

Quokk'adventure

RAPPORT DE PROJET

Table des matières

| | |
|---|----|
| Introduction..... | 2 |
| Synopsis..... | 2 |
| Mise en œuvre du modèle | 3 |
| Modèle Commande..... | 3 |
| Dans Quokk'adventure..... | 4 |
| Retour en arrière | 4 |
| Liste des actions | 4 |
| Replay du niveau | 5 |
| Diagramme de séquence pour l'exécution d'une commande | 5 |
| Mise en place de l'environnement de développement | 7 |
| Prérequis | 7 |
| Documentation d'utilisation..... | 8 |
| Démarrage..... | 8 |
| Accueil | 9 |
| Fonctionnalités communes | 9 |
| Bouton « mute »..... | 9 |
| Gameplay..... | 10 |
| But du jeu | 10 |
| Mouvement du Quokka..... | 11 |
| Déplacement de caisses | 11 |
| Manger une pomme..... | 12 |
| Annulation du dernier mouvement..... | 12 |
| Historique des coups | 13 |
| Affichage du temps et du nombre de coups | 14 |
| Niveau réussi | 14 |
| Niveau final..... | 15 |
| Bugs | 16 |
| Conclusion | 16 |
| Annexes | 16 |

Introduction

Ce projet a été réalisé dans le cadre du projet de fin de semestre du cours "modèles de conception réutilisables à la HEIG-VD durant le semestre de printemps 2021.

Le cours est encadré par:

- (professeur) Rosat Sébastien
- (assistant) Decorvet Grégoire
- (assistant) Sousa e Silva Fábio

Chaque groupe s'est vu attribuer un design pattern pour réaliser une application autour de ce dernier. Nous avons reçu le command design pattern.

Ce design pattern permet de créer un objet intermédiaire entre l'invocateur et le récepteur de la commande. Cet objet est très pratique pour conserver un historique et pour annuler des commandes.

Nous avons donc choisi d'illustrer ce modèle de conception au travers d'un sokoban.

Pour rappel voici la description de ce type de jeu :

« <...> le joueur doit ranger des caisses sur des cases cibles. Il peut se déplacer dans les quatre directions, et pousser (mais pas tirer) une seule caisse à la fois. Une fois toutes les caisses rangées (c'est parfois un vrai casse-tête), le niveau est réussi et le joueur passe au niveau suivant, plus difficile en général. L'idéal est de réussir avec le moins de coups possible (déplacements et poussées). » [Extrait de Wikipedia](#)

Nous visualisons aisément l'utilité du design pattern dans ce genre de jeu. Dans de telles applications, il n'est pas rare d'avoir un historique de ses coups, de pouvoir rejouer un niveau ou même annuler une action. Ces trois choses peuvent sembler complexes à mettre en place. Cependant grâce au modèle, cela devient presque trivial.

Synopsis

Vous incarnez un jeune quokka qui doit ranger des graines dans ses caisses. Malheureusement, vous avez eu les yeux plus gros que le ventre et vous avez réuni trop de nourriture. Après tout, ce n'est pas une mauvaise chose, le climat est parfois rude, il est donc dur de trouver à manger quand vous le souhaitez. Qu'à cela ne tienne, vous gardez tout. Après tout, les graines, ça se conserve. Malheureusement, déplacer l'entièreté de vos provisions vous occuperait pour des semaines.

Bien décidé à ne pas abandonner votre précieux butin, vous réfléchissez à une idée pour le protéger... Soudain, une idée vous vint en tête : « Puisque je ne peux pas déplacer mes graines vers mes boîtes, ce sont les boîtes qui iront aux graines ». Fière de votre nouveau plan d'action, vous attellez à la tâche sans perdre une seconde de plus. Toutefois, faites bien attention à vos décisions, vous risquez de vous retrouver bloquer.

