Projet Informatique – Sections Electricité et Microtechnique

Printemps 2019 :*Obstructed Dodgeball* © R. Boulic

Rendu2 (dimanche 14 avril 23h59)

1. Buts, forme du lancement du programme, organisation du code et tests

But : lecture et affichage graphique, sauvegarde, relecture, test de l'interface graphique.

SyntaxeS du lancement de l'exécutable pour ce rendu2 (section 7.3.1 de la donnée) :
Le mode **Error** doit **continuer à fonctionner comme pour le rendu1**, c'est-à-dire qu'il n'y a pas de création d'interface graphique et qu'on quitte le programme soit à cause de la première erreur, soit après lecture avec succès. A partir du rendu2, ces deux syntaxes *supplémentaires* doivent être opérationnelles :

```
./projet nom_fichier.txt
```
ou
```
./projet
```

Avec ces deux syntaxes on lance le programme avec l'interface graphique. La première syntaxe ouvre immédiatement le fichier fourni sur la ligne de commande et, en cas de succès, affiche l'état initial du jeu. En cas d'échec, c'est-à-dire dès la première erreur détectée à la lecture du fichier, le message d'erreur est affiché, les structures de données sont effacées puis le programme attend qu'on utilise l'interface graphique pour ouvrir un autre fichier.

Avec la seconde syntaxe, le programme attend qu'on utilise l'interface pour indiquer un fichier à ouvrir. Le comportement est le même que ci-dessus pour le traitement de la lecture.

Architecture à adopter (fig 9b de la donnée) :
Le module projet contient la fonction main() en charge d'analyser si la ligne de commande est bien conforme aux **3** syntaxes acceptées à partir de ce rendu2.

Responsabilités du module **gui** : le traitement des deux syntaxes supplémentaires par main() va conduire à déléguer la création de l'interface graphique avec le module **gui** pour créer l'interface visible à la Figure 7 de la donnée. Les commandes « Exit », « Open », « Save » et l'affichage du message de **succès** ou **d'échec** à droite des boutons devront être opérationnels (section 5 de la donnée).

Ce module **gui** exploitera l'interface du module **simulation** pour déléguer les commandes de lecture et sauvegarde de fichier. Comme pour le rendu précédent, le module **simulation** délègue aux autres modules du Modèle les tâches de gestion de leur structure de données (lecture, sauvegarde). Le module **gui** peut aussi exploiter le module **tools** si nécessaire.

L'utilisation des commandes « Start/Stop » et « Step » seront (très) partiellemement testées, également en faisant appel au module **simulation**. Le module **simulation** offrira une implémentation (très) partielle à l'aide de *stubs* car la simulation du jeu ne sera disponible qu'au rendu final (relire la section 4.3 de « méthode de developpement de projet »). Il est suffisant d'*afficher un message* confirmant le changement d'état du jeu ou la demande de simulation d'un seul pas de simulation (Step).

C'est aussi le module qui doit effectuer la transformation de coordonnées entre le modèle et la fenêtre GTKmm pour le dessin des éléments représentant l'état du jeu. Seul le module **gui** peut effectuer des appels à GTKmm.

Evaluation du rendu2 : en plus d'évaluer les critères sur la qualité du code (ne pas oublier de corriger tout warning obtenu au rendu1), nous effectuerons une évaluation manuelle de votre programme comme suit :

- il doit continuer à fonctionner en mode **Error** : nous lancerons au moins un test dans ce mode.

- lancement avec les 2 nouvelles syntaxes
- types de séquence de test SANS relancer le programme :
    - lecture avec succès, suivi d'une sauvegarde, suivi d'une lecture différente avec succès, suivi de la lecture du fichier sauvegardé, etc…
    - lecture avec échec, lecture avec succès, lecture avec échec, etc…
      Dans le cas d'une lecture avec échec les structures de données doivent être détruites pour ne pas perturber la lecture suivante. Vous pouvez rafraichir l'affichage après cette destruction si vous désirez en avoir une confirmation visuelle (écran blanc) mais ça n'est pas obligatoire ; la démo ne le fait pas.
- un Clic sur **Start** doit le faire passer à **Stop,** et vice-versa, et faire afficher un message depuis une fonction stub de simulation.
- un Clic sur **Step** doit faire afficher un message depuis une fonction stub de simulation.

- **Affichage** : les proportions, formes et couleurs des éléments du jeu doivent être respectées. La convention des couleurs des joueurs en fonction du nombre de touches restant est classique : elle exprime un danger croissant quand le nombre de touches restant nbt diminue.

nbt = 4, safe ,	Green
nbt = 3, getting hot here,	Yellow
nbt = 2, pretty risky,	Orange
nbt = 1, near death experience,	Red

2. Forme du rendu2

Pour chaque rendu **UN SEUL membre d'un groupe** (noté **SCIPER1** ci-dessous) doit télécharger un fichier **zip** sur moodle (pas d'email). Le non-respect de cette consigne sera pénalisé de plusieurs points. Le nom de ce fichier **zip** a la forme :

SCIPER1_ SCIPER2.zip

Compléter le fichier fourni **mysciper.txt** en remplaçant 111111 par le numéro SCIPER de la personne qui télécharge le fichier archive et 222222 par le numéro SCIPER du second membre du groupe.

Le fichier archive du rendu1 doit contenir (**aucun répertoire**) :

- Fichier texte édité **mysciper.txt**
- Votre fichier **Makefile** produisant un executable **projet**
- votre code source (.cc et .h)

*On doit pouvoir produire l'exécutable **projet** à partir du Makefile après décompression du contenu du fichier zip.*

Auto-vérification : Après avoir téléchargé le fichier zip de votre rendu sur moodle (upload), récupérez-le (download), décompressez-le et assurez-vous que la commande make produit bien l'exécutable et que celui-ci fonctionne correctement.

Exécution sur la VM: votre projet sera évalué sur la VM du second semestre (disponible depuis le 6 mars).

Backup : Vous êtes responsable de faire votre copie de sauvegarde du projet. Il y a un backup automatique seulement sur votre compte myNAS. Sur la VM, vous devez activer vous-même le backup (icone « engrenage » en haut à droite, choisir system settings, choisir backup et activer cette fonction en précisant les paramètres). Une alternative est de s'envoyer la dernière version du code source par email.

Gestion du code au sein d'un groupe :

- vous pouvez envisager d'utiliser **gdrive.epfl.ch** pour définir un répertoire partagé par les 2 membres du groupe et pas plus. Il n'y a pas d'éditeur de code en mode partagé.
- L'option c4science n'est recommandée que pour ceux qui maitrisent déjà git. Si vous l'utilisez, vous devez supprimer l'accès public (par défaut) à votre code.