Dossier Projet Informatique
SUP–2022

PEnuts
Rapport De Projet
Par CAPZ

Alexandre POIRIER-COUTANSAIS
Cloé LACOMBE
Ziane LAYADI
Pierrick MADE

Vendredi 25 mai, 2018

Version française

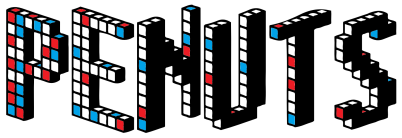


Table des matières

1 Type et origine du projet **3**
1.1 Concept du projet . . . 3
1.2 Origine du projet . . . 6
1.3 Objet de l'étude . . . 6
1.4 État de l'art . . . 7

2 Présentation de CAPZ **8**

3 Parties du projet **9**
3.1 Répartition des tâches . . . 9
3.2 Programmation . . . 10
3.2.1 Mouvements du Joueur . . . 10
3.2.2 Actions des joueurs . . . 11
3.2.3 Boutons & Mécanismes . . . 13
3.2.4 Ennemis & IA . . . 16
3.2.5 Camera et Lumières . . . 18
3.2.6 Vision . . . 19
3.2.7 Collisions . . . 20
3.2.8 Création de niveaux . . . 21
3.2.9 Multijoueur . . . 26
3.2.10 Menu & Options . . . 28
3.3 Design . . . 33
3.3.1 Codes Couleurs . . . 33
3.3.2 Graphismes . . . 34
3.3.3 Musiques . . . 36
3.3.4 Effets sonores . . . 37
3.3.5 Animations . . . 38
3.4 Management . . . 39
3.4.1 Site Internet . . . 39
3.4.2 Aspect Économique . . . 43
3.4.3 Ressources . . . 44
3.5 Progression . . . 45

4 Conclusion **46**
4.1 Conclusions Personnelles . . . 46
4.2 Conclusion General . . . 49

Introduction

Cette année, étant des étudiants de premières année à Epita, nous devions réaliser un projet informatique de six mois par groupes de quatre. Ce projet nous permet de mettre en pratique les connaissances acquises durant les cours, les TD et les TP. Il nous permet également d'améliorer nos compétences personnelles, en particulier celles ne pouvant pas être misent en pratique pendant les cours.

Notre groupe se nome CAPZ et nous avons travaillé sur un jeux multijoueur. Ce jeux est PEnuts et son concept principal est d'évoluer dans un jeux de puzzle reposant sur la complémentarité entre les joueurs. Cette complémentarité est principalement basée sur la différence entre les actions et la vision des personnages. En effet, dans la même salle, les joueurs ne sont pas capable de voir les mêmes choses ou d'interagir avec les mêmes objets. Comme un jeux de puzzle, l'objectif des joueurs est de terminer une salle pour avoir accès aux niveaux suivants. Dans ce sens, les joueurs ne pourront pas évoluer en solo, ils auront besoin d'améliorer leurs compétences en communication pour résoudre le puzzle ensemble.

Ce concept de projet ne vient pas d'un jeux en particulier. En effet, beaucoup d'aspects de notre jeux peuvent être retrouvés dans d'autres, mais nous ne voulions pas être basé sur un jeux précis. Nous voulions trouver un nouveaux game-play impliquant de la coopération et de la communication. Autour de ce point nous avons développé le concept de la complémentarité et de la différence de vision. Ce que nous avons implémenté dans notre jeux.

Ce rapport de projet vous présentera en détails la progression réalisée depuis le début du projet. Une première section donnera des informations détaillés sur le concept du jeux et sur le projet dans sa globalité. Ensuite, une deuxième partie abordera la progression et l'organisation du projet suivi par tout les détails pour chacune des tâches. En d'autres termes, nous regarderons la distribution des tâches, leur progression au fil du temps et leurs détails. Cette section essaiera de répondre aux questions suivantes : Quelles étaient les attentes dans cette partie du projet, qu'a-t-il été achevé finalement ? Comment cette partie a-t-elle évolué durant le projet et comment cela a-t-il été fait ? Finalement, la dernière section sera une conclusion pour ce projet, elle contiendra une conclusion personnelle pour chaque membre du groupe et une conclusion général.

1 Type et origine du projet

1.1 Concept du projet

Comme expliqué dans l'introduction, le principal concept de notre jeux est la complémentarité et la communication entre deux joueurs pour résoudre un puzzle.Pour donner une meilleure idée de ce à quoi ressemble le jeu, nous allons commencer par un exemple. Dans cet exemple, nous pouvons voir la vision du niveau par chaque joueur.

Voici les détails pour pouvoir comprendre les croquis de la page suivante (Je vous pris de nous excuser pour la non-traduction de la légende ci-dessous).


This block is movable, by being pushed by a player, a player cannot step on nor can he cross the tile the block is on.

This is a trigger bounded to the signal 2, the signal 2 is active (true) as long as a player or a movable block is on top, otherwise the signal 2 is inactive (false).


This is a void tile, a player cannot cross nor step on it.

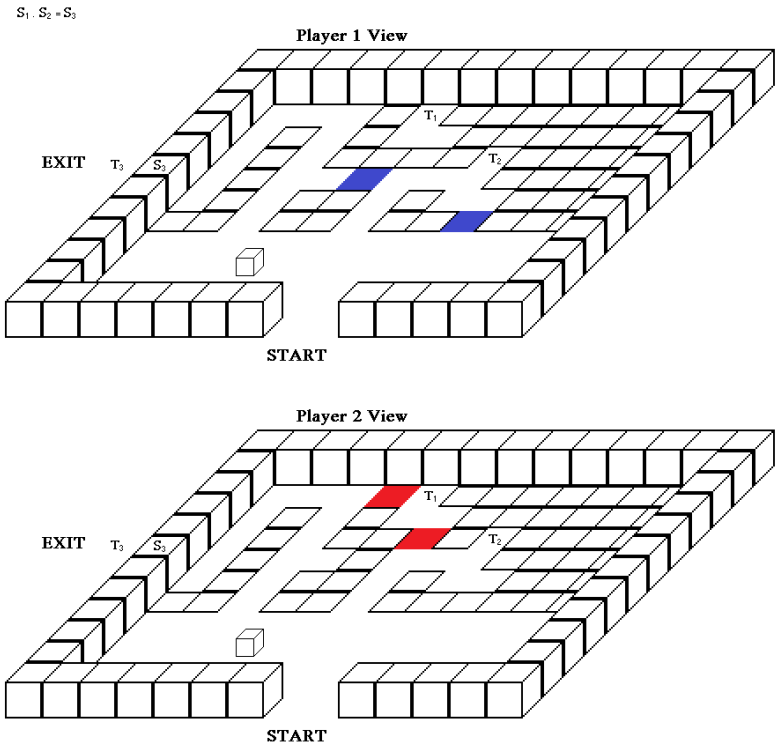

This is a block tile, a player cannot cross nor step on it.

This block tile is bounded to a signal. If the signal 3 is active (true) the block disapear and act as a standard floor tile. If the signal is inactive (false) the block act as a standard block tile.

This is a simple logic operation, here the signal 3 is active (true) if and only if both signal 1 and 2 are active (true).

The colors here represent tiles which are only seen as a standard floor tile by the player seeing the color, for the other player, it acts as an obstacle (either a block tile or a void tile).

In this case the player 2 sees the blue tiles as void tiles, however the player 1 sees them as blue floor tiles.



Ce niveau a été le premier concept pour notre jeu, les joueurs doivent atteindre le point final ensemble. Ici, le bloc Bleu n'est visible que par le joueur Bleu, il est aussi le seul à pouvoir le traverser, le bloc Rouge fonctionne de la même manière pour le joueur Rouge. Notez que le bloc mobile, représenté par le petit bloc, peut traverser les blocs rouge et bleu. Un joueur ne peut pas traverser là où se trouve le bloc mobile, cependant dans l'implémentation du bloc mobile dans notre jeu, c'est un peu différent, le joueur peut en théorie pousser le bloc sur le côté et le contourner, mais c'est beaucoup plus difficile à faire et tous les niveaux pourront être réalisés sans traverser un endroit où le bloc mobile est activé. Ici, les deux joueurs doivent s'appuyer l'un sur l'autre pour pousser le bloc afin de pouvoir le pousser sur le bon déclencheur. Quand un déclencheur est activé par un joueur ou un bloc, il active un signal, ici une porte 'ET' est présente, la porte ne s'ouvrira que lorsque les deux déclencheurs seront activés, le dernier déclencheur, déclencheur 3, permettra au joueur qui a franchit la porte d'ouvrir la porte au prochain joueur qui gardait la porte ouverte de l'intérieur.

Dans l'objectif de donner une meilleur idée des principes de notre jeux, nous avons décidé de répondre à des questions qui sont souvent posées dans un jeux-vidéo.

Quel est le principal concept de notre jeux, sa spécificité ?

Nous voulions créer un jeux de puzzle principalement joué en coopération entre deux joueurs. Ensuite, la spécificité de celui-ci est la complémentarité entre les différentes visions et actions des joueurs.

Quel est l'objectif des joueurs ? Comment peuvent-ils gagner ?

Le but Ultime des joueurs est évidemment de terminer le jeux en franchissant toutes les salles. À une plus petite échelle, son objectif est de finir une salle pour avancer dans le jeux.
Cependant, pour nous, un objectif simple et important de ce jeux est la satisfaction de résoudre un puzzle en communicant avec un partenaire.

Comment peuvent-ils perdre ?

Il y a des objets capables de tuer les joueurs, les trous et les ennemis. Cependant, étant dans un jeux de puzzle, les joueurs perdent si ils ne réussissent pas à résoudre le niveau.

Dans quel environnement les joueurs vont-ils évoluer ? Quel est le type de salle ?

Les joueurs évoluent dans des salles avec des mécanismes et des obstacles. Ce sera un monde cubique avec des détails vraiment simple.

Quelles sont leurs actions et leurs mouvements ?

Les joueurs ont le droit de bouger dans quatre directions pour se déplacer dans la pièce. Ils sont aussi capable d'interagir avec des objets comme des ennemis ou des boutons activant des portes. Les joueurs peuvent prendre des dommages si un ennemie les touches. La restriction concernant les ennemis se fait au niveau de la vision.. En effet, un des joueur ne peut voir ces ennemis seulement si l'autre joueur les tag avec son arme. Ils auront donc des armes spécial en fonction de leur couleur.

1.2 Origine du projet

Quand nous avons commencé à parlé du jeux nous avons essayé de définir ce que chacun de nous voulait exactement. En d'autres termes, quel type de jeux nous voulions, quel type d'actions pour le joueur, quel type d'environnement et ainsi de suite.

Par la suite, nous avons compris que beaucoup de bon jeux ont uniquement une caractéristique physique qui change (le temps, l'espace, la gravité,...). Ces jeux sont simple à prendre en main, mais compliqué à maîtrisé parfaitement. C'était un point que nous voulions mettre en avant. Finalement, en cherchant des idées originales et des concept à implémenter nous avons commencé à approcher l'idée de jouer avec la vision respective des joueurs. Finalement, en développant nos idées et nos exemples, nous avons terminé sur ce type de projet.

1.3 Objet de l'étude

Ce projet nous a permis de nous améliorer sur beaucoup de points. En groupe ou individuellement, il nous apporté beaucoup d'outils indispensables. C'est six mois de projet ont été pour nous une première expérience. C'était donc entièrement gratifiant.

Voici les principales compétences que ce projet nous a apporté :

- Travail de groupe :
  - Management
  - collaboration & coopération
  - Gestion du planning
  - Organisation d'un projet
  - GitHub utilisé pour coder en collaboration
- Connaissances techniques :
  - Découverte de Unity
  - C# utilisé pour les scripts dans Unity
  - LaTex utilisé pour les rapports
  - HTML et CSS pour le site internet
  - Mise en avant de l'apprentissage autonome
  - Et bien plus encore, pour plus d'infos les conclusions personnelles sont disponibles page 46.

1.4 État de l'art

Pour penser notre jeu, nous avons évidemment eu une influence de tous les jeux auxquels nous avons joué dans nos vies, mais certains sont plus pertinents. Ces exemples sont les plus pertinents par rapport à notre projet. Nous avons donc utilisé les points forts de ces jeux pour améliorer notre projet.