Mise en œuvre du modèle

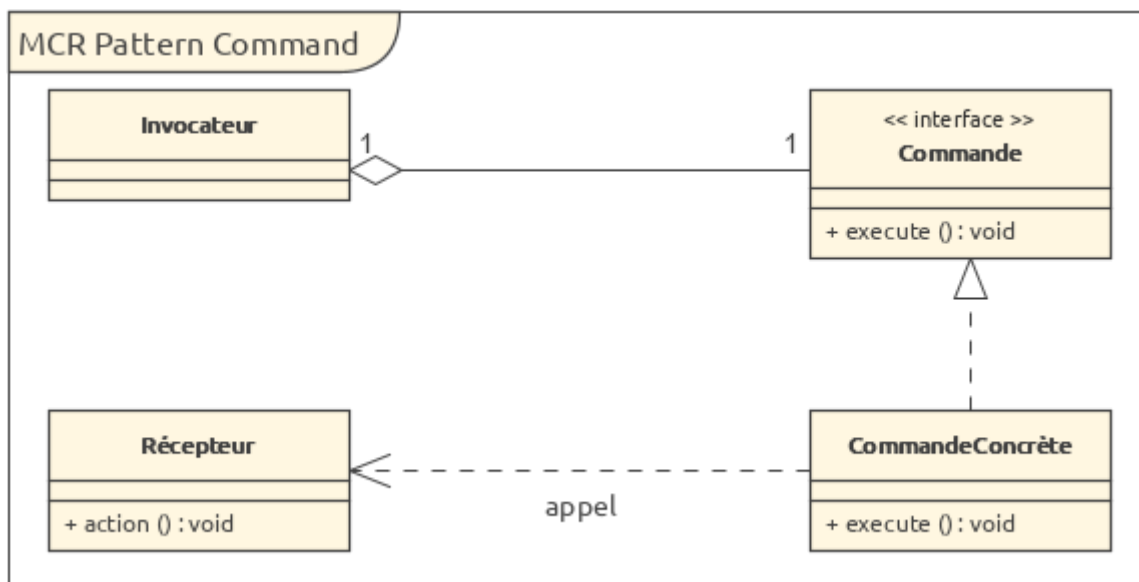
Modèle Commande

Le modèle commande est un modèle comportemental permettant de séparer le code initiateur et l'action. Ce modèle permet une délégation de la responsabilité : une classe va déléguer une requête à un objet *commande* à la place de l'implémenter directement.

Ses objectifs sont de pouvoir envoyer des requêtes sans connaître leurs caractéristiques ainsi que d'encapsuler lesdites requêtes. Il permet aussi d'éviter de dupliquer du code, de rendre les commandes sécurisées ainsi que de rendre le code plus facile à maintenir.

Les utilisations les plus connues de ce modèle sont : Gérer des actions venant d'une interface graphique ou d'entrées utilisateur, sauvegarder un état pour pouvoir y revenir, stocker les commandes pour pouvoir les réutiliser plus tard et créer des logs pour pouvoir réagir en cas de crash.

UML exemple du pattern :



Dans Quokk'adventure

Dans notre projet, nous avons 3 manières visibles d'utiliser le modèle Commande : le retour en arrière (1.), la liste des actions (2.) et le replay du niveau (3.).

Ces 3 utilisations reposent sur le fait que les commandes générées par l'utilisateur seront stockées dans une stack qui nous sert d'historique de commandes et vont pouvoir être réutilisées plus tard.

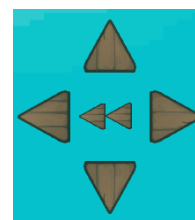


De plus, toutes les actions de l'utilisateur sont gérées par des commandes. Cela nous a permis, sans problème, d'avoir plusieurs entrées utilisateur comme les flèches cliquables sur l'interface graphique et les flèches du clavier.

Retour en arrière

Le bouton de retour en arrière va permettre à l'utilisateur d'annuler sa dernière action.

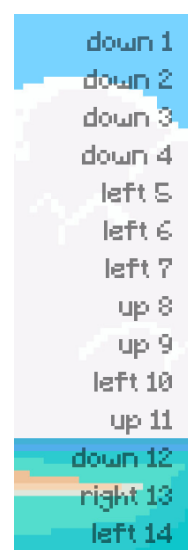
Pour ce faire nous allons simplement aller chercher la dernière commande entrée par l'utilisateur, la dépiler de l'historique et utiliser sa méthode undo.



Liste des actions

La liste des actions de l'utilisateur est affichée sur la droite de l'écran. Cette liste permet tout d'abord de visualiser clairement les commandes générées par l'utilisateur à chaque déplacement. De plus cette liste va permettre à l'utilisateur, en cliquant sur une des actions, de revenir en arrière, avant d'avoir exécuté cette commande.

Pour implémenter cette utilisation du modèle, nous allons tout d'abord afficher chaque commande, sous forme de bouton dans un scroller sur la droite de l'écran, lorsque celle-ci est générée. Quand l'utilisateur clique sur un des boutons, nous le dépilons et annulons toutes les commandes jusqu'à traiter la commande représentée par ce bouton.



