Alexandre Bonnet (Groupe 2)

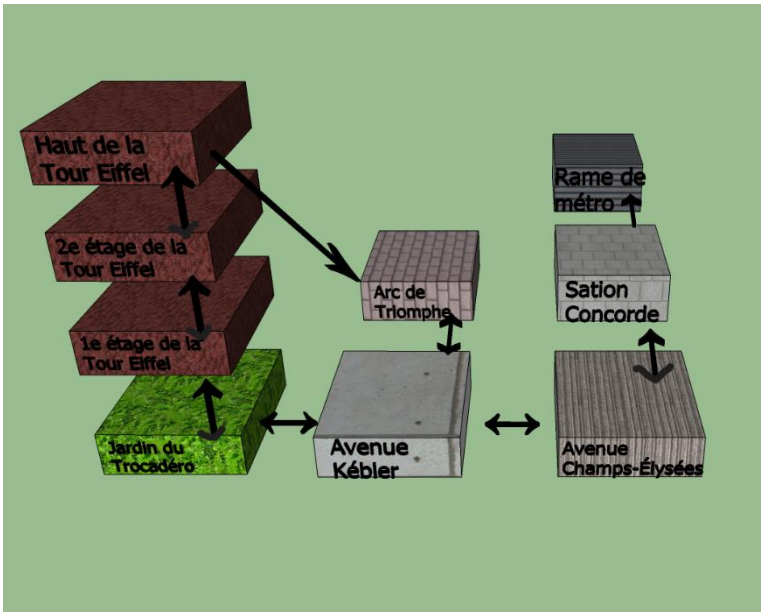
Rapport projet Zuul

Partie I)

Titre : José est posé

Auteur : Alexandre Bonnet

Phrase Thème : Dans les rues de Paris, José le pigeon essaie de retrouver son nid.



Plan en 3D

Résumé :

José se retrouve perdu dans Paris, il doit donc voler à travers des lieux connus afin de retrouver son nid. Au long de l'aventure, il pourra ramasser des vers (bonus) ou des bouchons en plastique (malus). Il y aura d'autres pigeons qui pourront être agressifs et d'autres péripéties sur le chemin.

Scenario détaillé :

José le pigeon se réveille dans le jardin du Trocadéro. Il doit donc retrouver son nid en passant par divers endroits. Il lui faut en premier accéder au dernier étage de la tour Eiffel afin de visiter le sage dans le but d'obtenir le nom de la station où l'on devra descendre. Le sage parle beaucoup, il faudra donc déchiffrer son message.
En revanche, il est impossible de monter dans la tour car notre rival bloque la porte, il faut le batter dans un duel de pierre, feuille, ciseaux afin de débloquer la porte.
Au 2^{e} étage se trouve un donut, il est possible de le manger, mais il sera utile plus tard.
Ensuite, en essayant de partir en direction de l'avenue des champs Elysées, nous nous rendons compte que la porte est bloquée, elle peut être ouverte à l'aide d'un bâton de sucette.
Un bruit d'enfants qui jouent se fait entendre à l'arc de triomphe, si on charge notre combo en réussissant à résoudre des multiplications, il est possible de faire lâcher la sucette aux enfants.
Une fois la sucette acquérie, il nous est impossible de rentrer dans le métro sans pass.
Il est donc possible de chiper un pass à un passant au café Kleber, ou de donner des donuts aux contrôleurs (ne marche pas à tous les coups).
Enfin il ne nous reste plus qu'à descendre à la station indiquée par la Sage.

Alexandre Bonnet (Groupe 2)

Liste des items :

- Branche (beamer)
- Pass Navigo (le pass vous permettra de débloquer la salle du métro)
- Lunettes : (permettent de chiper un pass navigo)
- Cookie (augmente poid max)
- Donut (permet de corrombre les controleurs)
- Sucette (clé de avenue des champs elysées)

Personnages : Pigeon rival (André) (celui-ci va vous empêcher d'aller en haut de la tour Eiffel, il faudra le battre en combat singulier afin de pouvoir passer)
Contrôleurs RATP (si vous rentrez dans le métro sans avoir débloqué l'option Navigo , les contrôleurs vous attrapent ! il est toutefois possible de leur donner un donut pour tenter de passer)
Le sage (le sage vous donnera de façon énigmatique l' arrêt où se trouve votrenid)
Enfants mal élevés (ils possèdent une clé que vous souhaitez obtenir, interagissez avec eux pour l'obtenir)

Situations : Vous ne gagnez que si vous arrivez au nid saint et saufs.
Vous pouvez perdre si : vous vous faites attraper par les contrôleurs ou les enfants, si vous perdez contre votre rival pigeon, si vous descendez à la mauvaise station de métro.

