PROJET DE SYNTHÈSE D'IMAGES

OVER THE SEA

Steeve Vincent

Mélodie Mersch

IMAC 1

2015/2016

SOMMAIRE

INTRODUCTION

Dans *Over the sea*, le joueur doit manipuler un aéroglisseur afin de lui faire atteindre un maximum de *artefacts*. Le but du jeu est donc d'avoir le plus de points à la fin du temps imparti.

Ainsi, notre jeu ne comporte pas vraiment de mode « jouer seul » car sinon il n'y aurait pas vraiment d'intérêt à y jouer. Dans le mode « un joueur », le joueur joue contre l'ordinateur.

Voici une liste de nos travaux :

Eléments demandés et codés qui fonctionnent

- Manipulation de l'aéroglisseur au clavier : rotation gauche-droite, accélération
- Physique : application du principe fondamental de la dynamique pour le calcul de l'accélération et de l'effet des frottements
- Collisions avec les *artefacts* et les obstacles
- Gestion des bords de la carte
- Données de la carte éditées sur un fichier texte
- Zoom arrière et avant sur la carte
- Indication des check points non parcourus

Eléments non demandés (options) et codés qui fonctionnent

- Création de différentes cartes
- Points bonus
- Ajout d'un menu
- Rajout de son
- Mode multijoueurs (3 sur le clavier, manette et souris)
- Rajout d'une intelligence artificielle

Eléments non demandés (options) et pas codés mais pour lesquels nous avons des choses à dire

- Implémentation de quadtree
- Système ressort pour un meilleur calcul de collisions
- Choix dans le menu de différents aéroglisseurs
- Modes de jeu différents
- Choix des contrôles

I. MODÉLISATION PHYSIQUE

1) Mouvement de l'aéroglisseur

Afin de modéliser au mieux la trajectoire de l'aéroglisseur, nous lui avons appliqué le principe fondamental de la dynamique selon lequel, nous avons

$$\Sigma\vec{F} = m.\vec{a}$$

avec m la masse de l'aéroglisseur
$\vec{a}$ son accélération
$\Sigma\vec{F}$ la somme des forces appliquées sur l'aéroglisseur

Nous avons décidé de garder les forces de frottements et la force du moteur de l'aéroglisseur car ce sont les plus importantes dans notre modélisation. Elles permettront de restituer au mieux la trajectoire du bateau.

On a alors $\Sigma\vec{F} = \overrightarrow{F_f} + \overrightarrow{P_m}$

où $\overrightarrow{F_f} = -f.\vec{v}$ et $\overrightarrow{P_m} = p.\vec{v}$

avec f le coefficient de frottement
$\vec{v}$ la vitesse de l'aéroglisseur
p le coefficient propre à la puissance du moteur de l'aéroglisseur

La valeur du coefficient de frottement est choisie dans la fonction `initMap` car il dépend du type de terrain et la valeur de p est définie dans `initHovercraft` comme ce coefficient dépend de l'aéroglisseur.

La valeur de l'accélération en x et en y est définie dans la fonction `updateHovercraft`.
En projetant sur x, elle vaut $a_x = p.a.\cos(\alpha)$
En projetant sur y, elle vaut $a_y = p.a.\sin(\alpha)$
avec a le coefficient d'accélération et α l'angle de l'aéroglisseur par rapport au repère.
Elle sera ensuite appliquée sur l'hovercraft grâce à `updateObject`.

La force de frottements est appliquée dans la fonction `applyFrottements`, cette dernière met à jour les vitesses en x et en y en leur ajoutant respectivement $a_x - f.v_x$ et $a_y - f.v_y$

Enfin, la rotation de l'aéroglisseur est gérée dans `updateHovercraft` qui ajoute à sa vitesse angulaire $a_{Angle}.a_{rot}$
avec a_{Angle} l'accélération angulaire et a_{rot} le coefficient d'accélération angulaire.

2) Gestion des collisions

Pour le calcul des collisions, nous avons considéré que tous les objets sont composés de 3 types de formes : segment, polygone (ensemble de segments) et cercle. Savoir si deux objets se recontrent, c'est donc se demander quelles sont les formes qui sont en collision. Nous avons donc implenté différentes fonctions selon les types de formes qui entrent en choc.

- Collision entre deux cercles : `collision_C_C`
La fonction calcule la distance entre les deux objets et regarde si elle est inférieure ou non à la somme des rayons des deux cercles. La vitesse de l'aéroglisseur sera alors égale à l'opposé de sa vitesse.

- Collision entre un cercle et un polygone : `collision_C_P`
La fonction vérifie si un ou plusieurs point(s) du polygone se trouve(nt) dans le cercle ou non.
S'il y a une collision, on cherche ensuite quel segment est responsable de celle-ci. La nouvelle vitesse de l'aéroglisseur sera alors égale à l'opposé du vecteur directeur de ce segment.

