RAPPORT DE PROJET FINAL

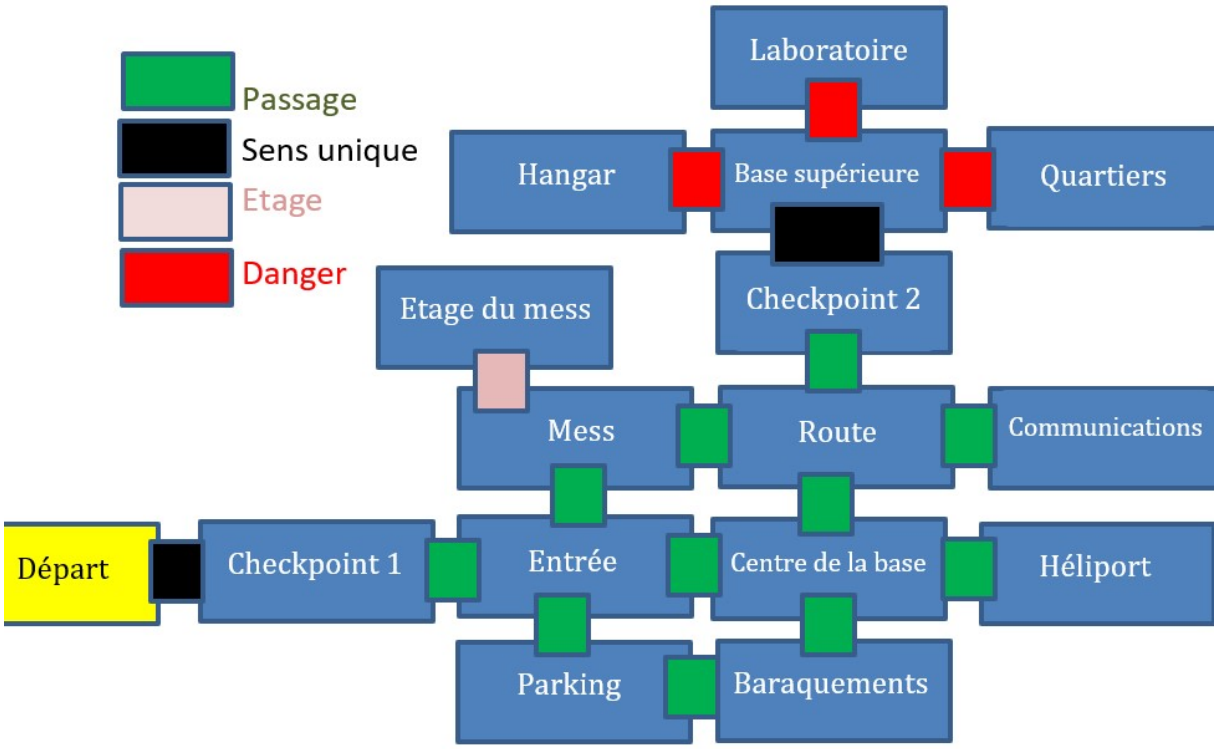
Investigation in Area 51

Alexandre DIAS

A3P 2019/2020

Un agent du FBI doit enquêter sur des phénomènes anormaux dans la Zone 51

Il faut parcourir un certain nombre de zones, dans un temps limité (en actions), s'équiper, puis affronter la cause des phénomènes anormaux et secourir une garde.



Scénario :

On est appelé en renfort dans la zone 51, afin d'enquêter sur des « phénomènes anormaux ». On a refusé de vous en dire plus avant que vous soyez sur place.

Tout d'abord, il est fortement conseillé de récupérer ce qui est dans la zone de départ. Les rations militaires auraient des propriétés insoupçonnées, les « Beamers » sont toujours utiles et un badge peut servir. Ensuite, on est prêt à passer le premier checkpoint. On est alors averti qu'il n'est pas autorisé de re sortir de la base. « Vous pouvez récupérer votre arme de service, nous l'avons démantelée afin d'éviter tout risque ». On récupère donc notre revolver calibre 50, sans barillet ni

balles. Il faudra le réassembler et le recharger par la suite ; mais il serait judicieux de commencer par enquêter et inspecter la base.