7.5 /

```
private void printRoomInfo(){
    System.out.println(this.aCurrentRoom.getDescription());
    System.out.print("Exits :");
    if (this.aCurrentRoom.aUpExit!=null){
            System.out.print("Up ");
    } if (this.aCurrentRoom.aDownExit!=null){
            System.out.print("Down ");
    } if (this.aCurrentRoom.aNorthExit!=null){
            System.out.print("North ");
    } if (this.aCurrentRoom.aSouthExit!=null){
            System.out.print("South ");
    } if (this.aCurrentRoom.aWestExit!=null){
            System.out.print("West ");
    } if (this.aCurrentRoom.aEastExit!=null){
            System.out.print("East ");
    } System.out.println(" ");
}
```

```
private void goRoom(final Command pDirection){
    //A
    if(!pDirection.hasSecondWord()){
        System.out.println("Où veux-tu aller ?");
        return;
    }
    //B
    Room vNextRoom = aCurrentRoom.getExit(pDirection.getSecondWord());

    //C et D
    if (vNextRoom==null){
        System.out.println("Tu ne peux aller dans cette direction");
        return;
    } else {
        this.aCurrentRoom = vNextRoom;
        printRoomInfo();
    }
}
```

```
private void printWelcome(){
    System.out.println(" ");
    System.out.println("================");
    System.out.println("Bienvenue dans Paris !");
    System.out.println("Dépêches toi, tu dois Retrouver ton nid !");
    System.out.println("Tapes 'help' si tu as besoin d'aide");
    System.out.println("================");
    System.out.println(" ");
    printRoomInfo();
}
```

Ici J'ai créé une nouvelle procédure "printRoomInfo" dans Game Afin d'éviter la duplication de code. J'avais déjà eu l'idée mais je ne l'avais pas implémentée. Cale nous donne toutes les infos de la room actuelle (description et sorties). Nous pourront donc faire appel à printRoomInfo dans printWelcome ougoRoom

Alexandre Bonnet (Groupe 2)

7.6 /

```
public Room getExit(final String pDirection){
    if (pDirection.equals("up")){
        return aUpExit;
    } else if (pDirection.equals("down")){
        return aDownExit;
    } else if (pDirection.equals("north")){
        return aNorthExit;
    } else if (pDirection.equals("south")){
        return aSouthExit;
    } else if (pDirection.equals("west")){
        return aWestExit;
    } else if (pDirection.equals("east")){
        return aEastExit;
    }
    return null;
}
```

J'ai créé un getter qui nous permet d'avoir les sorties de la room actuelle dans toutes les directions. Si il y a une sortie cela nous retourne son nom, sions c'est null qui est retourné.

Ensuite, afin de se déplacer dans la prochaine salle nous rajoutons ceci à la classe Game :

Room vNextRoom = aCurrentRoom.getExit(pDirection.getSecondWord());

7.7 /

```
public String getExitString(){
    String txtExit = "Exits :";
    if (this.aExits.get("up") != null) {
    txtExit += " up";
    }
    if (this.aExits.get("down") != null) {
    txtExit += " down";
    }
    if (this.aExits.get("north") != null) {
    txtExit += " north";
    }
    if (this.aExits.get("south") != null) {
    txtExit += " south";
    }
    if (this.aExits.get("west") != null) {
    txtExit += " west";
    }
    if (this.aExits.get("east") != null) {
    txtExit += " east";
    }
    return txtExit;
}
```

Cette procédure nous permet d'obtenir un String avec toutes les sorties de la salle actuelle pour apres la printer d'une seule fois.

```
private void printRoomInfo(){
    System.out.println(this.aCurrentRoom.getDescription());
    System.out.println(this.aCurrentRoom.getExitString());
    System.out.println(" ");
}
```

7.8/

```
import java.util.HashMap;
import java.util.Set;

public class Room
{
    private String aDescription;
    private HashMap<String, Room>aExits;
```

Ici j'ai créé une hashMap aExits qui permettra de stocker les Rooms.
Apres j'ai changé le constructeur de la classe Room, le getter getExits . J'ai ensuite modifié intégralement setExits.

```
public Room(final String pDescription){
    this.aDescription = pDescription;
    aExits = new HashMap<String, Room>();
}
```

```
public void setExits(final String pDirection, fina
   aExits.put(pDirection,pNextRoom);
} // change les parametres

/**
* retourne les sorties de la room
*/
public Room getExit(final String pDirection){
    return aExits.get(pDirection);
}
```

```
vJardinDuTrocadero.setExits("up",vPremierEtageTourEiffel);
vJardinDuTrocadero.setExits("east",vAvenueKleber);
```

7.9/ Nous devons modifier la méthode getExitString() pour l'adapter à une hashmap.

```
public String getExitString(){
    String txtExit = "Exits :";
    Set<String> keys = aExits.keySet();
    for (String vExit : keys) {
        txtExit += ' ' + vExit;
    }
    return txtExit;
}
```

7.11/ Dans printLocationInfo() nous allons printer la méthode getLongDesciption

```
public String getLongDescription(){
    return this.aDescription + ".\n" + getExitString();

}
```

7.14 et 25/ Nous voulons être capables d'ajouter des commandes à notre guise. Nous allons donc créer un tableau de commandes valides.

```
private final String[] aValidCommands
= {"go","quit","help","look","eat"};
```

Et nous modifions la méthode qui sert à retourner la liste de commandes.

```
public String getCommandList(){
    String vCommand = "";
    for(String command : aValidCommands){
        vCommand += command + ", ";
    } return vCommand;
}
```

Nous avons par exemple ajouté une commande "look" qui retourne la description de la salle courante.
Ou encore la commande "eat" qui nous print :
"t'as ramassé les miettes sur le sol , tu es rassasié"

7.16/ Nous avons créé une methode "showCommands" qui retorune les commandes existantes.

```
/**
 * returns a String list of valid commands
 */
public String getCommandList(){
    String vCommand = "";
    for(String command : aValidCommands){
        vCommand += command + ", ";
    } return vCommand;
}
```

Nous pouvons ensuite l'utiliser pour la commande help par exemple.

