Université du Québec à Chicoutimi

Rapport de Projet

Trimestre d'Automne 2017

BOURDELAS Pablo - BOUP07029506
DESROUSSEAUX Florian - DESF28039508
DUPORT François - DUPF18029406
PELLICER Marion - PELM12599502

Sommaire :

Introduction : 2
1. Rappels sur le concept et les mécaniques de jeu 3
1.1. Concept de base 3
1.2. Mécaniques de jeu 3
2. Présentation des interfaces 7
2.1. Menu de démarrage 7
2.2. Le menu de Pause 8
2.3. Interface du jeu 8
3. Diagramme de Classes 9
3.1. Menus et Scènes 9
3.2. Eléments de Jeu 10
3.3. Interactions 11
3.4. 11
4. Diagramme des tâches et organisation 12
5. Choix de conception et de technologies 14
6. Tests effectués 15
7. Visuels du Jeu 16
Conclusion 16

Introduction :

Ce document a pour but de décrire le travail effectué lors de la production du jeu « Doreimi ! » par notre équipe et de donner une appréciation générale du jeu et de la qualité de notre travail au lecteur. Il s'agit d'expliquer nos réalisations mais également notre cheminement intellectuel et nos choix de développement.

Dans ce but nous commencerons par rappeler le concept ainsi que les mécaniques du jeu avant de présenter l'interface. Nous détaillerons ensuite les diagrammes de classes utilisés puis notre organisation ainsi que nos choix techniques. Nous finirons par expliquer les tests que nous avons fait et nous inclurons des visuels du jeu.

1. Rappels sur le concept et les mécaniques de jeu

1.1. Concept de base

Dans DoreiMi!, le joueur incarne une star de la pop japonaise qui doit voyager à travers le Québec pour se rendre au *Cosmic Nihon Festival*.

DoreiMi! est un jeu de plateforme en temps réel accès sur la résolution d'énigmes. Le tout se place dans un univers faisant un lien entre le Japon et le Québec, avec un style graphique de type pixel-art.

Le joueur contrôlera le personnage d'Yhdol qui placera et donnera des ordres à ses fans pour permettre la résolution des puzzles et l'évolution dans l'histoire. C'est un jeu avec un contexte musical important qui sert à accentuer l'immersion du joueur dans le jeu et à dynamiser le gameplay.

1.2. Mécaniques de jeu

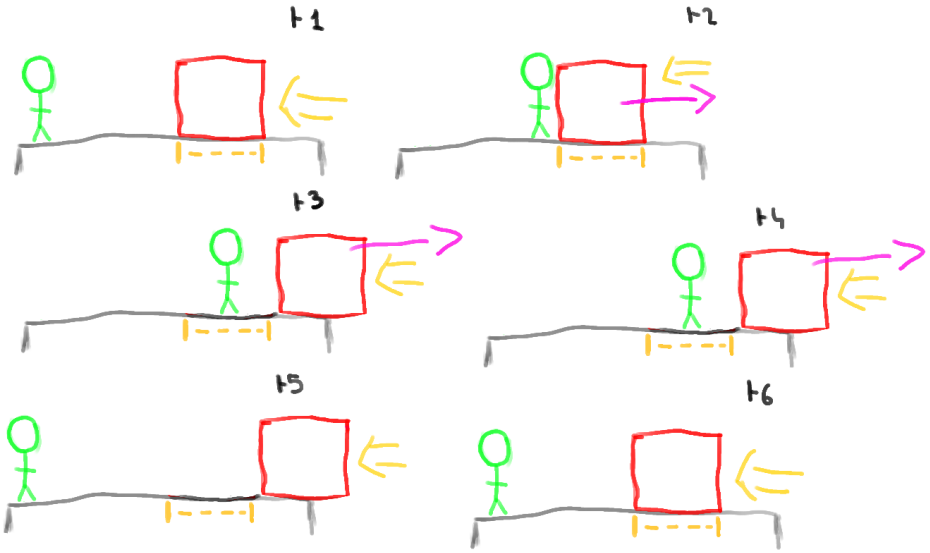
Même si la physique s'éloigne de la réalité pour les actions des otakus, le reste des comportements est basé sur une physique réaliste. Les personnages ainsi que les objets sont soumis à la gravité de façon standard.

Les personnages peuvent se déplacer latéralement et sauter.

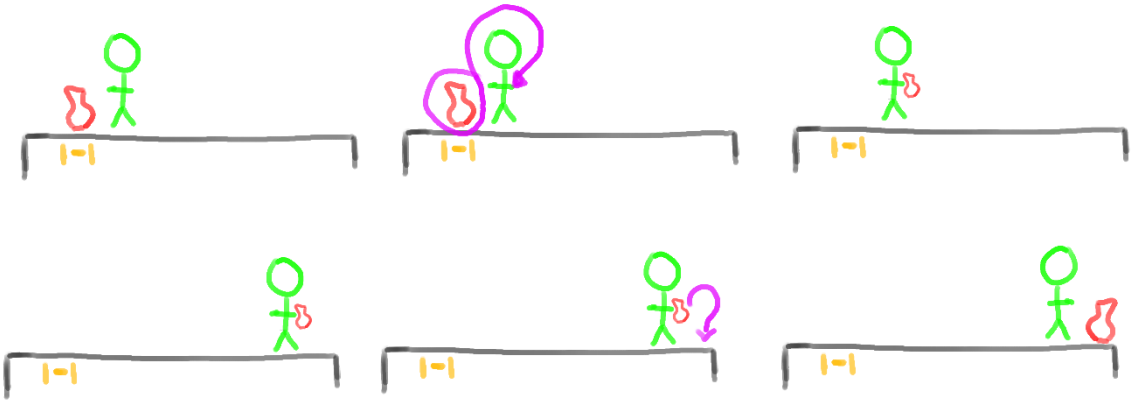
Yhdol possède la capacité d'activer des interrupteurs et a un inventaire d'otakus disponibles qu'elle peut utiliser.

