

8INF955

Principe de conception et développement de jeux vidéo
Automne 2017

Rancor of the Titan

Rapport final

Lorane Lebrun - LEBL11579508
Antonin Celestin - CELA19059503
Pol Enault - ENAP30079605
Martin Rohmer - ROHM04119503

UQAC
Université du Québec
à Chicoutimi

Sommaire

Introduction - rappel du concept et des mécaniques de jeu	2
1. Présentation des interfaces	2
1.1. Les menus	2
2. Diagramme de classes et structure du projet	5
2.1. Structure du projet	5
2.2. Diagramme de classes	6
2.2.1. Menus	6
2.2.2. Pièges et pouvoirs	6
2.2.3. Général	7
3. Répartition des tâches dans l'équipe	8
4. Justification des choix de conception et des technologies utilisées	8
4.1. Choix des technologies	8
4.2. Choix de conception	9
5. Explications de nos mécaniques de jeu	9
5.1. Le Drag&Drop des pouvoirs	9
5.2. Réactions des pièges aux actions du joueur, réactions d'Atalante en fonction de l'état des pièges	10
6. Description des tests effectués	12
Conclusion - retours post-mortem	13

Introduction - rappel du concept et des mécaniques de jeu

Rancor of the Titan propose une expérience nouvelle : dans notre jeu le joueur n'a aucun contrôle sur le personnage et doit lui faire atteindre ses objectifs en agissant uniquement sur l'environnement. C'est un *side-scrolling puzzle platformer* qui prend place au cœur des temples de la mythologie grecque.

L'objectif final est de détrôner Zeus, le dernier gardien de l'Olympe, en gagnant des nouveaux pouvoirs à chaque niveau. Chaque pouvoir permet de contrôler une partie de l'environnement et de permettre à Atalante (le personnage que l'on suit) de déjouer les pièges qui protègent les temples. Ainsi, le joueur peut contrôler la pluie, le vent et la croissance des végétaux. Les pièges qui se dressent sur son chemin sont des murs, des fosses sans fond et d'autres dotés de piques à leur base. Pour utiliser un pouvoir, le joueur *drag&drop* le bon pouvoir au bon endroit et au bon moment (il clique sur l'icône du pouvoir, maintient le clic tout en déplaçant l'icône à l'endroit souhaité et relâche).

1. Présentation des interfaces

1.1. Les menus

Quatre menus sont présents au sein de ce prototype : le menu principal, le menu de pause, le menu de victoire et le menu de défaite. L'habillage de ces différents menus est le même, qu'il s'agisse de l'image d'arrière-plan représentant un temple grec ou bien de l'aspect des boutons et du texte. Les différents menus ont été uniformisés ainsi afin de conserver une ambiance et une logique cohérente dans le jeu.

Les boutons ont tous la même couleur vert clair. Placer son curseur au dessus d'eux change leur couleur en vert foncé pour tous les menus, à l'exception du menu principal. La musique des menus correspond à la musique du jeu à l'exception de celle du menu principal. Cliquer sur un bouton déclenche un effet sonore pour nous confirmer que le clic a bien eu lieu. À l'exception de deux boutons, cet effet sonore est le même pour tous.

Le menu principal (voir la figure ci-dessous) est un passage obligatoire pour démarrer le jeu. Il consiste, contrairement aux autres menus, en une scène indépendante du jeu. Initialement il devait permettre d'accéder à une interface de sélection du niveau. Dans le cas de notre prototype cependant, un unique niveau est disponible qui se lancera sans choix préalable. Pointer l'un des boutons déclenche une animation grossissant le bouton et un changement de couleur.

Le bouton *Start* permet de démarrer le jeu. *Help* ouvre une interface contenant un texte qui explique le fonctionnement du menu. Cette interface pourrait être enlevée, ou servir à autre chose ; nous avons donc décidé de la conserver pour notre prototype. Le bouton *Quit* permet de quitter le jeu.