7.19/
Afin de pouvoir avoir acces aux images, nous modifions les attreeibuts du constructeur room en ajoutant le nom de l'image pour chaque pièce.

```
Room vJardinDuTrocadero = new Room("Vous êtes dans le jardin du Trocadéro","Images/_JardinDuTrocadero.jpg");
```

Alexandre Bonnet (Groupe 2)

7.20-21/
Nous créons des attributs pour le nouvel objet item. Dans Room nous créons des getters et setters d'items afin de pouvoir affecter un item à une room.

Nous créons ensuite dans Room une methode "get item description" afin de retourner les attributs des objets présents dans la room.

```java
public class Item
{
    private String aDescription;
    private int aPoids;

    /**
     * Constructeur d'objets de classe Item.
     */
    public Item(final String pDescription, final int pPoids){
        this.aDescription = pDescription;
        this.aPoids = pPoids;
    }

    /**
     * Obtient la description de l'objet.
     */
    public String getItemDescription(){
        return "Il y a : " + this.aDescription + " . Cela pèse "
    }
}
```

7.22/
Nous créons une hashmap d'items dans Room, puis deux méthodes pour ajouter et enlever des items. Nous modifions ensuite la méthode getItemString de Room.

```java
private HashMap<String, Item>aItems;
```

```java
/**
 * Ajoute un objet à la liste des objets possédés par le joue
 */
public void addItem(final String pName,final Item pItem){
    this.aItems.put(pName,pItem);
}
```

```java
/**
 * Supprime un objet de la liste des objets possédés par le j
 */
public void removeItem(final String pName,final Item pItem){
    this.aItems.remove(pName,pItem);
}
```

```java
/**
 * Obtient une représentation sous forme de chaîne des objets possédés par l
 */
public String getItemString(){
    StringBuilder aItemDescription = new StringBuilder("Items : " + "\n");
    for (Item aItem : aItems.values()) {
        aItemDescription.append(aItem.getItemDescription() + "\n");
    }
    return aItemDescription.toString();
}
```

7.23/

7.28/

Afin d'automatiser les tests nous créons des fichiers textes.

7.29/
Nous déplaçons toutes les information dans leur classes respectives. Cela nous force donc à changer la classe GameEngine où il nous faut passer par des accesseurs de player pour obtenir la room actuelle par exemple.

7.30-31/
Il faut d'abord créer une hashmap d'items dans la classe Players qui possède les mêmes méthodes que celle de Room (add, remove…).
Nous allons aensuite créer les fonctions take et drop qui vont :

- Prende l'objet de la hashmap de la room pour la mettre dans celle de player
- Prende l'objet de la hashmap du player pour la mettre dans celle de la room

7.32/

Alexandre Bonnet (Groupe 2)

```
private int aPoidsMax;
private int aPoids;
public int getPoids(){
    return this.aPoids;
}
public void setPoids(final int pPoids){
    this.aPoids = pPoids;
}
public int getPoidsMax(){
    return this.aPoidsMax;
}
```

Dans player :

Dans items :
Getter de aPoids.

Dans gameEngine :

```
private void take(final String pCommand){
    Item vItem = this.aPlayer.getCurrentRoom().getItem(pCommand);
    if(vItem != null){
        int vTotPoids = this.aPlayer.getPoids() + vItem.getPoids();
        if(vTotPoids <= this.aPlayer.getPoidsMax()){
            this.aPlayer.addItem(pCommand,vItem);
            this.aPlayer.setPoids(vTotPoids);
            this.aPlayer.getCurrentRoom().removeItem(pCommand,vItem);
            this.aGui.println(this.aPlayer.getCurrentRoom().getItemString());
            this.aGui.println(this.aPlayer.getItemString());
        } else {
            this.aGui.println("You can't take more items");
        }
    } else {
        this.aGui.println("There is no " + pCommand + " here");
    }
}

/**
 * allows player to transfer items from inventory to room
 */
private void drop(final String pCommand){
    Item vItem = this.aPlayer.getItem(pCommand);
    if(vItem != null){
        this.aPlayer.removeItem(pCommand,vItem);
        this.aPlayer.setPoids(this.aPlayer.getPoids() - vItem.getPoids());
        this.aPlayer.getCurrentRoom().addItem(pCommand,vItem);
        this.aGui.println(this.aPlayer.getCurrentRoom().getItemString());
        this.aGui.println(this.aPlayer.getItemString());
    } else {
        this.aGui.println("You don't have any " + pCommand);
    }
}
```

```
private void take(final String pCommand){
    Item vItem = this.aPlayer.getCurrentRoom().getItem(pCommand);
    if(vItem != null){
        int vTotPoids = this.aPlayer.getPoids() + vItem.getPoids();
        if(vTotPoids <= this.aPlayer.getPoidsMax()){
            this.aPlayer.addItem(pCommand,vItem);
            this.aPlayer.setPoids(vTotPoids);
            this.aPlayer.getCurrentRoom().removeItem(pCommand,vItem);
            this.aGui.println(this.aPlayer.getCurrentRoom().getItemString());
            this.aGui.println(this.aPlayer.getItemString());
        } else {
            this.aGui.println("You can't take more items");
        }
    } else {
        this.aGui.println("There is no " + pCommand + " here");
    }
}

/**
 * allows player to transfer items from inventory to room
 */
private void drop(final String pCommand){
    Item vItem = this.aPlayer.getItem(pCommand);
    if(vItem != null){
        this.aPlayer.removeItem(pCommand,vItem);
        this.aPlayer.setPoids(this.aPlayer.getPoids() - vItem.getPoids());
        this.aPlayer.getCurrentRoom().addItem(pCommand,vItem);
        this.aGui.println(this.aPlayer.getCurrentRoom().getItemString());
        this.aGui.println(this.aPlayer.getItemString());
    } else {
        this.aGui.println("You don't have any " + pCommand);
    }
}
```

