

Rapport projet IPO :

Titre : Alerte au pied de l'arc en ciel !

lien pour la page web : Nous n'y avons pas encore accès. Mais voici un aperçu :

Alerte au pied de l'arc en ciel


Auteur : Manon HERMANN

JEU D'AVENTURE

THEME

SCENARIO

PLAN



Theme :

Dans une forêt magique, un farfadet doit retrouver les pièces volées au pied de l'arc en ciel.

Resume du scenario complet

<h4>Scenario</h4> <p>Dans une forêt, un farfadet est appelé d'urgence au pied d'un arc en ciel car des pièces d'or du chaudron ont été volées ! Il devra alors traverser la forêt pour tenter de retrouver 5 pièces d'or perdu. Au cours de son chemin il rencontrera des personnages qui l'aideront en échange d'un autre objet... Mais lorsqu'il aura enfin trouvé toutes les pièces il se rendra compte qu'il lui manque la clé pour refermer le chaudron : quelqu'un l'a volé ! Il faut retrouver cette personne.</p>	<h4>Liste des personnages</h4> <p>Fée, Lutin, Homme Arbre, Elfe, Sorcière</p>
<h4>Liste des objets</h4> <p>5 pièces d'or, la clé du chaudron, 2 champignons, 1 verre d'eau, 1 livre magique</p>	<h4>Informations en plus</h4> <p>Situation gagnante : récupérer les 5 pièces d'or (dans le temps imparti) et répondre juste à l'énigme</p> <p>Situation perdante : ne pas réussir à récupérer les pièces à temps et/ou répondre faux à l'énigme</p> <p>Enigme posée par la sorcière (dans la salle secrète) : "Combien de personnages avez vous rencontré ?"</p>

© Unlited. All rights reserved. | Design: HTML5 UP | Modified by Manon Hermann

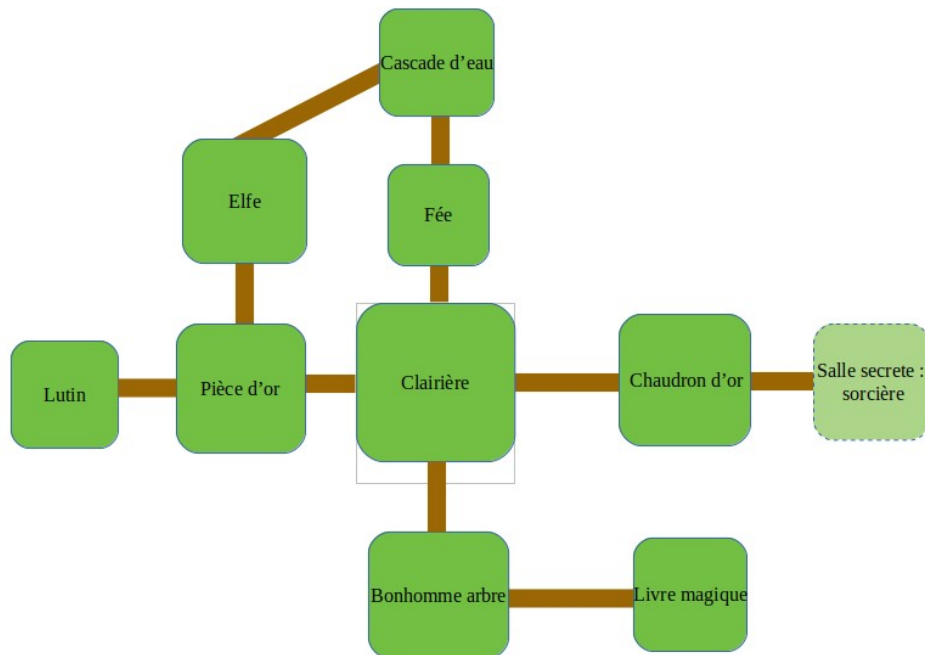
I. A. Auteur :
Manon HERMANN

B. Thème :
Dans une forêt magique, un farfadet doit retrouver les pièces volées au pied de l'arc en ciel

C. Résumé du scénario (complet) :

Dans une forêt, un farfadet est appelé d'urgence au pied d'un arc en ciel car des pièces d'or du chaudron ont été volées ! Il devra alors traverser la forêt pour tenter de retrouver 5 pièces d'or perdu. Au cours de son chemin il rencontrera des personnages qui l'aideront en échange d'un autre objet... Mais lorsqu'il aura enfin trouvé toutes les pièces il se rendra compte qu'il lui manque la clé pour refermer le chaudron : quelqu'un l'a volé ! Il faut retrouver cette personne.

D. Plan complet :



E. Scénario détaillé :

Vous incarnez un farfadet qui doit retrouver les pièces d'or qui ont disparus dans le chaudron au pied d'un arc en ciel. Elles sont dispersées dans la forêt. La 1^{ère} vous est donnée par la fée après un court dialogue. La 2^{sd} vous est donnée par l'elfe une fois que vous aurez retrouver son livre magique placer dans une autre salle. La 3^{ème} est posé par terre dans une salle, il suffit de la ramasser. La 4^{ème} vous est donnée par le lutin une fois que vous lui serez aller chercher deux champignons disponible dans une autre salle. La 5^{ème} vous est donnée par l'arbre vivant une fois lui être chercher un verre d'eau trouvable dans une autre salle.

