Projet POO2 : IA Tetris

Daeschler David (L3S5)

Choix d'implémentation

J'ai choisi d'utiliser le langage JavaScript pour développer le Tetris et son intelligence artificielle. Avec les derniers moteurs JavaScript (notamment V8 de Google), ce langage est devenu très performant et permet de créer des applications plus lourdes qu'il y a quelques années. Bien que sa syntaxe puisse gêner beaucoup de monde, je trouve au contraire sa syntaxe très simple et facile à comprendre. De plus, son aspect asynchrone est un énorme avantage. On peut ainsi exécuter des lignes de codes en attendant des évènements (requête AJAX, Time Outs…).

Pour l'implémentation du Tetris, j'ai d'abord hésité entre utiliser un canva HTML5 ou bien dessiner le Tetris en manipulant astucieusement le DOM. Mon expérience passée en développement web m'a rapidement fait choisir la deuxième solution. En effet, un tetris n'est rien d'autre qu'une série de carrés bien agencés, il est donc très facilement représentable avec des balises . Il devient alors aisé de manipuler la structure du tetris avec l'aide de jQuery et CSS3.

Structure du projet

L'organisation des fichiers du projet est la suivante :

- css/ : feuilles de styles
  - reset.css : désactive les valeurs par défauts des propriétés css des navigateurs
  - design.css : feuille de style du tétris
- images/ : liste des images utilisées par le projet
- javascript/ : fichiers javascript
  - Blocks.js : classe représentant les différents blocs du tétris
  - Helper.js : quelques fonctions pour simplifier le code
  - IA.js : classe de l'intelligence artificielle
  - Tetris.js : classe représentant un tétris
  - Lib_Transit.js : librairie JS permettant de manipuler les transitions CSS3 via javascript
- index.html : page de lancement du tetris

On en déduit rapidement les trois seules classes du projet : Tetris, Block, et IA. Elles sont toutes les trois indépendantes. Helper est une classe static.

Code minimal pour lancer un tetris

Pour lancer un nouveau tetris, on commence par inclure jQuery depuis les serveurs Google. On placera toutes les lignes suivantes entre les balises <head>.

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/2.0.3/jquery.min.js"></script>
```

Puis la librairie transit également :

```
<script src="javascript/lib_Transit.js"></script>
```

On inclue ensuite les feuilles de styles :

```
<link rel="stylesheet" href="css/reset.css" type="text/css" />
<link rel="stylesheet" href="css/design.css" type="text/css" />
```

On peut ensuite placer dans le <body> une balise qui servira de conteneur au tetris.

```
<div id="tetris_1"></div>
```

Juste avant </body>, on inclue les fichiers relatifs au tetris et on déclare une variable globale dont tous les fichiers ont besoin : « fastMode ».

```
<script> var fastMode = false; </script>
<script src="javascript/Helper.js"></script>
<script src="javascript/IA.js"></script>
<script src="javascript/Tetris.js"></script>
<script src="javascript/Blocks.js"></script>
```

Et pour finir, on peut enfin instancier un tetris et une IA.

```
<script>
var tetris = new Tetris('tetris_1');
var intel = new IA(tetris);
</script>
```

« tetris_1 » doit être l'identifiant du conteneur qui va accueillir le tetris.

Toutes ces lignes sont écrites dans « index.html ».

Pour lancer le projet : lancez index.html dans un navigateur web récent. (chrome / firefox)

Il est important de noter que les moteurs javascripts récents désactivent les timer à intervalle régulier très court (de l'ordre de quelques ms) lorsque l'onglet du navigateur n'a plus le focus (par sécurité et pour économiser l'utilisation CPU). Il est donc primordial, lorsque le tetris est lancé, de ne pas switcher d'onglets ni de mettre la fenêtre en arrière-plan. Cela pourrait causer de gros bugs.

Le tetris devrait alors fonctionner normalement.

Vous pouvez à tout moment activer/désactiver l'IA du tetris, et activer/désactiver le mode rapide, pratique pour voir jusqu'où peut aller l'IA.

Développement et difficultés rencontrés

J'ai tout d'abord commencé par développer le tetris pour le rendre fonctionnel en mode utilisateur, sans penser à aucun moment à l'IA, afin de bien la rendre indépendante. Cette première partie de développement a pris environ 4 jours. J'ai ensuite ajouté l'IA au projet, qui comme demandé ne fait que émuler des évènements clavier. On peut facilement déclencher de faux évènements claviers en javascript, en précisant le code de la touche appuyé. Voici les trois fonctions principales que l'IA utilise pour bouger les pièces.