Les otakus possèdent des actions différentes en fonction de leur type :

- Le Doh peut pousser des objets lourds ainsi que les maintenir.



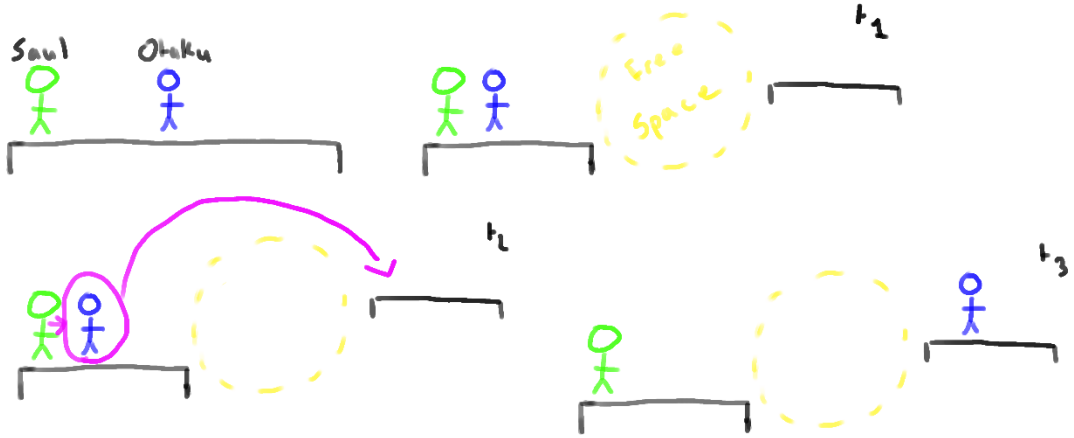
- Le Rei peut porter des petits objets.



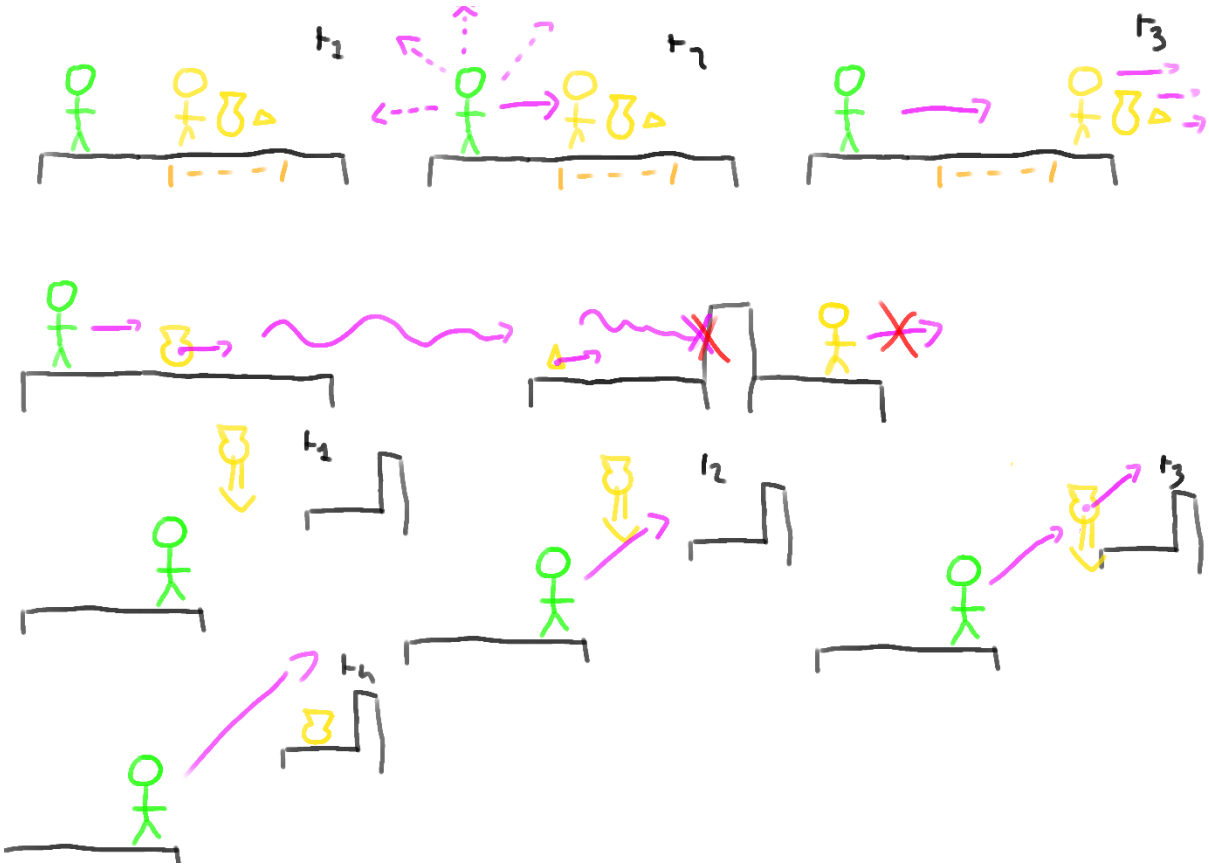
- Le Mie peut téléporter Yhdol près de n'importe quel Pha. Il ne sert à rien si aucun Pha n'est placé.

