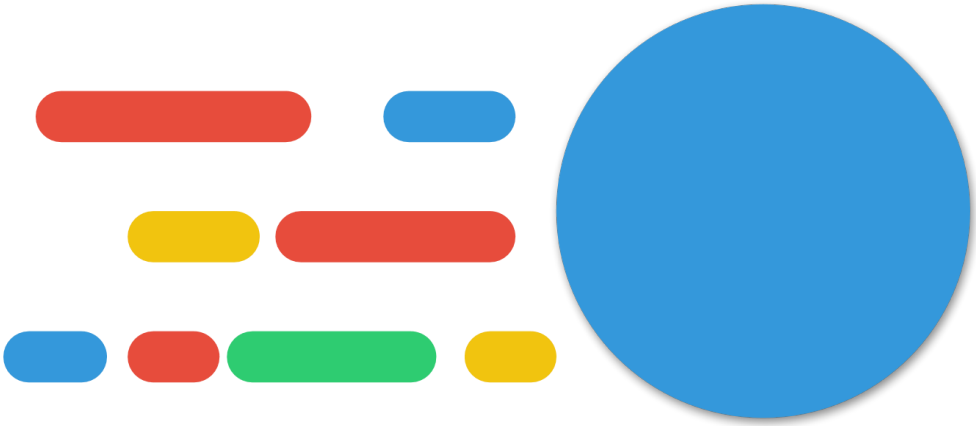
Rapport de soutenance



Colors

Rémi VERNAY

P1D

Alexandre BERMUDEZ

Jean-Baptiste DESPUJOL

Romain GREGOIRE

Alex POIRON

Sommaire

1. Introduction 3

2. Reprise du cahier des charges 4
2.1. Mises à jour. 4

3. Réalisation du projet. 5
3.1. Graphisme/Design 5
3.2. Site Internet 7
3.3. Interface 9
3.4. Personnage 12
3.5. Obstacles 14
3.6. Shop/Comptes 15
3.7. Editeur de niveaux 17
3.8. Scènes UNITY. 18

4. Conclusion 19

1. Introduction

Avant de parler de la partie code et projet pur, nous allons reprendre les avis des membres du groupe sur le travail fourni par chacun. L'ambiance au sein du groupe est excellente et la répartition des tâches s'est faite de manière très organisée. Aucun des membres du groupe n'a failli à la tâche qui lui était attribuée, seuls des problèmes plus techniques sur l'utilisation de Unity ont pu retarder l'avancement de notre projet. Nous sommes dans l'ensemble satisfaits du travail fourni.

Colors étant un jeu où le joueur se déplace horizontalement de manière automatique (un side scroller), pour l'aspect développement nous nous sommes d'abord occupés du personnage puis de la réalisation des obstacles. Compte tenu de l'objectif que nous nous étions fixés, à savoir de faire un jeu aux graphismes et au design tous deux orientés vers un style **flat design.** Nous avons donc bien avancé sur le personnage et relativement bien sur les obstacles bien que nous devons à présent trouver de nouvelles formes pour ces derniers.

Pour ce qui est de l'interface nous avons également fait des progrès importants puisque nous avons réalisé le menu du jeu et nous sommes d'ores et déjà capables de sélectionner les différentes options qui apparaissent à l'écran. De même, nous avons déjà réalisé le squelette du site internet et il ne reste donc plus qu'à le remplir. Pour ce qui est du shop ainsi que de l'éditeur nous ne nous étions fixé aucun objectif en particulier, nous avons seulement entamé certaines recherches sur comment nous pourrions implémenter le shop, mais cette tâche semble assez complexe pour le moment.

2. Reprise du cahier des charges

Voici le tableau d'avancement pour le projet Colors présenté dans le cahier des charges:

Tâches / Soutenances	Soutenance 1	Soutenance 2	Soutenance 3
Graphisme / Design	30%	90%	100%
Site Internet	40%	70%	100%
Interface	20%	70%	100%
Personnage	40%	80%	100%
Obstacles	30%	70%	100%
Shop / Comptes	0%	60%	100%
Editeur de niveaux	0%	50%	100%
Scènes UNITY	20%	60%	100%

Comme évoqué précédemment dans l'introduction, suite à certaines discussions, nous avons décidé de modifier certains pourcentages notamment au niveau des graphismes, du site internet et des obstacles. Pour les graphismes, nous avons directement eu des idées précises dans nos têtes et nous voulions tout créer sans télécharger de presets graphiques présents sur Internet. Ainsi nous étions motivés et nous avons commencé à réaliser tous les graphismes liés au menu du jeu. Alexandre et Alex en parleront plus précisément respectivement en tant que responsable et suppléant. En conséquence, voici le nouveau tableau d'avancement:

Tâches / Soutenances	Soutenance 1	Soutenance 2	Soutenance 3
Graphisme / Design	70%	90%	100%
Site Internet	30%	70%	100%
Interface	20%	70%	100%
Personnage	40%	80%	100%
Obstacles	20%	70%	100%
Shop / Comptes	0%	60%	100%
Editeur de niveaux	0%	50%	100%
Scènes UNITY	20%	60%	100%