Replay du niveau

Quand l'utilisateur finit un niveau, il a l'occasion de pouvoir revoir sa performance. En effet, en cliquant sur le bouton « Review » le niveau va se relancer et se rejouer automatiquement sans que l'utilisateur ne puisse interagir avec le déroulement de l'action.

Pour ce faire, nous créons un nouveau niveau qui va récupérer l'historique du niveau actuel et le parcourir à l'envers en exécutant chaque commande relativement à un timer. Procéder de cette manière va exécuter les commandes comme si l'utilisateur les avait générées sur le moment, mais en utilisant les commandes stockées dans l'historique.



Diagramme de séquence pour l'exécution d'une commande

Le point d'entrée peut être atteint de trois façons

- En appuyant sur les touches de claviers [W,A,S,D].
- En appuyant sur les flèches du clavier
- En cliquant sur les flèches affichées à l'écran

GameScreen

Scène principale du jeu

Tableau

Tableau d'acteurs qui vont être influencés par les commandes.

MoveCommand

Une commande qui va déplacer un acteur.

PushCommand

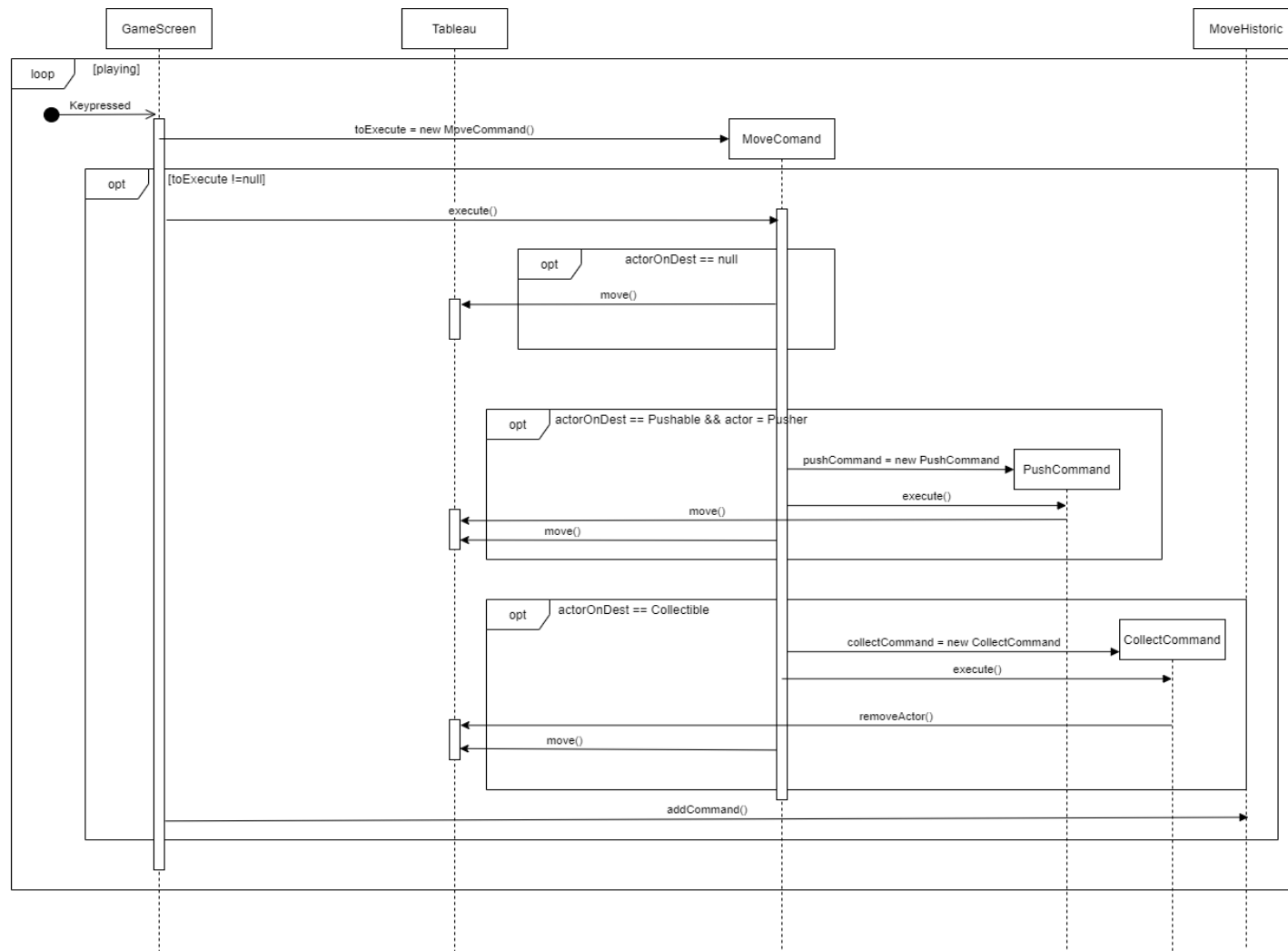
Une commande qui déplace un acteur uniquement s'il a été poussé par un autre acteur.

