Prison Break

"Be smarter than the guardians"

17 décembre 2015
Université de Lausanne, HEC

Object Oriented Programming

Professeur: Benoît Garbinato

Développeurs en herbe:

- Florent Plomb
- Matthieu Harbich
- Onur Erdogan
- Bryan Cornelius

UNIL | Université de Lausanne
HEC Lausanne

1. Description du jeu

Prison Break est un jeu dans lequel le personnage principal Zuul est un prisonnier qui tente de s'évader. En faisant face à plusieurs contraintes, notamment le gardien de la prison, Zuul doit trouver un moyen de prendre la fuite. En cas de réussite, la partie est gagnée. A l'inverse, si Zuul ne parvient pas à surmonter les défis qui lui sont adressés, la partie est perdue.

2. Mode d'emploi

2.1. Les personnages

- Zuul

Zuul est un prisonnier qui après s'être endormi, lors de la digestion de son repas habituel, se réveille et constate que sa cellule est ouverte. Dès lors, il tente de s'échapper.

- Le gardien

Le gardien de la prison se déplace aléatoirement dans les emplacements du jeu, Zuul à une certaine probabilité de le rencontrer, lors de chaque déplacement. En cas de confrontation, Zuul doit entamer un combat sous forme de quizz, afin d'éviter que le gardien puisse activer l'alarme.

3. Les objets

3.1. Transportables

- Peau de Banane

Une peau de banane est à disposition dès le début d'une partie. En effet, il s'agit du reste du repas et Zuul l'utilise en tant que bonus pour éviter de confronter un garde, lors de la première rencontre. Son poids est de 1 unité.

- Echelle

Une échelle est répartie aléatoirement dans une des 10 pièces du jeu. Il s'agit d'un objet complémentaire à la couverture, permettant à Zuul de s'évader de la prison en grimpant le mur, situé dans la cour de la prison. Son poids est de 8 unités.

- Couverture

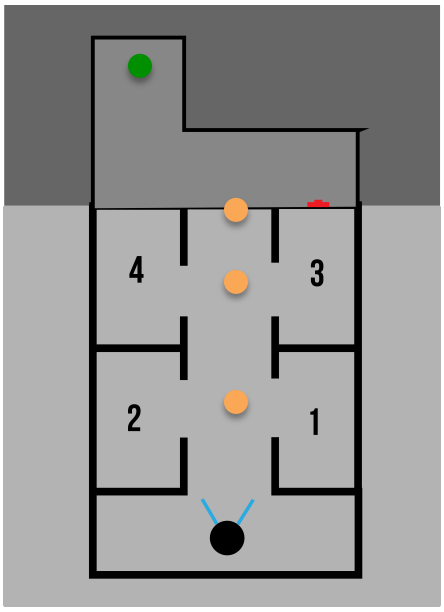
Une couverture est répartie aléatoirement dans une des 10 pièces du jeu. Il s'agit d'un objet complémentaire à l'échelle et permettant à Zuul de s'évader en couvrant les barbelés sur le mur. Son poids est de 5 unités.

3.2. Fixes

- Interrupteur

Un interrupteur se trouve dans un des emplacements de la prison et permet de désactiver l'alarme, en cas de déclenchement par le gardien. L'alarme est désactivée en utilisant la commande « Use ».

4. Les emplacements



Le jeu est formé de 10 emplacements, répartis en 3 catégories :

- La cellule de Zuul

Il s'agit du point de départ du jeu, le joueur peut uniquement sortir de la cellule (point noir du plan).

- Les cellules des prisonniers

Il y a quatre cellules de prisonniers, dans lesquelles peuvent se trouver les objets transportables, répartis aléatoirement.

