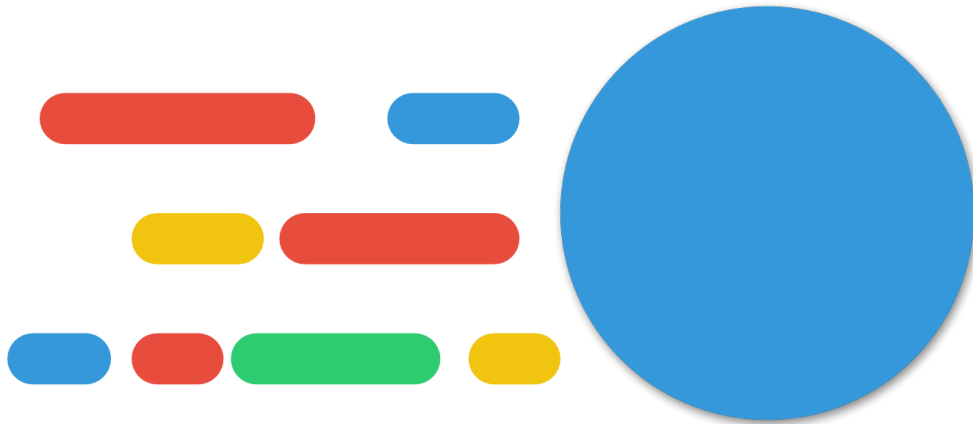


Rapport de Projet



Colors

Rémi VERNAY

P1D

Alexandre BERMUDEZ

Jean-Baptiste DESPUJOL

Romain GREGOIRE

Alex POIRON

Sommaire

1.	Introduction	.4
2.	Reprise du Cahier des Charges.	.5
2.1	Points importants	.5
2.1.1	Origine et nature du projet	5
2.1.2	Concept de début de projet	.5
2.1.3	Buts et intérêts généraux	.6
2.2	Mises à jours	.6
3.	Présentation du Projet	.7
3.1	Graphismes/Design	.7
3.1.1	Présentation	7
3.1.2	Application au projet	9
3.1.2.1	Couleurs	9
3.1.2.2	Typographie	.10
3.1.2.3	Icônes	11
3.1.2.4	Formes	.12
3.1.2.5	Influences	.12
3.2	Site Internet	12
3.2.1	Objectif	.12
3.2.2	Recherches préalables	.13
3.2.3	Création du site	.13
3.3	Personnage	.15
3.3.1	Présentation et explications	.15
3.3.2	Partie technique	.16
3.4	Obstacles	.18
3.4.1	Présentation	.18
3.4.2	Implémentation	.21
3.5	Boutique et Comptes	24
3.5.1	Idées d'implémentations	.24
3.5.2	Recherches préalables	25
3.5.3	Mise en place du système de base de données	25
3.5.4	Apparences	.30
3.6	Interface	.31
3.6.1	Présentation	.31

3.6.2	Implémentation Unity	32
3.6.2.1	Menu principal	32
3.6.2.2	Menu des options	33
3.6.2.3	Menu de connexion	33
3.6.2.4	Menu de création de compte	34
3.6.2.5	Menu de sélection de niveaux et de magasin	34
3.6.3	Liaison à la base de données	35
3.7	Niveaux	37
3.7.1	Assemblage	37
3.7.2	Extension	39
4.	Récit de réalisations	40
4.1	Récit général de la réalisation	40
4.2	Aspect musical du jeu	43
4.3	Ressentis personnels	46
4.3.1	Alexandre Bermudez	46
4.3.2	Jean-Baptiste Despujol	46
4.3.3	Romain Gregoire	47
4.3.4	Alex Poiron	48
5.	Conclusion	49
6.	Annexes	51

1. Introduction

Tout au long de ce rapport de Projet, nous allons suivre un plan thématique où toutes les tâches réalisées sur notre jeu seront présentées, détaillées et expliquées. Nous avons décidé de choisir un plan tel que celui-ci car nous pensons que c'est le meilleur moyen d'expliquer l'ensemble des choses qui ont été réalisées tout en restant le plus claire possible.

Avant de rentrer dans le vif du sujet, nous aborderons dans une première partie la reprise de notre Cahier des Charges remis en Janvier 2019. Cela permettra de rappeler nos intérêts, nos objectifs ainsi que la genèse de notre groupe et projet.

Nous avons donc pensé logiquement à introduire notre sujet pour que la lecture se fasse de la manière la plus fluide possible. Depuis la date de la remise du cahier des charges jusqu'à la dernière soutenance, nous avons modifié des objectifs données dans le cahier des charges. Cette partie sert donc à exprimer nos mises à jour et leurs buts.

Ainsi, dans la partie qui suit, nous arriverons à la présentation globale du projet. Nous avons découpé ici le projet en plusieurs tâches qui ont été rédigées par tous les membres du groupe. On retrouve à droite de chaque titre de partie les noms du responsable et de son suppléant.

Nous détaillerons dans ces parties les avancées réalisées ainsi que les raisons derrière les choix de conception.

Nous avons préféré ce type de plan plutôt qu'un récit chronologique car il est plus simple à suivre car chaque partie est facilement identifiable.

Enfin nous arrivons à la dernière partie qui est celle du récit de la réalisation. Cette partie va nous permettre d'expliquer vraiment nos ressentis sur le projet. Nous expliquerons chacun à notre tour comment nous avons vécu ce projet et quelles sont les conclusions et les nouvelles connaissances que nous en avons tiré. Nous détaillerons également la manière dont s'est faite la création de notre groupe de projet.

2. Reprise du Cahier des Charges

2.1 Points importants

Lors de notre remise de notre Cahier des Charges, nous y avons laissé nos premières idées et par conséquent nos attentes sur notre futur projet. Ainsi dans cette première sous partie, nous résumerons les points importants cités dans celui-ci.

2.1.1 Origine et nature du projet

L'idée fondamentale de faire un jeu pour la réalisation de ce projet est venue de chaque membre de ce groupe. Par la suite, nous étions tous en accord pour créer un jeu très épuré, et ne pas partir dans un projet trop complexe de manière à étoffer les mécaniques et la conception générale du jeu tout au long du développement.

Notre projet est donc un jeu où l'on contrôle un cercle qui peut changer de couleurs via les touches d'une manette ou d'un clavier. C'est donc cette mécanique qui sera au cœur de la jouabilité de notre jeu. Cette mécanique permettra au joueur d'éviter les différents obstacles qui se trouveront en travers de son chemin. Le joueur sera confronté à différentes formes d'ennemis tels que des cercles, des barres et d'autres formes géométriques de diverses couleurs bloquant son avancée.

2.1.2 Concept de début de projet

Le jeu s'inscrit dans le genre Runner. En effet, le fond du niveau se déplace automatiquement par rapport au joueur. Mais ce dernier aura également la possibilité de se déplacer horizontalement en addition aux mouvements verticaux. Cela a pour but de donner plus de liberté et de profondeur au gameplay.

On retrouve 2 principes fondamentaux dans notre projet : un système de changement de couleur couplé à un système de niveau dynamique avec des obstacles. Le principe essentiel pour que notre personnage survive est qu'il doit être de la même couleur que l'obstacle qu'il traverse, sinon le joueur recommence le niveau. Le personnage sera forcé d'avancer à un rythme soutenu.

On prévoit un mode Infini où le joueur pourra s'entraîner sur les différents types d'obstacles et réaliser le meilleur score possible. Cette implémentation ajoute beaucoup à la rejouabilité.

Le personnage représentant le joueur sera un cercle de couleur ne possédant qu'une seule vie. Ainsi, si ce dernier n'évite pas un obstacle, il sera forcé de recommencer le niveau depuis le départ. Le personnage sera différenciable du fond grâce à une légère ombre. Malgré les apparences de personnage que nous intégrerons, la forme générale du personnage ne variera pas afin de conserver un équilibre entre tous les joueurs.

Les obstacles forceront le joueur à changer de couleur pour pouvoir passer au travers de ces derniers. Ils apparaîtront sous de multiples formes comme par exemple un cercle s'agrandissant pour remplacer le fond actuel, des barres de couleur bloquant le passage du joueur.

2.1.3 Buts et intérêts généraux

Le but de ce projet était de créer un jeu vidéo facile à comprendre, facile à jouer et aux graphismes épurés. Nous souhaitons partir d'une base de jeu simple et tout au long du développement du jeu, améliorer et perfectionner chaque aspect de ce dernier.

De plus, nous étions sûrs d'apprendre à travailler en équipe, à répartir les tâches, et à apprendre à se remettre en question. Nous étions également convaincu que cette tâche allait nous permettre de nous rendre compte de la difficulté que représente la réalisation d'un projet tel que celui-ci et enfin de savoir si cette activité nous plaît réellement. Nous mesurons d'ores et déjà la chance que nous avons d'avoir beaucoup de matériel disponible gratuitement (salle informatique complète, compte Rider pour le C# et plus encore).

2.2 Mises à jour

Lors de l'élaboration du Cahier des Charges, nous ne savions pas réellement dans quel travail nous allions nous embarquer. C'est pourquoi dans ce dernier, nous avons évoqué la mise en place d'un éditeur de niveaux dans notre jeu. Nous avons très clairement sous-estimé le temps que cette tâche allait nous prendre et nous avons dû abandonner l'idée pendant la création du projet. De plus nous avons parlé d'un mode de jeu où le niveau serait infini et le score du joueur dépendrait uniquement de la patience et du niveau de jeu du joueur. Cette option dans notre projet a également été abandonnée.

Au niveau de la jouabilité, nous avons décidé de ne pas laisser la possibilité au joueur de se déplacer horizontalement indépendamment de la caméra. Nous avons fait ce choix afin de contraindre ce dernier à réellement user de la mécanique de changement de couleurs plus que de l'évitement pur et dur.

A l'inverse nous n'avions pas cité ni dans le cahier des charges ni dans les précédents rapports de soutenance l'implémentation de musiques ainsi que de bruitages dans le jeu. En effet, nous avons décidé d'implémenter ces détails pour rendre notre jeu plus complet et plus professionnel.

3. Présentation du Projet

3.1 Graphisme & Design (Alexandre, Alex)

Pour la partie du Graphisme & Design ce sont Alexandre et Alex qui sont respectivement responsable et suppléant et qui parleront dans les lignes suivantes de leurs avancées sur cette partie.

3.1.1 Présentation

Notre premier objectif était de créer entièrement le design du jeu et de n'utiliser aucun graphisme déjà présent sur Internet. Cela nous permet d'être plus créatifs et d'être réellement plus libres. Néanmoins nous nous sommes inspirés d'un style de graphisme épuré, minimaliste qu'est le **Flat Design**. Nous respectons aussi au plus possible les lignes de guidage du **Material Design** pour créer notre design et l'identité de notre jeu.

Ainsi, tous les graphismes créés suivent les adjectifs précédemment cités, que ce soit pour tous les menus, les personnages, les niveaux et ainsi de suite. De plus, une de nos missions est de créer de la fluidité et de garder dans la cohérence notamment dans les couleurs mais aussi dans la navigation des menus et ainsi dans l'intégralité du projet.

En parlant des **couleurs** il y a 3 principes à respecter :

-La hiérarchie : Il faut que les couleurs représente une hiérarchie d'importance entre les éléments. Elles doivent également montrer si l'objet est interactif ou non. Les éléments les plus importants sont mis en valeur par leur couleur.

-L'association : Pour ne pas aller à l'encontre du premier principe on ne peut pas associer n'importe quelles couleurs ensemble. Par exemple sur un fond clair, il est vraiment déconseillé d'y disposer un objet clair également. Cela entraînerait des problèmes de lisibilité.

-L'harmonie : Il se doit que la couleur choisie reste la même dans l'entièreté du projet. C'est pour cela que l'utilisation d'une palette est fortement conseillé.

Les couleurs choisies ne sont pas des couleurs vives mais des couleurs coupées avec du blanc, cela donne un rendu plus lisse et plaisant à l'œil.

Nous allons ensuite lister les éléments autres que la couleur indispensables à un rendu **Flat Design**.

Pour la **typographie** : La police Roboto est la police par défaut pour le Flat Design. Elle est moderne et a été adoptée par un grand nombre de personnes. La police doit paraître sobre et simpliste.

Les titres doivent être mis en évidence grâce à leur couleur mais aussi leur épaisseur d'écriture. Les formes ont un rôle essentiel dans les interfaces et dans notre jeu.

L'arrangement des objets dans une interface est très importante. En effet l'emplacement des boutons doivent être **prédictible**. Cela doit être intuitif pour l'utilisateur, il ne doit pas avoir à chercher comment accéder à un objet.

Cet arrangement doit être **ordonné**, pour cela l'utilisation de grilles et de repères est fortement conseillé. **La fluidité** est un point important pour une interface, elle doit répondre rapidement à l'entrée utilisateur. Nous en parlerons plus en détails dans la partie interface.

La présence d'icônes est essentielle et permet d'illustrer les interfaces. Les icônes doivent respecter les lignes du Flat Design, pour cela des banques d'images existent et contiennent des milliers d'images adéquates.

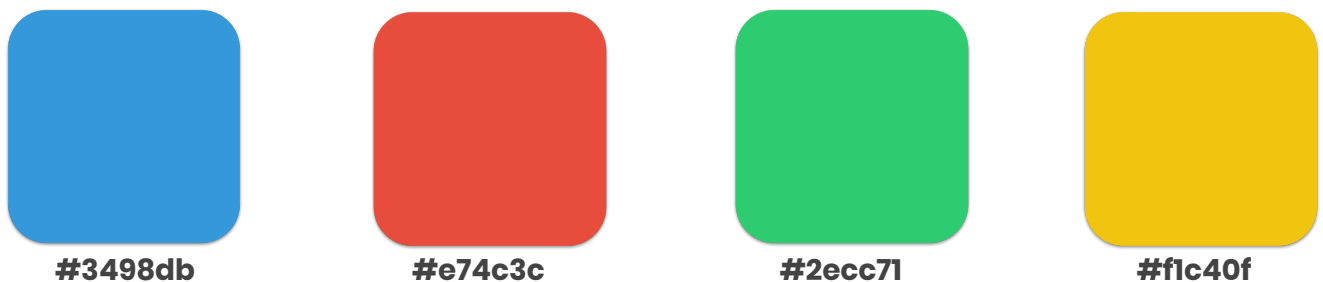
Il y a 2 types de **formes** principales, des cercles et les carrés. Pour ces derniers il faut différencier ceux avec des bords carrés et d'autre avec des bords arrondis. Cette dernière différences est utile pour la hiérarchie des objets.

3.1.2 Application au projet

3.1.2.1 Couleurs

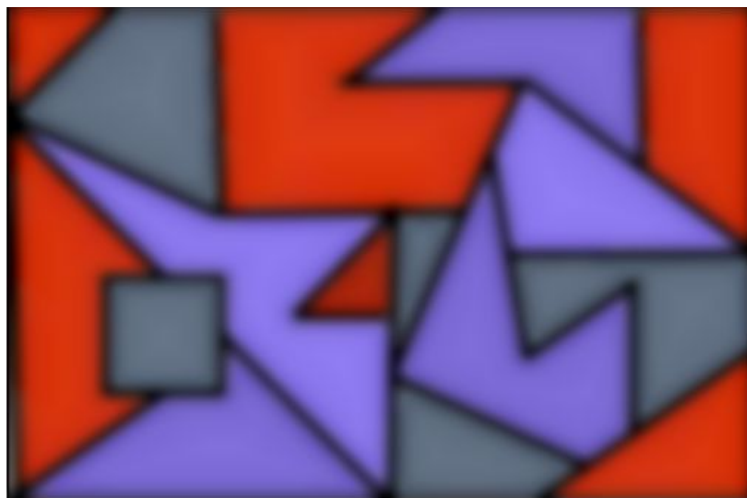
Pour cela nous procédons à l'utilisation d'une palette de couleur qui est très importante pour conserver une cohérence générale, c'est l'un des facteurs les plus importants qui définit un bon design. Nous avons pris les couleurs du site : flatuicolors.com de la Palette VI.

Voici les couleurs principales sélectionnés ainsi que leurs codes HEX :



Comme nous pouvons le voir, ces couleurs vont bien ensemble, c'est le principe d'harmonie. Elles ont toutes cet effet "adouci", en effet, elle n'agressent en aucun cas l'œil.

Pour le fond des niveaux, nous avons en premier essayé de faire un fond composé de formes géométriques colorés :



Nous avons remarqué qu'avec ce fond le principe d'association n'est pas respecté c'est pour cela que nous avons renoncé à un fond coloré.

De ce fait nous avons choisi un fond blanc avec de petits carrés gris :



De plus, dans tous nos menus créés, si le joueur sélectionne un bouton sur lequel aller, alors sa couleur de sélection dudit bouton change et augmente d'une teinte de gris.

C'est ce genre de petits détails que quasiment personne ne remarque alors qu'ils sont présents dans la plupart des jeux et applications. Ce sont ces détails qui permettent une expérience utilisateur fluide et agréable.

Cette teinte de gris est donc présente dans tous nos menus que ce soit le principal, celui de sélection de niveaux ou même celui de la boutique.

Ce détail a donc toute son importance dans la boutique. Elle permet au joueur, lorsqu'il a débloqué une apparence, de pouvoir savoir si elle a été équipée ou non.

