Implémentation du jeu Pong

Paul Nassiri

Kylian Leroy

Hugo Crepin

Sommaire:

Introduction :

Membres du groupe:

Nassiri Paul :

- Développement de l'adversaire
- Architecture du projet
- Création du système de minuterie
- Détection des buts
- Gestion des collisions de la raquette
- Lecture et parsing de la carte
- Rebond aléatoire de la balle
- Rédaction du rapport

Leroy Kylian:

- Collision et déplacement de la balle
- Gestion du score
    - Mise à jour à chaque but
    - Affichage au dessus du jeu
    - Condition d'arrêt de la boucle principale
    - Sauvegarde du score final dans un fichier externe
- Création de statistiques
    - Parsing d'un fichier externe
    - Calculer les valeurs souhaitées
    - Page d'affichage
- Création de la boucle principale
- Rédaction du rapport

Crepin Hugo:

- Création du menu
- Déplacement de la raquette
    - Détection des touches pressées
    - Mise à jour des positions
    - Vérifications collisions
- Création des diapositives et du plan pour la présentation
- Récupération des inputs instantané

Contexte:

Présentation:

Le projet consiste à développer le jeu Pong (page Wikipedia), en utilisant le langage de programmation Shell. Le jeu repose sur une mécanique simple, le joueur et le robot possèdent une raquette . Pour gagner un point il faut faire rebondir la balle sur sa raquette et faire passer la balle derrière la raquette de son adversaire pour gagner un point. Le premier joueur à 3 points gagne !

Le jeu est adapté pour fonctionner sur un terminal et pour donner au terminal bash une dimension dynamique et interactive.

Objectif :

- L'objectif central est de concevoir un jeu vidéo capable de fonctionner directement dans un terminal sans utiliser de bibliothèques ou d'outils externes.

- Le joueur doit pouvoir se déplacer immédiatement sans délai ni confirmer par entrée comme pour une commande classique. On cherche une réactivité des touches sans délai (que ce soit dans le jeu ou dans le menu)

- Un autre objectif est un affichage fluide du jeu pour offrir au joueur une expérience similaire à un jeu vidéo classique. Contrer les limitations du terminal pose un problème intéressant et nous force à maîtriser toutes les commandes qu'il nous met à disposition.

- Travailler sur ce jeu offre une excellente opportunité pour monter en compétence en Bash. Notamment sur 3 aspects qui nous paraissent fondamentaux, la lecture et écriture de fichiers, l'utilisation des structures de contrôle ainsi que la modularisation en bash.

- Outre le bash, le jeu vidéo sur un terminal offre un vrai défi d'algorithme par le nombre de tâches effectué entre chaque rafraîchissement. Il y a un réel objectif d'optimisation du jeu notamment dans la boucle principale. Nous décidons pour pousser au maximum l'utilisation du bash de ne pas utiliser de structure de données.

- Nous souhaitons profiter des SAE pour nous préparer au monde de l'entreprise et améliorer notre travail d'équipe. L'objectif est de développer nos compétences techniques et organisationnelles pour simuler des conditions de travail proches de celles rencontrées en entreprise notamment en utilisant github.

Fonctionnement général:

L’application est un jeu à 1 joueur contre une IA basique. Avant de pouvoir jouer, l’utilisateur sera dans un menu contenant plusieurs informations. Le joueur contrôle sa raquette au clavier en utilisant les touches q et d. Le score est utilisé pour créer les statistiques du joueur.

Algorithmes :

Architecture du projet:

L’architecture du projet consiste en 3 répertoires (modeles, shared et scripts).

- modeles : Répertoire contenant des modules spécifiques gérant les différents éléments clés du jeu comme la balle (balle.sh), la raquette (raquette.sh), l’adversaire (bot.sh) et le score (score.sh).

- shared : Répertoire contenant les fonctions et données utilitaires communes au reste de l’application, comme les valeurs de base du jeu (variables.sh) ou des fonctions fréquemment utilisées (utils.sh).

- scripts : Répertoire contenant les scripts nécessaires aux différentes fonctionnalités du jeu, comme la boucle principale (jouer.sh), la gestion du menu (menu.sh), des statistiques (stat.sh), et de l’analyse de la carte (niveau.sh).

Le projet comprend aussi le fichier main.sh permettant de lancer le jeu, et 2 fichiers texte map.txt et historique.txt étant utilisés respectivement pour l’initialisation de la carte et des statistiques.

Boucle principale:

(flowchart en annexes)

En amont de la boucle principale, le programme charge les différents modules, lis la carte donnée dans le fichier et initialise avec le contenu de la carte les variables du jeu.