Une fois passé le checkpoint, on trouve dans le parking un morceau de tasse, ce qui nous renvoie au mess. Une fois arrivé là-bas, on apprend qu'une garde a disparu, et que son amie est à l'étage. On s'y dirige donc. L'amie nous dirige vers l'héliport car la disparue avait pour habitude de voler pendant son temps libre. L'officier de vol nous dit qu'un de ses hélicoptères a disparu. Peut-être que le centre des communications pourra nous donner plus de détails quant aux circonstances de la disparition.

Au centre des communications, on nous apprend que le contact a été perdu avec l'hélicoptère alors qu'il revenait par le nord à la base.

On peut alors se diriger vers le 2e checkpoint ou choisir d'explorer plus la base. Il est recommandé de récupérer le barillet du revolver dans les baraquements ainsi que la balle que donne le garde du 2E checkpoint. Une fois passé le 2e checkpoint, une fois encore impossible de faire marche arrière, sauf si l'on est en possession d'un Beamer chargé dans un lieu antérieur. Il faut ensuite se diriger vers l'un des trois lieux de la base supérieure. Ici, on rencontre l'alien. Là, plusieurs options sont possibles.

Lieux / Items / Personnages

La zone de départ : Contient quelques items utiles, impossible d'y accéder à nouveau à moins d'utiliser le Beamer.

Items : FBI_Beamer (permet la téléportation vers un lieu mémorisé)

military_ration (permet de doubler le poids maximum porté par le personnage, remplace le magic cookie)

FBI_badge (un badge)

Le premier checkpoint : Explications sur le jeu et la situation de la base et du personnage.

Items : sandbag (juste un sac de sable, ne sert à rien mais pèse très lourd.)

revolver (arme de service, vitale mais pas en état de marche lorsque ramassée)

Entrée / Centre de la base / Route / Base supérieure : Lieux de transit, permettant l'accès à divers endroits du jeu.

Parking : Premier lieu à indices, on y trouve un morceau de porcelaine.

Item : cup (indice pointant implicitement vers le mess)

Mess : Lieu de réunion et cantine des soldats, on y trouve de la nourriture et des renseignements.

Personnage : Un garde suggérant d'aller à l'étage pour en savoir plus.

Item: military_ration

Etage du Mess : C'est ici que certains mangent.

Personnage : L'amie de la disparue, elle nous apprend que cette dernière volait pendant son temps libre.

Item : pilot_card (carte de pilote de la disparue)

Baraquements : C'est là que vivent les soldats hors de leur service. L'enquête ne nous y mène pas particulièrement.

Item : barrel (barillet du revolver, essentiel à la victoire. Une fois acquis, on peut assembler le revolver)

Héliport : Zone d'atterrissage et de décollage des hélicoptères.

Personnage : L'officier de vol, qui signale qu'un de ses appareils a disparu dernièrement. Indique au joueur le centre des communications.

Centre des communications : Centre névralgique des opérations de la base, gère le trafic aérien.

Personnage : Officier en charge, indique que le contact avec un hélicoptère a été perdu lorsqu'il revenait à la base par le nord.

Item : Guards_Beamer (second Beamer, peut être utile, surtout si l'on n'a pas ramassé celui de la zone de départ)

Checkpoint 2 : Deuxième et dernier checkpoint. En allant plus loin, il est impossible de faire demi-tour, car cette partie de la base est en quarantaine.

Personnage : Garde, indique au joueur que la partie nord de la base est en quarantaine, lui donne une balle de .50 cal

Item : bullet (balle de calibre 50, pouvant être chargée dans le revolver une fois qu'il a été assemblé.)

Hangar / Laboratoire / Quartiers : Zones de fin de jeu, ne pouvant être quittées qu'en utilisant un Beamer chargé. Alien présent.

Personnage : Alien : Empêche tout mouvement du joueur autre que le tir ou la téléportation, lui faisant perdre la partie s'il tente autre chose.

Son élimination fait remporter la victoire.

Conditions de victoire et de défaite

1) On possède un revolver assemblé et chargé : On tire sur l'alien, sauve la garde et gagne la partie
2) On fait la mauvaise action en présence de l'alien : Conduit à la défaite

3) On prend trop de temps avant d'arriver à l'alien : Défaite
4) Prise de photo en présence de l'alien : Fin mystère

Enigmes / Combats

Enigmes : Déterminer l'ordre des lieux à inspecter à partir des indices trouvés dans chacun d'entre eux et des dialogues avec les personnages secondaires. Nécessité de trouver les composants du revolver et d'en effectuer la restauration dans le bon ordre.

Combat : Duel au revolver contre l'Alien, dont résulte l'issue de la partie.

