
{
  if(obj.gameObject.name == "vide")
  {
    vie--;
    PlayerPrefs.SetInt("vie", vie);
    if(vie <= 0)
    {
      // Vous pouvez charger une scène game over par exemple
      Application.LoadLevel("gameover");
    }
    else{
      Application.LoadLevel(Application.loadedLevelName);
    }
  }
  // ...
}
```

N'oubliez pas de créer la variable `vie` de type `int` en début de script et associez-lui une valeur dans la fonction `Start` comme ceci :

```
void Start()
{
  // ...
  vie = PlayerPrefs.GetInt("vie");
  if(vie <= 0 || vie == null)
  {
    vie = 3;
  }
}
```

Il ne nous reste plus qu'à gérer l'affichage des images de cœurs. Pour que le script puisse accéder à l'UI, vous devez impérativement lui ajouter au début la directive suivante, ainsi que nous l'avons déjà vu à la Section 9.2, Ajout d'un timer.

```
using UnityEngine.UI;
```

Puis, toujours dans le script de collisions, nous créons un tableau de variables à l'aide de l'expression `public Image[] cœurs;`.

Maintenant nous allons utiliser la fonction `Update` pour gérer l'affichage des cœurs. Si le personnage a trois vies, on affiche trois cœurs, s'il en a deux, on en affiche deux, etc. Voilà un code très simple à comprendre qui permet de faire cela :

```
void Update()
{
  if(vie == 3)
```

```
        {
            cœurs[0].enabled = true;
            cœurs[1].enabled = true;
            cœurs[2].enabled = true;
        }
        if(vie == 2)
        {
            cœurs[2].enabled = false;
            cœurs[1].enabled = true;
            cœurs[0].enabled = true;
        }
        if(vie == 1)
        {
            cœurs[2].enabled = false;
            cœurs[1].enabled = false;
            cœurs[0].enabled = true;
        }
        if(vie == 0)
        {
            cœurs[0].enabled = false;
            cœurs[1].enabled = false;
            cœurs[2].enabled = false;
        }
    }
```

Et voilà, notre barre de vie est fonctionnelle ! Vous pouvez tester votre jeu et vous verrez que vous perdrez des cœurs quand vous tomberez dans le vide. Voilà à quoi ressemble ma vie après avoir perdu une fois :

Figure 11.7 : *Fonctionnement de la barre de vie*

Dans ce chapitre, vous avez vu comment

- créer un menu ;
- créer des boutons ;
- associer une fonction d'un script à un clic ;
- créer une barre de vie.

12

Sauvegarder des informations

Les sauvegardes sont présentes dans tous les jeux. Sauvegarder une information permet de garder en mémoire des informations importantes que le joueur souhaite conserver. Par exemple vous pouvez garder en mémoire le dernier niveau débloqué pour que le joueur puisse reprendre la partie là où il s'est arrêté. Vous pouvez également enregistrer son score pour conserver les meilleurs temps.

12.1. Garder en mémoire le dernier niveau débloqué

Unity propose un système très simple nous permettant de créer des sauvegardes, il s'agit des `PlayerPrefs`. Les `PlayerPrefs` sont des clés associées à une valeur stockée sur le disque de l'ordinateur ou du mobile. Par exemple, la clé `vie` peut être associée à la valeur `3`. Ainsi nous savons que le joueur possède trois vies.

Pour garder en mémoire une valeur, vous devez utiliser la fonction suivante : `PlayerPrefs.SetInt("Clé", valeurInt);`. Dans cet exemple, nous stockons un `int`. Pour stocker un `float` il faut utiliser `SetFloat` et pour les chaînes de caractères `SetString`. Ce sont les trois types de valeurs que vous pouvez stocker sur le disque. Dans notre cas, nous souhaitons sauvegarder le dernier niveau débloqué ; nous stockons donc le nom du niveau débloqué, qui est une chaîne de caractères. Nous écrirons alors `PlayerPrefs.SetString("dernierNiveau", niveauSuivant);`.

Le plus simple est d'utiliser la variable `string niveauSuivant` en haut du script de collisions et d'ajouter l'enregistrement en mémoire lorsque le joueur touche l'objet de fin du niveau. Il faut donc modifier la fonction suivante :

```
else if(obj.gameObject.name == "fin")
{
  if(Timer.finTimer == false)
  {
    Application.LoadLevel(niveauSuivant);
  }
}
```

et ajouter l'enregistrement comme ceci :

```
else if(obj.gameObject.name == "fin")
{
  if(Timer.finTimer == false)
  {
    PlayerPrefs.SetString("dernierNiveau", niveauSuivant);
    Application.LoadLevel(niveauSuivant);
  }
}
```

Ce script va vous permettre de proposer au joueur de reprendre sa progression exactement là où il s'était arrêté.

12.2. Charger une valeur stockée

Nous avons vu comment enregistrer une information, mais comment la récupérer ? Pour charger en mémoire une information vous devez utiliser la fonction `PlayerPrefs.GetInt("Clé");` ou `PlayerPrefs.GetFloat("Clé");` ou encore `PlayerPrefs.GetString("Clé");` selon le type de la valeur à récupérer. Par exemple, pour récupérer la valeur du dernier niveau enregistré, vous écrirez `PlayerPrefs.GetString("dernierNiveau");`. Cette information peut ensuite être stockée dans une variable. Voilà notre script de menu légèrement modifié afin de charger le dernier niveau débloqué :

```
private string dernierNiveau;

void Start()
{
  dernierNiveau = PlayerPrefs.GetString("dernierNiveau");
  if(dernierNiveau == "" || dernierNiveau == null)
  {
    dernierNiveau = "Jeu";
  }
}

public void chargerNiveau(string niveau)
{
  Application.LoadLevel(dernierNiveau);
}
```

Comme vous pouvez le constater, je crée une variable qui va prendre la valeur du dernier niveau débloqué. Si aucune valeur n'était en mémoire, cette variable deviendra `"Jeu"` qui correspond au premier niveau. Ensuite dans la fonction `chargerNiveau`, nous pouvons ouvrir la scène `dernierNiveau`. Pour contrôler, essayez de jouer et de débloquer un niveau, puis quittez et relancez le jeu pour voir si le niveau 2 se lance bien lorsque vous cliquez sur JOUER.

Cette technique peut être utilisée pour enregistrer la vie, l'expérience, l'équipement ou toute autre information du jeu. Vous pouvez même sauvegarder la position d'un objet en stockant trois `float` représentant les x, y et z du Transform. Si vous voulez sauvegarder le meilleur score, vous pouvez créer une fonction de cette forme :