La condition de continuation de la boucle est que le score d’aucun joueur n’est supérieur à une variable contenu dans variable.sh

À chaque itération, le clavier est lu afin de déplacer la joueur, la position et l’accélération de la balle est mise à jour, le bot est déplacé, puis nous vérifions si il y a un but

et mettons à jour le score en conséquence. La gestion du temps avec date permet l'utilisation d'un système de minuterie qui va déterminer si l'une des actions précédentes est réalisée ou non à chaque itération.

Après la boucle principale, le gagnant est déclaré, le score final est alors sauvegardé dans l'historique, et nous pouvons retourner au menu.

Fonctions principales:

lire_niveau :

La fonction lire_niveau lit un fichier .txt contenant la configuration du niveau sous un format personnalisé, similaire à un CSV. La fonction interprète ces lignes et initialise les objets correspondants en appelant leurs fonctions dédiées.

afficher_niveau :

La fonction afficher_niveau affiche dans le terminal les différents objets initialisés par lire_niveau, en utilisant la fonction écrire. Elle est appelée uniquement au début de la manche pour dessiner l'état initial. Par la suite, seuls les éléments modifiés sont mis à jour (comme dans deplacer_raquette).

deplacer_raquette :

deplacer_niveau vérifie si la touche q ou d est pressé, puis elle vérifie que le déplacement soit valide vis à vis des collisions. La dernière case de la raquette est ensuite effacée et une nouvelle case de la raquette est affichée dans le sens de la direction afin de ne pas tous les effacer puis réécrire.

deplacer_balle :

deplacer_balle gère les déplacements de la balle, détecte les collisions avec murs/objets, calcule les rebonds, met à jour sa direction et sa position, puis l'affiche dans le terminal.

Fonctions communes:

ecrire : Prend 2 nombres et un caractère, écris ce caractère aux coordonnées données par les nombres dans le terminal.

charAt : Prend 2 nombres et renvoie un entier dont la valeur dépend du caractère aux coordonnées données par les 2 paramètres. Permet notamment de vérifier s' il y a une collision.

Commandes et options de lancement:

Afin de lancer le projet, il faut exécuter le fichier « ./main.sh », qui lancera le menu. Il n'y a pas d'option de lancement particulière. Pour changer la carte, il faut modifier le fichier « map.txt ». Si la carte est dans un fichier d'un autre nom, il est nécessaire de remplacer l'assignation de la variable map dans le fichier «jouer.sh » (ligne 16) par le nouveau fichier.

Gestions des entrées / sorties:

Lecture d'une carte:

La lecture de la carte consiste à la parcourir et à initialiser les variables qui serviront au jeu en fonction du contenu du fichier texte donné dans la fonction jeux.

Ce fichier texte possède un format particulier. Chaque ligne commence par une lettre, correspondant au type d'objet que la ligne décrit, suivi de 2 à 4 nombres en fonction de l'objet, pouvant indiquer la position, la longueur ou la taille. Un exemple est donné en annexe.

(Voir map.txt en annexes)

Exportation des résultats et parsing pour analyse:

Après la partie, le résultat sera inscrit dans le fichier historique.txt, qui va contenir les scores de toutes les parties. Ce fichier sera ensuite parcouru et son contenu utilisé pour la page de statistiques du menu du jeu.

Conclusion:

Bilan:

En conclusion, nous avons atteint nos principaux objectifs. Le jeu offre une fluidité satisfaisante, avec une latence minimale entre les inputs, et a été réalisé sans recours à des bibliothèques ou outils externes. De plus, ce projet nous a permis de progresser significativement en programmation Bash et en algorithmie. Sur le plan collaboratif, nous avons franchi un cap en professionnalisation grâce à l'utilisation de GitHub, qui a facilité le travail en parallèle, le versionnage, ainsi que la correction et la vérification mutuelle du code.

Améliorations possibles:

Il existe plusieurs pistes d'améliorations quant au projet rendu, notamment l'implémentation de la vérification ou de la création de map, qui actuellement sont créées à la main et dont le contenu peut trop facilement faire faire une erreur au jeu.

Une autre amélioration possible sont les options de lancement, comme la spécification du niveau du bot, ou le choix de la carte, qui est pour le moment renseigné dans le code de l'application.

Finalement, l'implémentation du son et de plus de couleurs, afin d'améliorer l'expérience de l'utilisateur, serait possible.

Difficultés rencontrées:

Parmi les difficultés rencontrées, il y a le mouvement de la balle, qui était trop rigide et parfois saccadé par le nombre de tâches à faire entre deux frames, l'implémentation d'un système de minuterie à permis une animation plus fluide.