Ce qu'il reste à faire :

Une amélioration graphique du jeu, intégrant une prévisualisation des zones adjacentes à celle dans laquelle on est.

Différents niveaux de difficulté (actuellement, le jeu présente une difficulté moyenne), pouvant inclure un nombre d'actions plus ou moins limité dans le temps, plusieurs balles nécessaires pour tuer l'alien, etc…

Réponse aux exercices

7.5

printLocationInfo() Affiche description du lieu et sorties.

Appliqué dans goRoom() et printWelcome().

7.6

getExit(direction) permet de supprimer les cas particuliers (« north », « east », …) des vérifications, permettant d'implémenter plus simplement de nouvelles directions.

7.7

Similairement à la 7.5, on ajoute getExitString() dans la classe Room, puis on adapte printLocationInfo() à cette nouvelle méthode.

7.8

Avec HashMap, on crée une carte virtuelle en associant des clés avec des variables, puis on utilise setExit() pour implémenter individuellement les sorties de chaque lieu (contrairement à setExits() qui contraignait à renseigner explicitement les sorties où il n'y a rien, et restreignait les possibilités d'ajouts de sorties autres (up, down…).

7.9

On incorpore la méthode keySet() après avoir importé Set et Iterator.

7.10

Tout d'abord on crée une String « returnString » égale à « Exits : ». On parcourt le set de key, en effectuant des itérations, puis on ajoute la key associée à la sortie de « returnString ».

7.11

getLongDescription() concatène description du lieu et sorties disponibles. Meilleure lecture du code. Appliqué dans printLocationInfo() et dans look().

7.14

Procédure look(), appelle getLongDescription() pour réafficher description et sorties.

7.15

Nouvelle procédure (ici « picture() », permet d'afficher que l'on prend une photo).

7.16

Création de fonctions permettant d'afficher quelles sont les commandes valides de manière « responsible ». Utilisé dans help().

7.17

Un changement de la classe Game n'est plus nécessaire, seul validCommands prend en compte les nouvelles commandes.

7.18

Ajout d'une méthode getCommandList() dans CommandWords, de getCommandString() dans Parser, retrait de showCommands. printHelp() devient plus responsible en intégrant automatiquement les nouvelles commandes grâce à System.out.println(this.aPartser.getCommandString()) ;

7.18.3

Les images du jeu ont pour la plupart été ajoutées dans le dossier Images. « Images » textuelles créées sur Paint afin de combler le manque d'images pour certaines zones.

7.18.4

Titre du jeu : Investigation in Area 51

7.18.6

Intégration de UserInterface et GameEngine.

7.18.8

Ajout de boutons permettant toutes les actions actuellement possibles dans le jeu.

7.19

Cf. Classe Item

7.21

Cf. Classe Item

7.22

Cf. Classe Item

7.22.1

Ajout de quelques Items au jeu, dont un dans la zone de départ.

7.23

Ajout de la commande back, permettant de retourner à la zone précédente

7.24

Si un second mot est entré après « back », le jeu affichera « It’s not how the command works » .

Si le joueur est déjà à la zone de départ, le jeu affichera « You’re at the beginning of the game ».

Si le joueur persiste, le jeu affichera « You’re already at the beginning of the game”.

7.25/26

On utilise une Stack afin de mémoriser les Rooms visitées par le joueur.

7.26.1

Les deux docs ont été générées.

7.28.1

La commande test et les 3 fichiers ont été créés. La commande est fonctionnelle, tout comme le jeu.

7.29

On a créé la classe Player, qui a été responsabilisée. Elle est en charge de la pièce dans laquelle le joueur est, du nom du joueur, des items qu’il porte, des pièces visitées.

7.30

Les procédures take() et drop() ont été créées et sont fonctionnelles

7.31

Il faut créer une liste d’items

7.31.1

La classe ItemList a été créée, elle régit les inventaires de chaque joueur et de chaque Room.

7.32

Le poids max a été créé comme attribut dans player, et des vérifications sont faites à chaque ramassage d'objet (booléen canTake()) afin de le permettre ou non. Le poids porté varie à chaque ramassage ou dépôt d'objet, et ne dépasse jamais le poids max. autorisé. Sac de sable supérieur au poids max. créé au premier checkpoint afin de vérifier la fonctionnalité.