```
public void SaveBestScore()
{
  if(score > PlayerPrefs.GetInt("bestScore"))
  {
    PlayerPrefs.SetInt("bestScore", score);
  }
}
```

Dans cet exemple, la sauvegarde ne se fait que si le nouveau score est supérieur à l'ancien score. Il existe d'autres techniques permettant de sauvegarder des données comme par exemple la sauvegarde dans des fichiers ou la sauvegarde en ligne sur un serveur web ou dans une base de données, mais ces techniques sont plus complexes à mettre en place.

Dans ce chapitre, vous avez vu comment :

- stocker des informations en mémoire ;
- charger des informations en mémoire ;
- utiliser les PlayerPrefs.

13

Publier son jeu

Une fois votre jeu terminé, sa publication sur une ou plusieurs plateformes le rendra accessible aux joueurs du monde entier. Dans ce chapitre, je vous montrerai comment publier un jeu sur Google Play pour Android et vous donnerai quelques astuces pour le publier sur l'iTunes Store d'Apple pour iOS. Je vous donnerai également des conseils pour mettre en avant votre jeu et obtenir plus de téléchargements.

13.1. Compiler son jeu

Avant de parler de publication, nous devons naturellement passer par l'étape de la compilation, comme nous l'avons vu au chapitre Compiler et tester. Rappelez-vous que celle-ci consiste à assembler tous les éléments qui composent votre jeu pour en faire une application mobile. Cette étape se passe au sein de Unity. Le logiciel s'appuie sur le SDK que nous avons téléchargé au chapitre Outils de développement mobile pour générer un fichier APK que nous pourrons ensuite publier sur Google Play.

Note › Pour les développeurs iOS, la compilation ne se fait pas entièrement dans Unity. En effet, ce dernier assemble les éléments pour générer un projet qu'il faudra ensuite ouvrir avec le logiciel Xcode pour pouvoir finaliser la compilation. Pour compiler un jeu iPhone, vous avez donc l'obligation de travailler sous Mac OS avec Xcode et vous devez avoir un compte développeur Apple. La procédure est plus complexe que pour le développement d'une application Android.

Avant de compiler le jeu, pensez à vérifier dans les paramètres du projet (menu FILE/BUILD SETTINGS) que vous avez bien intégré toutes les scènes du jeu, sélectionné la plateforme ciblée et configuré les paramètres de l'application dans la fenêtre PLAYER SETTINGS.

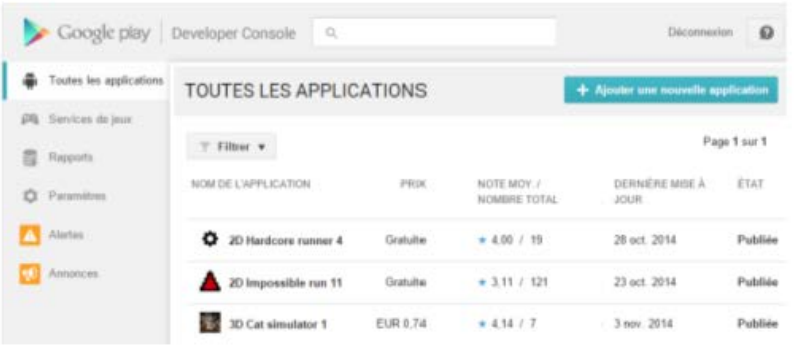
Dans la section PUBLISHING SETTINGS des paramètres de l'application, utilisez ou créez une keystore en utilisant un mot de passe. Il faudra toujours utiliser la même keystore pour vos futures mises à jour afin de vous identifier en tant que développeur de l'application.

Une fois que tout est prêt, vous pouvez cliquer sur le bouton BUILD pour créer le fichier APK. Attribuez-lui un nom et un emplacement. Une fois terminé, vous obtenez l'application que vous pourrez envoyer sur Google Play.

13.2. Publier son jeu

La publication d'un jeu se fait par l'intermédiaire des consoles développeurs Android ou iTunes (Apple). Pour vous y connecter, vous devez être enregistré en tant que développeur (voir le chapitre Outils de développement mobile pour plus d'informations sur les espaces développeurs Android et Apple).

Figure 13.1 : Console développeur Android



Note : L'inscription à l'espace développeur Android vous coûtera 25 € pour une licence valide à vie. Pour développer des applications iOS, vous devrez débourser 100 € par an. Chez Windows Phone, la licence coûte entre 25 € et 75 € par an. Google propose donc un tarif très compétitif.

Connectez-vous à votre compte développeur Google. Sur la page de la console, vous pouvez publier un nouveau jeu en cliquant sur le bouton AJOUTER UNE NOUVELLE APPLICATION (voir Figure 13.1). Donnez un nom à votre application et uploadez son fichier APK (voir Figure 13.2).

Figure 13.2 : Envoyer l'APK

Après avoir importé votre fichier APK, vous pouvez passer à la création de la fiche de votre produit.

Figure 13.3 : Fiche Google Play

Vous devez renseigner le titre et fournir à la fois une description courte et une description longue de votre jeu. Ces textes seront visibles sur sa page produit. Il faudra également envoyer plusieurs captures d'écran correspondant aux différents appareils ciblés.

Figure 13.4 : Captures d'écran

On vous demandera aussi de choisir une icône ainsi que des images publicitaires. Respectez bien la taille des images qui est imposée et le format. Vous pouvez spécifier une URL si vous avez créé une vidéo promotionnelle. Cette vidéo doit être publiée sur YouTube.

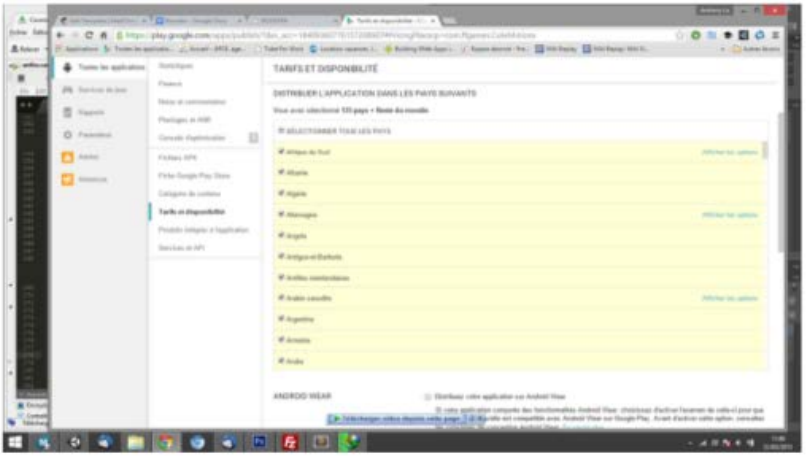
Choisissez une catégorie pour votre jeu afin de le classer dans Google Play (voir Figure 13.5).

Figure 13.5 : Classification

Vous pouvez indiquer votre site web ainsi que votre e-mail. Une fois terminé, vous devez passer au formulaire vous permettant de définir à quel public s'adresse votre jeu (tranche d'âge). Vous devrez répondre à quelques questions afin de préciser s'il y a de la violence ou des scènes choquantes afin de définir s'il est adapté aux enfants ou aux adultes. Cela ne vous prendre que trois minutes.

L'étape suivante permet de définir le prix du jeu ainsi que les pays pour lesquels vous souhaitez le publier.

Figure 13.6 : Prix et pays



Lorsque tout est configuré, le bouton de publication devient actif et vous pouvez soumettre votre jeu à Google qui le validera. Il sera disponible sur le store dès sa validation. Celle-ci peut prendre plusieurs jours avant d'être effective.

Note › *Pour pouvoir diffuser des publicités dans votre jeu, vous devez l'avoir mis en ligne une première fois afin d'obtenir son URL sur le store. Celle-ci est nécessaire pour l'obtension d'un code permettant l'affichage des publicités. Nous verrons cela au chapitre Monétiser son jeu. Beaucoup d'autres services nécessitent que votre jeu soit publié.*

Note › *La procédure de publication sur la console développeur iTunes est sensiblement la même, fourniture d'images, icônes, description, vidéo, etc. Sachez qu'Apple est beaucoup plus strict et plus long à valider l'application.*

13.3. Un peu de marketing

La concurrence est énorme dans le secteur des jeux mobiles. Nombreux sont les jeux qui ne décollent jamais et n'obtiennent que très peu de téléchargements. À vrai dire, 90 % des jeux ne marchent pas. Voici donc quelques conseils qui vous aideront à mettre en avant vos jeux et amélioreront leur visibilité. Vous augmenterez ainsi vos chances de décoller et d'obtenir de nombreux téléchargements.

Soignez votre description

Certains développeurs n'écrivent que quelques mots pour parler de leur jeu. Vous devez en faire une grande description d'au moins 1000 caractères et y inclure des mots clés. Si votre jeu est un jeu de ping-pong, vous devez placer cinq fois le mot "ping-pong" au sein de votre description longue. Vous devez aussi placer les mots "gratuit", "sport", "en ligne" afin de parler de la catégorie du jeu ou de ses fonctionnalités. Si un utilisateur recherche "jeu de ping pong gratuit en ligne", votre application sera proposée. Il existe des outils permettant de connaître les recherches des utilisateurs afin de choisir de bons mots clés. Je vous propose de regarder Google Trends et Google Adwords pour vous aider.

Créez des visuels propres et professionnels

Vos images doivent donner envie de jouer au jeu. Vous devez publier cinq images par appareil ciblé (cinq captures d'écran pour les mobiles, cinq captures pour les tablettes 7 pouces, cinq pour les tablettes 10 pouces, etc.). C'est sur ces images, ainsi que l'icône principale, que la première impression de votre jeu va se faire, ce sont elles qui vont donner envie aux joueurs de télécharger votre jeu.

Créez une vidéo de présentation

Cette vidéo ne doit pas dépasser les 60 secondes et vous devez utiliser une musique entraînante et montrer des images qui donnent envie. Inspirez-vous de ce que font les jeux à succès.

Traduisez votre jeu dans les cinq principales langues

Vous pouvez éventuellement utiliser Google Traduction pour limiter les frais mais pensez à faire des phrases simples afin d'obtenir une bonne traduction.

