
ÉCOLE
POLYTECHNIQUE
MONTRÉAL

**PHS4700**
**Devoir #1**
**Trajectoire d'une balle au tennis**
**Présenté à M. Guy Marleau**

Matthieu Ouellette-Vachon – 1325531

22 Septembre 2009

# Introduction

Ce document est produit dans le cadre du devoir #1 du cours « Physiques pour les applications multimédia ». Le but de ce laboratoire est de faire une simulation par ordinateur d'une balle de tennis à l'aide de deux types d'accélérations différentes. La simulation a lieu à l'intérieur d'un terrain de jeu délimité au centre par un filet. Deux zones sont visibles, celle du joueur (aussi appeler Zone #1) et celle de l'adversaire (Zone #2). La première simulation utilise une accélération constante qui est en fait la constante de gravité de la terre soit 9.8 m/s$^2$ vers le bas en z. La deuxième simulation quant à elle utilise une accélération qui varie en fonction de la vitesse de la balle. Le programme à réaliser lit en entrée le type de simulation (1 ou 2) puis la position initiale de la balle et ensuite lit la vitesse initiale de la balle et démarre la simulation. La position initiale doit se trouver dans la zone du joueur (Zone #1) mais il n'existe pas de contrainte au niveau de la vitesse initiale.

# Problème

Le problème du devoir consiste à résoudre les équations de mouvements afin de réaliser une simulation réaliste au point de vue physique. Il y a quatre équations différentielles à résoudre, deux pour la simulation #1 et deux autres pour la simulation #2

## Simulation #1

Pour cette simulation, le programme informatique utilise une méthode analytique pour faire la simulation.  Voici comment est obtenue la solution analytique :

$$\frac{dv(t)}{dt} = g$$

$$\frac{dr(t)}{dt} = v(t)$$

Or comme g est constant, on a alors :

$$v(t) = v_0 + \int_{t_o}^{t} a(t')dt' = v_0 + \int_{t_o}^{t} g\, dt'$$

$$v(t) = v_0 + gt$$

$$r(t) = r_0 + \int_{t_o}^{t} v(t')dt' = r_0 + \int_{t_o}^{t} (v_0 + gt\,dt') = \int_{t_o}^{t} v_0 dt' + \int_{t_o}^{t} gt\,dt'$$

$$r(t) = r_0 + v_0 t + g\frac{t^2}{2}$$

Ce sont donc ces deux équations qui sont programmées dans le logiciel de simulation. La première permet de déterminer la vitesse finale de la balle lors de l'arrêt de la simulation. La deuxième permet de déterminer à tout moment la position de la balle. Cette dernière est évaluée à chaque 0.005 secondes pour avoir une bonne série de points à afficher.

### Simulation #2

Pour cette simulation, le programme utilise une méthode numérique pour faire la simulation. La méthode utilisée est celle de Runge-Kutta d'ordre 4. Le programme peut aussi utiliser la méthode d'Euleur mais dans les sources remises, c'est la méthode de Runge-Kutta qui est utilisée. Les équations à résoudre pour ce problème sont :

$$\frac{dv(t)}{dt} = g - k * v^2(t)$$

$$\frac{dr(t)}{dt} = v(t)$$

Où g est la gravité (0, 0, -9.8) $m/s^2$ et k = 0.01 $secondes^{-1}$. Au temps $t_0$ la vitesse de la balle est donnée par $v_0$ (fournis par l'utilisateur) et la position par $r_0$ (fournis par l'utilisateur). La résolution de la première équation à l'aide de l'analyse numérique nous permettra d'obtenir la vitesse de la balle au temps $t_i$. Cette vitesse au temps $t_i$ sera ensuite utilisée pour résoudre la deuxième équation ce qui nous donnera alors la position de la balle au temps $t_i$. À l'aide de ces valeurs, on détermine si la balle dépasse le filet, ou termine sa trajectoire soit dans la zone adverse, soit dans la zone du joueur ou encore à l'extérieur du terrain.

# Solution

La solution proposée est un projet Visual Studio 2008. Le fichier de la solution se trouve dans « build/vss9/GameEngine.sln ». Cette solution contient deux projets. Le projet « GameEngine »

qui est le cœur du programme de simulation. Il contient le code pour l'affichage de la fenêtre, les solutions numériques, et bien d'autres trucs utiles. La source de ce projet se trouve dans le répertoire « src ». Le projet « Tp1 » contient le code requis pour faire toute la simulation demandée, le code source pour ce projet se trouve dans « src\tp1 ».