- Les couloirs

5 emplacements se trouvent dans les couloirs du jeu.
Un permettant de pénétrer ou de sortir des deux premières cellules de prisonniers, ainsi qu'un emplacement pour les deux dernières. Un permettant de choisir la direction au bout du premier couloir (les trois carrés orange). Un représentant le point ou se trouve l'interrupteur de l'alarme (point rouge du plan). Le dernier représente le point de fuite de la prison (point vert du plan).

5. Conditions

Plusieurs conditions, réparties en catégories, sont à connaître et respecter:

<u>Générales</u>

- Afin de gagner la partie, Zuul doit déposer l'échelle et la couverture à l'emplacement de fuite.
- Si au point de fuite, Zuul essaie de s'évader uniquement avec l'échelle, la partie est perdue.
- Les objets transportables ont tous un certain poids et Zuul ne peut pas porter plus de 10 unités. En d'autres termes, l'échelle et la couverture, biens complémentaires pour gagner la partie, ne peuvent pas être transportés simultanément.
- Si Zuul utilise l'échelle pour grimper un mur, ne se trouvant pas dans le point de fuite, il sera téléporté aléatoirement dans une salle du jeu.

<u>Confrontation</u>

- A chaque déplacement, Zuul à une certaine probabilité de rencontrer le gardien, qui se déplace aléatoirement dans le jeu, tout en respectant le plan défini.
- La peau de banane permet d'éviter (automatiquement) une seule et unique confrontation avec le gardien, lors de la première rencontre.

- La confrontation est représentée par un quizz, générant des questions aléatoirement. Afin de pouvoir gagner le quizz, Zuul doit répondre correctement aux 3 questions posées. Si une seule des questions est fausse, l'alarme sera déclenchée par le gardien.
- De plus, lorsque l'alarme est déclenchée et que Zuul rencontre le gardien, la confrontation doit être gagnée. Le cas échéant, la partie sera perdue.

Alarme

- Pour désactiver l'alarme, Zuul doit atteindre l'emplacement dans lequel se trouve l'interrupteur et utiliser la commande « Use ».
- Si l'alarme est éteinte et que le joueur l'enclenche, les mêmes conditions doivent être respectées.

6. Liste et fonctionnalités des commandes

6.1. Liste

Go : Permet au joueur de se déplacer dans 4 directions (Nord, Sud, Est et Ouest), la direction doit être entrée comme deuxième mot.
Quit : En mentionnant se mot, le joueur peut quitter le jeu à tout moment.
Look : Le joueur obtient le nom de la pièce dans laquelle il se trouve.
Back : Le joueur retourne dans la pièce précédente, s'il y en a une.
Take : Permet au joueur de ramasser des objets transportables, répartis aléatoirement dans les pièces du jeu.
Drop : Permet au joueur de laisser des objets transportés dans la pièce où il se trouve.
Climb : Quand le joueur transporte l'échelle, il peut utiliser cette commande à tout moment. Deux scénarios peuvent arriver:
1) Le joueur est dans la bonne pièce et utiliser l'échelle pour s'évader.
2) Le joueur ne se trouve pas dans la bonne pièce et en grimpant l'échelle, il est transporté aléatoirement dans une pièce du jeu.
Help : Aide le joueur en lui présentant toute les commandes utilisables dans le jeu.
Use : Permet d'utiliser les objets fixes (l'interrupteur de l'alarme).
Score : Permet de visualiser les statistiques de toutes les parties jouées.

6.2. Fonctionnalités

Base de données

Nous avons ajouté une base de données intégrée à notre jeu, permettant de stocker :

- Les historiques des parties jouées (le nom, le nombre de mouvements et les points cumulés lors des réponses au quizz), afin de créer un classement des joueurs.
- Les questions et réponses du quizz

Login

A chaque début de partie, le joueur doit donner un nouveau nom à Zuul. Cela lui permet de comparer les scores obtenus avec ceux des parties précédentes. De plus, une condition vérifie que le joueur n'ait pas utilisé le même nom deux fois.