7.33/
Dans item:
Getter de aDescription

Dans Player;

```
public String getItemString(){
    StringBuilder aItemDescription = new StringBuilder("Your items are : " + "\n");
    for (Item aItem : aItems.values()) {
        aItemDescription.append(aItem.getDescription() + "\n");
    }
    aItemDescription.toString();
    int vPoids = 0;
    for (Item aItem : aItems.values()) {
        vPoids +=aItem.getPoids();
    }
    String vDescirption = aItemDescription  +"Le total pèse : "+ vPoids;
    return vDescirption;
}
```

```
public String getItemString(){
    StringBuilder aItemDescription = new StringBuilder("Your items are : " + "\n");
    for (Item aItem : aItems.values()) {
        aItemDescription.append(aItem.getDescription() + "\n");
    }
    aItemDescription.toString();
    int vPoids = 0;
    for (Item aItem : aItems.values()) {
        vPoids +=aItem.getPoids();
    }
    String vDescirption = aItemDescription  +"Le total pèse : "+ vPoids;
    return vDescirption;
}
```

7.34/
Dans Game Engine;

```
} else if ( vCommandWord.equals( "eat" ) ) {
    if (!vCommand.hasSecondWord()){
        aGui.println("Eat what?!");
    } else {
        eat(vCommand.getSecondWord());
    }
```

methode "interpretCommand"

Alexandre Bonnet (Groupe 2)

```
private void eat(final String pCommand){
    if(this.isCommestible(pCommand)){
        Item vItem = this.aPlayer.getItem(pCommand);
        if(vItem == null){
            this.aGui.println("You don't have any " + pCommand);
        } else {
            if(vItem.getDescription().equals("cookie")){
                this.aGui.println("You have eaten magic cookie you can now carry more items");
                this.aPlayer.setPoidsMax(15);
            }
            if(vItem.getDescription().equals("pain")){
                this.aGui.println("NomNomNom");
            } if(vItem.getDescription().equals("miettes")){
                this.aGui.println("Eww c'est des cailloux");
            }
        this.aPlayer.removeItem(pCommand,vItem);
        this.aPlayer.setPoids(this.aPlayer.getPoids() - vItem.getPoids());
        this.aGui.println(this.aPlayer.getItemsWeight());
        } return;
    } else {
        aGui.println("This item is not commestible");
        return;
    }
}
```

```
private void eat(final String pCommand){
    if(this.isCommestible(pCommand)){
        Item vItem = this.aPlayer.getItem(pCommand);
        if(vItem == null){
            this.aGui.println("You don't have any " + pCommand);
        } else {
            if(vItem.getDescription().equals("cookie")){
                this.aGui.println("You have eaten magic cookie you can now carry more items");
                this.aPlayer.setPoidsMax(15);
            }
            if(vItem.getDescription().equals("pain")){
                this.aGui.println("NomNomNom");
            } if(vItem.getDescription().equals("miettes")){
                this.aGui.println('Eww c'est des cailloux');
            }
        this.aPlayer.removeItem(pCommand,vItem);
        this.aPlayer.setPoids(this.aPlayer.getPoids() - vItem.getPoids());
        this.aGui.println(this.aPlayer.getItemsWeight());
        } return;
    } else {
        aGui.println("This item is not commestible');
        return;
    }
}
```

<=Optionnel

7.42/

Nous allons donc compter le nombre de déplacements en pour faire office de temps, si la limite est dépassée, le jeu est perdu.