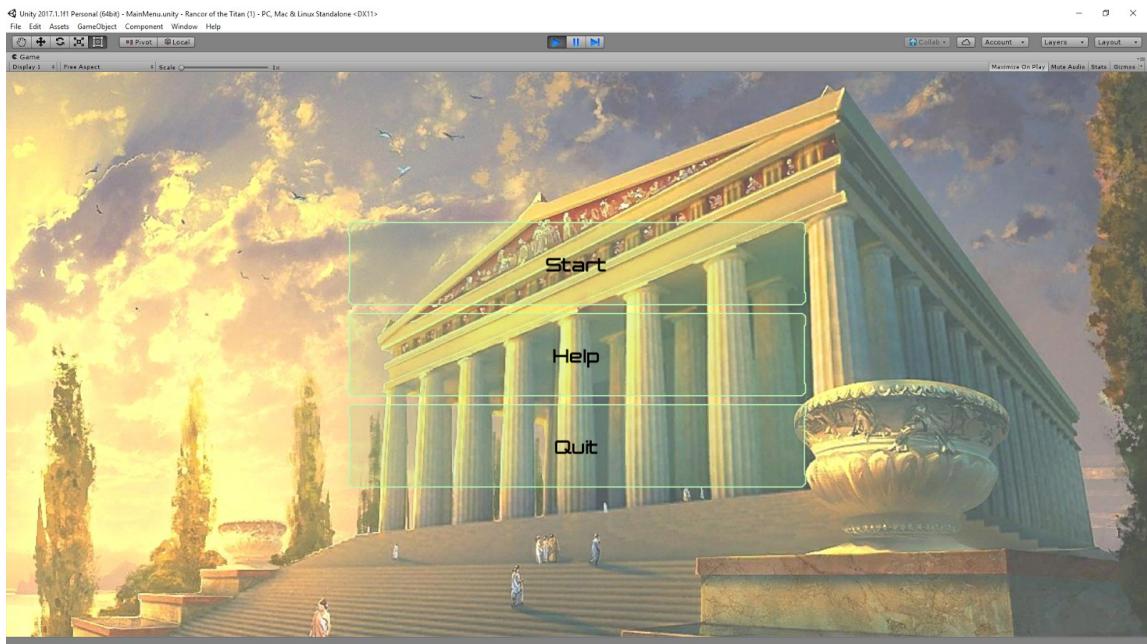


Figure 1 : menu principal

Appuyer sur le bouton *Start* déclenche un effet sonore unique signifiant le démarrage du jeu. Une transition de l'écran en blanc accompagne également cet effet. La scène du jeu est ensuite chargée, et le niveau lancé automatiquement.

Appuyer sur le bouton *Help* déclenche l'effet sonore par défaut des boutons des menus et une transition de droite à gauche de l'interface. L'interface principale disparaît à gauche tandis que l'interface d'aide apparaît de la droite. Appuyer sur le bouton *Back* déclenche la transition inverse et le même effet sonore.

Appuyer sur le bouton *Quit* déclenche un effet sonore unique signifiant l'arrêt du jeu. Cet effet s'accompagne également d'une transition en noir de l'écran. Les processus associés au jeu s'arrêtent et l'écran du joueur est ramené au bureau.

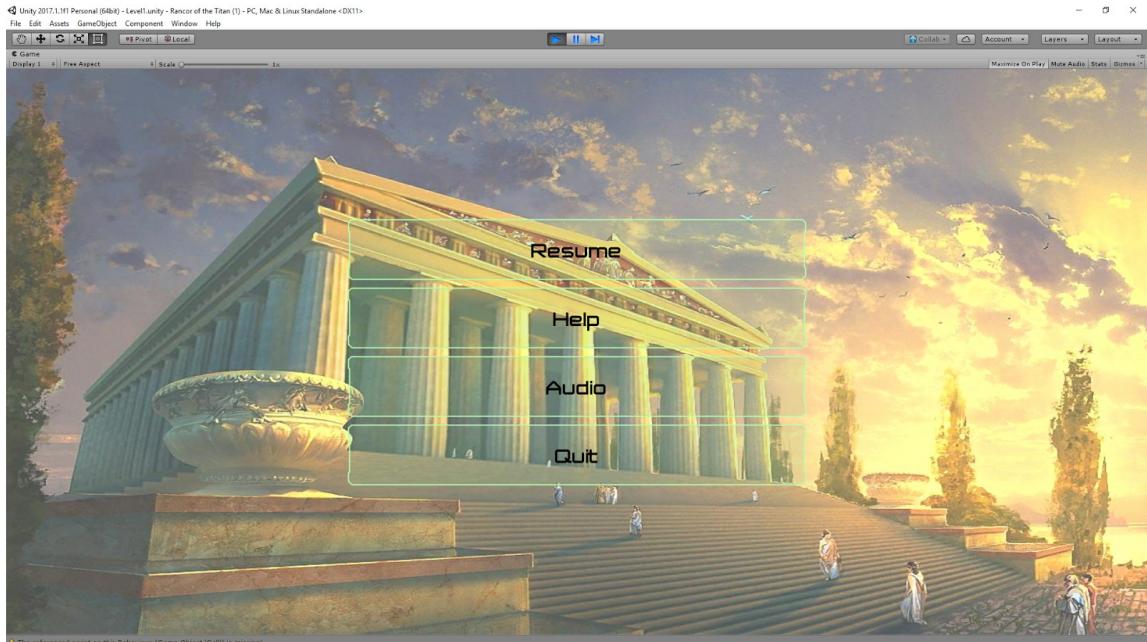


Figure 2 : menu de pause

Le menu de pause est accessible en appuyant sur la touche *Esc* du clavier. Après discussion sur le placement d'un bouton d'accès à ce menu dans l'un des coins supérieurs de l'interface de jeu qui conviendrait mieux à un jeu sur plate-forme mobile, nous avons convenu de laisser l'accès à ce menu grâce à l'appui de la touche *Esc* dans un souci pratique.

Ce menu comme son nom l'indique déclenche une pause du jeu lors de son invocation. Il est composé de quatre boutons : *Resume* qui ré-affiche l'interface de jeu et relance son exécution ; *Help* qui explique brièvement le fonctionnement du jeu ; *Audio* qui permet d'accéder au contrôle sonore du jeu ; *Quit* qui permet de quitter le jeu.

L'interface sonore est accessible via le bouton *Audio* et présente trois sliders (barres disposant d'un curseur qu'on nommera *pastille*) et un bouton *Back* ayant le même comportement que ces homonymes. Les trois sliders permettent de contrôler les différents volumes sonores du jeu. Le placement de la pastille sur le slider indique le volume du son correspondant. Tout à gauche pour un volume nul, tout à droite pour un volume maximal (équivalent à 1). Le déplacement de ces pastilles modifient la valeur de l'*audio source* correspondant, entre 0 et 1 pour les deux du bas : *Music Volume* et *Sound Effects Volume*. Celui du haut, *Master Volume*, contrôle le volume principal du jeu, modifiant la valeur de l'*audio listener*. Un changement de couleur du bleu au blanc de la pastille indique que l'on est en train de la déplacer avec son curseur.