- Le Pha sert de balise pour la téléportation d'Yhdol par un Mie.

- Le Saul peut lancer un otaku. Le lancer est possible si on dispose d'une ligne droite sans obstacle entre l'origine et la cible.


Saul
Otaku
Free space
t_1
t_2
t_3

- Le La peut créer des courants d'airs poussant les personnages et les objets légers.
  La force créée par le courant d'air est de 0.8g (La force ne doit pas permettre de pouvoir voler ou léviter), pour Yhdol. Les objets sont poussés avec grande force sans tenir compte de la gravité.



- Le Xi peut se spécialiser pour utiliser la capacité de n'importe quel autre otaku. Une fois la spécialisation choisie, elle est définitive.

La résolution des énigmes se fait en activant des interrupteurs qui permettent d'activer la zone de sortie.

2. Présentation des interfaces

2.1. Menu de démarrage

Comme le montre l'image ci-dessus, le menu de démarrage du jeu est très simpliste. Le background du menu représente un paysage sous une nuit étoilée. Les couleurs froides présentent sur l'image fournissent un côté mystérieux bien approprié à notre jeu puisque le personnage incarné par le joueur se retrouve dans l'obligation d'explorer différents lieux pour atteindre ces objectifs. L'image a été choisi de telle manière à ce que le joueur puisse avoir une première idée de l'univers du jeu.

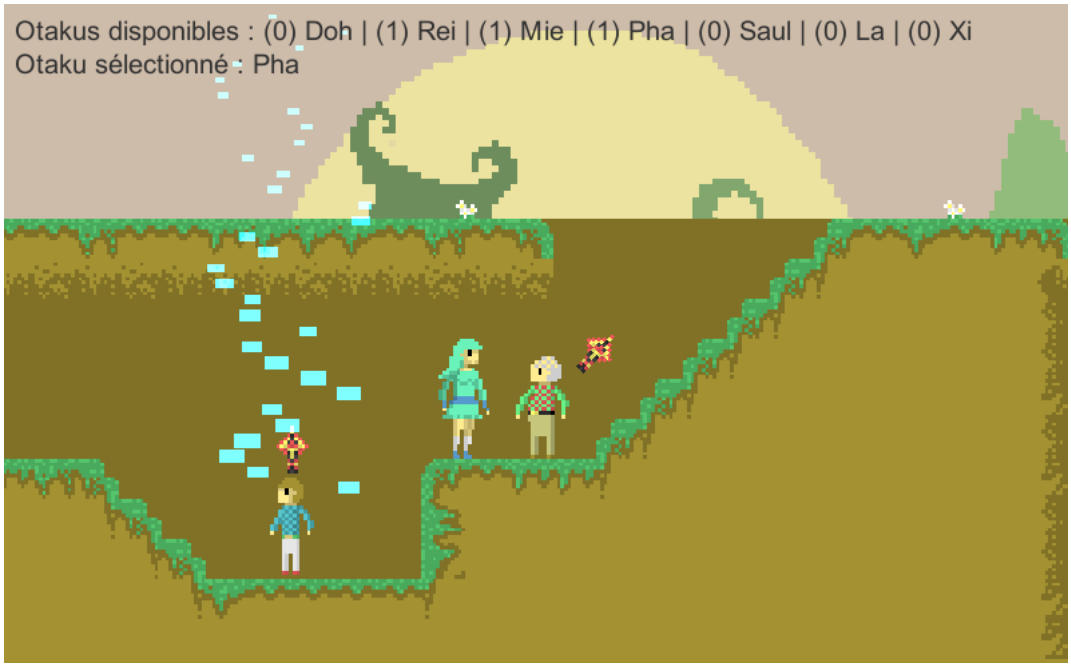
Ensuite nous avons les boutons de sélection « Continuer la partie », « Jouer » et « Quitter ». Le premier permet au joueur de continuer le jeu au dernier niveau auquel il s'est arrêté. Le bouton « Jouer » peut être utiliser pour charger n'importe quel niveau qui a déjà été accédé par le joueur afin que celui-ci puisse s'il le désire recommencer un niveau qu'il avait déjà validé. Enfin le dernier bouton met fin au jeu et ferme le programme.

Pour terminer le nom du jeu apparaît au milieu supérieur de l'écran afin que ce soit la première chose que le joueur voit lorsque le jeu se lance.

2.2. Le menu de Pause

Tout aussi sobre, le menu de pause permet différentes actions. Depuis celui-ci il est possible d'activer ou désactiver le son du jeu, de reprendre la partie, de recommencer le niveau depuis le début dans le cas où le joueur penserait avoir fait une erreur et enfin de quitter la partie pour retourner au menu de démarrage.