Une fois ces 5 pièces récoltées il faudra retourner au chaudron d'or pour les déposées. C'est à ce moment que vous vous rendez compte que la clé pour fermer ce chaudron a aussi disparu. Une salle se débloque à gauche du chaudron. C'est dans cette salle que se trouve une sorcière détenant la clé. Elle vous posera une question, si la réponse est juste alors vous gagner le jeu, sinon vous perdez.

F. Détails des lieux, items, personnages :

Chaque lieu contient un personnage avec lequel le joueur peut interagir ou un objet qu'il peut ramasser.

Liste des personnages : Fée, Lutin, Homme Arbre, Elfe, Sorcière

Liste des items : 5 pièces d'or, la clé du chaudron, 2 champignons, 1 verre d'eau, 1 livre magique

Liste des lieux : - « Chaudron d'or » où le joueur commence le jeu, c'est ici qu'il devrait rapporté les 5 pièces d'or ;

- « Sorcières » c'est la salle secrète du jeu qui se déverrouille lorsque le joueur à fini de rapporter toutes le pièces au chaudron ;

- « Clairière » on peut y trouver des champignons ;

- « Fée » on peut parler avec une fée qui nous donne une pièce d'or ;

- « Cascade d'eau » on peut y récolter de l'eau ;

- « Elfe » on peut parler avec un elfe qui nous demande de retrouver son livre magique contre une pièce d'or ;

- « Pièce d'or » c'est l'endroit ou on peut ramasser une pièce d'or tombé a terre ;

- « Lutin » on peut parler avec un lutin qui nous demande de lui trouver 2 champignons en échange d'une pièce d'or ;

- « Bonhomme arbre » on peut parler avec un arbre vivant qui manque d'eau, pour vous remercier de l'eau que vous lui donnerez il vous offrira une pièce d'or ;

- « Livre magique » c'est l'endroit où vous pouvez ramasser le livre magique de l'elfe ;

G. Situations gagnantes et perdantes :

Il est possible de gagner lorsque les 5 pièces d'or sont récupérées et ramenées au chaudron d'or (dans le temps imparti) et que le joueur saura répondre juste a l'énigme

Il est possible de perdre si le joueur ne récupère pas toute les pièces dans le temps imparti, ou s'il n'a pas le temps de les ramener au chaudron d'or, ou s'il ne répond pas correctement a l'énigme

H. Éventuellement énigmes, mini-jeux, combats :

Une énigme sera posé par la sorcière pour qu'elle veuille bien vous rendre la clé du chaudron d'or :

« Combien de personnages avez-vous rencontrer dans ce jeu ? »

La bonne réponse est « 4 ».

II. Réponses aux exercices :

Exercice 7,0 (optionnel) : Pour la création de la page web du projet, j'ai créé un dossier `public_html` dans lequel il y a un document `.html`. Pour faire un site web avec une belle mise en forme, j'ai choisi de prendre un template sur internet (cf référence dans les sources). Ayant quelque base en langage HTML et CSS, j'ai pu modifier certains paramètres pour qu'il soit à l'image de ce que je voulais représenter.

Exercice 7,4 : Les 5 classes des TP 3,1 et 3,2 ont été copiées dans un nouveau projet BlueJ. Les tests relatifs aux différentes classes ont disparus, et celles-ci sont maintenant à la racine du projet.

Exercice 7,5 : Dans cet exercice on souhaite éviter la duplication de code notamment pour les méthodes `printWelcome` et `goRoom` (classe `Game`) qui contiennent chacune un bout de code permettant d'afficher les sorties valides de la pièce où se trouve le joueur. Pour cela, on crée dans la classe `Game` une méthode `printLocationInfo` qui reprend exactement ce que faisaient ces dernières méthodes en double. Puis dans `printWelcome` et `goRoom` on supprime cette duplication pour appeler cette nouvelle méthode.

Exercice 7,6 : Le jeu fonctionne correctement, mais il y a du couplage de code notamment pour définir les sorties des pièces courantes. Nous allons donc créer un accesseur `getExit` dans `Room` qui permettra d'obtenir directement ces informations. Donc il faut remplacer les morceaux de code en appelant l'accesseur comme par exemple `<nextRoom = currentRoom.getExit(« East »);>`. Ainsi de grosse partie de code peuvent être remplacée par une seule ligne de code.

Exercice 7,7 : Par la suite nous créerons une méthode `getExitString` dans `Room` qui affichera les sorties possibles de la pièce courante. Il faudra donc l'appeler dans `printLocationInfo` de la classe `Game` pour qu'à chaque entrée dans une nouvelle pièce le joueur sache quelles sorties sont possibles.