Extensions obligatoires

« Look » a été rajoutée, afin d'informer le joueur par rapport à la pièce dans laquelle il se trouve. Elle retourne donc la description de la pièce en question.

« Back » permet au joueur de retourner dans la dernière pièce dans laquelle il se trouvait. Elle peut être utilisée plusieurs fois de suite, jusqu'à qu'il n'y ait plus de pièce précédente.

Autre extension

« Score » permet de visualiser les statistiques de toutes les parties jouées. La base de données gère les historiques et l'appel de la méthode affiche ces derniers.

Gestions des objets

Un poids maximum de 10 unités définit le nombre d'objets que Zuul peut transporter en même temps. De plus, une répartition aléatoire de ses derniers, se fait à chaque début de partie. Quatre commandes spécifiques ont été créées, afin de les gérer :

« Take » permet de ramasser des objets transportables et de les placer dans l'inventaire (liste d'objets) de Zuul, si la limite maximum de poids n'est pas dépassée. Cette dernière se vérifie au moment où l'objet est ramassé, grâce à une condition vérifiant que la somme du poids des objets transportés et de celui que le joueur veut ramasser, ne dépasse pas le poids maximum transportable.
La commande vérifie également qu'il s'agisse bien d'un objet existant dans cette pièce, ainsi que d'un objet transportable.

« Drop » permet au joueur de déposer un objet transporté, dans n'importe quelle pièce du jeu. La commande vérifie également que l'objet existe.

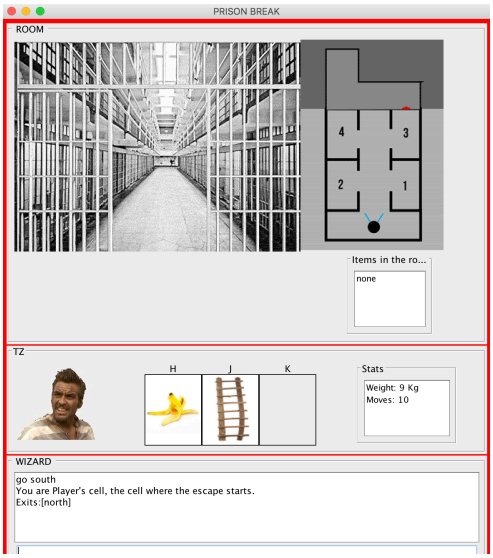
« Climb » est l'unique commande utilisant l'extension du parser à 3 mots. Elle permet au joueur de pouvoir grimper l'échelle, s'il la déposée dans la salle où il se trouve. Selon les conditions définies, le joueur peut soit utiliser l'échelle dans l'unique pièce d'évasion, dans ce cas là, en mentionnant les 3 mots-clés, il gagne la partie. Soit il peut utiliser l'échelle dans une autre pièce, dont l'action le téléportera aléatoirement dans une pièce du jeu. La notion de pièce magique est donc satisfaite à travers cette action.

« Use » permet d'utiliser les objets fixes, l'interrupteur de l'alarme dans le cas du jeu. Cela s'effectue en changeant son état.

Gestion des paramètres de jeu

La classe « GameParams » a été créée, afin de pouvoir gérer les paramètres de jeu. Le joueur peut définir le poids maximum transportable, ainsi que ceux des objets liés. De plus, le joueur peut choisir son mode de jeu, démo pour démontrer les fonctionnalités du jeu (les items sont initialisés dans des salles prédéfinies et le gardien est immobile, dans un emplacement choisi) ou full pour un déroulement normal (tout se génère aléatoirement).

Interface graphique



Cette dernière a été développée sur plusieurs aspects :
- Une carte du jeu a été rajoutée, afin que le joueur puisse connaître sa position, en tout temps.
- Un inventaire a également été crée, contenant les objets transportés par Zuul.
- Un cadrant rouge s'affiche autour de la fenêtre quand l'alarme est déclenchée.
- Une fenêtre apparaît à chaque début de partie, le joueur doit choisir le nom qu'il veut donner à Zuul, afin de pouvoir générer des statistiques de jeu.