CollectCommand

Une commande qui va supprimer un acteur du tableau, et appliquer un effet spécial à l'acteur qui est passé dessus.

MoveHistoric

Permet de gérer/stocker un historique de commande pour pouvoir annuler une commande ou rejouer le niveau.



Lorsque nous annulons une commande, nous appelons la méthode *undo* de MoveHistoric qui se charge de dépiler la dernière commande et de l'annuler, s'il y en a une.

Mise en place de l'environnement de développement

Prérequis

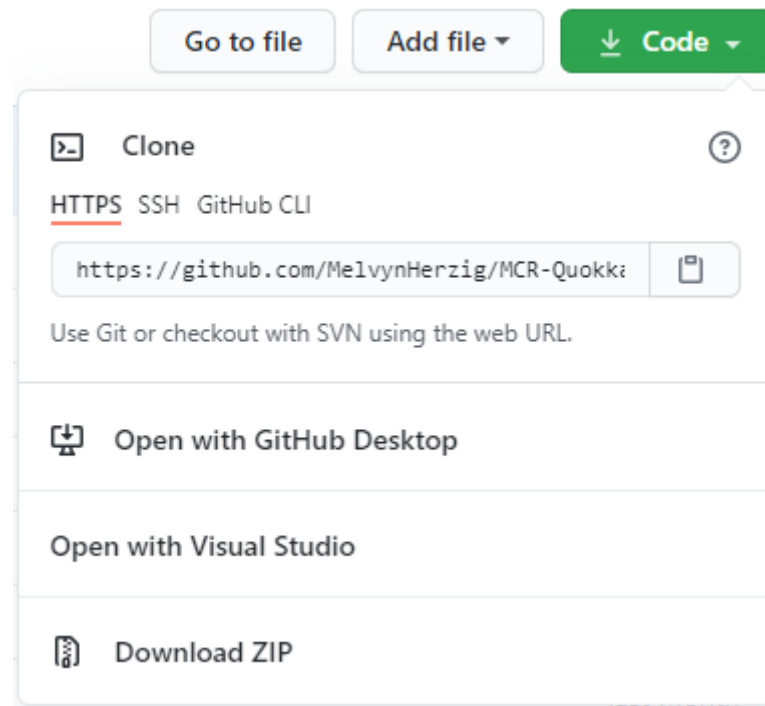
Pour la mise en place de l'environnement de développement, ayez installé préalablement :

- Java (JDK) en version 11.0.8
- [IntelliJ](#) ultimate 2021.1.2

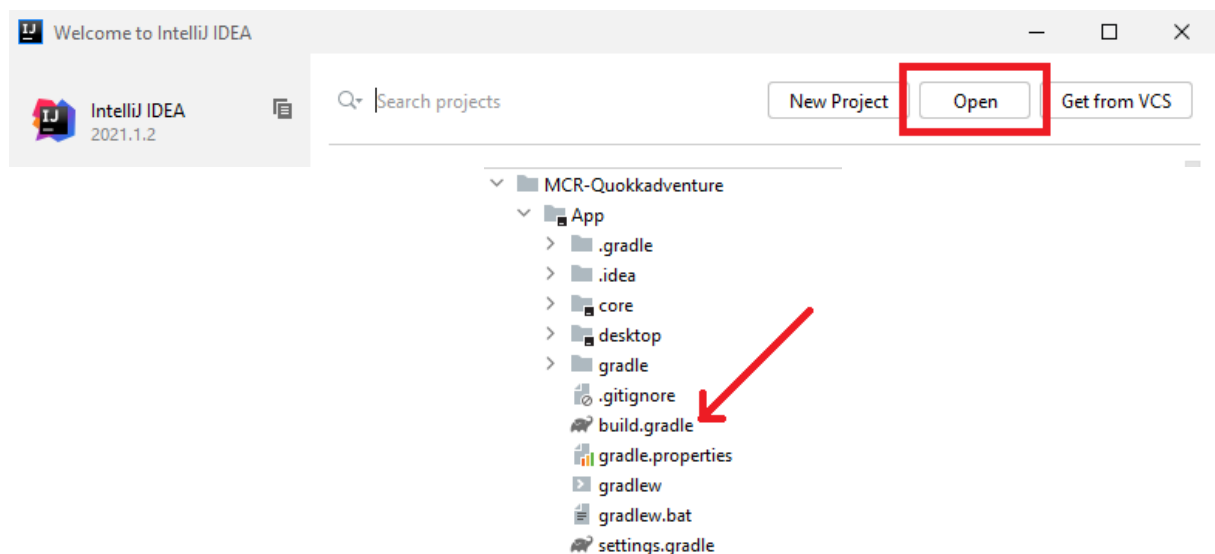
Le projet a été développé avec [libgdx](#) en version 1.10

