Projet de Programmation 2
Jeu d'échec html

24 octobre 2013

L'objectif du projet est de réaliser en binômes un serveur web permettant de jouer aux échces. La composition des binômes devra être fournie le plus tôt possible, au plus tard à la rentrée. De nouvelles versions de ce sujet (avec notamment de nouvelles idées d'extensions) seront éventuellement postées dans les news.

On rappelle qu'il s'agit d'un *projet* de *programmation* orienté *objet* et *réseau*, ce sont donc ces aspects qui seront évalués avant tout.

1 Sujet

1.1 Quelques liens utiles :

— Java : `http://java.sun.com/javase/6/docs/api/`
— Eclipse : `http://www.eclipse.org/`
— Modifier le classpath : `http://whitesboard.blogspot.com/2009/02/eclipse-classpath.html`
— Utiliser des commantaires javadoc :
`http://www.siteduzero.com/tutoriel-3-35079-presentation-de-la-javadoc.html`

1.2 Description générale

L'objectif de ce projet est de réaliser un jeu d'échec en ligne sous forme de pages web. L'idée est que l'on lance un serveur qui implémente le protocole http, en génèrant à la volée les pages html à envoyer au client. Pour jouer, les utilisateurs lancent ainsi simplement un navigateur web quelconque et ouvrent l'url de votre serveur (`http://localhost:12345/` , typiquement). Ces pages html seront en fait des formulaires proposant des boutons qui permettent de bouger les pièces, sauvegarder une partie, etc.

1.3 Base fournie

Un exemple de page html est fourni dans le fichier `base.html` (avec la css à côté, `stylesheet.css`). Ouvrez-la dans un navigateur, vous pouvez constater que les pièces blanches sont cliquables. Essayez de cliquer sur l'une d'entre elles, constatez que l'adresse du navigateur a changé légèrement : les coordonnées de la pièce sur laquelle vous avez cliqué apparaît dans l'URL (`?E2.x=12&E2.y=13` par exemple). Ignorez les deux cases gris foncée (E3 et E4) pour le moment.

Nous allons procéder par étapes pour obtenir une première version fonctionnelle.

1.4 Version 0, serveur web

La première étape est de réaliser un serveur web ultra-simple mono-client. Il s'agit donc d'écouter sur un port TCP (par exemple 12345, qu'on rendra configurable facilement), d'accepter une connexion entrante, de réceptionner sa requête `GET`, de récupérer l'URL qui est dedans, et selon que c'est `base.html` `stylesheet.css`, ou autre fichier de `pieces/`, envoyer au client le contenu de l'un ou de l'autre fichier. Il faudra bien sûr enlever la partie `http://localhost:12345/` , et éventuellement `?E2.x=12&E2.y=13`, qui sera traitée à part. Renseignez bien sûr les en-têtes http habituels, notamment le `Content-Type`.

Cette version devrait ainsi se comporter de la même façon qu'en ouvrant la page directement dans le navigateur.

1.5 Version 1, serveur web dynamique

Il s'agit maintenant de générer automatiquement le contenu html `base.html` . Définissez-vous une hiérarchie d'objets pour décrire le plateau de jeu avec ses pièces. Écrivez alors une méthode qui affiche le contenu du plateau de jeu au format html, pour obtenir le même résultat que le fichier `base.html` (sauf les cases E3 et E4).

1.6 Version 2, ça bouge !

Il s'agit maintenant de prendre en compte les clics sur les pièces : lorsque l'on clique sur une pièce, il faut générer une page présentant des images cliquables sur les cases où cette pièce a le droit d'aller. C'est en fait une variante de la méthode précédente, prenant en paramètre la case où se situe la pièce que l'on désire bouger, qui devrait obtenir le même résultat que le fichier `base.html`, en particulier les cases E3 et E4, dans le cas où c'est sur le pion E2 que l'on a cliqué. Pour pouvoir distinguer du cas précédent, on donnera à ces images un nom légèrement différent.

On se contentera des déplacements de base de chaque pièce, sans prendre en compte la prise de pièces, le roque, la prise en passant ou la promotion des pions. On pourra dans un premier temps négliger les collisions avec ses propres pièces.