De plus, essayez de publier des articles sur des blogs spécialisés, des forums ou essayez de faire tester votre jeu par des sites ou une chaîne YouTube. Il faut faire parler de vous sur les réseaux sociaux afin d'attirer du trafic sur votre page.

Si vous avez un peu d'argent à investir, vous pouvez faire appel à une régie publicitaire comme Google AdMob ou AdBuddiz pour faire la promotion de votre jeu dans d'autres applications.

Lisez régulièrement des sites spécialisés ou des articles afin de rester à jour sur les tendances et les bonnes pratiques nécessaires à un travail propre et adapté. Ne vous découragez pas si vos premiers jeux ne fonctionnent pas aussi bien qu'espéré.

Dans ce chapitre, vous avez vu comment :

- publier votre jeu ;
- paramétrer la fiche produit ;
- améliorer votre visibilité sur les stores.

14

Monétiser son jeu

Nous voilà maintenant à l'étape de la monétisation de votre jeu afin de générer des revenus grâce à votre application. Il existe plusieurs façons de monétiser une application. Voici les trois principales :

- créer un jeu payant ;
- créer un jeu gratuit avec des achats intégrés ;
- créer un jeu gratuit et afficher des publicités.

Environ 95 % des jeux téléchargés sont gratuits. Les joueurs ne regardent plus les jeux payants, c'est pourquoi je vous invite à éviter de créer des jeux payants car vous risquez d'être déçu et d'avoir travaillé pour rien. En effet, les achats sont de moins en moins fréquents et si vous obtenez un total de 100 téléchargements pour votre jeu à 0,79 €, vous aurez gagné moins de 50 € après les différents prélèvements. Les jeux proposant des achats intégrés sont beaucoup plus populaires mais pour proposer des achats intégrés il faut que le jeu ait été pensé pour accueillir une boutique. En effet, vous devrez mettre en place un système cohérent donnant envie au joueur de payer pour avancer plus vite. Cette solution est selon moi davantage adaptée aux gros studios qui créent de gros jeux en ligne et addictifs dans lesquelles la progression peut être accélérée en payant. Les achats intégrés sont moins faciles à mettre en place pour les petites équipes ou pour les développeurs indépendants car leurs moyens sont plus faibles. La meilleure solution et la plus simple pour eux est de diffuser des publicités dans le jeu. En effet, les joueurs sont maintenant habitués à voir des publicités dans les jeux mobiles si ces publicités sont bien placées et ne gênent pas l'expérience utilisateur. Les publicités peuvent vous rapporter beaucoup plus d'argent que les deux solutions précédentes. Sur la base de mes propres réalisations, j'ai pu comparer les différentes solutions, et dans de nombreux cas la publicité s'avère beaucoup plus rentable que les deux autres. C'est cette solution que je vais vous présenter en détail dans ce chapitre.

14.1. Comprendre le fonctionnement d'une régie publicitaire

Pour afficher des publicités dans votre jeu, vous devez passer par une régie publicitaire qui vous donnera un kit de développement à intégrer dans l'application. Ce kit de développement va permettre de récupérer sur un serveur distant les différentes campagnes publicitaires qui sont en cours dans le pays où se trouve le joueur. La publicité est chargée sur son téléphone et affichée là où vous avez demandé son affichage. Le SDK va "mettre en pause" le jeu et instantanément afficher la publicité. Le joueur pourra alors la fermer ou cliquer dessus.

Il existe deux types de régies publicitaires :

- les régies qui payent au clic ou à l'affichage, c'est-à-dire à partir du moment où le joueur clique ou voit la publicité, mais les sommes perçues sont très faibles (quelques centimes tout au plus) ;
- les régies qui payent à l'installation. Dans ce second cas, vous percevrez une petite somme d'argent (entre 1$ et 5$) lorsqu'un joueur installera un jeu qui était présenté dans une publicité. Les sommes perçues sont moins nombreuses mais plus importantes qu'une régie qui fonctionne au clic.

Vous devez donc choisir une régie publicitaire en fonction de vos attentes. Les éléments à prendre en compte sont :

- le nombre d'annonceurs : plus il y a d'annonceurs, plus votre taux d'affichage sera important ;
- les revenus générés : certaines régies ont un meilleur rendement que d'autres et certaines régies prélèvent un plus grand pourcentage que d'autres ;
- la simplicité d'installation et de configuration : optez pour une régie ayant un SDK facile à intégrer et compatible avec Unity ;
- le SAV : certaines régies parlent français, discutent avec vous, vous aident à augmenter vos revenus et d'autres régies ne vous contacteront jamais.

Une fois que vous avez choisi votre régie publicitaire, vous pouvez commencer à monétiser vos jeux en intégrant leur solution. Je vais vous présenter la régie publicitaire que j'utilise pour mes propres jeux et qui selon moi est le meilleur choix selon les critères donnés ci-dessus. Il s'agit d'AdBuddiz.

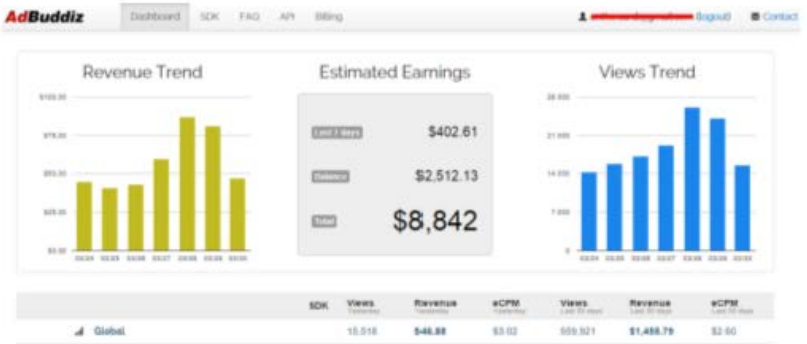
14.2. Présentation d'AdBuddiz

AdBuddiz est une régie publicitaire basée à Paris, l'équipe parle plusieurs langues dont le français et contacte régulièrement les développeurs pour les aider à optimiser leurs revenus. AdBuddiz paye à l'installation en moyenne entre 2$ et 4$, ce qui génère des revenus honorables, et diffuse des publicités dans le monde entier. Les publicités sont propres et variées. Le très gros avantage est aussi la simplicité d'utilisation de leur SDK autant pour Android que iOS. Deux lignes de code suffisent à afficher la publicité et c'est ce que nous verrons dans ce chapitre.

Afin de créer un compte développeur, rendez-vous sur le site officiel et inscrivez-vous. Cela vous permettra d'ajouter vos applications au tableau de bord et de récupérer le code permettant l'affichage des publicités dans votre jeu.

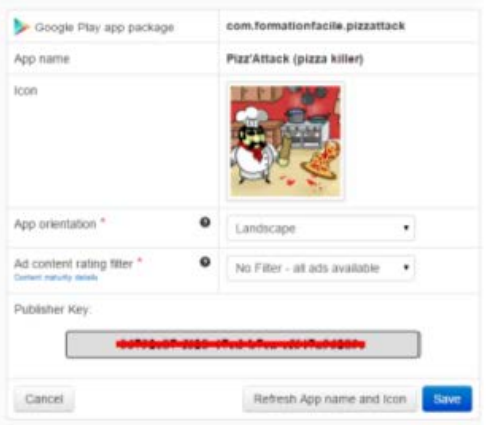
Le tableau de bord vous donne des informations sur les performances de vos applications. Vous pouvez visualiser les revenus générés ainsi que le détail pour chaque application :

Figure 14.1 : Tableau de bord d'AdBuddiz



En cliquant sur une application, vous aurez la possibilité d'accéder à ses paramètres pour récupérer votre code ou définir le type de publicités que vous souhaitez diffuser.

Figure 14.2 : Éditer une application



Enfin, dans le menu SDK, vous pouvez télécharger le kit de développement pour Unity et accéder à la documentation.

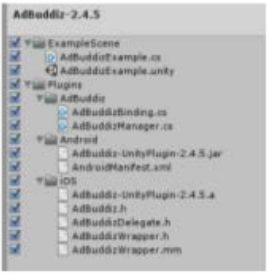
Figure 14.3 : SDK AdBuddiz

Je vous invite donc à ajouter votre application dans le tableau de bord et à télécharger le SDK pour Unity. Gardez de côté la clé de votre application car nous allons en avoir besoin pour l'intégration des publicités.

14.3. Intégration dans Unity

Retournons dans Unity afin d'importer et de configurer AdBuddiz. Commencez par cliquer sur ASSETS/IMPORT PACKAGE/CUSTOM PACKAGE afin d'aller chercher le SDK d'AdBuddiz que vous avez téléchargé. Le package contient quelques scripts qu'il faudra ajouter à votre projet :

Figure 14.4 : Importer le SDK AdBuddiz



Vous avez maintenant un dossier `Plugins` dans lequel se trouve le SDK ainsi qu'un fichier `AndroidManifest` qui sert à ajouter des permissions à votre application comme par exemple l'accès au réseau internet afin de télécharger les publicités sur les serveurs d'AdBuddiz.

Nous allons maintenant passer à la partie code. Ce que je vous propose c'est d'afficher une publicité lorsque le joueur termine un niveau. Afin de ne pas gêner le joueur, nous n'allons afficher les publicités que dans 30 % des cas. Ce chiffre nous donnera de bons résultats et il s'agit d'un bon compromis entre monétisation et confort utilisateur. Pensez que si l'utilisateur trouve qu'il y a trop de pubs dans votre application, il le fera savoir dans les commentaires et vous risquez de perdre des téléchargements.

Ouvrez le script `CheckCollisions`, dans lequel vous allez créer une fonction `start` qui nous servira à initialiser AdBuddiz et à charger la publicité :

