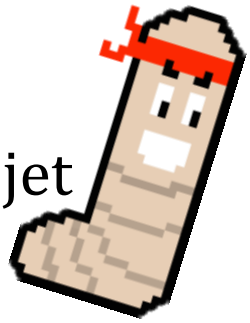
Worms

Rendu Final du Mini Projet



Un projet par : CHARNAY Jacques, BERTRAND Paul, BERNARD Maxime, BOURSAUD Thomas.

Table des matières

I. Cahier des Charges

Notre projet consiste à recréer le jeu « Worms ». Ce jeu est un jeu de type tour par tour en 2D ou des vers de terre, les « Worms », s'affrontent en équipe sur un terrain entouré d'eau. Dans ce jeu, plusieurs Worms s'affrontent sur une carte grâce à des armes variées (lance-roquettes, grenade, fusils, etc ...). Le but est de battre tous les joueurs de l'équipe adverse.

Le but de ce projet est donc de fournir à l'utilisateur une expérience de jeu agréable au plus proche du jeu original « Worms ». Ainsi notre projet remplit les différentes fonctionnalités :

- Générer un terrain praticable pour le joueur et destructible par les armes ;
- Permettre au joueur d'incarner un « Worms » pouvant se déplacer sur ce terrain ;
- Proposer deux types d'armes au joueur, le bazooka et la grenade, pouvant exploser ;
- Les explosions générées par ces armes pourront affecter le terrain et endommager les « Worms » en leur enlevant de la vie ;
- Proposer une expérience de jeu tour par tour similaire à celle des jeux vidéo actuels ;
- Avoir un moteur physique s'appliquant aux « Worms » et aux projectibles.

Afin de faciliter la prise en main du jeu, nous mettons à disposition un tutoriel expliquant les commandes de bases. Ce tutoriel se trouve dans le dossier « Rendu ».

II. Principe de l'algorithme

Classe	Classe(s) héritière(s)	Description de la classe
Weapons	- Bazooka - Grenade - ↳HolyGrenade (1) - Teleporteur *Rmq : HolyGrenade hérite de Grenade*	Classe codant les « lanceurs » de projectiles. Permet de dessiner un indicateur renseignant le joueur sur l'angle du lancer et sur la puissance de celui-ci.
Projectile	- Rocket - GrenadeProjectile - ↳HolyGrenadeProjectile (1) - Balle (inutilisée) *Rmq : HolyGrenadeProjectile hérite de GrenadeProjectile*	Classe permettant l'apparition d'objet mouvant, les projectiles. Cette classe permet de leur appliquer la physique choisie (i.e. la gravité, la puissance du tir). Elle permet également l'explosion de ces projectiles.
Menu	⌀	Ce menu est une IHM permettant de sélectionner les paramètres préliminaires nécessaires au lancement d'une partie. En effet, ce menu permet de choisir la couleur de son équipe et donc de la couleur de ses Worms ainsi que leurs noms (aléatoirement choisis de base). On peut également choisir le terrain sur lequel on veut s'affronter.
FenetreJeu	⌀	Classe principale du jeu. Elle permet la gestion de l'affichage graphique (affichages des Worms, explosions). Elle permet aussi la gestion des différentes phases de jeu et l'apparition des Worms au début de la partie.
MoteurPhysique	⌀	Le moteur physique de ce projet permet de calculer l'accélération et la vitesse d'un objet en fonction des forces que l'objet subit. Il permet donc l'application d'une ou plusieurs forces à un objet en mouvement (projectiles, Worms). Il permet donc de modéliser notamment la gravité, les trajectoires des projectiles et les rebonds. Cette fonction permet aussi de détecter si un objet est en contact direct avec un bloc (utile pour les rebonds).
Force	⌀	Permet de stocker en mémoire une force ayant une composante selon x et y.
Worms	⌀	Permet de définir complétement un « Worms ». En effet, cette classe gère le nom, la couleur (i.e. l'équipe du Worms), la vie de celui-ci et l'affichage graphique. Les méthodes de cette classe permettent d'afficher à l'écran toutes ces informations. Cette classe permet d'enlever (/rajouter) de la vie au Worms. Cette classe permet également de déplacer le Worms.

