EP60.92

Projet d'application pluridisciplinaire

La chasse aux trésors

2011-2012

I. Objectifs

- Mettre en œuvre les compétences acquises ou en cours d'acquisition en:
  - Modélisation UML, Réseau, Base de données, Programmation Orientée Objet, Web, Architecture répartie, Programmation concurrentielle.
- Travailler en groupe (3 ou 4 personnes) sur un projet de grande taille
- Faire évoluer un programme développer par d'autres personnes (avec ses bugs, ses manques de commentaires, ses erreurs de conception et de développement).

II. Contexte technique

- Modélisation UML, outils disponibles : Fujaba, Umbrello, Objecteering, Rational Rose
- Langage Orienté Objet : Java (dont API Swing, Net, Thread, …)
- Base de données : SQL, SGBD (Postgres), JDBC
- Web : HTML, Javascript, php
- Rédaction de la documentation en anglais

III. Conseils de développement

- Utilisez Eclipse et Java 1.5 ou 1.6
- Commentaires : entête des classes et des méthodes en javadoc, commentaires pertinents dans le code
- Respect des règles Java et UML pour la nomination des attributs et des méthodes (exemple : uneMethode(unParamètre), unAttribut)
- Regroupement des méthodes par type (constructeur, modifieur, accesseur)
- Noter ses interventions dans le code en encadrant les ajouts/modifications/suppression par des /* GroupeX 2011 : debut */ … /* GroupeX 2011 : fin */. Pour les suppressions mettre les lignes en commentaire sans les supprimer des fichiers.

IV. Les grandes étapes du projet

1. 16/02/2012
   - Découverte du projet
   - Description générale du code
   - Identification de quelques bugs
   - Récupération de la version précédente du projet et de l'éditeur de niveau
2. 16/02/2012 → 23/02/2012
   - Création des groupes
   - Choix d'une correction à effectuer sur le code.
3. 23/02/2012 → 16/03/2012
   - Découverte et compréhension du code. Production du diagramme de classe complet du projet et du diagramme de séquence principal.

- Correction du code
- Réflexion sur une proposition d'amélioration du jeu

4. 16/03/2012
   - Restitution et test des corrections effectuées
   - Proposition d'amélioration et validation des propositions
5. 16/03/2012→05/04/2012
   - Répartition précise des tâches au sein de chaque groupe. Planning. Plan de validation. Plan d'intégration individuel et global
   - Document de conception et planning à remettre (en anglais)
   - Début du développement des nouvelles fonctionnalités
6. 05/04/2012→10/05/2012
   - Développement
7. 10/05/2012
   - Test et validation
8. 01/06/2012
   - Remise du rapport final (en anglais) : avancement du code par rapport à la conception, difficultés rencontrées.
   - Remise des codes sources

V. Description de la version précédente

Principe général du jeu

Les joueurs : ils sont répartis en n équipes, chaque équipe étant composée de plusieurs joueurs (4 pour le moment) : le maître des clés, l'explorateur, le sorcier, le piégeur

Chaque joueur doit découvrir à l'aide d'indices un des éléments nécessaire pour l'ouverture du trésor.

- *L'explorateur doit découvrir l'emplacement du trésor.*
- *Le sorcier doit découvrir l'incantation permettant au trésor de sortir.*
- *Le maître des clés doit découvrir la clé permettant l'ouverture du coffre.*

Chaque élément ou indice (sauf le premier) est invisible sur la carte, pour le révéler, il faut se placer sur la case où il se situe et taper la combinaison de touches appropriée :

- *Pour révéler un indice : « ctrl + numéro de l'indice »*
- *Pour révéler un élément : « ctrl + e ».*

Sur la carte, il n'y a qu'une seule incantation et qu'un seul trésor, mais deux clés permettant de l'ouvrir. Lorsque l'incantation ou le trésor sont révélés, leur emplacement est visible par tout le monde mais seul le sorcier peut lire l'incantation et le maître des clés récupérer la clé.