```
void Start()
{
  AdBuddizBinding.SetAndroidPublisherKey("TEST_PUBLISHER_KEY_ANDROID");
  AdBuddizBinding.CacheAds();
}
```

Attention › Pensez à remplacer `TEST_PUBLISHER_KEY_ANDROID` par la clé qui a été générée pour votre application.

Nous allons maintenant afficher la publicité si le joueur touche l'objet de fin du niveau :

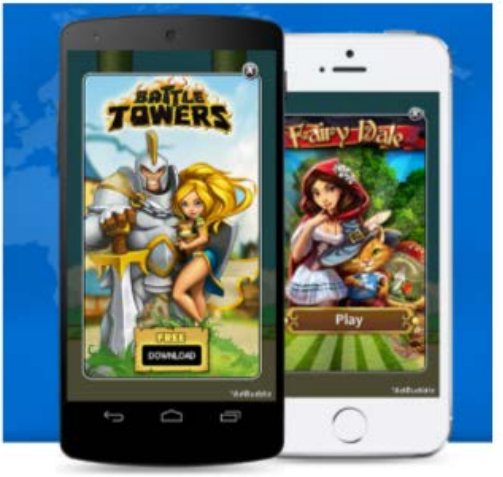
```
if(Random.Range(0,10) >= 7)
{
  AdBuddizBinding.ShowAd();
}
```

Le test complet pour notre jeu est le suivant :