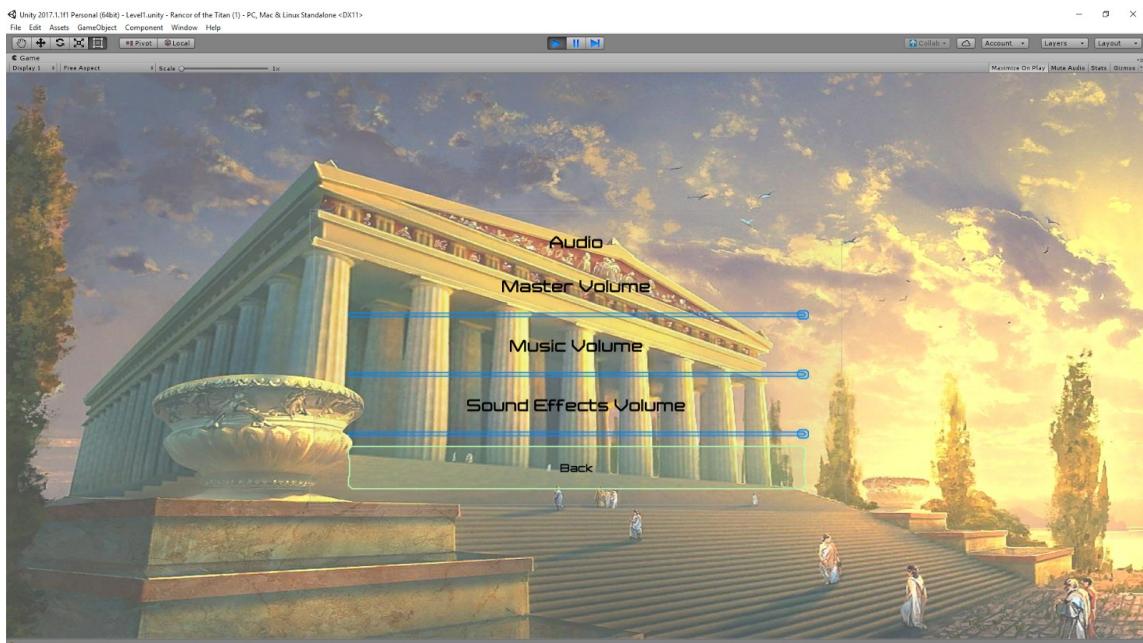


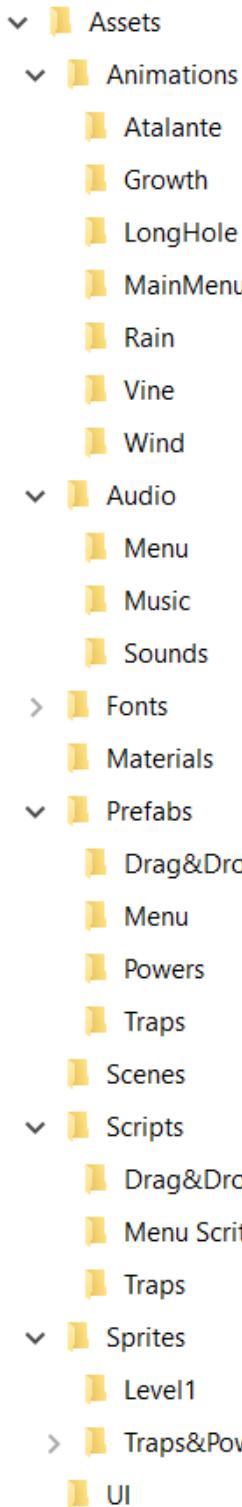
Figure 3 : interface des réglages sonores

Les menus de victoire et de défaite ne se distinguent l'un de l'autre que par le contexte dans lequel ils sont appelés et le texte les accompagnant. Le menu de victoire apparaît une fois que Atalante passe la porte de fin du niveau et présente un texte dans sa partie supérieure indiquant son succès au joueur. Le menu de défaite est invoqué quand Atalante meurt et son texte précise sa défaite au joueur.

Les deux menus présentent un bouton *Restart* qui permet de redémarrer le niveau et un bouton *Go to main menu* ramenant au menu principal. Ceci dans une volonté de pouvoir choisir son niveau même si cela n'est pas disponible sur ce prototype.

2. Diagramme de classes et structure du projet

2.1. Structure du projet



À gauche se trouve la hiérarchie des fichiers du projet, contenu dans le dossier principal nommé Assets.