Exercice 7,8 : Concept de la `HashMap` pour simplifier les déplacements directionnels et réduire les erreurs de codage. Il faut tout d'abord importer (grâce à la java doc) la `HashMap`. Il faut ensuite l'initialiser avec les attributs ainsi que dans le constructeur. Mais cette création de `HashMap` entraîne une modification dans d'autres méthodes de `Game` et `Room`. Ainsi le `setExit` créé précédemment peut-être résumé en une ligne grâce aux fonctionnalités de la `HashMap` `<exits.put(pDirection, pRoom);>`.

Exercice 7,8,1 : Ajout d'un déplacement vertical dans le jeu. Dans mon cas j'ai choisi la possibilité d'accéder à la pièce *Lutin* en allant vers le haut.

Exercice 7,9 : Ici il va falloir redéfinir les pièces (les instancier de nouveau) car la `HashMap` facilite leur définition. Il faudra aussi modifier `getExitString` en se servant des « key » de la `HashMap`. Ainsi on utilisera la méthode `keySet` pour aller chercher les sorties associées à la pièce courante.

Exercice 7,10 (optionnel) : La méthode `getExitString` de la classe `Room` a été créée pour éviter la duplication de code dans la classe `Game`. Elle renvoie les sorties possibles de la pièce où le joueur se trouve. Elle fonctionne grâce aux propriétés de la `HashMap`, c'est-à-dire qu'on va lui envoyer la position du joueur (la pièce associée) pour laquelle renvoie la clé associée permettant ainsi d'aller voir à cet emplacement. On y trouvera les sorties reliées.

Exercice 7,10,1 : Mise à jour de la javadoc pour chaque classe et méthode de celle-ci. On y indique à quoi sert la classe/méthode, puis pour chaque méthode quel(s) paramètre(s) il faut lui assigner ainsi que ce qu'elle retourne.

Exercice 7,10,2 : Génération et visualisation de la javadoc grâce a BlueJ, elle comprend bien tout ce qui a été renseigné.

Exercice 7,11: Nous allons maintenant anticiper les prochains TP en considérant que les pièces auront des items et/ou personnes. Il faudra donc le dire dans la description (lorsqu'on arrive dans la pièce), pour cela on crée une méthode *getLongDescription* dans la classe *Room* qui ajoute ces détails. On va donc appeler cette méthode dans la méthode *printLocationInfo* de la classe *Game*.

Exercice 7,12 (optionnel) : ...en cours...

Exercice 7,13 (optionnel) :

Exercice 7,14 : On veut créer une nouvelle fonctionnalité dans le jeu, l'acteur de regarder. Pour cela il faut tout d'abord mettre à jour le tableau constant de *CommandWords* qui répertorie toutes les commandes valides du jeu. Ensuite, il faut logiquement la créer dans *Game*, elle renverra la méthode *getLongDescription*. Et la rajouter dans le répertoire des commandes de la méthode *processCommand*.

Exercice 7,15 : De même, on veut faire une commande pour manger. Il faut procéder de même que l'exercice précédent mais avec une nouvelle action *eat*.

Exercice 7,16 : On remarque *printHelp* n'est pas à jour vis à vis des nouvelles commandes possibles, on pourra les rajouter à la main mais il faudrait le faire à chaque fois, et cela induit plusieurs erreurs possible. On va donc créer une méthode *showAll* dans la classe *CommandWord* qui affiche les commandes valides (à partir du tableau constant). Or cette méthode n'est pas directement accessible depuis la classe *Game*, il faut donc créer une méthode intermédiaire *showCommands* dans la classe *Parser*.

Exercice 7,17 (optionnel) : Pour ajouter une nouvelle commande, il faut forcément changer la classe *Game*. En effet, il faut la définir dans cette classe et ajouter sa référence dans la méthode *processCommand*.

Exercice 7,18 : Pour mieux organiser l'implantation du code et rétablir la conception de base de certaines méthode. Nous allons modifier la méthode *showAll* en *getCommandWord* pour quelle renvoie une String et ne s'occupe pas de la renvoyer. Ainsi ça sera *printHelp* de *Game* qui se chargera de l'afficher. Il faut donc en conséquence modifier la méthode intermédiaire de la classe *Parser* en la faisant renvoyer une String.

III. Mode d'emploi (instructions, comment démarrer le jeu, ...) :

Pour démarrer le jeu, il faut créer un objet jeu en faisant un clic droit sur la classe *Game*, puis lancer la méthode *Play()*. Une fenêtre du jeu va alors s'ouvrir.

Les commandes de base pour jouer a ce projet Zuul sont :

- *go* + la direction (en anglais) pour se déplacer
- *help* pour obtenir les commandes valides
- *quit* pour quitter le jeu (ne sauvegarde pas la progression du jeu en cours)
- *look* pour avoir la description de la pièce (objet, personnages et sorties)
- *eat* pour manger

IV. Sources :

template du site internet : <https://html5up.net/>