```
// ...
else if(obj.gameObject.name == "fin")
{
  if(Random.Range(0,10) >= 7)
  {
    AdBuddizBinding.ShowAd();
  }
  if(Timer.finTimer == false)
  {
    PlayerPrefs.SetString("dernierNiveau", niveauSuivant);
    Application.LoadLevel(niveauSuivant);
  }
}
//...
```

Et c'est terminé ! Les publicités sont maintenant gérées par votre application. Voilà à quoi ressemblent les publicités :

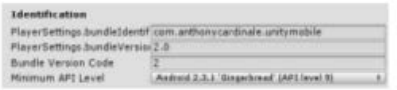
Figure 14.5 : Publicités AdBuddiz



14.4. Mettre à jour son jeu

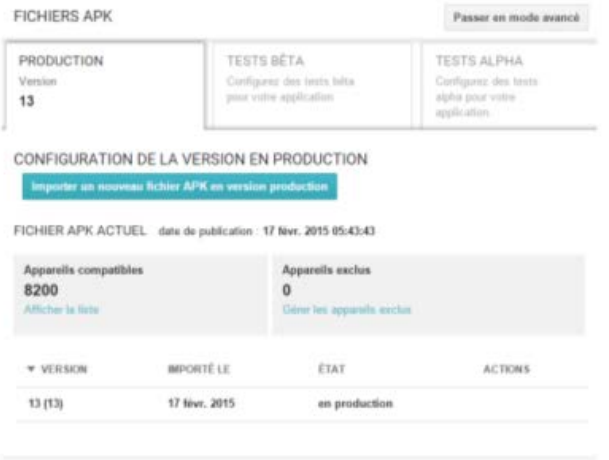
Une fois la configuration du SDK terminée, vous pouvez recompiler votre jeu pour générer un nouveau fichier APK. La seule chose à respecter est de modifier le numéro de la version de votre application afin de pouvoir la publier sur le Play Store.

Figure 14.6 : Version de l'APK



Ensuite, retournez dans la console développeur Android et importez-y le nouveau fichier APK.

Figure 14.7 : Mise à jour de l'application dans la console



Comme pour le premier enregistrement, vous aurez un délai d'attente avant que les modifications soient effectives. Les utilisateurs seront ensuite invités à télécharger la nouvelle version de votre application.

AdBuddiz met à jour les statistiques toutes les 24h, vous pourrez voir chaque jour les revenus qui auront été générés. Les régies publicitaires s'adaptant selon votre jeu et vos joueurs, vous devez attendre quelques jours avant d'avoir les résultats réels.

Dans ce chapitre, vous avez vu comment :

- choisir une régie publicitaire ;
- intégrer le SDK d'AdBuddiz ;
- afficher des publicités dans votre jeu ;
- mettre à jour votre application ;
- suivre les statistiques.

15

Les services de jeux

Les services de jeux sont devenus incontournables. Ils permettent de mettre en place des fonctionnalités qui améliorent l'expérience des utilisateurs. Ces fonctionnalités sont recherchées par les joueurs, il s'agit par exemple de classements mondiaux, de succès à débloquer ou d'un mode multijoueur. Nous allons voir comment intégrer cela à notre jeu.

Note › Google games services est l'outil officiel pour la mise en place des services de jeux sous Android. Du côté iOS, c'est le Game Center.

Intégrer un service de jeux tel que Google Games services permet d'améliorer l'expérience utilisateur et ainsi d'obtenir plus de téléchargements et plus de satisfaction. Les jeux qui font partie de Google Games services se remarquent par des icônes sur la fiche produit. Les joueurs les recherchent spécifiquement car cela leur permet d'être en compétition grâce au classement mondial ou aux challenges à obtenir. Google Games services vous permet aussi d'analyser vos joueurs pour savoir d'où ils viennent, combien de temps ils jouent à votre jeu, quel est le modèle de leur téléphone, quel est leur niveau préféré, etc. Cela vous aidera à améliorer votre produit pour mieux répondre à leurs attentes. Dans ce chapitre, je vais vous montrer comment créer un classement mondial et comment créer un système de succès afin que vous ayez les bases pour bien vous lancer.

15.1. Intégrer Google Games services

Pour mettre en place Google Games services, votre application doit d'abord être publiée sur Google Play. En effet, sa publication vous fournit des éléments nécessaires à l'activation du service de jeux. Vous devrez ensuite paramétrer l'application dans Google Play et mettre en place un plug-in que j'ai développé. Ce sont les deux étapes que nous allons réaliser ensemble au cours de cette section.

Pour commencer, vous devez déclarer votre jeu au service de jeux via la console développeur. Rendez-vous dans le menu SERVICES DE JEUX et cliquez sur AJOUTER UN JEU.

Figure 15.1 : Ajout d'un jeu dans Google Games services

Renseignez le nom et la catégorie de votre jeu pour pouvoir passer à l'étape suivante. Puis vous devez en fournir une description et ajouter des images pour préparer sa fiche de présentation. Dans la suite de ce chapitre, j'utiliserai un jeu que j'ai développé et qui est en ligne pour vous présenter les différentes étapes. En effet, le jeu développé au cours de ce livre n'a pas été mis en ligne, je me baserai donc sur le jeu CAT SIMULATOR.

Figure 15.2 : Formulaire d'inscription à Google Games services

Après avoir enregistré vos modifications, cliquez sur le menu APPLICATIONS ASSOCIÉES à votre gauche et sélectionnez ANDROID pour le type d'application. Ensuite, dans le champ NOM DU PACKAGE vous devez aller chercher votre application. Voici en exemple une capture d'écran de la configuration d'un de mes jeux Android :

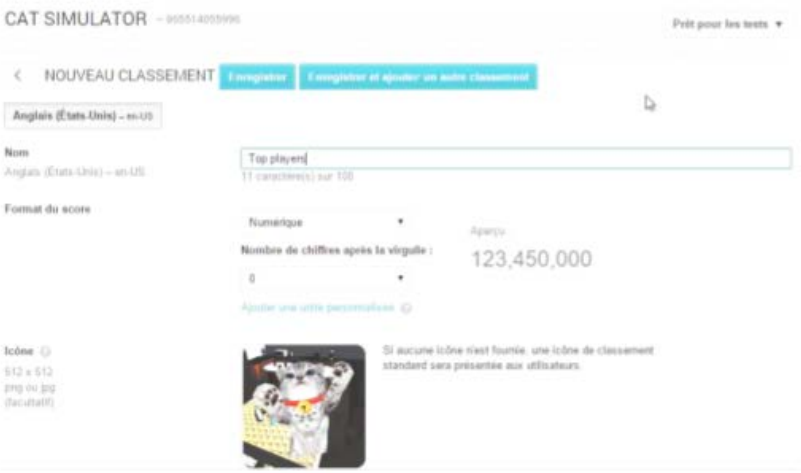
Figure 15.3 : Association de l'application

Enregistrez et cliquez sur le bouton AUTORISER, puis poursuivez la procédure en cliquant sur CONTINUER. Vous arrivez sur une nouvelle page où vous devez cliquer sur CREATE CLIENT pour terminer la configuration. Nous pouvons passer à la partie intéressante du chapitre !

15.2. Création d'un classement et de réussites

Nous allons créer un classement mondial et cinq réussites pour notre jeu. Rendez-vous dans le menu classement et cliquez sur AJOUTER UN CLASSEMENT. Vous arrivez sur une page où vous devez spécifier le nom du classement, son icône, et définir des paramètres optionnels afin de terminer la configuration :

Figure 15.4 : Création du classement



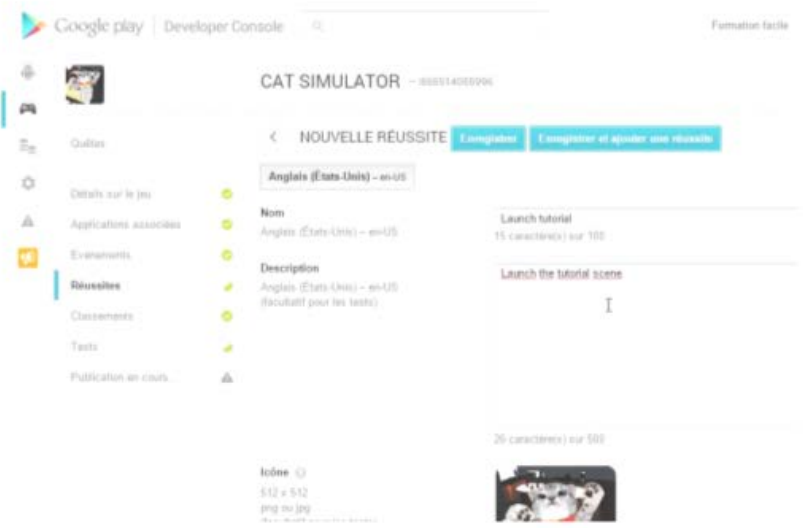
Enregistrez vos modifications. Vous obtenez une clé d'identification de votre classement qui sera utilisée dans Unity pour envoyer le score du joueur à celui-ci.

Figure 15.5 : Clé du classement

Passez maintenant dans la rubrique RÉUSSITES afin d'ajouter cinq réussites à votre jeu. Nous pouvons par exemple ajouter les réussites suivantes : Lancer le jeu - Ramasser un pot de peinture - Ramasser une pièce - Finir un niveau - Finir le jeu.

Vous devez au minimum créer cinq réussites pour que ce soit valide. Pour créer une réussite, attribuez-lui un nom, une description, une icône ainsi qu'un nombre de points :

Figure 15.6 : Création d'une réussite



Répétez cette opération cinq fois. Une fois terminé, vous obtenez quelque chose qui devrait ressembler à cela :

Figure 15.7 : Liste des réussites

Allez ensuite dans le menu publication et cliquez sur PUBLIER VOTRE JEU pour envoyer les modifications sur le Play Store :

Figure 15.8 : Publication des modifications

15.3. Mise en place des services au niveau de l'application