GestionTerrain	⌀	Permet de générer le terrain ou vont s'affronter les Worms grâce à une image bitmap. (Anciennement, ce générateur pouvait créer des terrains aléatoirement mais pour des soucis d'esthétisme nous avons opté pour des terrains pré-faits.)
Map	⌀	Permet de stocker plus facilement les informations nécessaires à la génération d'un terrain (adresse du fond d'écran, textures, etc…)
Block	⌀	Permet d'associer des coordonnées x et y à un bloc.
Inventaire	⌀	Permet d'afficher une interface graphique interactive où le joueur peut sélectionner son arme.
GestionTours	⌀	Cette classe permet de gérer le système tour à tour du jeu. Elle choisit quel joueur va jouer en alternant entre les deux équipes. Cette classe permet également d'afficher à l'écran qui doit jouer et à indiquer une mort. La fin du jeu est ainsi réaliser par cette classe qui détecte lorsqu'une équipe n'a plus de joueur. Elle affiche alors l'écran de fin.

(1) *Remarque :* « ↳ » signifie ici que l'élément hérite de l'élément au dessus.

Pour mieux comprendre la physique du jeu :

Afin de comprendre comment le moteur physique fonctionne et mieux appréhender la logique de notre code, nous avons mis en place un mode affichant les données physiques calculées à l'écran. Ce mode est accessible en appuyant sur la touche « c » du clavier.

Figure 1 : Capture d'écran du mode montrant la physique du jeu

Ce mode permet ainsi un affichage des blocs plus clair (en violet). Les Worms sont également entourés d'un rectangle vert symbolisant l'espace qu'occupe le Worms à l'écran. Ce phénomène s'applique également aux projectiles. Ce mode affiche également, pour chaque Worms, son nom, sa position x et y sur l'écran en pixel et sa position convertit dans l'espace en mètres. L'accélération et la vitesse selon x et y sont aussi affichées.

III. Bibliographie

- *Javadoc de la librairie Slick2D, http://slick.ninjacave.com/javadoc/ :* Cette javadoc nous a grandement aidé pour réaliser notre mini-projet. En effet, nous avons choisi cette librairie afin de pouvoir créer un programme type jeu plus facilement.
- *Tutoriels sur Slick2D et en particulier http://www.shionn.org/tutoriels-slick-2d :* Ce blog explique comment créer un jeu de type « RPG » grâce à la librairie Slick. Nous avons donc suivi quelques-uns de ces tutoriels afin de réaliser notre projet. Ces tutos nous ont permis de nous familiariser un peu plus avec cette librairie.
- *Les cours du PC :* Les cours du Premier Cycle nous ont grandement aidé pour l'implémentation de l'IHM (le menu) mais aussi pour comprendre les aspects de l'affichage graphique grâce à la librairie Slick. Les listes ont aussi été également utiles.
- *Notre projet sur Github https://github.com/Reddragio/WormsSlick2 :* Cette plateforme nous a permis de créer une sorte de cloud pour mieux gérer ce projet de groupe.

IV. Suggestion d'améliorations du projet, bugs connus

Quelques améliorations du programme peuvent être envisagées. Tout d'abord l'apparition des Worms en début de partie qui pourrait être aléatoire. Evidemment, les textures sont perfectibles. On pourrait, en effet, imaginer coder une animation du Worms lors de son déplacement (avec des sprites) ou bien lors d'action spécifiques. On pourrait également imaginer l'ajout d'armes variées comme des armes au corps à corps. Aussi, l'ajout d'items interactifs apparaissant sur la carte pourrait être implémenté (comme des caisses redonnant de la vie). Enfin, avec plus de temps, l'ajout du vent intervenant dans la direction des projectiles aurait pu être implémenté. L'ajout de blocs avec différentes résistances et d'un didacticiel pour familiariser le joueur avec les commandes aurait pu être ajouté au projet.