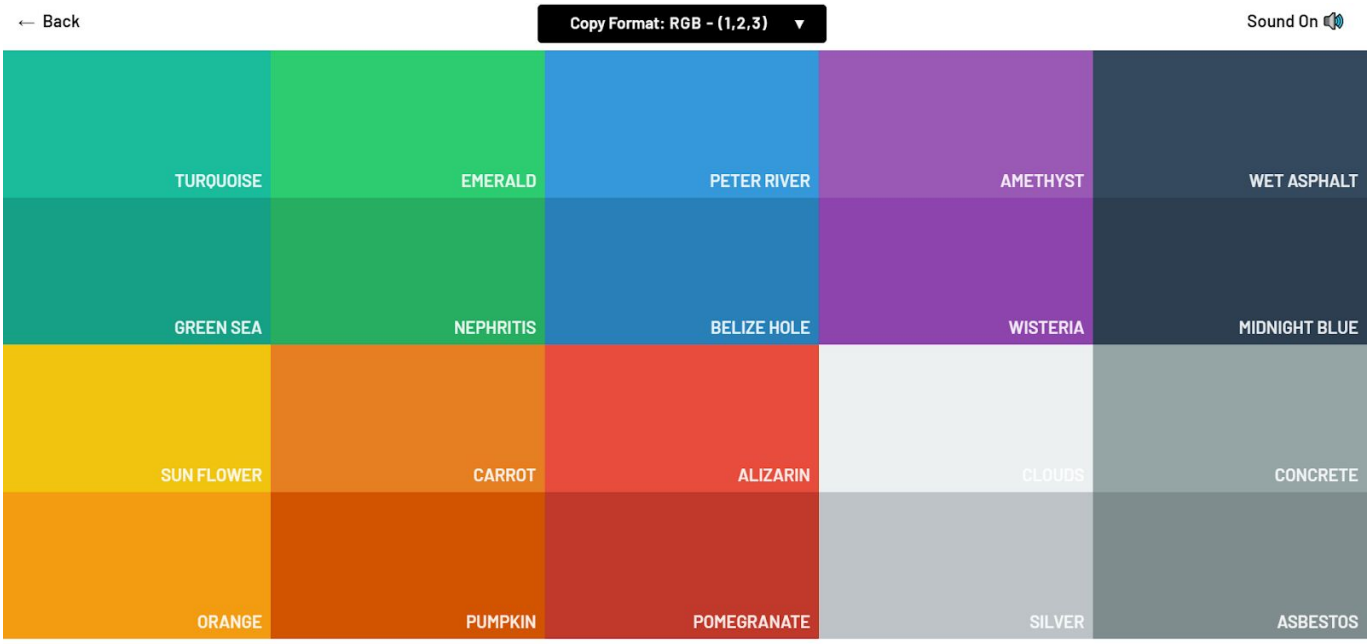
3. Réalisation du projet

3.1 Graphisme/Design (Alexandre, Alex)

Avec Alex, nous voulions entièrement créer le design du jeu et n'utiliser aucun graphisme déjà présent sur Internet. Cela nous permet d'être plus créatifs et d'être réellement plus libres. Ainsi nous nous sommes séparés les tâches.

Pour chaque graphique créé nous avons utilisé deux sites internet. Le premier est un site de palettes de couleurs qui permet d'utiliser des couleurs qui concordent entre elles. On peut ainsi récupérer le code RGB de chaque couleur qui peut également servir dans certaines lignes de code. Nous y reviendrons plus en détails dans la partie Obstacles.

Pour tous les graphismes actuellement faits, nous utilisons la Flat UI Palette V1 suivante :



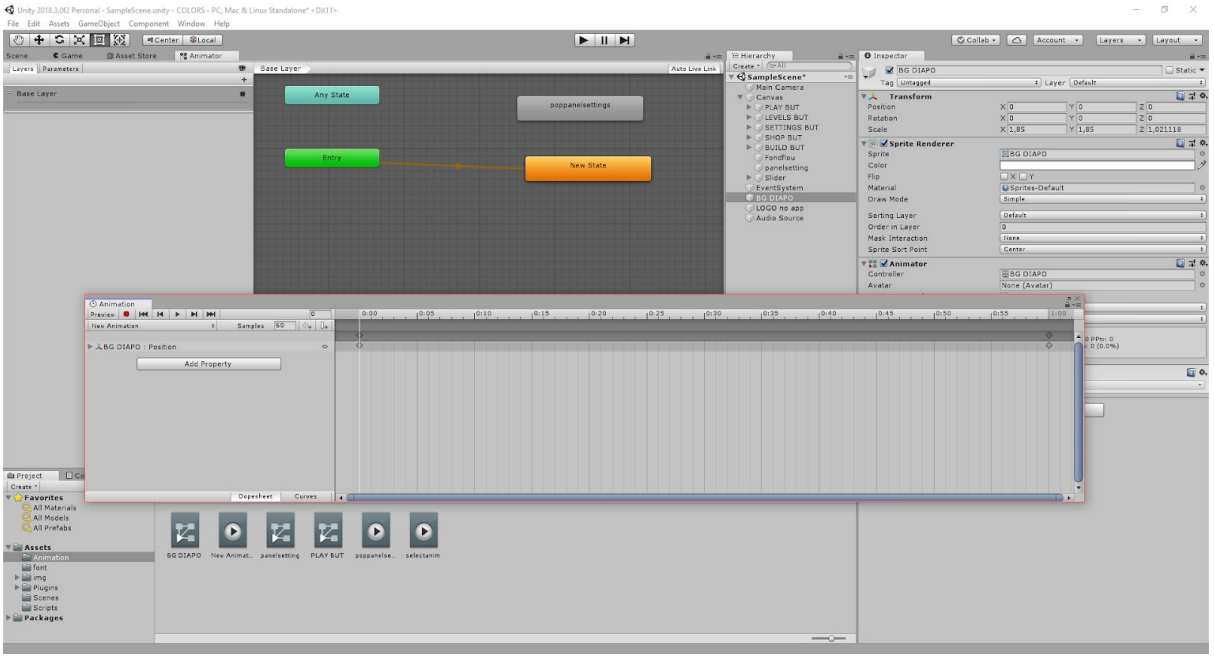
L'utilisation d'une palette est très importante pour conserver une cohérence générale, c'est l'un des facteurs les plus importants qui définit un bon design. Fort de cela, nos interfaces se ressemblent pour ne pas perdre l'utilisateur ainsi il saura directement quel bouton utiliser.