Il est vrai que sur la plupart de nos graphiques le gris et le blanc sont les couleurs principales. C'est grâce à cela que nous pouvons donc jouer sur les teintes pour ce qui est des objets ainsi que le personnage principal.

3.1.2.2 Typographie

Nous avons décidé de ne pas utiliser le Roboto car cette police est utilisée de partout dans le monde du Flat Design. Or nous voulons nous créer une identité, de ce fait nous avons sélectionné la police **Poppins** (qui est actuellement utilisé pour l'écriture de ce rapport).

Nous utilisons principalement la version grasse (Bold) de cette police pour pratiquement toute les zones de texte. Nous avons eu un soucis lors de la première implémentation de la police dans notre jeu : tous les textes étaient flous et pixelisés.

Une recherche simple sur internet a réglé le problème, en effet nous avons appris qu'il y a 2 tailles de texte différentes. Il y en a une générale pour l'intégralité de la scène, et une qui est définie pour chaque zone de texte. Or la taille générale était trop petite donc quand on augmentait la taille des zones de texte, il y avait un effet de zoom. Nous avons donc mis une taille de 250 pour la générale. Grâce à cela, nous avons plus qu'une seule variable pour la taille de police.

La qualité du rendu final des zones de texte fut une grande préoccupation pour ne faire paraître notre jeux comme non fini. Nous avons maintenant besoin d'icônes pour illustrer nos interfaces.

3.1.2.3 Icônes

Pour nos icônes, nous avons utilisé généralement le site flatuicolors.com. Lorsqu'il nous manquait une icône très spécifique, nous utilisions des logiciels de traitement d'image tels qu'Adobe Illustrator et Photoshop. Avec ces deux logiciels nous avons pu créer des images de type Flat Design avec leurs outils de forme et ombre. Le choix des icônes était très important car il nous fallait des images qui représente le mieux un bouton menant vers un autre menu par exemple. Pour le magasin nous avons choisi un chariot de supermarché et pour les réglage un engrenage. Ces deux images illustrent bien ces deux idées et sont donc très intuitives pour le joueur.



Une fois le choix de l'icône effectué nous devions télécharger l'image source au format SVG. C'est la forme vectorisée de l'image. Avec ce format il n'y a aucune perte de qualité lors d'un zoom sur l'image. Cela est rendu possible car elle est enregistrée en tant qu'ensemble de formes et non comme pixels. Une fois la taille de l'image décidée, nous exportons l'image rognée au format PNG pour pouvoir l'intégrer à UNITY avec de la transparence. Comme le montre l'image ci-dessus et pour respecter le choix des couleurs nous avons choisi de colorer les icônes en gris foncé.

3.1.2.4 Formes

Les formes constituent la majorité des interfaces, elle sont donc très importantes. Utiliser principalement des carrés à bords arrondis nous a donc semblé être la meilleure option pour adoucir la vue de ces icônes.

De ce fait cet aspect arrondi est rappelé partout dans les interfaces. Il permet “d’adoucir” les formes pour conserver une certaine harmonie.

Nos boutons ressemblent donc à ceci :



3.1.2.5 Influences

Lors de l’élaboration de nos graphismes, nous avons forcément suivis nos influences, ce que l’on connaît, ce que l’on apprécie voir. Nous nous sommes fortement inspirés du design général de Google ainsi que celui d’Apple. Ces deux grandes multinationales spécialisées dans l’informatique n’utilisent pas ce type de design par hasard. C’est par sa facilité de compréhension par le plus grand nombre que nous avons choisi cette direction artistique.

Vous trouverez en **Annexes** tous les menus extraits directement de Unity

3.2 Site Internet (Jean-Baptiste, Alexandre)

3.2.1 Objectifs de début de projet

Le site internet, dès le début, nous est apparu comme une manière de développer l’univers visuel de notre jeu. C’est pour cela que nous voulions tout mettre en oeuvre pour installer une cohérence entre les menus du jeu et l’esthétique du site internet. Cela impliquait une présentation épurée avec seulement les informations nécessaires. Pour autant, nous ne voulions pas que le site paraisse vide. Notre défi était donc de découvrir pas à pas l’utilisation des langages permettant la création d’un tel site pour pleinement satisfaire nos attentes.

Cela nous motivait vraiment puisque nous utilisons tous les jours ces plateformes d'informations. Nous allions pouvoir découvrir des nombreux détails qui ne sautent pas du tout aux yeux lorsque l'on est simplement du côté utilisateur d'un site internet.

3.2.2 Recherches préalables

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 quelques 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.

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.

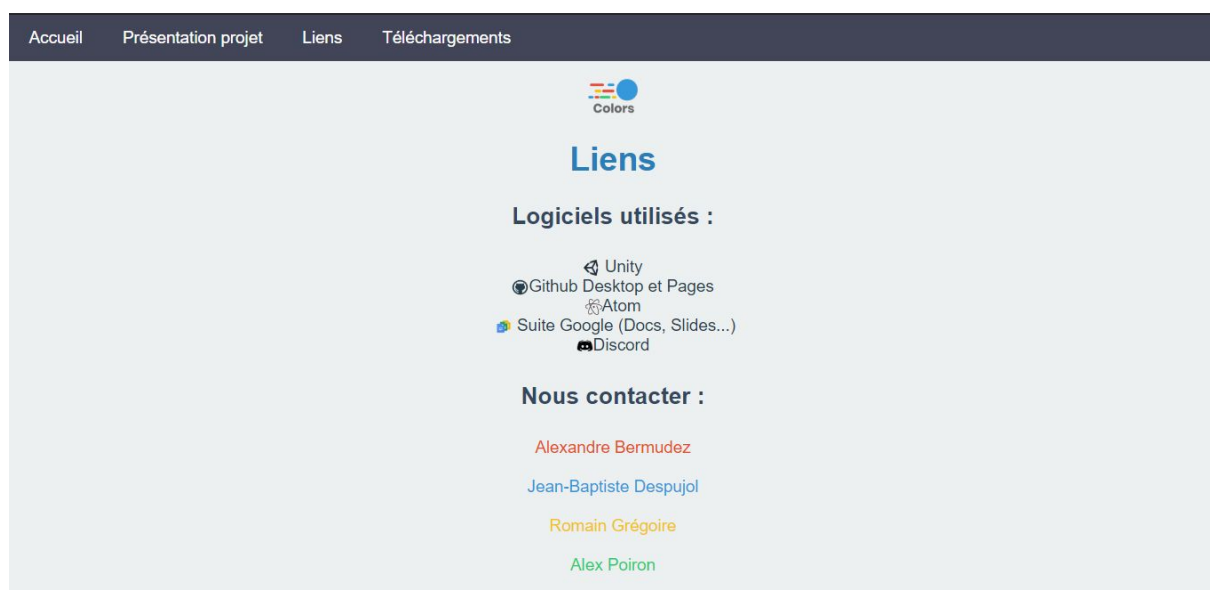
Pour ce qui est du code pur, nous nous sommes rendus compte que la compréhension de la notion de classe (même si elle est moins poussée qu'en C#) était primordiale car elle revient souvent dans ce type de développement. Sans cette notion, impossible de choisir exactement quelle zone de texte mettre en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

3.2.3 Création du site

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 présentées dans la partie graphisme 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 ont pris beaucoup de temps. Nous voulions, comme dit précédemment, produire un site propre et épuré. C'est pour cette raison que le site manquait encore de contenu. Malgré cela, le résultat visuel nous satisfaisait déjà avant la première soutenance.

Tout d'abord dans l'onglet Liens, nous avons ajouté les logiciels qui nous ont permis de réaliser le projet. Leurs différents noms sont des liens cliquables menant à leur site internet respectif. Nous avons retiré la décoration du texte afin qu'ils ne ressemblent pas à des liens classiques. Sans cette manipulation, le texte se serait coloré en violet une fois le lien visité. Nous avons organisé ces différents liens sous forme d'une liste verticale. Nous avons centré tous les éléments afin de coller à la pureté recherchée.



Onglet des liens

Nous avons rajouté les logos des différents logiciels. Il nous a fallu trouver des images au format PNG afin de ne pas avoir de fond contrastant avec la couleur du fond du site. Les logos sont donc très bien intégrés.

Pour ce qui est de la partie "Nous contacter" nous avons simplement lié le texte avec une méthode "mailto". Cette dernière permet d'ouvrir le logiciel d'email par défaut de l'ordinateur. Un nouveau mail sera créé avec le champ du destinataire déjà rempli par l'adresse d'un des membres du groupe.

Pour ce qui est de l'onglet présentation nous avons mis les membres du groupe et leur photo. Nous avons d'abord créé ces vignettes à l'aide de Google Slide, que nous avons ensuite exportées ces photos au format PNG encore une fois pour qu'il n'y ait pas d'information ou de fond superflus. Nous nous sommes

également servis de descriptions d'images afin que le nom de chaque membre soit aligné avec la photo correspondante.

Ensuite nous avons ajouté 4 paragraphes détaillant un problème majeur que chaque membre a rencontré et comment nous l'avons réglé. Il nous permet de nous rendre compte des grandes difficultés auxquelles nous avons fait face.

Finalement nous avons créé un historique de réalisation qui permet de suivre nos avancements majeurs dans le projet.

Pour finir, pour ce qui est du dernier onglet, nous avons stockés les documents téléchargeables sur le Git de notre groupe. Nous avons fait en sorte que lorsqu'un utilisateur clique sur les liens, un nouvel onglet s'ouvre dans le navigateur pour que le site et le document soient ouverts en même temps.

Les derniers liens n'étaient pas encore disponibles et sont donc grisés avant la dernière soutenance. Ils renvoient vers la page de téléchargement. Pour l'utilisateur, cela aura seulement l'air d'un lien qui ne mène nulle-part.

Une fois, tous les documents finalisés, nous les ajoutons dans l'onglet Téléchargement.

Toutes autres pages du site non montrées dans le rapport sont présentes en **Annexes**

3.3 Personnage (Romain, Jean-Baptiste)

3.3.1 Présentation et explications

Nous allons maintenant commencer à parler de la partie jeu pur en parlant de l'élaboration du personnage. Pour **Colors** nous voulions que notre personnage soit le plus épuré et agréable à contrôler possible. Pour cela nous nous sommes rapidement mis d'accord sur comment nous allions le représenter, nous allions le matérialiser par un cercle qui se déplace automatiquement de manière horizontale. Le joueur peut seulement contrôler les déplacements verticaux.

A ces idées venaient se greffer la mécanique centrale de notre jeu le changement de couleur. L'objectif étant de forcer le joueur à modifier sa couleur pour passer au travers des différents obstacles présents sur sa route.

Cela étant il nous est apparus assez rapidement que les déplacements verticaux étaient absolument incompatibles avec le gameplay de notre jeu. En effet, comme il n'existait pas encore de pièces à ramasser pour déverrouiller des

apparences, il paraissait plus intuitif et rentable pour le joueur d'esquiver les obstacles plutôt que de les traverser.

Pour remédier à cela nous avons choisi de repenser complètement les déplacements de notre joueur. L'idée que nous avons retenu était de remplacer les déplacements verticaux par des sauts. Grâce à cela nous conservions une part de verticalité en enlevant les esquives évidentes mais tout en laissant une part de contrôle et de prise de décision pour le joueur.

A cela vient s'ajouter le fait que nous avons implémenté des pièces que le joueur peut ramasser afin de débloquent de nouvelles apparences. Nous pensions que cela inciterait le joueur à prendre des risques afin d'aller chercher les pièces présentes dans les obstacles. Par la même occasion nous avons ajouté le système de collision avec les obstacles, et après les tests il nous est effectivement apparu que le système de saut était tout à fait adéquat avec les niveaux.

Enfin pour la troisième et dernière soutenance nous n'avons ni ajouté ni changé d'élément majeur sur notre personnage. Nous n'avons implémenté que deux éléments cosmétiques à savoir :

- Les apparences avec une simple surcouche visuelle apposée au personnage.

- La traînée de couleur semblable à ce que vous pouvez voir dans le logo de notre projet.

Ensuite pour ce qui est de la traînée nous pensions que le côté graphique de la traînée apportait un plus sur le plan esthétique et rendait le jeu plus coloré encore et semblait être la continuité logique de notre projet.

3.3.2 Partie technique

Ceci étant dit, rentrons maintenant dans la partie technique de ce personnage. Comment bouge t-il ? Comment ses interactions sont rendues possibles ? Afin de répondre à ces questions nous allons vous expliquer les lignes de codes suivantes qui font parti du script implémenté au personnage.

```
void Update()
{
    transform.Translate(vitesse, 0,0);
    if (Input.GetKeyDown("joystick button 4"))
    {
        rb.velocity = Vector2.up * haut;
        audiosource.PlayOneShot(JumpSound);
    }
    if (Input.GetKeyDown("joystick button 1"))
    {
        currentcolor = "Red";
        sr.color = colorred;
        gameObject.tag = "Red";
    }
    if ((Input.GetKeyDown("joystick button 3")))
    {
        currentcolor = "Yellow";
        sr.color = coloryellow;
        gameObject.tag = "Yellow";
    }
    if (Input.GetKeyDown("joystick button 0"))
    {
        currentcolor = "Green";
        sr.color = colorgreen;
        gameObject.tag = "Green";
    }
    if (Input.GetKeyDown("joystick button 2"))
    {
        currentcolor = "Blue";
        sr.color = colorblue;
        gameObject.tag = "Blue";
    }
}
```

Code qui nous permet de changer les couleurs de notre personnage avec une manette.

Nous avons choisi de mettre des conditions pour chaque cas, c'est à dire pour chaque commande pour notre personnage. Expliquons-les :

- Le premier **if** prend en compte le cas où le joueur décide de sauter. Nous avons décidé que pour sauter, le joueur doit appuyer sur la gâchette gauche. Donc si ce bouton est appuyé alors la variable de saut défini est actualisée.
- Les quatres autres **if** définissent les quatre changements possibles de couleur. Nous avons implémenté ce changement de couleur par le biais des 4 boutons **A, B, X** et **Y** qui, sur une manette de Xbox One, sont représentés chacun par nos 4 différentes couleurs. Si l'un de ces quatre boutons est appuyé, alors on stocke sa couleur dans une variable appelée **currentcolor** et ensuite on change son étiquette (**tag** en anglais) que l'on utilise dans d'autres fonctions.

Vous trouverez en **Annexes** dans la partie **personnage**, toutes les autres informations sur les fonctions utilisées pour notre personnage ainsi que ses variables.

3.4 Obstacles (Alex, Romain)

3.4.1 Présentation

Cette partie a été réalisé par Romain ainsi qu' Alex respectivement en tant que responsable et suppléant de cette tâche. Cette dernière est la partie qui nous a demandé probablement le plus d'imagination.

Lorsque nous avons débuté la réalisation de ceux-ci nous n'avions jamais utilisé la plateforme UNITY, nous avons donc pour la première soutenance eu beaucoup de mal à nous préparer correctement sur le plan technique pur, on pourrait même aller jusqu'à dire que nous ne comprenions pas du tout les différents principes liés aux boîtes de collisions. Néanmoins cela ne nous a pas empêché de réaliser une base d'ennemis que nous pourrons utilisé dans les différents niveaux.

A l'origine nous n'avions en tête que deux idées d'obstacles majoritairement imprégnés du jeu *Color Switch* qui nous a grandement aidé dans la réalisation de notre projet:

Tout d'abord les obstacles ronds, lorsque nous les avons implémenté nous ne savions pas comment réaliser les différentes fragmentation de couleur. Nous avons donc commencé par les sectionner en quart chacun possédant son propre collider. Nous souhaitons également les animés de manière à ce qu'ils tournent

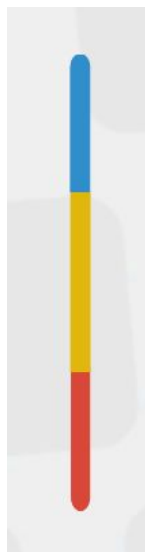
sur eux mêmes, néanmoins cela a représenté une difficulté particulière au niveau des rotations.



1er type d'obstacle en forme de cercle tournant sur lui même

En effet le script que nous avons utilisé présentait visiblement des failles puisque les objets ne tournaient pas sur eux mêmes sans se déplacer comme nous le souhaitions mais gravitait autour du même point . Et à notre grand désarroi nous n'avons toujours pas élucidé le problème qui "gravitait" autour de ce centre invisible. Malgré cela nous avons réussi dans la globalité des niveaux à disposer les obstacles de manières à ce que leur léger mouvement ne soit pas dérangerant pour le joueur.

Ensuite viennent les barres verticales .Pour cet obstacle nous ne savions à l'origine pas comment les réaliser puisque deux options s'offraient à nous. La première était de faire comme nous l'avons fait avec les obstacles circulaires et ainsi éditer nous même le script, et la seconde était d'utiliser l'**animator** pour réaliser les déplacements.



2e type d'obstacle : Barre avec un mouvement vertical.

De prime abord les deux méthodes semblent à peu près équivalentes en terme de réalisation puisque dans un cas nous avons déjà utilisé la méthode et dans l'autre cas l'**animator** est assez facile d'utilisation. Nous avons à l'origine choisi d'utiliser la première méthode, malheureusement il est apparu que les mouvements étaient beaucoup trop rapides et paraissaient saccadés, nous avons donc choisi rapidement de la réaliser à partir de l'animator.