Pour continuer le développement de Quokk'adventure :

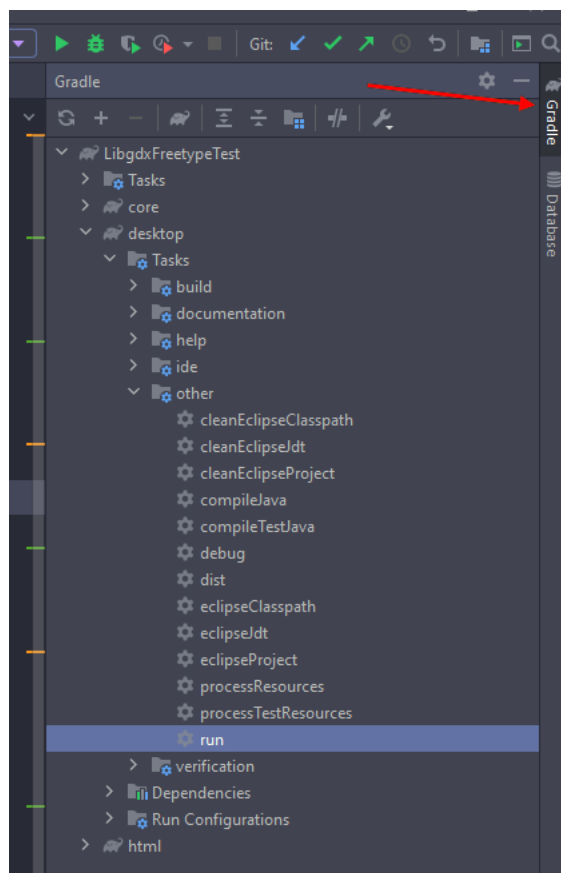
1. Rendez-vous sur le [github](#) du projet
2. Clonez le répertoire



3. Ouvrez IntelliJ
4. Sélectionnez « Open » et importez le fichier « App/build.gradle »



Au terme de ces 4 étapes, vous avez importé le projet dans IntelliJ. Si vous n'êtes pas familier avec Gradle, vous pouvez lancer le build et l'exécution du projet de la façon suivante :