Pour ouvrir le coffre, les trois joueurs doivent se trouver à l'emplacement du trésor, le maître des clés doit y déposer la clé et le sorcier réciter son incantation.

La carte

La carte aux trésors est un damier pouvant avoir n'importe quelle taille. La taille de la carte peut éventuellement dépasser la taille de l'écran.

Sur chacune des cases se trouvent, soit le fond de carte, soit un élément particulier, franchissable ou non.

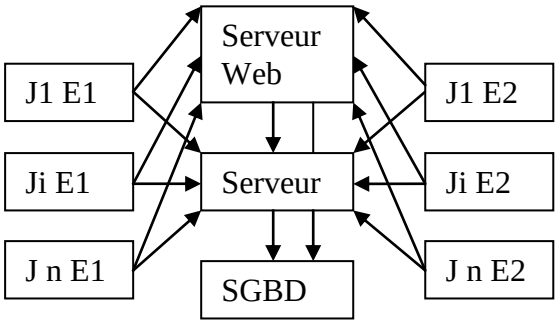
Les éléments peuvent être des éléments décoratifs ou liés aux indices (exemples : un volcan, une rivière, un pont, une plage, un village, des rochers, ….).

Architecture de l'application

- *Le jeu fonctionne en local ou en réseau (un poste par joueur).*
- *L'inscription des joueurs à une partie ainsi que la visualisation des scores se faire par le Web via un site Internet (version non mise à jour, à faire évoluer)*
  - *L'accès est sécurisé.*
  - *Après authentification, un joueur peut télécharger l'application, consulter la liste des autres joueurs, s'inscrire à une partie et visualiser les résultats.*
- *L'ensemble des informations concernant les joueurs et les parties sont stockées dans une base de données.*

La modélisation obtenue est la suivante

Architecture : client serveur ***C ⟶ S***



Fonctionnement en mode connecté
Communication entre

- *le serveur Web et le serveur : socket*
- *le serveur Web et le SGBD : JDBC*
- *le serveur et le SGBD : JDBC*

Fonctionnement du serveur : un processus (thread) créé par partie, les 2n joueurs sont gérés par un seul thread, fonctionnement synchrone des joueurs par rapport au serveur. On envoie régulièrement des données des joueurs vers le serveur, même si aucune action n'a été effectuée. Si plusieurs actions ont été effectuées entre deux envois, il faut envoyer la liste de ces actions (Si on réalise plusieurs envois par secondes, une seule action au plus a pu être effectuée pendant ce lapse de temps).
Une autre solution aurait été un fonctionnement asynchrone avec un ou plusieurs (un par joueur par exemple) threads par partie côté serveur. Dans ce cas, les joueurs n'envoient des informations au serveur qu'après une action. Le serveur, ainsi que les joueurs, doivent donc pouvoir recevoir et traiter des informations à n'importe quel moment. Les envois peuvent également être fais à la demande du serveur.
Ce n'est pas cette dernière solution qui a été retenue.
La première solution (celle envisagée) a l'avantage de faciliter le développement et la synchronisation des parties de l'application répartie (synchronisation automatique). Par contre, si un joueur ne fait pas d'actions entre deux envois, on enverra plusieurs fois la même information. Il faudra donc veiller à ne pas envoyer de données trop volumineuses.
Rôle de chaque partie de l'application et données circulant sur le réseau
Plusieurs solutions sont possibles