2.3. Interface du jeu

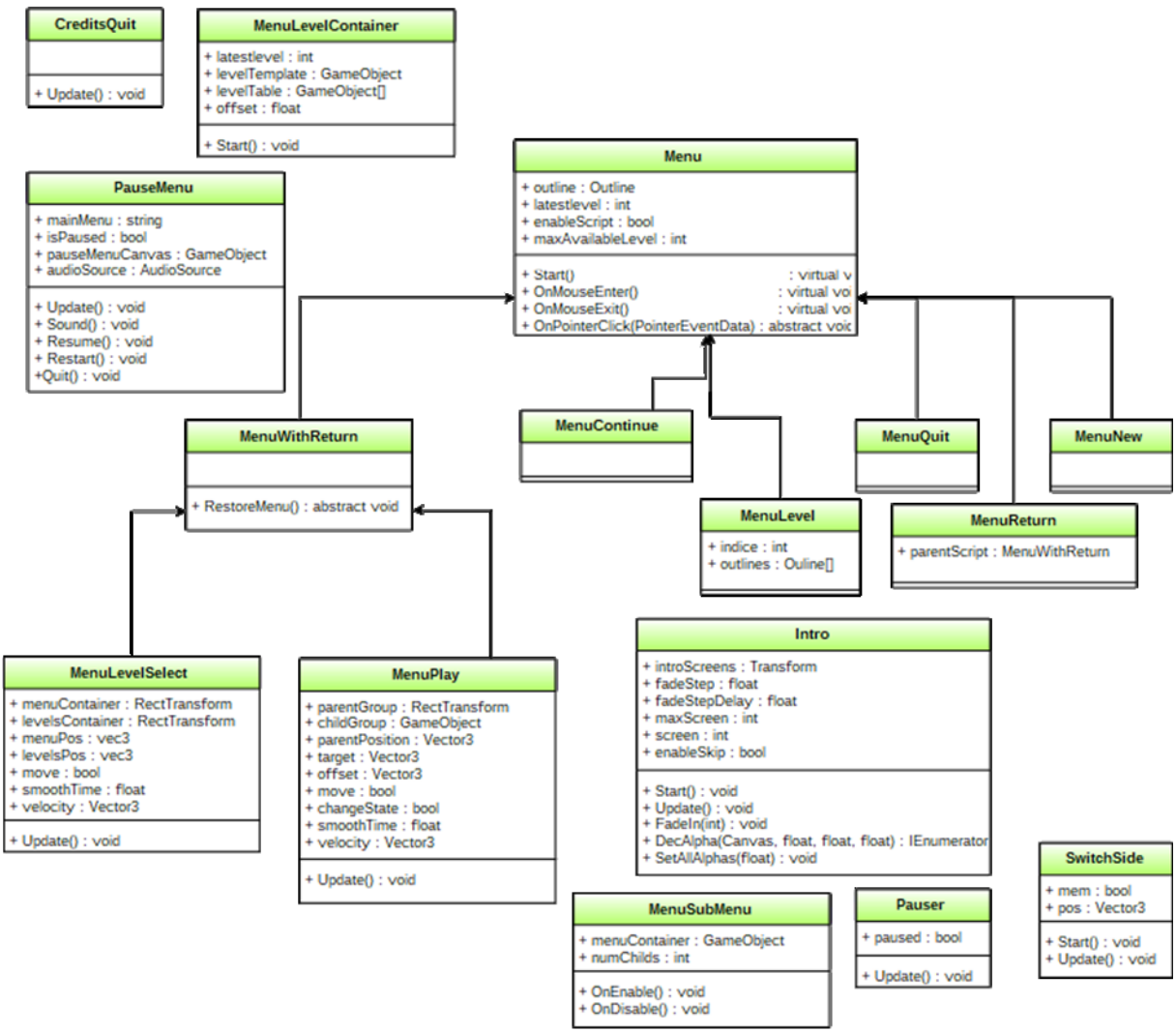


Pour ce qui est de l'interface in-game, les choix qui ont été fait ont été les suivants. Nous n'avions pas besoin de placer beaucoup d'information sur l'écran moins on en mettait et mieux c'était. Au final nous étions tous d'accord pour dire que le joueur n'avait besoin que de connaître l'état de son inventaire ainsi que l'Otaku actuellement sélectionné pour pouvoir progresser aisément dans le jeu.
Du coup pour l'inventaire, nous avons listé l'ensemble des Otakus disponibles dans le jeu avec à côté le nombre disponible dans l'inventaire. En dessous de la barre