Une gestion d'événements a également été rajoutée :
- Les touches H, J, K, permettent au joueur de « droper » des objets transportés.
- Les flèches du clavier peuvent être utilisées pour indiquer les directions à prendre.

7. Gestion des problèmes

Problèmes survenus

- A l'origine, nous sommes partis sur une base de données externe, afin de stocker les données des parties et du quizz. Nous nous sommes aperçus, qu'en testant le jeu avec l'émulateur d'Xcode, cela n'était pas compatible. De plus, il était nécessaire de connecter manuellement la base à chaque lancement de jeu… Par conséquents, nous nous sommes dirigés vers une base de données intégrée à notre programme, ce qui nous a permis d'améliorer sensiblement l'expérience utilisateur, ainsi que la compatibilité multiplateforme.

- Le fichier ressources/BD-PrisonBreak.sql a posé certains problèmes, notamment concernant l'encodage du texte des questions du quizz. Des caractères spéciaux apparaissaient à la place d'accents. L'encoder en UTF-8 a réglé le problème.

- La coordination entre les événements textuels et les raccourcis claviers a été problématique. Nous avons du la corriger plusieurs fois, afin de pouvoir garantir des actions similaires.

- La version du code fourni pour le projet n'était pas compatible avec la version actuelle de Netbeans. En effet, ce dernier n'était plus supporté par la version IOS9, ce qui empêchait l'exécution correcte de l'application. Voici la solution trouvée sur un forum :

Code déprécié :

```
[window addSubview:viewController.view];
```

Code corrigé :

```
[window setRootViewController:viewController];
```

Source : http://stackoverflow.com/questions/30884896/application-windows-are-expected-to-have-a-root-view-controller-at-the-end-of-a

- La manipulation de la bibliothèque swing a été un point délicat. En effet, la création et la modification d'interfaces graphiques demandent une certaine expérience, contenu du nombre conséquent d'objets à connaître et manipuler (Jframe, Jpanel…).

- Malgré l'efficacité du travail à travers le programme GitHub (une description détaillée est fournie dans le point concernant la répartition du travail), nous avons rencontré des erreurs de conflits, qui nous ont ralenti plusieurs fois dans le déroulement du projet.

Couplage et cohésion

- Nous avons développé indépendamment toute la base de données quizz, en respectant le modèle MVC (Model View Controller). Cela permet, en cas de besoin, de s'affranchir de cette dernière sans modification majeure du code.

- La gestion des objets a été définit, afin d'avoir le moins de couplage et le plus de cohésion possible. En effet, une classe abstraite « Item » a été créée et deux sous-classes abstraites « Fixed » et « Transportable » également, héritant de la première. Cela a permis de pouvoir définir précisément la spécification des classes d'objets transportables et fixes du jeu.

- Une classe abstraite « Person » a également été créée, dont la sous-classe « Guardian » hérite, afin de pouvoir rendre la spécification des personnages la plus indépendante possible. Même s'il y a finalement qu'une seule sous-classe « Guardian », cela a été anticipé, si le jeu venait à être développé davantage.

Traitement de comportements non-conformes des utilisateurs

Rien n'a été laissé au hasard, les cas d'utilisations ont été gérés, afin de ne pas laisser l'utilisateur dans une situation sans solution, voici quelques exemples :

- Si l'utilisateur essaie de fermer la fenêtre du quizz durant une partie, cette action sera impossible tant qu'il n a pas répondu aux 3 questions posées ou qu'il ne s'est pas volontairement rendu au garde, à l'aide du bouton « surrender ».

- Lors du démarrage du jeu, le joueur est obligé d'entrer un nom pour Zuul. De plus, il ne peut pas utiliser un nom déjà enregistré.

- Lors de l'utilisation de commandes, tous les cas possibles ont été imaginés, cela afin de ne pas tolérer d'exceptions.