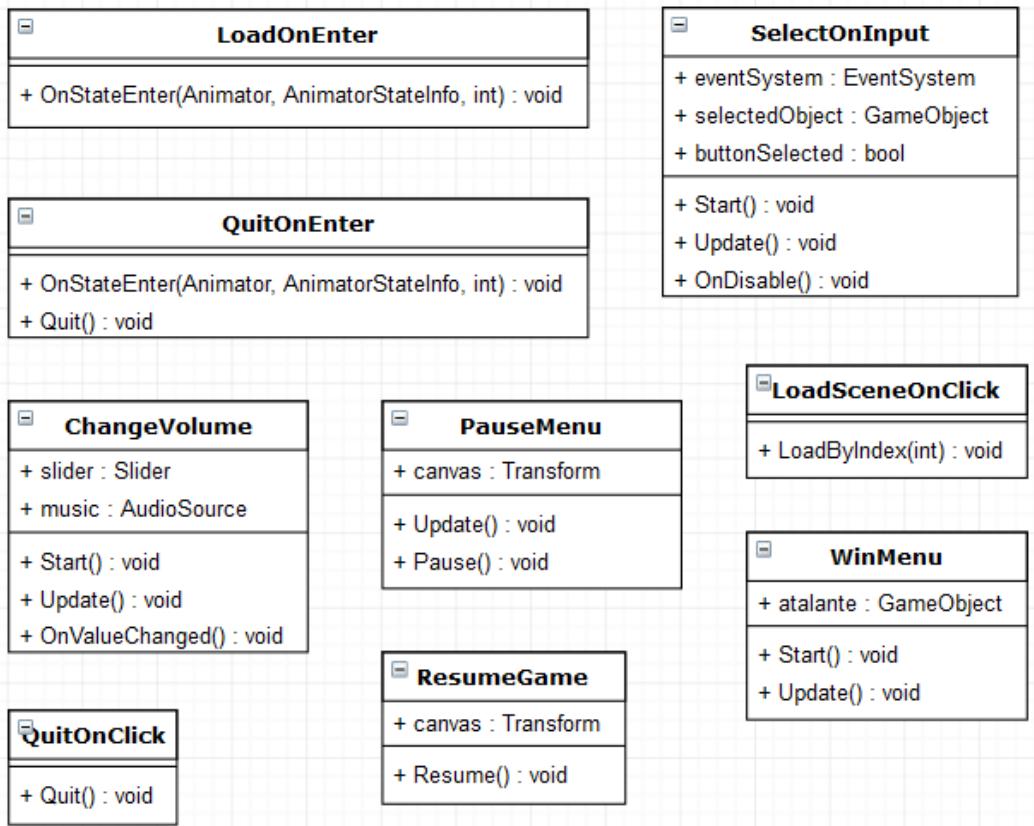
Cette hiérarchie possède une base commune à la grande partie des projets Unity, tels les dossiers Animations, Scripts, Sprites etc. Voici la description de leur contenu respectif :

- Le dossier Animations contient tout ce qui a rapport aux animations (fichiers .controller et .anim), le tout rangé en fonction des éléments animés.
- Le dossier Audio renferme l'ensemble des fichiers audios (sons et musiques) rangés en fonction de leur utilisation (musiques du menu, du jeu, sons d'ambiance etc.).
- Le dossier Fonts renferme la police utilisée pour les textes des menus.
- Le dossier Materials contient les matériaux utilisés par les menus (sliders, textures...).
- Le dossier Prefabs regroupe les éléments préfabriqués, qui n'ont ensuite besoin que d'être intégrés à la scène pour être utilisés. Parmi ces éléments, on retrouve les boutons du menu, les sliders, les pouvoirs etc.
- Scenes contient les deux scènes de notre prototype : le menu principal et le premier niveau du jeu.
- Dans le dossier Scripts nous retrouvons tout le code écrit pour ce projet. Les éléments principaux sont à la racine de ce dossier (le GameManager par exemple, ainsi que le contrôleur de la caméra etc.). Quant au reste des scripts, ils sont répartis en dossiers selon qu'ils concernent la mécanique de Drag&Drop, le menu, les pouvoirs ou encore les pièges.
- Le dossier Sprites contient, comme son nom l'indique, l'ensemble des sprites utilisés en jeu : le fond du jeu, les sprites d'Atalante nécessaires à ses animations, les éléments du décor et enfin les sprites des pouvoirs et pièges.

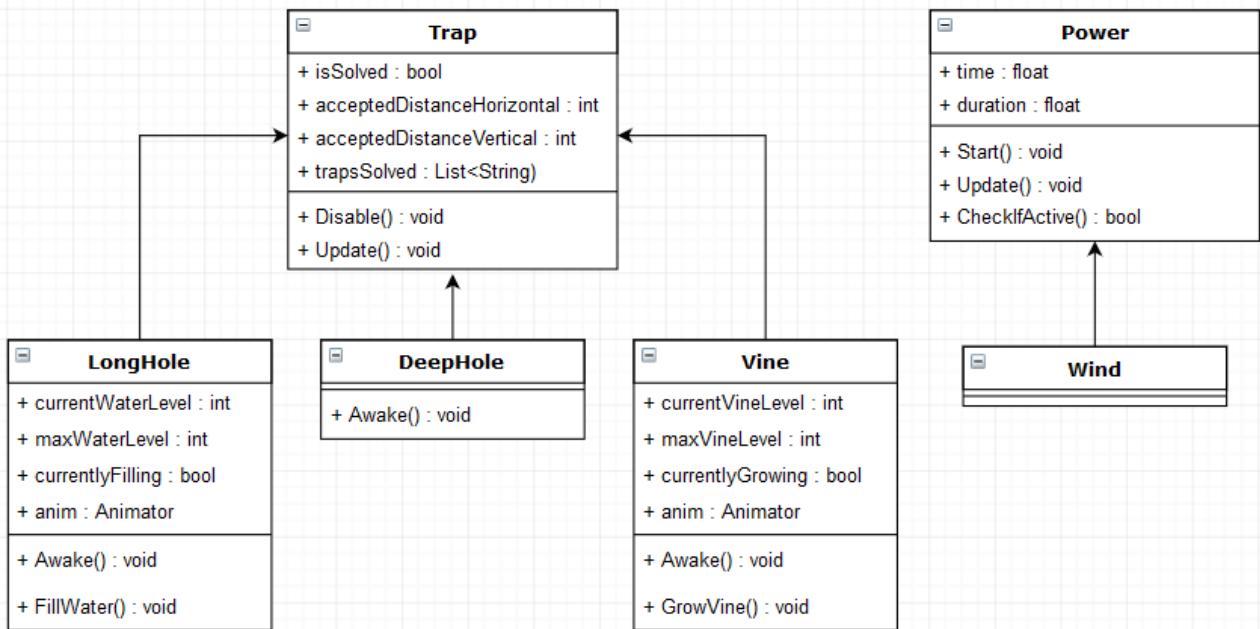
Cette hiérarchie est basique, mais familière et pratique.