L'aspect principal de *Portal 2* est le même que celui de notre projet, c'est un jeu de puzzle. *Portal 2* est très populaire et est reconnu comme le meilleur jeu PC de 2011 selon Metacritic. Le joueur doit utiliser son esprit pour traverser différentes pièces à l'aide du 'portal gun' lui permettant de créer 2 portails dans lesquels il peut voyager. Ici, un aspect du monde réel est changé et le joueur doit s'adapter et se familiariser avec ce changement. Le mode coopération permet à deux joueurs de coopérer pour résoudre des énigmes plus difficiles, exigeant que les deux joueurs utilisent leurs portails pour traverser les pièces. La force du mode coopération est qu'il combine à la fois les puzzles et l'interaction sociale entre les joueurs pour créer une expérience de jeu unique.

La série *Zelda* est aussi un jeu de résolution de labyrinthes où le joueur doit souvent compléter différents puzzles dans les temples. Cette série est connue par presque tous les joueurs comme l'un des piliers des jeux vidéo. Dans ce jeu, le joueur doit combiner différentes mécaniques qu'il acquiert à travers le jeu pour résoudre des énigmes. De plus, il doit souvent sortir des sentiers battus pour résoudre les énigmes. Sa force repose sur des puzzles de tout niveaux (simples à difficiles) ce qui oblige le joueur à utiliser son esprit.

Keep talking and nobody explode est un jeu de coopération où un joueur doit désamorcer une bombe alors que les autres joueurs doivent lire un manuel pour lui donner des instructions. Ce jeu est populaire parmi de nombreux groupes de joueurs car il implique toujours une forte coopération entre les joueurs. Le manuel dans ce jeu est vraiment long et exhaustif et oblige les autres joueurs à le lire et à donner des instructions au désamorceur car il ne peut pas trouver dans un délai raisonnable toutes les spécifications pour chaque module dont la bombe est composée. Chaque module a toujours une et une seule combinaison d'actions qui entraîne la désactivation du module. La bombe est souvent composée de plusieurs modules qui permettent au désamorceur de donner une tâche différente aux différents joueurs, nécessitant des compétences de coordination. Force : Coopération provenant d'un jeu simple.

2 Présentation de CAPZ

Cette année, en tant qu'étudiants de première année à Epita, nous avons dû faire un projet informatique de six mois en groupes de quatre. CAPZ a été créé dans ces circonstances. L'autre point qui a aidé la création du groupe était la motivation. En effet, nous sommes un groupe d'étudiants motivés, notamment en informatique et en fait, dans ce que nous aimons. Ce qui signifie qu'un projet choisi devrait être un point de motivation pour nous tous. Enfin nous avons pensé que nous avions le même point de vue sur les jeux vidéo, donc nous avions les mêmes attentes pour ce projet. Toutes ces circonstances nous ont motivés à créer ce groupe.

En réalité, ce n'était pas si facile. En effet, les groupes de projet n'ont pas été créés facilement. Chaque étudiant devait penser aux personnes avec qui il voulait être pour une année complète de travail sur un projet... Nous étions donc tous à la recherche du bon groupe, celui correspondant à nos attentes. La création de notre groupe a été faite très rapidement. Le fait que Pierrick, Alexandre et Cloé soient dans la même classe a aidé à sa création. En effet, à l'exception de Ziane qui n'était pas dans la classe internationale, les autres se connaissaient. C'est finalement Cloé qui a présenté Ziane à Pierrick et Alexandre. Nous étions tous bons en programmation, nous étions tous motivés et chacun d'entre nous avait quelque chose à ajouter à un projet comme celui-là. C'était la création du groupe.

Cloé
Une imagination débordante et de l'énergie à partager

Alexandre
Motivation dans tout ce qui touche à la logique et à l'informatique

Pierrick
Organisation et calme dans n'importe quelle situation

Ziane
Son rêve, devenir un développeur de jeux-vidéos.

3 Parties du projet

Pour pouvoir organiser un projet et respecter les délais, la division des tâches ne pouvait être négligée. Par conséquent, cette section présentera en détail les différentes tâches de notre projet, leur distribution et leur progression dans le temps.

3.1 Répartition des tâches

Pour un projet de cette envergure - 6 mois - la division du travail était vraiment importante. En effet, sans la faire, tout le monde travaillerait sur la même chose, et le projet n'aurait pas pu évoluer comme ça. Cette distribution a été faite au début du projet en tenant compte de nos souhaits et de nos compétences. Cependant, le problème que nous avons rencontré à ce moment était notre connaissance des projets de groupe et nos compétences dans Unity. En effet, cela signifiait que notre distribution ne pouvait pas être parfaite. De ce fait, c'est pourquoi cette distribution a évolué au cours du projet en fonction de nos compétences et des nécessités du moment. Ce tableau est là pour vous présenter la répartition que nous avons utilisée.

Tâches	**Membre en charge**	**Son suppléant**
Mouvements du Joueur	Ziane	Alexandre
Actions du Joueur	Ziane	Cloé
Boutons et Mécanismes	Alexandre	Cloé
Camera et Lumières	Ziane	Pierrick
Vision du joueur	Ziane	Pierrick
Collisions	Alexandre	Cloé
Code Couleur	Alexandre	Pierrick
Graphismes	Cloé	Ziane
Animations	Alexandre	Pierrick
Conception de Niveaux	Cloé	Alexandre
Audio	Cloé	Pierrick
Site internet	Pierrick	All
Administration	Pierrick	Alexandre
Multijoueur	Ziane	Pierrick
Réseaux	Pierrick	Ziane
Menu et Options	Pierrick	Cloé

3.2 Programmation

3.2.1 Mouvements du Joueur

Dans cette partie, les mouvements du joueur ont été implémentés. Par conséquent, le but de cette partie était de lier les recherches sur le jeu que nous voulions créer à quelque chose de plaisant pour le joueur. Nous devions créer des contrôles permettant au joueur de se déplacer dans les pièces en tenant compte de l'environnement voulu. Finalement nous avons décidé que le meilleur moyen était de lui permettre d'avoir seulement quatre directions.

La mise en œuvre de cette fonctionnalité a été faite très rapidement car le joueur est un point clé dans notre jeu. À la première présentation tous les mouvements du joueur étaient fait.

De petits changements ont été faits au fil du temps pour donner une meilleure expérience au joueur. En effet, des sessions de test ont été faites pour essayer de jouer avec ces mouvements. Nous avons finalement décidé que ce sont les mouvements les plus appropriés dans notre type de jeu. Ceux-ci sont difficiles à comprendre au début mais sont efficaces à la fin.

Implémentation en jeu :

- Mouvements dans quatre directions
  - flèches-[gauche/droite] : rotation y-Axe de ± 90 °
  - flèches-[haut/bas] : en mouvement [avant/arrière]
- Un saut est disponible pour nous si nous en avons besoin.

Recherches :

Pour faire ces mouvements, des recherches ont été faites sur le mouvement d'un objet 3D et sur la simulation de la gravité dans un espace 3D. Ces recherches au début du projet nous ont aidés à comprendre le fonctionnement des corps rigides. Combiné avec ces recherches, nous devions également comprendre comment définir la position d'un objet et comment lui donner un vecteur de mouvement. Enfin pour donner ces mouvements spéciaux, nous avons utilisé la fonction de rotation. Le mouvement étant disponible, la dernière étape était de lier nos mouvements aux touches du clavier. Par conséquent, il y eu des recherches pour découvrir comment implémenter un menu de choix des contrôles. En effet, le but était de permettre au joueur de choisir ses propres touches de mouvements / actions.

3.2.2 Actions des joueurs

Le but de cette partie était de créer toutes les actions des joueurs dans le jeu. Notre joueur est autorisé à interagir avec certains objets. En jeu, le joueur peut saisir / pousser un cube, il peut aussi appuyer sur des boutons, tirer sur des ennemis, marcher sur les blocs de ses couleurs ou traverser le portail d'arrivée.

Cette partie est principalement liée au jeu et aux opportunités de puzzle que nous voulions ajouter. En effet, la mise en œuvre de nouvelles choses dans cette partie a été faite à mesure que la création des niveaux évoluait. La partie sur la création de niveaux peut être vue à la page 21. De plus, certaines fonctionnalités étaient nécessaires, comme les portails d'arrivée par exemple.

Implémenté :

- Blocs mobiles : Le joueur est capable de pousser quelques blocs. Ce sont des blocs spéciaux permettant au joueur de les mettre sur des boutons à bascule par exemple.
- Blocs relatifs à la vision : Les joueurs peuvent marcher sur certains blocs disponibles uniquement pour eux, ce sont des blocs de couleur. La description peut être trouvée dans la partie de la vision ci-dessous à la page 19.
- Tires : Un joueur est capable de tirer des balles de tag pour marquer les ennemis, l'autre joueur peut seulement tuer ces ennemis quand ils sont marqués. Pour plus de détails sur les ennemis, vous pouvez aller à la page 16.
- Portails d'arrivée : Pour finir une pièce, le joueur doit traverser un portail positionné à la fin du niveau.

Blocs mobiles :

Nous avions besoin d'un bloc que le joueur puisse déplacer pour résoudre certains casse-tête. Ce bloc devait se déplacer dans la même direction que le joueur et ne pas aller dans des directions aléatoires. Cette première restriction a été faite pour la première présentation, mais le bloc n'a pas très bien fonctionné. En effet, parfois même si le joueur le poussait, le bloc ne bougeait pas et c'était un problème sérieux car sans ces blocs fonctionnant parfaitement, le puzzle ne pouvait être résolu. L'objectif était de résoudre ce problème pour la deuxième présentation. À ce moment, nous avons compris

que nous devions lever le bloc tout en le poussant pour éviter un problème de collisionneur entre le bloc et le sol. Nous avons donc modifié le script du bloc. Maintenant, quand le joueur touche le bloc, le bloc se soulève un peu et son axe 'y' est bloqué pour qu'il ne tombe pas. Cet axe n'est libéré que lorsque le joueur arrête de pousser le bloc. En d'autres termes, lorsque le joueur touche le bloc, le bloc "vole" comme si le joueur l'attrapait, puis quand le joueur recule, le bloc revient à une gravité normale. Après quelques modifications supplémentaires sur le multijoueur, le bloc était presque fini. Le dernier problème que nous avions avec ce bloc était qu'en tombant, si le trou dans le sol avait la taille d'un cube, le bloc était bloqué en train de tomber. Donc, la seule chose que nous avons fait était de bloquer l'axe des rotations pour faire en sorte que le bloc ne puisse pas tomber en rotation et qu'il ne tombe que directement sans être bloqué. Finalement, après toute cette évolution, notre bloc est complètement fonctionnel.

Tires :

Pour créer ces balles et ces armes, nous avons principalement utilisé des colliders et des Mesh-filter permettant de toucher ou de faire disparaître l'ennemi. Pour initialiser une balle, nous avons créé un gameobject qui permet le spawn d'une balle puis nous lui avons appliqué un vecteur spécial. Ce gameobject placé devant le joueur lui permet de tirer. Ensuite, lorsqu'une balle touche un objet, des actions sont appliquées en fonction de cet objet. La dernière fonctionnalité est une fonction anti-spam, elle limite le nombre de tires possible dans un laps de temps précis.

Pistolet de détection : Cette arme permet au joueur de marquer l'ennemi. Quand il marque l'ennemi, l'autre joueur est capable de le tuer avec son pistolet de dommages.

Pistolet de dommages : Cette arme appartient au joueur qui ne peut pas voir l'ennemi. Cela lui permet de tuer un ennemi marqué.

Portails d'arrivée :

Pour donner au joueur l'opportunité de finir une pièce, nous avons implémenté des portails d'arrivée. Un portail est juste un bloc spécial avec un 'collisionneur'. Il a aussi des particules autour pour le mettre en avant. Le collisionneur est donc lié à un script qui activera la scène de fin lorsque le joueur le touchera. Par conséquent, quand un joueur touchera ce portail, l'équipe sera téléportée sur la scène de fin.

3.2.3 Boutons & Mécanismes

Dans cette section, nous avons configuré et implémenté les différentes interactions des boutons (ou commutateurs) avec la carte.

Cette partie a vraiment évolué liée à l'expérience de jeux et au type de mécanique de puzzle que nous voulions utiliser. Cette partie était vraiment dure en terme de réflexion, en effet nous devions chercher de nouveaux mécanismes mais liés à notre type de jeux.

