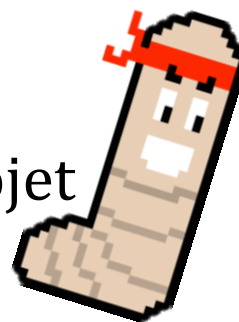




Worms

Rendu Final du Mini Projet



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

Table des matières

I – Cahier des Charges	1
II – Problème posé.....	1
III – Principe de l’algorithme	2
IV - Bibliographie	4
V – Suggestion d’améliorations du projet, bugs connus	4
VI – Carnet de route	5

I – Cahier des Charges

le cahier des charges de votre programme (quelles fonctionnalités sont proposées à l'utilisateur ?)

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 projectiles.

II – Problème posé

la description du problème posé (quelles sont les problèmes envisagées ?)

Supprimer cette partie ??????????????

III – Principe de l’algorithme

le principe de l’algorithme (comment les problèmes rencontrés sont-ils résolus par le programme)

Classe	Classe(s) héritière(s)	Description de la classe
Weapons	<ul style="list-style-type: none">- Bazooka- Grenade- HolyGrenade- Teleporteur <p><i>Rmq : HolyGrenade hérite de Grenade</i></p>	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	<ul style="list-style-type: none">- Rocket- GrenadeProjectile- HolyGrenadeProjectile <p><i>Rmq : HolyGrenadeProjectile hérite de GrenadeProjectile</i></p>	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 complètement un « Worms ». En effet, cette classe gère le nom, la couleur (i.e. l’équipe du

	Ø	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	Ø	Permet de générer le terrain où 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	Ø	Permet de stocker plus facilement les informations nécessaires à la génération d’un terrain (adresse du fond d’écran, textures, etc...)
Block	Ø	Permet d’associer des coordonnées x et y à un bloc.
Inventaire	Ø	Permet d’afficher une interface graphique interactive où le joueur peut sélectionner son arme.
GestionTours	Ø	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 réalisée par cette classe qui détecte lorsqu’une équipe n’a plus de joueur. Elle affiche alors l’écran de fin.

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.

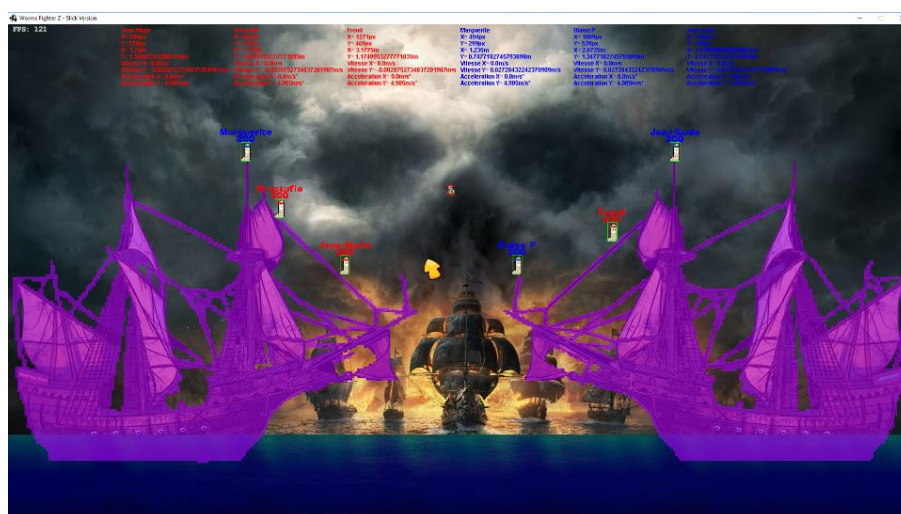


Figure 1 : Capture d’écran du mode montrant la physique du jeu

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.

IV- 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.

V – 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

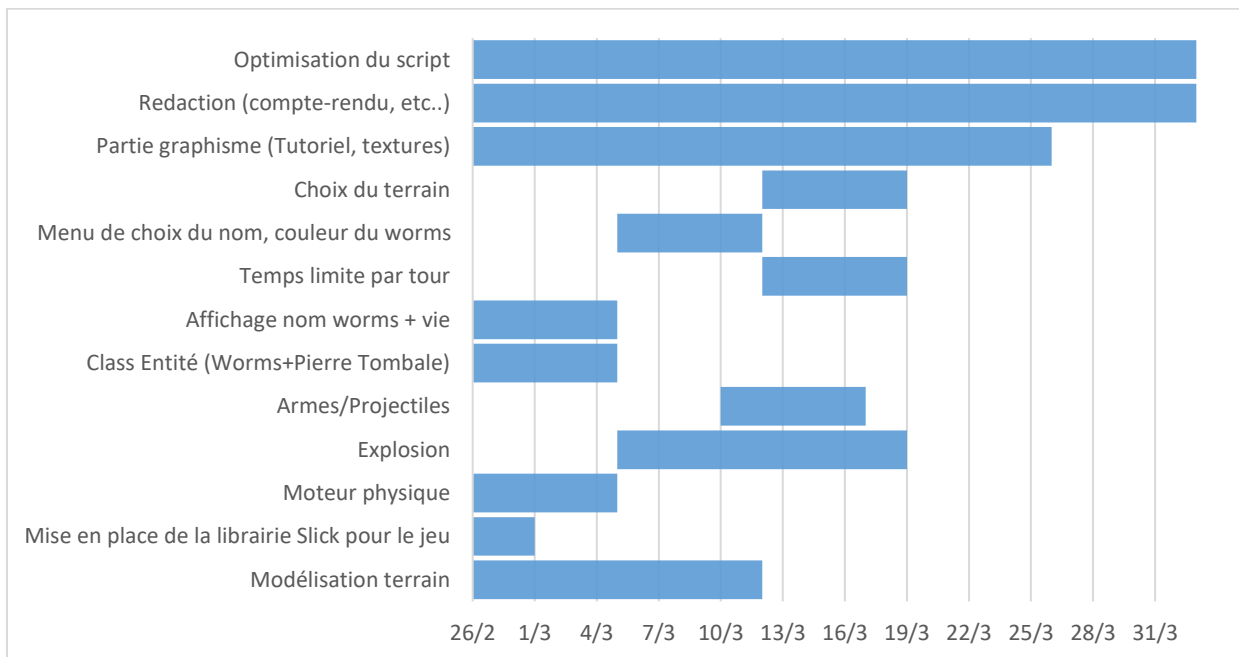
Néanmoins, ces bugs restent assez rares.

VI – Carnet de route

carnet de route et échéancier décrivant l'évolution de votre travail au fil des semaines

Insérer un tableau qui fait quoi

Tâches	Personne(s) de l'équipe en charge de la tâche



un document explicatif de 6 pages maximum au format PDF précisant