Projet platformer : notice d'utilisation

1. Contrôles

Figure 1 : Les contrôles du jeu (guide accessible en sélectionnant l'option « Guide » depuis le menu principal/menu de démarrage)

But du jeu : Le joueur contrôle un cube bleu. L'objectif est de compléter les différents niveaux en atteignant un bloc vert. Si le cube bleu contrôlé par le joueur entre en contact avec un cube rouge, le niveau est échoué. Le joueur a la possibilité de retenter sa chance sur le niveau échoué autant de fois que voulu.

Le jeu est contrôlable à la manette (Xbox One) et au clavier. Les contrôles sont présentés dans la figure ci-dessus.

La navigation dans les menus du jeu se fait en utilisant la souris, le stick directionnel (stick gauche) ou le d-pad de la manette, ou les touches directionnelles du clavier (haut et bas, ou Z et S).

La sélection dans les menus s'effectue avec un clic souris, une pression sur le bouton A de la manette, ou une pression sur la touche « Entrer » au clavier.

Parmi les contrôles mentionnés dans la figure ci-dessus, le *crouching*, *fast-fall*, ainsi que la possibilité de mettre le jeu en pause n'ont pas été implémentés.

Quitter le jeu s'effectue depuis le menu principal (option « Quit »). En jeu (dans un niveau), il est possible de retourner au menu principal à tout moment en complétant le niveau ou en échouant à le terminer (le cube bleu touche un bloc rouge).

Note : Retourner au menu principal à la fin d'un niveau conserve la progression.