Pour mettre en place les services de jeux au niveau de notre application, nous allons recourir à un plug-in que j'ai développé. En effet, cette étape est assez complexe du fait que Google ne propose pas de kit compatible de façon native avec Unity. Il est donc obligatoire de développer des fonctions en Java, de créer un package avec Eclipse et d'intégrer ce package (fichier `.jar`) à votre projet Unity. Ce travail est assez fastidieux et pas facile à faire même pour des développeurs expérimentés.

Ce plug-in va nous simplifier grandement le travail. Il contient énormément de scripts et de fonctions qui gèrent tout à notre place et permet d'intégrer les games services très rapidement et très facilement.

Dans Unity, importez et installez le plug-in `GameServicesUnity` que j'ai développé pour la mise en place des games services. Vous pouvez le télécharger sur mon site.

Attention *› Vous pouvez rencontrer des erreurs si vous utilisez plusieurs plug-ins. En effet, il ne peut y avoir qu'un seul manifest dans votre projet. Si vous utilisez d'autres plug-ins, vous aurez à modifier manuellement le fichier manifest pour ajouter les bonnes permissions et les bonnes activities.*

La première chose à faire est de cliquer sur le nouveau menu GOOGLE PLAY GAMES/ANDROID SETUP :

Figure 15.9 : Android setup

Une fenêtre va s'ouvrir et vous demandera d'entrer l'ID de votre application. Celui-ci se récupère dans la console développeur Google Play à droite du nom de votre application :

Figure 15.10 : Récupération de l'ID de l'application

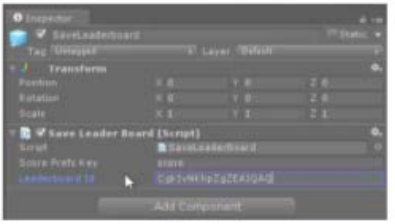
Entrez le code et validez. Vous pouvez désormais utiliser les Games services.

Voyons maintenant comment envoyer en ligne le meilleur score du joueur. Ce score doit impérativement être stocké dans les `PlayerPrefs`. Par exemple, dans mon cas, j'ai un score qui est stocké de la façon suivante : `PlayerPrefs.SetInt("score", monScore);`. La clé est donc le mot `"score"`. C'est très important de garder de côté cette clé.

Pour envoyer le score du joueur en ligne, vous devez faire glisser le prefab `SaveLeaderboard` sur la scène de votre menu principal. Le prefab se trouve dans le dossier `Scripts`. Le script prend deux paramètres : le nom de votre clé des `PlayerPrefs` (pour moi il s'agit de `"score"`) et l'ID de votre classement que vous pouvez récupérer dans la console développeur. Voilà ce que cela donne :

Note : *Le prefab `SaveLeaderboard` du plug-in est un objet déjà configuré qui contient les scripts qui vont gérer la transaction du score de l'application vers les serveurs de Google en ligne. Le script placé sur ce prefab vérifie que le score est supérieur au précédent et si tel est le cas, le nouveau meilleur score est envoyé en ligne. Google se charge ensuite de créer un classement avec tous les scores qui ont été envoyés par tous les joueurs.*

Figure 15.11 : Envoyer le score



Note › *Encore une fois, le plug-in va nous simplifier la vie pour cette étape. Le plug-in va faire le lien entre votre jeu et les services de jeux de Google afin d'indiquer à Google que le joueur a débloqué telle ou telle réussite.*

Le plug-in va se charger d'envoyer le score du joueur en ligne. Pour paramétrer les réussites, c'est tout aussi simple. Vous devez utiliser le script `UnlockAchievement` qui se trouve dans le dossier `Scripts` du plug-in. Ce script permet de débloquer une réussite si un événement se produit. Par exemple, si vous voulez débloquer la réussite *Obtenez un score supérieur ou égal à 10* vous devez placer le script sur une scène et le configurer de la façon suivante :

Figure 15.12 : *Débloquer une réussite*

Le premier paramètre du script est l'ID de la réussite à débloquer. Cette fois encore, vous récupérerez l'ID dans la console développeur à côté de la réussite en question. Le deuxième paramètre est le type d'événement. Il s'agit soit d'un clic sur un objet soit d'une condition. S'il s'agit d'une condition, par exemple `score = 10`, vous devez utiliser les deux autres paramètres suivants : le troisième est la clé du `PlayerPrefs` qui doit être pris en compte dans la condition et le quatrième la valeur à atteindre pour satisfaire la condition. Dans notre cas, nous souhaitons avoir un score de 10.

Vous pouvez ainsi configurer les différentes réussites que vous avez créées. Une fois terminé, vous devez ajouter le script `LoginGP` sur un objet du menu principal de votre jeu. Ce script va permettre au joueur de se connecter avec son compte aux services de jeux.

Note › *Le script `LoginGP` permet au joueur de se connecter à son compte Google afin de bénéficier des services de jeux proposés. Le script va transférer les informations à Google afin de vérifier si l'e-mail et le mot de passe sont bons afin d'identifier le joueur pour associer les scores à ce joueur.*

Lorsque le joueur lance le jeu, une fenêtre pop-up s'ouvre et lui propose de se connecter s'il possède un compte Google (voir Figure 15.13).

Figure 15.13 : Connexion à Games services

Une fois connecté, un message de bienvenue apparaît afin d'informer le joueur que la connexion est réussie.

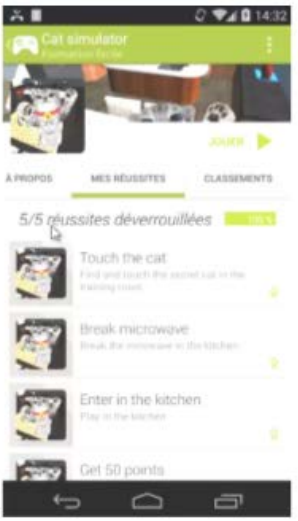
Figure 15.14 : Message de bienvenue

Le message de la Figure 15.15 apparaît lorsque le joueur débloque une réussite. Google indique au joueur ce qu'il a débloqué et combien de points il a obtenu.

Figure 15.15 : Réussite déverrouillée

Sur la page de l'application dans Google Play, le joueur peut consulter l'ensemble des scores qu'il a obtenu et l'ensemble des succès déverrouillés.

Figure 15.16 : Application games services



Dans ce chapitre, vous avez vu comment intégrer les services de jeux Google grâce à l'utilisation d'un plug-in. Vous savez :

- mettre en place un classement mondial ;
- créer des réussites ;
- gérer la connexion du joueur à Google.

16

Améliorer son application

Une fois publiées, vos applications doivent être maintenues et améliorées. Google et Apple attachent beaucoup d'importance aux mises à jour. Si votre jeu est mis à jour, cela signifie que vous apportez des nouveautés, que vous corrigez des bugs, que vous continuez à travailler dessus même après sa sortie et que vous continuez à l'améliorer. Les applications à jour sont plus mises en avant sur les stores que les applications abandonnées par les développeurs. Il est donc essentiel de faire évoluer votre jeu et de répondre aux attentes des joueurs.

16.1. Suivre les recommandations

Vous aurez certainement beaucoup d'idées pour améliorer votre application avec les retours des joueurs. Mais avant de réfléchir à quoi faire, vous pouvez tout simplement suivre les recommandations de Google, aussi appelées *conseils d'optimisation*. Lorsque votre application est en ligne, Google analyse le comportement des utilisateurs et vous fait ensuite des recommandations, qui apparaîtront dans votre console développeur.