Le deuxième est un site regroupant des milliers d'icônes flat design. Ce site est extrêmement pratique pour notre projet. En effet, nous pouvons télécharger des images pour représenter des boutons ou options. Elle sont dans un format vectorisé ce qui permet d'avoir une qualité infinie et donc de ne pas avoir d'image pixelisée ou floue. On peut les modifier avec Photoshop, nous pouvons donc customiser l'image à notre guise.

Nous nous inspirons énormément du Material design, il est utilisé par google pour Android et tous ses services et pages web. Il est donc grandement accepté par tout le monde, l'utilisateur va donc retrouver des principes de base et être face à une interface intuitive. Ce design s'inspire du monde réel et de ses textures, notamment de la manière dont elles reflètent la lumière et projettent les ombres.

La police est claire et moderne et souvent en gras en jouant avec des gradients de gris pour exprimer l'importance du texte. Malgré son uniformité, nous pouvons avoir une identité grâce au choix de la police, la palette de couleurs ou même la disposition des éléments.

Un des aspects les plus importants est l'animation : il faut qu'elle soit fluide et réaliste. Nous avons donc utilisé les courbes d'animation pour ajouter de l'inertie à notre objet. Cette méthode d'animation est la même que celle d'After Effect.



3.2 Site Internet (Jean-Baptiste, Alexandre)

N'ayant jamais codé en HTML et CSS, ces deux langages étaient nouveaux pour nous. En effet, la notion de balisage bien que simple, nous a tout de même pris quelque temps à être automatique. Également, il fut difficile de se dire qu'un langage est dédié au contenu tandis que l'autre prend en charge la mise en forme. Mais au fil de leur découverte, nous avons rapidement compris que ces deux langages fonctionnaient très bien ensemble et se complétaient.

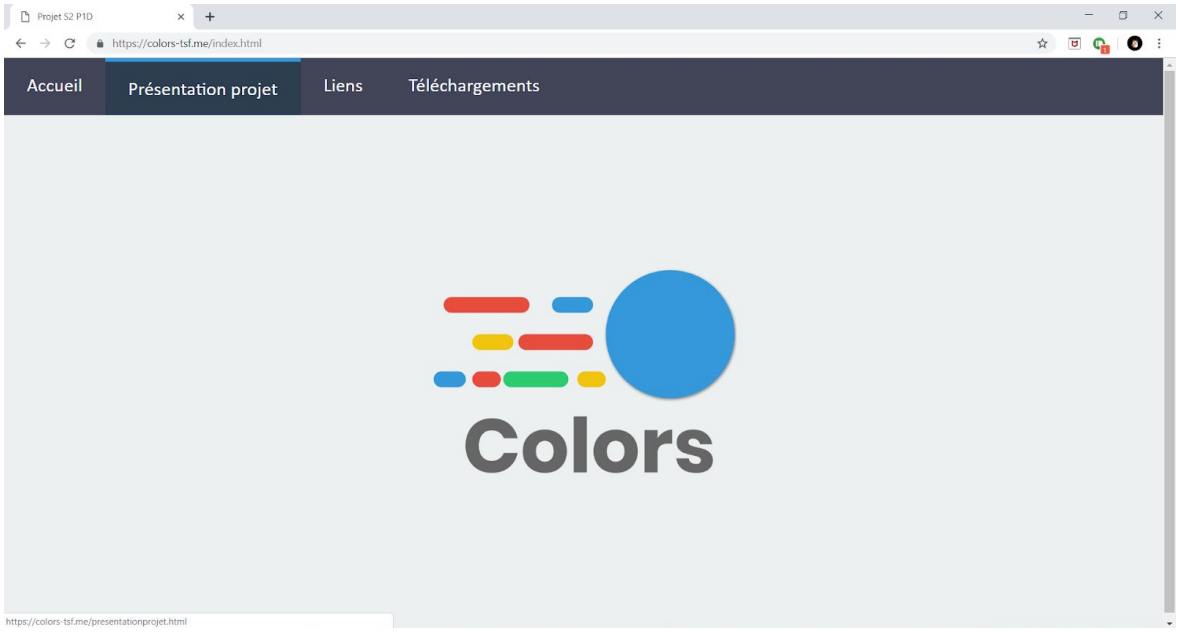
C'est grâce à des tutoriels vidéo que nous avons pu apprendre petit à petit comment créer la base d'un site et comment mettre en page ce dernier afin de le rendre agréable visuellement. Au début de notre apprentissage, nous avons suivi des vidéos entières afin de connaître et comprendre les bases du HTML et du CSS. Au fil du temps, nous avons commencé à sélectionner les éléments qui nous semblaient importants car nous n'allions pas avoir le temps de tout couvrir avant la première soutenance. Ainsi, après cela, nous avons recherché spécifiquement des éléments de mise en forme. Par exemple, un moyen de créer un menu horizontal afin de rendre la navigation plus intuitive.

Pour ce qui est du code pur, nous nous sommes rendus compte que la compréhension de la notion de classe (même si elle est moins poussée qu'en C#) était primordiale car elle revient souvent dans ce type de développement. Sans cette notion, impossible de choisir exactement quelle zone de texte mettre en telle ou telle police.

Tout d'abord nous avons mis en fond le logo de notre projet afin que le site soit facilement identifiable à notre groupe. Nous avons créé une liste afin

d'exposer les différents onglets pour accéder aux différentes pages du site. Cette liste étant de base disposée verticalement, nous l'avons codée de manière à ce qu'elle s'affiche de façon horizontale pour plus d'esthétisme. Ce menu est commun à tous les onglets afin de pouvoir naviguer vers n'importe lequel de ces derniers. Nous avons codé un affichage alternatif lorsque la souris est passée sur les onglets. Pour cet affichage, nous avons utilisé les couleurs du logo afin de créer une concordance visuelle générale.

A cause du temps d'adaptation aux langages, la recherche et la mise en forme du côté esthétique nous a pris beaucoup de temps. Nous voulions produire un site propre et épuré, à l'image de notre jeu. C'est pour cette raison que le site manque encore de contenu. Mais nous sommes satisfaits du résultat visuel actuel.