Mécanismes créés :

- Classe de gestionnaire de signal
- Plaque de pression
- Portes
- Connecteur 'et'
- Connecteur 'non'
- Signal à sens unique (laser)
- Signal avec ordre des Boutons

Classe de gestionnaire de signal :

Le gestionnaire de signal est un préfabriqué spécial qui n'apparaît pas sur l'écran du lecteur, mais il a un rôle très important dans le mouvement du mécanisme. Ce script gère un dictionnaire de chaînes d'étiquettes de signaux et leur état actuel, notez que l'état d'un signal n'est pas un booléen, mais un entier, en dehors du script du gestionnaire de signal, il est perçu comme un booléen par la méthode getSignal. Lorsque l'entier correspondant à la chaîne de signal est supérieur à 0, le signal est perçu comme Vrai, Faux dans le cas contraire. Lorsqu'un mécanisme demande de définir un signal sur Vrai, la valeur entière est incrémentée. Il est décrémenté lorsqu'un signal Faux est demandé. Cela permet à plusieurs sources de définir un signal sur Vrai sans s'écraser les unes les autres, ainsi, le gestionnaire de signal agit comme un connecteur 'ou' ; lorsqu'au moins une source a défini le signal sur Vrai, le signal est Vrai, même si une autre source le définit sur Faux.

Plaque de pression :

La plaque de pression est une plaque qui, lorsqu'un joueur ou un bloc intervient dessus, envoie un signal Vrai au gestionnaire de signal sous l'étiquette du signal spécifié. Lorsque le joueur ou le bloc n'est plus dessus, il se

réinitialise, envoyant un signal Faux au gestionnaire de signal. Puisque les deux méthodes intégrées OnTriggerEnter et OnTriggerExit étaient un peu imprécises (au moins pour la perspective du joueur, elles ne déclenchaient souvent pas quand elles semblaient devoir l'être), nous avons utilisé la méthode OnTriggerStay, quand cette méthode est appelée elle définit le "timer" de la porte à 10, lorsque la minuterie passe de 0 ou 1 à 10 dans la méthode Update, les scripts envoie un signal Vrai au gestionnaire de signal, lorsque le timer atteint 1 dans la méthode Update, un signal Faux est envoyé au gestionnaire de signal. Chaque fois que la méthode Update est appelée, le timer est diminué de un s'il est supérieur à 0. Puisque la méthode OnTriggerStay est appelée au même rythme, le chronomètre reste près de 10 tant qu'un joueur ou un bloc est sur lui. Quand le joueur ou le bloc s'en va, le chronomètre diminue et atteint 1, envoie un signal Faux au gestionnaire de signal, puis atteint 0. L'animation est juste la plaque de pression qui est décalée vers le bas.

Portes :

La porte est un bloc normal qui est désactivé quand le signal attribué est défini sur Vrai, il est réactivé lorsque le signal est Faux. Notez que ce bloc peut également être utilisé comme une pièce de plancher, permettant un niveau plus propre.

Connecteurs 'et'/'non' :

Le connecteur 'et' est une porte qui vérifie deux signaux dans le gestionnaire de signal et émet l'opération "et" des deux signaux dans le gestionnaire de signaux.
Par conséquent, la porte 'non' est une porte qui vérifie un signal dans le gestionnaire de signal et émet l'opération "non" le signal dans le gestionnaire de signal.

Signal à sens unique (laser)

La plaque de pression n'était pas vraiment intuitive pour la mécanique du laser, le but était de verrouiller la plaque de pression une fois qu'elle était pressée. Cependant, comme le joueur s'attendait à ce qu'il revienne, cela pourrait souvent conduire à une incompréhension du puzzle. En introduisant une nouvelle fonctionnalité, le joueur peut mieux comprendre le puzzle. Le laser est essentiellement une plaque de pression qui reste bloquée dans la même position quand elle est déclenchée. Le Laser envoie uniquement un

signal Vrai lorsque la méthode OnTriggerStay est appelée pour la première fois, puis reste à sa position indépendamment des interactions. L'animation est la désactivation des faisceaux laser. (Ce laser peut être vu à la page 23)

Signal avec ordre des Boutons :

Nous avons fait un système qui gère l'ordre dans lequel les boutons ont été déclenchés, de sorte que nous pourrions utiliser cela comme une énigme dans un nouveau niveau. C'est un tout nouveau système. Nous avions les signaux avant, mais nous ne les utilisions pas du tout pour ça. Pour ce nouveau signal, un tableau est utilisé ; chaque bouton est à sa position dans la ligne des boutons. Un compteur est là pour vérifier à quelle position dans le tableau nous sommes. Chaque fois qu'un bouton est déclenché, le gestionnaire de signal vérifie si c'est le bon bouton en vérifiant si le bouton à l'endroit i (le compteur) dans le tableau de boutons est le même que le bouton qui vient d'être déclenché. Si c'est le cas, nous ajoutons un au compteur. Sinon, nous ne faisons rien. Lorsque le compteur est au même nombre que le nombre d'éléments dans le tableau, la porte est détruite. Pour utiliser ce système, nous avons eu une idée de puzzle. Cette idée était d'avoir deux pièces, chacune accessible par un seul joueur (une chambre bleue et une rouge par exemple). Dans une pièce, le joueur entre et peut lire un texte indiquant un nombre. Pour ce faire, nous avons créé un texte en 3D et nous y avons placé un collisionneur. Fondamentalement, le rendu visuel est désactivé, le texte n'apparaît donc pas. Chaque fois que le joueur entre dans le collisionneur, le moteur de rendu visuel est réglé sur "active" et à chaque fois que le joueur sort du collisionneur, le moteur de rendu visuel est défini sur "inactive". Pour détecter seulement le joueur et aucun autre gameobject, nous avons vérifié que l'objet gameobject avait le gametag "joueur" quand quelque chose entre dans le collisionneur. En d'autres termes, le texte ne s'affiche pas quand autre chose qu'un joueur entre dans la pièce. Pour ce faire, nous avons créé un gestionnaire de signal avec un tableau, dans lequel nous avons placé les boutons dans le bon ordre. Chaque fois qu'un bouton est enfoncé, le signal vérifie s'il est bon ou non. Sinon, rien ne se passe. Si c'est le cas, un compteur est incrémenté et le signal attend le prochain bouton. Ces deux systèmes fonctionnent parfaitement. Il ont été mis en œuvre dans le niveau 5 visible à la page 24.

3.2.4 Ennemis & IA

Dans notre jeu, nous avons choisi de créer des ennemis avec des mécaniques liées aux spécifications de notre jeu : la vision. Au début, nous n'avions pas pensé que l'ajout d'ennemis pouvait être une bonne idée. Nous disions que cela changerait complètement notre façon de jouer. Mais en effet, il fallait changer un peu notre façon de jouer. Sans ennemis, le joueur pouvait rester sans bouger pendant qu'il résolvait le niveau. De cette façon, nous avons créé des ennemis pour ajouter quelque chose de "mobile" à notre jeu.

Créé :

Des ennemis suivant le joueur qui approche ont été créés. Plus précisément, lorsqu'un joueur pénètre dans la zone sélectionnée de l'ennemi, l'ennemi commence à le viser et le suit tant qu'il est encore dans son champ de tir. Ces ennemis sont capables de trouver leur chemin à travers la pièce même s'il y a des trous, il prendra le chemin le plus court pour atteindre sa cible. De plus, lorsqu'il n'y a plus de cible, l'ennemi revient à son point de départ. Enfin, lors du suivi d'un joueur, une petite animation apparaît sur l'ennemi, pour alerter le joueur qu'il est en danger.

Détection du joueur :

Le monstre détecte un joueur une fois que celui-ci entre dans le rayon de détection. Cet ennemi visera le joueur le plus proche dans sa portée. Par conséquent, si un autre joueur s'approche, l'ennemi changera de cible. Pour ce faire, une fonction calculant la distance de chaque joueur a été faite, à ce moment nous avions juste besoin d'ajouter une condition pour la portée de sa vision. Pour trouver ces joueurs nous avons utilisé une fonction appelée "FindGameObjectWithTag(string tag)" cette fonction nous permet de trouver la liste des joueurs existants. Une fois qu'un joueur est trouvé, l'ennemi commence à bouger.

Mouvements des ennemis :

Ces ennemis ne volent pas, ils doivent donc trouver leur chemin à travers les obstacles pour atteindre le joueur. Ils utilisent un script de contrôle utilisant un NavMesh pour se déplacer sur la carte. Pour mettre en œuvre cette fonctionnalité, de nombreuses recherches sur les intelligences artificielles ont été faites. Cette méthode de NavMesh nous permet de déplacer un objet automatiquement sur un chemin donné. Une fois qu'un joueur est trouvé,

un algorithme de backtracking est activé pour déterminer le meilleur chemin pour l'atteindre (tout cela en évitant les trous et les murs). Finalement cet ennemi revient à sa position de départ quand il n'y a plus de joueurs dans sa portée.

Dégâts :

Maintenant que l'ennemi peut atteindre le joueur, quelque chose doit arriver. Le monstre poursuivra le joueur à travers la carte. Si celui-ci arrive assez proche, le joueur subit des dégâts. Pour tuer le joueur, l'ennemi peut infliger des dégâts en restant proche de lui ou simplement en l'éjectant de la carte.
Nous ne voulions pas que le joueur soit abattu d'un seul coup par l'ennemi. Par conséquent, nous avons créé une barre de vie pour les deux, le joueur et l'ennemi. Cette barre de vie est placé juste au-dessus de leur tête. Ces barres peuvent être vues dans l'image ci-dessous. Il y avait aussi du travail sur celles-ci pour s'assurer qu'elles soit toujours devant la caméra.

Évolution du concept :

Comme indiqué dans l'introduction de cette section, nous avons choisi d'implémenter des ennemis uniquement lors de la seconde présentation. Ce n'était pas pour leur mise en œuvre difficile, c'était principalement pour notre game-play. En effet, nous ne voulions pas qu'ils changent complètement notre façon de jouer. Les ennemis peuvent être très difficiles à implémenter dans la création de puzzle pour garder le niveau intéressant, sans placé des ennemis inutiles. Par conséquent, il y avait un travail à faire pour bien les intégrer. Finalement nous avons décidé de jouer un peu avec ces ennemis et surtout avec leurs spécifications : un joueur est capable d'étiqueter le monstre avec son arme puis l'autre est capable de le voir et de le tuer seulement une fois qu'il a été marqué. En effet, nous avons essayé d'utiliser ces ennemis comme un nouveau mécanisme de puzzle même si c'était vraiment difficile.

À la page 25 l'ennemi est visible en rouge. Les barres de vie respectives de celui-ci et du joueur son également visibles en verte.

3.2.5 Camera et Lumières

Cette section est là pour donner au joueur une meilleure expérience de jeu. C'est exactement le même but que pour les sections graphiques et audio. En effet, si le jeu n'est pas plaisant en termes de beauté, vous y jouerez moins qu'un jeu plaisant. Par conséquent, certains détails comme celui-ci ont une section entière dans ce rapport. De plus, une bonne mise en œuvre de la caméra aide vraiment le joueur à comprendre l'environnement et à s'y adapter.

Évolution de la Camera :

Au début du projet, nous avons pensé qu'une caméra à la première personne pourrait être une bonne idée pour immerger le joueur dans son personnage.

Finalement, nous avons compris que seule une vue à la troisième personne serait meilleure. Le but de ce point de vue pour la caméra est de donner au joueur un aperçu facile de la carte et de faciliter l'identification de l'environnement. Par conséquent, nous n'avons eu aucun problème pour implémenter cette vue. Comme la caméra n'a pas besoin de bouger, il n'y a pas eu beaucoup de recherches dans cette partie. Les recherches principales ont été de trouver ce bon point de vue.

Évolution des Lumières :

La partie éclairage est principalement pour un aspect très agréable. Par conséquent, cette partie a évolué au cours de la création du jeu. Nous touchions à cette partie à chaque fois qu'un niveau était terminé. Pour chaque pièce nous avons adapté l'éclairage à la caméra et au niveau. Pour sa création, nous avons utilisé les outils d'éclairage de base de Unity. Vous pouvez voir des captures d'écran de nos niveaux à la page 23.