Les bugs connus sont :

- Pas de dégâts de chute.
- Les projectiles passent à travers 1 ou 2 blocks
- Mauvais affichage de l'animation de l'explosion
- Bug de collision impliquant des dégâts de chute injustifiés

Néanmoins, ces bugs restent assez rares.

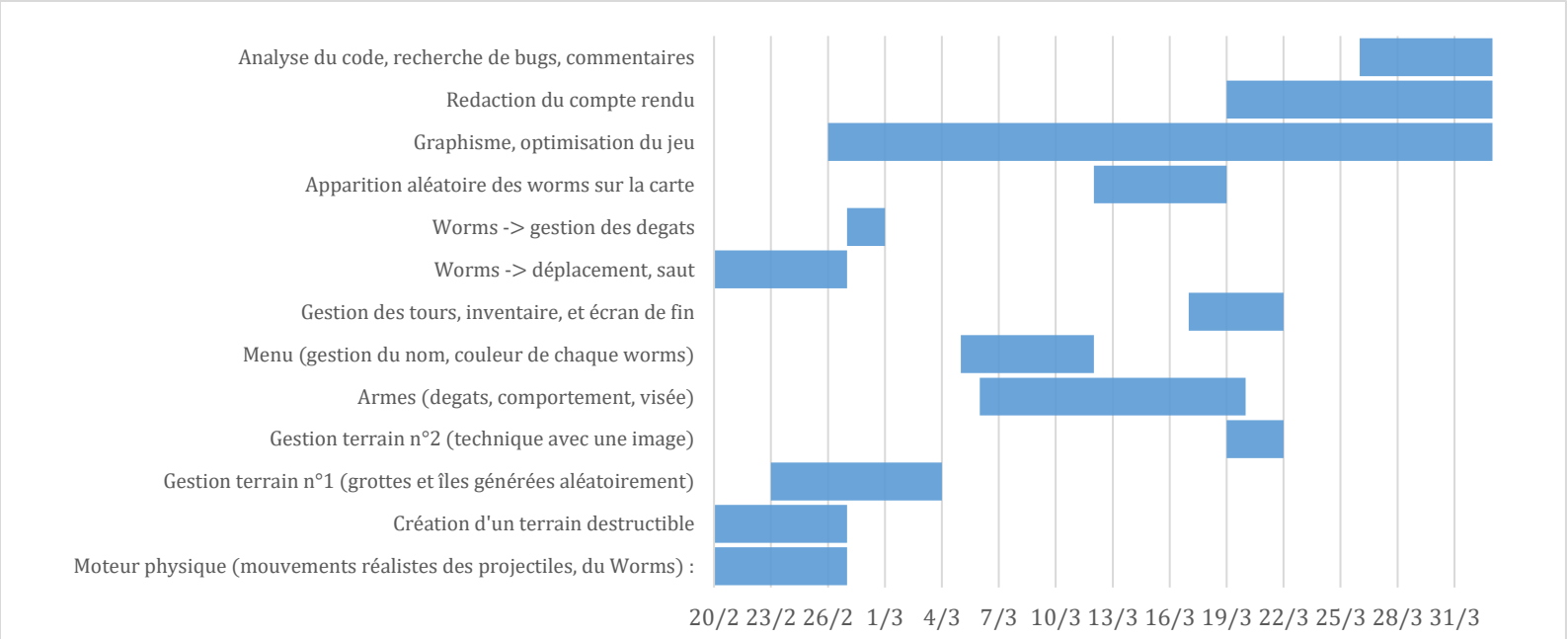
V. Problèmes rencontrés et solutions apportées