Dans Player :

```
private int aMovesMax;
private int aMoves;
```

Initialiser les deux dans le constructeur.

```
public int getMoves(){
    return this.aMoves;
}
public void addMove(){
    aMoves++;
}
public int getMovesMax(){
    return this.aMovesMax;
}
```

Dans GameEngine :

Nous créons une méthode loose, si on déplace le nb max (nous créons donc win en mêm temps)

```
private void loose(){
    this.aGui.showImage("Images/_Defaite.jpg");
    this.aGui.println("vous avez predu\n \n ");
    endGame();
}

private void win(){
    this.aGui.showImage("Images/_Victoire.jpg");
    this.aGui.println("vous avez gagné\n \n ");
    endGame();
}
```

Nous avons ensuite verifié si le nombre de déplacements est superieur au max
Et enfin ajouté un déplacement des qu'on bouge de pièce
(un retour "go where" ou "there is no door" n'est pas compté comme un déplacement)

Alexandre Bonnet (Groupe 2)

```
public void goRoom( final Command pCommand ){
    if ( ! pCommand.hasSecondWord() ) {
        // if there is no second word, we don't know where to go...
        this.aGui.println( "Go where?" );
        return;
    }

    String vDirection = pCommand.getSecondWord();
    String vString = this.aPlayer.goRoom(vDirection);
    // Try to leave current room.
    Room vNextRoom = this.aPlayer.getCurrentRoom().getExit( vDirection );
    if(this.aPlayer.getMoves()+1 >= this.aPlayer.getMovesMax()){
            loose();
            return;
    }

    if ( vString == null ) {
        this.aPlayer.addMove();
        this.printRoomInfo();
        if (this.aPlayer.getCurrentRoom().getImageName() != null )
            this.aGui.showImage(this.aPlayer.getCurrentRoom().getImageName() );
    } else {
        this.aGui.println(vString);
    }
}
```


```
public void goRoom( final Command pCommand ){
    if ( ! pCommand.hasSecondWord() ) {
        // if there is no second word, we don't know where to go...
        this.aGui.println( "Go where?" );
        return;
    }

    String vDirection = pCommand.getSecondWord();
    String vString = this.aPlayer.goRoom(vDirection);
    // Try to leave current room.
    Room vNextRoom = this.aPlayer.getCurrentRoom().getExit( vDirection );
    if(this.aPlayer.getMoves()+1 >= this.aPlayer.getMovesMax()){
            loose();
            return;
    }

    if ( vString == null ) {
        this.aPlayer.addMove();
        this.printRoomInfo();
        if (this.aPlayer.getCurrentRoom().getImageName() != null )
            this.aGui.showImage(this.aPlayer.getCurrentRoom().getImageName() );
    } else {
        this.aGui.println(vString);
    }
}
```


7.42.2/
Nous allons rester sur l'IHM actuelle pour le moment.
7.43/
Dans Room:
Nous allons créer une méthode "isExit" pour vérifier si la room précédente est une sortie de la room actuelle.

```
public boolean isExit(final Room pRoom){
    return this.aExits.containsValue(pRoom);
}
```

Dans Player:
On rajoute la 1ere condition, cela nous permet de ne rien modifier dans GameEngine.

```
/**
 * Retourne à la salle précédente visitée par le joueur.
 */
public String back(){
    if(!this.aCurrentRoom.isExit(this.aPreviousRooms.peek())){
        return "this was a one-way door";
    } else if(!this.aPreviousRooms.empty()){
        this.aCurrentRoom = this.aPreviousRooms.pop();
        return null;
    } else {
        return "no previous room";
    }
}
```


```
/**
 * Retourne à la salle précédente visitée par le joueur.
 */
public String back(){
    if(!this.aCurrentRoom.isExit(this.aPreviousRooms.peek())){
        return "this was a one-way door";
    } else if(!this.aPreviousRooms.empty()){
        this.aCurrentRoom = this.aPreviousRooms.pop();
        return null;
    } else {
        return "no previous room";
    }
}
```


7.44/
Dans CommandWords:
J'ai ajouté deux commandes "saveLoc" et "goLoc" pour le beamer

Alexandre Bonnet (Groupe 2)

Dans Player:
J'ai ajouté un attribut "aSavedLocation" qui contient la room sauvagardée. (ensuit j'ai fait un getter et setteur de cet attribut).

Dans GameEngine
J'ai ajouté un item "branche" dans le jardin du Trocadéro qui nous fera office de beamer.

Nous ajoutons ensuite dans interpretCommand les deux commandes

```
} else if(vCommandWord.equals("saveLoc")){
    saveLoc();
}   else if (vCommandWord.equals("goLoc")){
    goLoc();
}
```

Puis nous créons les deux méthodes qui leur correspondent.

```
private void saveLoc(){
    if(this.aPlayer.getItem("branche")!=null){
        this.aGui.println("your location is saved");
        this.aPlayer.setSavedLocation(this.aPlayer.getCurrentRoom());
        return;
    } else {
        this.aGui.println("no branche on you");
        return;
    }
}
```

```
private void goLoc(){
    if(this.aPlayer.getItem("branche")!=null){
        if(this.aPlayer.getSavedLocation()==null){
            this.aGui.println("you didn't save your location");
            return;
        } else{
            this.aPlayer.setCurrentRoom(this.aPlayer.getSavedLocation());
            this.aGui.println("You have been transported to your saved location");
            this.aGui.println(this.aPlayer.getCurrentRoom().getLongDescription());
            this.aGui.showImage(this.aPlayer.getCurrentRoom().getImageName());
            this.aPlayer.setSavedLocation(null);
            return;
        }
    } else {
        this.aGui.println("no branche on you");
        return;
    }
}
```