3.2.6 Vision

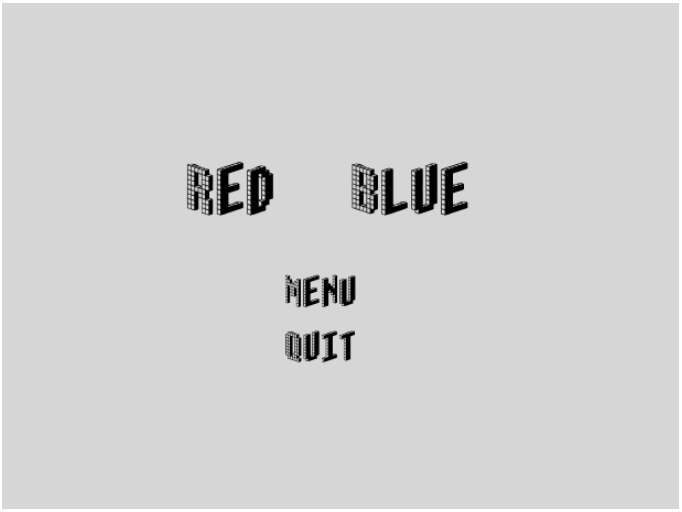
Spécifique à notre projet, c'est la partie où la différenciation entre les deux joueurs en terme de vision apparaît. En effet, si nous voulons que les deux joueurs aient une différenciation dans le jeu, nous devions faire un menu au début pour leur permettre de choisir leur couleur. Ensuite, un script sera appelé pour charger la carte différemment en fonction de la couleur choisie.

Menu :

Un menu complet lié à aux autres menus du jeu a été créé. Ce menu a été réalisé avec les mêmes spécifications de conception que celles utilisées dans le menu principal (voir page 28). Dans ce menu les joueurs ont accès à deux boutons leur permettant de sélectionner la couleur qu'ils souhaitent. Ils peuvent également revenir au menu principal ou quitter le jeu. Lorsque un joueur choisit une couleur, il charge la carte avec les bonnes caractéristiques.

Sélection de la Vision :

Lorsque le joueur sélectionne la couleur, un script est appelé. Ce script charge la carte sélectionnée juste avant avec les bonnes fonctionnalités. Ces caractéristiques sont : les blocs visibles (en termes de couleur sélectionnée), l'ennemi (visible ou non), le pistolet du joueur (tag ou kill) et son point d'apparition.



3.2.7 Collisions

Les collisions sont les interactions entre le joueur et son environnement (objet, trous, tires, etc.). En fonction des caractéristiques d'un objet, le joueur interagira différemment avec lui. Ces collisions sont vraiment importantes dans le jeu, si elles ne sont pas bien faites, des problèmes et des bugs peuvent se produire facilement. C'est pourquoi cette partie a évolué avec le jeu tout le temps. Pour éviter les problèmes dus aux collisions, le travail a été fait en prenant cela en compte à chaque fois.

Création : Une grande partie des collisions ont été créées en même temps que les objets mêmes (grâce à Colliders et rigidBody). Donc les recherches ont été faites directement pour la création en tant que tel.

Kill Zone : Une kill zone a été créé pour détecter une chute de quelque chose. Sans quelque chose comme ça, un joueur qui tombe ne mourra jamais, ce serait un peu ennuyeux. Par conséquent, quand quelque chose tombe, il est détecté. Lorsqu'un bloc est détecté, il réapparaît à sa position d'origine. Cependant, lorsqu'un joueur tombe, le niveau redémarrera complètement.

3.2.8 Création de niveaux

Cette partie était une partie importante de notre projet. C'est dans cette section que nous avons envisagé la création d'un nouveau type de puzzle, en utilisant des mécanismes, des visions et en fait la complémentarité des joueurs.

Introduction & Évolution :

La création de niveaux peut sembler simple à première vue, cependant, créer un niveau qui se sent fini et propre prend un peu plus de temps. En effet, plus vous passez de temps sur un seul niveau, plus il sera raffiné. Ainsi, nous passons beaucoup de temps sur les détails afin de cacher la solution sans conduire le joueur dans de fausses voies avec des détails non désirés. Habituellement pour penser nos puzzle nous commençons avec un mécanisme de base de puzzle avec lequel nous voulons jouer, puis nous évoluons autour de celui-ci, en vérifiant les failles et en ajoutant des éléments autour de lui. Au fur et à mesure que le puzzle grandit, nous commençons à le compresser pour le rendre plus difficile à résoudre, puis nous refaisons une passe de vérification des failles et d'ajout d'éléments potentiels. Lorsque nous finissons un casse-tête, nous essayons de nous mettre dans l'esprit d'un joueur différent, un joueur qui a du mal à comprendre et à utiliser la force brute (essayant différentes approches) et un joueur qui essaie de trouver le chemin le plus rapide. peut-être même essayer de trouver des solutions non désirées. Si le puzzle n'est pas aussi bon que prévu, nous recommençons à partir de l'élément de base du puzzle que nous voulions faire évoluer. Nous ne commençons jamais à créer un puzzle directement dans le créateur de la scène d'Unity, nous commençons toujours sur papier ou sur un grand tableau blanc, puisque nous devons souvent redessiner des parties du puzzle pour le rendre meilleur et plus dur.

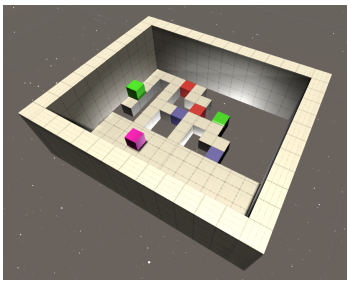
Tutoriels :

Nous avons mis en place quelques niveaux et quelques tutoriels, les tutoriels ne sont que des niveaux plus simples qui visent à initier les joueurs à un nouveau mécanisme ou à un moyen de résoudre un casse-tête. Par exemple le premier tutoriel se compose d'une ligne simple avec une porte et deux plaques de pression de chaque côté de la porte, un joueur doit marcher sur la première plaque de pression et le second passe par la porte ouverte . La deuxième plaque permettant au premier joueur de passer. Les joueurs comprendront alors le fonctionnement des plaques de pression et de la porte.

Dans le deuxième tutoriel il y a deux plaques de pression devant la porte et un bloc mobile, un joueur ne peut pas permettre au second joueur d'atteindre l'autre côté de la porte, il doit pousser le bloc sur la deuxième plaque de pression pour être capable d'appuyer sur les deux plaques de pression en même temps, ouvrant la porte pour le deuxième joueur. Ensuite, le deuxième joueur peut maintenir la porte ouverte avec une autre plaque de pression de l'autre côté pour permettre au premier joueur de passer. La mécanique du bloc est alors comprise. Le troisième tutoriel introduit des blocs de couleur, car avant qu'une porte sépare les joueurs de la fin du puzzle, deux plaques de pression se tiennent devant la porte et une derrière la porte. Les deux premières plaques de pression sont précédées d'un bloc de couleur, une plaque de pression est après un bloc bleu et une après un bloc rouge. Un bloc mobile est également présent. Comme dans le deuxième tutoriel, les deux plaques de pression doivent être pressées pour ouvrir la porte. Ainsi, un joueur doit pousser le bloc sur l'une des boîtes de couleur (celle qu'il voit) et le mettre sur la plaque de pression derrière lui. L'autre joueur peut alors se tenir sur l'autre plaque de pression, ouvrant la porte, et comme dans les deux premiers tutoriels, le joueur qui passe peut maintenir la porte ouverte pour le joueur derrière. Cela leur permettra de comprendre la différence dans la vision entre les deux joueurs. Le quatrième tutoriel est juste deux lignes liées au début et à la fin de la carte, chacun d'entre eux ont soit un bloc bleu, soit un bloc rouge au début , un laser et une porte inversée (une porte et un connecteur 'non'). Quand un joueur atteint un laser, la porte est fermée, empêchant le joueur de revenir, les deux joueurs peuvent atteindre le point final à travers leur ligne respective (bleu et rouge). Il n'y a pas de difficulté à résoudre celui-ci, mais le joueur apprend comment fonctionne la mécanique du laser.

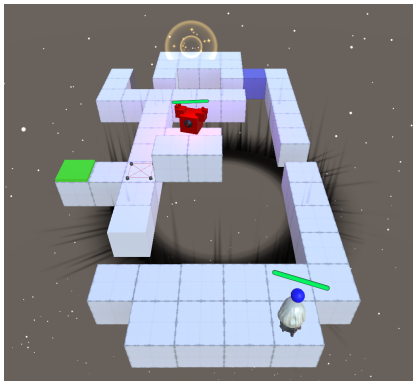
Niveau 1 :

Le niveau 1 est le même niveau que le premier dans l'exemple page 4.



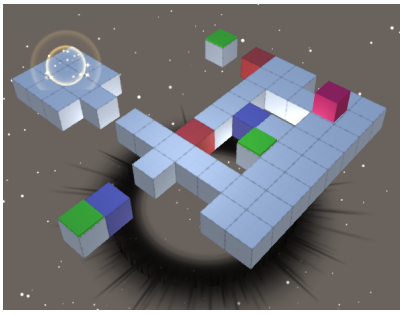
Niveau 2 :

Le niveau 2, comme sur l'image ci-dessous, concerne l'ennemi. En principe, un joueur doit marquer l'ennemi en se positionnant à un endroit précis, puis l'autre joueur peut avancer, tuer l'ennemi et ouvrir la voie au deuxième joueur. Le laser empêche un joueur de ramener l'ennemi à un endroit que l'autre joueur peut marquer.



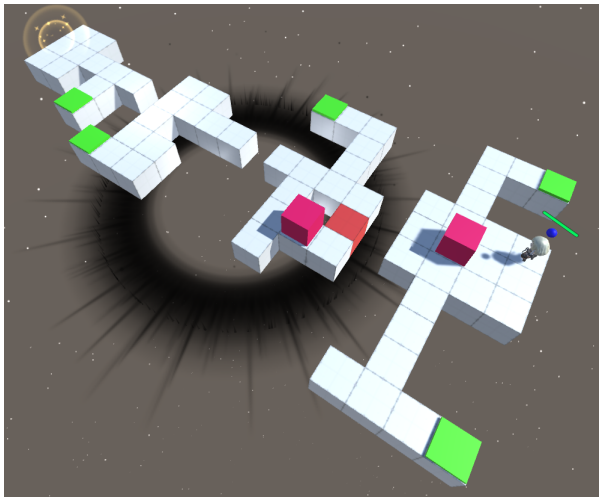
Niveau 3 :

Le niveau 3 comme montré dans l'image ci-dessous consiste à tromper le joueur dans de nombreuses fausses solutions, ils croient d'abord que beaucoup de solutions sont possibles, mais finalement se coincent à un point et doivent en explorer un autre. Il y a 3 plaques de pression, deux permettent l'accès à une autre plaque de pression et une ouvre la porte finale. Un bloc mobile est présent et peut être placé sur les 3 plaques de pression, cependant les joueurs doivent agir ensemble pour placer le bloc sur le bon.



Niveau 4 :

Le niveau 4 représenté sur l'image ci-dessous est composé de 4 parties, une position de départ avec un bloc mobile et des plaques de pression, une première partie centrale avec un bloc mobile et une plaque de pression, une deuxième partie médiane avec une plaque de pression, et la partie terminale avec une plaque de pression. Ces parties sont séparées par des portes qui empêchent le joueur de voyager librement entre elles. La solution ici est un peu complexe car les joueurs doivent s'organiser pour appuyer sur les plaques de pression et déplacer le bloc. placer le bloc sur certaines plaques de pression nécessite que les deux joueurs manipulent le bloc, par exemple dans la partie de départ, un joueur doit être placé à un endroit, l'autre peut ainsi pousser le bloc devant lui, puis il peut pousser le bloc sur la plaque de pression.



Niveau 5 :

Pour le dernier niveau, nous voulions quelque chose composé de plusieurs mécaniques et puzzles, donc un peu plus complexe que les autres niveaux. Nous avons utilisé trois puzzles pour ce niveau :

D'abord, nous avons utilisé les casse-têtes habituels de la plaque de pression.