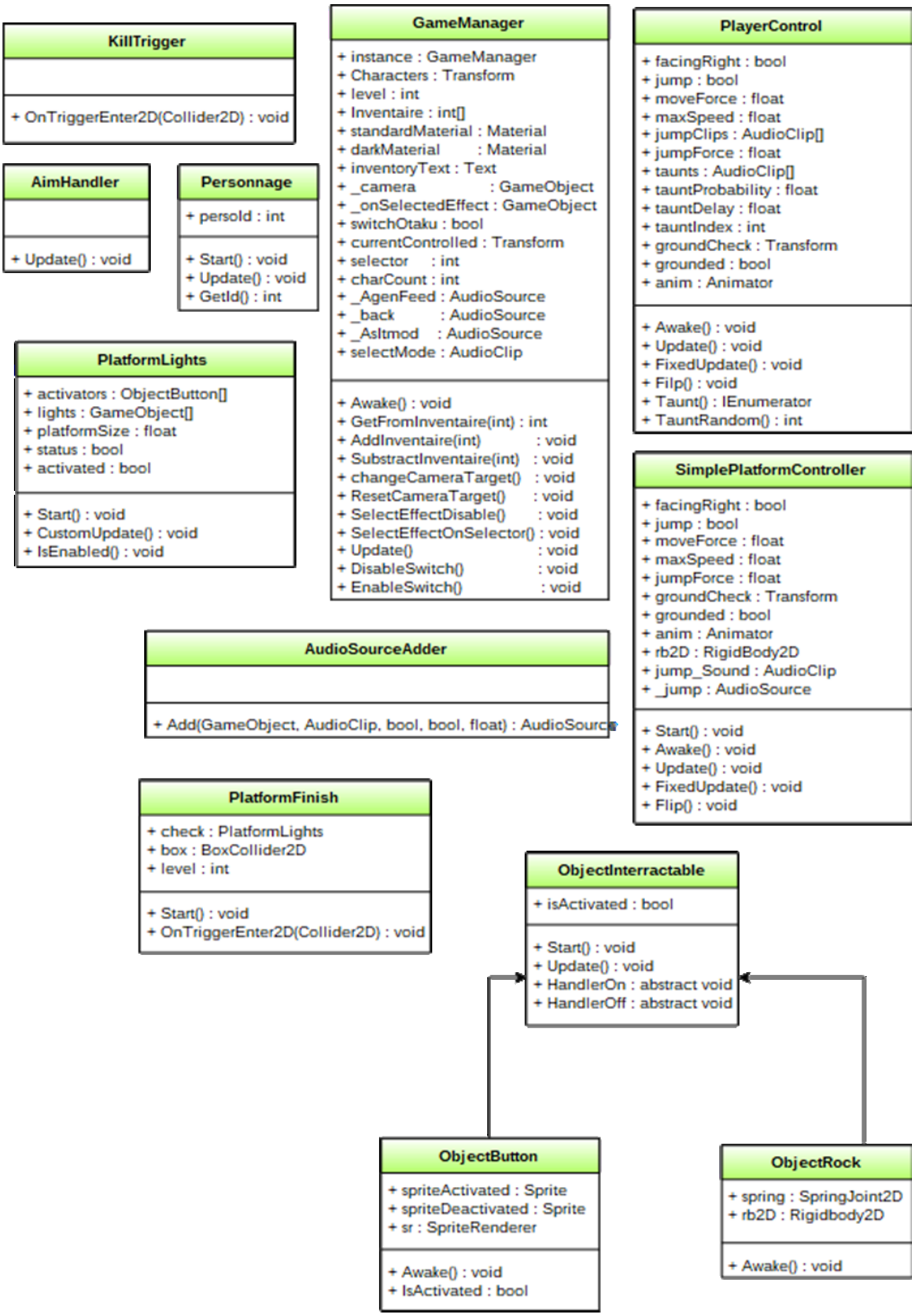
d'inventaire le nom de l'Otaku actuellement sélectionné apparaîtra pour que le joueur soit sûr de sa sélection.