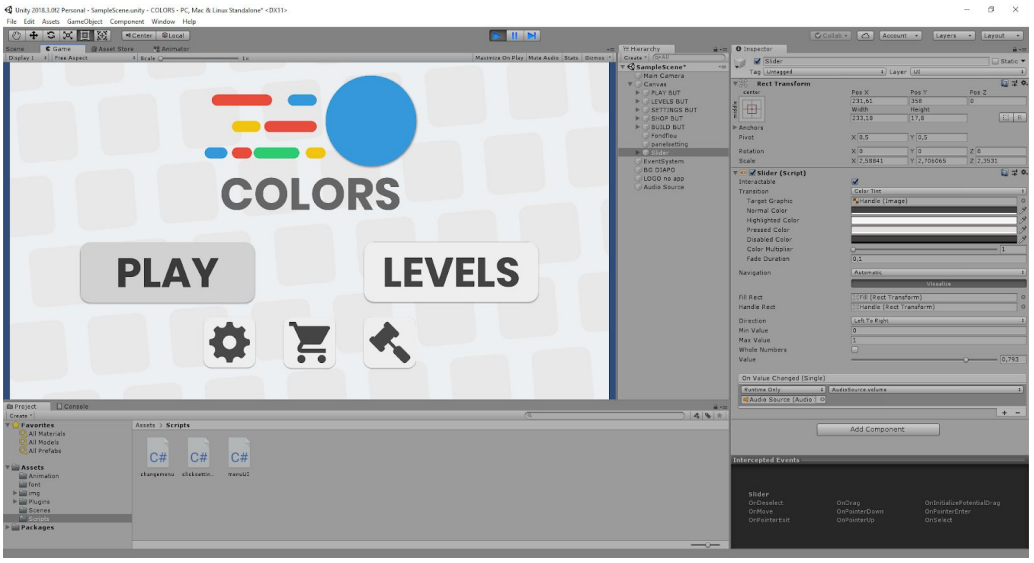


Notre objectif à réaliser pour la prochaine soutenance au niveau du contenu est de remplir le site internet avec les informations nécessaires. Et pour ce qui est de l'esthétisme du site, nous pensons continuer dans la lancée actuelle afin que toutes les pages du site soient uniformes pour rester en accord avec le design général du projet.

3.3 Interface (Alex , Jean-Baptiste)

Depuis la remise du cahier des charges, nous avons réalisé le menu général du jeu. Nous avons donc intégré les graphismes dans Unity et avons tout découpé pour pouvoir faire des boutons interactifs. Pour chaque bouton, nous avons dû coder la sélection de celui-ci dans le menu par le joueur avec une manette de Xbox One.

Nous avons donc intégré les graphismes dans Unity et avons tout découpé pour pouvoir faire des boutons interactifs. Pour chaque bouton, nous avons dû coder la sélection de celui-ci dans le menu par le joueur avec une manette.

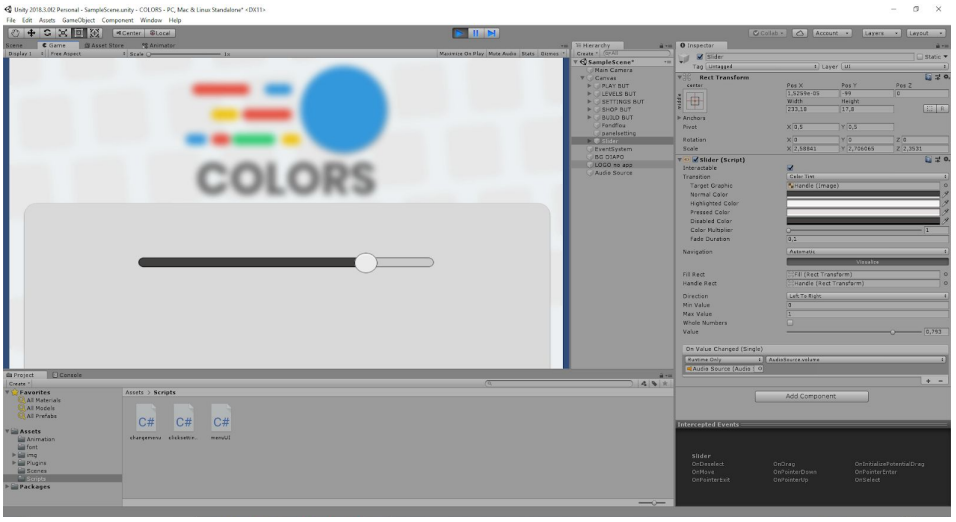


Lorsque le joueur se place sur un des 5 boutons vu au-dessus, celui-ci se fonce d'une teinte de gris, ce qui permet au joueur de savoir où se trouve son curseur.

Pour réaliser ceci, il faut comprendre que le joystick d'une manette possède la même représentation qu'un cercle trigonométrique classique. Les mouvements verticaux sont liés à l'axe des sinus et les mouvements horizontaux à l'axe des cosinus. Pour capter ce mouvement de joystick dans les lignes de code, on utilise la fonction **Input.GetAxis**. Nous n'avons pas utilisé les entrées utilisateurs (inputs) horizontales basiques de Unity car les paramètres entre le jeu en lui-même et dans le menu sont différents et nous voulions régler au mieux la sensibilité de la manette dans le menu pour que celui-ci soit fluide.

