R5.A.10 Nouveaux paradigmes de base de données

Rapport sur le jeu Punto de Clément MONFORT

mécanique du jeu

Technologie utilisée

Le langage de programmation utilisé pour développer le **Punto** est le **C** avec la librairies **OCML**, une librairie maison basée sur la librairie **OCL** et **CSFML** un binding de la librairie graphique **C++ SFML**. Le principal défaut est que je devais presque tout développer à partir de rien ce qui donne un temps de développement plus long qu'un autre langage mais offre cependant un contrôle totale sur ce qui se passe dans l'application et de meilleure performance.

Pour installer les dépendances un script shell est disponible mais il fait pour les distro Debian uniquement, il installe **docker**, **C**, **CSFML**, **make**, **gcc**, **libbson** et les drivers pour **mongodb**, **MySQL** et **MySQL**.

Le langage de programmation utilisé pour le déploiement est un **script shell** utilisant l'outil make (un ***Makefile***) pour déployer les 2 **containers docker MySQL** et **mongodb** ainsi que compiler le programme **Punto.exe**. La base **SQlite** est déjà présente dans l'application mais une commande est fournie dans le **readme** pour la générer à nouveau.

Programmation & algorithme

Le jeu à été divisé en plusieurs sections :

- Une partie contenant les ressources graphiques nécessaires comme les **polices d'écritures** ou les **sons**.

- Une partie contenant la scène du jeu c'est à dire les éléments représentant l'interface graphique et le plateau de jeu et son état actuel comme le score maximal et à qui il est attribué.

- Une autre partie représentant le groupe de joueurs, leurs identité, l'état de leur decks...

- Et enfin la partie système qui contient la fenêtre de jeu et éléments sonores ainsi que les variables importantes comme le nombre de cartes à aligner, des compteurs pour le nombre de tours et des manches jouer ainsi que le tour du joueur en cours.

Les cartes & Plateau de jeu

Les **cartes** contiennent des **variables** pour leurs **couleurs** ainsi que leurs **valeurs** et sont graphiquement des élément appelé **sfShape** d'une forme carré avec un **texte reflétant leurs valeurs**, les bordures de la carte ainsi que le texte porte la **couleur** qui est attribué à la **carte**.

Le **plateau de jeu** est une **matrice** d'une **longueur 6** par une **largeur 6** de **carte blanche** d'une **valeur** de **0**.

Les decks des joueur est un **tableau** de **9 à 18 cartes** dépendamment de si ils sont **4** ou **2**, elle on une valeur de **1** à **9** et une **couleur jaune**, **rouge**, **verte** ou **cyan**, mélanger en début de partie grâce à une **génération procédurale** de **nombre pseudo aléatoire**.

Les Joueurs

Les joueurs sont représentés par un **tableau de joueur** qui à une **taille** de **2** ou **4**. Les **objet joueur** sont composé d'un **deck**, une **série** de **carte monochromatique** avec une valeure allant de **1** à **9** si la **taille du groupe** est de **4** ou de **deux série** avec **deux couleurs distinct** si le groupe est d'une **taille de 2** et des **variables** comme la **taille courante du deck**, le **nom du joueur** et d'un **compteur du nombre de tour** jouer.

Quand un **joueur** à **fini son tour** on passe au **prochain** joueur du tableau, quand le compteur atteint **la taille du tableau** il retourne à **zéro** et on **incrémente** le **compteur** du nombre de **manches**.

Interface Utilisateur

Les joueurs disposent d'un **A.T.H** (affichage tête haute) au dessus du **tableau de jeu** qui indique le **nom** du joueur en train de jouer ainsi que la **carte** qu'il doit jouer, combien il lui reste de **carte dans son deck**, le nombre de **mouvements**, **tours** et **manches** effectués.

Le jeu s'affiche dans une **fenêtre en plein écran** sur l'écran le plus à droite, il affiche des **messages** dans la **console** concernant l'insertion dans la base de données, et des messages de fin de partie.