Nos impressions se sont avérées exactes puisque les déplacements sont devenus plus fluides mais également plus faciles à régler pour les prochains niveaux.

Néanmoins nous nous sommes rendus compte assez rapidement que notre jeu ne pouvait contenir seulement deux obstacles.

Cela nous mène donc à la seconde soutenance. En amont de celle-ci, nous avons ajouté des idées de plus mais celles-ci restait sensiblement les mêmes, à savoir que nous voulions rajouter des carrées à la place des cercles. Ces derniers se comportent de la même manière que les cercles en tournant autour de leur centre. Mais nous n'étions pas satisfaits du rendu dans le jeu et nous avons donc préféré les retirer du jeu.

C'est donc avant la dernière soutenance que nous avons eu le plus d'idées. En effet nous avons rencontré un problème avec la rotation d'un de nos cercles. Nous nous sommes rendus compte du potentiel de la fonction transform.rotate. En effet appliquée séparément sur chacun des éléments composant nos différents sprites, nous pouvions réaliser un nombre beaucoup plus important d'obstacles ainsi que de variations dans les déplacements de ceux-ci.

Nous avons donc implémenté tout d'abord des croix pour le premier niveau, des fleurs pour le niveau 2 et enfin une nouvelle catégorie d'ennemis ceux-ci noir que nous avons longtemps hésité à implémenter du fait que nous pensions que cela aurait pu empiéter sur la mécanique de changement de couleur.

Les croix sont composées de deux couleurs tournant dans le sens horaire et anti-horaire. Les cercles, eux, ne tournaient que dans ce dernier sens. Avec les croix nous n'avons rencontré qu'un seul et unique problème quelque peu dérangent pour l'attention du joueur, la superposition des branches.

En effet, je le rappelle, tous nos ennemis sont intégrés dans le même plan z, et la superposition de deux éléments dans ce même plan entraîne forcément un effet de scintillement de clignotement du fait qu'UNITY ne sait pas quel élément placé en priorité sur l'autre.

Pour ce qui est de la réalisation d'un obstacle de type fleur, il s'agit plus d'un hasard heureux que d'une découverte volontaire. En effet, lorsque Romain cherchait une solution au problème de rotation du cercle, il a tenté d'appliquer le script de rotation à chacun des morceaux du cercle pour essayer d'enlever le problème de centre de rotation. Le résultat obtenu a été totalement différent de celui attendu. De ce fait, en appliquant le script chacun des quarts, ces derniers ont tourné autour de leur propre centre sans jamais se croiser donnant un mouvement de rotation harmonieux qui nous a donc fait penser à une fleur.

Enfin nous allons terminer la partie de présentation des différents obstacles en parlant des obstacles noirs. Il s'agit ici d'une catégorie très particulière d'obstacles, arrivé à ce stade du jeu il n'est plus question de passer à travers les objets mais de mettre en avant les réflexes du joueur par rapport à l'esquive. L'implémentation de cet obstacles résulte d'une longue interrogation à propos des mécaniques de jeu que celui-ci devait conserver.

Il est assez évident que notre jeu n'a en aucun cas un objectif compétitif, mais nous ne pensions pas que cela devait amputer une partie de l'habileté du joueur lié à un autre élément que le changement de couleur. De plus ces ennemis ont deux avantages majeurs dans la création et la gestion des niveaux. Tout d'abord, ils permettent de forcer le joueur à se déplacer vers un ennemi en particulier ou un objet spécial.

Le principe de l'obstacle est qu'il est défini avec une forme géométrique mais également par une couleur. On rappelle que si le joueur rentre en collision avec un obstacle d'une couleur différente que lui même, alors le jeu s'arrête et le niveau recommence du début. Nos obstacles sont en mouvement constant pour rendre l'ensemble plus dynamique. C'est ce dont nous allons parler dans Implementation.

3.4.2 Implémentation

Maintenant, entrons un peu plus dans la partie technique, celle de l'implémentation de nos obstacles dans nos niveaux. Nous allons donc parler de deux ajouts majeurs à la création de nos obstacles. Il faut savoir que lorsque nous créons un objet dans UNITY, dans le cas ici de symboles graphiques ("**Sprite**" en anglais), nous pouvons lui associer plusieurs caractéristiques :

La première: la rotation. Cela peut paraître simple quand on y pense mais faire une rotation dans un plan qui n'existe pas dans notre jeu présentait un certain nombre de problèmes. En effet, comme nous l'avons déjà abordé dans la partie précédente, nous avons remarqué que nos obstacles tournaient autour d'un seul et même point, comme si nous devions créer une orbite. A cause de cela nous avons quelque difficultés à créer de nombreux ennemis en rotation constante. Pour réussir à créer ces obstacles, il faut donc leur implémenter un script en C#, donc une fonction qui leur donnera cette caractéristique de tourner sur eux mêmes ou encore d'utiliser un **Animator**.

```
public float speed = 1f;
public Transform circle;

void Start()
{
    transform.Rotate(0f, 0f, speed*Time.deltaTime / 4);
}

void Update()
{
    transform.Rotate(0f, 0f, speed*Time.deltaTime / 4);
}
```

Fonction qui permet à certains obstacles de pouvoir tourner sur eux-mêmes

La seconde: les boîtes de collisions. Pour tous nos obstacles nous avons importé des boîtes de collisions ("**Box Collider 2D**" en anglais) qui nous ont énormément servis dans la suite du jeu. Dans Unity il existe 5 types de collider 2D différents qui peuvent être implémentés sur les ennemis, à savoir:

- les Circle collider 2D
- les Box collider 2D
- les Polygon collider 2D
- les Capsule collider 2D
- les Edge et Composite collider mais que nous n'aborderons pas dans ce rapport puisqu'ils ne nous ont pas été utiles dans la réalisation de notre projet.

Nous allons commencer par parler des circles collider. A l'origine lors de la prise en main d'UNITY, nous pensions pouvoir créer nos cercles de manière à n'avoir qu'un seul collider pour les ennemis. Avec les différentes couleurs renseignées à partir d'un script pour délimiter chaque morceau de l'obstacle. C'était sans compter sur le fait que le changement de couleur et le système de

récupération de couleurs de chaque ennemi allait être fait à partir de tag. Un système inhérent à Unity permet de marquer les objets, nous empêchant ainsi de n'utiliser qu'un seul collider.

Nous avons donc dû trouver une alternative à ceux-ci. Cela nous a donc mené au second type d'objet : les polygon collider2D. Comme nous l'avons expliqué précédemment nous n'avions pas pu utiliser le circle collider 2D pour nos obstacles ronds du fait des tags. Nous avons donc dû utiliser d'autres formes de collider 2D pour le remplacer. Il semble assez évident que les capsules ainsi que les box colliders sont assez mal choisis pour être utilisés avec des ennemis circulaires. Les edges et composite colliders étant utilisés respectivement pour des zones de contact (angles ouverts par exemple) et/ou des fusions. Ils n'étaient donc pas non plus appropriés pour nos quarts de cercle.

Il ne nous restait donc plus qu'un seul type de collider: les polygon collider 2D. L'avantage de ces colliders est qu'ils permettent de créer à peu près n'importe quel type de collider, d'autant plus que le système intrinsèque à UNITY est très efficace pour tracer lui-même ces colliders.

Ensuite pour la grande majorité, nos ennemis, que ce soit les croix ou encore les barres, sont composés soit de box collider soit de capsule collider. Contrairement à ce que nous avons pu avoir avec nos obstacles circulaires nous n'avons ici pas eu de problème puisque nous les avons déjà quasiment tous croisés au préalable. Nous avons donc simplement découpé notre sprite de barre par défaut en autant de membre que nécessaire de manière à disposer les différentes couleurs sur celui-ci.

Bien sûr les obstacles ne sont pas les seuls à avoir hérité du système de collider, les pièces elles aussi ont eu droit à leur part. Nous ne reviendrons pas sur le fonctionnement de celles-ci puisqu'il réutilise les propriétés des circle colliders déjà largement abordé au préalable dans le rapport. Une seule chose qu'il est important de noter est que nous avons réglé la taille des colliders de manière différentes en fonction du type d'objet en question.

En effet les colliders ne sont pas exactement superposés sur les objets. Nous avons fait le choix de réduire volontairement la taille des colliders des ennemis par rapport à leur sprite tout simplement pour que le niveau soit réalisable et au contraire nous avons fait le choix d'agrandir la taille des colliders des pièces pour qu'elles soient légèrement plus simples à ramasser.

Au vu de ce que nous venons de dire il apparaît donc que les colliders sont des éléments centraux de notre jeu. Dans tous les jeux de plateforme en général, la gestion des pièces reste très importante dans la conception des différents niveaux. On pourrait même aller jusqu'à dire que la réussite des jeux de

plateforme et du gameplay en lui même se fait sur les collisions et la gestion des “boîte de frappe”.

```
void OnTriggerEnter2D(Collider2D other)
{
    if (!gameObject.CompareTag(other.tag) &&
!other.CompareTag("MoneyCoin"))
    {
        UnityEngine.Debug.Log("Game Over");
        SceneManager.LoadScene(0);
    }
}
```

Fonction qui détecte si le joueur détecte un obstacle d'une couleur différente.

La troisième : les animations. Pour certains obstacles, il nous fallait faire varier les mouvements nous avons implémenté avec des animations. Sur UNITY, il faut ajouter à notre objet une caractéristique **Animator** et par la suite on réalise l'animation à la main via un système de clés d'images (“**KeyFrames**” en anglais). Ce système est également utilisé sur After Effects. Le but de ces animations est de varier les mouvements des obstacles et donc de ne pas avoir de sensation de répétitions pour le joueur.

3.5 Boutique et Comptes (Jean-Baptiste, Alexandre)

3.5.1 Idées d'implémentation

Pour cette partie, dès le départ du projet, nous nous sommes accordés sur le choix de créer un système de boutique et de comptes dans notre jeu. En effet, ces éléments augmente de manière non négligeable la rejouabilité. Le joueur après avoir pris connaissance du système voudra collectionner toutes les apparences (**skins** en anglais) à débloquent en récupérant les pièces disséminées au long des niveaux.

Pour ce qui est de la manière de stockage, nous avons pensé à garder les données utilisateurs (pseudonyme, mot de passe, nombre de pièces récoltées, skins achetés, niveaux débloqués) directement dans les fichiers du jeu. Après réflexion, il nous a semblé que cette manière était assez classique. Nous avons donc choisi d'aller plus loin en gardant toutes ces informations en ligne. Cela impliquait donc d'utiliser un système de base de données. Cette implémentation était pour nous la meilleure puisqu'elle permettait aux joueurs de créer par

exemple leur compte, acheter des apparences sur un premier appareil, et d'ensuite, sur un deuxième retrouver sans aucun soucis toutes leurs données.

3.5.2 Recherches préalables

Alexandre ayant déjà réalisé un système de boutique dans un précédent projet, nous ne partions pas de zéro pour en créer un pour **Colors** (c'est ce que nous pensions). Nous allions donc utiliser un fichier Google Sheet comme base de données regroupant les informations des utilisateurs. Grâce à l'afit du premier semestre, nous voulions 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'aurions 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 voulions utiliser une API (Application Programming Interface). Cette dernière aurait permis 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 butions toujours sur la position des éléments de notre Sheet car nous nous retrouvions 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.

3.5.3 Mise en place du système de base de données

Avant la première soutenance, et peu après la seconde, nous avons concentré nos recherches vers l'utilisation d'un Google Sheet comme support pour une base de données (database) pour les joueurs. Comme dit précédemment, nous avons compris que cette implémentation allait être assez complexe. Nous nous sommes tout de même obstinés puisqu' Alexandre avait déjà implémenté ce type de système. Il lui avait été très facile d'utiliser et de

comprendre le principe d'une bibliothèque existante dans son code de jeu en python.

Mais après avoir écumé la plupart des informations disponibles à ce sujet sur internet, et après avoir réalisé de nombreux tests infructueux, nous avons compris qu'il était préférable d'explorer une autre voie pour le stockage de nos données en ligne. Nous avons donc recherché d'autres moyens de réaliser une base de données pouvant être liée à notre projet UNITY.

Nous avons découvert le système de gestion de données MySQL. C'est un service libre, open source, qui permet de créer la base de données dont nous avons besoin. Nous avons ensuite trouvé un tutoriel vidéo très détaillé qui montrait comment mettre en place une base de données localement sur nos ordinateurs et comment la lier à notre jeu. Cette méthode utilisait des fichiers PHP. Ce nouveau langage, de prime abord, peu faire peur dans la mesure où nous ne l'avions jamais utilisé auparavant. Mais il est en réalité assez intuitif et les explications données dans le tutoriel sont assez claires.

Nous avons bien compris après quelques recherches que ce langage permet de réaliser des actions sur des serveurs web. Il nous a donc paru étrange d'utiliser un tel langage pour des informations stockées localement. Mais après un peu de réflexion, nous avons compris que notre stockage local se comportait comme un serveur, à la seule différence qu'il n'était pas connecté à internet.

Dans le processus, nous avons utilisé le logiciel Mamp. Ce dernier nous permet de stocker facilement la database sur nos appareils. Après avoir choisi le dossier de dépôt, le logiciel nous redirige vers une page internet MySQL. Sur cette dernière, nous pouvons créer facilement une base de données ainsi que les tables qui la constituent. Les tables sont très facilement configurables, on choisit le type de données à stocker dans chaque ligne, le nombre maximal de caractères ou même la valeur par défaut du paramètre.

Prenons comme exemple la ligne qui va contenir les noms d'utilisateurs de chacun des joueurs. Ici le nom de cette ligne est "*username*". Ce nom va nous permettre, lorsque nous l'utiliserons comme paramètre dans les fonctions constituant les fichiers, de requêtes PHP afin de localiser et d'accéder facilement aux noms pour vérifier si celui entré par l'utilisateur est valide.

Ensuite, il y a la colonne type, elle permet logiquement de renseigner si la ligne conservera des nombres ou des chaînes de caractères par exemple.

Juste à côté, il y a la colonne taille, elle permet de fixer une limite au nombre de caractères. Si nous ne voulons pas que les joueurs possèdent des pseudos de plus de 16 caractères nous remplirons en conséquence cette colonne par ce

nombre. Néanmoins, cela ne fonctionne pas exactement de la même manière pour les entiers. La valeur de la taille remplie est la puissance de dix. Si nous entrons par exemple 9 dans cette case correspondant au nombre de pièces, le joueur pourra amasser jusqu'à 1 million d'unités de monnaie du jeu.

Vient ensuite la colonne des valeurs par défaut. Cette colonne nous est seulement nécessaire pour le nombre de pièce puisque c'est la seule valeur qui doit être assignée à zéro lors de la création de chaque compte.**(Voir Annexe 1.5.1)**

Ensuite, nous avons commencé à écrire notre fichier PHP qui allait communiquer et modifier notre base de données pour créer de nouveaux utilisateurs du jeu.

La dernière colonne intéressante à expliquer est celle où l'on peut choisir l'unicité des valeurs contenues dans une ligne. En effet, nous en avons besoin pour les noms d'utilisateur. Chaque nom doit être unique pour qu'il n'y ait pas de conflit lors de la connexion de deux comptes portant le même pseudonyme. Si nous avions laissé la possibilité d'avoir plusieurs fois le même nom, nous aurions eu beaucoup plus de mal à vérifier quel était le compte qui voulait se connecter. Cette unicité nous facilite donc réellement le login de chaque joueur.**(Voir Annexe 1.5.2)**

Nous avons recopié ce fichier qui était présent dans le tutoriel vidéo. Mais nous avons compris l'intégralité des manipulations effectuées lors de son exécution. La première fonction "*mysqli_connect*" crée un lien avec la base de données. Le premier paramètre de cette fonction prend l'URL de l'hôte, le deuxième paramètre prend l'utilisateur de l'hôte, le troisième prend le mot de passe, et finalement le dernier prend le nom de la base de données. Comme à ce stade l'hôte est local, il n'y a pas d'URL classique dans les paramètres.

"*mysqli_connect_errno()*" renvoie "True" si la connexion à la base de données n'est pas possible. La fonction *echo* en PHP est l'équivalent d'un *print* que nous connaissons en C#. Chaque sortie comporte un chiffre de référence pour comprendre de quelle erreur il s'agit. Ces sorties renvoyées par le fichier PHP seront ensuite captées par le script de UNITY. Si aucune erreur n'a été rencontrée durant l'exécution, le fichier renvoie 0.

Ensuite, nous avons créé des variables "*username*" et "*password*" qui nous seront utiles pour la suite.

On initialise "*namecheckquery*" comme une requête pour récupérer les données du nom d'utilisateur dans la base. Nous utilisons ensuite "*namecheckquery*" dans la méthode "*mysqli_query*". Cette dernière permet simplement d'appliquer une requête à notre base de données. Si cette action n'est

pas réalisable, la méthode *“die”* est appelée, écrit un message et quitte l’exécution du fichier.

Une fois la requête précédente réussie, nous vérifions que le nom d’utilisateur donné par le joueur n’est pas déjà présent dans la base. C’est la méthode *“mysql_num_rows”* qui va nous aider à le faire car elle renvoie le nombre de lignes d’un élément spécifique (le nom d’utilisateur fourni par le joueur ici).

