2016-2017
Projet C

Jeu de course en 2D

Mathieu HANOTAUX
Rodrigue KPAKPABIA

esiea
ECOLE D'INGENIEURS
DU MONDE NUMERIQUE

Encadreur : M Michael FRANçOIS

SOMMAIRE

REMERCIEMENTS

Nous tenons tout particulièrement à remercier notre suiveur, M Michael François son suivi tout au long de ce projet.

INTRODUCTION

Dans le cadre de la troisième année de cycle ingénieur à l'ESIEA, nous avons eu pour tâche de réaliser un projet en informatique, plus précisément un jeu de simulation automobile avec un mode de gestion automatique. Nous devions réaliser ce projet dans le langage C sous linux. Le but était de réaliser un jeu de voiture de type « Traffic Racer ». C'est-à-dire un jeu de voiture dans lequel on contrôle un véhicule et dont le but est de gagner un maximum de points en doublant d'autre véhicules contrôlés par une IA.
Nous devions aussi implémenter un mode autonome et construire une IA qui devrait accumuler le plus de points possibles tout en évitant de percuter les autres véhicules.

Dans ce rapport nous illustrerons comment le problème a été abordé et nous expliquerons les grandes lignes de la réalisation de ce projet.

1.ETUDE DU PROJET

Afin de mener à bien ce projet, une étude préalable du sujet était nécessaire.
Pour aborder le sujet, il a fallu mettre en place des structures de données afin de pouvoir gérer nos voitures qui apparaitront à l'écran.

Nous avons commencé par définir les structures que nous avons utilisé dans le programme répondant au minimum requis du sujet, et nous avons continué de remplir ces structures au fur et à mesure que l'on ajoutait des fonctionnalités au programmes.

L'affichage étant libre, nous avons décidé d'utiliser la bibliothèque SDL très puissante, permettant de réaliser des projets graphiques sans vraiment de limite.
De nombreuses bibliothèques annexes tels, SDL_GFX, SDL_Image, SDL_TTF, SDL_Mixer permettent de manipuler du texte, des images ou encore des sons simplement.
De plus, depuis la version 2.0 la SDL supporte l'accélération matérielle depuis le GPU augmentant considérablement les possibilités de création graphique ou d'animation.

Afin de construire un code le plus propre possible, nous avons essayé de suivre le plus possible le pattern design MVC. Ainsi dans le menu et dans le jeu nous avons séparé les évènements de l'affichage et de la mise à jour du jeu.

Durant la conception nous avons réalisé plusieurs versions du programme afin de pouvoir facilement annuler un bug qui serait survenu suite à une modification.

2.ELABORATION DU PROJET

- Le menu

Afin de réaliser le menu nous avons utilisé deux images, une d'un bouton enfoncé une autre d'un bouton avec du relief en 3D, ainsi quand l'utilisateur passe sa souris sur un des boutons, celui-ci change d'état et l'utilisateur comprends facilement qu'il peut cliquer dessus. Le menu supporter aussi les touches UP, et DOWN pour se déplacer et BACKSPACE pour entre dans le mode sélectionné. Ainsi il n'est pas nécessaire de posséder une souris pour utiliser le jeu.

- L'affichage d'une partie

Dans le coin à gauche en haut on affiche le HUD, c'est-à-dire que l'on peut retrouver ici les informations importantes sur la partie en cours, comme :

- Le temps depuis le début de la partie
- La vitesse actuelle
- Le nombres de points
- Le meilleur score

Juste en dessous on peut trouver un affichage du nombre de vie restant au joueur. Celui-ci commence une partie avec 5 vies, donc 5 cœurs rouges.
A chaque accident, il perd une vie et le cœur rouge est remplacé par un cœur grisé.
Quand le joueur n'a plus de vies/cœurs il perd la partie et le jeu se termine.

Centré à droite on retrouve le terrain sur lequel le joueur va affronter la circulation intensive des autres voitures.
Celui-ci est composé d'une petite image de route que l'on va répéter en fonction du nombre de routes et de la hauteur de celle-ci. L'utilisateur au final ne se rendra compte de rien.