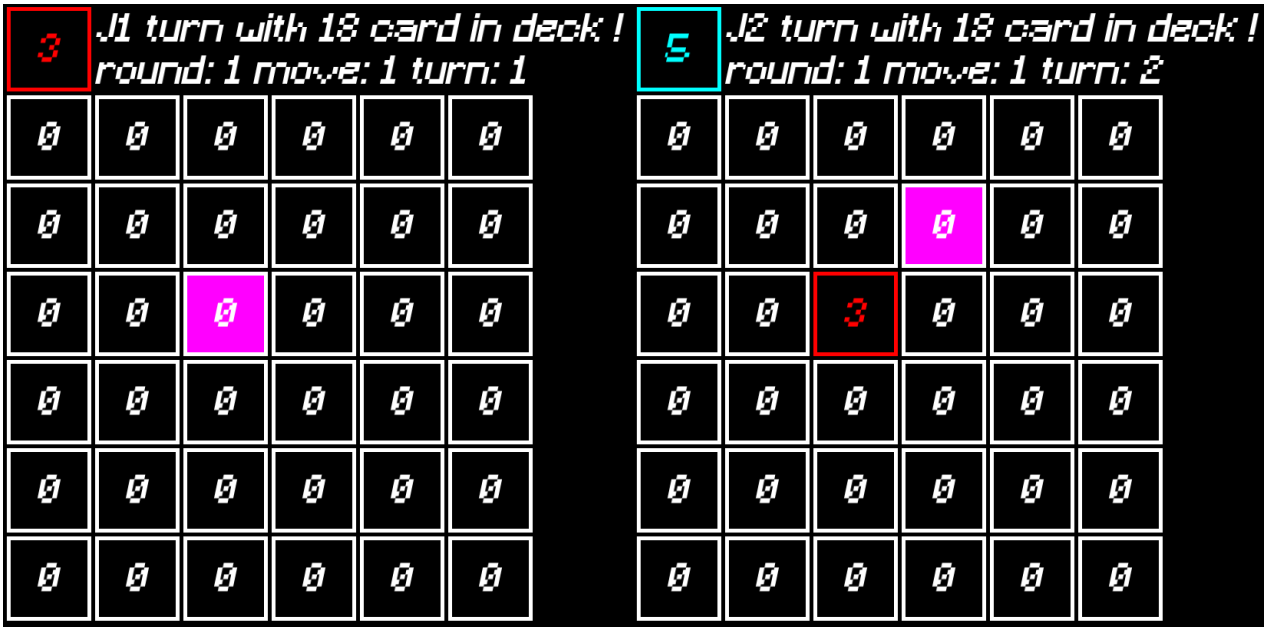
Gestion des événement

Les **événements** représentent toutes les **entrées** réalisées par le joueurs ainsi que les **réaction** du plateau de jeux relative à ses derniére :

- Toute action à une réaction sonore et graphique.

- Les actions de fin de partie :
  - Appuyer sur echap ou Alt + f4.
  - En cas de combinaison gagnante.
  - Si tous les decks sont vides.
  - Si il n'y aucune carte plus petite que la carte courante du joueur.

- Indiquer au joueur où est placée la souris comparé au tableau de jeu :
  - Si la souris est bien dans le plateau de jeu :
    - La carte sera afficher en **blanc** si la souris est sur la carte.
    - La carte sera affichée en **magenta** si il est possible d'insérer la carte.
      - Il est possible d'insérer une carte si la carte du plateau est plus petite et à une couleur différente de la carte du joueur.
        - Et est dans l'une des 4 cases du centre.
        - Ou à une carte d'une autre couleur que blanche dans ses 8 cartes adjacentes.
          - L'événement est enregistré dans la base de données dans la table **Event**.



- Trouver le plus grand score :
    - Quand un joueur insère une carte on récupère ses coordonnées et on vérifie la ligne x ainsi que la colonne y pour vérifier s'il y a un meilleur score.
        - Si les coordonnées suivent l’un des vecteur des diagonales elles sont aussi vérifiées.
            - Si le plus grand score trouvé a une suite plus longue que le meilleur score actuel ou la même longueur mais un score plus petit il devient le meilleur score.
                - Si le meilleur score à une longueur égale à la longueur maximum on envoie un événement de fin de partie.
                    - Le score est enregistré dans la base de données **Highscore**.

