TelePac

**Lien vers la page web: http://www.esiee.fr/~legraing/index.html**

Groupe 3 J

Guillaume LEGRAIN

Florian MARTIN

Sommaire :

I.A) Auteurs

I.B) Thème

I.C) Résumé du scénario

I.D) Plan

I.E) Scénario détaillé

I.F) Détail des lieux, items, personnages

I.G) Situations gagnantes et perdantes

I.H) Eventuellement énigmes, mini-jeux, combats, etc.

I.I) Commentaires (ce qui manque, reste à faire, ...)

II. Réponses aux exercices (autres que I.)

III. Mode d'emploi (si nécessaire, instructions d'installation ou pour démarrer le jeu)

IV. Déclaration obligatoire anti-plagiat (préciser toutes les parties de code que vous n'avez pas écrites vous-même et citez la source, sauf les fichiers zuul-\*.jar qui sont fournis évidemment)  
V, VI, etc... : tout ce que vous voulez en plu

I.A) Auteurs

LEGRAIN Guillaume

MARTIN Florian

I.B) Thème

TelePac

Who would have ever thought PAC-Man could teleport himself to escape from the ghosts.

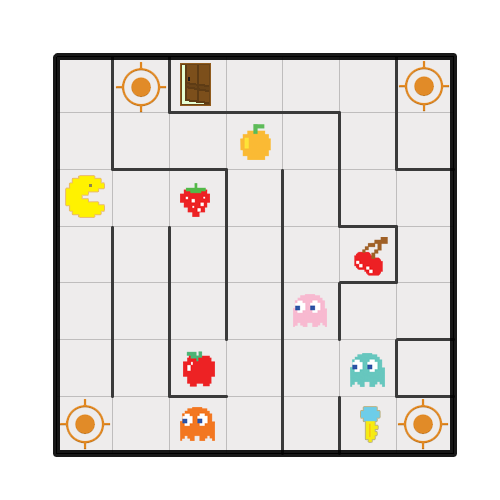
Choose wisely, the transporters can be faulty.

Feeling tired, Eat some fruit to gain strength.

I.C) Résumé du scénario

Pac-man est perdu dans un labyrinthe remplit de fantôme. Il lui faut manger les fruits disséminés dans les salles pour vaincre les esprits et s’emparer de la clé ouvrant la sortie. Pour se déplacer plus rapidement il aura accès à des télé porteurs. Mais certains peuvent l’envoyer dans des lieux aléatoire, il lui faudra donc être prudent.

I.D) Plan



I.E) Scénario détaillé

Pac-man commence l’aventure en haut à gauche du labyrinthe. Celui-ci se dévoilera au fur et à mesure de sa progression. Il lui faut trouver les quatre fruits éparpillés dans les salles pour gagner en force et ainsi pouvoir dévorer les fantômes lui bloquant le passage. Il lui faudra dans un même temps retrouver la clé elle aussi caché dans le labyrinthe. Celle-ci ouvre la porte de sortie gardée par un fantôme nécessitant d’avoir trouvé l’intégralité des fruits pour être vaincu.

I.F) Détail des lieux, items, personnages

Lieux : Un labyrinthe de 7\*7 cases possédant un lieu de départ, une porte de sortie et quatre cases de téléportation

Items : une pomme, une orange, une cerise, une fraise et une clé

Personnages : Pac-man (joueur), quatre fantômes

I.G) Situations gagnantes et perdantes

Situation gagnante :

Le joueur atteint la porte de sortie avec en sa possession les quatre fruits et la clé

Situations perdantes :

Le joueur se retrouve face à un fantôme alors qu’il n’a pas suffisamment de fruits

Le joueur n’a plus de mouvement disponibles

II. Réponses aux exercices

Exercice 7.0 : Création de la page web

Exercice 7.1 : Découverte de zuul-bad

I La classe Room

Création de la première salle avec pour attributs la description de la salle et quatre sorties (Nord, Sud, Est, West)

II La classe Command

Création de la commande « go ». On doit déterminer dans un premier si un ou deux mots ont été tapés. Ensuite on doit vérifier si les commandes tapés sont valides d’où l’introduction d’une fonction booléenne « isUnknown »

III La classe Game

-Création d’une méthode « createRooms » ayant pour but d’initialiser le jeux (les salles, les sorties, le lieu courant)

-La procédure « goRoom » doit s’assurer de la présence d’un second mot (direction) et doit vérifier l’égalité de ce second mot avec une direction avec la fonction « equals ». Si les conditions sont réunis « goRoom » change la current room

-On ajoute de nouvelles méthodes : 2 procédures renvoyant un message de bienvenue et un message d’aide. La première doit pouvoir faire appel à la current room et à ses différentes sorties. La seconde doit pouvoir renvoyer le nom de chaque commande disponible. On crée donc des procédure auxquelles on fera appel pour renvoyer ces informations et cela indépendamment du message écrit.

