ENSSAT LANNION

IMR1 - PROJET POO2 - 2014/2015
BOULDER DASH

1 Présentation

L'objectif de ce projet est de développer un clone du jeu **Boulder dash**. Tout le monde connait certainement ce jeu, ou en tout cas certaines versions de ce jeu. Pour plus d'informations, vous pouvez consulter la page wikipedia dédiée (cf. Wikipedia, `http://fr.wikipedia.org/wiki/Boulder_Dash`) et surtout le site suivant qui donne beaucoup d'informations sur le déroulement du jeu et les niveaux proposés : `http://www.boulder-dash.nl/index.html`.

Afin de présenter les différents éléments nécessaires, une capture d'écran est présentée à la figure 1. Sur cette dernière, on peut observer les différents éléments du jeu :

- Boulder* : peut tomber, rouler, faire exploser d'autres éléments ;
- Diamond* : peut tomber, rouler, ce qui doit être ramassé par le personnage ;
- Magic wall : mur qui peut changer d'état et soit faire disparaître des rochers qui le traverse, soit les changer en diamant ;
- Brick wall : simplement un mur, mais il peut exploser ;
- Steel wall : un mur indestructible ;
- Expanding wall$^{\dagger *}$: un mur qui a le même aspect que le précédent, qui a la particularité de se multiplier si il y a de la place libre à gauche ou à droite ;
- Rockford† : le personnage principal
- Dirt : ce qui permet à tous les éléments de tenir en place, Rockford peut se déplacer à travers et ainsi la faire disparaître ;
- Firefly$^{\dagger *}$, Butterfly$^{\dagger *}$, Amoeba$^{\dagger *}$: éléments qui peuvent se déplacer et faire exploser Rockford notamment, ces éléments ne sont pas demandés dans le projet mais peuvent être insérés en plus.

Les éléments marqués par † sont animés. Une spécification très complète des états et actions des différents éléments est présentée à l'adresse : `http://www.elmerproductions.com/sp/peterb/BDCFF/objects/index.html`. Vous pouvez vous en inspirer, en faisant l'effort d'adapter ce qui est proposé au contexte du langage Java ainsi qu'aux contraintes sur l'utilisation de MVC.

Le travail à réaliser ce décompose en deux parties complémentaires : le jeu lui-même et un éditeur de niveaux. Pour le jeu, il est attendu d'avoir l'affichage du plateau et de tous les éléments qu'il inclut, l'interaction avec le jeu (au clavier) pour déplacer le personnage, l'animation des éléments insérés dont le personnage lors de ces déplacements, les diamants et les rochers (en cas de chute ou lorsque le personnage pousse un rocher). Afin de faciliter le travail, le découpage suivant du projet est proposé :

$*$. Elément dont l'implémentation n'est pas demandé.

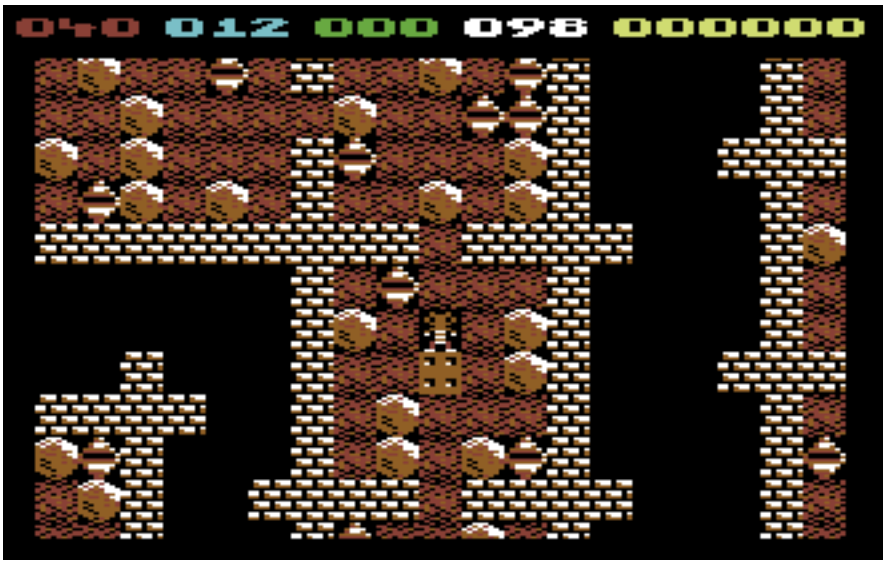

FIGURE 1 – Capture d'écran du jeu *Boulder Dash*.

