TP 1 : Découverte d'Eclipse, premiers projets

L'objectif de ce TP est de se familiariser avec les concepts de la plate-forme Eclipse vus en cours. Il permet également un premier (?) contact avec le JDT (*Java Developement Tool*) qui sera vu plus en détails par la suite.

Lancer et configurer Eclipse

Exercice 1 :

1. Lancer Eclipse à partir d'un terminal ou à partir du menu d'applications.
2. Choisir (pour le moment) l'emplacement de votre espace de travail à la racine de votre répertoire maison.
3. Cliquer sur l'icône du plan de travail (*Workbench*) (pour revenir à la page d'introduction (*Help / Welcome*).
4. Dans le menu *Help*, choisir *About Eclipse SDK*. Visualiser les differentes informations disponibles (*features*, *plug-ins*, *configuration*).
5. Dans *Window / Preferences*, regarder la version du JRE (*Java Runtime Environment*) qui est utilisé par défaut. S'assurer qu'il s'agit bien du JRE Sun et sinon changer cette version pour celle de Sun.

Programmation Java sous Eclipse

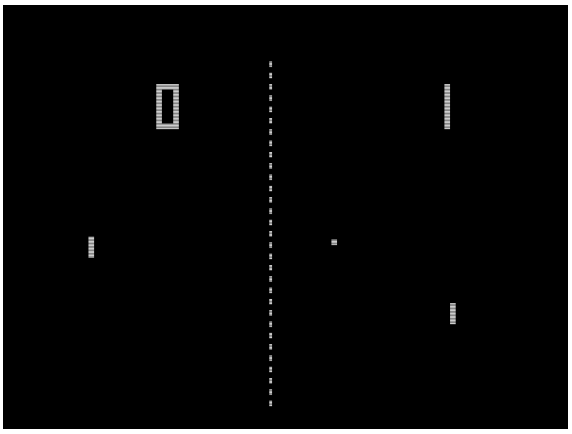
Exercice 2 : Création d'un projet sans code source préexistant.

1. Créer un nouveau projet « HelloWorld » dont le type est « Java Project ».
2. Rajouter une classe « HelloWorld » à ce projet (sans *package*, et avec une ébauche de méthode main()).
3. Rajouter une ligne affichant un message dans la méthode main().
4. Lancer votre programme.

Exercice 3 : Création d'un projet à partir de sources.

1. Pour le problème suivant, récupérer le sous-répertoire `pong` `http://www.di.ens.fr/~labatut/ED6/tp-1/Pong/`.
2. Créer un nouveau projet « Pong » dont le type est « Java Project », en choisissant l'option « à partir de sources » et indiquer le répertoire qui contient le sous-répertoire `pong`.
3. Dans la vue *Package Explorer*, vérifier que le project « Pong » contient bien un *package* pong.
4. Visualiser les fichiers importés dans l'éditeur et observer les différentes vues du plan de travail.

La suite du TP consiste à écrire un petit jeu video de type Pong (cf. Wikipédia : `http://fr.wikipedia.org/wiki/Pong`). Pour faciliter la tâche, une partie significative du code est fournie (celle qui s'occupe des graphismes et de l'interaction avec le(s) joueur(s)). Il ne reste « plus » qu'à comprendre les parties critiques du code fourni et à remplir les trous.



Exercice 4 : Le code source fourni ne compile pas, et Eclipse indique un certain nombre de problèmes (regarder dans la vue correspondante).

1. Résoudre ces problèmes : il s'agit uniquement de rajouter (ou compléter) des champs privés et des constructeurs dans les classes Ball, Paddle, Score et Table. Eclipse peut générer des constructeurs triviaux...
2. Une fois que toutes ces erreurs ont été corrigées, localiser (en utilisant la fonction recherche d'Eclipse) la méthode main().
3. Lancer le programme.

Exercice 5 : Rajouter aux classes précédentes des méthodes publiques très simples de type accesseur ou affecteur ; Eclipse peut aussi le faire automatiquement...

Pour l'instant, le programme ne fait rien (ou presque) à part dessiner le terrain, les raquettes, la balle et les scores.

Exercice 6 :

1. Trouver quelle est la fonction qui gère l'état du jeu, c'est-à-dire la position des différents objets graphiques mobiles (les raquettes et la balle) ansi que les scores : pour cela, partir de la fonction main() puis suivre les appels de fonction.
2. Trouver dans quelles variables sont stockées les actions du (des) utilisateur(s).
3. Rajouter le code nécessaire au déplacement de la raquette de gauche (d'une part en faisant attention à ce qu'elle ne sorte pas de l'écran et d'autre part en tenant compte de la variable qui contient la vitesse de déplacement des raquettes).
4. Idem pour la raquette de droite.

Exercice 7 :

1. Rajouter le code nécessaire au déplacement de la balle. Il faudra compléter la classe Ball avec des champs indiquant la direction actuelle de déplacement de la balle : les deux composantes d'un vecteur unitaire.
2. Gérer les rebonds sur les limites hautes et basses du terrain (la balle se « réfléchit » sur le bord...).
3. Gérer la sortie de balle du terrain à gauche et à droite (la balle dépasse la raquette du joueur correspondant). Dans ce cas, les scores doivent être mis à jour et la balle et les raquettes doivent revenir dans leur position initiale.
4. Gérer les rebonds sur les raquettes des joueurs.

5. Pour éviter que les rebonds soient trop « prévisibles », moduler la direction de rebond de la balle sur une raquette par un terme fonction de la position du point d'impact sur la raquette (par exemple : près du centre de la raquette, la balle rebondit avec une direction plus horizontale que sur les bords de la raquette).

Le jeu à deux joueurs doit maintenant être fonctionnel.

Exercice 8 : S'il reste du temps, en vrac... :

1. Rajouter le code permettant de déclarer un des joueurs vainqueurs lorsqu'il arrive à 11 points (constante MAX_SCORE).
2. Gérer les cas d'égalité (les scores sont tous les deux à 10 ou plus, mais il faut 2 points d'écart pour gagner).
3. Modifier aussi la vitesse (en norme) de la balle lors d'un rebond sur une raquette.
4. Ajouter deux autres joueurs (pour faire des doubles).
5. Remplacer le deuxième joueur par un robot : ce robot peut, par exemple, se déplacer vers l'intersection de sa ligne de fond avec la trajectoire de la balle après un rebond (uniquement quand la balle va dans sa direction, sinon il se contente de revenir au milieu)...