II. DONNÉES DE TERRAIN

Les données de chaque terrain sont éditées sur un fichier texte qui est lu par le programme avant le lancement du jeu grâce à la fonction `readLine`.

Dans chaque fichier texte, il faut spécifier à la première ligne la texture du terrain, le coefficient de frottement, sa largeur, sa hauteur, la musique ainsi que la durée du jeu (en secondes) sous cette forme : « texture:___ frottement:___ largeur:___ hauteur:___ audio:___ time:___ »

A la ligne suivante, il faut inscrire le nombre d'obstacles et d'*artefacts* :
« nbobstacles:___ nbitems:___ »

Les sections suivantes donnent les informations sur les obstacles et les *artefacts*.

Pour chaque obstacle, il faut indiquer, à chaque ligne :
- son nombre de formes, son nombre d'effets, sa résistance, son rayon de collision, s'il peut bouger ou non (1 ou 0), sa force, s'il est cassable ou non (0 ou 1) , sa position :
« nbformes:__ nbeffets:__ vie:__ rayoncolision:__ statique:__ force:__ incassable:__ position:(__,__) »

formes

- son type (r pour rectangle et c pour cercle), les coordonnées de son sommet en haut à gauche et sa taille pour le cas du rectangle ou les coordonnées de son centre et son rayon pour le cercle, sa couleur (rouge, vert, bleu, opacité), s'il est rempli ou non (1 ou 0), sa texture :
« type:___ topleft:(__,__) size:(__,__) couleur:(__,__,__,__) rempli:__ texture:__ »

effets

- son type, sa résistance, son effet de rebond
« type:__ resistance:__ rebond:__ »

On fait ceci nbobstacles fois.

Pour chaque *artefacts*, il faut indiquer, à chaque ligne :
- son nombre de formes, son nombre d'effets, sa vie, son rayon de collision, s'il peut bouger ou non (1 ou 0), sa force, s'il est cassable ou non (0 ou 1) , sa fréquence d'apparition :
« nbformes:__ nbeffets:__ vie:__ rayoncolision:__ statique:__ force:__ incassable:__ freq:__ »

formes

- son type (r pour rectangle et c pour cercle), les coordonnées de son sommet en haut à gauche et sa taille pour le cas du rectangle ou les coordonnées de son centre et son rayon pour le cercle, sa couleur (rouge, vert, bleu, opacité), s'il est rempli ou non (1 ou 0), sa texture :
« type:___ topleft:(__,__) size:(__,__) couleur:(__,__,__,__) rempli:__ texture:__ »

effets

- son type (r pour rectangle, c pour cercle et p pour polygone), le nombre de points qu'il rapporte
« type:__ valeur:__ »

On fait ceci nbitems fois.

III. IMPLÉMENTATION DE L'INTELLIGENCE ARTIFICIELLE

Dans la structure `BotHovercraft`, nous avons implémenté une pile de direction pour les aéroglisseurs dirigés par l'ordinateur.

Celle-ci est vide au début du jeu. Ensuite, à chaque frame, la fonction `updateBotHovercraft` regarde quel est l'*artefact* le plus proche.
S'il n'y en a pas, aucune direction n'est donnée à l'aéroglisseur.

Sinon, si l'aéroglisseur a déjà une cible mais ce n'est pas la plus proche, la pile de direction va être vidée et une nouvelle va être initialiser. La nouvelle direction qui lui sera donnée sera celle de la ligne droite le reliant directement à l'*artefact*. S'il y a un obstacle, la fonction implentée va vérifier si on peut le contourner par la gauche ou par la droite. Elle va alors créer une nouvelle direction qu'elle va empiler.
On empile ainsi à chaque fois une direction. L'aéroglisseur suivra toujours celle qu'il a empilée en dernier. On dépile si la direction d'en-dessous ne mène plus à un obstacle.

On procède ainsi jusqu'à arriver à la cible.

IV. PROBLÈMES RENCONTRÉS

Après avoir testé notre code une dernière fois, nous nous sommes aperçus que les différentes commandes au clavier ne fonctionnaient pas sur Linux et que seule la manette était reconnue. Cependant, nous n'avons pas eu de problème sur Windows. Il ne nous a malheureusement pas été possible de résoudre ce problème par manque de temps.

CONCLUSION

Ce projet nous a permis de découvrir par nous-même comment pouvait être élaboré un jeu vidéo en 2D. En effet, il ne nous a été donné aucune structure pour créer ce jeu et nous devions donc l'imaginer.
De plus, ce projet a été enrichissant au niveau de la physique car nous avons du nous demander comment pouvaient être modélisées les collisions entre les objets.

Nous nous sommes également rendus compte du temps nécessaire pour créer un jeu qui peut paraître simple au premier abord. Malgré le temps imparti, nous sommes satisfaits du résultat. Cependant, si nous avions eu plus de temps, nous aurions pu laisser le joueur choisir son aéroglisseur dans le menu ou le type de contrôle qu'il veut utiliser (clavier, manette ou souris).