De chaque côté de la route on retrouve les décors qui vont défiler en fonction de la vitesse du joueur.
Pour la route on va afficher les « images de routes » avec un certain décalage sur l'axe des Y. Ce DY va être proportionnel à la vitesse actuelle du joueur, sa vitesse maximum et la hauteur d'une image de route.
Ce système modulo la taille d'une image de route va permettre de faire défiler l'affichage de la route plus ou moins vites en fonction de la vitesse du joueur.

Il faut pour un bon rendu, avoir une image de route initiale qui a des points de repères particuliers pour voir le défilement, mais il faut qu'ils soient discrets. Avec la vitesse qui augmente l'effet d'optique du déplacement peut fatiguer les yeux si le repère est trop visible.

- Gestion des voitures

Afin de gérer facilement les voitures entre elles, on a choisi d'implémenter des listes chainées de nos voitures, qui sont facilement manipulable par des fonctions récursives (surtout utilisé pour l'affichage et certains calculs) et des parcours itératifs pratique pour comparer les voitures entre elles.

- Gestion des Décors

La gestion des décors est très similaire à celle des voitures, les structures étant très proches, (on peut considérer qu'un décor est une voiture qui à une vitesse nulle) nous n'avons pas eu besoin de recréer toutes les fonctions de gestions mais juste d'adapter celles des voitures.

- Gestion mémoire

Lors de la création d'une partie les images du jeu sont chargé en mémoire ainsi que les sons utilisés type klaxon et explosion.
A la fin d'une partie, toute les images sont supprimées pour éviter les fuites de mémoire.
Lors de la fermeture du programme on efface également les zone mémoire dédié à la musique ainsi que toutes les bibliothèques qui utilisent de la mémoire.
Les voitures possèdent chacune une image qui leur ais propre, choisie aléatoirement lors de leur génération. Cependant une image n'est pas chargée en mémoire pour chaque voiture sur l'écran. Les voitures possèdent une variable qui les relis à une image d'une couleur particulière.
Dans la fonction d'affichage on ne choisira d'afficher que les voitures qui sont comprises dans l'espace visible.

Les listes chainées sont générées à la volé pendant le déroulement du programme, et la mémoire est bien sûr libéré lors de la fin d'une partie aussi bien pour une voiture que pour un décor.

- Génération de décors et voiture

Afin de générer une nouvelle voiture au fur et à mesure de l'avancement du joueur dans la partie, il faut sélectionner une position initiale devant le joueur, mais pas au hasard si l'on veut éviter de faire apparaitre une voiture sur l'emplacement d'une voiture déjà existante.
La stratégie adoptée a été de rechercher dans la liste de voiture la voiture en tête de position, une fois trouvé on laisse une marge à la nouvelle voiture crée dépendante de la vitesse de la précédente.
La nouvelle voiture se voit donner une vitesse maximum aléatoire, une vitesse actuelle de 100km/h, une couleur une file de voiture (parmi les 4 voies) et d'autres attributs.
Le même système a été adopté pour la génération des décors avec les caractéristiques qui lui correspondent.

- Gestion des calculs
-

En utilisant des listes chainées et en faisant des comparaisons, on est amené à avoir des calculs de complexité N^2 ce qui est problématique vu la génération régulière de voiture.
Au bout d'un moment le programme va s'arrêter faute de mémoire ou de ressources processeur.
Nous avons choisi de limiter le nombre de voiture présentes dans les listes mais sans interférer sur l'expérience utilisateur.
Pour ce faire nous avons décidé de ne garder que 60 voitures au total, soit 30 devant et 30 derrière le joueur, nombre que nous avons jugé suffisant pour que l'utilisateur ne se rende compte de rien.

Ainsi la génération de nouvelles voitures est bloquée s'il y'a plus de 30 voitures devant et on va supprimer la voiture la plus éloigné du joueur si le même phénomène se produit derrière. Une fois de plus cette gestion a aussi été utilisé pour les décors.

