

Avant-projet d'informatique : Babyfoot virtuel

Mageri Filali Maltouf

8 janvier 2010

1 Présentation du sujet choisi

Le projet consiste à réaliser une implémentation simplifiée d'un jeu de babyfoot :

- simulation physique des mouvements de la balle
- collision entre les joueurs, la balle et les murs
- modélisation 3D de la table et des joueurs
- interface homme/machine ergonomique
- implémentation d'un joueur non humain
- détection des fautes

2 Analyse de la solution envisagée

La solution envisagée se compose des packages suivants :

– core

1. Baby.java : programme principal
2. Damier.java : affichage d'un damier
3. Ecran.java : écran d'aide au début du jeu
4. **Panneau3D.java** : construction du babyfoot, démarrage de tous les services (physique, affichage, contrôles, son)
5. Tapis.java : tapis sur le babyfoot

– ia

1. **IA.java** : contrôle d'une équipe par l'intelligence artificielle
2. **Oracle.java** : prédiction de trajectoires

– ihm

1. Clavier.java : contrôle d'une équipe par l'utilisateur au clavier
2. Texte2D.java : affichage de texte 2D dans un environnement 3D
3. Wiimote.java : contrôle d'une équipe par l'utilisateur avec une wiimote

– obj

1. Balle.java : affichage de la balle sous forme de sphère

2. Barre.java : construction du modèle de la barre
3. Joueur.java : construction du modèle du joueur cylindre/parallélépipède/modèle
4. Objet.java : interface définissant un objet 3D

– **outils**

1. Constantes.java : tous les paramètres du jeu
2. Couleurs.java : définition de couleurs de base pour l’affichage 3D
3. Matériaux.java : définition de matériaux pour l’affichage 3D
4. Triplet.java : structure de donnée permettant de manipuler des triplets
5. Visionneuse.java : classe permettant de visualiser des fichiers .obj

– **physik**

1. Collision.java : détection des collisions à intervalle régulier
2. IntersectionSP.java : intersection d’une sphère et d’un parallélépipède rectangle
3. Intervalle.java : structure de données permettant de manipuler des intervalles
4. PhysiqueBalle.java : mouvement de la balle → extension de la classe PhysiqueSolide
5. PhysiqueBarre.java : mouvement de la barre → extension de la classe PhysiqueSolide
6. PhysiqueJoueur.java : mouvement d’un joueur attaché à la barre → extension de la classe PhysiqueSolide
7. Physik.java : interface définissant un système physique (solide rigide) → sera remplacée par la classe PhysiqueSolide
8. Synchro.java : synchronisation des données entre PhysiqueBarre et PhysiqueJoueur → sera remplacée par la classe Liaisons
9. PhysiqueSolide.java : classe définissant la physique des corps solides

– **struct**

1. Matrice3.java : matrices de taille 3×3
2. Matrice4.java : matrices de taille 4×4
3. Quaternion.java : quaternions (représente les rotations de \mathbb{R}^3)
4. Triplet.java : triplets de flottants
5. Vecteur3.java : vecteurs de taille 3

– **wiimote**

1. Controle.java : classe qui enregistre les données reçues par la wiimote

– **zik**

1. MP3.java : classe permettant de lire des fichiers .mp3

Le moteur physique permet de modéliser des solides rigides simples. On modélise les joueurs par un parallélépipède rectangle et la balle par une sphère. Ceci permet de calculer plus facilement les collisions (solutions algorithmiques développées dans la littérature).

La gestion des collisions se fait entre la balle et chaque joueur. Il faut tenir compte du temps de calcul. Certaines optimisations sont décrites dans les ouvrages cités en annexe.

L’intelligence artificielle est calquée sur les jeux de sports en équipe.