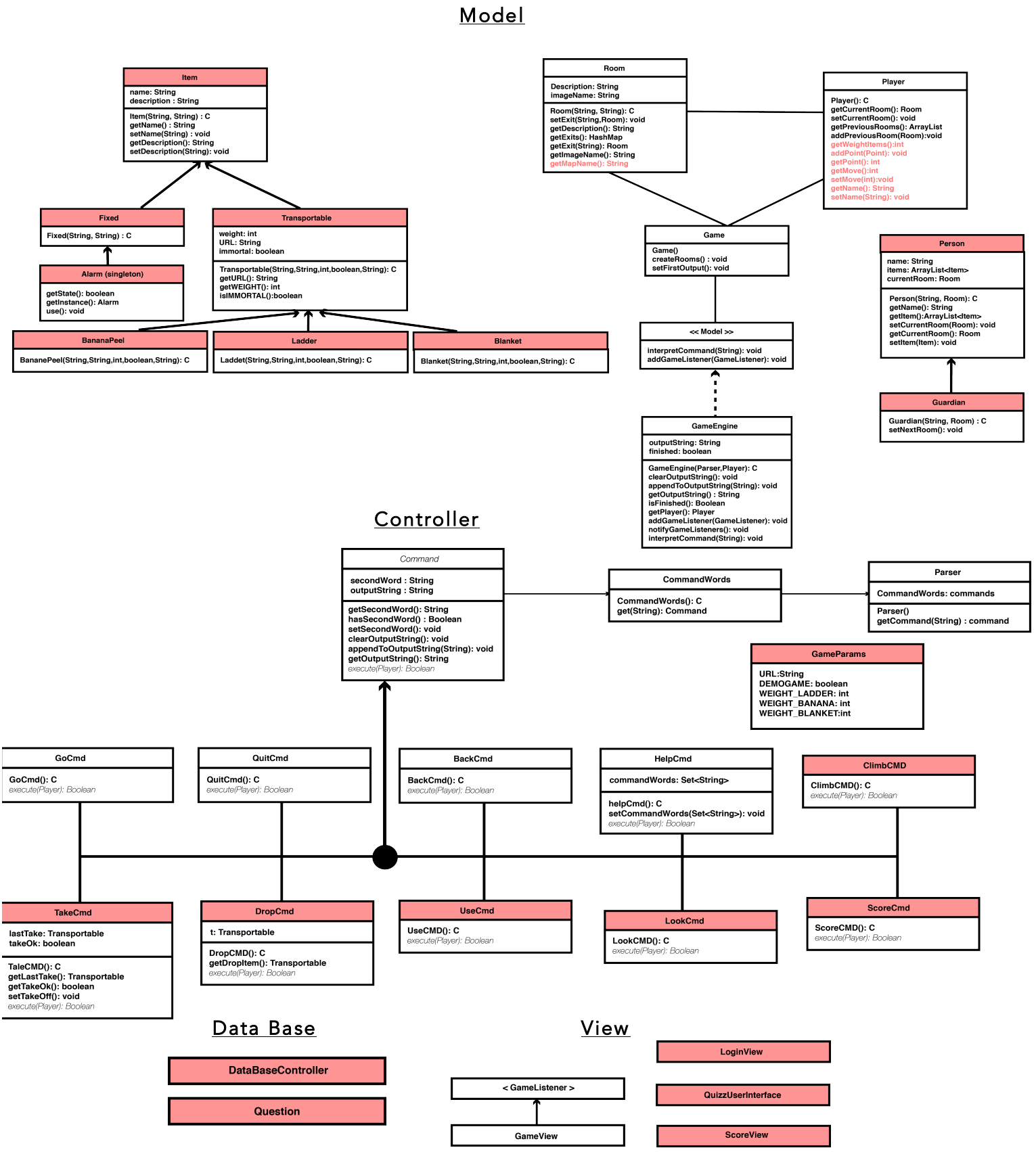
- Si le joueur tente de lancer deux versions du jeu simultanément, étant donné que la base de données ne peut être connectée qu'à un seul programme, un message d'erreur signale au joueur, qu'il ne peut pas effectuer cette action.