2. Fonctionnalités, réglages, ajustements et personnalisation

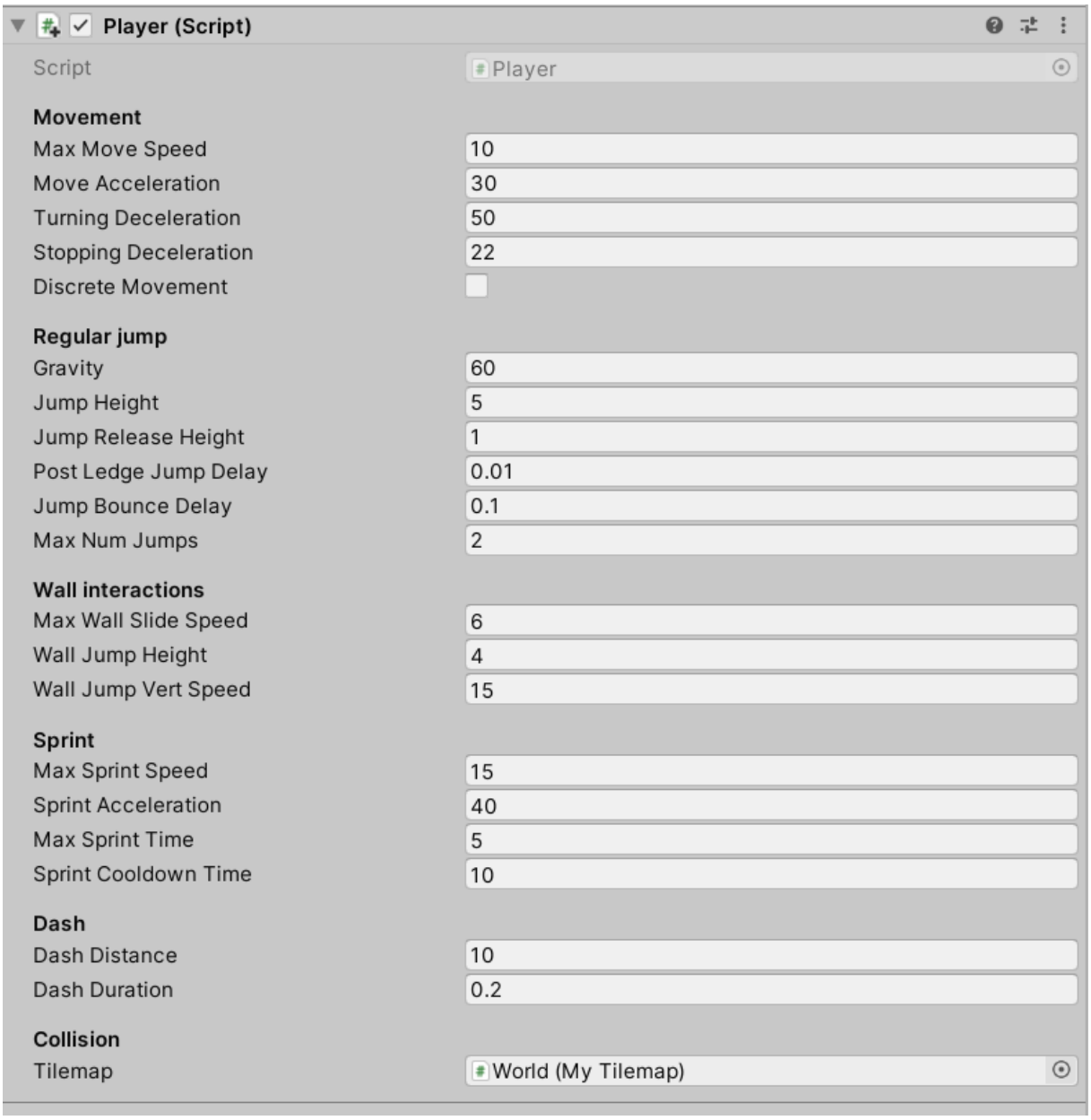

Figure 2 : Le principaux réglages concernant la mobilité du joueur

La figure ci-dessus explicite les différents éléments permettant de paramétrer la mobilité du joueur, ainsi que les valeurs retenues pour chacun de ces éléments. Nous en donnons une description succincte ci-dessous :

Remarque : Toutes les vitesses horizontales configurables dans le panneau de la figure 2 agissent comme des vitesses cibles que le personnage doit atteindre selon l'état dans lequel il se trouve (marche, course, etc…). Tant que le personnage n'a pas atteint sa vitesse cible, sa vitesse actuelle est modifiée par l'accélération correspondant à son état (« Move

Acceleration » pour la marche, « Sprint Acceleration » pour la course, par exemple). Toutes les vitesses horizontales configurables dans le panneau de la figure 2 sont modulées par l'intensité (valeur réelle comprise entre -1 et 1 inclus) avec laquelle la commande directionnelle horizontale est activée : par exemple, si le stick directionnel est incliné vers la droite de manière à avoir parcouru 70% de sa course, la vitesse à atteindre par le personnage sera multipliée par 0.7 ; si le stick directionnel est incliné vers la gauche de manière à avoir parcouru 60% de sa course, la vitesse à atteindre par le personnage sera multipliée par - 0.6.

Dans le cas où « Discrete Movement » est activé, l'intensité avec laquelle la commande horizontale est activée ne peut prendre que les valeurs -1, 0, et 1.

- « Max Move Speed » : La Vitesse horizontale maximale du personnage en l'air ou au sol (sans courir ; vitesse de marche).
- « Move Acceleration » : L'accélération horizontale de base appliquée au personnage lorsque le joueur essaie de le diriger au sol (sans courir) ou en l'air.
- « Turning Deceleration » : Accélération horizontale de sens opposé à la vitesse du personnage appliquée à ce dernier lorsque sa direction de progression change.
- « Stopping Deceleration » : Accélération horizontale de sens opposé à la vitesse du personnage appliquée à ce dernier lorsqu'il doit s'arrêter (lorsque le joueur relâche toute commande directionnelle horizontale par exemple).
- « Discrete Movement » : cf. remarque ci-dessus.
- « Jump Height » : Hauteur de saut du personnage lorsque le joueur maintient le bouton de saut enfoncé.
- « Jump Release Height » : Distance verticale positive (vers le haut) parcourue par le personnage en l'air, après un saut (une fois que le joueur relâche le bouton de saut). Utilisée conjointement avec « Jump Height », cette propriété permet au joueur de réaliser des sauts de hauteur variable.
- « Post Ledge Jump Delay » : propriété inutile et retirée dans la version finale.
- « Jump Bounce Delay » : Utile lorsque le personnage est en l'air. Durée maximale avant que le joueur ne touche le sol pendant laquelle le joueur peut toujours sauter.