1. construction du modèle,
2. construction des interfaces pour le jeu et l'éditeur de niveau,
3. animation des éléments,
4. détection de collision,
5. affichage des éléments d'information.

2 Construction du modèle

La première étape dans la construction du jeu est la définition du modèle du jeu. L'analyse du jeu met en évidence, au minimum, des classes qui correspondent au différents éléments affichables. De plus, une classe représentant un niveau est nécessaire et doit permettre sa construction ainsi que sa sauvegarde/chargement dans/depuis un fichier.

En pratique, les éléments affichables sont des sprites et certains d'entre-eux peuvent être animés. Pour conserver toutes les informations du modèle, une classe Game apparaît également nécessaire et permet en outre de conserver d'autres informations telles que le score.

3 Interfaces graphiques

3.1 Interface de jeu

L'interface du jeu doit comporter deux parties. La première est l'affichage du plateau de jeu et la deuxième est le panneau d'information indiquant l'état courant du jeu. Cette organisation est représentée de manière schématique sur la figure 2. Le choix de l'organisation de l'aire d'information est laissé à votre appréciation en fonction des informations que vous aurez prises en compte dans votre modèle.

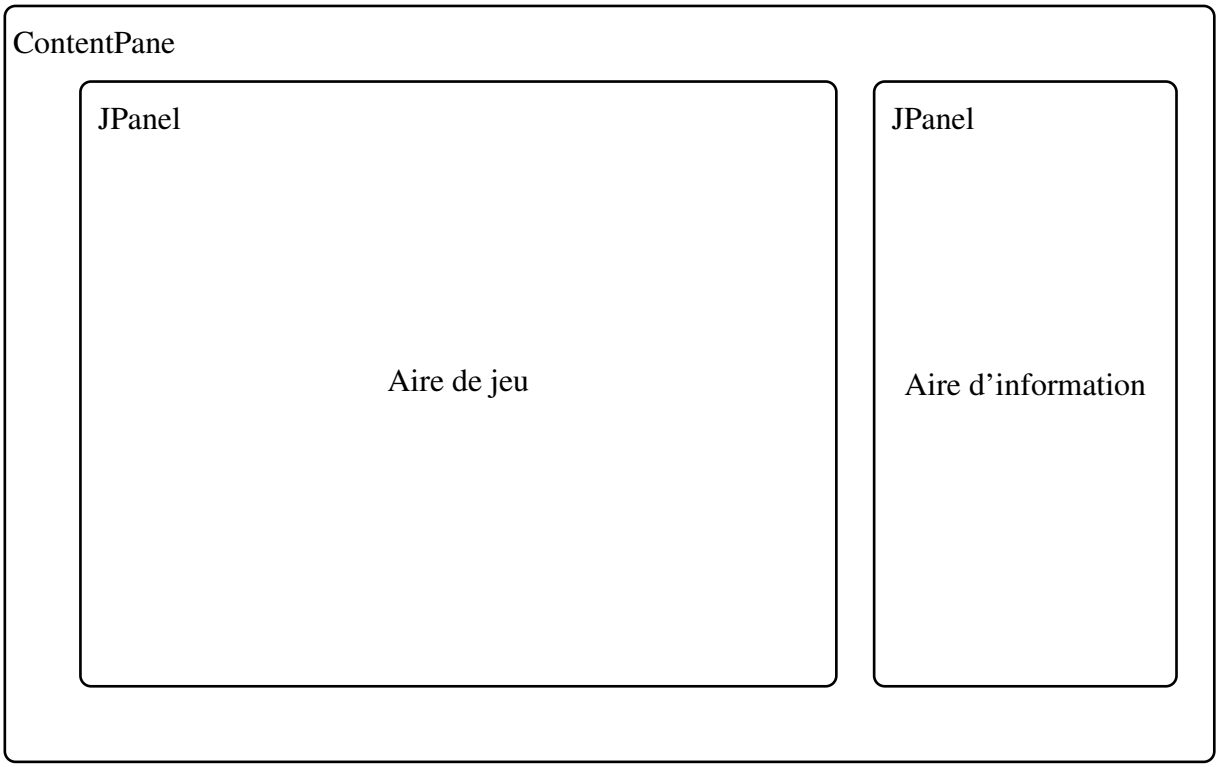

FIGURE 2 – Architecture de l'interface graphique principale

Également, l'ajout d'un menu est nécessaire afin de pouvoir effectuer les actions suivantes : *Nouvelle partie*, *Pause*, *Quitter* et obtenir de l'aide sur l'utilisation du jeu. Ces différents éléments seront à compléter au fur et à mesure du développement du jeu.