Enfin, lorsque l'utilisateur clique sur l'une de ces images, déplacez la pièce dans votre hiérarchie d'objets. La méthode définie en version 1 devrait du coup afficher correctement la nouvelle situation. Bien sûr, il faut alterner entre rendre cliquables les pièces blanches et les pièces noires.

2 Extensions

Une fois cette version de base fonctionnelle, de nombreuses extensions sont possibles et intéressantes à développer. Il n'est pas obligatoire de toutes les développer pour avoir une bonne note, mais cela y contribue bien sûr fortement, il faut en tous cas en développer plusieurs. La liste n'est également pas exhaustive. Si vous avez des idées, discutez-en avec votre chargé de TD. Nous posterons éventuellement de nouvelles versions du sujet avec d'autres idées.

2.1 If-modified-since

Pour éviter au navigateur de recharger entièrement à chaque fois toutes les images, émettez l'en-tête `Last-Modified` (que l'on positionnera à la date de dernière modification du fichier), et ajoutez le support de l'en-tête `If-Modified-Since`.

2.2 Notation de partie

En tournoi comme en cours, il est d'usage de noter la partie. Créez donc une hiérarchie d'objets pour enregistrer la liste des coups joués pendant la partie, et affichez-la en-dessous du plateau de jeu. Cf wikipedia pour la notation standard des coups.

2.3 Annuler/Rétablir, Avant/Arrière

On peut vouloir revenir en arrière, ajoutez un bouton pour cela.

On peut vouloir finalement retourner en avant, ajoutez un bouton pour cela. On affichera dans la liste des coups la position actuelle. En fait ces boutons sont des boutons avant/arrière.

Pour simplifier, on peut dire que si l'on joue un coup, on efface tous les coups qui suivaient à partir de la position actuelle.

2.4 Export de la partie

Plutôt que de faire un copier/coller à la main depuis la page html, il est pratique d'avoir un bouton "télécharger la partie", qui amène à une version texte pur de la liste des coups joués. Pour forcer le téléchargement de celle-ci (plutôt que simplement l'afficher dans le navigateur), on ajoutera ceci dans les en-têtes http :

```
Content-disposition: attachment; filename=moves.txt
```

2.5 Langue

Il est plus agréable que l'interface soit dans la langue naturelle de l'utilisateur. Rendez toues les éléments de votre interface facilement traduisibles dans le code source, et ajoutez le support de l'en-tête http `Accept-Language`.

2.6 Authentification http

On peut vouloir protéger l'accès au serveur. Mettez en place l'authentification http : il s'agit de renvoyer une erreur 401, le navigateur web va alors demander un login et un mot de passe à l'utilisateur et réitèrera la requête, avec cette fois-ci le login et le mot de passe, qu'il suffit donc de vérifier

2.7 Import d'une partie

Ajoutez des méthodes pour importer une partie existante (dans laquelle on pourra naviguer à l'aide des boutons Avant/Arrière). Le principe est d'ajouter un formulaire avec l'attribut `method=post`, et d'y mettre un `input` de type `file` et un bouton pour soumettre le formulaire. Le fichier est alors émis par le navigateur web juste après ses en-têtes http.

2.8 Entrée des coups à la main

Plutôt que cliquer, certains préfèreront taper le coup voulu (e.g. « e4 »). Ajoutez un formulaire en-dessous du plateau de jeu qui permet de rentrer le coup à la main, ce qui devra avoir le même effet que d'avoir cliqué sur une pièce puis sur une case de destination. Supportez d'abord la notation pour débutants « E2-E4 », puis la notation standard « e4 » (attention aux éventuelles ambiguités).

2.9 Prise de pièces

Lorsqu'une pièce arrive sur une case occupée par une pièce adverse, cette dernière doit être mise à côté du plateau, comme "prise".

2.10 Promotion des pions

Lorsqu'un pion atteint la dernière ligne, le joueur doit pouvoir choisir en quelle pièce il se transforme.