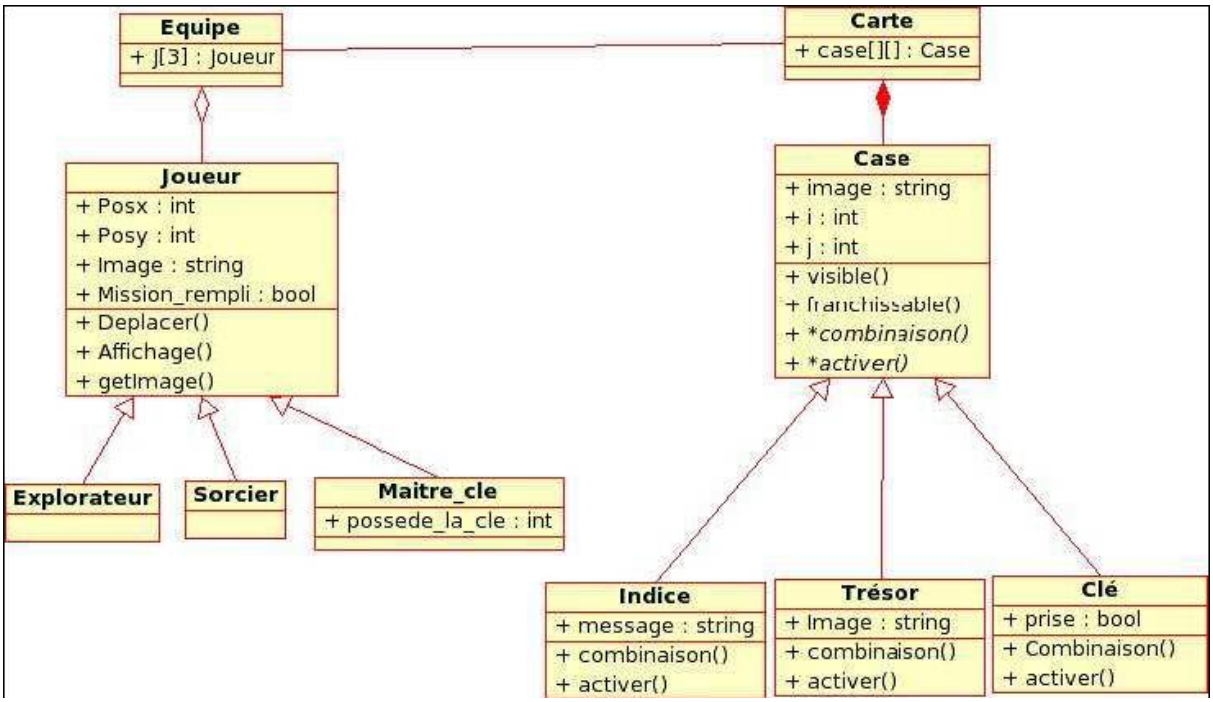
1. *Le serveur sert à centraliser les traitements : chaque joueur envoie la combinaison de touches qu'il a effectuées depuis le dernier envoi (déplacement, activation de cases, …). Le serveur interprète les actions et calcule la nouvelle carte (nouvelle position des joueurs et cases modifiées) puis renvoie la carte à chaque joueur qui ne fait qu'afficher la nouvelle carte. Avec ce protocole, le client n'a pas grand-chose à faire, le serveur est surchargé et le réseau aussi.*
2. *Le serveur sert de relai : chaque joueur envoie sa position courante ainsi que la liste des cases qu'il a activées depuis le dernier envoi. Le serveur retransmet à tous les joueurs ces informations et chaque joueur réinterprète les touches pour recalculer sa nouvelle carte. Dans ce cas, le serveur est allégé, le réseau également mais le client a plus de calculs à effectuer et chaque calcul est effectué six fois.*

Nous avons adopté la deuxième solution avec en plus côté serveur la gestion des monstres qui est commune à tous les joueurs.

Le serveur est le seul à connaître les affectations des joueurs dans les équipes ainsi que la carte de la chasse au trésor.
C'est lui qui charge la carte à partir de la base de donnée et qui la transmet aux six joueurs. Les images ne seront donc transférées qu'une seule fois.

Le jeu se termine lorsqu'une équipe a révélé et pris le trésor (les trois joueurs sont sur la même case et les combinaisons ont été tapées). Seul le serveur peut faire cette vérification et terminer le jeu en l'indiquant à chaque joueur.