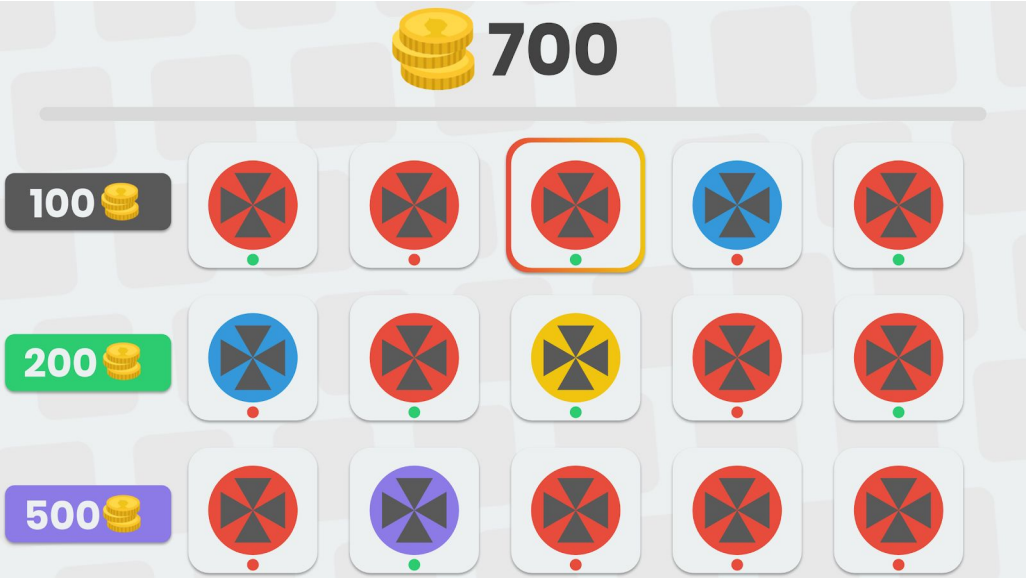
De plus, il fallait également coder le fait que le joueur puisse se déplacer dans le menu librement dans toutes les directions. Ensuite, nous avons créé la suite du menu quand nous avons choisi d'aller dans les paramètres du jeu. Pour cela, il fallait créer une animation avec une seconde fenêtre qui s'ouvre et flouter

le menu qui passe donc en arrière plan. Ensuite, nous avons rajouté le système de volume avec une barre (slider) qui doit aussi être contrôlé avec les joysticks de la manette.



Voici à quoi ressemble le menu du jeu lorsqu'on appuie sur le bouton des options. Le "slider" pour le volume est bien fonctionnel avec la manette si l'on bouge le joystick horizontalement.

Pour la prochaine soutenance, l'interface du menu des options sera complète avec l'ajout des boutons sur les commandes et sur les crédits qui ouvriront chacun un autre menu. Nous ferons de même pour les interfaces du menu des niveaux, du shop et de l'éditeur de niveau. Pour réaliser tout ceci, nous utiliserons le même système présenté précédemment avec le joystick de la manette. Néanmoins, cela devrait prendre un peu plus de temps puisqu'il y a beaucoup plus d'éléments à rendre interactifs avec le joueur dans le menu de la boutique qui ressemblera à l'image ci-dessous :



Par la quantité et la possibilité de boutons à faire interagir avec le joueur, nous pensons que le shop est le menu où l'interface sera la plus compliquée à gérer.

3.4 Personnage (Romain, Jean-Baptiste)

Afin de débuter la partie "Gameplay" de notre jeu nous avons entamé la réalisation du personnage. Pour cela, nous avons décidé de commencer à donner à notre personnage ses principes fondamentaux à savoir : le déplacement horizontal (side scrolling) du personnage, qui dans notre cas est automatique, les déplacements verticaux ainsi que les collisions avec les différents obstacles présents sur la carte.

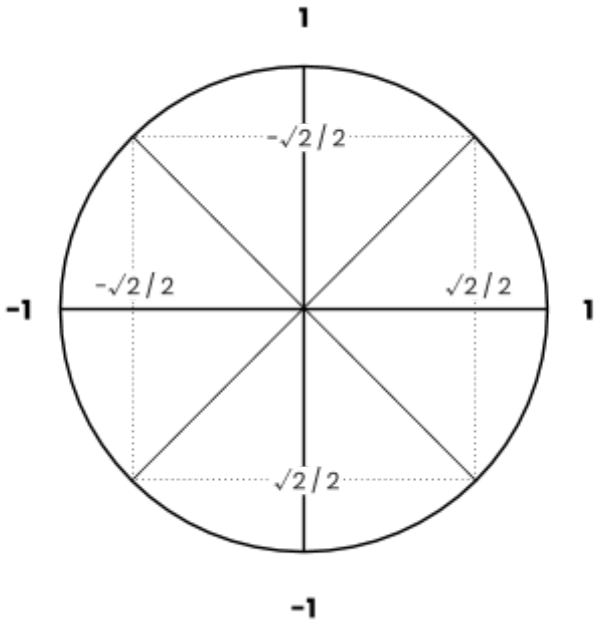
Tout d'abord, en adéquation avec le design épuré nous sommes partis sur une sphère afin de représenter notre joueur (comme dans la majorité des sidescroller). La sphère se déplace de gauche à droite en prenant de la vitesse au fur et à mesure du niveau. Pour cela, nous avons remplacé les caractéristiques de base des material2D, le **sphere de collision** afin de réaliser nous-mêmes les futures animations de collisions. Nous avons donc créer un **rigidbody2D** pour la sphère, à cela nous avons ajouté un script de manière à éditer nous mêmes les propriétés de notre personnage.

Nous avons donc commencé par donner une vitesse à notre personnage, dans l'esprit du Side Scrolling nous avons fait le choix d'augmenter la vitesse de notre sphère au cours du temps grâce à la fonction **transform.Translate(vitesse*(Time.time/5), 0,0);** Les modifications du vecteur vitesse restent néanmoins à ajuster en fonction de la distance de chaque niveau

ainsi qu'à la difficulté de ceux-ci. De plus, il est aussi important de noter que cette valeur ne peut pas augmenter de manière exponentielle le moteur de scène de unity bloque la vitesse maximum de notre projectile à un certains seuil, en raison de cela pour des questions de logique nous devrons réaliser des mesures de manière à ce que la vitesse maximum du projectile soit atteint sur les dernières frames du parcours.