2.11 Base de parties

Il est utile de mémoriser les parties passées. Mettez en place une hiérarchie d'objets pour sauvegarder en mémoire différentes parties, entre lesquelles on pourra donc basculer.

Ajoutez des méthodes pour sauvegarder les parties dans un fichier lors de la terminaison du serveur, pour être rechargées lorsque l'on relance le serveur.

2.12 Multijoueur

Pour l'instant, le serveur n'accepte qu'une connexion à la fois. Laissez-le accepter deux connexions, l'une prendra les blancs, et l'autre les noirs. Notez qu'il faut bien sûr que le navigateur demande de temps en temps au serveur web si l'autre joueur a joué son coup. On pourra ajouter à la section `head` la balise `<meta http-equiv="Refresh" content="1">` pour demander au navigateur de rafraîchir la page toutes les secondes. Il sera à nouveau utile d'implémenter le support de `If-modified-since`.

2.13 https

Il est plus sage de chiffrer les échanges entre le navigateur web et le serveur, notamment pour éviter que le mot de passe soit transmis en clair ! Mettez en place le chiffrement ssl.

2.14 Connexion à FICS

FICS est un serveur de jeu d'échecs en ligne, auquel on peut se connecter publiquement. Ajoutez un module pour connecter votre serveur à FICS, y lancer une partie, et récupérer/envoyer les coups. Il suffit par exemple de se connecter à l'aide de `telnet` sur le port 5000 de `freechess.org` pour voir à quoi cela ressemble.

3 Organisation

Le projet est travaillé et étudié en binôme mais la notation est individuelle. Le projet *doit* être réalisé avec un outil de gestion de révision (svn, git, ...) : directement dans votre *home* ou sur la savanne du CREMI (`https://services.emi.u-bordeaux1.fr/projet/savane/`). Les enseignants évalueront la contribution de chacun des éléments du binôme au travail commun. Les enseignants se réserveront la possibilité de modifier la composition de chacun des binômes. Afin de permettre un travail profitable, il est conseillé de ne pas créer de groupes avec des niveaux trop différents.

4 Rapport

Il s'agit de mettre en valeur la qualité de votre travail à l'aide d'un rapport. Pour cela le rapport doit explicitement faire le point sur les fonctionnalités du logiciel (lister les objectifs atteints, lister ce qui ne fonctionne pas et expliquer - autant que possible - pourquoi). À cet effet, on proposera des jeux de tests permettant de mettre en valeur la correction du logiciel (est-ce qu'il fait bien ce qu'on attend de lui ?).

Ensuite le rapport doit mettre en valeur le travail réalisé sans paraphraser le code, bien au contraire : il s'agit de rendre explicite ce que ne montre pas le code, de démontrer que le code produit a fait l'objet d'un travail réfléchi et même parfois minutieux. Par exemple, on pourra évoquer comment vous avez su résoudre un bug, comment vous avez su éviter/éliminer des redondances dans votre code, comment vous avez su contourner une difficulté technique ou encore expliquer pourquoi vous avez choisi un algorithme plutôt qu'un autre, pourquoi certaines pistes examinées voire réalisées ont été abandonnées.

Il s'agira aussi de bien préciser l'origine de tout texte [1] ou toute portion de code empruntée (sur internet, par exemple) ou réalisée en collaboration avec tout autre binôme. Il est évident que tout manque de sincérité sera lourdement sanctionné.

Pour conclure on pourra traiter des limites et extensions possibles des logiciels proposés. Enfin on présentera une bibliographie (livres, articles, sites web) brièvement commentée. Ce rapport est le témoin de vos qualités scientifiques mais aussi de vos qualités littéraires (style, grammaire, orthographe, présentation). Pour présenter votre logiciel, on pourra adopter le plan suivant :

1. Présentation des fonctionnalités
2. Valorisation du travail réalisé
3. Diagramme des classes
4. Conclusion
5. Bibliographie commentée
6. Annexes et code du projet

5 Soutenance

La présentation finale du projet se fera en salle de TD autour d'une démonstration, des questions individuelles pourront être posées.

1. Directement dans le texte, par une note en bas de page comme celle-ci ou par une référence bibliographique entre crochet [1].