Pour le deuxième casse-tête, nous avons eu une nouvelle idée de puzzle. Cette idée était d'avoir deux pièces, chacune accessible par un seul joueur (une chambre bleue et une rouge par exemple). Dans une pièce où le joueur entre, un texte indiquant un numéro est affiché. Enfin, nous avons utilisé l'ennemi.

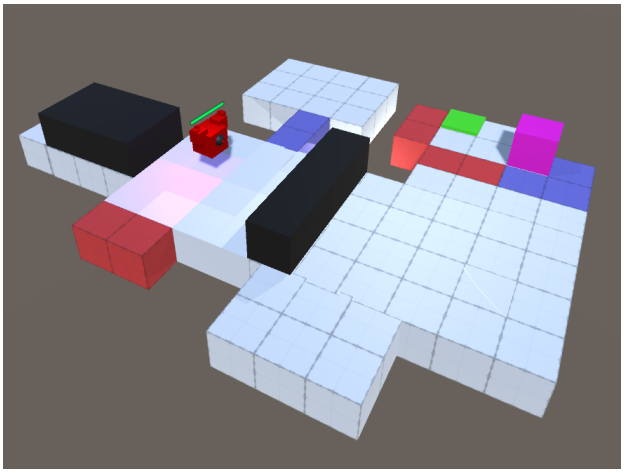
L'utilisation de tous ces puzzles nous a permis de créer un niveau où les deux joueurs devaient résoudre un puzzle pour accéder au suivant : ici, ils doivent d'abord résoudre un puzzle avec les plaques de pression pour accéder à l'ennemi, puis ils doivent vaincre le ennemi pour accéder au dernier casse-tête de plaque de pression. En le résolvant cela permet aux joueurs d'accéder aux boutons avec un ordre. La difficulté principale avec ce la création de niveaux était de créer un niveau intéressant à jouer, ce qui signifie difficile mais pas impossible. En dehors de cette difficulté de base de conception de niveau, nous avons également rencontré quelques difficultés avec ce niveau en particulier. Nous avons fait face à trois problèmes principaux :

Tout d'abord, tout ce dont nous avions besoin ne fonctionnait pas systématiquement. Les objets utilisés dans de ce niveau on été créées par tout le monde. C'est pourquoi des problèmes de compatibilité était présent durant sa création.

Le problème principal concernait l'ennemi. Une de ses fonctions pour détecter le joueur ne fonctionnait pas et l'ennemi ne trouvait pas le joueur dans ce niveau, ce qui était un énorme problème car le niveau avait vraiment besoin de son ennemi.

Enfin, la difficulté de ce niveau était la caméra. Comme un joueur est téléporté mais pas l'autre, nous devions créer un script pour la caméra qui détecterait quand le joueur est téléporté et ne suivrait que celui-ci.

Après tout ces problèmes rencontrés, le niveau est quasiment fonctionnel. Ce qui est un bon point.



3.2.9 Multijoueur

Comme le but de notre projet est de créer un jeu basé sur la complémentarité des joueurs, le multijoueur a dû évidemment être implémenté. Cette partie contenait toute la création de la structure multijoueurs. Cette partie a été faite très rapidement pour la première présentation. En effet, comme le reste du jeu est basé sur cela, nous ne pouvions pas la laisser traîner.

Progression au fil du temps :

En fait, après la première présentation, nous n'avons plus touché à la base de l'implémentation multijoueurs. Mais nous devions suivre chaque fois que nous implémentions quelque chose de nouveau (comme nous devions le lier au gestionnaire de réseau). Cette partie a donc continué d'avoir des recherches liées à son implémentation pendant tout le semestre pour éviter les bugs.

Création :

Pour la création du multijoueur, nous avons utilisé le gestionnaire de réseau contenu dans Unity. Le gestionnaire de réseau est un composant gérant les aspects réseau d'un jeu multijoueurs. Voici une liste des fonctionnalités du gestionnaire de réseau que nous avons utilisées :

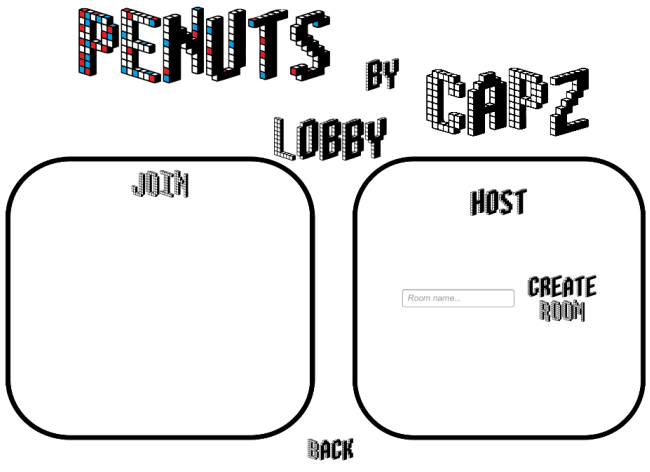
- Gestion de l'état du jeu : Un jeu multijoueur en réseau peut fonctionner en trois modes : en tant que client, en tant que serveur dédié, ou en tant qu'hôte qui est à la fois un client et un serveur en même temps.
- Service multijoueur : Il a été utilisé pour publier notre jeu et permettre à tout le monde de créer une salle (Hôte et Client) ou Rejoindre une salle (Client).
- Gestion de Spawn : Nous l'avons utilisé pour gérer la génération (instanciation en réseau) de GameObjects en réseau à partir de Prefabs. En d'autres termes, nous l'avons utilisé pour faire apparaître nos joueurs et nos objets sur la carte.
- Gestion de scène : La plupart des jeux ont plus d'une scène. En effet, il existe un menu en plus de la scène où le jeu est réellement joué. Le gestionnaire de réseau est conçu pour gérer automatiquement les transitions d'état et de scène d'une manière qui fonctionne pour un jeux multijoueur.
- Matchmaking : Lié au service multijoueurs, il aide à rejoindre ou à créer une salle.

Création du Lobby

Après la création des paramètres multijoueurs expliqués ci-dessus, le jeu était fonctionnel. Mais ce n'était pas complètement lié à nos attentes. En effet, l'interface pour l'utilisateur n'était pas agréable à utiliser ou la gestion des scènes ne fonctionnait pas comme nous le voulions. Pour résoudre ce problème, nous devions aller plus loin dans ce gestionnaire de réseau. Nous avons commencé la création d'un menu de lobby qui permet au joueur de créer ou de rejoindre une salle avec n'importe qui. Ce menu a été réalisé avec le même design que celui utilisé dans les autres menus (voir page 29). Ce menu était le plus difficile à créer. Nous avons dû comprendre complètement comment le gestionnaire de réseau fonctionne pour le rendre fonctionnel.

Lobby Présentation :

Dans ce menu, l'utilisateur a la possibilité de rejoindre une salle, d'organiser une partie ou de revenir au menu principal.
Host : Le cadre d'hébergement était le plus simple à créer (ne tenant pas en compte du bouton Back). Nous avons créé une case pour l'entrée de texte correspondant au nom de la pièce et un bouton appelant un script créant une pièce.
Join : Le cadre Join répertorie toutes les pièces disponibles ou affiche un texte d'état pour informer l'utilisateur de la validité de connexion.



3.2.10 Menu & Options

Dans cette partie, nous avons implémenté différentes options et interfaces pour aider et donner quelques configurations à l'utilisateur. En effet, si le joueur peut couper les effets sonores, changer la qualité de l'écran, utiliser un menu pause, une sélection de niveau et ainsi de suite, il aura une meilleure expérience du jeu.

Progression :

Lors de la première présentation, le menu n'était qu'un écran où le joueur pouvait choisir s'il voulait jouer ou quitter. Pour la deuxième présentation, un excellent travail a été accompli. En effet, des menus fonctionnels ont été créés, adaptés à notre design et liés aux paramètres du Multijoueur. Il nous a fallu beaucoup de temps avant la deuxième présentation pour les créer tous, mais cela a finalement été fait. Ces menus n'ont pas vraiment évolué pour la dernière présentation car il n'y en avait pas besoin. Le lobby a tout de même été corrigé en raison d'un problème et le sélecteur de niveau a changé pour pouvoir accueillir plus de scènes.

Implémentés :

- Différentes scènes de menu adaptées à notre design. Cela comprend :
  - Un Menu Principal avec des boutons Play, Settings and Quit
  - Un Lobby (Play) : permettant au joueur de rejoindre ou d'héberger une partie
  - Un menu de sélection : permettant au joueur de choisir le niveau
  - Menu de sélection des couleurs : où le joueur peut choisir la couleur qu'il veut avoir.
  - Menu Settings : permet au joueur de modifier les paramètres de volume ou les paramètres d'écran.
  - Crédits & Menu de fin : apparaissant lorsque les joueurs finissent un niveau
  - Quit & Pause : Laissant le joueur revenir au menu ou quitter le jeu s'il veut
- Conception : Tous ces menus ont été alignés sur le même design

Recherches :

De nombreuses vidéos et ressources ont été utilisées pour cette partie. En effet, nous devions comprendre clairement comment le multijoueur, la qualité, la résolution, l'audio ou la gestion des scènes étaient contrôlées.

Design :

Ces menus ont été créés avec un design simple permettant à l'utilisateur de trouver facilement son chemin dans toutes ces scènes et boutons. Nous voulions aussi que le menu soit vraiment agréable, c'est pourquoi seulement des choses utiles sont dedans. Nous l'avons fait en utilisant des images & TextMeshPro avec une police correspondant à notre jeu (police cubique). TextMeshPro est une asset disponible dans Unity permettant d'avoir de très beaux éléments de texte. Cette conception est également spécifique à notre environnement, en effet, nous voulions quelque chose de vraiment simple sans grands détails. Pour voir des animations sur des boutons et ainsi de suite, le meilleur moyen est de jouer à notre jeu, vous pouvez le trouver sur notre page web. Cependant, pour avoir un aperçu, vous pouvez simplement regarder les images présentées ci-dessous pour chaque menu.

Menu principal :

C'était le premier menu créé. C'est avec celui-ci que le design a été créé. En effet, c'est à ce moment que des boutons, des modèles pour le texte ou le fond ont été implémentés. Tout cela étant fait, nous avions compris les bases de la création de menu.

Chaque fois que vous appuyez sur un bouton, il désactive le cadre contenant le menu principal et en active un autre. Par exemple, lorsque vous appuyez sur le bouton Settings, c'est le cadre des options qui sera activée, de même pour le bouton Play.

Enfin, le bouton Quit permettant au joueur de quitter le jeu utilise un script simple appelant la fonction Application.Quit().

Lobby :

Ce menu a été le plus difficile à créer car nous avons dû comprendre très bien les fonctionnalités du gestionnaire de réseau. L'implémentation de ce menu a été expliquée dans la partie multijoueurs à la page 27.

La particularité qui n'a pas été expliquée dans la partie multijoueurs était la façon dont les boutons et le texte d'état fonctionnent. Les boutons pour rejoindre une pièce ne sont disponibles que si une pièce est créée, sinon les boutons ne feront rien de spécial. Pour le texte de l'état, nous avons utilisé un script pour changer son contenu chaque fois qu'il est mis à jour. S'il n'y a pas de chambres disponibles dans la liste, il dira "There is no rooms for the time..." sinon, pour 3 pièces il affichera "There is 3 rooms !"

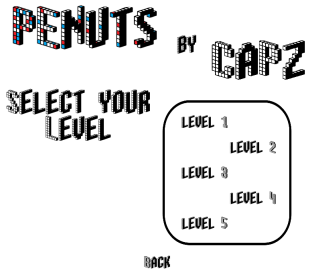
Enfin le bouton Back permettant au joueur de revenir au menu principal fonctionne exactement comme le bouton pour entrer dans ce menu (Activer un cadre ou non).

Menu de sélection du niveau :

Ce menu est là pour laisser le joueur sélectionner le niveau auquel il veut jouer. Le menu est également lié au menu vision qui permet au joueur de changer de couleur au début d'un niveau.

Chaque bouton de niveau charge la scène spécifiée et les joueurs sont envoyés dans celui-ci. Par conséquent, ces boutons utilisent les fonctions de gestion de scène du gestionnaire de réseau

Enfin, le bouton Back est exactement le même que celui présenté dans le menu du lobby. Et ce sera la même chose pour chacun de ces boutons.