Les autres propriétés sont relativement faciles à comprendre.

Les niveaux sont construits avec un éditeur de *tilemaps* réalisé par Joao. Cet éditeur s'utilise depuis le mode édition d'Unity. Il permet de positionner des *tiles* ou blocs de nature différente (*prefabs* différentes avec des propriétés propres). Trois types de blocs sont disponibles dans l'état actuel du jeu :

- Des blocs solides blancs (« *Solid Tiles* ») matérialisant des murs, des obstacles infranchissables par le personnage.
- Des blocs verts (« *Level Goals* ») : le niveau est réussi lorsque le personnage touche un de ces blocs.

- Des blocs rouges (« *Death Blocks* ») : le joueur doit recommencer le niveau si son personnage entre en contact avec l'un de ces blocs.

Pour créer une *tilemap* dans une scène, il suffit de placer un GameObject vide dans cette scène, et d'y adjoindre le script Assets/Scripts/MyTilemap.cs. Tous les blocs créés dans la scène avec l'éditeur de *tilemaps* (voir Assets/Editor/TilemapEditor.cs) sont enfants du GameObject vide portant le script MyTilemap.cs.

Voilà un aperçu de l'éditeur de *tilemaps* :

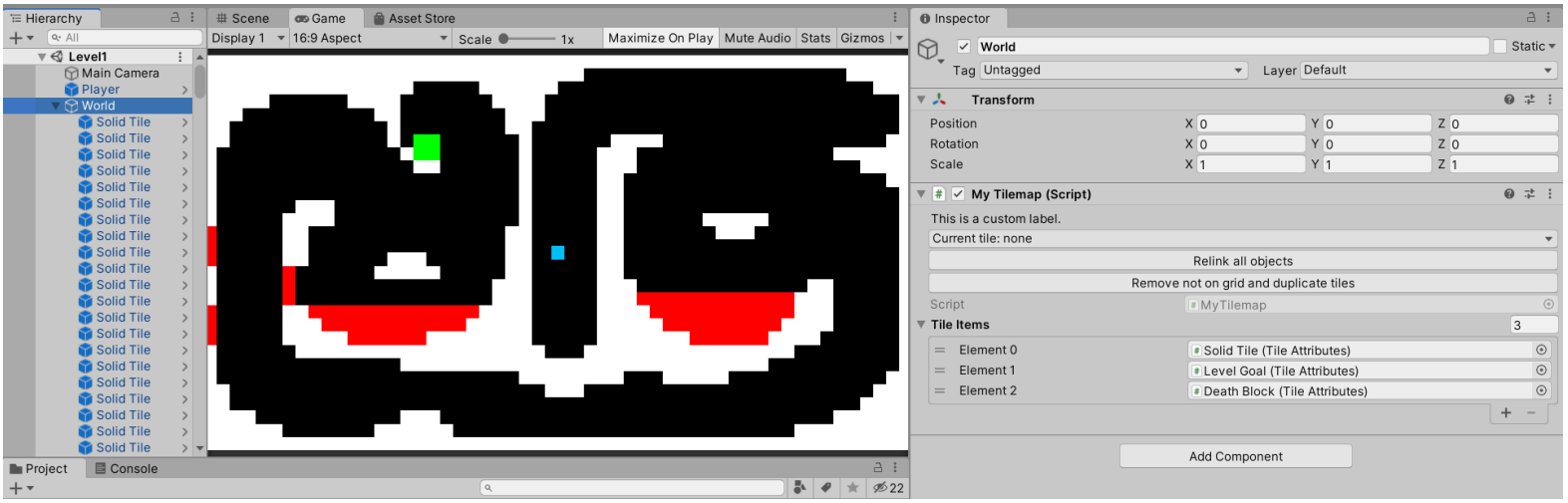

Figure 3 : Vue de l'éditeur de tilemaps

3. Feedbacks

Il est possible de désactiver les feedbacks implémentés en sélectionnant l'option « Settings » depuis le menu principal.