Si tout s’est bien passé jusqu’ici, nous allons pouvoir passer à l’ajout de nos données utilisateur à notre base.

Pour respecter un certain niveau de sécurité, nous avons utilisé le hachage. En effet, nous hachons le mot de passe de l’utilisateur lorsqu’il crée son compte.

Nous créons d’abord un grain de sel personnel afin de sécuriser l’encryption. Le grain de sel personnel est une petite donnée additionnelle qui renforce significativement la puissance du hachage. Nous en créons un afin éviter d’utiliser un grain de sel préfait facilement trouvable à l’aide de dictionnaires disponible sur internet. Cela compromettrait la sécurité des informations.

La méthode *“crypt”* applique notre grain de sel sur le mot de passe et renvoie sa version encryptée.

Nous pouvons maintenant ajouter le nom d’utilisateur, le grain de sel, et le mot de passe haché à notre base de données. La sécurité est vérifiée car même si une personne avait accès à la base de données, il serait impossible pour cette dernière de récupérer le mot de passe puisque le hachage est une encryption à sens unique.

Si toutes les étapes requises par le fichier PHP ont été réalisées, ce dernier renvoie le code 0. Cette information permettra au script Unity de savoir que l’utilisateur a été créé correctement.

Maintenant, pour ce qui est de Unity au niveau du test, nous avons créé une scène assez basique pour vérifier la bonne communication avec la base de données. **(Annexe 1.5.3)**

Pour les premiers tests, nous avons seulement créé de simples champs d’entrées d’utilisateurs. Ils nous ont permis de vérifier que la connexion entre la base de données et notre jeu fonctionnait correctement.

Pour cette scène d’enregistrement nous avons besoin d’un script pour paramétrer les boutons et les champs d’entrée.

```
IEnumerator Register()
{
    WWWForm form = new WWWForm();
    form.AddField("name", nameField.text);
    form.AddField("password", passwordField.text);
    WWW www = new WWW("http://localhost/sqlconnect/register.php", form);
    yield return www;
    if (www.text == "0")
    {
        Debug.Log("User created successfully.");
        UnityEngine.SceneManagement.SceneManager.LoadScene(0);
    }
    else
    {
        Debug.Log("User creation failed. Error #" + www.text);
    }
}
```

Script Unity pour la création de comptes

D'abord, nous créons un objet WWWform qui va contenir les entrées utilisateur dans les menus du jeu. Ensuite, une fois les données récupérées par le jeu, le script envoie une requête au fichier PHP vu précédemment avec l'objet WWW.

C'est ici, que ce que renvoie le fichier PHP est important, s'il renvoie 0, le script envoie un message au Debug d'Unity pour donner l'information que tout s'est bien passé. Sinon, le script renvoie un message d'erreur.

Maintenant que nos comptes peuvent être enregistrés sur notre base de données, nous devons nous occuper de la partie de connexion.

Nous reprenons une scène Unity assez équivalente à celle d'enregistrement. C'est en majeure partie le script et le fichier PHP qui va changer. **(Voir Annexe 1.5.4)**

Il nous faut récupérer toutes les informations de notre utilisateurs (le nom, le grain de sable, le mot de passe haché, et le nombre de pièces).

Nous vérifions ensuite qu'il existe exactement un utilisateur avec le même nom pour qu'il n'y ait pas de problème au niveau des pièces et mots de passe.

Ensuite on hache le mot de passe donné par l'utilisateur lors de sa connexion pour pouvoir le comparer au mot de passe présent sur la base de données (c'est le seul moyen de vérifier la validité du mot de passe car, comme dit précédemment, le hachage est à sens unique).

Pour finir, nous voulions pousser le principe jusqu'au bout nous avons décidé de mettre en ligne cette base de données, mais nous avons rencontré un soucis : notre site **colors-tsf.me** n'accepte pas les pages dynamique comme le PHP. De ce fait nous avons pris un autre domaine :

colors-tsf.000webhostapp.com. Il nous permet d'exécuter nos fichier PHP. L'hébergeur nous permet de créer directement une base de données MySQL très facilement. Nous avons donc simplement recopié la base données locale pour recopier les identifiants données par le site dans nos fichiers PHP.

De ce fait nous pouvons nous connecter depuis n'importe quel ordinateur partout dans le monde.

Des informations supplémentaires quant à l'implémentation de cette base de données dans les menus du jeu seront détaillées dans la partie Interface.

3.5.4 Apparences

Pour qu'il y ait un système de boutique, il faut qu'il y ait des éléments à acheter. Nous nous sommes alors lancés dans la réalisation de différentes apparences.

Nous étions conscients qu'il fallait seulement ajouter un logo, une référence à la pop culture par-dessus notre personnage pour créer ce système. C'est pour cela que nous nous devons de créer une banque d'images au format PNG.

Nous avons beaucoup hésité quant au nombre de skins que nous voulions proposer aux joueurs. Nous nous sommes accordés sur une quinzaine d'apparences afin d'inciter le joueur à collecter le plus de pièces possibles.

Pour réaliser ces différentes apparences, nous avons utilisé plusieurs méthodes. Dans les cas les plus faciles, il nous suffisait de trouver des icônes ou des photos sur le site <https://www.flaticon.com/>. Grâce à ce site nous pouvions récupérer des images de couleurs de notre choix au format PNG. Il fallait que les images ne soient pas pleines. Nous avons donc sélectionné des images avec seulement des contours de formes sur ce site. Cela permet une lisibilité optimale pour le changement des couleurs.

Ensuite, s'il n'existait pas de version PNG d'un logo de série par exemple, il nous a fallu éditer l'image originale à l'aide d'un logiciel de traitement de photo. Nous nous sommes servis de GIMP pour cette tâche. D'abord, nous avons téléchargé les images sur internet. Nous les avons ensuite ouvert dans GIMP. Après cela, nous avons utilisé l'outil de détournage pour détacher le logo du fond. Une fois l'objet détourné, nous devons ajouter un canal alpha afin de créer un fond adéquat

à l'exportation de ladite image en PNG. Après avoir fait cela, il nous a suffi de supprimer le fond de base. Une fois tout cela réalisé, nous n'avons plus qu'à exporter le fichier.

Pour ce qui est des prix des skins, nous les avons adaptés au nombre de pièces disponibles dans les niveaux. En effet les cinq premiers skins seront les plus simples et donc les rapides à débloquent. Les cinq suivants sont ceux qui commencent à être particuliers. Et les cinq derniers sont ceux que nous préférons ainsi que ceux possédant des formes particulières, spécialement les apparences recouvrant une plus grande surface.

Ces apparences permettent de faire envisager au joueur de retenter le niveau avec plus de 15 apparences différentes il y en aura pour tous les goûts !

3.6 Interfaces (Alex, Jean-Baptiste, Alexandre)

3.6.1 Présentation

L'interface est une partie importante de notre projet, elle nous permet de nous rapprocher d'un rendu propre et fini. Il nous faut une interface fluide et surtout intuitive. Toute notre partie interface sert à "englober" la partie réelle de jouabilité dans notre jeu. Pour nous il est essentiel de réaliser une belle interface ainsi que complexe pour se rapprocher réellement d'un rendu soigné, propre voire même professionnel. Cette partie du projet représente une majeure partie du temps passé sur Unity du fait des multiples recherches mais aussi de l'implémentation de notre design sur Unity.

Une interface est utile pour communiquer entre deux systèmes dans notre cas nous voulons que l'utilisateur puisse communiquer avec le jeu mais aussi la base de données. Ceci est beaucoup plus intuitif que de taper des lignes de code pour pouvoir changer une valeur dans la base donnée.

Nous avons décidé de créer cinq interfaces pour nos menus :

- Menu de connexion
- Menu de création de compte
- Menu principal
- Le magasin
- Le menu de sélection du niveau

Pour les menus de connexion et de création de compte nous avons dû communiquer directement avec la base de données pour pouvoir vérifier les données émises par l'utilisateur.

Une fois les données stockées localement grâce à la page d'identification nous pouvons donc les afficher sur nos menus et permettre, grâce à des scripts,

de modifier ces dernières. Tous nos menus sont contrôlables à la manette. Ceci étant dit, nous allons maintenant voir comment le design de l'interface est retranscrit dans Unity.

3.6.2 Implementation Unity

3.6.2.1 Menu principal

Nous avons commencé par la création du menu principal, c'est le menu le plus important, celui que le joueur verra le plus souvent. Tout commence par la création d'une feuille de canvas, c'est ici que va reposer tous nos objets. L'ajout du fond d'écran fut simple, nous avons juste ajouté une image avec comme source notre fond d'écran puis nous avons recadré cette image afin qu'elle remplisse tout l'écran.

En parlant de la caméra, elle est dimensionnée en fonction de notre canvas comme cela elle ne filmara que notre menu dans son intégralité peu importe la taille de notre canvas. La caméra utilise une projection orthographique puisque notre jeu est en deux dimensions nous n'avons pas besoin de l'effet de profondeur. Une fois le fond d'écran intégré le logo du jeu a été à son tour mis sur la scène Unity.

Un des premiers problème apparut, le logo était invisible lors du rendu de la scène. Après de multiples recherches nous avons appris la notion "d'ordre de couche" ou appelé *Order in layer*. Ce paramètre représente l'ordre de priorité d'une image. De ce fait le fond d'écran a été initialisé à l'indice -1 puis l'image à 0. Cette solution a réglé tous nos soucis d'objet invisible lors du rendu.

Les boutons sont le coeur d'une interface, nous avons donc utilisé les boutons de base de Unity. Leurs textures initiales ne correspondent pas à notre design, de ce fait nous avons créé un fichier "img" comportant toutes les textures de boutons provenant de nos rendus au format PNG expliqué dans la partie design.

Une fois l'image du bouton changé et les dimensions du bouton en fonction du ratio de l'image nos boutons ressemblent enfin à notre vision des choses. Vous trouverez en annexe l'image de ce bouton (annexe 1.6.1)

3.6.2.2 Menu des options

Dans notre menu principal un **sous menu** est présent, il apparaît au-dessus du menu principal, celui des options. Mis à part contenir deux boutons qui sont de la même forme que ceux du menu il y a la présence d'une barre de contrôle de volume (Slider). Nous avons changé sa texture comme précédemment puis grâce à un script nous pouvons contrôler le volume du son présent sur la scène qui est exécutée à chaque fois que la valeur du Slider change.

En parlant du **son**, c'est un objet nommé Source Audio qui n'a pas été facile pour nous de l'implémenter comme nous le voulions. En effet lors du changement de scène la musique recommençait depuis le début. Nous avons donc cherché une solution, la première était de sauvegarder le temps d'écoute (Time code) de la musique et lors du changement de scène d'appliquer ce temps d'écoute à la musique. Mais ce n'était pas parfait, on observait cette transition qui n'était pas très fluide donc nous avons cherché une autre solution.

Nous avons finalement adopté une solution trouvée sur internet qui est d'utiliser la fonction *DontDestroyOnLoad* (trouvable en annexe 1.6.2) qui permet de garder un objet lors du changement de scène. De ce fait maintenant l'objet de son est sauvegardé et appliqué sur toutes les scènes sans être détruit, il y a donc plus de coupure entre les scènes et la musique semble être continue sur tous les menus.

Cet objet, sauvegardé remplace celui présent sur la scène du menu principale pour que le contrôleur de volume soit toujours effectif.

Pour le moment notre menu principal est implémenté voyons maintenant comment le menu de connexion et de création de compte sont créés.

3.6.2.3 Menu de connexion

Pour **le menu de connexion** nous avons repris de la même façon notre fond d'écran, de même pour le logo du jeu. Puis nous avons intégré un carré arrondi blanc avec un ordre de couche de 0 comme ça il est rendu au-dessus de notre fond.

L'élément principal de ce menu est la zone d'entrée (Input Field). C'est ce qui permet à l'utilisateur d'entrer des données comme son nom d'utilisateur ou mot de passe. Nous avons modifié la police d'écriture pour que cela corresponde à notre design. A chaque fois que l'utilisateur change la valeur de cette zone la fonction *VerifyInputs* est exécuté pour vérifier si les conditions pour que le bouton de connexion soit débloqué. Pour la connexion il suffit simplement que les 2 champs d'entrée ne soient pas vides.

Le dernier élément présent sur cette scène est donc un bouton “valider” qui va lancer la procédure d’identification qui est décrite dans la partie Compte et Base de données.

3.6.2.4 Menu de création de compte

Le menu de **création de compte** est composé d’exactement les mêmes éléments que celui de connexion. Les seules différences sont les conditions de validation et que le bouton de validation lance cette fois la procédure d’enregistrement. Les conditions pour la création de compte sont les suivantes:

- Le nom d’utilisateur doit être composé d’au moins 1 caractère
- Le mot de passe doit être composé de plus de 7 caractères
- Le mot de passe doit contenir au moins un chiffre

Ces conditions se traduisent avec le code suivant :

```
submitButton.interactable = (nameField.text.Length >= 1 &&
passwordField.text.Length >= 8 && passwordField.text.Any(char.IsDigit));
```

3.6.2.5 Menu de selection de niveau et magasin

Le menu de sélection de niveau et le magasin fut les plus simples à retranscrire sur Unity, après avoir intégré le fond sur notre canvas nous avons simplement créé une matrice de 5x3 de bouton ayant respectivement leur nom de niveau ou leur apparence. La navigation est donc intuitive dans un rectangle parfait.

Maintenant que la forme du menu est réalisé nous avons eu une première idée pour la navigation : donner un numéro pour chaque bouton afin de pouvoir les identifier. Puis nous avons vu le problème comme une matrice où nous pouvons nous déplacer. Nous avons réalisé le même principe avec le magasin qui ressemble plus à une matrice donc l’algorithme était plus facile. (Annexe 1.1.4)

Cette méthode fonctionne bien, mais nous nous sommes rendu compte qu’il y avait quelques bugs. Nous avons découvert par la suite qu’un système de navigation intégré à Unity interférait avec le nôtre. Ce dernier est beaucoup plus simple à comprendre mais surtout aucun script n’est nécessaire pour le faire fonctionner. Nous pouvons donc sélectionner quel bouton est sélectionné si l’on se dirige vers le haut vers le bas, à droite et à gauche depuis le bouton présent. Cela facilite grandement la navigation.

3.6.3 Liaison à la base de données

En effet nous utilisons la base de données pour garder en mémoire les objets déverrouillés par les joueurs, mais aussi leur avancée dans les niveaux (niveau terminé, terminé à 100%, non fait ou indisponible).

Nous utilisons cette même base pour l'achat des objets et l'affichage du pseudo et du nombre de pièces que le joueur possède.

En premier lieu voyons comment ce système est organisé au niveau de la base de données

Pour effectuer cette liaison entre l'interface et la base de données nous devons dans un premier temps les stocker dans la base. J'ai donc créé 30 colonnes supplémentaires : 15 pour les objets et 15 pour l'état des niveaux. Elles contiennent un seul chiffre dont voici la traduction :

Pour le magasin:

0 : Objet non acheté
1 : Objet acheté
2 : Objet équipé

Pour les niveaux :

-1 : Niveau indisponible
0 : Niveau non réalisé
1 : Niveau réalisé
2 : Niveau réalisé à 100%

Il est très simple par la suite de modifier la structure de la base données pour pouvoir ajouter un niveau ou un objet de cosmétique.

Une fois la structure de la base de données réalisée nous devons récupérer ces données en fonction de chaque joueur.

Nous voulons récupérer ses données en fonction de son nom d'utilisateur.

Nous avons donc déclaré les variables dans notre fichier PHP (\$namecheckquery) que nous allons utiliser pour modifier notre interface :

```
$namecheckquery = "SELECT username, salt, hash, score,  
    lvl11, lvl12, lvl13, lvl14, lvl15, lvl21, lvl22, lvl23, lvl24,  
    lvl25, lvl31, lvl32, lvl33, lvl34, lvl35,  
    skin1, skin2, skin3, skin4, skin5, skin6, skin7, skin8, skin9,  
    skin10, skin11, skin12, skin13, skin14, skin15  
    FROM players WHERE username='" . $username . "'";
```

Cela va nous permettre de savoir si la connexion peut être établie et que les identifiants sont corrects. Tout cela pour ensuite retourner les informations correspondantes au joueur.

Pour réaliser ces vérifications, il faut que nous séparions les éléments avec un caractère spécial pour faciliter le "Parsing" que nous effectuerons sur UNITY.

De ce fait nous séparons tous les éléments par le caractère “\t” comme montré ci-dessous.

```
echo "0\t" . $existinginfo["score"] . "\t" . $existinginfo["lvl11"] .  
"\t" . $existinginfo["lvl12"] . "\t" . $existinginfo["lvl13"] . "\t" .  
$existinginfo["lvl14"] . "\t" . $existinginfo["lvl15"]  
    . "\t" . $existinginfo["lvl21"] . "\t" . $existinginfo["lvl22"] .  
"\t" . $existinginfo["lvl23"] . "\t" . $existinginfo["lvl24"] . "\t" .  
$existinginfo["lvl25"] . "\t" . $existinginfo["lvl31"]  
    . "\t" . $existinginfo["lvl32"] . "\t" . $existinginfo["lvl33"] .  
"\t" . $existinginfo["lvl34"] . "\t" . $existinginfo["lvl35"] . "\t" .  
$existinginfo["skin1"] . "\t" . $existinginfo["skin2"];
```

Au final les informations sont contenues dans une grande chaîne de caractère :

www.text

Nous utilisons donc la fonction Split pour avec chaque informations séparément avec le paramètre : “\t”

Nous pouvons donc appliquer la diffusion des données à nos boutons en changeant leurs couleurs en fonction. Pour les données suivantes voici l'interface correspondante :

0	0	1	0	0	2	0	0	1	0	0	2	0	0	-1	-1	-1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	----	----	----

Le premier 0 correspond au code d'erreur dans notre cas il n'y a pas d'erreur.

