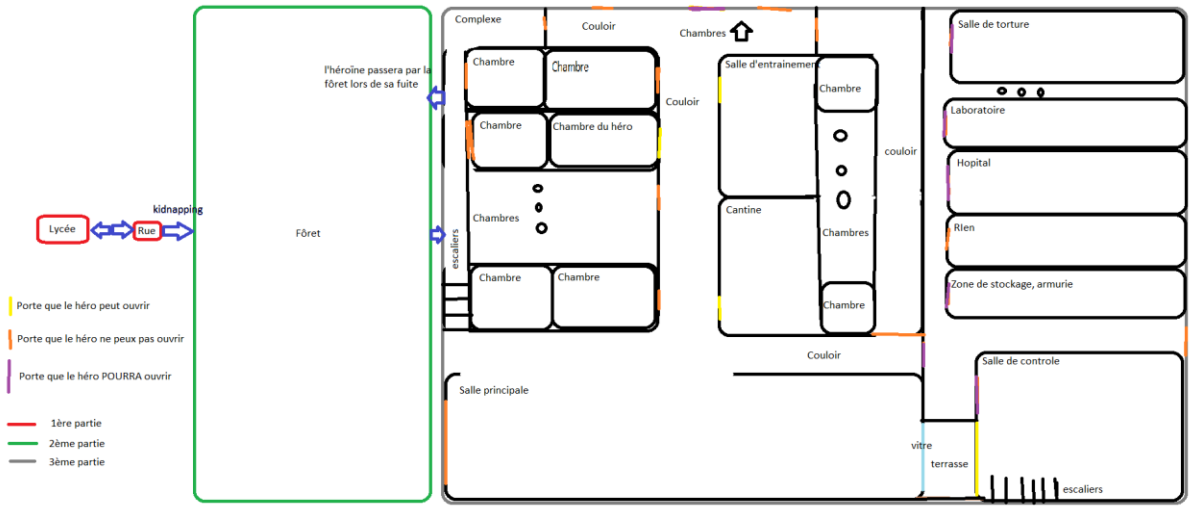
Rapport de projet

- Titre du jeu : Test
- Auteur : Jeanne Boulet
- Thème : Jeu d'aventure ou une fille doit survivre dans une arène puis échapper à ses ravisseurs.
- Résumé du scénario : Lili, une jeune fille de 16 ans se fait kidnapper à la sortie du lycée. Celle-ci se réveille au milieu de la forêt et devra faire face à de nombreux ennemis pour survivre. En frôlant la mort dans cette arène, elle est sauvé par des scientifiques dans un mystérieux complexe. Lili devra maintenant s'échapper de ce lieu.
- Plan :




- Scénario détaillé :

1ère partie : Le kidnapping

- Lili se fait kidnapper dans la rue en sortant du lycée

2ème partie : La forêt

- Réveil dans la forêt
- L'héroïne circule à travers une forêt pour s'échapper
- Elle doit y trouver une map pour pouvoir s'échapper

3ème partie : Le complexe

- Réveil dans une chambre du complexe
- Les ravisseurs exercent des expériences sur notre héroïne.
- Plusieurs choix pour s'échapper :
  - Ne pas trouver le moyen de s'échapper et mourir des expériences
  - Mourir de faim
  - Effectuer une fuite réussi en utilisant la manière forte (enfoncer la porte)
  - Effectuer une fuite réussi en utilisant une manière plus discrète (crocheter la porte)
  - Il est possible d'enquêter sur le pourquoi l'héroïne est ici en circulant dans les pièces
- Réponses aux exercices :
  - Exercice 7.5
    - Pour éviter la duplication de code on utilise une méthode distincte dont l'unique tâche est d'afficher les informations liées à la situation courante. On créée la procédure printLocationInfo().