2.2. Diagramme de classes

2.2.1. Menus



2.2.2. Pièges et pouvoirs



2.2.3. Général

ChangeMasterVolume	DragAndDropItem
+ slider : Slider	+ draggedItem : DragAndDropItem
+ Start() : void	+ icon : GameObject
+ Update() : void	+ empty : Color
+ OnValueChanged() : void	+ full : Color
	+ cell : GameObject
	+ power : String
	+ source : AudioSource
	+ OnItemDragStartEvent : event DragEvent
	+ OnItemDragEndEvent : event DragEvent
	+ DragEvent(DragAndDropItem) : delegate void
	+ OnBeginDrag(PointerEventData) : void
	+ OnDrag(PointerEventData) : void
	+ OnEndDrag(PointerEventData) : void
	+ MakeRaycart(bool) : void
	+ MakeVisible(bool) : void
	+ SetBackgroundState(bool) : void
Loader	
+ gameManager : GameObject	
+ Awake() : void	
BackgroundMove	
+ player : Transform	
+ initPlayerPos : Vector3	
+ diffPos : Vector3	
+ coefX : float	
+ coefY : float	
+ Awake() : void	
+ FixedUpdate() : void	
+ TrackPlayer() : void	
GameManager	AtalanteBehavior
+ instance : GameManager	+ maxSpeed : float
+ RainTile : GameObject	+ acceleration : float
+ WindTile : GameObject	+ deceleration : float
+ GrowthTile : GameObject	+ jumpForce : float
+ Awake() : void	+ winCanvas : Canvas
+ CreatePower(String, Vector3) : void	+ loseCanvas : Canvas
+ ReactiveTrap(Vector3) : void	+ endLongHole : bool
	+ rb : RigidBody2D
	+ anim : Animator
	+ state : String
	+ audio : Audio
	+ trap : Trap
	+ audio : AudioSource
	+ cpt : int
CameraFollow	
+ xMargin : float	+ Awake() : void
+ yMargin : float	+ OnTriggerEnter2D(Collider2D) : void
+ xSmooth : float	+ Update() : void
+ ySmooth : float	+ Run() : void
+ maxXandY : Vector2	+ Climb() : void
+ minXandY : Vector2	+ CheckLongHole(Collider2D) : void
+ offset : Vector3	+ CheckDeepHole(Collider2D) : void
+ player : Transform	+ CheckVine(Collider2D) : void
+ Awake() : void	+ CheckTrap(GameObject) : bool
+ CheckXMargin() : bool	+ LoadLoseCanvas() : IEnumerator
+ CheckYMargin() : bool	+ LoadWinCanvas() : IEnumerator
+ FixedUpdate() : void	
+ TrackPlayer() : void	

3. Répartition des tâches dans l'équipe

Tâche	Personne(s) y ayant participé
Game Manager	Lorane
Comportement d'Atalante	Martin
Menu principal et animations du menu	Antonin
Mécanique de Drag&Drop	Antonin
Menus de jeu, gestion sonore	Antonin
Pouvoirs et pièges	Pol et Lorane
Animations	Martin, Pol et Lorane
Recherche d'assets	Travail collectif
Design du niveau	Lorane et Martin
Structure globale du projet, lien entre les parties	Lorane

4. Justification des choix de conception et des technologies utilisées

4.1. Choix des technologies

Nous avons choisi de développer le jeu sur Unity car de nombreux tutoriels étaient disponibles, la portabilité sur téléphone est facile et l'outil est utilisable gratuitement : c'est donc parfait pour le développement d'un tel projet, avec pareilles mécaniques.

L'outil Unity Collab nous a permis de travailler chacun sur nos parties sans avoir de gros conflits. C'est un outil pratique puisque utilisable gratuitement (pendant le développement de notre jeu tout du moins). Son principal défaut serait sa gestion des conflits, mais une bonne répartition du travail et de la communication dans l'équipe nous ont permis d'éviter ce genre de situation.

4.2. Choix de conception

Nous nous sommes orientés vers la 2D car nous pensions que l'accès à des assets serait plus simple, et que la 3D n'apporterait que très peu d'intérêt dans notre jeu (dont la vue est exclusivement de côté). Cela rend également le jeu plus léger, ce qui est un bon point pour le portage du jeu sur smartphones ou tablettes.

Le prototype a été développé pour PC et se joue à l'aide de la souris. Néanmoins il a été pensé pour être joué sur téléphone : le *side-scroller* étant un genre particulièrement adapté à cette plate-forme, tout comme la mécanique de *drag&drop* sur un écran tactile. Il était tout simplement plus pratique (car plus simple et rapide) de commencer le développement sur PC, notamment pour réaliser nos tests.

Nous avons essayé de fournir un maximum de feedbacks au joueur : des retours lui indiquant l'état du jeu et les conséquences de ses actions, même les plus minimes. Cela nous semblait très important pour lui permettre d'avoir une compréhension du jeu et de ses mécaniques plus rapides et intuitives. Certains de ces feedbacks ont déjà été expliqués pour les menus (changement de couleur des boutons, sons particuliers lors d'un clic sur eux etc.). Les feedbacks apparaissant *in game* sont expliqués plus en détails dans la partie suivante du rapport.