Dans un second temps nous nous sommes attelés à créer les effets de collision avec les obstacles, dans notre jeu tout contact avec une surface n'étant pas de la même couleur que celle du personnage du joueur entraînera une défaite et renverra le joueur au début du niveau (pas de point de sauvegarde dans le niveau). Pour cela, malgré la dimension en 2D du jeu, nous avons utilisé des éléments en 3D afin que l'on puisse, à l'avenir, ajouter des animations et des effets lorsque les objets entreront en collision. En effet, si nous avions utilisé des objets 2D, donner un effet de mouvement agréable à jouer aurait été plus complexe puisque nous aurions eu besoin d'importer des images en 2 dimensions (sprites) déjà existantes. Avec les objets en 2D, les collisions seraient apparues moins naturelles tandis qu'avec des objets en 3D, grâce à leur masse, ils dégagent une impression de solidité pour les personnages et pour les obstacles.

Pour ce qui est des déplacements de notre personnage nous avons, comme dans le cas de l'interface, utilisé la fonction **Input.GetAxis()** mais aussi les fonctions **Input.GetKeyCode()** pour le clavier. Notre personnage peut donc se déplacer dans le plan vertical pour esquiver les obstacles.



La dernière méchanique qu'il reste à implémenter est celle de changement de couleur. Bien que nous ayons respecté les délais imposés par le cahier des charges compte tenu de l'avancement du personnage, nous avons tout de même fait le choix de commencer nos recherches pour récupérer les inputs de chaque

bouton de la manette. Le mécanisme sera donc assez simple puisque nous utilisons une manette de Xbox One chacun des boutons A B X Y représenteront une couleur que le personnage pourra prendre. Ces couleurs seront donc le rouge, le vert, le bleu et le jaune.

3.5 Obstacles (Romain, Alex)

Pour les obstacles nous avons réalisé le premier à savoir une boîte de collision (Box collider) animée se déplaçant de haut en bas. Il s'agit du premier style d'ennemi que l'on pourra rencontrer dans le jeu. Pour créer chacun des obstacles que nous créerons, nous utiliserons toujours les mêmes **composants** à savoir:

-**un rendu de mailles (mesh renderer)** qui permet de gérer les formes et la taille de notre obstacle

-**un animateur** pour déplacer l'obstacle

-**une capsule de collision** pour soumettre l'obstacle à la physique que nous appliquerons

-**un corps rigide (rigidbody)** pour les futures animations entre le personnage et les obstacles

Nous avons commencé à chercher des moyens pour que l'objet puisse traverser les obstacles si ceux-ci sont de la même couleur. Cela nous a donc mené jusqu'aux tags. Cette forme d'objet particulier à Unity nous permettra de reconnaître plus rapidement les différentes couleurs de chacun des obstacles présents sur le niveau de manière à ce que l'on ne soit pas forcé de déterminer les couleurs de chaque obstacle au moment du contact. Ceci, en effet, pourrait probablement nous faire perdre en fluidité puisque le jeu ne pourrait obtenir la couleur de l'obstacle qu'au moment de la collision

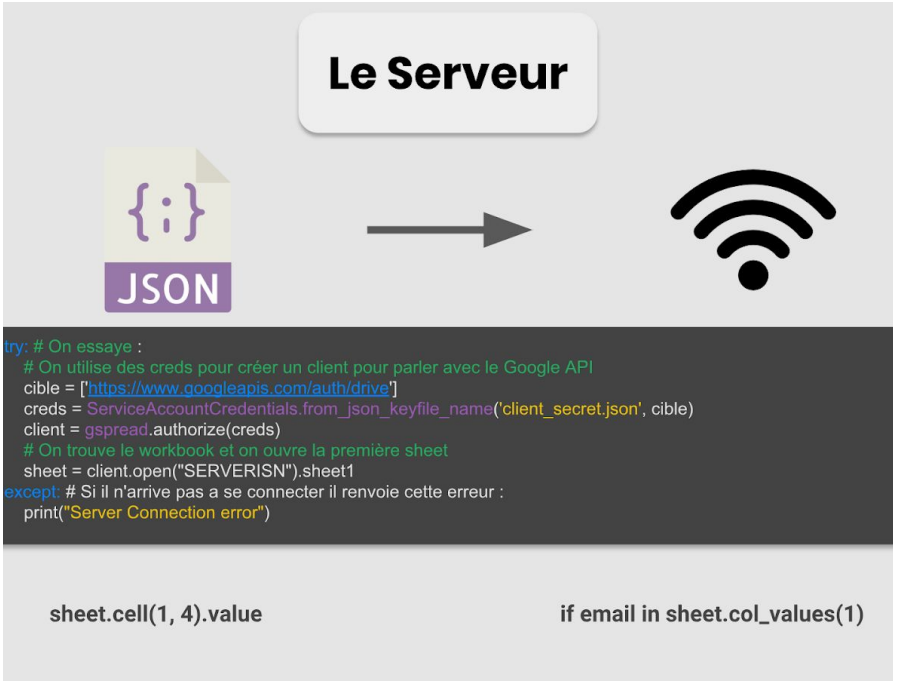
Créer les obstacles ne représentent donc pas une difficulté particulière en soit. Néanmoins nous avons décidé de réduire l'avancement auquel nous étions sur ce point, car pour respecter les consignes que nous nous sommes fixés sur le jeu nous voulons que l'aspect graphique de celui-ci et donc des obstacles soit le plus attrayant possible. Forts de cela, nous pensons que le plus important sera d'être innovant sur la forme et les mouvements des obstacles.