```
IA.push_key_times = function(k, times)
{
	var nb = 0;
	var it = setInterval(function()
	{
		IA.press_key(k);
		IA.unpress_key(k);
		nb++;

		if(times == nb) clearInterval(it);
	},IA.secureDelay);
}

IA.press_key = function(k)
{
	var e = jQuery.Event("keydown");
	e.which = k;
	e.keyCode = k;
	$(document).trigger(e);
}

IA.unpress_key = function(k)
{
	var e = jQuery.Event("keyup");
	e.which = k;
	e.keyCode = k;
	$(document).trigger(e);
}
```

Le seul lien entre l'IA et le tetris est un attribut. L'IA possède une référence vers l'instance du tetris, mais n'appel jamais aucune méthode de l'objet du tetris. Elle ne fait que consulter la grille, la cloner pour la modifier, et l'évaluer.

La principale difficulté du projet a été de bien gérer le coté asynchrone de javascript. Il faut obligatoirement passer par de nombreuses closure pour éviter les conflits, car toutes les références vers l'objet « this » change dès qu'on change de contexte, c'est-à-dire ici essentiellement les fonctions de callback des timers.

Javascript a une manière particulière de gérer les évènements claviers lorsqu'on reste appuyé sur une touche. J'ai donc du ajouté une grosse couche par-dessus la gestion d'évènements pour éviter le spamming d'appels aux fonctions de déplacement (il faut encore passer par des timers). Il m'a alors fallu faire très attention, car j'ai utilisé différents timers pour chaque type de cas (premier appui de touche, appui continu, appui pour déplacer, appui pour faire une rotation…). J'ai donc du callé l'IA pour qu'elle émule le déclenchement de touches sans perturber les timers. En effet, si l'IA a besoin de faire 3 déplacements à droite, si elle fait ses 3 appels trop vite, la sécurité des

timers du côté du tetris risque de masquer les derniers appels, et la pièce n'aurait bougé que d'une case. Par défaut, l'IA se comporte donc comme un humain et ne spam pas l'appui des touches. Il est possible de désactiver complètement ces timers en activant le mode « speed ».

Mis à part cet aspect, je n'ai rencontré aucune difficulté particulière. Javascript est un langage que je maitrise depuis plusieurs années, ce ne sont pas donc pas les compétences techniques qui ont manqué.

Fonctionnement de l'IA

L'IA fonctionne comme ceci :

```
Toutes les 100ms :
   Si un nouveau tour a démarré et que l'IA est activé
      Tour++
      Recherche meilleur position :
      Pour toutes les rotations et positions possibles
          Si il y a de la place pour placer la pièce
          Et si il elle touchera un sol à cette position
          Et si on peut l'y placer en ligne droite en se décalant un peu
             Clonage de la structure de la grille
             Incrustation de la pièce dans la grille clonée
             Eval1 = Evaluation de la grille avec la pièce courante
             Récurrence : recherche meilleur position avec pièce suivante
             Eval2 = Evaluation de la grille avec en plus la pièce suivante
Retourner la position/rotation qui correspond à la plus grande valeur de Eval1+Eval2
Faire une rotation de la pièce jusqu'à obtenir celle voulu
Déplacer la pièce dans la bonne colonne
Descendre la pièce jusqu'à ce que le tetris refuse de descendre plus bas et passer au tour
suivant
```

L'évaluation est basée sur un principe simple : choisir la situation la moins pire.

Etant donné qu'il n'existe aucune meilleure solution pour placer une pièce, l'évaluation ne fait que baisser le score total sans jamais l'augmenter. La meilleure solution sera alors celle qui aura le score le plus grand (mais sera en dessous de 0).

Je trouve qu'augmenter le score de l'évaluation selon le nombre de lignes n'avait pas trop de sens, j'ai alors préféré diminuer le cout des diminutions selon le nombre de lignes. En effet, il est difficile d'équilibrer les bonnes et mauvaises situations dans un tetris, j'ai donc plutôt considéré uniquement les mauvaises.

L'évaluation commence par compter le nombre de lignes. Elle compte ensuite la hauteur de la plus haute tour et soustrait ce nombre*10 au score.

En fonction du nombre de lignes, l'IA estimera le cout du cas « faire un trou ». Pour aucune ligne, chaque trou réalisé fera descendre le score de 200. Pour chaque ligne supplémentaire, le cout du trou est divisé par 2 jusqu'à atteindre 0 dans le cas ou on fait 4 lignes.

Enfin, la fonction essayera d'obtenir le tetris le plus « tassé » possible, en parcourant chaque case vide et en faisant -10*Y pour chacune d'entre elles.

Cet algorithme permet d'obtenir une IA qui fera en moyenne entre 1000 et 6000 lignes.