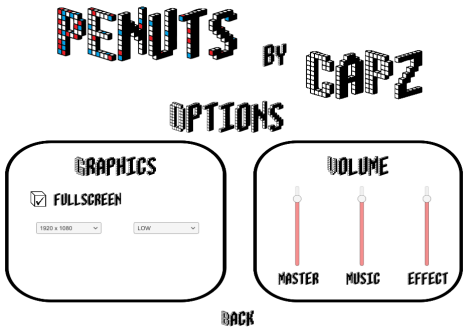
Menu des Options :

Comme son nom peut nous le faire comprendre, ce menu est une base dans les jeux. Il permet au joueur de changer les paramètres comme l'audio ou les graphiques.

La section graphique laisse le joueur activer le plein écran, changer la résolution ou changer la qualité. Toutes ces fonctionnalités ont été implémentées avec l'interface de base de Unity, y compris les boutons, Dropdown ou Toggle. Ensuite, ces interfaces devaient être liées à des scripts pour contrôler les paramètres du jeu. Pour l'implémenter, nous devions comprendre comment Unity supervise les paramètres graphiques. Ensuite, en utilisant des fonctions liées aux graphiques, nous pouvons modifier ces paramètres directement dans notre jeu.

La section Audio est là pour laisser le joueur changer les volumes disponibles. En effet, il y a trois curseurs qu'il peut déplacer pour changer le volume. Le premier est ici pour contrôler l'ensemble de l'audio du jeu et les autres sont respectivement pour changer le volume de la musique et le volume SFX. Pour les créer, nous avons utilisé les fonctionnalités de base des curseurs liés à un mélangeur audio contrôlant nos volumes. En effet, un curseur peut être utilisé pour sélectionner une valeur dans une certaine plage, il suffit de la lier grâce à un script au volume de chaque table de mixage audio.

Enfin, si nous avons le temps juste avant cette dernière présentation, nous allons peut-être ajouter quelques fonctionnalités dans ce menu. Les idées principales pour le moment étant quelque chose pour changer la musique de fond ou quelque chose pour changer les touches du joueur.



Menu de Fin & Crédits :

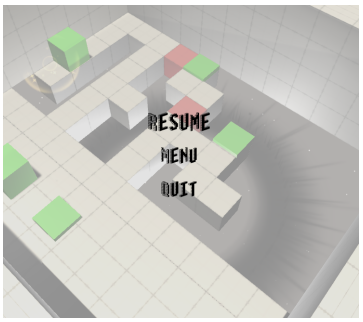
Cette scène apparaîtra lorsque les joueurs termineront un niveau. En effet, si l'un des joueurs entre dans un portail d'arrivée, les deux joueurs sont envoyés sur cette scène de fin.

Les boutons utilisés dans ce menu sont vraiment simples. Vous pouvez quitter le jeu (Application.Quit), vous pouvez aussi revenir au menu (gestionnaire de scène). Enfin vous pouvez cliquer sur PEnuts pour accéder à notre site web (utilisant Application.OpenURL).

Pause Menu de Pause :

Un menu de pause est essentiel dans un jeu, en effet, il permet au joueur de quitter le jeu ou de revenir au menu même s'il était en jeu.

Pour sa création, nous avons utilisé un script lié au joueur, chaque fois que le joueur appuie sur la touche Échap, le jeu sera mis en pause. Plus précisément, le temps sera arrêté dans le jeu et un cadre apparaîtra.



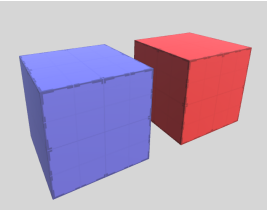
3.3 Design

3.3.1 Codes Couleurs

Nous avons dû faire quelques choix pour mettre de bons codes de couleur dans le jeu. En effet, ces couleurs aident les joueurs à voir quelque chose et à donner des informations. Ces informations peuvent concerner le type d'objet, les interactions possibles, etc. C'est pourquoi cette partie ne pouvait pas être négligé si on voulait donner une expérience agréable aux joueurs.

Les deux couleurs principales de notre jeu sont le bleu et le rouge, ce qui différencie la boîte bleue qui est seulement vue par le joueur bleu de la boîte rouge qui est seulement vue par le joueur rouge.

Nous souhaitions que le joueur puisse voir directement l'impact d'un mécanisme sur un autre, par exemple ; Comment le joueur sait-il quelle plaque de pression sur deux ouvre une porte spécifique ? Pour résoudre ce problème, nous avons pensé permettre aux différents mécanismes d'avoir une couleur différente, qui correspondrait aux lien entre ces objets. Pour cela, nous avons dû modifier le système de gestion des signaux pour tenir compte des différentes couleurs pour chaque signal, une nouvelle méthode a été ajoutée qui permet à un mécanisme de récupérer la couleur d'un signal donné. Chaque mécanisme peut traiter la couleur différemment en fonction de leur modèle 3D, il n'y a pas de contraintes sur lesquelles les parties sont colorées ou non. Les couleurs sont prises à partir d'une banque de 20 couleurs différentes, les 10 dernières étant un peu plus proches des 10 premières, car elles sont plus difficiles à différencier, elles ne seront utilisées que si le niveau nécessite plus de 10 signaux. Si les 20 couleurs sont toutes utilisées, le jeu en générera aléatoirement un nouveau, ne correspondant à aucune des couleurs précédentes. Le problème avec cela est que certaines couleurs peuvent apparaître très proches les unes des autres, d'où la raison pour laquelle elles ne sont utilisées que dans le cas improbable où un niveau nécessite plus de 20 signaux.



3.3.2 Graphismes

La mise en œuvre de particules, d'assets et de meilleure design est dans cette partie. Comme les codes de couleur, mieux l'aspect graphique est mieux la compréhension et l'expérience du joueur peut être. De plus, tout cela est là pour donner vie et dynamisme au jeu.

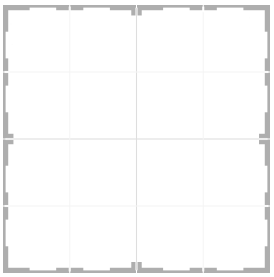
Évolution :

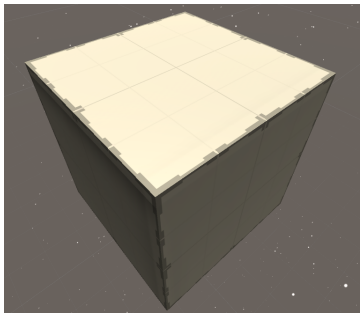
Au début du projet, comme personne dans le groupe n'était bon dans le design, nous avons décidé que, nous n'irions pas loin et que nous ne pouvions pas baser notre jeu sur nos graphismes. Nous avons compris que commencer à faire notre propre conception graphique prendrait vraiment beaucoup de temps et nous avons préféré que le game-play soit bien fait. C'est pourquoi presque toutes les particules, modèles, matériaux et textures de ce projet proviennent de l'assets store de Unity.

L'évolution des graphismes a été principalement réalisée juste avant les présentation. En effet, comme ils ne dépendent pas des autres sections en termes de codage, ils n'étaient pas compliqués à mettre en œuvre et nous pouvions aller aussi loin que nous le souhaitions. À chaque fois ces graphismes étaient essentiels pour une expérience agréable. Sans cela, nous ne pourrions pas distinguer un bloc du portail d'arrivée ou d'un bouton. C'est pourquoi cette partie a évolué en parallèle principalement pour l'expérience du joueur.

Les Blocs :

Comme notre personnage évolue dans le monde d'un bloc, notre texture pour ces blocs devait être bien choisie. Finalement, le matériel utilisé pour nos blocs a été trouvé très rapidement dans l'assets Store. Nous l'avons prise car c'était une texture simple et agréable comme nous le voulions.





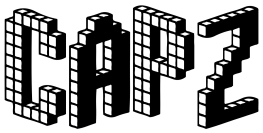
Le Personnage :

Pour le personnage nous avons trouvé un modèle qui correspond à notre attente en terme de style. Au début, nous pensions que nous le changerions rapidement. Finalement, il c'est avéré correspondre plutôt bien au style du jeux.



Police d'écriture :

Prendre notre propre police était quelque chose d'essentiel pour donner plus de détails à notre environnement.



Particules :

Ajouter des particules est un must dans un jeu, cela donne de la vie, du dynamisme et des éléments de compréhension. En effet, ces particules rendent un jeu plus attrayant et agréable. Dans notre jeu nous avons utilisé des particules pour beaucoup de choses :

- arrière-plans
- Portails d'arrivées
- Ennemis
- Balles

Vous pouvez voir toutes ces particules dans les images disponibles à la page 23. Mais pour les voir bouger, vous aurez besoin de jouer à notre jeu.

3.3.3 Musiques

Pour nous, la musique a un impact réel sur le joueur, de la bonne musique peut l'aider à s'immerger dans l'environnement du jeu. C'est pourquoi nous avons fait de notre mieux pour faire quelque chose de bien dans cette partie du jeu, même s'il n'y a pas de musiciens dans le groupe.

Création :

Nous avons compris très vite, que sans musiciens, la création d'une musique serait compromise. Après réflexion, nous avons décidé de prendre directement cette musique sur Internet. À ce moment nous avons cherché pendant un bout de temps une bonne musique correspondant à nos attentes. Premièrement, nous avions besoin de quelque chose avec un copyright libre. Ensuite, nous voulions quelque chose de bien pour laisser le joueur se concentrer sur le jeu. Nous avons trouvé une première musique juste avant la première présentation, mais ce n'était pas vraiment ce que nous voulions.

Après un long moment sans y toucher, nous avons finalement décidé que nous pouvions laisser le joueur choisir la musique qu'il voulait. C'était un compromis de génie, nous pouvions mettre différents styles de musique ou des tempos différents, puis le joueur aurait juste à choisir dans le menu.

Le problème de trouver cette musique était toujours là, en effet, nous pouvions maintenant trouver n'importe quoi, mais n'importe quoi de bon lié au jeu. C'est à ce moment que l'idée de prendre un algorithme pour générer de la musique nous est venue. Chaque piste générée est une combinaison unique de sons, de rythmes et d'instruments. Par conséquent, nous avons juste dû laisser l'algorithme trouver la chanson la plus appropriée pour nous.

L'algorithme que nous avons utilisé s'appelle Computoser. Cet algorithme est actuellement expérimental. Il peut générer à la fois de bonnes et de mauvaises pièces mais nous pouvions réessayer à nouveau tant que le son ne correspondait pas à nos attentes. Ce site était parfait, nous pouvions choisir les instruments utilisés, le tempo, l'ambiance, etc. Avoir des musiques dans notre jeu était maintenant vraiment à notre échelle. Nous avons finalement trouvé de bonnes musiques de fond pour notre jeu grâce à cet algorithme.

Implémentation :

La mise en œuvre de la musique dans Unity ne fut pas longue. Nous avons utilisé les outils de base et créé un petit script pour éviter que cette musique soit coupée quand la scène changeait.

3.3.4 Effets sonores

Cette partie, négligée dans beaucoup de projets, peut en effet donner une réelle immersion aux joueurs. Par conséquent, nous avons essayer d'adapter les effets sonores à l'environnement. Nous voulions un jeu où le joueur pouvait facilement s'immerger. Nous pensions avoir besoin d'une bonne ambiance pour atteindre notre objectif. C'est la raison pour laquelle nous avons fait un sérieux point sur la partie SFX. Nous avons dû trouver les effets sonores de plusieurs objets.

Création :

Nous avons commencé la mise en œuvre de ces effets sonores juste à la fin de ce projet. En effet, c'était important seulement si notre jeu était déjà bien commencé.

Le choix des effets sonores était le plus long, nous devions trouver les bons sons, et ils devaient être libres de droits. Pour implémenter les effets sonores, il existe une fonction dans Unity appelée "effet de sons". Il suffit juste de l'appeler au bon moment. Ainsi, par exemple, pour les signaux c'est quand ils sont déclenchés.

Tous ces sons sont disponibles sur notre site ou dans le jeu.

Boutons & plaques de pression :

Nous avons créé un effet sonore pour les boutons et les plaques de pression. Nous avons mis le même effet sonore pour ces deux objets car, de notre point de vue, il était plus logique que toutes les parties "cliquables" du jeu aient le même son lorsqu'elles sont activées.