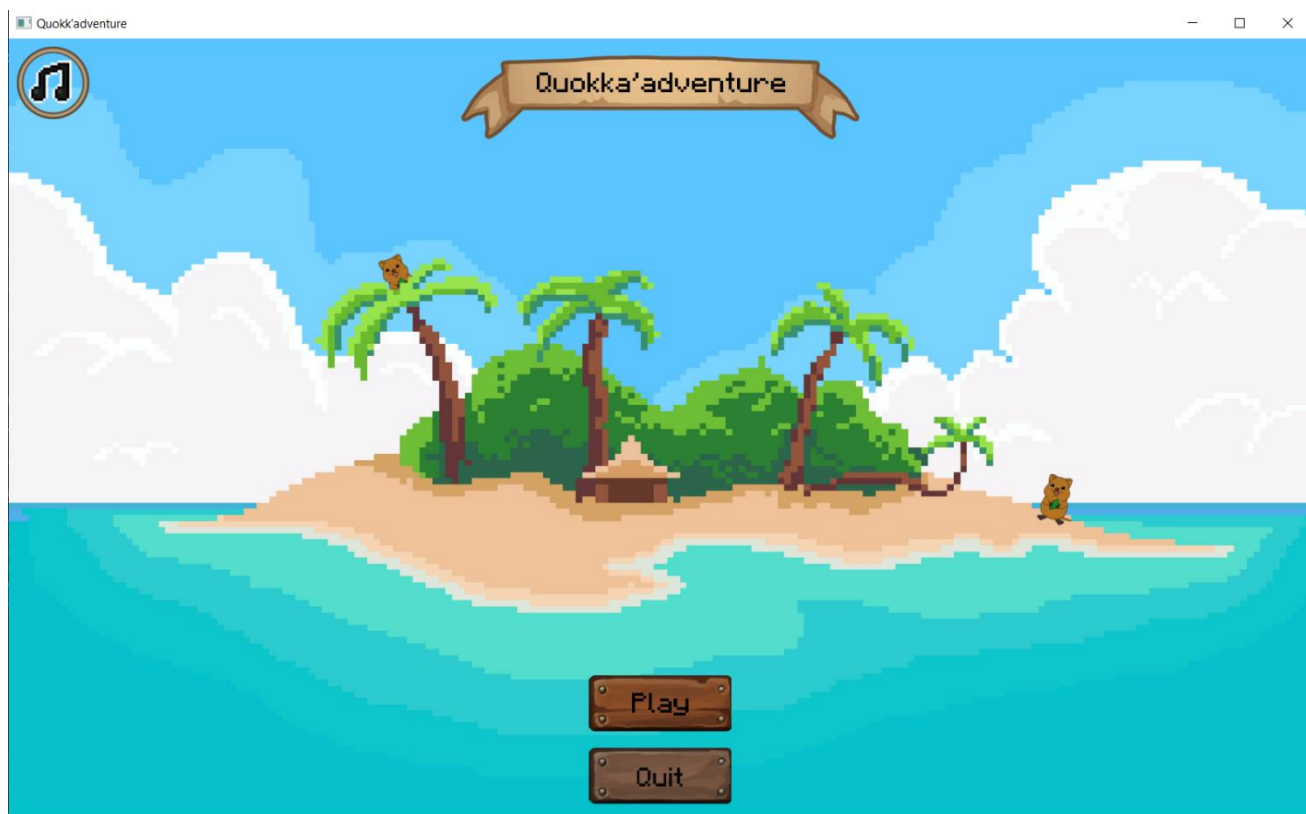
Documentation d'utilisation

Démarrage

1. Avoir une JVM java (version 1.8 minimum).
2. Télécharger le fichier « quokkadventure.jar » dans l'onglet release de notre github (<https://github.com/MelvynHerzig/MCR-Quokkadventure/releases>).
3. Lancer l'application :
 - a. **Windows** : double cliquer sur l'application pour la lancer
 - b. **Linux** : Appeler Java en lui passant le fichier .jar en paramètre :
`java -jar <PATH_TO_JAR>`
 - c. **Mac** : double cliquer sur l'application pour la lancer

Accueil

Après avoir lancé l'application, la page d'accueil suivante s'affiche :



Vous pouvez quitter l'application en cliquant sur « Quit ».

Vous pouvez également cliquer sur « Play » afin de commencer à jouer. Jouer revient à commencer le premier niveau. Pour apprendre à jouer, consulter la section Gameplay.

Fonctionnalités communes

Bouton « mute »

Ce bouton est présent dans toutes les pages / interfaces du jeu afin de pouvoir désactiver ou activer la musique.

Si la croix rouge est affichée sur le bouton, c'est que la musique est inactive.

Sinon la musique est active.




Gameplay


Voici à quoi ressemble un niveau :

But du jeu

Le but du jeu est de bouger le


quokka  dans toutes les directions voulues afin de


déplacer les caisses  sur

les points d'arrivée . Il faut donc éviter de bloquer les

caisses contre les murs .

Il peut également y avoir des caisses plus lourdes à déplacer

. Pour réussir à les déplacer, il faut d'abord

manger une pomme  afin que le quokka gagne de la force supplémentaire.



Mouvement du Quokka

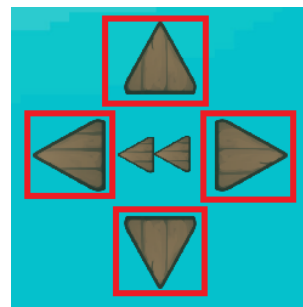
Pour déplacer le quokka, vous avez 2 possibilités.

La première, utiliser les touches « flèches » du clavier et « WASD ».

La seconde option est d'utiliser les boutons « flèches » sur l'interface graphique.



