README: Description du livrable 2

NB: *Nous avons mis un .exe si vous êtes sous Windows. Si vous n'êtes pas sous windows, envoyez nous un mail et nous vous enverrons une application adéquate.*

Structure globale :

La structure de nos fichiers est :
Resources: *contient toutes les ressources artistiques du jeu*
Source: *contient tous les éléments source du jeu (codes et scènes) rangés dans des dossiers détaillés ci-dessous en "Description des scripts et fonctions"*
Chaque dossier est ensuite séparé en "Scene" qui contiennent les scènes qui contiennent toutes les informations et paramètres utilisés par Godot et en "Scripts" qui contiennent les codes sources

- **Autoload**
  - Scene
  - Script
- **Actors**
- **Entities**
- **Levels**
- **UserInterface**
- **World**
- **Gravity**
- **Time**

Les fichiers **default_env.tres**, **project.godot** sont des fichiers contenant les différents paramètres globaux du jeu (comme les commandes utilisateur, la résolution de l'écran etc)
Les fichiers **icon.png** et **icon.png.import** représentent l'icône de l'application (pour le moment non touchée)

Conventions :

Variables : minuscules complètes
Classes (scènes) : Majuscule à chaque début de mot
Méthodes : minuscules avec underscore si besoin
Variables indiquant un état (booléen) : minuscule avec underscore et commençant "is_"
Nodes : on garde une trace du type de la node (ex: "ReloadTimer", "JumpTimer" etc)

How to play :

Dans Gravitime, le joueur peut manipuler la gravité et le temps afin de résoudre divers puzzles. Voilà comment contrôler ces aptitudes :

Contrôles du joueurs :

Aller à gauche : Flèche gauche / « Q »
Aller à droite : Flèche droite / « D »
Sauter : Flèche haut / « Z »
Attaquer : « E »

Contrôle du temps:
Lancer un enregistrement temporel (5 sec) : Touche « F » [Clone pas encore fonctionnel]
Au bout de ce temps, joueur et ennemis reviennent à leurs positions initiales

Contrôle de la gravité:
Modifier la gravité d'une zone : Cliquez dans la zone et faites glisser la souris dans la direction souhaitée. Plus l'on glisse loin, plus la gravité est forte. [Pour l'instant il manque l'indicateur visuel indiquant la direction et l'intensité du vecteur champs de gravité de la zone]

Description des scripts et fonctions :

Actors : *Contient tous les acteurs du jeu, soit le Player, les ennemis et les clones temporels*

Actor.gd : *Interface qui détermine des variables et méthodes utilisées par tous les acteurs*
→ **apply_gravity()**: *fonction qui applique la gravité actuelle à l'acteur (joueur, clone, ennemi)*

Enemy.gd : *Ennemi principal du jeu, qui se contente pour le moment de bouger en ligne droite jusqu'à ce qu'il rencontre un mur ou du vide auquel cas il rebrousse chemin*
→ **_ready()**: *appelée à l'instanciation de l'ennemi; ajoute l'instance au groupe "timecontrol"*
→ **_physics_process()**: *appelée à chaque frame du jeu; attribue sa vitesse à l'ennemi*
→ **calculate_new_velocity()**: *fonction appelée à chaque frame, calcule la vitesse à attribuer à l'ennemi et teste notamment si celui-ci rencontre un vide ou un mur*
→ **hit()**: *fonction appelée par la classe MeleeAttack à chaque fois que l'ennemi est touché par l'attaque du joueur (MeleeAttack)*
→ **save()**: *fonction appelée par TimeControl, qui sauvegarde la position de l'ennemi lorsque la trame temporelle de rembobinage débute*
→ **timeReset()**: *fonction temporelle appelée par TimeControl à la fin de la sélection de la trame temporelle de rembobinage, qui change la position de l'ennemi à sa position au début de ladite trame (donnée par fonction save())*
→ **_on_PhysicalHitbox_area_entered()**: *fonction appelée dès que l'ennemi rentre dans une zone. Teste notamment si l'ennemi est rentré un champ de gravité pour activer ses effets.*
→ **_on_PhysicalHitbox_area_exited()**: *fonction appelée dès que l'ennemi sort d'une zone. Teste notamment si l'ennemi est sorti d'un champ de gravité pour désactiver ses effets.*