La mise en place de l'aire de jeu nécessite l'utilisation du modèle afin de le dessiner. Pour cela, vous pourrez utiliser par défaut les sprites fournis (voir répertoire *fourniture*).

À l'issue de cette partie, vous devez avoir une interface permettant d'afficher l'interface principale avec son menu, ses deux zones d'affichage ainsi que les éléments du jeu de manière statique.

3.2 Interface de l'éditeur

L'architecture de l'interface de l'éditeur de niveau est laissé à votre appréciation (elle peut être tout à fait similaire à celle du jeu). Néanmoins, une partie indispensable doit permettre de sélectionner les éléments de jeu à ajouter qui pourront ensuite être déposés sur l'aire de jeu. Comme précédemment, un menu est nécessaire notamment pour permettre le chargement et la sauvegarde du niveau édité.

4 Animation des éléments et interaction avec l'utilisateur

L'animation des éléments du jeu doit mettre en oeuvre la notion de boucle de jeu afin de rafraîchir régulièrement l'affichage. Pour cela, je vous invite à consulter le tutoriel présent à l'adresse suivante : `http://zetcode.com/tutorials/javagamestutorial/animation/`. En appliquant ce tutoriel, vous devez mettre en place le déplacement autonome du personnage. Notez que cela se traduit par l'écriture d'une boucle de jeu, exécutée à intervalle régulier qui va demander la mise à jour des entités présentes dans le jeu et ensuite rafraîchir l'affichage. L'étape suivante consiste à faire le lien entre le personnage et les actions de l'utilisateur. Il s'agit donc de mettre en oeuvre un *KeyListener* permettant d'interpréter les appuis sur les flèches gauche/droite/bas/haut afin d'agir sur le personnage. L'animation d'un élément consiste simplement à changer l'image courante de l'élément à intervalles réguliers et à déplacer l'image sur le plateau (si déplacement de l'élément).

Une autre partie de l'interaction consiste, pour l'éditeur, à sélectionner un élément de jeu et de le placer sur le plateau de jeu.

5 Détection des collisions

La détection des collisions est un élément important du jeu qui va permettre de prendre en compte qu'un rocher qui tombe va s'arrêter en rencontrant un mur, que le personnage ne va pas traverser les murs, etc. Une méthode est d'utiliser la classe *java.awt.Rectangle* et sa méthode *intersect* afin de vérifier si l'intersection entre deux rectangles est non vide.

6 Affichage des éléments d'information

Au cours du jeu, certains éléments du modèle évoluent et nous souhaitons permettre à l'utilisateur de consulter ces informations. Par exemple, le score, le nombre d'aliens encore présents, le nombre de vies restantes (si cela est géré) peuvent être affichés dans le panneau d'information de l'interface principale.

La mise en oeuvre de cet affichage correspond à une vue du modèle représenté par le jeu. Il devient alors naturel d'utiliser le pattern MVC pour obtenir une synchronisation entre la vue et le

modèle et faire évoluer les informations au cours de la partie. Ce principe devra être appliqué autant que possible en considérant que le jeu est le modèle (dont le plateau, le personnage, le score, etc.), que l'interface est l'aire de jeu ou bien l'aire d'informations et enfin que le contrôleur se contente de gérer les actions de l'utilisateur afin d'agir sur le modèle.

7 Travail à réaliser

Vous devez réaliser l'application décrite précédemment en binôme. La note du projet sera fonction, bien entendu, de la qualité de la modélisation employée, de la qualité d'écriture du code source, mais également du niveau de finalisation du jeu et de l'éditeur. Notez que la note tiendra compte du niveau de réalisation mais que l'utilisation de MVC est une contrainte forte.

Vous devrez rendre une archive contenant les éléments suivants :
— le code source de l'application ;
— un fichier jar exécutable ;
— un rapport d'une dizaine de pages décrivant votre réalisation **au format PDF** ;
— une vidéo de démonstration du fonctionnement de votre application (screencast).

L'archive devra être nommée de la manière suivante : *NOM1_NOM2.zip*. N'utilisez pas d'accent dans le nom de l'archive.

Le rapport doit bien entendu être structuré (introduction, ...) et comporter un jeu d'essai de votre application. L'objectif de ce rapport n'est pas d'écrire un catalogue des différents composants utilisés mais de présenter votre application, son fonctionnement ainsi que les difficultés rencontrées et le niveau de réalisation final atteint. Il devra également faire ressortir la contribution de chacun.