Les feedbacks retenus sont les suivants :

1. Animation « squishy » lorsque le personnage touche le sol après un saut : Cette animation est réalisée en modifiant la hauteur et la largeur du personnage selon un profil sinusoïdal amorti.
2. Étirement horizontal du personnage en début de *dash*.
3. *Screen shake* lorsque le personnage est en *dash*.
4. Traînée de fumée lorsque le personnage glisse sur un mur (réalisée à l'aide d'un système de particules).

Les feedbacks 1, 2 et 3 sont réalisés avec des transitions personnalisées appliquées à des propriétés spécifiques des GameObject concernés (position, hauteur, largeur, etc…).

Un dérivé du patron de conception « stratégie » a été utilisé pour produire des transitions caractérisées par des *easing functions* différentes. La classe de base exposant l'interface (il s'agit d'une classe abstraite en réalité) des transitions est Assets/Scripts/Transition.cs.

Pour implémenter les feedbacks 1 et 2, nous avons introduit la classe DampedWaveTransition, fille de Transition. Elle permet de réaliser des transitions d'une valeur A à une autre B selon un profil sinusoïdal de fréquence ajustable, et amorti exponentiellement selon un facteur d'amortissement configurable également. Dans le cas du feedback 1, la fréquence des oscillations est relativement élevée, contrairement au cas du feedback 2 (pour lequel le facteur d'amortissement est en revanche beaucoup plus élevé pour obtenir une transition rapide).

Le feedback 3 est obtenu en appliquant une transition aléatoire à la position de la caméra grâce à la classe NoisyLinearTransition.

4. Play tests

Play test 1 : Joueur régulier

Le premier testeur est un joueur de 16 ans habitué aux jeux-vidéo.

Dès la première prise en mains, le testeur est positivement surpris de la bonne maniabilité du personnage. Selon lui, il est très agréable de déplacer le personnage, surtout dans les airs (mécanique de *dash* très appréciée).

Le joueur éprouve toujours un réel plaisir à déplacer le personnage, même en l'absence de *feedbacks*.

Ce premier testeur arrive à comprendre les enjeux des niveaux (éviter les blocs rouges qu'il assimile à de la lave, et atteindre les blocs verts pour gagner) instantanément, sans avoir consulté le guide d'utilisation au préalable.

Lors d'un premier test, ce joueur exprime comment il aurait préféré pouvoir réaliser un *dash* avec un bouton situé sur le dessus de la manette ou une gâchette, au lieu du bouton X.

Après déplacement du bouton de *dash* de X à RB, le joueur se plaint, dans un deuxième test, de devoir utiliser RB également pour faire courir son personnage. Il aurait souhaité que les boutons permettant de réaliser un *dash*, et de courir soient dissociés. Il suggère de réserver le bouton X pour faire courir le personnage.

Le joueur est surpris de la vitesse verticale parfois atteinte par le personnage lors d'une longue chute. Cela laisse supposer qu'il faudrait limiter la vitesse de chute du personnage.

Le joueur remarque le bug suivant : lors d'une collision entre le personnage et un mur, et après que le joueur relâche le stick directionnel, le personnage est repoussé par le mur. Le

testeur trouve en premier lieu ce comportement amusant, mais change rapidement d'opinion lorsqu'il atteint un niveau plus difficile dans lequel ledit comportement devient déstabilisant. Contrôler le personnage dans des endroits exigus devient difficile pour le joueur.

Le joueur suggère que ce comportement puisse être intégré pleinement dans le jeu à certains endroits pour augmenter la difficulté.

Enfin, le testeur suggère de placer des bonus (pièce géante) dans certains endroits non-utilisés et difficiles d'accès de certains niveaux. Nous n'avons pas implémenté cet élément.