Ennemis :

Pour l'ennemi, nous avons ajouté des sons pour toutes ses actions.

Son de détection : Quand il détectera un joueur, en plus des particules un son alarmera le joueur.

son des dégâts : Lorsque des dommages sont causés au joueur, un son rapide est joué.

Mort : Quand un ennemi meurt, il y a aussi un effet sonore.

Le Joueur :

Il y a des effets sonores quand il envoie une balle ou quand il meurt.

3.3.5 Animations

Comme la partie graphique, les animations des joueurs et la fluidité des mouvements ne peuvent que donner un meilleur aspect au jeu et à l'environnement.

Nous n'avons pas fait beaucoup d'implémentations d'animations. Avec un temps passé très important nous aurions pu créer de petites fonctionnalités. Mais la rentabilité temps/rendu n'était pas assez satisfaisante. Nous nous sommes donc penchez sur d'autres point plus important dans notre jeux. Les animations n'ont pas vraiment d'impact sur notre game-play. Nous avons préféré nous concentrer sur les graphiques, les particules, les sons ou le design plus que dans les animations.

Finalement, nous avons touché un peu aux animations pour changer celles du personnage trouvé dans l'assets store. En effet, nous n'aimions pas ses animations de marche donc nous avons changé cela, mais ce n'était pas un travail difficile.

La deuxième fois que nous avons touché aux animations, c'était pour les boutons de notre menu. En effet, de légères animations ont été créé pour les menus. Le reste du temps ce n'était pas vraiment des animations, c'était surtout des effets de particules.

3.4 Management

3.4.1 Site Internet

Au fur et à mesure que le projet évoluait, nous devions faire de même avec un site Web. De cette façon, un site Web a été créé peu après le début du projet. Ensuite, notre objectif était de le faire évoluer à travers l'évolution de notre jeu. Comme nous sommes des élèves internationaux notre site web est entièrement en anglais.

URL du site : `penuts.fr`

Pages de notre site web :

- Home
- Contact page
- Contribution
- A project presentation :
  - CAPZ
  - History
  - Progression
- Links for downloads :
  - The Game (allowing to choose the wanted OS for the download)
  - Reports (to show our reports & documents)
  - Other things used for the project (musics, pictures, videos,...)

Création du Site :

Comme personne dans le groupe n'avait de compétences en codage web au début du projet, nous avons décidé de ne pas recréer la roue. En effet, pour avoir un vrai site web fonctionnant sur toutes les plateformes (ordinateurs, smartphones, tablettes, ...) nous avions besoin de plus de temps. Par conséquent, pour sa création, nous avons utilisé un modèle bootstrap et css pour comprendre comment cela fonctionnait, puis nous l'avons mis à jour en fonction de nos attentes. Comprendre comment le codage Web fonctionnait était intéressant. En effet, nous avons pu découvrir toutes ces langues en les utilisant et en créant de nouvelles fonctionnalités sur notre site (page de contact / menu / boutons / liens ...). L'utilisation de parties de modèles était très instructive et la création d'un bon site web était maintenant dans nos compétences.

Publication du site :

Le but de cette partie était d'envoyer notre site sur Internet et de permettre à quiconque de le trouver avec une URL simple.

De nombreuses recherches ont été effectuées pour comprendre le fonctionnement d'une page Internet. Les principaux concepts que nous devions comprendre pour notre objectif étaient l'hébergement et le nom de domaine. En effet, un site internet doit être hébergé quelque part connecté à Internet avec ses données (contenu des pages). Enfin, cet hôte doit pouvoir être trouvé sur Internet, pour cela les DNS (Domain Name Servers) sont là pour le référencer avec un nom plus simple (quelque chose comme 10.27.384.28 est transformé en penuts.fr). Ces noms compréhensibles sont là pour aider les utilisateurs à trouver une page spécifique.

- Host : Nous avons utilisé GitHub pour héberger notre site Web. Git était l'hébergement le plus approprié disponible. En effet, c'était gratuit, pas difficile à utiliser et permet une mise à jour facile du site.
- DNS : Nous avons utilisé Infomaniak pour acheter un nom de domaine. C'était le moins cher disponible pour avoir une bonne URL. Nous possédons maintenant penuts.fr pour deux ans, ce qui aidera les utilisateurs à trouver notre page de projet.

En le publiant, nous avons également eu l'occasion de comprendre le fonctionnement du référencement de pages de google. En effet, un certain travail a été fait pour référencer notre site web, permettant au "robot de google" de trouver de meilleurs tags ou descriptions.

Contenu de notre site :

La structure du site Web et la publication effectuée, nous pouvions commencer à créer du contenu. Pour cela, nous avons essayé d'ajouter des choses au fur et à mesure que le projet évoluait.

Menu : Le menu est disponible dans toutes nos pages, c'est une barre en haut de la page permettant à l'utilisateur d'avoir un lien vers les autres pages. C'est réactif, cela signifie que cette barre va adapter sa largeur à l'écran. Tout notre site web est réactif, c'est vraiment une fonctionnalité agréable pour l'utilisateur.

Home page : Cette page est la première que le visiteur découvrira. Par conséquent, c'était celle qui évoluait le plus. Elle évoluait en lien avec nos images, notre design et nos autres pages. En effet, cette page contient des liens vers toutes les autres pages, des photos et une présentation rapide du projet.

Contact page : Évidemment, cette page est ici si quelqu'un veut nous contacter. Tout commentaire ou feedback, une question sur le projet ou sur le jeu, cette page est ici pour ces situations. Cette page contient un formulaire PHP permettant au visiteur de nous envoyer un message directement depuis le site. Il a également à sa disposition nos mails juste à côté.
Vous pouvez visiter juste ici : `penuts.fr/contact`

Contribution page : Cette page est là pour permettre aux utilisateurs de signaler un problème, un bug à corriger, un lien mort ou autre chose sur le site, le jeu ou toute autre chose. En fait, cette page contient un document que tout le monde peut remplir. Ce document est structuré pour permettre à l'utilisateur de comprendre où il doit ajouter quelque chose s'il le souhaite. Il peut ajouter, tout ce qui est lié à notre projet (idées, problèmes, bugs et ainsi de suite). Nous avons utilisé `framapad.org` pour créer ce document. C'est un éditeur de texte collaboratif, qui permet à n'importe qui avec le lien de modifier le document en même temps.
Jetez-y un coup d'œil et aidez-nous : `penuts.fr/contribute`

CAPZ : C'est là que vous trouverez des informations sur notre groupe. Ce projet n'aurait pas pu être bien fait si un membre était absent. Découvrez donc chacun de nous, membres de ce groupe de projet : CAPZ.
Again, the link is just here : `penuts.fr/CAPZ`

History & Progression : Comme la page CAPZ, cette page va nous décrire. Dans cette page, vous pouvez en découvrir plus sur notre groupe & notre projet. En effet, derrière ce jeu il y a des gens, plus particulièrement des étudiants. Par conséquent, ce projet n'a pas seulement été fait comme ça, il a un peu d'histoire derrière.
Découvrez cette histoire ici : `penuts.fr/history`

Download - The Game : Cette première page de téléchargement est essentielle. Comme vous l'avez compris, nous avons créé un jeu, alors cette page est là pour le télécharger. Dans son contenu, vous trouverez des liens pour

chaque système d'exploitation (Windows, Linux et macOS). Vous trouverez également toutes les versions de notre jeu : light-version, full-version, old-versions, etc.
Si vous voulez jouer, c'est juste ici : `penuts.fr/download-GAME`

Download - Images : Deuxième page de téléchargement, cette page est clairement une fonctionnalité. Il permet à l'utilisateur d'accéder à toutes les images de notre projet. Dans cette page vous pouvez trouver des logos, des captures d'écran du jeu, des fonds d'écran et d'autres contenus.
Voici le lien pour voir notre galerie : `penuts.fr/images`

Download - Music : Vous voulez encore plus de contenu, vous pouvez découvrir dans cette page la musique et les effets sonores utilisés pour notre jeu. Vous pouvez écouter ces fichiers audio directement sur la page, mais vous pouvez également les télécharger et les utiliser. Pour plus d'informations sur les sons, vous avez une section juste au-dessus.
Écoutez nos musiques juste ici : `penuts.fr/musics`

Download - Videos : Exactement comme la page de musique, vous pouvez trouver des vidéos utilisées pour ce projet. C'est vrai que nous n'avons pas utilisé de vidéos dans notre jeu mais il y a des vidéos pour donner un aperçu du gameplay. En outre, il existe des liens vers de nombreuses vidéos utilisées pour nous aider à terminer le jeu. Principalement des tutoriels Youtube.
Voir plus : `penuts.fr/videos`

Download - Reports : Vous pouvez télécharger ce rapport ! Sur cette page, vous pourrez voir tous nos rapports sur le projet : cahier des charges, premier rapport, deuxième rapport et rapport final. Vous pouvez les lire directement sur la page grâce à une visionneuse pdf mais vous pouvez également les télécharger.
Lisez nos rapports : `penuts.fr/reports`

3.4.2 Aspect Économique

Ce projet était un jeu à but non lucratif. Nous ne gagnerons pas d'argent sur cette création. Le but de ce projet était principalement une contribution de connaissance.
Nous ne voulions pas aller trop loin dans le financement de ce projet. Mais nous avons convenu que si nous devions donner seulement quelques euros pour avoir un meilleur projet final, nous le ferions. Par conséquent, nous en avons utilisé pour le site Web et peut-être pour le CD.

Site Internet : Nous avons décidé d'acheter un nom de domaine pour deux ans (2 €/personne). En effet, comme expliqué dans la partie du site juste au dessus, un joli nom de domaine permet aux utilisateurs de retrouver notre page. Mais nous avons également pris en compte que nous pourrions aussi vouloir partager notre site Web, et pour ce faire, il était plus facile d'avoir un nom de domaine court et agréable.

Rapports : Pour imprimer tous ces rapports, nous devions utiliser beaucoup de papiers et d'encres (total : 200 pages). Comme nous ne savions pas vraiment combien cela coûtait, nous avons décidé d'acheter le petit déjeuner pour celui qui a imprimer les rapports.

CD & Manuel d'utilisateur : Arrivant à la fin de ce projet, la présentation finale arrive également. Pour cette présentation, nous devons arriver avec un boîtier de CD personnalisé, un CD avec le jeu et un manuel d'installation/d'utilisation. Tout le design est fait et le manuel est créé. Mais nous ne les avons pas imprimés pour l'instant. En effet, pour le moment nous avons essayé de trouver un partenariat avec une imprimerie. Pas vraiment concluant, nous verrons juste avant la présentation si nous imprimons tout nous même ou si nous payons pour avoir une très belle boite contenant tout notre jeu.