- Gestion du déplacement

Pour gérer le déplacement des voitures et des décors on se réfère à la vitesse des autres voitures si une voiture se déplace plus vite que la nôtre alors on va faire monter sa position en Y, de même si une voiture va moins vite elle va descendre sur l'axe des Y.

- Gestion du mode autonome

Ce mode a été la vraie phase compliquée du projet, en effet comment faire éviter à notre voiture toutes les autres tout en allant plus vite.
Dans un premier temps nous avons réalisé l'IA autonome sur notre joueur et pour aller plus loin nous avons appliqué celui-ci à toutes les voitures.
Pour se faire nous avons choisi de comparer la position et la distance respectives pour chaque voiture et avec chaque autre voiture de la liste.
On récupère la distance présente avec la voiture devant nous la plus proche, mais aussi la voiture la plus proche devant à droite et à gauche pour savoir si on peut doubler.
On calcul aussi ces distances avec les voitures derrière à droite et à gauche. En effet comme dans la vie réelle il faut vérifier ses rétroviseurs/angles morts avant de doubler, ici on vérifie qu'une voiture n'arrive pas à toute vitesse pour nous percuter pendant la tentative de dépassement.

La stratégie qui suit est simple, on s'intéresse d'abord à la voiture devant nous si cette distance est suffisante (calcul proportionnel à la vitesse de notre voiture) alors on accélère jusqu'à notre vitesse maximum.
Si une voiture est devant nous alors on vérifie que l'on puisse doubler par la gauche ou la droite en vérifier bien la présence de voiture derrière.
Ensuite si on peut doubler on va choisir la file ou la voiture devant est la plus éloigné afin de parcourir un maximum de distance.

On applique ce raisonnement à toute les voitures de la liste et on obtient un système de circulation autonome.

Le système autonome à un nombre infiniment petit d'accidents, le programme continue à gagner des points sauf malchance jusqu'à la mise en veille de la machine.
Un compteur de collision disponible sur le HUD offre une bonne visibilité des performances du mode autonome.

- Gestion du mode manuel

Passer du mode automatique au mode manuel est facilité par l'utilisation des listes chainés, ainsi si le joueur est la tête de liste dans le mode auto, il suffit de l'enlever pour le mode manuel et on conserve une circulation autonome tout en laissant la gestion de la voiture au joueur.
Cependant n'étant plus membres de la liste les autres voitures ne nous calcules pas lors de la gestion automatique et elles peuvent sans problème nous rentrer dedans en doublant.

Cette fonctionnalité (doubler) pour les autres voitures a été retiré du mode manuel pour faciliter l'expérience utilisateur.

Le joueur peut utiliser les 4 touches pour se déplacer et cela presque simultanément. Excepter les touches UP et DOWN il est bien possible d'accélérer tout en tournant, cela est possible en vérifiant quand les touchent sont enfoncés et relevé à l'aide de booléens et non juste des événements de la SDL.

Il est possible de klaxonner en même temps que la conduite avec la touche espace.

- Gestion des collisions

Pour gérer les collisions nous avons utilisé une fonction très basique en SDL qui comparent la position de deux images.
Cette gestion n'est pas très précise car les images sont des rectangles tandis que l'affichage des voitures est plus arrondi.

Lorsque qu'une collision se produit on va afficher un effet d'explosions au centre des deux voitures concernées.
Cet effet dure quelques secondes est le produit de la superposition de plus de 60 images d'une explosion affiché les unes après les autres ce qui produit un effet très impressionnant. Ajouté à un bruitage d'explosion l'impression de réalisme est renforcé.

- Gestion des Scores

Les 10 meilleurs scores du jeu sont sauvegardés dans une fichier et son accessible à travers le menu après un appuis sur le bouton score.
On sauvegarde dedans uniquement le score des utilisateurs et non de l'IA.
On garde le nom de la personne qui est demandé à la fin d'une partie le score et le temps qu'a durée la partie.
Un joueur gagne des points uniquement la première fois qu'il dépasse une voiture, pour éviter d'avoir un joueur qui va garder la même vitesse qu'une voiture et engranger ainsi beaucoup de points.
La quantité de point gagné par dépassement est égale à la différence de vitesse entre le joueur et les autres voitures.