- Exercice 7.6
  Ajout d'une hashmap pour stocké les sorties et modification de goRoom() et de printLocationInfo() .
  Problème : L'utilisation de champs public qui rend visible qu'elle possède des sorties mais aussi la manière dont elle est stockée.
  ⇨ Viol de l'encapsulation.
- Exercice 7.7
  Création de getExitString() qu'on utilise après dans printLocationInfo(). On évite ainsi de nouveaux couplages.
- Exercice 7.8
  Modification de la classe Room et ajout de la méthode setExits(). On facilite la création de room et on rajoute de nouvelle directions.
- Exercice 7.9
  Rajout de la méthode keySet() qui renvoie une vue d'ensemble des clé de la hashmap
- Exercice 7.10
  Modification de la méthode getExitString() qui permet une meilleur visibilité du code.
- Exercice 7.11
  Amélioration de la classe Room pour réduire le couplage avec la création de getLongDescription() et ainsi modification de la classe Game
- Exercice 7.14
  Ajout d'une nouvelle commande look()
- Exercice 7.15
  Ajout d'une nouvelle commande eat()
- Exercice 7.16
  Ajout d'une méthode showAll() dans Command Words rajoutant automatiquement les commandes dans printHelp().
- Exercice 7.18
  On encapsule toutes les informations de l'interface utilisateur au sein d'une classe.
  Modifications de showAll(). On la renomme en getCommandList().
  Rajout dans les classes de la méthode getCommandList().
- Exercice 7.18.6
  - Ajout de l'attribut aImage et ajout de la méthode getImageName()
  - On enlève Scanner au profit de StringTokenizer
  - Création de la classe GameEngine
  - Transfert de la majeur partie du code de Game dans Game Engine
  - Création de la classe UserInterface
  - Intégration de la nouvelle organisation du jeu
  - Rajout des imports un à un.
- Exercice 7.18.8
  Ajout d'un bouton dans la classe UserInterface. Plus précisément dans la procédure createGui() à l'aide de JPanel. Le bouton renvoi la méthode look(). Il est possible de faire la même chose pour toute les autres commandes ! Je prends la décisions pour l'instant de ne faire qu'un bouton. Il est aussi possible de redimensionner le bouton et de le placer comme bon nous semble avec GridLayout. Et il est aussi possible de mettre une image sur le bouton avec la classe Icon.
- Exercice 7.19.2
  - Quelques images sont déplacées dans un dossier « Images » à la racine du jeu
  - Modifications de la méthode showImage()

- Exercice 7.20
  - Ajout d'une classe Item de 2 attributs (le nom et le poids), d'un constructeur naturel et de getters.
  - Ajout d'un item « Bag » dans la room vMiddleForest.
- Exercice 7.21
  Ajout de la méthode getImageDescription() dans la classe Room
- Exercice 7.22
  - Ajout d'une hashmap dans Room pour stocker les items
  - Ajout de la méthode addItems() pour ajouter des items dans la même classe
- Exercice 7.23
  - Création d'une hashmap (attribut aPreviousRoom dans la classe GameEngine)
  - Création d'une méthode goBack()
  - Rajout de la commande « Back » dans InterpretCommand et dans la classe CommandWords
- Exercice 7.26.1
  Génération des 2 javadocs
- Exercice 7.28.1
  - Création d'une méthode test dans la classe GameEngine
  - Ajout de l'import java.util.Scanner dans la classe GamEngine
- Exercice 7.28.2
  - Création des deux fichers court.txt et long.txt
- Exercice 7.29
  - Création de la classe Player, on y met les attributs aCurrentRoom et aPreviousRoom initialement dans la classe GameEngine, un constructeur, un getter et un setter pour aCurrentRoom, une méthode popPreviousRoom(), la méthode addPreviousRoom()
  - On y met le code des commandes
  - Enfin, on effectue tous les changements (rajout d'un attribut aPlayer dans game engine et modifications des codes)
- Exercice 7.30
  Ajout d'un attribut Item aItem dans la classe GameEngine et d'une commande take() et drop() dans la même classe qui permet de porter un item et de le poser
- Exercice 7.31
  On remplace l'attribut aItem par un attribut HashMap aInventory. On peut désormais porter plusieurs items. On rajoute un item Item1 dans la première pièce du jeu qui nous permet de tester les commandes.
- Exercice 7.31.1
  Création d'une classe Itemlist, on y met un attribut HashMap aItems avec un constructeur, un getter Item, un getter Hashmap, une procédure removeItem(), une procédure addItem() et une méthode getItemListDescription() pour avoir la liste des items.
- Exercice 7.32
  Rajout d'un attribut int aMaxWeight, modification de la méthode take() pour que le joueur ne puisse pas prendre un objet si celui-ci est trop lourd.
- Exercice 7.33
  Rajout d'une commande « inventory » qui renvoie l'inventaire.
- Exercice 7.34