7.46/

Dans gameEngine:

J'ajoute une room "cabine téléphonique" au nord de l'arc de Triomphe qui nous servira de transporteur.
Il sera possible de ressortir de la salle ou d'utiliser la commande voyager (exclusive à la cabine) afin de se téléporter.
Puis je crée une liste des Rooms où la cabine sera autorisée à transporter le joueur.

```
Room[] vPossibleRooms = new Room[] {vJardinDuTrocadero,vPremierEtageTourEiffel,
vDeuxiemeEtageTourEiffel,vDernierEtageTourEiffel,vArcDeTriomphe,vAvenueKleber,vAvenueDesChampsElysees};
```

Nous créons une nouvelle classe "TransporterRoom" qui hérite de Room qui importe le module random

Exercice pas terminé (il est pas essentiel)

7.48/

Je crée une nouvelle classe "Characters" et lui donne les attributs "name" , "dialog" et "minigame".
Les minijeux possibles seront :
reveal (le sage revele la station),
passNavigo (les controleurs vérifient le pass)
pierreFeuilleCiseaux

Alexandre Bonnet (Groupe 2)

Dans Room:

Je crée toutes les méthodes nécéssaires (copy paste de items)

```
/**
 * Ajoute un objet à la liste des objets possédés par le joueur.
 */
public void addCharacter(final String pName,final Character pCharacter){
    this.aCharacters.put(pName,pCharacter);
}

/**
 * Supprime un objet de la liste des objets possédés par le joueur.
 */
public void removeCharacter(final String pName,final Item pCharacter){
    this.aItems.remove(pName,pCharacter);
}

/**
 * Obtient un objet de la liste des objets possédés par le joueur.
 */
public Character getCharacter(final String pNom){
    return this.aCharacters.get(pNom);
}

/**
 * Obtient une représentation sous forme de chaîne des objets possédés par le joueur.
 */
public String getCharacterString(){
    String txtCharacter = "Characters : ";
    Set<String> keys = aCharacters.keySet();
    for (String vCharacter : keys) {
        txtCharacter += ' ' + vCharacter+",";
    }
    return txtCharacter;
}
```

Dans GameEngine j'attribue des personnages à des rooms

```
vJardinDuTrocadero.addCharacter("André", new Character("André","Tu devras me battre à pierre,feuille,ciseaux
vDernierEtageTourEiffel.addCharacter("Le Sage", new Character("Le Sage","Bienvenue dans mon Sanctuaire","reve
vStationDeMetro.addCharacter("Controleur", new Character("Controleur","Ticket svp ?","passNavigo"));
```

Je crée ensuite une commande interract où il faudra entrer le nom du personnage.

```
else if (vCommandWord.equals("interract")) {
    if (!vCommand.hasSecondWord()){
        aGui.println("Interract with who ?");
    } else {
        interract(vCommand.getSecondWord());
    }
```

```
private void interract(final String pCommand){
    Character vCharacter = this.aPlayer.getCurrentRoom().getCharacter(pCommand);
    if(vCharacter != null){
        this.aGui.println(vCharacter.getDialog());
        if(vCharacter.getMinigame().equals("reveal")){
            reveal();
            return;
        } else if(vCharacter.getMinigame().equals("passNavigo")){
            passNavigo();
            return;
        } else if(vCharacter.getMinigame().equals("pierreFeuilleCiseaux")){
            pierreFeuilleCiseaux();
            return;
        }
    } else {
        this.aGui.println(pCommand + " is not here");
    }
}
```

Alexandre Bonnet (Groupe 2)

Minijeux:

Chaque minijeu va donc avoir sa méthode et 2 attributs booléens.
Pour savoir si le minijeu est actif et pour savoir s'il a déjà été joué.
Lorsque nous changeons de Room, les minijeux actifs sont désactivés

```
public void reveal(){
    //"Liberté", "Charenton","Ecole", "Maisons" aLastStation
    if (aLastStation.equals("Liberté")){
        this.aGui.println("mhhh Ton nid se trouve dans ma station préférée,\nc'est la sensation que tu ressens"
        +" quand tu voles avec tes ailes au dessus de Paris,\n une sentation que je ne suis pas prêt d'oublier\n"
        +"quand j'étais jeune je voyagais beaucoup,\n mais maintenant il est vrai que je reste plus ici, tranquilou");
    } else if (aLastStation.equals("Charenton")){
        this.aGui.println("mhhh Ton nid se trouve dans une station que j'aime beaucoup,\n elle me rapelle des souvenirs"
        +" quand j'étais jeune j'allais ici avec ma mère,\n elle m'offrait toujours une glace à la fin des balades"
        +"\n ahh que de souvenirs, je.. oh pardon.. votre nid se trouve à dans une station qui rime avec 'menton'");
    }else if (aLastStation.equals("Ecole")){
        this.aGui.println("mhhh Ton nid se trouve dans une station que je n'aime pas"
        +"\n elle me rapelle mon enfance, où j'étais enfermé des heures durant dans ces institutions"
        +"\n je m'y suis fait des copain c'est sûr, et j'y ait appris beaucoup c'est vrai"
        +"\n mais bon bien que cela soit un passage très formateur de la vie, j'en garde un souvenir terni");
    } else if (aLastStation.equals("Maisons")){
        this.aGui.println("mhhh Ton nid se trouve dans une station dont j'aime le nom"
        +"\n on s'y sent bien, c'est là qu'on se réunit à noël avec la famille"
        +"\n enfin Noël ET mon anniversaire, savez vous que je suis né le 24 Décembre ?"
        +"\n on me demande souvent ce que cela fait, mais boh, c'est un anniversaire comme un autre à vrai dire");
    }
}
```