7.33

Ajout d'une commande « inventory » renvoyant le contenu de l'inventaire, ainsi que toutes les informations relatives au poids.

```
public void inventory()

  {

     this.aGui.println(this.aPlayer.getInventoryString());

     this.aGui.println("Current weight carried: "+
this.aPlayer.getWeight()+"/"+this.aPlayer.getMaxWeight()+"\n");

  }//inventory()
```

7.34

Ajout d'un booléen dans la création des items, aEatable, définissant si l'item est comestible ou non. Utilisé via le getter « isEatable() ». Le « magic cookie » est remplacé dans le jeu par les « military_ration », pour des raisons d'immersion. « eat military_ration » double le poids max porté par le joueur.

Extrait de la méthode « eat() » :

```
tring vItem = pCommand.getSecondWord();

     if (this.aPlayer.getInventory().hasItem(vItem) &&
this.aPlayer.getInventory().getItem(vItem).isEatable())

     {

        this.aPlayer.setWeight(this.aPlayer.getWeight()-
this.aPlayer.getInventory().getItem(vItem).getItemWeight());

        this.aPlayer.getInventory().removeItem(vItem);



        this.aGui.println("You eat "+ vItem);

        this.aPlayer.setMaxWeight(this.aPlayer.getMaxWeight()*2);
```

```
        this.aGui.println("You now feel much stronger ! \n That was unexpected!");
    }
```

7.34.1

Fichiers test mis à jour

7.34.2

Javadoc générées

7.42

Time limit implémentée dans l'interprétation de commandes. Certaines des actions du joueur incrémentent un compteur « timeSpent ». La partie est perdue lorsque plus de 40 de ces actions sont réalisées (valeur modifiable afin de changer la difficulté du jeu).

7.42.2

Exercice passé.

7.43

Ajout d'une Hashmap <String, Boolean> aTrapDoor dans la classe Room, liée aux sorties. Les trapdoor sont définies individuellement, à chaque sortie. « false» signifie que la sortie n'est pas piégée, « true » qu'elle est piégée.

7.44

Classe Beamer créée, héritant d'Item. Possède deux attributs en plus de ceux présents dans les Items, à savoir aChargedRoom, qui est une Room enregistrée dans le Beamer lorsqu'on le charge, et le booléen aCharged, qui indique l'état du Beamer.

Méthode charge() :

```
  private void charge(final Command pCommand)
  {
    if(!pCommand.hasSecondWord())
    {
      this.aGui.println(" What should be charged?");
      return;
    }
    String vStringBeamer = pCommand.getSecondWord();
    Beamer vBeamer = (Beamer)this.aPlayer.getInventory().getItem(vStringBeamer);
```

```
    if(vBeamer == null)
    { this.aGui.println("You  don't have it!");}
    else{
       vBeamer.charge(this.aPlayer.getCurrentRoom());

       this.aGui.println("The Beamer is charged!");
    }
  }//charge(.)
```

Méthode fire() :

```
private void fire(final Command pCommand)
  {
    if(!pCommand.hasSecondWord())
    {
       this.aGui.println("Which Beamer should be used?");
       return;
    }
    String vStringBeamer = pCommand.getSecondWord();
    Beamer vBeamer = (Beamer)this.aPlayer.getInventory().getItem(vStringBeamer);
    if(vBeamer == null) this.aGui.println("You don't have it !");
    else if(vBeamer.isCharged()==false) this.aGui.println("The Beamer isn't charged... \n You should probably do that first");
    else{
       this.aPlayer.clearPreviousRooms();
       this.aPlayer.setCurrentRoom(vBeamer.getChargedRoom());
       vBeamer.discharge();
       printLocationInfo();
    }
```

}//fire(.)

7.45.1

Fichiers de test modifiés (principalement exploration et victoire)

7.45.2

Javadoc générées.

Mode d'emploi :

Bien que le jeu ne soit pas supposé présenter de difficultés de « gameplay » très poussées, mieux vaut cliquer sur le bouton « Need help ? » en début de jeu. Les textes affichés peuvent être assez longs, il convient donc de naviguer verticalement dans le panneau d'affichage de texte du GUI après certaines actions et chaque changement de zone afin d'être certain de ne rien manquer.

Déclaration anti-plagiat

Aucun morceau de code n'a été recopié sur qui que ce soit, en dehors du code donné dans les fichiers zuul-*.jar figurant sur iCampus,

Images tirées des films Alien, des jeux Arma III, Grand Theft Auto V, Grand Theft Auto San Andreas et de Google