<u>Modélisation des éléments</u> (diagramme de classe)
Une carte est modélisée par un damier de n lignes et m colonnes.
Un joueur a une position dans ce damier.
Le damier est composé de cases et chaque case a des fonctionnalités différentes (objet actif ou inactif, franchissable ou infranchissable, visible ou non visible).
L'affichage de la carte peut se faire soit en un bloc soit en découpant la carte en n fois m cases. La différence principale est la manipulation des fonctions d'affichage (découpage d'une image, ..). Nous choisirons d'afficher la carte en un bloc et d'afficher par-dessus les éléments éventuellement actifs (personnages, indices, …).
Le diagramme de classe est présenté ci-dessous.



<u>Sécurité</u> : l'accès au jeu est sécurisé. Lorsque les joueurs veulent participer à une chasse au trésor, ils doivent connecter leur client au serveur et lui communiquer leur pseudo et mot de passe. Le serveur vérifie (via le SGBD) les identifiants et intègre les joueurs dans les différentes parties (informations récupérées via le SGBD).

<u>La partie interface Web</u> : cette partie a pour but de permettre aux joueurs de jouer ensemble et de consulter les résultats. Deux grandes parties sont présentes

1. *Une partie publique où l'on trouve les informations sur le jeu (principe, téléchargement du client) et l'inscription (définition d'un pseudo, d'un mot de passe, ...).*
2. *Une partie privée dans laquelle on trouve, après inscription et authentification, la liste des chasses aux trésors à venir et passées avec la possibilité de s'inscrire pour une chasse ou de consulter les résultats (d'une chasse, les scores des joueurs, ...). On peut s'inscrire selon deux modes : par joueur ou individuel. Le jeu individuel : un joueur demande à participer à une partie en tant que sorcier, maitre des clés ou explorateur, ou alors laisse le choix au système (dépend de la chasse proposée). Le jeu par équipe : les joueurs forment des équipes et inscrivent leur équipe complète pour une chasse aux trésors. Dans cette partie, les joueurs peuvent aussi déposer des chasses aux trésors éditées grâce à l'éditeur de niveau téléchargeable également.*

   <u>Les diagrammes d'échange</u>

<u>L'éditeur de niveaux</u>

Il permet d'éditer les cartes utilisées dans le jeu. La notice d'utilisation de l'éditeur est fournie en annexe.

VI. Corrections et améliorations (quelques propositions)

- Gestion de la base de données : récupération des parties, stockage des résultats.
- Gestion des envois des listes de cases modifiées.
- Amélioration de la gestion de la fin de la partie
- Problème de duplication des transports
- Gestion des monstres : vers quel personnage le monstre se dirige t il ?
- Ajout de personnages
- Ajout de pièges (tourbillon, …)
- Ajout d'objets (téléporteurs, …)
- Ajout de nouvelles cases
- Revoir la gestion des points d'attaque et de défense, notamment lorsque le joueur utilise un moyen de transport
- Gestion du magasin (revoir la collecte des rubis et l'achat d'objets).
- Réactivation de la zone de visibilité
  - Autour du joueur
  - Sur tout le parcours du joueur
  - En fonction de divers paramètres (potions, …)
  - On peut voir la carte mais pas les éléments actifs
- Réactivation du marcher/courir

Concernant l'éditeur :

- Ajout des interfaces permettant de configurer un personnage (ou un monstre), une case, un élément, un transport

Concernant le site web : il faut revoir la base de données et les scripts php pour avoir un fonctionnement cohérent par rapport à la classe Partie. La notion d'équipe privée doit être formalisée avec notamment la gestion d'un mot de passe. Revoir le tchat.

VII. Notation

- Travail par groupe de 3 (4) personnes
- Chaque groupe se verra attribuer un nombre de points entre 0 et 60 (80) en fonction du travail réalisé. La répartition des points sera effectuée par les membres du groupe eux même et devra être cohérente avec le travail de chacun (note comprise entre 0 et 20 bien entendu).
- Deux rapports seront à remettre en anglais :
  - rapport de conception présentant le projet, la conception, le planning et les propositions d'amélioration et correction.
  - rapport de remise de projet indiquant le fonctionnement de l'application et les différentes extensions développées.

Récupération des fichiers

http://calamar.univ-ag.fr/uag/ufrsen/coursenligne/egrandch/projet/