Le deuxième 0 correspond à l'argent que le joueur a sur son compte.

Puis les chiffres qui suivent sont l'état des niveaux de 1-1 jusqu'à 3-5

Voici donc la représentation graphique après l'application du script en fonction de ces données récupérées.



Ce même principe est le même pour le magasin. Un problème est identifiable lors de la présence de boutons désactivés, le joueur peut se déplacer sur ces derniers, mais aucune indication n'est montrée. De ce fait nous avons créé un script, qui désactive l'accès aux boutons grisés(Annexe).

Mais où sont stockées toutes ces données ? Nous avons créé une classe DBmanager où sont sauvegardées toutes les données lors de la connexion de l'utilisateur (Annexe 1.6.3). Les modifications sont effectuées sur cette classe puis grâce au script savedata, nous envoyons ces informations vers la base de données pour les sauvegarder.

3.7 Niveaux (Alex, Romain)

3.7.1 Assemblage

Dans cette partie, nous allons parler de la création de nos niveaux, donc de l'assemblage de toutes nos différentes créations présentés durant tout ce rapport. En effet la création d'un niveau relève d'un assemblage ou plus précisément d'une superposition d'élément qui aboutissent à la création d'un jeu vidéo. Du fait que la majorité des éléments de notre projet ont été réalisés séparément l'équilibrage représente la partie la plus importante du processus de création du niveau. C'est pour cela que ne créer que quelques obstacles ne permet pas de réaliser un jeu correct c'est l'alchimie et la diversité des motifs d'ennemis qui rend les niveaux intéressants.

Dans un premier temps, nous avons ajouté un fond pour nos niveaux toujours sous la maîtrise du **Flat Design** sur toutes nos créations. Il a fallu inclure un script en C# pour que le fond ne sorte pas de l'écran de jeu puisque notre personnage avance horizontalement.

Par la suite, nous avons ajouté des pièces que le joueur peut récupérer, pour ensuite les échanger dans la boutique de notre jeu. Pour cela nous avons créé un nouveau **symbole graphique (Sprite en anglais)**. A ce sprite nous avons donc donné l'image d'une pièce de monnaie venant de Flat Icons pour que le rendu soit propre. Pour que le joueur puisse collecter ces pièces il faut donc ajouter une fonction au personnage, qui va détecter quand celui va rentrer en collision avec une pièce et seulement une pièce.

Pour cela nous avons utilisé une fonction appelée **OnTriggerEnter2D** présentée ci-dessous :

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.étiquette != currentcolor)
    {
        UnityEngine.Debug.Log("Game Over");
    }

    if (other.IsTouching(CollisionPlayer) &&
other.Compareétiquette("MoneyCoin"))
    {
        Destroy(MoneyCoin.gameObject);
        CoinCounter++;
    }
}
```

Dans toute la première ligne du **If**, on vérifie en premier si le joueur est en collision avec n'importe quel objet. On spécifie si cet objet possède comme **étiquette**, donc comme nom "**MoneyCoin**", nom que nous avons associé à nos pièces. Si cette condition est vérifiée, alors on utilise la méthode **Destroy** qui est déjà implémentée dans UNITY qui permet tout simplement de faire disparaître un élément de la scène.

Enfin nous avons décidé d'implémenter un compteur de pièce qui apparaît en haut à gauche de l'écran. Pour cela il fallait ajouter un texte de type **UI (User Interface)**. Pour pouvoir actualiser notre compteur de pièce, nous avons simplement ajouté des variables dans notre script du personnage.