3.6 Shop/Comptes (Jean-Baptiste, Alexandre)

Alexandre ayant déjà réalisé un système de boutique dans un précédent projet, nous ne partions pas de zéro pour en créer un pour Colors. Nous allons donc utiliser un fichier Google Sheet comme base de données regroupant les informations des utilisateurs. Grâce à l'afit du premier semestre, nous avons décidé d'encoder ces informations avec le système RSA afin d'assurer un niveau de sécurité assez conséquent. Au cours de nos recherches, nous avons découvert qu'une classe «RSACryptoServiceProvider » était déjà implémentée en C#, nous n'aurons pas comme dans l'afit, à écrire beaucoup de fonctions préalables. Pour ce qui est de la communication entre le jeu, et le Google Sheet, nous allons utiliser une API (Application Programming Interface). Cette dernière permettra à nos deux applications de se comprendre et de pouvoir s'échanger leurs données, dans un sens comme dans l'autre.

Nous avons trouvé une documentation à ce sujet :
https://developers.google.com/sheets/api/quickstart/dotnet
On y trouve un exemple d'utilisation en C# de la library `Google.Apis.Sheets.v4.` Celle-ci inclut de nombreuses méthodes difficiles à aborder et à comprendre. Il est long de rechercher l'utilisation et le but de chacune d'entre elles indépendamment. La documentation nous fournit donc un code complet avec un document Google Sheet pour essayer. Mais après avoir vu cela, nous avons beaucoup de mal à personnaliser ce code afin de l'utiliser sur un document que nous avons créé. Nous butons toujours sur la position des éléments de notre Sheet car nous nous retrouvons toujours avec des erreurs de placement dans le document.

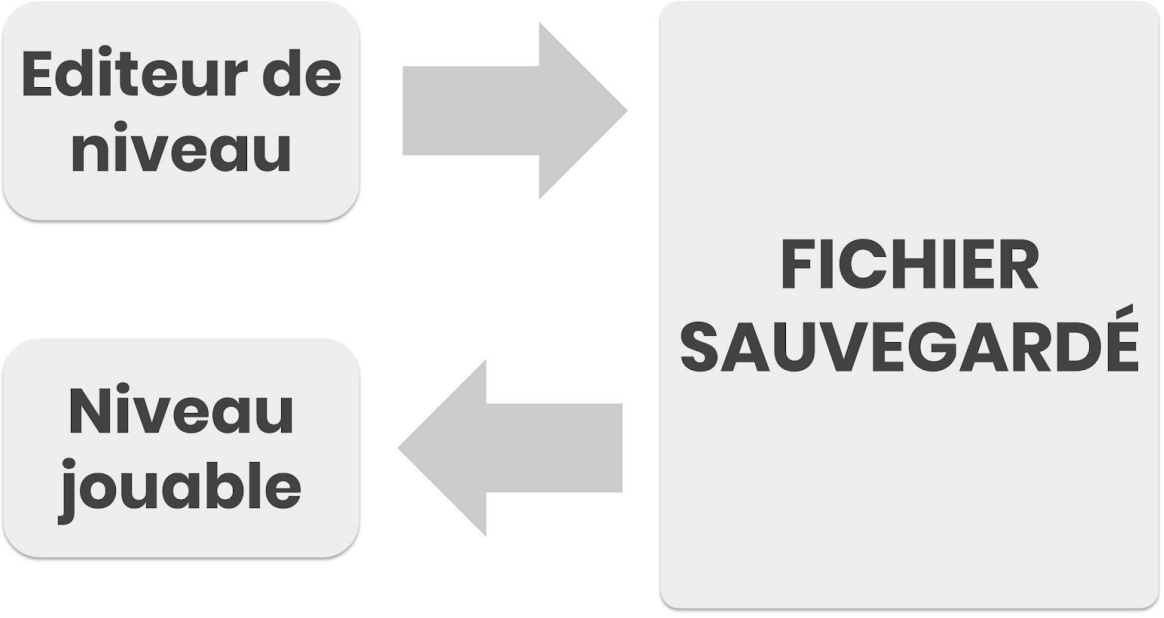
Ce système de base de données est plus simple à implémenter en Python grâce à la librairie Gspread. En effet, certains aspects sont communs aux deux langages mais l'utilisation de l'API était beaucoup plus directe en Python. Comme montré ci-dessous, l'API prenait moins de dix lignes à coder, tandis qu'en C#, il faut réaliser pas à pas l'ouverture, la vérification des accès (tokens) avant de pouvoir essayer d'accéder aux informations du fichier.



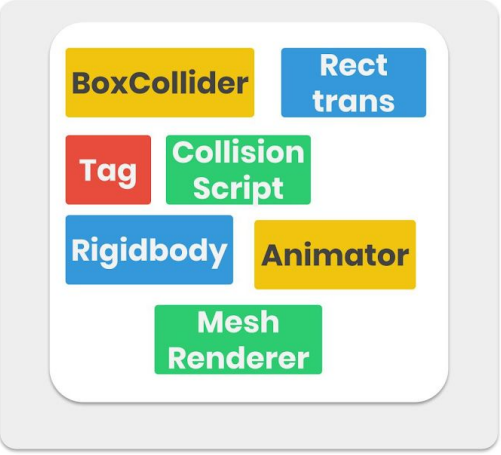
Code et utilisation en Python

3.7 Editeur de niveaux (Alexandre et Romain)

Nous n'avons que brièvement pensé à cette partie puisque c'est l'une des dernières tâches à effectuer. Dès le départ nous avons pensé nous inspirer du TP Christmas Lottery qui demandait de sauvegarder les paramètres des objets dans un fichier pour les recréer une fois le programme lancé.



Nous devons donc stocker dans le fichier tous les ennemis. Mais qu'est-ce qui définit un ennemi ? Voici un schéma du Save File auquel nous avons pensé.



Nous devons maintenant chercher comment créer un objet grâce à un script. Nous pensons s'inspirer de Mario Maker pour les interfaces. Pour la prochaine fois nous allons essayer de créer un obstacle depuis un script et de le sauvegarder de la même façon.