Problèmes rencontrés	Solutions apportées
La génération aléatoire donnait un terrain peu esthétique et agréable à jouer.	Suppression de la génération aléatoire, ajouts de terrains pré-générés beaucoup plus agréables visuellement.
Impossibilité de passer certains petits obstacles.	Ajout du double saut comme dans la version originale de Worms.
Images pas secondes trop faibles → mauvaise gestion de la physique du jeu.	Nous avons optimisé l'affichage du décor. Ainsi, au lieu d'afficher les blocks de 5*5 pixels un à un, le programme cherche à afficher à chaque fois les plus grandes bandes horizontales possibles.
Trajectoire roquette aberrante	Enfaite, c'était la limitation de vitesse à 5pixels/frame qui déviait la trajectoire de la rocket. Nous avons simplement désactivé cette sécurité pour obtenir des trajectoires parfaites. Le seul risque avec cette solution est qu'en cas de lag, il est possible que les rockets sortent complètement du terrain et provoquent un crash.
Lags (ralentissements) ponctuelles lors du tir des grenades et des rockets	Au sein de chaque classe de projectile, nous avons déclaré les images et les sont comme « final static » afin que, quel que soit le nombre d'instance des projectiles, Java ne charge qu'une et une seule fois les images et sons dans la RAM.

VI. Carnet de route

Nous avions prévu initialement le déroulement des différentes étapes de codage de notre projet. Nous avons ainsi tenté de respecter ces étapes.

Le déroulement de l'implémentation de notre projet peut être résumé dans ce diagramme de Gantt :



Résumé jour par jour du projet :

23 Février 2018 :
- Création du Github et envoi du travail déjà effectué :
- Création des classes (Projectiles, Balles, FenêtreJeu, Worms, Grenade, Rocket, Weapon, Worms) et import des images pour chaque Worms + textures terrain + fond de carte.
  A ce stade, le jeu affiche uniquement la carte et le Worms se déplace et sa gravitée est gérée.

24 février :
- Ajout d'une musique de fond
- Ajout de l'explosion expérimentale du terrain
- Correction de nombreux bugs physique
- Ajout de fonctionnalités de test (Téléportation expérimentale du Worms, réglage rayon de l'explosion)

26 février :
- Amélioration du déplacement du Worms (escalade ou non)
- Ajout de l'eau
- Ajout du saut

28 Février :
- Optimisation de l'affichage graphique pour gagner des fps (i.e. Images par secondes).
- Création d'une classe moteurPhysique qui gère des mouvements fiables et physiquement réalistes.

1er Mars :
- Amélioration du saut (éviter le double saut)
- Modification des valeurs de pesanteur et de vitesse pour le saut
- Optimisation de la gestion du terrain

3 Mars :
- Ajout d'un générateur de grotte aléatoire dans le terrain (contient des bugs)

5 Mars :
- Amélioration du générateur de grotte
- Ajout du double saut
- Ajout du Menu

6 Mars :
- Amélioration du menu (gestion des deux équipes, raccord avec le jeu)
- Ajout du choix de l'angle de visée expérimentale

7 Mars :
- Premier arme fonctionnelle : la grenade

8 Mars :
- Amélioration Menu
- Ajout générateur d'iles aléatoires
- Ajout du Bazooka fonctionnel

9-10 Mars :
- Affichage du nom et de la vie du Worms

12 Mars :
- Apparition aléatoire des Worms
- Ajout gestion des dégâts d'armes aux Worms
- Ajout de l'animation des explosions
- Ajout de l'inventaire
- Amélioration des iles
- Pierre tombale mort

14 Mars :
- Amélioration des dégâts (souffle d'armes), et dégâts de chute

19 Mars :
- Amélioration iles et failles :
- Amélioration graphisme du menu
- Gestions des tours (détection de fin de partie, morts, écran de fin, message en début de tour)
- Ajout HolyGrenade
- Amélioration du menu (noms)

21 Mars :
- Nouvelle génération de terrain (par image bitmap)
- L'eau devient mortelle
- Playlist de son
- Correction de bug rocket

25 Mars :
- Affichage des dégats

26 Mars :
- Amélioration de l'apparition des Worms
- Ajout de plusieurs cartes.

27 Mars :
- Ajout choix de la carte dans le menu
- Ajout de l'exécutable jar

28 Mars :
- Amélioration physique de la rocket (visée plus précise)
- Création exécutable + lanceur Windows/Linux