NEWTONOID

Projet Programmation Fonctionnelle 2SN

1 Sujet d'étude

Le jeu *Arkanoid* est un casse-brique commercialisé en 1986 [1]. Le but de ce projet est de reproduire plus ou moins le comportement de ce type de jeu, notamment en traitant *a minima* les points suivants :

— simuler le mouvement et les rebonds d'une balle, sur la raquette, les briques et les bords de l'écran. La balle sera constamment et très légèrement accélérée au fur et à mesure de la partie et en cas de rebond sur un objet. Elle sera également soumise à la pesanteur (comme en TP).
— gérer le déplacement de la raquette, en fonction des mouvements de la souris. La vitesse de la raquette lors du contact avec la balle doit permettre de communiquer une certaine impulsion à cette dernière, afin de l'accélerer ou de la ralentir horizontalement.
— représenter les briques présentes et détecter les collisions avec la balle, auquel cas les briques touchées disparaissent.
— calculer et afficher un score en fonction par exemple des briques touchées, qui peuvent porter des valeurs différentes.
— gérer la perte de balle et le redémarrage ou non de la partie en fonction du nombre de balles restantes.

Des *extensions* à ces exigences minimales sont envisageables (jouez à Arkanoid pour les découvrir !), parmi lesquelles :

— une animation pour les briques qui tombent lorsqu'elles sont touchées (au lieu de simplement disparaître).
— un pouvoir spécial pour certaines briques qui changent la configuration : accélération/ralentissement de la balle, changement de gravité, inversion des contrôles, élargissement/rétrécissement de la raquette, duplication de la balle, etc.
— un changement de tableau, avec éventuellement une autre configuration de briques, lorsque toutes les briques ont été touchées.
— des briques indestructibles, ou destructibles après plusieurs contacts seulement.

2 Critères de notation

Ce qui est évalué avec un **poids important** :

— la lisibilité du code et la qualité des contrats et des commentaires
— la présence de tests (unitaires et système)
— un des plus importants critères de notation est lié à la **conception**, en particulier **l'abstraction** et la **modularité** : emploi de types abstraits (signatures, modules, éventuellement modules paramétrés), définition de types pour les objets traités dans le jeu (note : `type t = int * int` n'est pas une *définition* de type, cela déclare juste un alias !).
— critères de conception et d'algorithmique **obligatoires** :
  — vous devez utiliser les *flux* pour représenter l'évolution de l'état du jeu (trajectoire de la balle, état de l'espace de jeu, score...).
  — la détection de collision doit reposer sur un algorithme efficace (inspirez-vous *par exemple* de la « *circle AABB detection collision* »).
  — la gestion efficace de l'espace de jeu doit reposer sur l'utilisation d'une structure de données efficace (utilisez un « *quadtree* » ou équivalent).
— la modularité sera évaluée, notamment la facilité pour changer les paramètres dynamiques, la configuration initiale des briques et de la balle, etc.

1. `https://fr.wikipedia.org/wiki/Arkanoid` et `https://www.youtube.com/watch?v=k1kKf8AdRac`

— un **rapport** doit être fourni et sa qualité compte *fortement* dans les critères de notation. Le rapport n'a pas besoin d'être long mais il doit être utile! C'est-à-dire qu'il doit aider à « rentrer » dans votre code, et permettre de comprendre vos choix les plus importants : choix de conception (abstraction, modularité, types, organisation des fonctions importantes) et choix algorithmiques majeurs. Un regard critique sur votre propre travail sera apprécié (choix rétrospectivement bons ou mauvais p. ex., lacunes, pistes d'amélioration intéressantes, blocages...).

Quelques points à respecter :
— dans tous les cas, votre application doit s'exécuter comme un programme binaire (pas seulement avec `utop`) : les professeurs utiliseront la commande `dune exec bin/newtonoid.exe` pour tester votre programme.
— le « plantage » de votre application par l'exception `Graphics.Graphic_failure` quand on clique sur la croix de fermeture de la fenêtre ne sera pas évalué négativement (il s'agit d'un bug de `Graphics`).
— le code devra être majoritairement fonctionnel, sauf par exemple en ce qui concerne les entrées-sorties (clavier, souris, écran, etc), pour lequel vous pouvez vous inspirer du code du TP7.
— l'effort avec lequel vous aurez développé des extensions, au delà du minimum requis, fera l'objet de **bonus**.
— le rendu graphique (et sonore) ne sera **pas** évalué. Seule une utilisation minimale de la bibliothèque `Graphics` est requise pour afficher la balle, les briques, etc. Un tel exemple d'utilisation minimal est proposé figure 1.

3 Travail

— Ce projet sera effectué par **groupes de 3 à 4 personnes** (pas plus !).
— Un squelette d'organisation du projet vous est fourni et **doit** être respecté.

4 Check-list pour le rendu

— un rapport —au format PDF uniquement— sera joint au code source dans le répertoire `rapport`.
— une archive —au format `.tar` uniquement— contenant vos fichiers sources et vos tests sera remise sous Moodle. **Faire un** `dune clean` **avant de produire l'archive.**

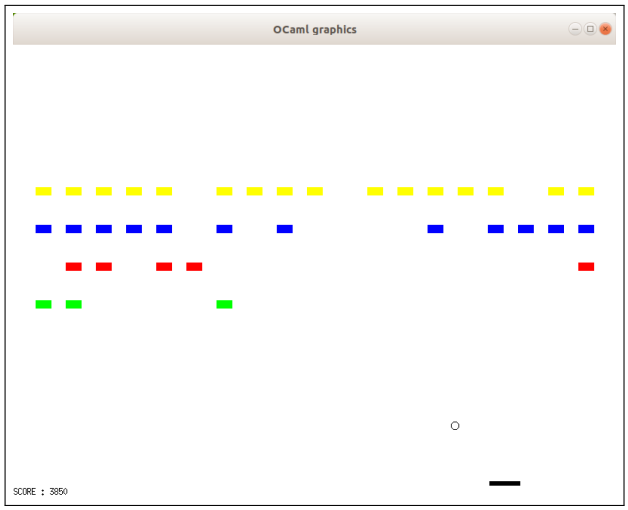

FIGURE 1 – Exemple d'interface graphique minimale.