```
public Text CurrentCoin;  
public int CoinCounter;
```

- La variable **currentcoin** de type **Text** est celle qui est utilisée dans le nouvel élément ajouté à la scène de type **Text**. C'est avec cette variable que l'on peut afficher le compteur sur la scène.
- La variable **coinlvl** qui est donc de type **int** va nous permettre d'actualiser notre compteur à chaque collision avec la fonction vu plus haut.

Ensuite, dans la fonction **Update** de UNITY, il nous suffit simplement d'actualiser le compteur à chaque collision entre le joueur et une pièce.

Une fois que toutes ces informations sont rassemblées et mises en place nous obtenons un niveau. Vous trouverez en **annexes** des impressions d'écrans de nos niveaux.

3.7.2 Extension

Notre jeu comporte trois niveaux différents avec chacun leur difficulté. Il va de soi que la création de plusieurs niveaux offrent aux joueurs de la diversité ainsi qu'une meilleure expérience de jeu. Nous avons varié la construction de nos niveaux en variant la position de nos obstacles ainsi que leurs mouvements pour accentuer la diversité.

De plus, de manière évidente le fait qu'un jeu comporte plusieurs niveaux envoie notre projet directement dans une autre dimension, celle du professionnalisme. Notre but également dans ce projet est non seulement de rendre quelque chose de jouable, de fonctionnel, mais aussi quelque chose qui se rapproche le plus possible des standards des jeux vidéos créés aujourd'hui

4. Récit de réalisation

4.1. Récit général de la réalisation

Comme vous devez probablement vous en douter le récit de notre projet commence par la sélection des membres de celui-ci nous allons donc débiter ce récit en racontant la formation du groupe

Romain et Alex furent les deux premiers éléments du groupe, ils avaient dès lors une idée assez claire de ce à quoi allait ressembler leur jeu. Tous les deux étant des joueurs assidus de jeu mobile ils se sont rapidement tournés vers quelque chose de ce style. Par la suite ils ont contacté Alexandre et Jean-Baptiste qui ont été eux aussi ont trouvé que le principe de runner avec un aspect coloré pourrait s'intégrer correctement dans le relief des jeux-vidéos compte tenu du style de jeu auquel une grande majorité de la population joue à l'heure actuelle.

En effet pour faire le choix du style de jeu que nous allions choisir nous nous sommes posés deux questions simples : Quel style de jeu peut potentiellement séduire le plus large panel de personne et ensuite quel allait être l'idée centrale du jeu. Les réponses ont été assez simples à trouver. En effet avec le développement des smartphones et des tablettes la majorité des activités vidéoludiques se fait sur ces plateformes, généralement dans les transports.

Il nous fallait donc un système de jeu au gameplay rapide avec des éléments cosmétiques implémenté, puisque qu'avec les innovations en terme de mécanique de jeu, les éléments cosmétiques arrivent en tête dans les classements des éléments les plus appréciés et recherchés par les joueurs.

La suite logique des choses fut de répartir les tâches entre les différents membres du groupe. Pour cela nous avons commencé par sélectionner en fonction des affinités de chacun avec les sujets mais également en fonction de ce que cela pourrait leur apporter. Le premier à s'être désigné pour la réalisation d'un des axes du projet a été Jean-Baptiste pour la réalisation du site internet, il considérait en effet ce projet comme une opportunité pour acquérir de nouvelles connaissances sur d'autres langages que le C#.

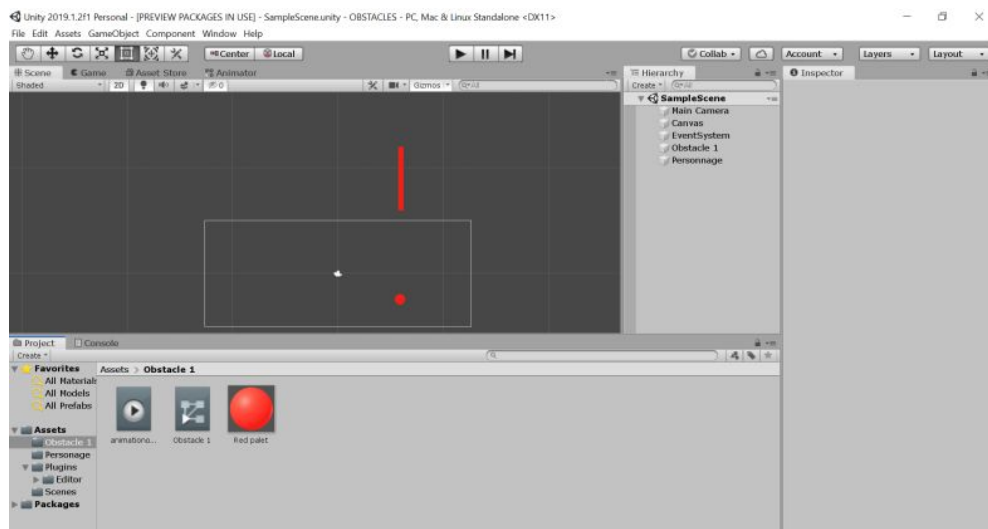
Est venu ensuite la partie interface boutique et compte qu'Alexandre a souhaité réaliser étant donné qu'il possédait déjà des connaissances sur le sujet principalement grâce à son projet d'ISN de terminale. Il souhaitait également participer sur la partie graphique qui est un élément chère à ses yeux étant un grand fan de graphisme dit "*material design*" et "*flat design*".

Pour ce qui est de la partie jeu Romain s'est porté volontaire en effet au vu de ses connaissances et de ses capacités en programmation. Il considérait tout comme Jean-Baptiste, le projet comme un moyen de l'aider à comprendre les

différents aspects de la programmation. Alex a également choisi d'intégrer la partie jeu du projet sensiblement pour les mêmes raisons que Romain.

Avant de commencer à entrer dans le vif du récit de la réalisation de notre projet, nous tenons à rappeler nous pensons tous du fait de l'implication requise pour l'avancement du projet que celui-ci nous a permis de gagner en autonomie et nous a permis de développer notre abnégation dans le travail et dans le dépassement de soi.

Le début de la réalisation du jeu a commencé environ une semaine avant la première soutenance orale. Nous savions que nous devions avancer sur la réalisation de celui-ci compte tenu du fait que depuis la formation du groupe aucun travail concret n'avait été fait. Uniquement de la documentation et de la mise en place par rapport aux différents outils dont nous nous sommes servis, Github desktop par exemple. Nous avons ainsi débuté en créant la toute première esquisse de niveau, nous étions encore bien loin du résultat voir même de ce que nous avons pu imaginer.



(premières images du jeu)

Comme vous pouvez le voir sur l'image ci-dessus: les couleurs n'étaient pas issues de la palette que nous voulions utiliser, les bordures des ennemis étaient très prononcées et il n'y avait même pas de fond. Plus grave encore compte tenu du fait que nous pensions intégrer des passages entre trois dimensions et deux dimension nous avons utilisé des propriétés d'objets 3D, on peut noter par exemple le fait que notre personnage n'était pas un sprite mais une sphère . Malgré tous ces défauts c'est ce fichier qui a été à l'origine de la compréhension du fonctionnement de Unity, plus particulièrement de l'intégration de script, de l'utilisation de l'**Animator** ou encore des déplacements du personnage.

Nous avons tiré un certain nombre d'enseignement de la première séance notamment au niveau de la gestion du temps que nous devons accorder au projet et à l'avance que nous devons prendre pour ne pas être débordé de travail. Ainsi deux semaines avant la seconde soutenance nous avons fait une refonte complète de notre projet en commençant par le fichier unity.

Nous avons donc refait le premier niveau de notre jeu en 2D, fais les modifications que nous souhaitions par rapport aux déplacements du personnage (cf partie personnage) mais surtout ajouté le premier véritable ennemi de notre jeu le cercle. Rétrospectivement deux semaines paraissent assez courtes au final pour faire toutes ces modifications et implémentation correctement. Lors de la seconde soutenance nous avons fait une grosse erreur nous avons omis d'implémenter des images de notre jeu pour préférer un diaporama dynamique et harmonieux.

A l'avis général du groupe cette soutenance a sonné comme un signal pour la partie jeu qui avait été bien moins avancé et mis en avant lors de cette même soutenance.

Comme je l'ai dis précédemment nous avons pris le projet comme une chance pour apprendre à gérer des "deadline" comme un ingénieur pourrait le faire dans son travail. Mais un élément nous a complètement échappé et il n'en est pas des moindres, il s'agit de la hiérarchisation des modules d'un projet. Pour expliquer plus clairement nous avons avant même de réaliser le jeu, avancé des éléments comme les interfaces ou encore le site internet qui, malgré le fait que ceci ait été très bien réalisé par les différents membres du groupe, reste des éléments complémentaires du projet et non la racine de celui-ci.

Pour remédier à ce problème nous avons fait une refonte de l'organisation des tâches de manière à optimiser nos résultats, nous nous sommes mis tous sur une seule et même tâche au lieu de sectionner en quatre comme nous le faisions avant. Il en est résulté un gain de temps évident, un meilleur rendu des niveaux du au fait qu'ils proviennent de la réflexion commune et non de l'idée d'une seule et même personne. La diversité qui est je le rappelle pour notre projet un des objectifs principaux de Colors .

La récit de la réalisation de notre projet se conclut donc à l'orée de la troisième et dernière soutenance. Nous espérons tous que les enseignements que nous avons tirés de nos erreurs mais aussi de nos réussites ont été les bons et que le résultat final saura satisfaire aussi les biens les attentes que nous nous sommes fixés que celles attendus d'un projet d'élève de première année

4.2. Aspect musical du jeu (Jean-Baptiste)

Depuis le début du projet, nous n'avions pas réellement réfléchi à la manière dont nous allions implémenter un univers musical à notre jeu. Pour autant, nous savions que c'était une partie plus que nécessaire à la finalisation du projet. En effet, on ne peut pas imaginer un jeu sans musique puisqu'en leur absence, le joueur aura directement une impression de vide sonore.

Pour remédier à cela, nous avons décidé de nous occuper nous-mêmes de la partie musicale du jeu.

Nous voulions, comme pour le design visuel du jeu, tout créer de A à Z. C'est pour cela que nous avons décidé de réaliser nos propres musiques sans en récupérer sur des sites proposant des bandes audio libres de droits. Cette envie de création a été accentuée par l'envie, également, de découvrir la base de la production sonore.

Pour réaliser ces musiques, nous avons utilisé le logiciel BoscaCeoil. Nous avons choisi ce dernier car il est très facile d'utilisation et qu'il permet de créer des musiques typées pour les jeux-vidéos. Il est également très pratique puisqu'il ne requiert que très peu de mémoire et peut même être utilisé directement sur navigateur.

En regardant une vidéo Youtube d'un développeur réalisant une musique pour son jeu, j'ai compris quelques bases de création de musique. J'ai choisi d'utiliser 4 boucles différentes pour créer une instrumentale qui soit travaillée : une mélodie assez simple avec une basse (en rouge), des éléments de batteries pour donner du rythme(en gris), quelques notes de musique 8 bits pour rester dans le thème du jeu-vidéo(en violet), et enfin seulement trois notes (pour éviter la surcharge) de piano pour ajouter un peu de profondeur à la musique(en bleu).

Toutes les musiques composées n'excèdent pas une minute de durée chacune. Ce choix a été fait pour des raisons techniques, nous ne voulions pas nous lancer dans une composition trop longue pour nos premiers morceaux, mais également pour des raisons pratiques puisque nos niveaux sont assez courts. Pour autant, si le joueur reste longtemps dans le menu principal, il ne va pas se retrouver sans musique après une minute de parcours des interfaces. Nous avons faits en sorte, dans UNITY, que les musique se jouent en boucle. Nous avons donc dû penser à rendre les boucles (des morceaux entiers cette fois) lisses. Pour cela, il a fallu que les fins de morceaux correspondent plus ou moins à leur début. Pour prendre l'exemple du morceau du menu, ce dernier finit par les notes de piano qui ouvrent la musique. Ainsi, le moment de boucler la musique se fait sans accroc.

Cela permet au joueur de ne pas entendre dès la première écoute que la musique n'est qu'en fait composée sur une seule minute.

Comme vu sur la capture d'écran ci-dessus, nous n'avons pas laissé les 4 boucles en continu. Il fallait faire varier la musique afin qu'elle ne soit pas trop redondante. Au début, nous avons seulement mis la boucle de piano. Nous avons ensuite rajouté la boucle avec les basses sur la mesure suivante, et ainsi de suite jusqu'à avoir les quatre boucles superposées.

Par la suite, nous retenons la batterie pour créer un pont, cette fois encore pour créer de la variation dans la musique. Une fois ce morceau finalisé, nous étions convaincu que ce morceau conviendrait parfaitement pour le menu du jeu. En effet avec son rythme assez lent mais avec une mélodie qui était un minimum tonique, elle allait donner envie au joueur de parcourir le jeu par la suite.

Tout de même, nous nous sommes rappelés que dans les jeux mobiles dont nous nous sommes inspirés, il était toujours possible de régler le volume du son dans le menu principal du jeu. Grâce à cette possibilité, un joueur qui n'aimait pas la musique du jeu pouvait réduire cette présence musicale, voire la faire disparaître. Cela permet également une certaine modularité de l'expérience du joueur.

En effet, ce dernier ne se sentira pas restreint à écouter en boucle une chanson qu'il n'aime pas et n'a pas choisi. Si ce dernier doit passer beaucoup de temps dans les menus également, la musique peut le déranger, et la possibilité de pouvoir régler son volume lui permettra d'apprécier les différentes interfaces sans gêne.

Pour ce qui est du tempo, la musique du menu a pour caractéristique un BPM (battement par minute) de 80. C'est un tempo assez faible puisque l'ambiance dans le menu doit être assez posée. Pour cette ambiance, nous avons choisi de faire quelque chose d'assez entraînant puisque c'est dans le menu que la première impression du jeu se fait. Cela dépeint donc une expérience de jeu amusante et rythmée.

Une fois cette première musique réalisée, nous nous sommes rendus compte qu'il était assez facile de créer quelque chose d'assez correct pour animer l'ambiance sonore de notre jeu. Cela nous permet de donner une expérience complète au joueur tant au niveau de l'image que du son.

Pour ce qui est des musiques de niveau, le battement par minute augmentera en fonction de la difficulté générale de chaque scène. Ainsi, les musiques lors des phases de jeu auront un tempo plus élevé que celle du menu. Par exemple pour la musique du premier niveau, le BPM est de 95, ce n'est pas

encore très élevé, mais on sent directement la différence de vitesse avec la musique des menus.

Cette musique, comme celles des deux autres niveaux, permet de souligner que l'attention du joueur est toute requise afin de parvenir à la fin du niveau ou de récupérer toutes les pièces de ce dernier.

Pour ce qui est des éléments sonores présents dans ce premier morceau en jeu. Nous étions partis sur une mélodie principale avec une flûte et une cloche selon BuscaCeoil mais qui s'apparente en réalité plus à un xylophone. Nous trouvions, lors de la composition, que ces instruments apportaient une sonorité intéressante. Mais une fois l'intégralité du morceau composé et après avoir écouté cette mélodie une bonne cinquantaine de fois, nous nous sommes rendus compte que cette dernière était très irritante et allait déranger le joueur très rapidement lors de sa partie.