-On crée une procédure « processCommand » renvoyant un booléen, capable d’appeler la bonne méthode en fonction de la commande passée en paramètre

IV Les classes CommandWords et Parser

La classe CommandWords contient les mots de commande acceptés par le jeu

La classe Parser lie les commandes tapée au clavier, vérifie si la commande est valide et construit l’objet « Command » correspondant auquel on fera appel

V Jeu fonctionnel

On ajoute dans la classe Game un attribut aParser et une procédure « play» qui doit lire répétitivement les commandes tapé au clavier et les exécuter jusqu’à ce que l’on tape « quit ». On introduit une boucle while qui s’exécutera jusqu’à ce qu’une variable booléenne soit égal à vrai. Ce changement d’état aura lieu lorsque l’on tapera la commande « quit ».

Exercice 7.1.1 : Thème

Exercice 7.2.1 : La classe Scanner

On crée un objet Scanner possédant le clavier en tant que paramètre. Les lignes de caractères tapés au clavier seront retranscrites dans une String.

Exercice 7.4 : Room

On intègre dans le jeu les différentes salles qui compose notre scénario

Exercice 7.5 : printLocationInfo

On ajoute une méthode affichant les informations de la current Room (nom pour l’instant)

Exercice 7.6 : getExit

On ajoute une méthode renvoyant les sorties de la current Room

Exercice 7.7 : getExitString

On modifie la méthode « printLocationInfo » pour qu’elle affiche aussi les sorties de la current Room. On fait appel pour cela à la méthode «  getExit »

Exercice 7.8 : HashMap, setExit

On modifie la classe Room pour qu’elle possède un attribut aExit sous la forme d’une HashMap regroupant ainsi toutes les sorties.

Exercice 7.9 : keySet

Exercice 7.10 : getExitString CCM ?

ToDo != TODO

Exercice 7.11 : getLongDescription

ToDo

Exercice 7.14 : look

On ajoute une commande « look » faisant appel à la procédure  « getLongDesciption » pour que le joueur puisse obtenir les informations de la current Room

Exercice 7.15 : eat

On ajoute une commande « eat » qui pour l’instant ne renvoie qu’une String mais aura une interaction futur lorsque on ajoutera les Items

Exercice 7.16 : showAll, showCommand

On ajoute une nouvelle méthode faisant appel à une boucle « for each » qui parcourt les commandes que l’on ajoute dans le jeu

Exercice 7.18 : getCommandList

On ajoute une nouvelle méthode renvoyant les noms des commandes que l’on ajoute dans le jeu

Exercice 7.18.1 : comparaison à zuul-better

ToDo

Exercice 7.18.2 : StringBuilder

Le StringBuilder permet d’ajouter à une première String de nouveau charactères (ex : on ajoute la liste des commandes disponibles à la String « les actions que vous pouvez executer sont : » ) et ainsi de pouvoir une seul et unique String

Exercice 7.18.3 : recherche d’image

Exercice 7.18.4 : titre du jeu

TelePac pour faire référence aux téléporteurs et à Pac-Man tout en conservant un nom court et accrocheur

Exercice 7.18.5 : HashMap Rooms

On créé une Hashmap contenant toutes les Room ce qui facilite l’organisation vu que le jeu contient 49 rooms

Exercice 7.18.6 : zuul-with-image

On ajoute des images au jeu

Exercice 7.18.8 : On ajoute des méthodes nécessitant un keyListenner afin de créer des boutons appelant les méthodes « quit », « look », …

Exercice 7.20 : Item

On ajoute un objet Item a une Room avec pour attribut son nom et un poids

Exercice 7.21 : Item description

Les informations concernant les items doivent être stockés dans un premier temps au même endroit que celle de la Room puisque ToDo

Exercice 7.22 :

Exercice 7.14 :

Exercice 7.14 :

Exercice 7.14 :