Minijeu pierre papier ciseaux:

Dans CommandWords:
J'ai ajouté pierre papier et ciseax dans la liste de commandes cachées.
(cette liste permet de stocker les commandes qui ne sont disponibles que dans une salle, elles ne seront pas affichées lors de "help").

```
private static final String[] aHiddenValidCommands ={"descendre","voyager","pierre","feuille","ciseaux"};
```

Dans Game Engine :

```
public  boolean         aMiniGamePierreFeuilleCiseaux;
```

On l'initialise à false.
Nous les remettons à false lorsque nous changeons de room afin de ne pouvoir faire un minijeu que dans une room.

Nous ajoutons les commandes dans "interpretcommand()"

```
else if (vCommandWord.equals("pierre")||vCommandWord.equals("feuille")||vCommandWord.equals("ciseaux")){
  if(vCommand.hasSecondWord()){
      aGui.println("Just one word needed here");
  } else if(aMiniGamePFC){
      pierreFeuilleCiseaux(vCommandWord);
  } else{
      aGui.println("No active Pierre-Feuille-Ciseaux game, go see a npc to start the game");
  }
```

Et dans "interract" nous ajouton "pfc" qui correspond a pierre feuille ciseaux

```
    }else if(vCharacter.getMinigame().equals("pfc")){
        this.aMiniGamePFC=true;
        this.aGui.println("Play your move 'pierre', 'feuille' or 'ciseaux'");
        return;
```

Alexandre Bonnet (Groupe 2)

```
public void pierreFeuilleCiseaux(final String pMove){
    String[] pnjMoves = {"pierre", "feuille","ciseaux"};
    Random rand = new Random();
    int i = rand.nextInt(pnjMoves.length);
    String pnjMove = pnjMoves[i];
    boolean victoire = pnjMove.equals("pierre")&&pMove.equals("feuille")
    ||pnjMove.equals("feuille")&&pMove.equals("ciseaux")
    ||pnjMove.equals("ciseaux")&&pMove.equals("pierre");
    boolean defaite = pMove.equals("pierre")&&pnjMove.equals("feuille")
    ||pMove.equals("feuille")&&pnjMove.equals("ciseaux")
    ||pMove.equals("ciseaux")&&pnjMove.equals("pierre");
    this.aGui.println("Adversaire : "+pnjMove+" !");
    if(pMove.equals(pnjMove)){
        this.aGui.println("it's a draw, let's try again");
        return;
    } else if(victoire){
        this.aGui.println("tu as gagné, tu peux passer");
        this.aMiniGamePFC = false;
        vJardinDuTrocadero.setExits("up",vPremierEtageTourEiffel);
        vPremierEtageTourEiffel.unlockDoor();
    } else if(defaite){
        this.aGui.println("tu as perdu. Retente ta chance");
        this.aMiniGamePFC = false;
    }
    return;
}
```

J'ai donc créé un attribut dans Room pour savoir si une room est Locked ou non, quand le duel contre le pigeon est gagné, le premier étage est débloqué.

Minijeu tuxMaths:

J'ai rajouté un autre miniJeu où, lorsqu'on interagit avec les enfants, il faut résoudre 3 multiplications afin d'obtenir une clé (cf "système de clés").

J'ai ajouté la commande cachée 'reponse' active seulement quand un minijeu est actif.
Il y a donc des attributs qui gardent :

- Nombre random 1
- Nombre random 2
- Nombre de Bonne réponses
- Boolean qui vérifie si le minijeu est actif ou non

```
else if(vCharacter.getMinigame().equals("tuxMaths")){
    if(aMiniGameTuxAnswerCount>2){
        this.aGui.println("On a eu ce qu'on voulait, n'embêtons plus ces enfants");
        return;
    } else {
        this.aMiniGameTux = true;
        this.aGui.println("Il faudrait récupérer la sucette des enfatnts \n"
        +"pour cela il faut résoudre 3 multiplications");
        tuxMaths("intro");
        return;
    }
```

```
public void tuxMaths(final String pString){
    if(pString.equals("intro")){
        tuxMathsRandomiser();
    } else {
        if (pString.equals(""+aMiniGameTuxA*aMiniGameTuxB)){
            aMiniGameTuxAnswerCount++;
            this.aGui.println("Bonne réponse !      Combo chargé à " +aMiniGameTuxAnswerCount+ "/3");
            tuxMathsRandomiser();
        } else {
            aMiniGameTuxAnswerCount=0;
            this.aGui.println("Mauvaise réponse :(      Combo chargé à " +aMiniGameTuxAnswerCount+ "/3");
            tuxMathsRandomiser();
        }
    }
    if(aMiniGameTuxAnswerCount>2){
        this.aGui.println("Vous avez réussi votre combo \n Un Enfant a fait tomber sa sucette !");
        this.aGui.println("");
        vArcDeTriomphe.addItem("sucette",new Item("sucette", 2));
        this.aGui.println(this.aPlayer.getCurrentRoom().getItemString());
        this.aMiniGameTux = false;
        return;
    }
}
public void tuxMathsRandomiser(){
    Random rand = new Random();
    int i = rand.nextInt(50);
    int j = rand.nextInt(50);
    aMiniGameTuxA = i;
    aMiniGameTuxB= j;

    this.aGui.println("avec la commande 'reponse' suivie du résultat, calculez \n"
        + aMiniGameTuxA + " x " +aMiniGameTuxB);
}
```