Nous avons donc échangé cette mélodie par une autre de piano plus discrète. En commençant par le piano, nous ne mettons pas en avant une mélodie entêtante. Celle avec le semblant de xylophone a été ajoutée en milieu de morceau afin de réduire sa présence.

Nous avons également diminué son volume afin qu'elle ne prenne en aucun cas le pas sur la mélodie de piano. Pour ce qui est des éléments de batterie, il sont bien plus présents et rythmés. En effet, nous n'avons laissé que très peu de d'espace musical entre chaque itération d'une même note. Pour finir sur ce morceau, nous n'avons pas utilisé de son de basse car les trois éléments musicaux énoncés plus hauts nous semblaient suffisant pour la création d'une musique engageante pour le joueur.

Pour clôturer cette partie audio du jeu, nous avons ajouté un son au saut de notre personnage. Pour cela, il nous a suffi de réaliser un bruit de bouche qui nous plaisait et qui n'était pas trop entêtant. Ainsi, c'est un bruit assez discret qui ressemble à l'explosion d'une bulle de savon. Ce dernier peut être répété souvent sans être dérangent.

Etant donné que c'était la première fois que nous implémentions un effet sonore au jeu, nous n'avions pas compris l'importance du timing de ce dernier. En effet, au départ, nous avions laissé un petit délai entre le début de l'enregistrement et le début du bruit en lui-même. Cela a causé un décalage non négligeable qui cause un problème de cohérence pour le joueur. Comme le son n'était pas synchronisé avec le saut, lors de nos sessions de tests nous nous sommes demandés s'il n'y avait pas un problème ou un bug engendrant le décalage.

Nous avons donc cherché si nous n'avions pas engendré un ralentissement avec du code avec une quelconque fonction lourde. Il n'en était rien, le seul problème était donc le montage de cette séquence audio. Il nous a donc suffi de raccourcir ce clip audio pour avoir le timing voulu pour le bruitage.

4.3. Ressentis personnels

Dans cette partie, chaque membre du groupe évoquera ses ressentis vis à vis du projet, de son organisation et de sa réalisation en général.

4.3.1 Alexandre Bermudez

Avec l'ISN j'ai pu découvrir le travail de groupe, mais ce projet et de loin plus complexe au niveau de l'organisation.

Grâce à cette expérience j'ai pu découvrir une réelle passion pour les interface et les bases de données. Le fait de faire interagir une interface en fonction des données de l'utilisateur me passionne. Me dire que nous utilisons le même système de base de données que les plus grands sites me fait relativiser sur nos progrès en informatique.

MySQL nous servira sûrement plus tard donc ce fut avec plaisir que je me suis renseigné sur ce sujet pour pouvoir l'implémenter sur le jeu.

J'ai beaucoup aimé partager mon savoir sur le design à mon groupe cela permet d'être assez satisfait du rendu lorsque les choses fonctionnent. J'ai pu apprendre de nouvelles notions de ce domaine surtout sur l'implémentation sur un moteur graphique. J'ai trouvé cette étape pas simple surtout quand on veut que le design théorique ressemble parfaitement à celui implémenté.

4.3.2 Jean-Baptiste Despujol

Avant le début de ce projet, l'idée de réaliser un jeu de A à Z était à la fois assez effrayante et excitante. Comme dit dans le cahier des charges, je pense que c'est un très bon moyen d'apprentissage. Nous découvrons des problèmes seuls, cherchons des solutions par nos propres moyens, nous discutons ensemble pour savoir de quelle manière aborder une tâche difficile afin que le travail des autres n'en soit pas impacté.

C'est cette réelle dimension de travail de groupe qui m'a fait réaliser l'importance de la communication dans un tel projet. Si nous ne nous tenions pas mutuellement au courant de l'avancement des modules, il était impossible pour les suppléants d'aider les membres en charge d'une partie du projet. Grâce au projet, Alexandre a pu m'apporter une aide non négligeable à l'élaboration et l'implémentation de la base de données de notre jeu. Il possédait déjà quelques

connaissances en PHP, ce qui nous a permis de comprendre et d'aborder plus facilement cette partie. Parfois, j'ai également pu soutenir Romain dans la partie personnage lorsqu'il butait sur la mise en place des skins pour notre joueur.

La gestion du temps est un point important dans la réalisation. Nous avons compris, même si cela semble évident, que travailler un minimum de manière régulière produisait un résultat plus qualitatif. Cela permet également de ne pas avoir à se presser juste avant une échéance. Le travail étant fait à l'avance, le stress est minimisé.

Pour ce qui est des tâches pures, au tout début du projet, je ne me sentais pas réellement prêt à travailler sur le jeu en lui-même. C'est pour cela que les parties dont je me suis occupé ne concerne pratiquement pas de code pour les niveaux de jeu. Néanmoins après avoir observé Romain et Alex travailler sur leur partie respective, je me suis rendu compte que ces parties m'intéressait beaucoup. J'ai pris beaucoup de plaisir à concevoir le site internet et le résultat final me rend très fier. Découvrir pas à pas les possibilités du HTML et CSS a été très prenant et m'a donné toujours envie d'aller plus loin pour arriver à une présentation esthétique.

Ensuite la base de données m'a permis d'acquérir des connaissances dans divers domaines. Tout d'abord, la communication web était tout nouveau pour moi. Le PHP semblait assez hostile, mais avec un peu de pratique et l'aide de tutoriels vidéo, on comprend tout de même la majeure partie de ce que l'on écrit. Quel bonheur lorsque l'on voit le message du debugger UNITY qui confirme l'échange d'information entre le jeu et la base de données. Une fois la base de données modifiée en ligne, nous nous rendons vraiment compte que nous avons accompli quelque chose de significatif à notre échelle. Cette partie m'a également permis d'apprendre les base du traitement de photo grâce au logiciel GIMP.

4.3.3 Romain Gregoire

Selon moi le projet de S2 est un excellent entraînement pour les élèves de premières années par bien des aspects. Tout d'abord il est important de rappeler que comme Alex je n'ais pas fais de spé ISN et je n'avais jamais codé au préalable de jeu, uniquement des programmes mathématiques, la réalisation d'un jeu comme celui-ci était donc une première pour moi. De plus au vu de mes difficultés en programmation j'ai pensé qu'il s'agissait d'une bonne opportunité pour essayer d'aborder cette matière par un autre biais que celui des travaux pratiques réalisé par les ACDC.

C'est pour cela que je me suis porté volontaire pour réaliser la partie jeu pur. Très honnêtement je pense que le projet est un très bon entraînement pour un

certain nombre de raison : par rapport à la gestion d'un projet, d'une équipe mais plus important encore, il permet aux élèves ayant des difficultés de les combler petit à petit en cherchant eux mêmes sur leur propre travail . De plus je pense qu'en tant que chef de projet, cela apporte un côté plus sérieux et nécessite certaine prise d'initiative, bien que les éléments de ce groupe ont toujours été très sérieux dans leur travail et que le "rôle" de de chef de projet n'a pas eu besoin d'être prépondérant dans l'organisation du groupe.

Pour conclure je pense que le projet n'a que des vertus et je suis très fier d'avoir pu le réaliser avec mon groupe.

4.3.4 Alex Poiron

Personnellement je n'avais jamais coder avant mon arrivée à l' EPITA et de ce fait, commencer un projet en équipe m'était totalement inconnu. De plus je n'ai pas fait l'option ISN en terminale et je n'avais donc jamais fait de projet. Grâce à celui-ci je me rends compte de l'organisation qu'il faut prendre ainsi que du temps de travail qui est conséquent.

Un projet tel qu'un jeu vidéo, dans notre exemple, prend énormément d'éléments en compte et de petits détails dont on ne se rend pas toujours compte. Ces petits détails prennent en réalité énormément de temps pendant la réalisation, on peut citer comme exemple le fait de rendre fonctionnel un menu ou même réaliser les graphismes d'un jeu.

De plus Unity était un logiciel complètement nouveau pour moi et partir de zéro comme ceci pour se lancer dans un projet paraît invraisemblable, on ne sait pas par où commencer. Il m'a donc fallu me renseigner sur internet pour pouvoir apprendre à utiliser Unity et quelles étaient les possibilités avec celui-ci. Par la suite, c'est avec la manipulation du logiciel que l'on progresse vraiment et à partir du moment où l'on rentre vraiment dans le logiciel qu'on s'y plonge réellement.

Comme je l'ai dit précédemment, il faut de l'organisation quand on se lance dans un projet comme celui-ci et ne pas partir dans tous les sens. Il faut donc savoir découper le travail en étapes, en différentes tâches.

Ainsi le fait d'avoir régulièrement dans le semestre à rendre un rapport de soutenance et en plus avoir une présentation de soutenance nous habitue à avoir à respecter des dates limites pour rendre des choses assez conséquentes, avec de la préparation en amont.

Ce projet de second semestre est donc très important pour moi par son but dans l'apprentissage mais également je suis fier de ce que mes amis et moi avons réalisé durant tous ces mois et de ce que nous avons accompli.

Conclusion

Nous allons faire un récapitulatif de tous les modules dont nous avons parlés dans notre rapport de projet.

Ayant mis un accent tout particulier sur les graphismes et le design, un élément cher à Alexandre, nous sommes heureux du côté visuel de ceux-ci, cela a été rendu possible en grande partie grâce à la palette de couleur flat design.

Pour ce qui est du site internet Jean-Baptiste a fourni des efforts conséquents dans la réalisation de celui-ci. La recherche constante d'esthétisme durant sa réalisation a permis au site de se démarquer. Cette recherche a également servi à rester à tout moment dans l'univers visuel du projet. On peut relever notamment le fait que les effets et les couleurs composant le site sont les mêmes que celles composant le jeu.

Pour ce qui est de la partie personnage et obstacles, ce n'était pas la partie la plus "technique" en terme de code. Malgré cela, celle ci nous a demandé un temps considérable en raison des mises en commun successives ainsi que des refontes consécutives de ces modules. La partie boutique et compte quant à elle représente la partie la plus complexe à implémenter pour deux raisons simples: l'assimilation du PHP mais également la fusion avec le jeu et le fichier Unity. Néanmoins nous avons réussi à l'implémenter et à stocker les skins correctement avec un script complémentaire sur Unity.

Ensuite, pour la partie interface, tout comme pour le site internet nous avons mis l'accent sur la concordance entre les couleurs, le design et les niveaux, de manière à ce que les menus et autres écrans ne viennent pas casser l'expérience de jeu .

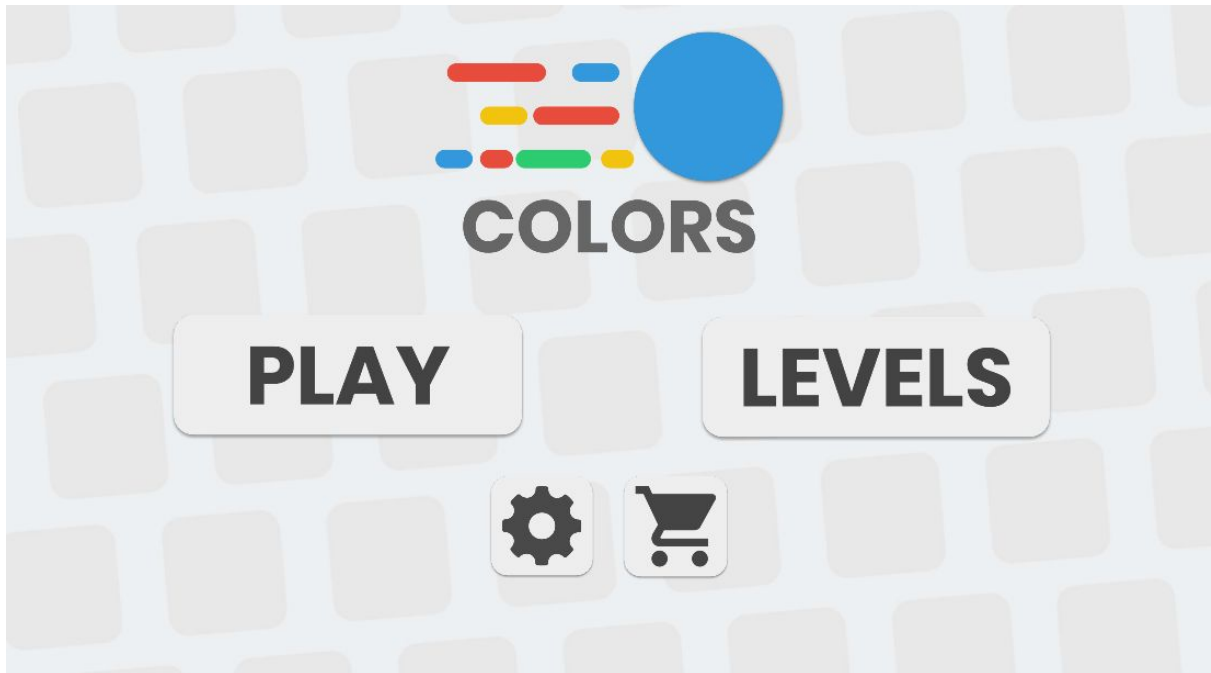
Pour terminer, les niveaux représentent l'aboutissement de notre travail. Ceux-ci ont représenté un certain nombre de difficultés dues, comme nous l'avons dit, à de nombreuses reprises dans nos attentes pour le résultat final de ces derniers.

Pour conclure nous sommes satisfaits du résultat de notre projet, compte tenu des attentes que nous nous étions fixés par rapport au résultat obtenu. Le projet est une source de connaissances très importante. De plus, le fait que celui-ci soit assez permissif quant au choix du sujet, nous permet de nous investir

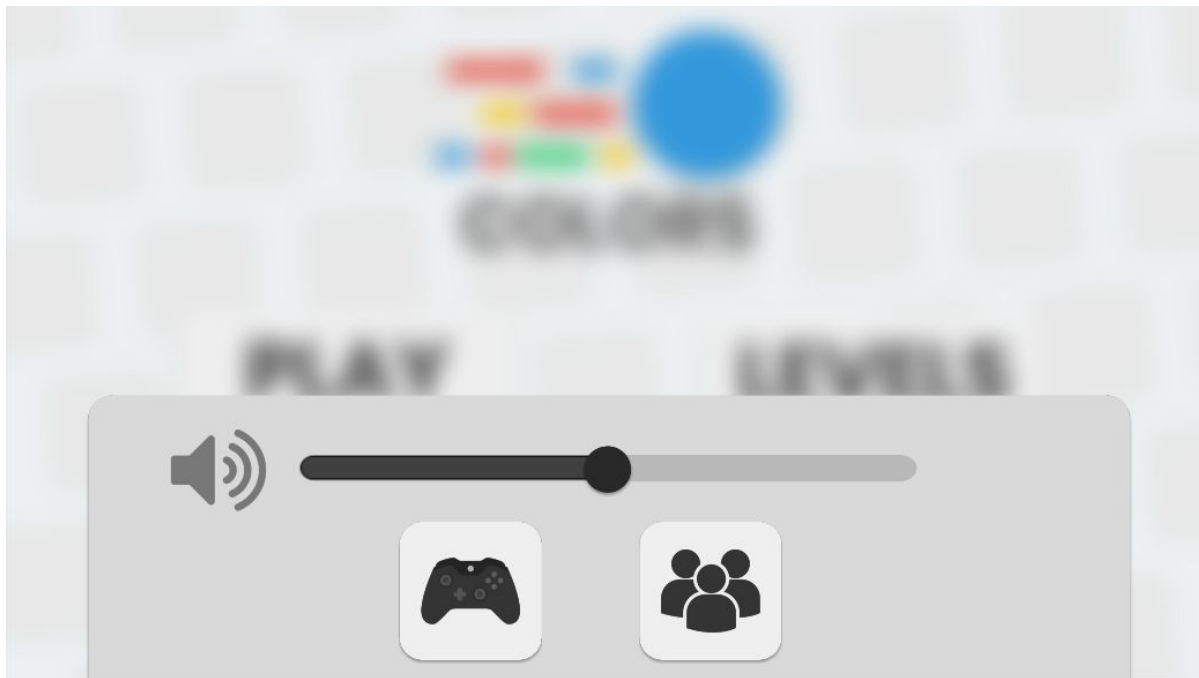
pleinement dans la réalisation de quelque chose qui nous plaît. Cela nous a donc permis de progresser et ou de perfectionner nos connaissances qui nous seront utiles dans la suite de notre scolarité.

ANNEXES

1.1 Graphismes/Design



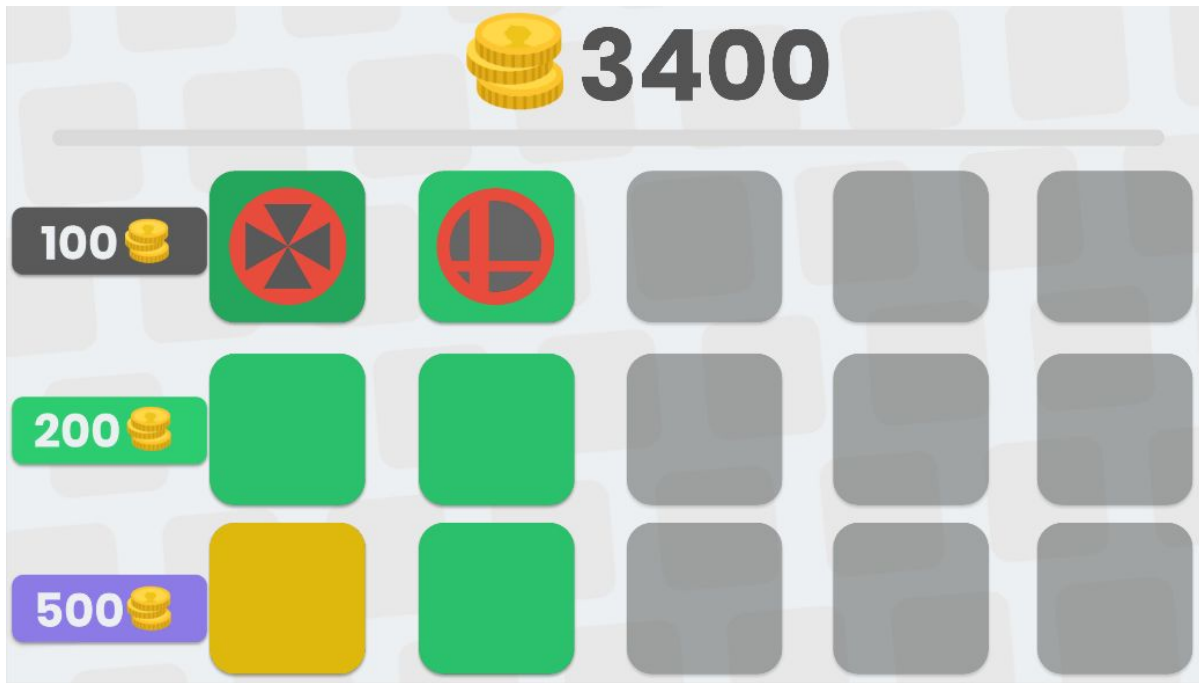
1. Menu principal du jeu



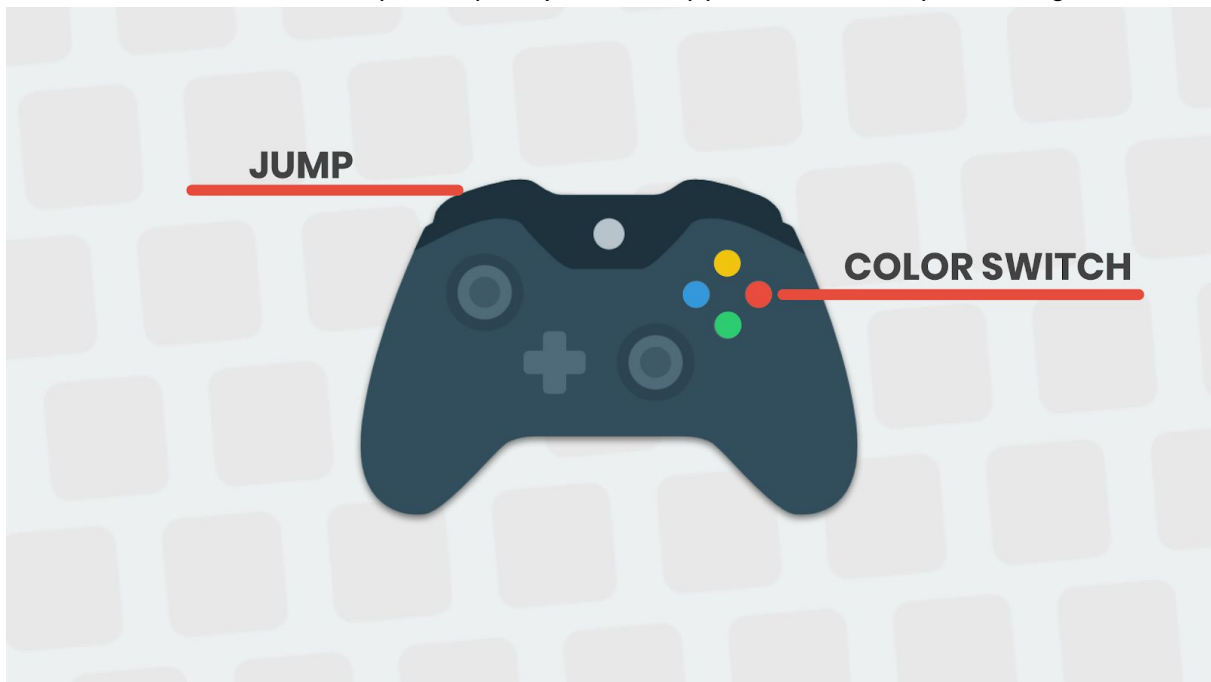
2. Menu des paramètres. Nous avons accès aux commandes et aux crédits



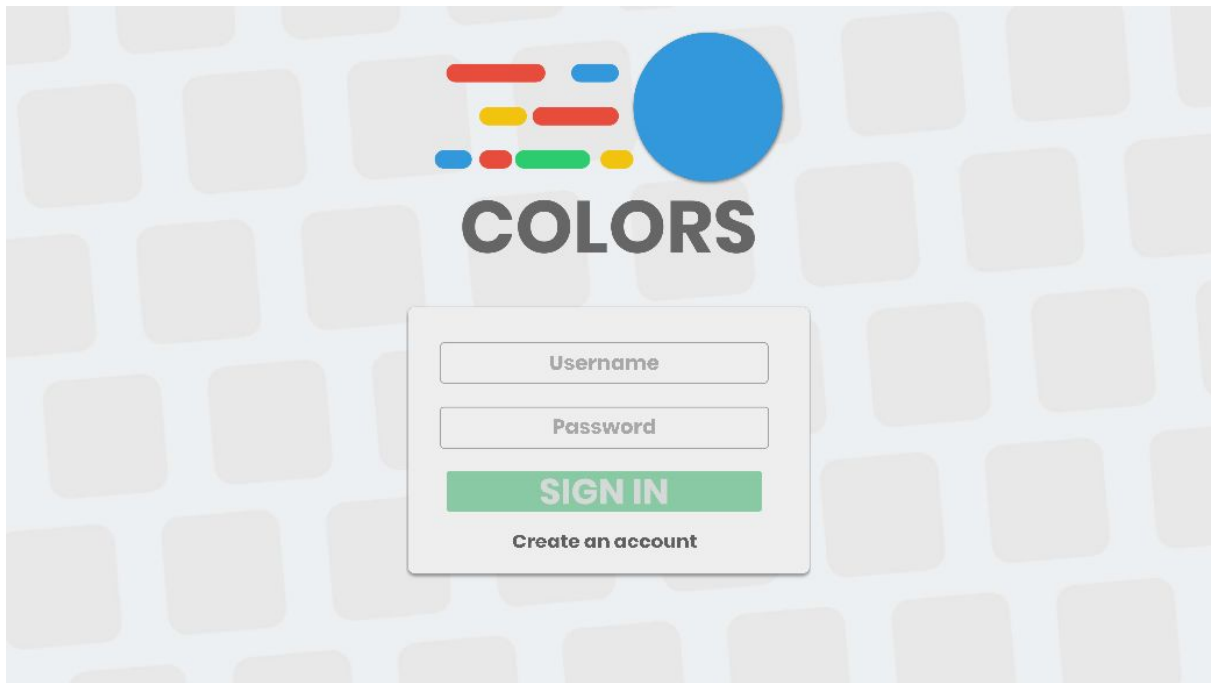
3. Menu des niveaux.



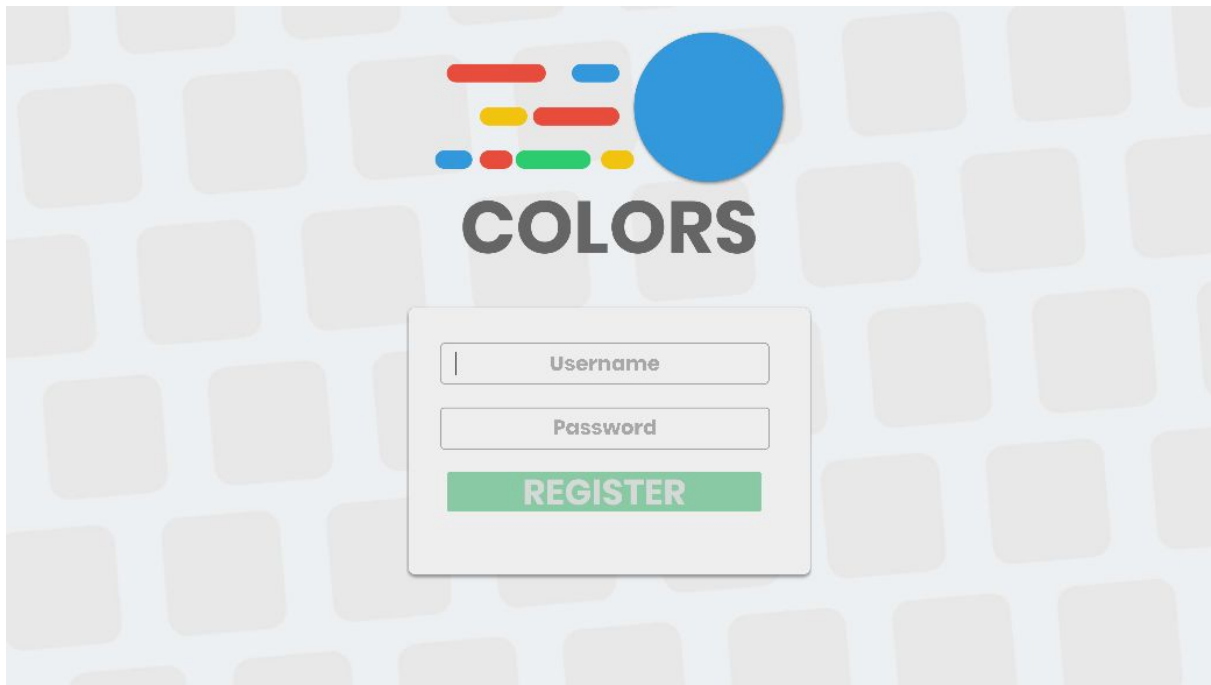
4. Menu de la boutique. On peut y choisir l'apparence de son personnage



5. Commandes du jeu.

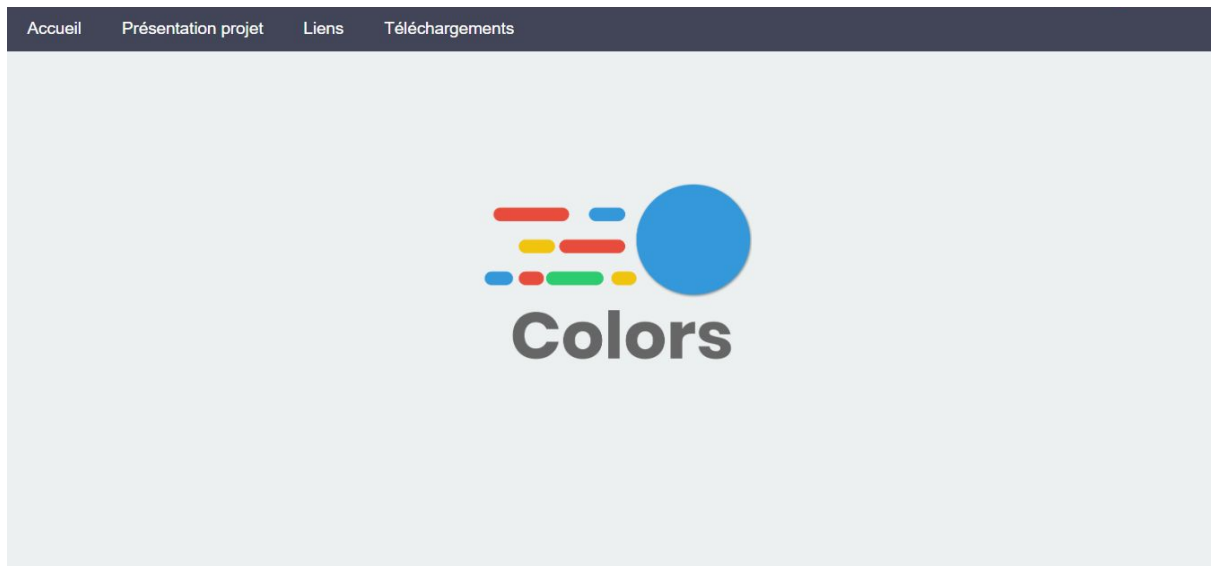


6. Menu de connexion.

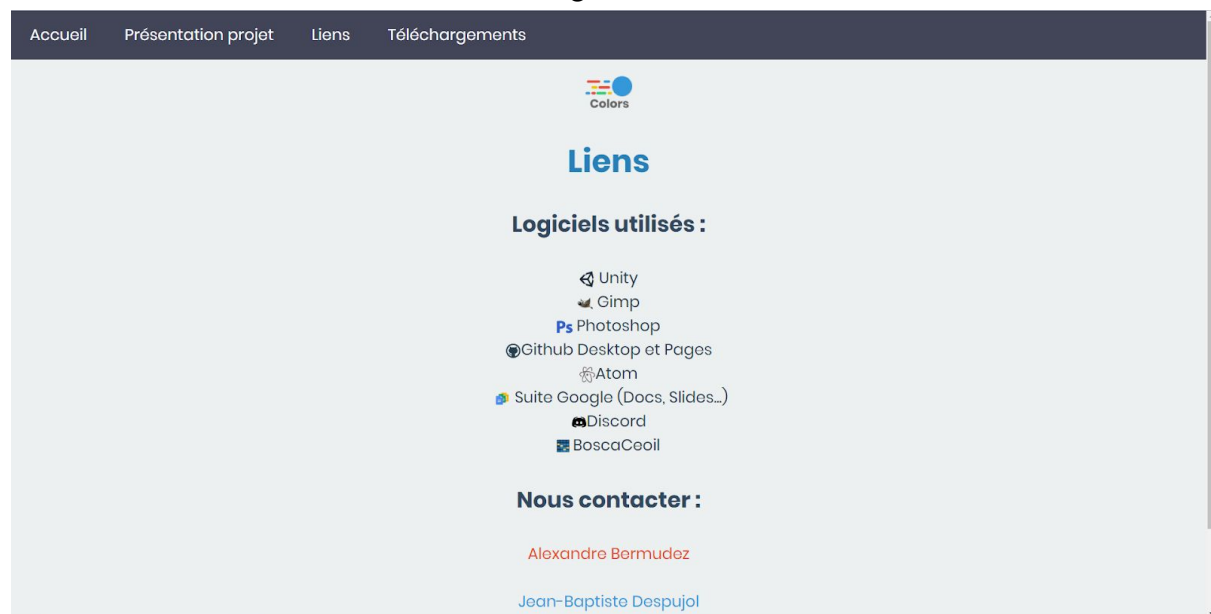


7. Menu de création de compte.

1.2 Site Internet




1. Onglet d'accueil.




2. Onglet des liens.

Accueil
Présentation projet
Liens
Téléchargements




Présentation du projet


Membres du groupe P1D




Alexandre Bermudez



Jean-Baptiste Despujol



Romain Grégoire



Alex Poiron

Problèmes rencontrés

Alexandre


Comprendre le PHP et MySQL n'a pas été chose aisée. Le système de "query" pour aller chercher les informations dans la base de données n'est pas simple au premier abord mais heureusement plutôt bien documenté. Implémenter exactement le design voulu dans Unity n'a pas été tâche facile non plus, Unity ne fonctionne pas comme un éditeur d'image traditionnel, les outils d'édition sont presque inexistant. De ce fait le design du jeu n'est pas exactement comme celui voulu. De plus un soucis avec le rendu final vient s'ajouter, certains éléments sont pixelisés pour certaines scènes de notre jeu, c'est un problème que nous devons régler au plus vite.

Jean-Baptiste

La création de la base de données nous a donné beaucoup de fil à retordre. Nous nous sommes entêtés à utiliser un Google Sheet alors que son implémentation semblait très complexe et sans fin. Mais nous étions certains que c'était le meilleur moyen de faire. Des bibliothèques étaient disponibles en C#, mais, même avec cette aide, nous ne comprenions pas du tout quelles étaient les étapes à réaliser pour l'accès à une telle base de données. Lorsque nous avons compris que nous nous obstinions pour rien, nous avons pris du recul et cherché les autres moyens d'héberger des données. Nous avons découvert les bases de données MySQL qui sont très bien documentées et faciles d'accès. C'est donc vers ce système de sauvegarde que nous nous sommes tournés. Grâce à elles nous avons réussi à stocker toutes nos données utilisateurs.

3. Onglet Présentation Projet

Accueil
Présentation projet
Liens
Téléchargements



Téléchargements

Cahier Des Charges

Rapport Soutenance 1

Rapport Soutenance 2

Rapport de projet final

Projet Final

Projet Final version lite

4. Onglet téléchargement

1.3 Personnage