Les codes relatifs au pouvoirs et aux pièges a été rendu le plus générique possible, afin qu'il soit facile d'en créer de nouveaux.

5. Explications de nos mécaniques de jeu

5.1. Le Drag&Drop des pouvoirs

Lorsque le joueur clique-gauche sur l'image d'un des pouvoirs, le jeu crée une nouvelle entité *icon* que le joueur peut diriger en fonction du pointeur (qu'il s'agisse de la souris ou du doigt sur un écran tactile). Une fois que le clic-gauche est relâché, le jeu crée une instance du pouvoir qui va ensuite vérifier s'il y a un piège en dessous. Si c'est le cas, le piège est désamorcé et Atalante utilisera donc l'animation correspondante.

Au niveau du feedback, lorsque le joueur clique sur l'image correspondant à un pouvoir, il voit apparaître un doublon de l'icône de pouvoir au niveau de son curseur. Ce doublon se déplace avec le pointeur de l'utilisateur. Tant que le clic-gauche n'est pas relâché, la case correspondant au pouvoir change de couleur pour signifier au joueur qu'il est actuellement en train de se servir dudit pouvoir. Au moment où l'utilisateur *drop* le pouvoir, c'est-à-dire qu'il relâche le clic-gauche ou enlève son doigt, la case redevient transparente et reprend sa couleur initiale. Simultanément, le doublon de l'icône de pouvoir reste en suspension à l'endroit où il a été relâché et l'effet sonore correspondant se déclenche.

5.2. Réactions des pièges aux actions du joueur, réactions d'Atalante en fonction de l'état des pièges

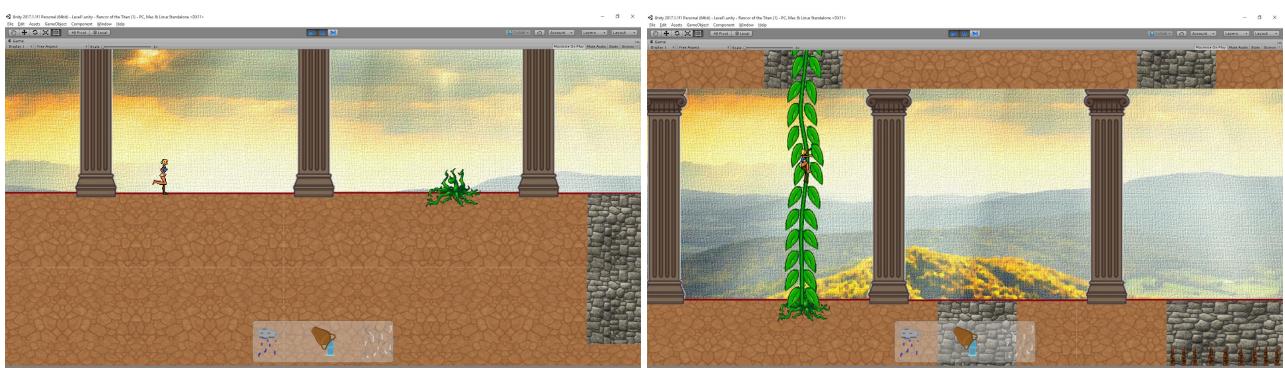
Le personnage Atalante peut effectuer plusieurs actions comme courir, sauter, nager, grimper et mourir. Ces actions sont effectuées en fonction des différents pièges qu'elle doit traverser. Un piège peut être dans deux états : franchissable ou non. S'ils ne sont pas franchissables alors Atalante ne les traverse pas, ce qui entraîne le plus souvent sa mort. La mort d'Atalante se traduit par une chute au sol appuyée par un bruit de cri pour signifier au joueur la mort de l'héroïne. Trois secondes plus tard, afin de laisser le temps au joueur de comprendre l'action, un écran de défaite s'affiche proposant au joueur de revenir au menu principal ou de recommencer le niveau. Un menu de victoire similaire s'affiche si Atalante atteint la fin du niveau. Il peut parfois être intéressant de tomber délibérément dans un piège, qui nous ouvrirait l'accès à un autre chemin possible vers la victoire.

Afin de désamorcer les pièges, il faut qu'un pouvoir adapté soit utilisé dans un périmètre précis. Ainsi par exemple, pour franchir la fosse dont le fond est tapissé de piques, le pouvoir de pluie doit être utilisé au dessus de la fosse. Si le pouvoir utilisé n'est pas le bon ou mal placé, le piège restera amorcé. Les pouvoirs ont chacun un temps d'activité fixe.