© CanStockPhoto.com



Voici les mouvements réalisés par le quokka en fonction de la flèche utilisée :

- Flèche haut : avance d'une case vers le haut
- Flèche bas : avance d'une case vers le bas
- Flèche droite : avance d'une case vers la droite
- Flèche gauche : avance d'une case vers la gauche

Déplacement de caisses

Déplacement de caisses normales

Voici une caisse normale :



Pour la déplacer, il faut se mettre à côté de la caisse et ensuite se déplacer dans la direction de la caisse. L'exemple ici montre comment déplacer la caisse d'une case sur la gauche comme l'indique la flèche rouge :



Il est évidemment impossible de déplacer une caisse si cette dernière est collée à un mur et que l'on essaie de la déplacer dans la direction de ce dernier. Ici la flèche rouge montre un mouvement irréalisable.



Déplacement de caisses lourdes

Voici une caisse lourde :



Le déplacement d'une caisse lourde se réalise de la même façon qu'une caisse normale. Il faut juste avoir mangé une pomme au préalable. Vous pouvez consulter comment manger une pomme [ici](#).



Tout comme les caisses normales, il est impossible de déplacer une caisse contre un mur.



Manger une pomme

Voici une pomme :



Si le quokka mange une pomme, il gagne de la force. Cette force supplémentaire permet de déplacer des caisses lourdes.

Pour manger une pomme, il suffit de se déplacer sur la pomme comme l'indique la flèche [bleue](#) sur l'exemple ci-dessous.



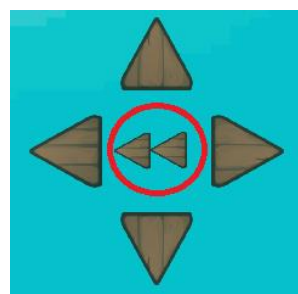
Annulation du dernier mouvement

Dans notre jeu, il est possible d'annuler votre dernier mouvement, ce qui est très pratique pour ne pas recommencer le niveau à chaque fois.

Pour ceci, il vous suffit de cliquer sur le bouton entouré en [rouge](#) comme montré sur l'image à droite ou sur la touche « u » du clavier.

Pour utiliser cette touche, il faut avoir au moins joué un coup.

Si vous appuyez plusieurs d'affilés sur le bouton « undo », il annulera le même nombre de mouvements que le nombre de pressions sur ce dernier.



Historique des coups

Un historique de tous les coups joués est affiché sur la droite de la fenêtre du jeu.

Sur la droite, vous pouvez observer un exemple d'historique de coup d'une partie en cours.

L'historique peut atteindre une taille assez conséquente suivant le nombre de coups jouer. C'est pour cela qu'il est possible de le faire défiler à l'aide de la souris, en maintenant enfoncé le clic gauche sur l'historique et en le déplaçant de bas en haut, telle une scroll-bar. Il est ainsi possible d'afficher des coups plus anciens et donc de consulter tout l'historique. Le scroll est uniquement disponible une fois que la liste de coups dépasse la taille de la fenêtre en hauteur.

L'historique peut également être utilisé afin de revenir à un coup désiré. En effet, il est possible de revenir à un coup spécifique de l'historique en réalisant un clic gauche sur le coup désiré.