```
public Rigidbody2D rb;
public string currentcolor;
public float vitesse = 5;
public SpriteRenderer sr;
public Color colorred;
public Color colorblue;
public Color coloryellow;
public Color colorgreen;
float restartTimer;
public AudioClip JumpSound;
private AudioSource audiosource;
private Transform mytransform;
public float haut = 200f;
```

1. Variables pour toutes nos fonctions pour le personnage

```
void Start()
{
    rb.velocity = new Vector2(vitesse * Time.deltaTime*2, 0);
    SetRandomColor();
    audiosource = GetComponent<AudioSource>();
    mytransform = GetComponent<Transform>();
}
```

2. Initialisations de nos variables dans la fonction Start

```
void SetRandomColor()
{
    Random index = new Random();
    int number = index.Next(0, 3);
    switch (number)
    {
        case 0:
            currentcolor = "Red";
            sr.color = colorred;
            gameObject.tag = "Red";
            break;
        case 1:
            currentcolor = "Yellow";
            sr.color = coloryellow;
            gameObject.tag = "Yellow";
            break;
        case 2:
            currentcolor = "Green";
            sr.color = colorgreen;
            gameObject.tag = "Green";
            break;
        case 3:
            currentcolor = "Blue";
            sr.color = colorblue;
            gameObject.tag = "Blue";
            break;
    }
}
```

3. Code qui nous permet de donner au personnage une couleur aléatoire dès le début du niveau.

```
void OnTriggerEnter2D(Collider2D other)
{
    if (!gameObject.CompareTag(other.tag) &&
!other.CompareTag("MoneyCoin"))
    {
        UnityEngine.Debug.Log("Game Over");
        SceneManager.LoadScene(0);
    }
}
```

4. Fonction qui détecte si le joueur détecte un obstacle d'une couleur différente.

```
public Transform player;
void Update()
{
    if (player.position.x > transform.position.x)
    {
        transform.position = new Vector3(player.position.x,0f,-10f);
    }
}
```

5. Fonction présente dans le script de la Caméra. Cela nous permet de faire suivre le joueur à la caméra puisque celui-ci avance continuellement

1.4 Obstacles

```
public float speed = 1f;
public Transform circle;

void Start()
{
    transform.Rotate(0f, 0f, speed*Time.deltaTime / 4);
}

void Update()
{
    transform.Rotate(0f, 0f, speed*Time.deltaTime / 4);
}
```

Fonction qui permet à certains obstacles de pouvoir tourner sur eux-mêmes

1.5 Boutique et Comptes

1.5.1

The screenshot shows the phpMyAdmin interface for a MySQL database. The 'Structure' tab is selected for a table named 'test'. The table structure is as follows:

Nom	Type	Taille/Valeurs	Valeur par défaut	Interclassement	Attributs	Null	Index	A_I	Commentaires	Déplacer une colonne
id	INT	10	Aucun(e)			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>		
username	VARCHAR	16	Aucun(e)			<input type="checkbox"/>	UNIQUE	<input type="checkbox"/>		
hash	VARCHAR	100	Aucun(e)			<input type="checkbox"/>	---	<input type="checkbox"/>		
salt	VARCHAR	50	Aucun(e)			<input type="checkbox"/>	---	<input type="checkbox"/>		
score	INT	10	Tel que défini : q			<input type="checkbox"/>	---	<input type="checkbox"/>		

Below the table structure, there are fields for 'Commentaires de table', 'Interclassement', and 'Moteur de stockage' (set to InnoDB). At the bottom right, there are buttons for 'Aperçu SQL' and 'Enregistrer'.

1. Page de création d'une base de données MySQL

1.5.2

```
<?php
    $con = mysqli_connect('localhost', 'root', 'root', 'unityaccess');
    //premier paramètre va être remplacé par l'url de l'hôte de la base de
    données

    //vérification que la connexion existe
    if(mysqli_connect_errno())
    {
        echo "1: conection failed"; //code d'erreur #1 = connexion ratée
        exit();
    }

    $username = $_POST["name"];
    $password = $_POST["password"];

    //vérification que le nom existe
    $namecheckquery = "SELECT username FROM players WHERE username='" .
$username . "'";

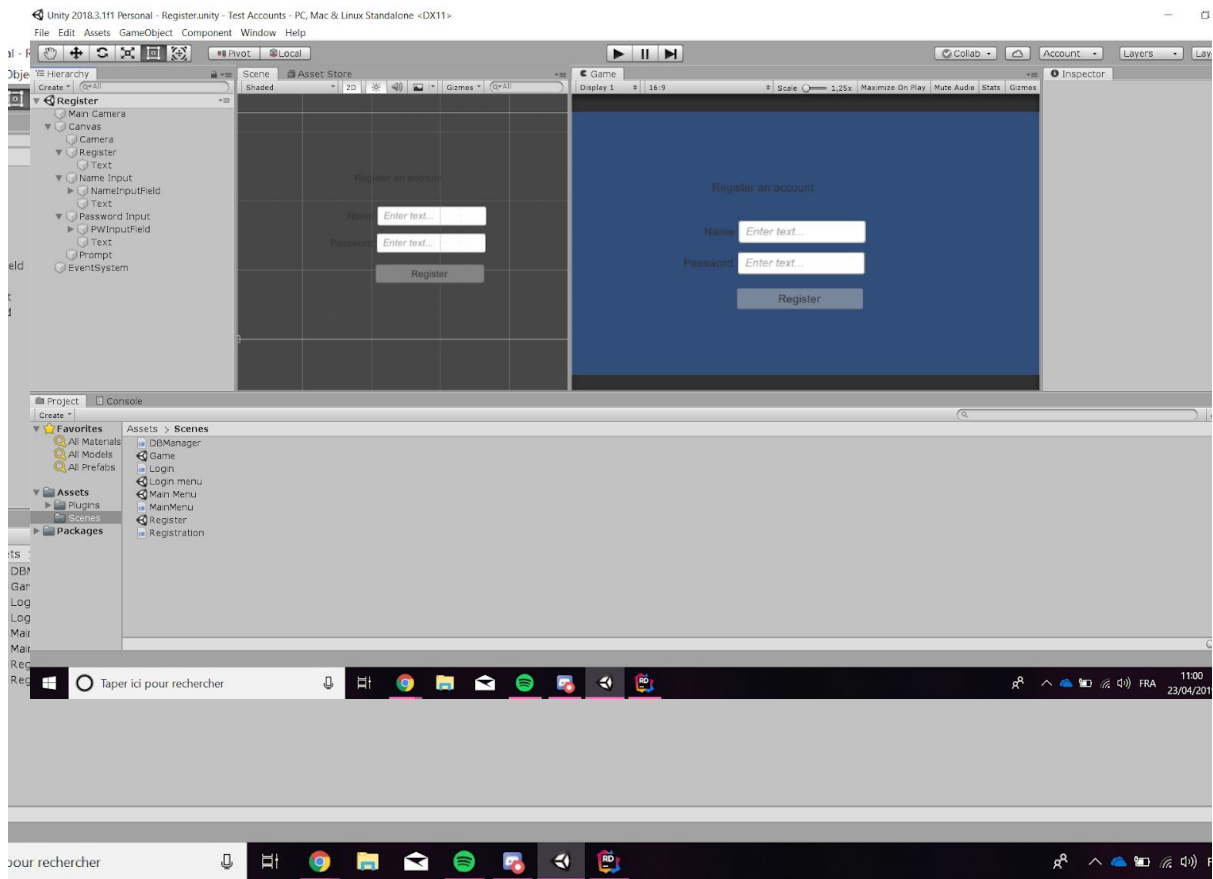
    $namecheck = mysqli_query($con, $namecheckquery) or die("2: Name
check query failed"); //code d'erreur #2 - requête de vérification du
nom ratée

    if(mysqli_num_rows($namecheck) > 0)
    {
        echo "3: Name already exists"; //code d'erreur #3 - le nom
existe, impossible de s'enregistrer
        exit();
    }
    //ajout de l'utilisateur a notre base de donnée
    $salt = "\$5\$rounds=5000\$" . "culasse" . $username . "\$";
    $hash = crypt($password, $salt);
    $insertuserquery = "INSERT INTO players (username, hash, salt)
VALUES ('" . $username . "', '" . $hash . "', '" . $salt . "')";
    mysqli_query($con, $insertuserquery) or die("4: Insert player query
failed"); //code d'erreur #4 - requete d'insertion ratée

    echo("0");
?>
```

2. Fichier PHP permettant de créer de nouveaux comptes

1.5.3



3. Scène UNITY de tests de création de comptes

1.5.4

```
<?php
$con = mysqli_connect('localhost', 'root', 'root',
'unityaccess'); //premier paramètre va être remplacé par l'url de la
base de donnée
//vérification que la connexion existe
if(mysqli_connect_errno())
{
    echo "1: conection failed"; //code d'erreur #1 = connexion
ratée
    exit();
}
$username = $_POST["name"];
$password = $_POST["password"];
//vérification que le name existe
```

```

$namecheckquery = "SELECT username, salt, hash, score FROM
players WHERE username='" . $username . "'";

$namecheck = mysqli_query($con, $namecheckquery) or die("2: Name
check query failed"); //code d'erreur #2 - requête de vérification du
nom ratée
if(mysqli_num_rows($namecheck)!=1)
{
    echo "5: Either no user with name or more than one"; //
code d'erreur #5 - nombre de names != 1
    exit();
}
//récupération des infos de login
$existinginfo = mysqli_fetch_assoc($namecheck);
$salt = $existinginfo["salt"];
$hash = $existinginfo["hash"]

$loginhash = crypt($password, $salt);
if($hash != $loginhash)
{
    echo "6: Incorrect password"; //code d'erreur #6 - le
password qui a été hash ne correspond pas à celui de la table de la
base de données
    exit();
}
echo "0\t" . $existinginfo["score"];
?>

```

4. Fichier PHP permettant de se connecter aux comptes existants

1.6 Interface



1. Bouton rendu sur Unity

```
private static BGsoundcontinue instance = null;
public static BGsoundcontinue Instance {
    get { return instance; }
}
void Awake() {
    if (instance != null && instance != this) {
        Destroy(this.gameObject);
        return;
    } else {
        instance = this;
    }
    DontDestroyOnLoad(this.gameObject);
}
```

2. Script permettant de garder la musique entre les scènes

```
Button[] allbut = {skin1, skin2, skin3, skin4, skin5, skin6, skin7,
skin8, skin9, skin10, skin11, skin12, skin13, skin14, skin15};
    for (int i = 0; i < 15; i++)
    {
        if (DBManager.skins[i] == "-1")
        {
            Button lebutton = allbut[i];
            lebutton.interactable = false;
            Button buthaut;
            if (i >= 5)
            {
                buthaut = allbut[i-5];
            }
            else
            {
                buthaut = null;
            }

            Button butbas;
            if (i <= 9)
            {
                butbas = allbut[i + 5];
            }
            else
            {
                butbas = null;
            }

            Button butdroit;
            if (i != 4 && i != 9 && i != 14)
            {
                butdroit = allbut[i + 1];
            }
            else
            {
                butdroit = null;
            }

            Button butgauche;
            if (i != 0 && i != 5 && i != 10)
            {
                butgauche = allbut[i - 1];
            }
        }
    }
}
```

```

else
{
    butgauche = null;
}
if (buthaut != null)
{
    Navigation nav = buthaut.navigation;
    nav.mode = Navigation.Mode.Explicit;
    nav.selectOnDown = null;
    buthaut.navigation = nav;
}
if (butbas != null)
{
    Navigation nav = butbas.navigation;
    nav.mode = Navigation.Mode.Explicit;
    nav.selectOnUp = null;
    butbas.navigation = nav;
}
if (butdroit != null)
{
    Navigation nav = butdroit.navigation;
    nav.mode = Navigation.Mode.Explicit;
    nav.selectOnLeft = null;
    butdroit.navigation = nav;
}
if (butgauche != null)
{
    Navigation nav = butgauche.navigation;
    nav.mode = Navigation.Mode.Explicit;
    nav.selectOnRight = null;
    butgauche.navigation = nav;
}

```

3. Script permettant la désactivation de l'accès des boutons désactivés

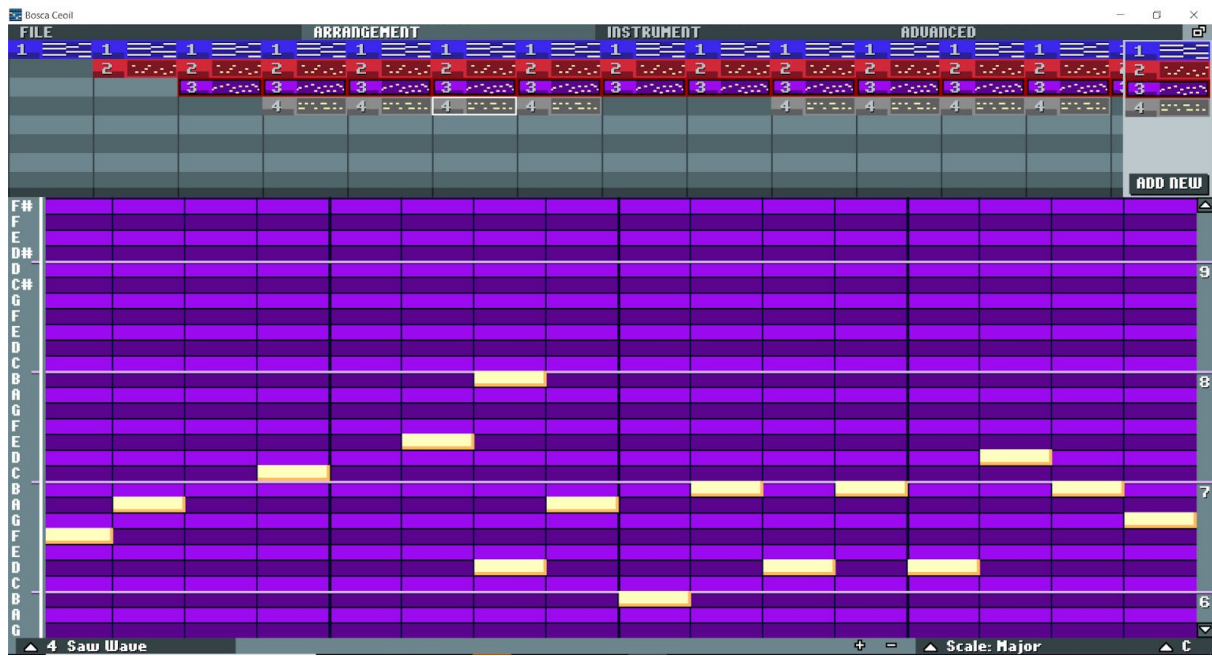
1.7 Niveaux

```
public Collider2D coin;
public int val = 1;
private GameObject UI;

// Start is called before the first frame update
void Start()
{
    UI = GameObject.FindGameObjectWithTag("CoinAmount");
}
void OnTriggerEnter2D(Collider2D other)
{
    if (other.IsTouching(coin) &&
        (other.CompareTag("Player") || other.CompareTag("Blue") || other.CompareTag(
"Green") || other.CompareTag("Red") || other.CompareTag("Yellow") ))
    {
        int coinUi = int.Parse(UI.GetComponent<Text>().text) + val;
        UI.GetComponent<Text>().text = coinUi + "";
        Destroy(gameObject);
    }
}
```

-
1. Fonction qui détruit les pièces quand le joueur est en collision avec elles et par conséquent actualise le compteur de pièces.

1.8 Récits de réalisation



Edition du fichier audio du menu principal du jeu dans BoscaCeoil