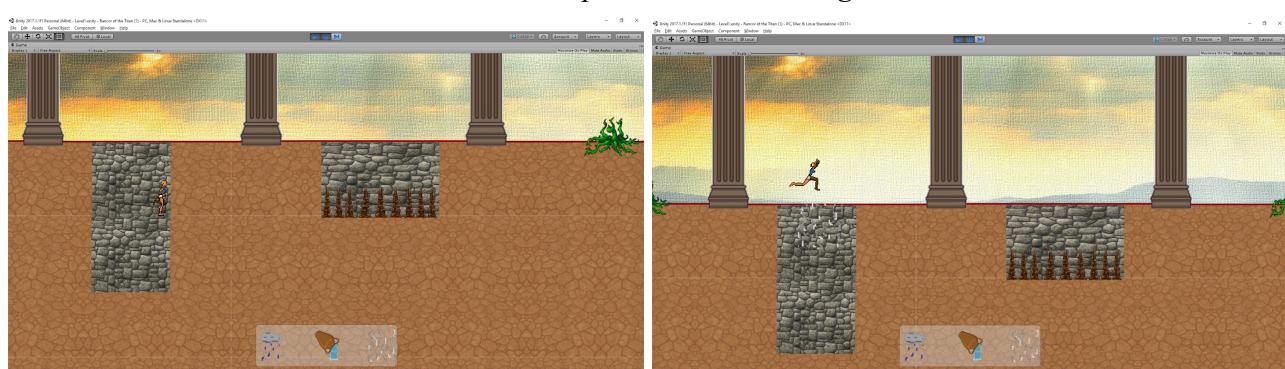
Dans le cas où le piège a été désamorcé, une animation correspondant au pouvoir traduit son désamorçage. L'animation d'Atalante s'adaptera à la traversée du piège désamorcé. Dans le cas d'une vigne, la poussée de cette dernière indique la résolution du piège et Atalante y grimpera alors. Dans le cas d'une fosse profonde, Atalante sautera au dessus d'elle. L'image de vent indiquera la possibilité de sauter plus loin. Pour la fosse à piques, le remplissage d'eau de cette dernière nous montre sa résolution tandis que sa traversée est indiquée par l'animation de nage d'Atalante.

Les pièges possèdent des caractéristiques différentes : certains sont désactivables immédiatement après l'utilisation du pouvoir adéquat, d'autres nécessitent un certain temps, d'autres encore ne sont franchissables que durant la durée du pouvoir. Ces caractéristiques permettent de faire varier le rythme du jeu, et obligent constamment le joueur à devoir tenter de prévoir les actions nécessaires. Il existe pour l'instant, dans notre prototype, trois pièges et trois pouvoirs différents. Chaque pouvoir n'exerce son effet que sur un piège en particulier : le pouvoir Amphore qui permet au piège Vigne de pousser rapidement, offrant à Atalante d'y grimper. Le pouvoir Pluie qui permet à la fosse à piques de se remplir, permettant à Atalante de la traverser à la nage (s'il n'y a pas d'eau, Atalante meurt). Le dernier pouvoir Vent permet à Atalante de sauter au dessus de fosses plus profondes, sans piques. Le Vent ne dure qu'un temps, et il faut activer le pouvoir au bon moment.

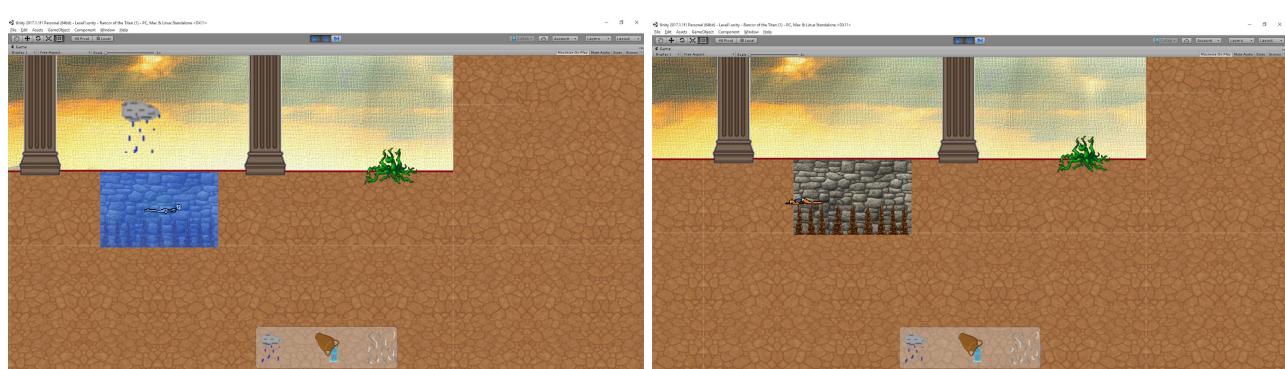
Les captures d'écran qui suivent sont issues du jeu et illustrent les états et l'apparence des différents pièges :



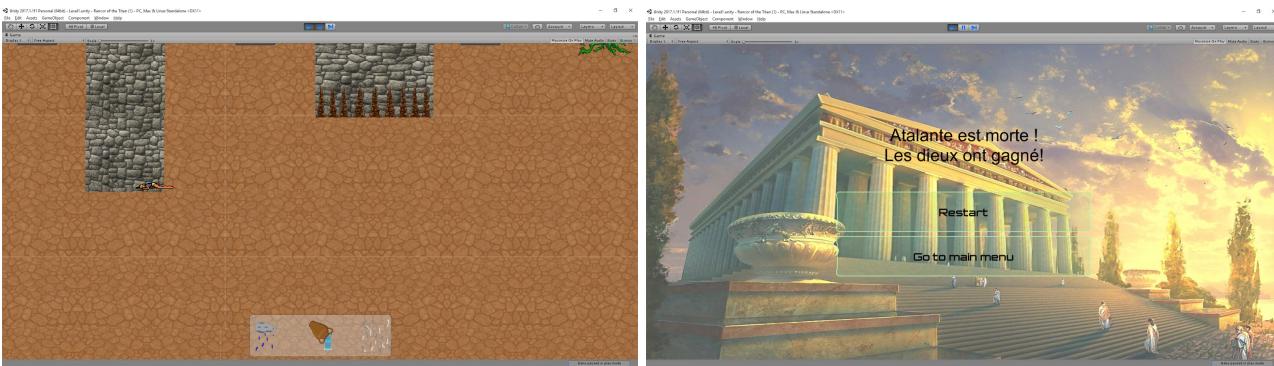
Figures 4 et 5 : Si le piège vigne a été résolu, alors la vigne a grandi et Atalante peut y grimper. Sinon, elle reste petite et Atalante l'ignore.