Figure 16.1 : *Les recommandations de Google*

Google vous dira si vous devez traduire votre jeu dans une langue populaire, si vous devez changer les images ou l'icône et si vous pouvez améliorer des choses dans votre application. Le mieux est d'essayer de suivre toutes les recommandations qui vous seront faites.

Google analyse également les plantages de votre application, et vous pouvez lire les logs afin de détecter d'éventuels bugs :

Figure 16.2 : Les plantages

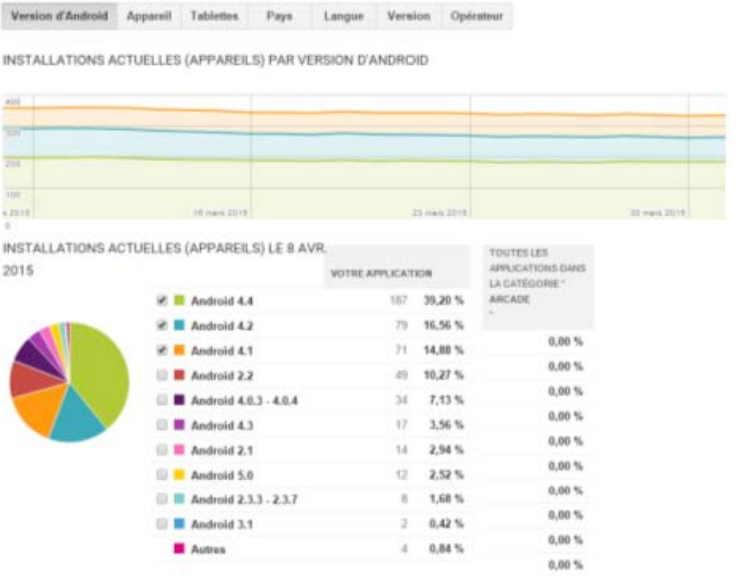
Vous y trouverez des informations importantes comme la version d'Android utilisée par le joueur, la date du plantage ou encore le bout de code au centre du plantage.

Les commentaires des joueurs sont aussi très importants. Vous devez lire tous les commentaires afin de prendre en compte ce que pensent les utilisateurs. Les commentaires vont vous aider à cerner ce qui va et ne va pas, ainsi vous pourrez adapter votre jeu. Lorsqu'un utilisateur vous met une mauvaise note, essayez de voir ce que vous pouvez faire pour améliorer son confort et répondez à son commentaire lorsque vous aurez apporté les modifications. Vous devez avoir les meilleures notes possibles afin d'obtenir plus de téléchargements. Vous ne pourrez pas répondre aux attentes de tout le monde, mais essayez de faire au mieux.

16.2. Lire les statistiques

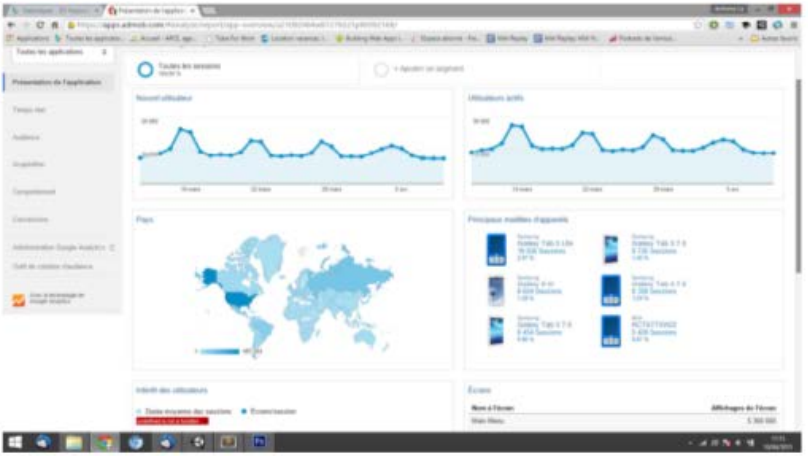
Les statistiques sont importantes afin de savoir quel est le public touché par votre jeu. Vous devez savoir quelle version d'Android vos joueurs utilisent, quel est leur âge moyen, quelle langue parlent-ils, dans quel pays ils se trouvent, etc. Cela vous permettra de savoir s'il faut traduire votre jeu dans une autre langue ou s'il faut le rendre compatible avec plus de versions d'Android par exemple.

Figure 16.3 : Les statistiques



Si vous utilisez un outil comme Analytics, ce que je vous conseille fortement de faire, vous aurez accès à de très nombreuses statistiques qui vous aideront à améliorer votre produit. Analytics fournit énormément d'informations et de données essentielles pour connaître le comportement exact des joueurs. Voici la page d'accueil avec le tableau de bord pour une application :

Figure 16.4 : Google Analytics



Avec Analytics, vous pouvez aussi voir en temps réel ce que font tous les joueurs qui jouent actuellement à votre jeu :

Figure 16.5 : Analyse en temps réel

L'analyse du comportement de vos joueurs est le meilleur moyen de détecter les points forts ainsi que les points faibles de votre application.

16.3. Ajouter des nouveautés

Si votre jeu fonctionne bien et que de nombreux joueurs l'ont téléchargé, vous devez l'enrichir afin de le maintenir au top niveau le plus longtemps possible. Si vous n'apportez pas des nouveautés à votre jeu, les joueurs vont se lasser et arrêter de jouer petit à petit. Vous pouvez augmenter la durée de vie et sa rejouabilité en ajoutant des fonctionnalités au cours des mises à jour.

Après avoir analysé vos joueurs, vous pouvez repérer ce qu'ils aiment et ajouter du nouveau contenu en fonction de cela. Vous pouvez ajouter de nouveaux niveaux, de nouvelles quêtes, de nouvelles missions ou tout simplement de nouveaux personnages ou décors. Il faut montrer aux joueurs que votre application est vivante et change régulièrement.

Vous devez vous faire remarquer afin d'être dans le top 10 des meilleurs jeux. Vous devez profiter des événements et des fêtes de l'année. Par exemple, la semaine d'Halloween, pensez à changer les décors, à ajouter des citrouilles ou des chauves-souris, afin de célébrer l'événement. Faites cela également pour Noël ou le Nouvel An. Vous pouvez aussi profiter des événements sportifs comme la coupe du monde de football pour changer les couleurs. Pensez à modifier la fiche de votre application afin de dire que vous avez ajouté de nouveaux décors.

Vous devez toujours annoncer les nouveautés dans la description de votre application. Si c'est Noël, vous pouvez être sûr que de nombreux utilisateurs vont effectuer la recherche "noël" sur Google. Alors profitez-en pour ajouter ce mot clé dans votre description un peu avant l'événement afin d'être référencé sur celui-ci et attirer de nouveaux joueurs.

Lorsque les modifications sont faites, vous pouvez compiler votre jeu et envoyer la nouvelle version de l'APK sur le Play Store :

Figure 16.6 : Mise à jour de l'APK

La mise à jour sera effective sous 24 heures.

Dans ce chapitre, vous avez vu comment apporter des modifications à votre application afin de l'améliorer et d'attirer toujours plus d'utilisateurs. Vous avez vu comment :

- suivre les conseils d'optimisation ;
- lire les statistiques ;
- profiter d'un événement pour faire le buzz.

Foire aux questions

1. Comment améliorer les performances de mon jeu ?

Les terminaux mobiles sont en général peu puissants. Afin de créer des jeux fluides, vous devez optimiser vos scènes. Le plus important est de limiter le nombre d'objets que vous utilisez. Ensuite, vous devez utiliser des modèles 3D simples, des petites textures légères et diminuer la distance de vision de la caméra. Vous pouvez également fusionner plusieurs petits objets pour en obtenir un seul plus grand et plus simple à calculer. Voir Chapitre Optimiser les scènes.

2. Comment charger une autre scène sans "bloquer" la scène actuelle ?

Lorsque vous chargez une nouvelle scène avec `Application.LoadLevel`, le jeu se "bloque" le temps du chargement. Afin de créer une transition douce, vous pouvez utiliser la fonction `LoadLevelAsync` pour charger une autre scène en arrière-plan de façon transparente.

3. Peut-on créer un terrain dans un jeu mobile ?

Les terrains sont des objets 3D complexes. Ils sont à éviter si vous créez un jeu mobile, car la plupart des téléphones n'ont pas assez de puissance pour les calculer. Cependant, il existe des outils comme `Terrain4Mobile` (payant) qui permettent d'optimiser les terrains afin de les rendre compatibles avec les mobiles.

4. Comment créer facilement des animations par script ?

`iTween` est un plug-in gratuit et très puissant pour Unity qui vous permet de créer des transitions et des effets par script. Vous pouvez retrouver l'ensemble des fonctions du plug-in en consultant la documentation officielle.

5. Comment enregistrer un son via le microphone ?

Vous pouvez enregistrer un son via le micro et le jouer comme nous l'avons vu dans le premier module, au chapitre *Créer des effets spéciaux*. Pour cela, vous pouvez utiliser la fonction suivante :