Bugs

Actuellement, aucun bug n'a été découvert malgré les nombreux tests effectués. Grâce à une gestion minutieuse des comportements non conformes et le fait que l'utilisateur soit guidé tout au long du jeu, ce dernier est stable.

8. Schéma de classes

Ci-dessous, le schéma de classes de base (en blanc) non-exhaustif, avec la présentation des classes ayant été développées ou créées (en rouge). Une explication du package Item suit le schéma.



Structure du package « Item »

Une classe abstraite mère « Item » a été créée dont les deux sous-classes « Transportable » et « Fixed » héritent. Chaque objet du jeu représente une classe fille d'une des sous-classes. Cette approche a été élaborée dans le but d'anticiper une éventuelle évolution du jeu, et de permettre ainsi à chaque classe d'objets créée, de posséder ses propres spécifications. La même démarche a été pensée pour la classe abstraite « Person ».

9. Répartition du travail

Afin de pouvoir coder simultanément sur le même projet, nous avons utilisé le programme de versionning Git, au travers de la plateforme GitHub.

GitHub
https://github.com

Cet outil nous a apporté plusieurs avantages[1] :

- Il nous a permis de tous travailler, indépendamment, sur le même projet. En effet, en utilisant le système de « branches », chacun a pu implémenter individuellement ses parties. Puis, régulièrement, nous fusionnions ces dernières pour actualiser la branche principale.
- Les algorithmes de fusions nous ont également permis de gérer les éventuels conflits, et cela avant d'actualiser la branche principale.
- GitHub nous a également permis d'effectuer une multitude de changements de dernière minute, qui n'auraient pas été facilement réalisable, si chacun travaillait sur un fichier source différent.
- Finalement, l'utilisation rigoureuse de cet outil, nous a permis d'être logique et efficient, dans le développement de notre jeu. Cependant, GitHub ne répond pas aux problématiques de répartitions des tâches. Cet aspect doit être défini au sein du groupe, pour pouvoir bénéficier des avantages de cet outil.

[1] http://blog.openclassrooms.com/blog/2009/05/27/avantages-et-defauts-de-git/

Voici la répartition des tâches, selon une liste non-exhaustive :

Tâches	Matthieu	Florent	Onur	Bryan
Développement de l'interface graphique		50%	50%	
Implémentation du package « Controller »	10%	40%	25%	25%
Implémentations des modèles	40%	25%	25%	10%
Création de la base de données		80%	20%	
Implémentation de Xcode			100%	
Implémentation des conditions de déroulement		70%	30%	
Gestion des évènements clavier			100%	
Elaboration du schéma de classes	100%			
Réalisation des images et du plan du jeu	100%			
Réalisation du visuel de la présentation	100%			
Gestion des textes du jeu				100%
Rédaction du rapport				100%
Réalisation du contenu de la présentation	25%	25%	25%	25%

10. Sources

- Images du jeu (modifiées par la suite) :

http://jouyanna.ch/IMG/arton1619.jpg
https://images.jacobinmag.com/2014/02/prison.jpg
https://upload.wikimedia.org/wikipedia/commons/7/77/Mur_de_prison.jpg
http://www.adirpdeparis.com/IMAGES2/archives/025.jpg
http://www.robertamsterdam.com/stena.jpg
http://i.imgur.com/gE37S.jpg
http://image.toutlecine.com/photos/o/0/b/o-brother-2000-08-g.jpg

- Gestion des problèmes rencontrés :

Les problèmes liés à l'implémentation du jeu ont été gérés à l'aide de deux forums :
https://openclassrooms.com/
http://stackoverflow.com/