3 Échéancier

Objectif	Date	Commencé ?
Simulation	fin janvier	✓
Collision	mi-janvier	✓
Modélisation	mi-janvier	✓
Interface homme/machine	fin février	✓
IA	début février	✓
Fautes	début mars	

4 État actuel

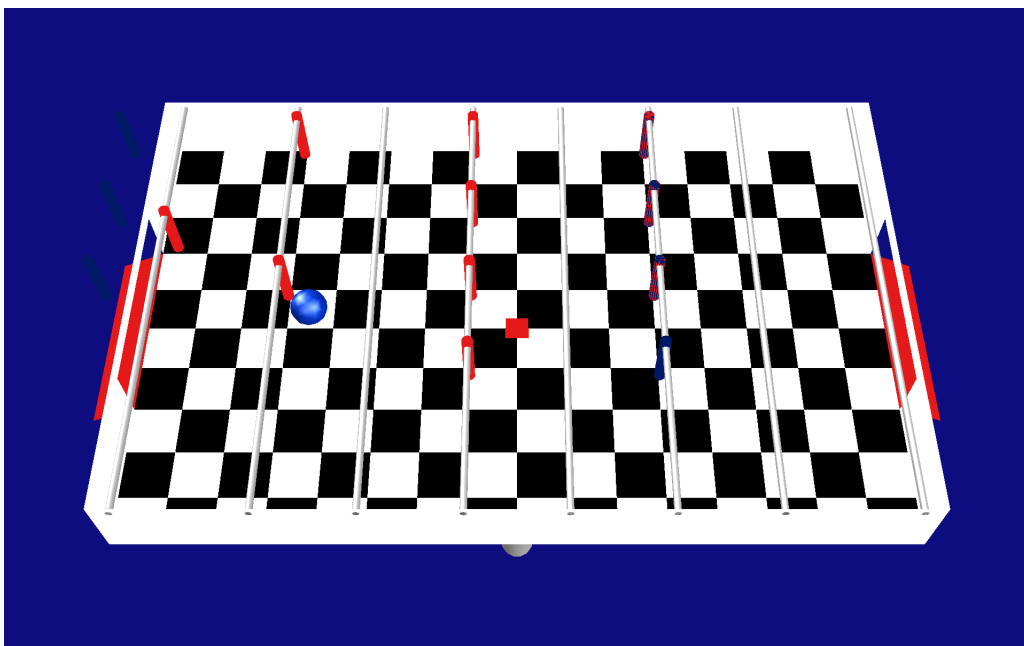


FIGURE 1 – Recopie d'écran de la version du 8 janvier 2010

5 Remarques

5.1 Interface homme/machine

Dans un premier temps, le mouvement des barres est commandé par le clavier. Les touches directionnelles \leftarrow et \rightarrow permettent de contrôler la rotation d'une barre donnée. Les touches \uparrow et \downarrow servent à translater la barre. Cette interface est gérée par la classe **Behavior** de **Java3D**.

Dans un deuxième temps, on utilise la wiimote. La bibliothèque **wiuse** a été programmée en langage C. La bibliothèque **wiuseJ** est une API Java fondée sur la version C. La wiimote est reliée à l'ordinateur par bluetooth. Une extension de la classe **Behavior** permet à chaque instant d'accéder :

- à trois données d'accélération (cartésiennes)
- à deux données angulaires (roulis, tangage)

Les valeurs enregistrées à intervalle régulier sont directement envoyées dans le moteur physique.

5.2 Fichiers manipulés

On crée un modèle simplifié des joueurs avec **Java3D** : ils sont représentés par un parallélépipède rectangle. La classe **Joueur** dispose également d'une méthode permettant de charger un fichier .obj. Ce format a été défini par Wavefront Technologies. Il est ainsi possible de charger des fichiers 3D créés avec Blender.

Pour ce qui est des fichiers sonores, on utilise la bibliothèque **JLayer** de Javazoom qui permet de lire des fichiers .mp3.

5.3 Librairies

1. *vecmath : The Vecmath API*, <https://vecmath.dev.java.net/>
2. *java3d : Java 3D Parent Project*, <https://java3d.dev.java.net/>
3. *wiiusej : Java API for Wiimotes*, <http://code.google.com/p/wiiusej/>
4. *MP3 library for the Java Platform*, <http://www.javazoom.net/javalayer/javalayer.html>

5.4 Références

1. Andrew Davison, *Pro Java 6 3D Game Development*, <http://fivedots.coe.psu.ac.th/~ad/jg2/>
2. Gino van den Bergen, *Collision Detection in Interactive 3D Environments*, Elsevier, 2004
3. Mat Buckland, *Programming Game AI by Example*, Wordware Publishing, 2005
4. Ian Millington, *Artificial Intelligence for Games*, Elsevier, 2006
5. Ian Millington, *Game Physics Engine Development*, Elsevier, 2007

6 Sources

Vous trouverez les fichiers relatifs au projet à l'adresse suivante : <http://code.google.com/p/mabave/>.