3. Diagramme de Classes

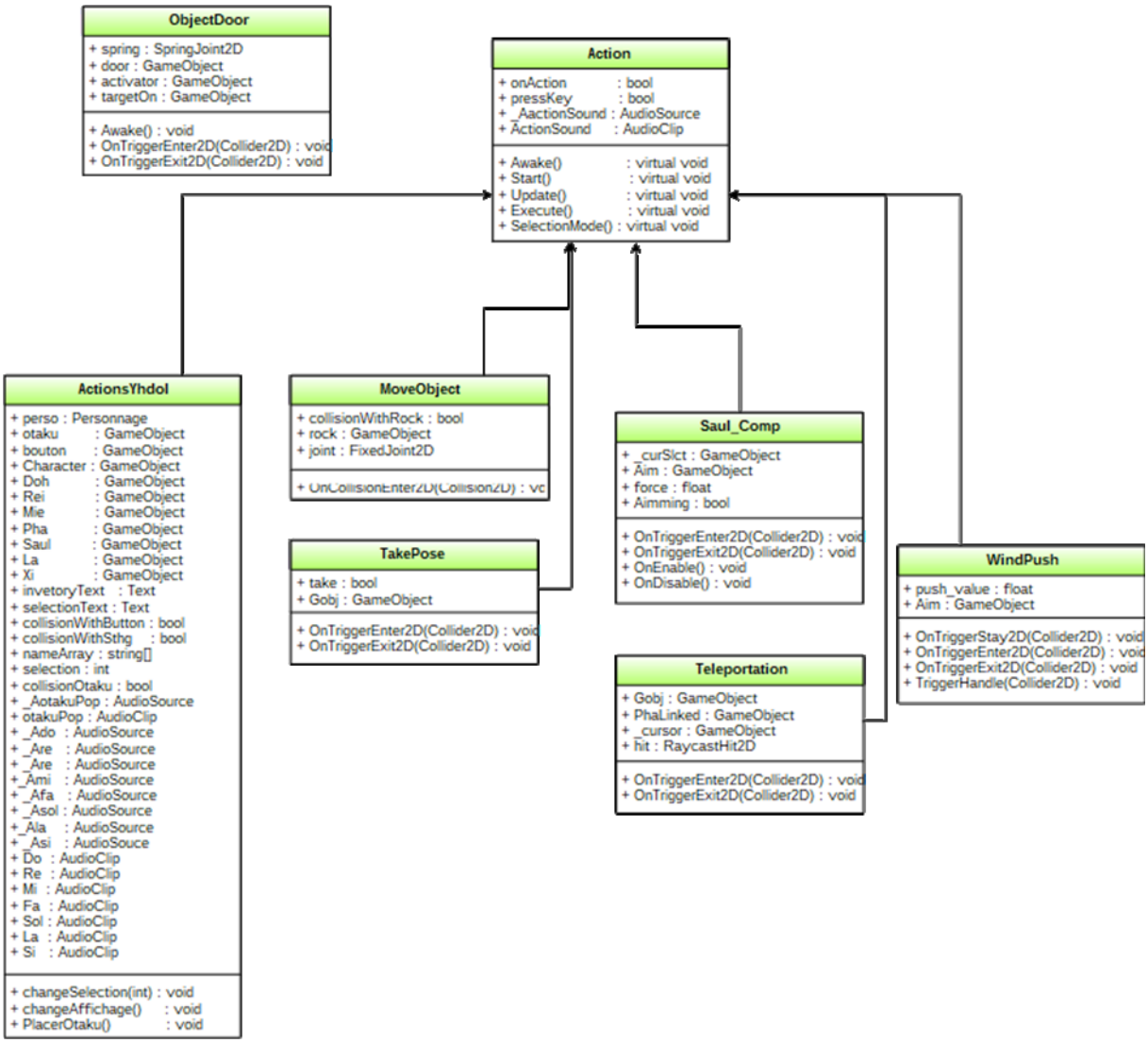
3.1. Menus et Scènes



3.2. Eléments de Jeu



3.3. Interactions



4. Diagramme des tâches et organisation

Nous nous sommes organisés de manière pyramidale. C'est-à-dire que plutôt que de donner à une personne la tâche de réaliser un bloc, nous avons dispatcher les tâches de manière à ce que personne n'ait à attendre la réalisation d'une tâche pour avancer de manière optimale. Ainsi les mécaniques de base et un environnement de test ont été notre priorité lors du premier sprint afin de pouvoir élargir la quantité de tâches exécutables.

Nous pouvons voir ci-dessous le tableau qui énumère les tâches exécutées et leur exécutant.
Plutôt que de décrire qui a fait quoi par catégorie, il est plus efficace de parler des domaines d'activités par personne.

Ainsi Pablo BOURDELAS, fut lors de ce projet notre graphiste. Il est responsable de tous les assets graphiques du jeu. Il a également conçu et mis en place les niveaux ainsi que créé certains Otakus. A la fin il s'est attelé au réglage de la caméra et aux effets sonores du jeu.

Florian DESROUSSEAUX a lui été en charge des objets de jeu (murs, rocher, etc) et de la gestion des scènes, ainsi que du menu principal. Il a également implémenté la sauvegarde de la progression et la possibilité de finir les niveaux. Il s'est aussi chargé de gérer la possibilité de changer de personnage contrôlé.

François DUPORT a réalisé l'implémentation du personnage et de ses scripts d'interaction. Il a également mis en place le GameManager comprenant l'inventaire ainsi que le dépôt et la récupération des Otakus. Il s'est aussi chargé de choisir et de mettre en place les musiques.

Marion PELLICER s'est chargée de créer la plupart des Otakus incluant leurs interactions avec l'environnement. Elle a également mis en place tout ce qui a trait au menu pause.

Tâche	Exécutant
Déplacement personnages	
Gestion Inventaire Player	DUPORT François
Depot et récupération Otakus	DUPORT François
Mode Selection	DUPORT François
Prefab pour Personnages	DUPORT & BOURDELAS
Scripting interrupteur	DESROUSSEAUX Florian
Scripting rocher	DESROUSSEAUX Florian
Scripting Mur mobile	DESROUSSEAUX Florian
prefab Objet	DESROUSSEAUX Florian

Game Manager	DUPORT François
Affichage Inventaire Player	DUPORT & DESROUSSEAUX
Changement personnage	DESROUSSEAUX Florian
Script Action (partagés par tous les persos)	PELLICER Marion
Script MoveObject (Doh)	PELLICER & DESROUSSEAUX
Script TakePose (Rei)	PELLICER Marion
Script Teleportation (Mie & Pha)	PELLICER Marion
Design des personnages	BOURDELAS Pablo
Animation des personnages	BOURDELAS Pablo
Création de tiles de base et d'objets	BOURDELAS Pablo
Image de fond de niveau	BOURDELAS Pablo
Création du niveau 1	BOURDELAS Pablo
Script viseur	BOURDELAS Pablo
Script Wind (La)	BOURDELAS Pablo
Script Push (Saul)	BOURDELAS Pablo
Menu Principal	DESROUSSEAUX Florian
Sauvegarde de l'avancement	DESROUSSEAUX Florian
Menu Pause	PELLICER Marion
Sélection des musiques	DUPORT François
Insertion des musiques dans le jeu	DUPORT François
Finition des otakus	Toute l'équipe
Introduction	DUPORT & DESROUSSEAUX
Possibilité de finir les niveaux	DESROUSSEAUX Florian
Ecran de fin de jeu	DESROUSSEAUX Florian
Mouvement de la Camera	BOURDELAS Pablo
Ajout de divers Feedback (Visuel et Sonore)	BOURDELAS Pablo
Tilling du lvl 2	BOURDELAS Pablo