Player.gd : *Le personnage que le joueur peut contrôler à l'aide des touches du clavier.*
→ **_ready()**: *appelée à l'instanciation du joueur; ajoute l'instance au groupe "timecontrol"*
→ **_physics_process()**: *appelée à chaque frame du jeu; attribue sa vitesse au joueur*
→ **get_direction()**: *traduit les commandes du joueur en direction*
→ **calculate_move_velocity()**: *calcule la vitesse à attribuer au joueur*
→ **_on_PhysicalHitbox_body_entered()**: *appelée lorsque le joueur rentre en contact avec un corps étranger et appelle la fonction hit() si ce corps est un ennemi*

→ **hit()**: *appelée par la fonction _on_PhysicalHitbox_body_entered(), est la fonction qui traduit le fait que le joueur est touché : pour le moment, il perd une vie et revient à sa position initiale ou appelle la fonction die() si il n'a plus de vie*
→ **die()**: *appelée par hit() si le joueur n'a plus de vie, se contente de détruire l'instance du joueur pour le moment*
→ **skills()**: *appelée à chaque frame par _physics_process(), qui permet de gérer les skills du joueur (contrôle temporel et attaque)*
→ **attack()**: *appelée par skills() et permet au joueur d'attaquer en créant une instance de MeleeAttack en face de Player*
→ **save()**: *appelée par TimeControl, sauvegarde la position actuelle du joueur*
→ **timeReset()**: *appelée par TimeControl, remet le joueur à sa position sauvegardée par save()*
→ **_on_PhysicalHitbox_area_entered(Area2D)**: *fonction appelée dès que le joueur rentre dans une zone. Teste notamment si le joueur est rentré un champ de gravité pour activer ses effets.*
→ **_on_PhysicalHitbox_area_exited(Area2D)**: *fonction appelée dès que le joueur sort d'une zone. Teste notamment si le joueur est sorti d'un champ de gravité pour désactiver ses effets.*

TimeClone.gd : *clone temporel du joueur qui sera instancié à la fin du rembobinage temporel effectué par TimeControl, qui répétera les actions du joueur qu'il a effectué durant la trame temporelle de sélection ; pour le moment n'est pas fini et n'est jamais instancié*
→ **_on_PhysicalHitbox_body_entered(Node)**: *appelée lorsque le clone rentre en contact avec un corps étranger et appelle la fonction hit() si ce corps est un ennemi*
→ **hit(dmg)**: *appelée par la fonction _on_PhysicalHitbox_body_entered(), est la fonction qui traduit le fait que le clone temporel est touché : pour le moment, il meurt immédiatement (appelle de die())*
→ **die()**: *appelée par hit(), se contente de détruire l'instance du clone*
→ **attack()**: *jamais appelée pour le moment, sera utilisée pour attaquer selon les actions du joueur durant la trame temporelle de sélection*

Autoload : *Contient tous les éléments du jeu qui sont "autoloadés", c'est-à-dire qu'ils sont toujours actifs quoiqu'il arrive, peu importe si on change de niveau ou non - cela permet d'avoir des variables qui peuvent être actualisées et lues par tous les éléments du jeu à n'importe quel moment*

GeneralData.gd : *contient toutes les informations qui doivent être conservées à travers le jeu (vie du joueur et plus tard un score)*
→ **reset()**: *appelée lors du reset du jeu (pour le moment jamais appelée car sans moyen de reset mis en place), elle remet à l'état initial la vie du joueur et la musique (non mise en place pour le moment)*
→ **set_player_hp()**: *fonction appelée par les autres scripts dans le jeu, permet d'actualiser la vie du joueur (qui est stockée dans GeneralData)*
→ **get_player_hp()**: *fonction appelée par les autres scripts pour que ceux-ci puissent lire et récupérer la vie du joueur contenue dans GeneralData*
→ **music_stop()**: *fonction pour le moment inutile, permettra d'arrêter la musique en cas de game over*

TimeControl.gd : *gère le rembobinage temporel en lancant une trame de sélection temporelle au cours de laquelle tous les éléments du jeu enregistrent leurs positions initiales et le joueur enregistre les différentes inputs de l'utilisateur ; puis à la fin de cette sélection (déterminée par un timer de 5 secondes), le "rembobinage" s'effectue, en remettant les entités à leurs positions initiales*

→ **timereset()**: *appelée par Player lorsque le skill de contrôle temporel est activée par l'utilisateur ; appelle la fonction save() de toutes les entités affectées par le rembobinage temporel (pour le moment seulement le joueur et l'ennemi) et lance le timer de sélection de la trame temporelle*