```
AudioSource aud = GetComponent<AudioSource>();
aud.clip = Microphone.Start("Built-in Microphone", true, 10, 44100);
aud.Play();
```

6. Comment utiliser la webcam ou la caméra du téléphone ?

Vous pouvez capturer la vidéo en passant par la webcam ou la caméra du téléphone. La vidéo peut être vue en direct sur un objet 3D comme par exemple un cube. Voilà la fonction à utiliser :

```
WebCamTexture webcamTexture = new WebCamTexture();
Renderer renderer =  GetComponent<Renderer>();
renderer.material.mainTexture =  webcamTexture;
webcamTexture.Play();
```

7. Comment faire une capture d'écran en jeu ?

Il est possible de proposer au joueur de faire une capture d'écran pendant le jeu. La photo sera enregistrée dans un fichier image. Pour cela, il existe la fonction suivante :

```
void OnMouseDown() { Application.CaptureScreenshot("Screenshot.png"); }
```

8. Comment ouvrir une URL ?

Vous pouvez ouvrir un site web en demandant à Unity d'ouvrir une URL via la fonction `Application.OpenURL("http://unity3d.com/");` .

9. Comment charger du contenu additionnel dans un jeu ?

Vous pouvez charger du contenu supplémentaire dans vos scènes en passant par la fonction `Application.LoadLevelAdditive("PlusDeChoses");` . Pour cela, vous devez

créer deux scènes et utiliser la fonction ci-dessus pour charger le contenu de la scène 2 dans la scène 1. Cela peut être utile pour charger un monde virtuel de façon progressive.

10. Comment conserver un objet après changement de scène ?

Dans certains cas, vous pouvez souhaiter conserver des objets même après changement de scène. Par exemple, si vous avez un objet qui contient un script gérant l'inventaire du joueur et que vous voulez garder cet objet dans toutes vos scènes, vous pouvez utiliser la fonction `void Awake() { DontDestroyOnLoad(transform.gameObject); }`.

11. Comment empêcher l'écran de se mettre en veille ?

Si vous ne touchez pas l'écran tactile pendant quelques secondes, celui-ci se met en veille et s'éteint. Pour éviter cela, vous pouvez utiliser la fonction suivante :

```
Screen.sleepTimeout = SleepTimeout.NeverSleep;
```

Liste des illustrations

Index

M

N

O

P

R

S

T

U

V

W

X