5. Choix de conception et de technologies

Pour les choix technologiques, nous avons opté pour un développement sous Unity pour différentes raisons. Tout d'abord, les membres de l'équipe n'avaient pas tous une première expérience dans le développement de jeux et Unity est assez rapide à prendre en main et fourni de bon résultats. De plus, pour ceux qui avaient de l'expérience, celle-ci était sous Unity c'était donc un choix tout à fait légitime que de partir là-dessus. Enfin, sachant que nous recherchions à faire un jeu d'énigmes en 2D, nous savions que Unity était bien approprié pour ce genre de jeu-là.

Nous avons choisi de créer une structure hiérarchique pour les différents menus, les objets ainsi que les actions nous allons voir pourquoi de tels choix ont été faits. Pour les menus, nous avons une classe mère contenant tous les attributs et classes propre à chaque menu que nous allons créer. La classe mère *Menu* est une classe abstraite, elle ne sert que de modèle pour les menus qui hérite d'elle. Chaque menu possède donc un entier *latestlevel* permettant de connaître le dernier niveau accédé, un booléen *enableScript* qui selon sa valeur affichera le menu à l'écran puis un dernier entier *maxAvailableLevel* qui sera utilisé afin de vérifier que le niveau existe et donc aussi permet de signaler la fin du jeu. Les méthodes communes à tous les menus sont la méthode *Start()* initialisant les paramètres du menu, les méthodes *OnMouseEnter()* et *OnMouseExit()* pour détecter la présence du curseur de la souris dans le menu et les différents boutons qui le compose. Enfin une dernière méthode *OnPointerClick(PointerEventData)* déclenché par un clic de souris, l'action associée au clic est donc définie dans cette méthode.

Pour les objets avec lesquels le joueur doit pouvoir interagir, nous avons également choisi de faire un héritage de classe. La classe mère étant la classe *ObjectInterractable*. Très basique, cette classe comprend un booléen *isActivated* indiquant si l'objet est en cours d'utilisation ou non, les méthodes *Start()* et *Update()* mais également deux autres méthodes *HandlerOn()* et *HandlerOff()*. Ces dernières contiennent les comportements que les objets doivent suivre lorsque pour la première, une action est effectuée sur l'objet, et pour la seconde, lorsque l'action est stoppée. Nous avons pris la décision de mettre en place ces deux méthodes car nous voyions les objets comme une entité du jeu à part entière dont le comportement est dirigé par la prise en main de cet objet par les différents personnages du jeu.

Enfin nous avons la famille des actions qui sont l'ensemble des scripts associés aux personnages. Nous remarquons que la classe mère Action possède une *AudioSource* et un *AudioClip*. En effet, nous voulions que chaque Otakus possède son propre effet sonore à savoir le son d'une note de musique, celle dont il porte le nom. Nous nous sommes mis d'accord pour dire qu'un personnage équivaut à une action, nous n'aurons jamais plus de classes héritées d'Action que de personnages. A savoir que la méthode *Execute()* est appelée par *Update()* et va « exécuter » l'action en question, c'est dans cette méthode aussi que pourront être appelées les méthodes *HandlerOn()* et *HandlerOff()* des objets.

6. Tests effectués

Lors du développement, nous nous sommes assurés au fur et à mesure que chaque fonctionnalité ne présentait pas de défaut. Nous avons pour ce faire créer une scène qui forçait l'utilisation des différentes fonctionnalités lors que nous en rajoutions une. Cela a permis de tester les différents ajouts dans différents contextes autre que la simple utilisation.
Lorsque nous avons retravaillé ce jeu pour le rendu final, nous avons également demandé à des connaissances de tester le jeu pour avoir un retour d'expérience d'utilisateurs hors du projet et qui donc ne teste pas les fonctionnalités dans le cadre absolu de leur utilité, cela nous a permis de trouver différents bugs et donc de les corriger.
Ainsi nous nous sommes également assurés que le jeu était jouable pour une personne ne connaissant rien du jeu en dehors des contrôles.