Minijeu pass Navigo:

Pour accéder au métro il faut soit avoir un pass navigo, soit corrompre les controleurs.

Pour obtenir un pass j'ai créé une nouvelle room "café kleber" où il est possible de chiper le pass d'un passant.
Si nous essayons de voler la pass, nous avons 30% de chances de réussite, en revenche, cela passe à 100% si nous portons des lunettes de soleil.

```
public void chiper(final String pString){
    if(!this.aPlayer.getCurrentRoom().getImageName().equals("Images/_CafeKleber.jpg")){
        aGui.println("Il y n'y a personne à chiper dans ce lieu");
        return;
    } else {
        if(pString.equals("passNavigo")){
            Random rand = new Random();
            int i = rand.nextInt(10);
            if(i<4||this.aPlayer.getItem("lunettes")!=null){
                aGui.println(pString + " a été chipé avec succes");
                aMiniGameChiper = false;
                this.aPlayer.addItem("passNavigo",new Item("passNavigo",3));
                aGui.println(this.aPlayer.getItemString());
                return;
            } else {
                aGui.println("Passant : Eh oh! Satané pigeon, pars d'ici");
                aMiniGameChiper = false;
                return;
            }
        } else {
            aGui.println("Vous ne pouvez chiper cet objet");
            return;
        }
    }
}
```

Alexandre Bonnet (Groupe 2)

SI vous n'avez pas de pass, vous pouvez interragir avec les controleurs et leur donner un donut.
Cette tentative aura 50% de chances de marcher.

```
public void donutGive(final String pString){
    if(!this.aPlayer.getCurrentRoom().getImageName().equals("Images/_StationDeMetro.jpg")){
        aGui.println("Il y n'y a personne à qui donner cet objet ici");
        return;
    } else {
        if(pString.equals("donut")){
            Random rand = new Random();
            int i = rand.nextInt(10);
            if(i>5){
                aGui.println(pString + " a été donné avec succes");
                aMiniGameDonut = false;
                aGui.println("La porte a été déverouillée");
                aGui.println("Controleur : Passez avant que je change d'avis");
                vRameDeMetro.unlockDoor();
                return;
            } else {
                aGui.println("Controleur : Ceci est un donut de pètre qualité, hors de ma vue !");
                aMiniGameDonut = false;
                return;
            }
        } else {
            aGui.println("Vous ne pouvez donner de " + pString);
            return;
        }
    }
}
```

Système de clés:

J'ai donc rajouté un système de clés, avec un attribut pour chaque room "isLocked" si un objet spécifique est porté par le joueur.

Dans GameEngine

```
vAvenueDesChampsElysees.lockDoor();
vAvenueDesChampsElysees.setKey("sucette");
```

Lorsqueu'n épreuve est finie, il suffit d'utiliser "unlockDoor()".
Dans Room:

Deux nouveaux attributs.

```
private boolean     aIsLocked;
private Item        aKey;
```

```
public void setKey(final String pKey){
    this.aKey = new Item(pKey, 0);
}
public Item getKey(){
    return this.aKey;
}
public boolean hasKey(){
    return this.aKey!=null;
}

public boolean isLocked(){
    return aIsLocked;
}
public void unlockDoor(){
    aIsLocked = false;
}
public void lockDoor(){
    aIsLocked = true;
}
```

Système de choix de station finale:

Le la station finale (aLastStation) est un attribut String de GameEngine.

Dans le constructeur de gameEngine on fait appel à la méthode "chooseRandomStation"

```
public String chooseRandomStation(){
    String[] stations = {"Liberté", "Charenton",
    "Ecole", "Maisons"};

    Random rand1 = new Random();
    int i = rand1.nextInt(stations.length);

    return stations[i];
}
```

```
private void descendre(final String pCommand){
    if(!this.aPlayer.getCurrentRoom().getImageName().equals("Images/_RameDeMetro.jpg")){
        aGui.println("Vous n'êtes pas dans une rame de métro");
        return;
    } else {
        if(pCommand.equals(aLastStation)){
            win();
        } else {
            loose();
        }
        endGame();
        return;
    }
}
```

Alexandre Bonnet (Groupe 2)

Déclaration anti Plagiat :
Je déclare ne pas avoir plagié de contenu de mon jeu Zuul