On ajoute une commande equip() qui permet à l'item « Bag » déjà présent dans la salle vMiddleForest de pouvoir porter des charges 2 fois plus lourdes.

- Exercice 7.34.1
  Mise à jour des fichiers test.
- Exercice 7.34.2
  Re-génération des 2 javadoc
- Exercice 7.42
  Création d'un attribut aTime dans la classe GameEngine ainsi qu'un attribut aKeyRoom. Modifications des méthodes goBack(), goRoom(), eat(), take() et drop().Un compteur d'action se lance dès que l'on rentre dans la salle vMiddleForest. Rajout 'un attribut aDay et d'une méthode passedDays() ;
- Exercice 7.42.2
  Je prends la décision de remettre cet exercice à plus tard.
- Exercice 7.43
  Modifications de la méthode goRoom() pour que la aPreviousRoom soit clear dès qu'on rentre dans une trap door.
- Exercice 7.44
  Ajout d'une méthode charge() et décharge(). Ajout d'une commande « note » et d'une commande «  to note ». Il faut avoir de quoi noter pour pouvoir effectuer la « téléportation »
- Exercice 7.45
  Ajout d'une classe Door, on y met dedans un attribut aTraped et aLocked des boolean. Un constructeur naturel, des setters, des getters. On peut maintenant ajouter des portes ou piégées ou fermés.
- Travail personnel
  Ajout de jours, c'est un attribut aDay qui lorsque le temps arrive a 0 passe au jour suivant.
  Ajout d'une « barre de nourriture », c'est un attribut aFood. Il faut désoramais se nourrir régulièrement sinon on meurt.
  Rajout des statistiques. Elles pourront être améliorer ou régresser selon le style de jeu. Et elles auront un impact sur le déroulement de l'histoire. Cet exercice n'est qu'au stade de test. Rajout d'une classe Stats, j'y mets un attributs de force, d'intelligence et de dextérité. Des setters, des getters. Un constructeur non naturel. Il serait possible d'en faire un naturel et de laisser choisir le joueur ces stats de commencement (Attribuer un certain nombre de points a mettre dans les 3 catégories par exemple). Ou même de faire plusieurs configurations et de laisser choisir le joueur laquelle il préfère. Je choisis d'attribuer moi-même les statistiques de base pour mon scénario.

Déclaration d'anti-plagiat :

Je soussigné(e) Jeanne Boulet

- déclare que ce mémoire est un document original fruit d'un travail personnel ;

- suis au fait que la loi sanctionne sévèrement la pratique qui consiste à prétendre être l'auteur d'un travail écrit par une autre personne ;

- atteste que les citations d'auteurs apparaissent entre guillemets dans le corps du mémoire ;

- atteste que les sources ayant servi à élaborer mon travail de réflexion et de rédaction sont référencées de manière exhaustive et claire dans la bibliographie figurant à la fin du mémoire ;

- déclare avoir obtenu les autorisations nécessaires pour la reproduction d'images, d'extraits, figures ou tableaux empruntés à d'autres œuvres