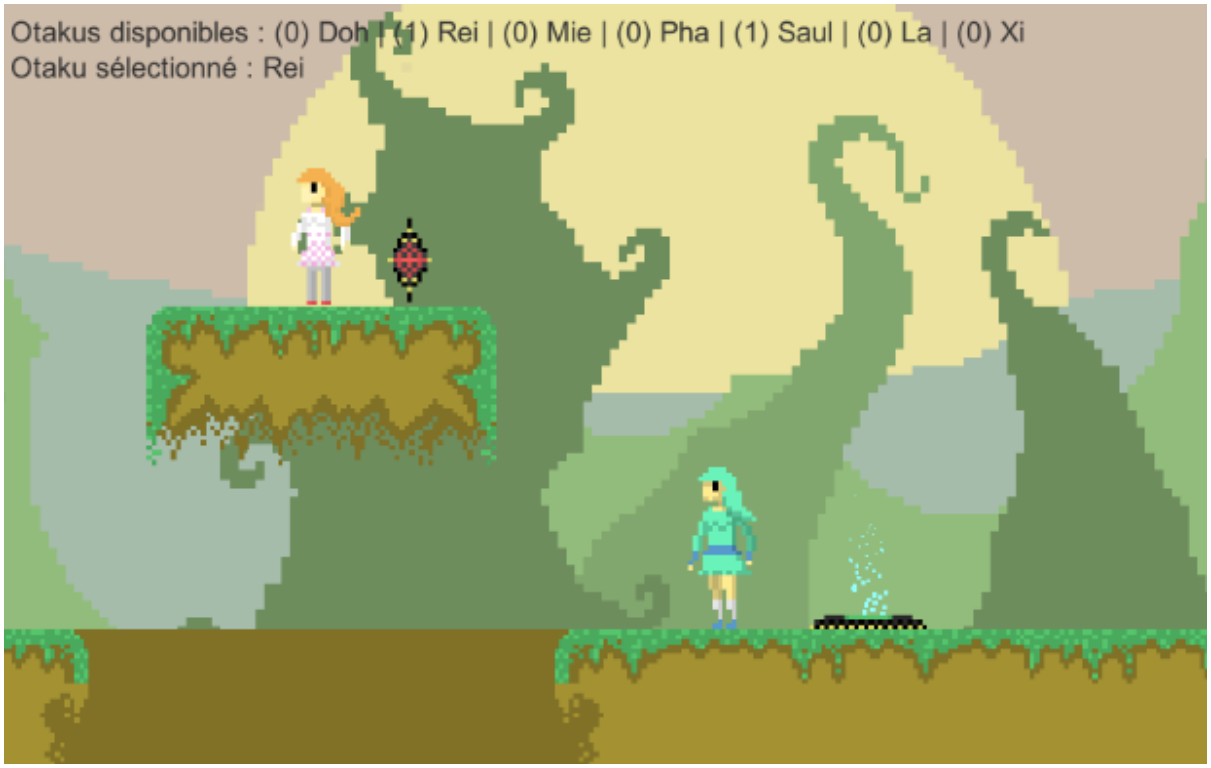
7. Visuels du Jeu

Nous avons produits plusieurs visuels pour le jeu en allant de la boite et le poster ainsi que quelques images en jeu.

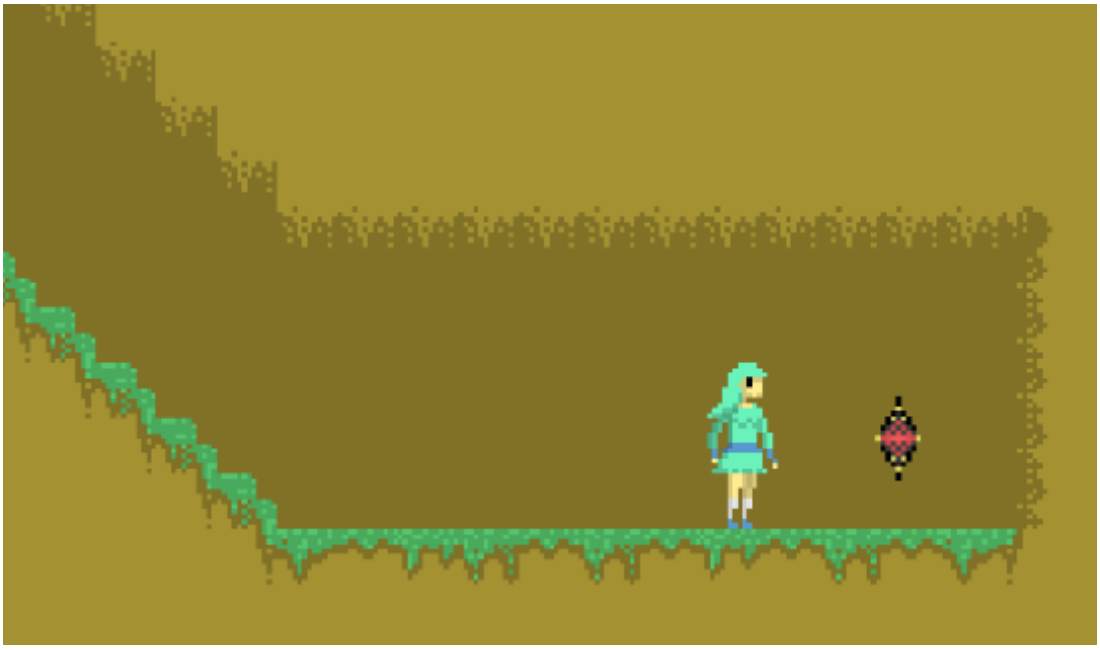
Le poster

La jaquette /boite

Une image du niveau 1 avec plusieurs otakus en action



La fin du niveau 1



Le bouton d’activation du portail de fin de niveau

La fin du niveau 2

Conclusion

Ainsi nous sommes satisfaits du travail effectué. Pour le futur nous pensons que l'ajout des plusieurs niveaux et de niveaux bonus rendra le jeu pleinement opérationnel et intéressant pour le public. L'approfondissement de l'histoire permettra d'augmenter l'attractivité du jeu. Même si encore du développement est nécessaire pour déployer le jeu pour le public, les tâches restantes sont mineures et prendront moins de temps que le temps de développement.