Bases de données

Utilisation

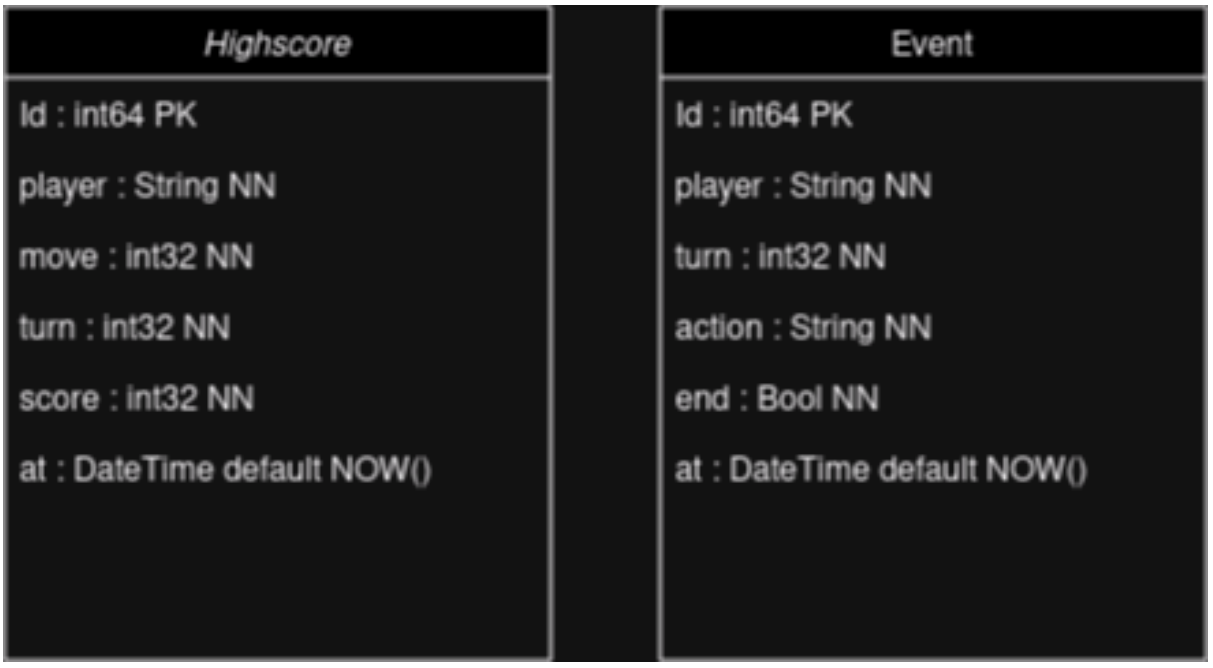
Les projets possèdent **3 bases de données**, les joueurs doivent en choisir une en début de partie et ne peuvent pas en changer avant la fin de la partie en relançant le jeu.

Déploiement

Il y tout d'abord une base de données relationnelle **MySQL** dans un c**onteneur docker** à l'adresse ***127.0.1.1:21002/tcp*** et une base **NoSQL mongodb** dans un conteneur docker à l'adresse ***127.0.1.1:21000/tcp***, Les deux sont construits depuis un Dockerfile et déployés avec un docker compose.

Enfin une base embarquée **SQLite** dans le fichier ***data/db/punto.db***, les requête à ces dernières sont directement faites par une connexion directe depuis l'application ***Punto.exe*** par le biais de librairies et drivers et non par le biais d'une api.

Architecture des Tables



- **Event** qui sert de **backlogs** de toutes les parties de **Punto** (les mouvements des joueurs) avec leur **noms**, le **tour** à laquelle **l'action** à été effectuer un **texte** expliquant l'action et un **booléen** indiquant si l'action à **terminé** la partie.

- **Highscore** qui contient tous les **meilleurs scores** de chaque partie, avec le **nom** du joueur, le nombre de **tour** totales de la partie, le nombre de **mouvements** qu'a fait le joueur et son score.

- Toutes les tables ont un identifiant propre et comportent la date et le temps de l'insertion de l'élément et Les insertion mongodb passe par un fichier bson avant d'être convertie en json.

Auto Insertion

Il est possible d'insérer un nombre de parties avec la fonction **auto** de **Punto.exe**, pour ce faire un **nombre aléatoire** est choisie entre le nombre de **tours minimum** et **maximum** d'une partie, pour chaque mouvement une insertion de de carte d'une **valeure et coordonnée aléatoire** est réalisé, la couleur l'est aussi si la taille du groupe est de 2.

```
→ Punto git:(main) ✗ ./Punto.exe auto sqlite 4 4
sqlite connected !
Generating 25 move.
Generating 24 move.
Generating 32 move.
Generating 20 move.
```

Autre utilisation

Transfert

Mis à part les requêtes qui peuvent être réalisées sur les bases il est possible de transférer toutes les informations d'une base grâce à la fonction **migrate** de **Punto.exe** de données vers une autre mais ne vide aucune d'elle.

```
→ Punto git:(main) ✗ ./Punto.exe migrate sqlite mysql
mysql connected !
sqlite connected !
```

Affichage

Grâce à la fonction **score** de **Punto.exe** il est possible de visualiser les scores d'une base de données.

```
→ Punto git:(main) ✗ ./Punto.exe score mongodb
mongodb connected !
———————— SCORE BOARD ————————
NAME    MOVE    TURN    SCORE
→ Punto git:(main) ✗ ./Punto.exe score sqlite
sqlite connected !
———————— SCORE BOARD ————————
NAME    MOVE    TURN    SCORE
J2      4       18      17
J1      6       25      17
J3      8       35      20
```