3.8 Scènes UNITY (Alex et Romain)

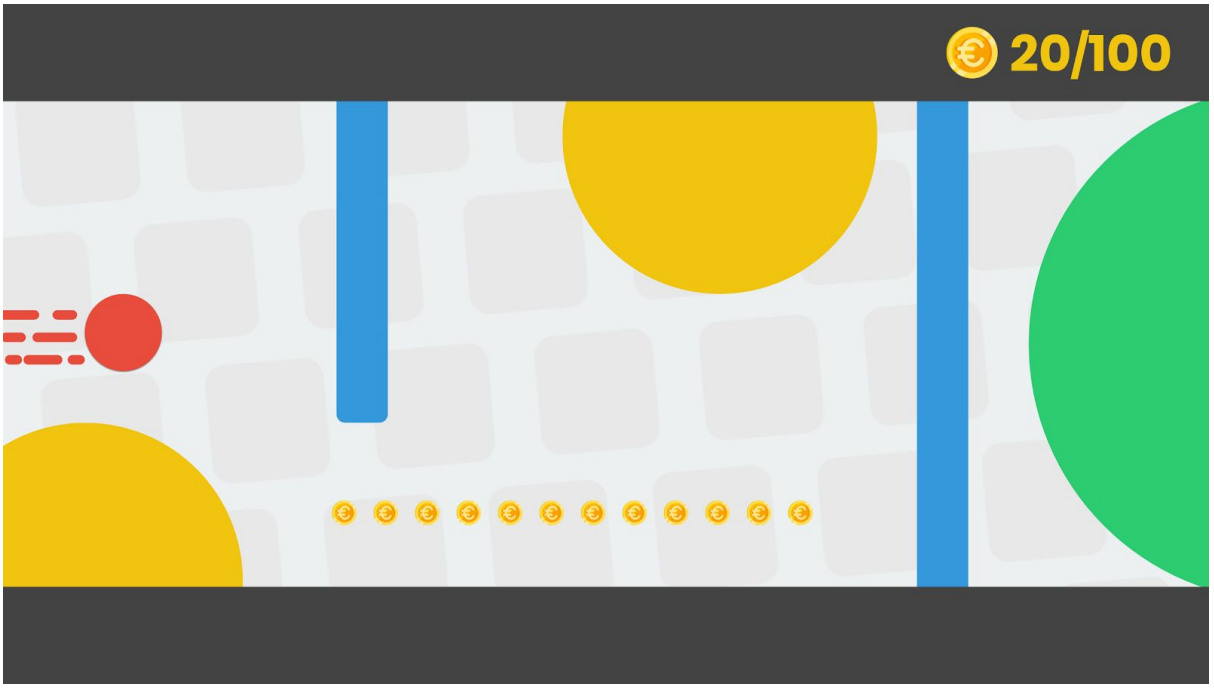
Depuis la remise du cahier des charges, avec Romain nous avons posé nos premières idées sur l'aspect du niveau. Notre jeu étant entièrement en 2D, nous voulons que la caméra suive la position et donc le mouvement du joueur uniquement sur l'axe horizontal. Nous avons fait des recherches de code pour que la position de la caméra change à chaque actualisation et donc à chaque fois que le joueur se déplace au travers du niveau.

Il faut donc créer un script dans la caméra principale (Main Camera) où il faut déclarer une variable **V1** de type **GameObject** et une seconde variable **V2** de type **Vector3** stockant respectivement le GameObject du joueur et la distance entre la caméra et la position du joueur.

Dans la méthode **Start** il faut donc initialiser notre deuxième variable de type Vector3: **V2 = transform.position-joueur.transform.position;**

Dans la méthode **Update** on actualise donc à chaque frame la position de la caméra avec la simple ligne de code suivante : **transform.position = joueur.transform.position + V1;** transform.position représente la position de la Main Camera.

Ensuite pour la prochaine soutenance, nous voudrions commencer à implémenter des pièces dans les niveaux du jeu. Le joueur peut ensuite, grâce aux pièces récoltées, aller débloquer des nouveaux personnages dans la boutique . Pour implémenter ces pièces dans les niveaux, nous avons pensé dans un premier temps à les animer pour qu'elles soient en mouvement continu. Il faudra donc coder tout un script quand le joueur rencontre cette chaîne de pièce, où les pièces disparaissent une après les autres. Il faudra enfin coder un compteur de pièces qui apparaîtra en haut à gauche de l'écran. Ainsi, on obtient le squelette d'un niveau du jeu suivant :



4. Conclusion

A l'issue de cette première échéance de projet, nous avons compris que nous nous étions engagés dans une tâche complexe. Les débuts ont été assez rudes car ils amenaient à la découverte de nouvelles notions auxquelles il fallait s'habituer rapidement. Aucun de nous n'avait jamais ouvert Unity avant ce projet, il a fallu par conséquent un temps d'adaptation pour comprendre le logiciel et ses us et coutumes.

Nous avons réalisé également que chaque détail compte et qu'il y a beaucoup de petits aspects qui nous semblent automatiques lors de l'utilisation d'un site internet par exemple qui sont en réalité assez complexes à créer.

Mais cette sensation de création de A à Z nous procure une fierté conséquente à chaque nouvelle avancée du projet.

Nous avons également compris que l'organisation est un élément primordial à la réalisation d'un projet, si cette dernière n'est pas présente dès le départ de chaque réalisation, il est très facile de se perdre et de rendre notre travail très brouillon. Elle nous permet donc de commencer sur des bases solides qui ne nous pénaliseront pas dans la suite du projet si des changements doivent être opérés sur des aspects réalisés au début de notre travail.

Pour le moment, nous sommes conscients que nous avons encore beaucoup de chemin à parcourir pour mener à bien notre projet. Mais une sorte de fierté se dégage d'ores et déjà de nos avancées dans nos domaines respectifs. Nous sommes motivés à continuer notre travail afin de produire un résultat final à la hauteur de nos attentes.