→ **_on_TimeResetTimer_timeout()**: *appelée à la fin du timer, lance le rembobinage en appelant la fonction timeReset() des entités affectée par le rembobinage temporel*

Entities

Box.gd : Une simple boîte que le joueur peut pousser pour des puzzles

→ **_ready()**: *appelée à l'instanciation de la caisse.*

→ **_on_PhysicalHitbox_area_entered(Area2D)**: *fonction appelée dès que la caisse entre dans une zone. Teste notamment si la caisse est rentrée dans un champ de gravité pour activer les effets de ce dernier.*

→ **_on_PhysicalHitbox_area_exited(Area2D)**: *fonction appelée dès que la caisse sort d'une zone. Teste notamment si la caisse est sortie d'un champ de gravité pour désactiver les effets de ce dernier.*

DoorButton.gd : Un boutton qui peut appeler des fonctions dans d'autres scripts

→ **_ready()**: *appelée à l'instanciation de l'entité ; place le bouton dans la bonne animation*

→ **_on_body_entered(KinematicBody2D)**: *quand un corps entre en collision avec le bouton, une animation joue*

→ **_on_body_exited(KinematicBody2D)**: *quand un corps sors de collision avec le bouton, une animation joue*

Laser.gd : Un piège simple qui capte les signaux envoyés par un bouton

→ **_ready()**: *appelée à l'instanciation; place le laser dans la bonne animation et le bon niveau de détection de collision*

→ **_on_Button_body_entered(Node)**: *désactive la détection des collisions du laser quand le bouton associé est enfoncé*

→ **_on_Button_body_exited(Node)**: *réactive la détection des collisions du laser quand le bouton associé est relâché*

→ **_on_Laser_body_entered(Node)**: *quand un corps entre en collision avec le laser, cette fonction appelle la fonction hit du corps, s'il en possède une*

Level_End.gd : *Un portail qui permet de changer de niveaux (pour le moment non utilisé)*

→ **change_level()**: *appelée par _on_Level_End_body_entered(); permet de changer de niveau en changeant la scène jouée*

→ **_on_Level_End_body_entered()**: *appelée lorsque le joueur rentre en contact avec le portail, appelle change_level()*

MeleeAttack.gd : *L'attaque du joueur qui permet d'attaquer un ennemi à côté de lui ; comme c'est une attaque de mêlée, elle doit s'effacer 0.5 seconde après son initialisation*

→ **_ready()**: *appelée lors de l'initialisation de l'instance de MeleeAttack, permet de lancer le timer DestroyTimer*
→ **_on_body_entered()**: *appelée lorsque qu'un corps étranger rentre en contact avec MeleeAttack et appelle la fonction hit() de ce-dit corps*
→ **_on_DestroyTimer_timeout()**: *appelée lorsque le DestroyTimer atteint sa limite de temps, détruit l'instance de MeleeAttack*

Gravity

GravityField.gd : *Une zone dont on peut modifier la gravité avec la souris. Il suffit de cliquer et de glisser la souris au sein de la zone pour agir sur le champ de gravité.*

→ **_input(InputEvent)**: *fonction appelée dès qu'un événement se produit (mouvement, ou clic de la souris ici). Traduit les commandes à la souris pour modifier le vecteur champ de la zone de gravité uniquement si le clic s'effectue au sein de la zone. Appelle calculate_gravity(Vector2, Vector2).*
→ **get_actual_gravity()**: *fonction qui renvoie le vecteur champ de gravité actuel de la zone.*
→ **calculate_gravity(Vector2, Vector2)**: *Cette fonction permet de calculer le nouveau vecteur champs de gravité à partir des inputs de la souris*
→ **_on_GravityField_mouse_entered()**: *Cette fonction est appelée quand la souris rentre dans la zone de gravité. Modifie le booléen indiquant si la souris est dans la zone sur true.*
→ **_on_GravityField_mouse_exited()**: *Cette fonction est appelée quand la souris sort de la zone de gravité. Modifie le booléen indiquant si la souris est dans la zone sur false.*

Levels : contiendra les futurs niveaux du jeu.

UserInterface : *Contient tous les éléments relatifs à l'interface utilisateur, comme les écrans de début/fin/pause et le HUD*

HUD.gd : *Element qui s'affiche perpétuellement sur l'écran et qui affiche les éléments importants tel la vie du joueur*
→ **_process(delta)**: *appelée à chaque frame du jeu, affiche la vie du joueur actualisée*

World - Contient la tilemap permettant de construire les niveaux