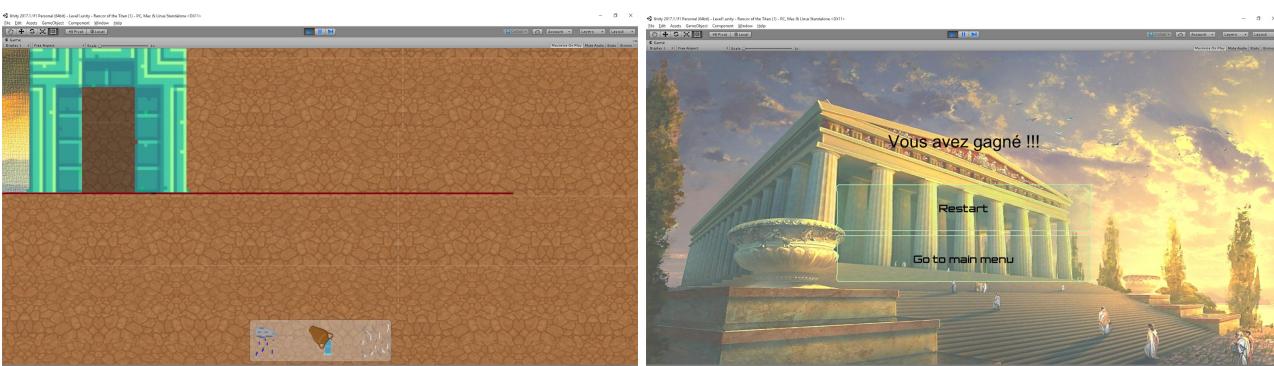
Figures 6 et 7 : La fosse profonde (celle de gauche sur les captures) est considérée comme résolue si le pouvoir du vent a été utilisé. Atalante sautera alors par dessus. Sinon, elle tombe et meurt.



Figures 8 et 9 : La fosse à piques se remplit d'eau suite à l'utilisation du pouvoir Pluie. Atalante peut alors nager pour la traverser. Sinon, elle meurt.



Figures 10 et 11 : On retrouve à gauche l'animation de la mort d'Atalante. À droite, le menu accompagnant la mort de l'héroïne.



Figures 12 et 13 : Il s'agit de la porte signifiant qu'Atalante a atteint son but, ainsi que l'écran de victoire.

6. Description des tests effectués

Le maximum de paramètres ont été variabilisés pour nous permettre de modifier facilement leur valeur (vitesses, durées, longueurs etc.). Dès le début du projet, nous avons créé une scène ne servant qu'aux tests. Dans cette scène, nous lancions les tests puis corrigeions les différents valeurs, les emplacements des colliders etc. Nous recommandons jusqu'à obtenir des résultats satisfaisant, et qui permettent une expérience de jeu la plus agréable possible.

Des tests finaux ont ensuite été effectués sur le *vrai* niveau de jeu. Ces tests étaient nécessaires afin de vérifier que les réactions finales des pièges et d'Atalante soient toujours conformes à nos attentes. En effet, en fonction du design du niveau Atalante a pu prendre des vitesses impromptues ou tomber dans les pièges sous des angles imprévus.

Conclusion – retours post-mortem

Ce projet était pour beaucoup d'entre nous une première expérience dans le domaine de la conception d'un jeu vidéo, et il nous a beaucoup appris. De l'idée ou du concept à l'origine du jeu jusqu'à son développement, nous sommes maintenant beaucoup plus au fait du déroulement des différentes étapes. De ce point de vue nous sommes donc très satisfaits : ce projet nous a permis d'avoir une vision globale ainsi qu'une première approche de la création d'un jeu vidéo. Cela nous sera très profitable à l'avenir.

Il y a eu cependant, lors de ce projet, quelques faits imprévus qui nous ont ralenti plus qu'on ne l'aurait pensé. Premièrement, l'implémentation des *core mechanics* qui ont été plus longues que prévues, même si elles étaient en soi assez simples à réaliser.

Vient ensuite l'aspect graphique de notre jeu. Le choix de la 2D n'a pas été aussi pratique qu'on le pensait pour la recherche d'assets, et nous avons perdu beaucoup de temps à tenter de trouver des sprites nous satisfaisant puis à les modifier pour qu'ils s'intègrent bien au jeu. Nous nous sommes finalement rabattus sur des sprites qui sont corrects mais sans nous convenir pleinement. Une personne ayant des compétences dans le domaine graphique aurait été un grand plus.

La conception du niveau du prototype – son level-design – a également été une tâche que nous avions sous-estimé. Un membre de l'équipe spécialement dédié à cette tâche (peut-être pour les deux derniers sprints par exemple) nous aurait permis de réaliser plusieurs niveaux. Cela nous aurait ainsi rapproché de l'objectif final, à savoir articuler notre jeu autour d'une histoire composée de niveaux successifs et proposant des pouvoirs déblocables au fil de l'aventure.

Ces éléments qui ont pu freiner le développement de notre projet sont bénéfiques en soi : nous savons maintenant à quel point ils peuvent être importants et nous pourrons mieux les prendre en compte pour les projets futurs... projets qui seront peut-être alimentés par la multitude d'idées nous étant venues en tête lors du développement du prototype de **Rancor of the Titan**.