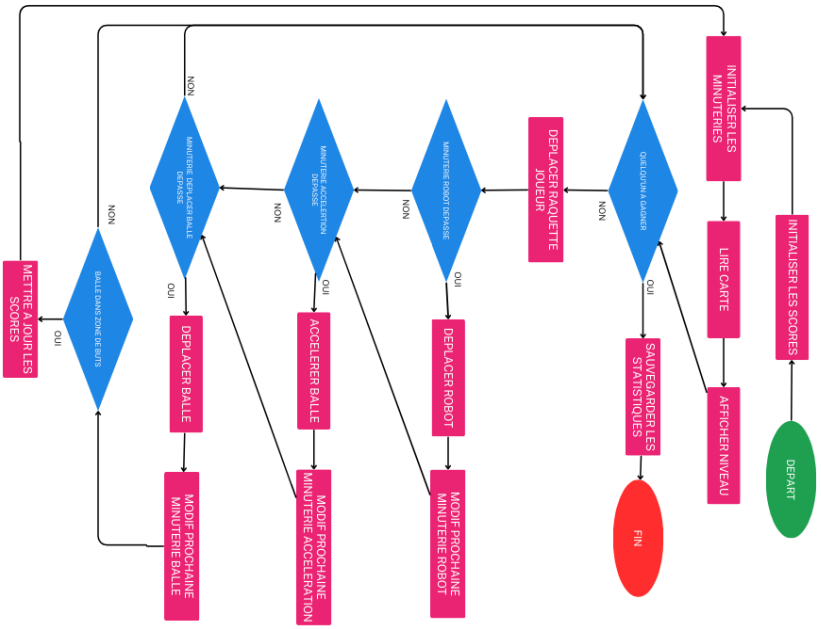
Enfin, travailler sans librairies et avec une utilisation minime d'internet nous a forcé à trouver par nous même des solutions et à nous améliorer algorithmiquement.

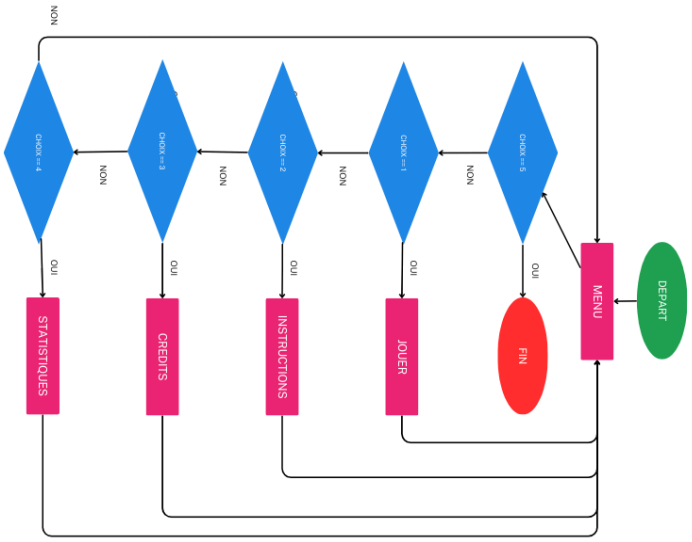
Annexes :

map.txt

```
w,1,10,1,30
w,30,10,1,30
p,5,39,3
b,5,10,3
o,5,20
gp,1,41,30,1
gb,1,8,30,1
```

map.txt


INITIALISER LES MINUTERIES
LIRE CARTE
AFFICHER NIVEAU
INITIALISER LES SCORES
DEPART
QUELQU'UN A GAGNER
SAUVEGARDER LES STATISTIQUES
FIN
DEPLACER RAQUETTE JOUEUR
MINUTERIE ROBOT DEPASSE
DEPLACER ROBOT
MODIF PROCHAINE MINUTERIE ROBOT
MINUTERIE ACCELERATION DEPASSE
ACCELERER BALLE
MODIF PROCHAINE MINUTERIE ACCELERATION
MINUTERIE DEPLACER BALLE DEPASSE
DEPLACER BALLE
MODIF PROCHAINE MINUTERIE BALLE
BALLE DANS ZONE DE BUTS
METTRE A JOUR LES SCORES
OUI
NON


DEPART
MENU
CHOIX == 5
CHOIX == 1
CHOIX == 2
CHOIX == 3
CHOIX == 4
FIN
JOUER
INSTRUCTIONS
CREDITS
STATISTIQUES
OUI
NON