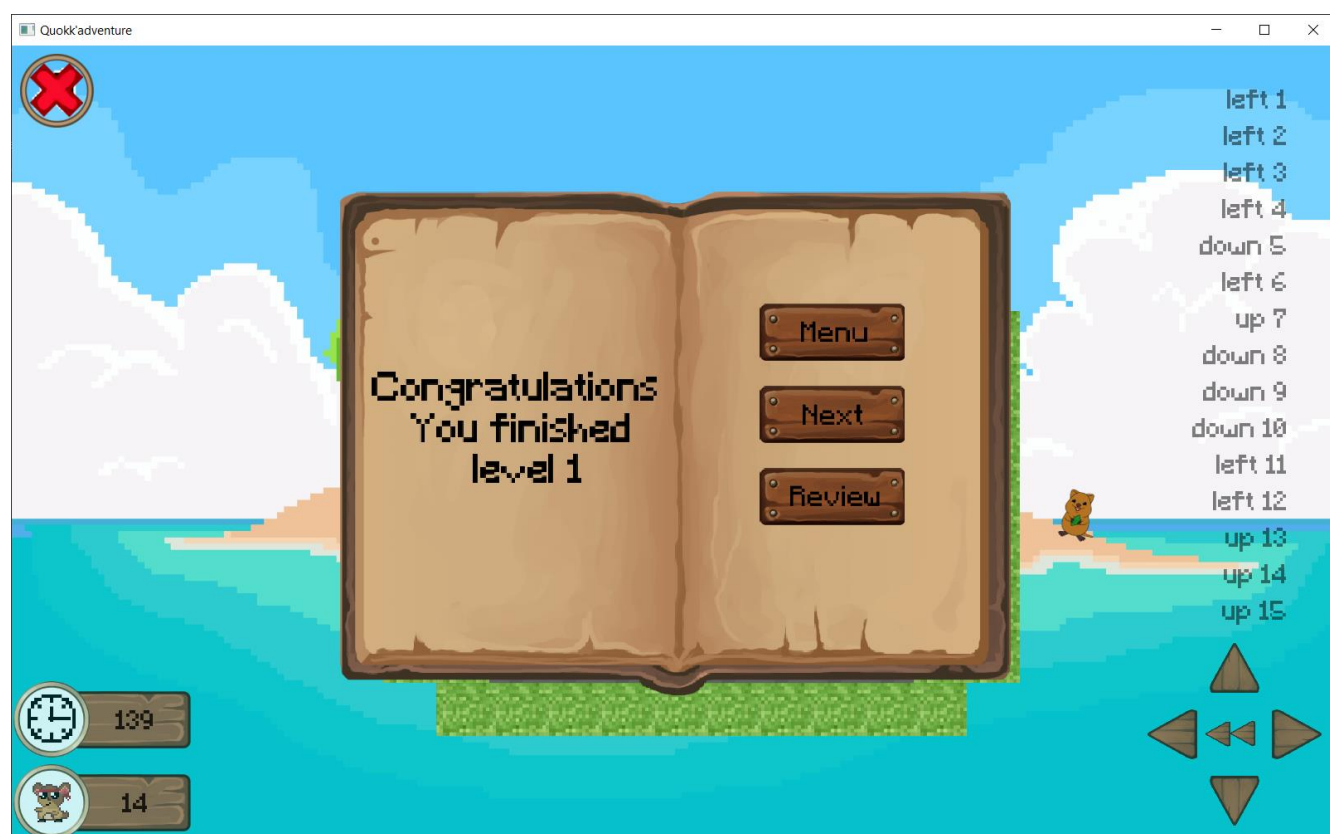
Affichage du temps et du nombre de coups

Il est possible de consulter en tout temps depuis combien de temps vous avez commencé le niveau. Ici indiqué à l'aide du rectangle orange.

Il est également possible de consulter le nombre de coups réalisés depuis le début du niveau. Ici indiqué avec le rectangle violet.

Niveau réussi

Lorsque vous terminez un niveau, c'est-à-dire, quand vous avez réussi à déplacer toutes les caisses, la fenêtre suivante va apparaître :



Vous avez donc 3 possibilités :

- Menu : pour retourner au menu principal
- Next : pour passer au niveau suivant
- Review pour rejouer tous les coups du niveau réussi

Niveau final

Lorsque vous aurez terminé le dernier niveau, le bouton « Next » ne sera plus disponible, car aucun niveau supplémentaire n'a encore été créé.

Notre jeu comporte actuellement 10 niveaux avec une difficulté plus ou moins progressive et un dernier niveau très compliqué.

Vous aurez donc fini notre petit jeu !

Voici l'écran d'affichage final obtenu :



Bugs

- Non-persistence de la musique coupée entre les niveaux
Si nous coupons la musique dans le niveau 1 et que nous passons au niveau 2, elle se réactive.
- Les graines s'affichent parfois au-dessus du Quokka et/ou des caisses
- Le contrôleur de la vitesse pour rejouer les niveaux est toujours visible dans l'écran de fin du niveau après avoir l'avoir rejoué au moins une fois.

Conclusion

Au terme de ce projet, nous avons réussi à démontrer l'utilité du patron de conception *commande* au travers de Quokk'adventure.

Nous avons implémenté l'ensemble des fonctionnalités que nous avions planifiées.

Le projet est fonctionnel, mais souffre de quelques bugs mineurs.

Comme précisé dans le répertoire GitHub, ce projet utilise du contenu non libre de droits. En conséquence, il doit être développé et utilisé uniquement à but éducatif.

Annexes

- Code sources au format .java
(MCR-Quokkadventure\AppData\src et MCR-Quokkadventure\AppData\desktop\src)
- Diagramme de classes
- Documentation Javadoc.