3.4.3 Ressources

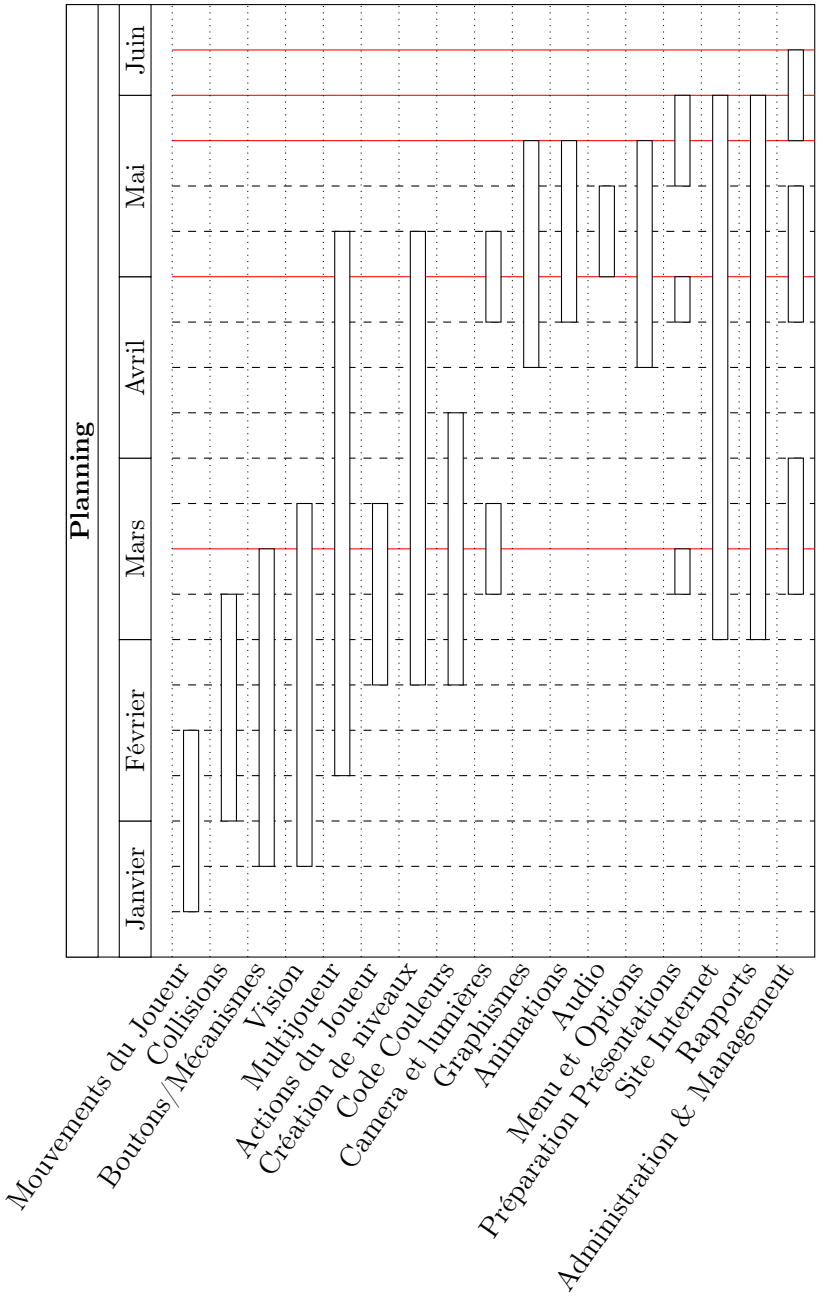
Pour ce projet, nous utilisons principalement des logiciels libres et open-source ou des outils disponibles comme :

- Unity (moteur de jeu)
- Overleaf (éditeur LaTex)
- MonoDevelop, Rider ou VisualStudio (éditeurs C#)
- Blender (animation 3D)
- Assets Store (design & particules)
- Brackets (editeur CSS et HTML)
- GitHub (Organisation de codage et hôte du site Web)
- GitHub Desktop / Kraken (interface GitHub)
- Audacity & BOSCA CEOIL & Computoser (editeurs de musiques)
- Photoshop & Paint (Design)
- Inno Setup (création d'installeurs)
- Internet (Pour la documentation et tout le reste)

Pour la partie équipement, nous avons principalement utilisé ceux déjà acquis :

- Ordinateurs (Pour l'utilisation de tous les outils présentés)
- Imprimantes (Pour l'impression des rapports et des plan de soutenances)

3.5 Progression



4 Conclusion

4.1 Conclusions Personnelles

Ziane LAYADI : J'ai appris à utiliser Unity avant d'entrer chez Epita, j'ai ensuite amélioré mes compétences grâce à beaucoup de vidéos. Dans ce projet, j'ai fait face au travail de groupe qui m'a fait évoluer dans ma façon de monter un projet. En effet, mon habileté à respecter les délais a évolué et être avec des camarades internationaux m'a aidé à améliorer mon anglais. Je n'aimais vraiment pas l'utilisation de GitHub et je pense que la répartition des tâches n'était pas bien faite. Mais même avec cela, nous avons réussi à créer quelque chose de vraiment bien. Le travail d'équipe a finalement été utile. J'ai vraiment aimé travailler sur ce projet. Ce projet symbolise pour moi une étape pour mon entrée dans le monde du travail en termes de travail de groupe. Cette expérience me sera vraiment utile.

Cloé LACOMBE : Je trouve que nous avons bien travaillé ensemble en tant que groupe. Nous nous entendions bien et cela nous a aussi aidés à avoir une bonne ambiance en travaillant. C'est une des raisons pour lesquelles j'ai beaucoup aimé ce projet de groupe. C'était la première fois que je travaillais en tant que membre d'un groupe. Cela m'a aidé à améliorer mes compétences de travail en groupe. En effet, avant ce projet, j'avais vraiment du mal à travailler avec d'autres personnes. J'avais souvent pas le même point de vue sur le projet ou les mêmes idées que le reste du groupe. Cette fois-ci, nous avons travaillé ensemble et discuté des idées, en faisant des compromis au besoin. C'est l'une des principales choses que ce projet m'a appris : j'ai appris à travailler en groupe. J'ai grandi sur de nombreux points avec ce projet :

Tout d'abord, il m'a appris à rester calme : je suis très tempéré et je m'énerve trop facilement. Je me suis rendu compte que ça pouvait poser un problème quand un de nos membres arrêtait d'utiliser github deux jours avant une soutenance, et au lieu d'avoir une conversation calme avec lui, je lui ai crié dessus, ce qui explique pourquoi il a complètement refusé de réessayer GitHub . Cela m'a appris que j'avais besoin de garder mon calme au lieu de m'énerver si facilement.

Deuxièmement, j'ai appris à faire des compromis. Avant ce projet, je voulais tout faire sans compromis sans spécialement demander l'avis des autres. Cette fois-ci, nous avons eu toutes nos idées ensemble, en faisant des compromis et en créant vraiment quelque chose de commun.

Tout le semestre, nous avons dû travailler périodiquement à cause de grosses charges de travail. Cela m'a appris à travailler longtemps et pas seulement pour un projet. J'ai donc appris à planifier mon travail pour inclure des contraintes dans mon emploi du temps. J'ai aussi évolué sur un point de vue technique :

Tout d'abord, j'ai appris à utiliser Unity. C'était très intéressant d'apprendre à utiliser un moteur de jeu. Avec la distribution des tâches j'ai eu l'occasion d'utilisé plusieurs parties de Unity.

Deuxièmement, j'ai dû apprendre la base de GitHub parce que c'est l'outil que nous avons utilisé pour travailler ensemble. J'ai trouvé que c'était très intéressant parce que je sais que nous allons l'utiliser nos prochaines années donc je pense que c'est une bonne chose que j'ai découvert GitHub cette année et appris les bases.

J'ai également appris pendant le codage du projet. J'ai appris à suivre les erreurs et à les corriger. Je suis contente de ce que j'ai fait, en particulier pour le level design et les mécaniques, j'ai dû créer l'idée et comment cela fonctionnerait. Je pense que c'était la partie la plus intéressante.

Finalement, j'ai vraiment aimé ce projet. J'ai aimé comment nous somme partie d'une idée et une fenêtre vide pour finalement avoir un jeu fonctionnel. J'ai appris beaucoup de choses, sur des points humains et techniques, et j'aime aussi beaucoup notre jeu qui est probablement l'une des principales raisons pour lesquelles je suis si contente de ce que nous avons fait.

Alexandre POIRIER-COUTANSAIS : Ce projet était mon premier jeu vidéo "complexe". Même s'il m'a apporté des connaissances vraiment utiles ce n'était pas un projet vraiment intéressant pour moi. J'ai appris à utiliser et à aimer GitHub qui est pour moi l'un des points clés de la programmation de groupe. J'ai appris à l'utiliser sur différentes plates-formes, à la fois via l'interface graphique et le shell. J'ai aussi appris à utiliser Unity même si je ne l'aime pas vraiment, il me semble un peu incohérent. J'ai eu du mal à comprendre comment l'API de Unity est écrite, et je ne l'aime toujours pas. Je n'ai pas beaucoup appris dans la partie codage puisque je suis déjà bon en C#, même si j'ai dû trouver quelques solutions pour que quelque chose fonctionne correctement à cause de la façon dont Unity fonctionne (ce qui ne devrait pas être un problème dans un API). J'ai appris un peu comment gérer un groupe, même si nous avons eu des moments difficiles quand certaines personnes ne comprennent pas et ne veulent pas comprendre GitHub ce qui a beaucoup ralenti la progression du travail. Nous avons correctement

respecté les délais même si j'aurais aimé avoir plus de temps pour jouer avec des choses plus amusantes comme la génération de cartes. J'aurais aimé faire un jeu plus raffiné, mais à cause des décisions de groupe et du temps je n'y suis pas arrivé. Cependant j'ai réussi à faire des niveaux comme je les voulais, j'ai essayé de les rendre aussi propres que possible et de leur donner autant de sens que je voulais, et je suis assez fier d'eux. S'il y a quelque chose que j'aurais fait différemment, c'est la mécanique des signaux, je pense qu'ils pourraient être un peu plus affinés et optimisés, et travailler de manière plus propre, mais avec la façon dont l'API est faite, il devient vraiment difficile de trouver une solution de contournement. Maintenant, je ne vais probablement pas faire un autre jeu vidéo puisque je n'ai pas aimé le faire autant que j'ai aimé travailler sur mes projets personnels. Étant donné que la plupart des étudiants veulent faire un jeu vidéo, je n'avais pas vraiment le choix. Je n'ai pas détesté ce projet, mais je ne l'ai pas aimé autant que ce que j'attendais.

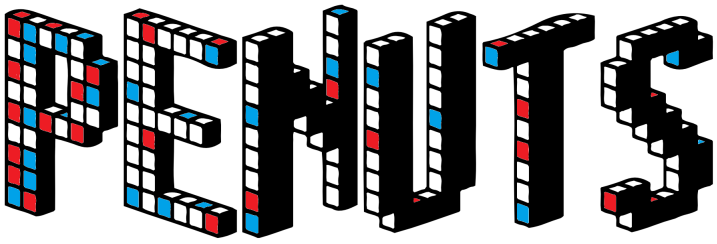
Pierrick MADE : Ce projet m'a vraiment motivé, même si nous n'avons pas pris assez de temps pour organiser des sessions de travail régulières, ce fut une expérience agréable. En effet, c'était la première fois que je devais faire un projet de cette envergure non supervisé.
J'ai appris beaucoup de choses grâce à ce projet, j'ai acquis des compétences humaines et techniques. Le plus évident étant les compétences techniques, je vais commencer avec celles-ci
J'ai découvert des logiciels et des outils comme Unity, Overleaf, Brackets ou GitHub. J'en suis vraiment content car je sais que j'en réutiliserai beaucoup. J'ai également appris comment créer un document en utilisant LaTeX, un très bon outil pour faire des rapports propres. C'était vraiment difficile au début, mais c'est devenu très puissant. Enfin avec la création du site Web, j'ai découvert de nombreuses fonctionnalités des langages HTML et CSS puis appris comment héberger et publier un site web. Savoir que j'ai une page que tout le monde peut voir sur Internet est aussi très gratifiant.
En termes de compétences humaines, j'ai surtout découvert comment superviser et organiser un groupe dans un projet. C'était vraiment difficile de gérer tous les problèmes et toutes les faiblesses découverts au fur et à mesure que le projet évoluait. Mais j'ai compris que c'était une partie du travail.
J'ai trouvé ça vraiment gratifiant de voir ces résultats avec la quantité de problèmes que nous avons rencontrés.

4.2 Conclusion General

Le but de ce rapport était de vous présenter l'évolution de ce projet tout au long de ces six mois. En effet, c'était un long projet et c'est enrichissant de pouvoir le conclure. Ce projet arrivant à sa fin, nous sommes heureux de rendre ce jeu. Ces six mois ont été une véritable expérience de travail pour nous tous. Cela nous a donné de nombreux outils utiles pour l'avenir, techniquement et humainement parlant(décrits dans nos conclusions personnelles). Tout au long du projet, même si nous avons rencontré beaucoup de problèmes, nous avons réussi à respecter les délais des présentations et les contraintes énoncées dans le cahier des charges. Enfin, nous sommes principalement satisfaits de pouvoir rendre ce jeu constituant une grosse quantité de travail.

" Science isn't about WHY,
it's about WHY NOT ! "

J.K. Simmons
Portal 2



Crédits & Sources

Membres du Projet :

Alexandre POIRIER-COUTANSAIS

Pierrick MADE

Cloé LACOMBE

Ziane LAYADI

Remerciements :

ACDCs

Krisboul

Sources :

Documentation Unity

Documentation C#

Assets store & Blender

Computoser

Documentation LaTeX

Documentation HTML/CSS

Tutoriels youtube

Beaucoup de jeux-vidéos