Pour exécuter la solution, deux options sont offertes. On peut soit démarrer le projet depuis Visual Studio. Dans ce cas, il faut aller dans les propriétés du projet « Tp1 » et sous l'onglet « Debugging » il faut changer la valeur du champ « Working Directory » pour qu'elle soit assignée à « $(SolutionDir)..\..\bin ». Sinon, à chaque compilation depuis Visual Studio, l'exécutable est copié dans le dossier « bin » à la racine du projet. Il suffit alors de double cliquer sur l'exécutable copier dans ce dossier pour lancer la simulation.

Lors de la simulation, certaines touches sont accessibles et permettent de faire les choses suivantes :

- La touche « a » permet de faire basculer l'affichage des axes (Rouge = X, Vert = Y, Bleu = Z)
- La touche « r » permet de rejouer la simulation depuis le début
- Le déplacement de la souris avec un bouton enfoncé permet de faire pivoter la caméra

# Résultats

Voici un tableau récapitulatif montrant différents résultats d'exécutions. Pour chaque ligne on a le type de simulation (Type #1 ou Type #2), la position initiale (au haut de la case) et finale (au bas de la case) de la balle, la vitesse initiale et finale (comme pour la position), une description de l'état de la balle à la fin de la simulation (voir paragraphe plus bas) et finalement le temps écoulé lors de la trajectoire de la balle.

L'état de la balle à la fin de la simulation peut être :

- La balle est sortie au fond du terrain (Fond)
- La balle est sortie à droite du terrain (Droite)
- La balle est sortie à gauche du terrain (Gauche)
- La balle frappe le filet (Filet)

- La balle tombe dans la zone adverse (Zone #2)
- La balle tombe dans la zone du joueur (Zone #1)

| Type | Position (m) | Vitesse (m/s) | État Final | Temps écoulé (s) |
|---|---|---|---|---|
| **Type #1** | (2.5, 2.5, 0.25)<br>(4.49, 11.8, 0.743) | (3.0, 14.0, 4.0)<br>(3.0, 14.0, -2.52) | Filet | 0.67 |
| **Type #2** | (2.5, 2.5, 0.25)<br>(4.58, 11.8, 0.618) | (3.0, 14.0, 4.0)<br>(2.94, 12.8, -2.89) | Filet | 0.70 |
| **Type #1** | (7.5, 4.0, 1.0)<br>(6.78, 10.7, -0.005) | (-4.5, 42.0, -5.5)<br>(-4.5, 42.0, -7.07) | Zone #1 | 0.165 |
| **Type #2** | (7.5, 4.0, 1.0)<br>(6.78, 10.5, -0.014) | (-4.5, 42.0, -5.5)<br>(-4.53, 39.4, -7.13) | Zone #1 | 0.160 |
| **Type #1** | (7.5, 4.0, 1.0)<br>(1.72, 19.4, -0.0235) | (-4.5, 12.0, 5.5)<br>(-4.5, 12.0, -7.09) | Zone #2 | 1.29 |
| **Type #2** | (7.5, 4.0, 1.0)<br>(1.61, 18.2, -0.0338) | (-4.5, 12.0, 5.5)<br>(-4.77, 10.4, -7.12) | Zone #2 | 1.27 |
| **Type #1** | (2.0, 4.0, 1.0)<br>(4.39, 27.9, -0.022) | (2.0, 20.0, 5.0)<br>(2.0, 20.0, -6.71) | Fond | 1.2 |
| **Type #2** | (2.0, 4.0, 1.0)<br>(4.33, 25.2, -0.013) | (2.0, 20.0, 5.0)<br>(1.95, 16.2, -6.71) | Fond | 1.18 |
| **Type #1** | (8.0, 3.0, 0.5)<br>(-2.17, 19.9, -0.018) | (-6.0, 10.0, 8.0)<br>(-6.0, 10.0, -8.61) | Gauche | 1.7 |
| **Type #2** | (8.0, 3.0, 0.5)<br>(-2.46, 18.3, -0.011) | (-6.0, 10.0, 8.0)<br>(-6.66, 8.58, -8.59) | Gauche | 1.65 |
| **Type #1** | (1.5, 6.0, 2.0)<br>(14.9, 14.4, -0.023) | (8.0, 5.0, 7.0)<br>(8.5, 5.0, -9.41) | Droite | 1.68 |
| **Type #2** | (1.5, 6.0, 2.0)<br>(13.9, 13.9, -0.017) | (8.0, 5.0, 7.0)<br>(7.07, 4.62, -9.51) | Droite | 1.64 |

# Conclusion

En conclusion, on voit très bien que dans le cas de la simulation #2, la balle s'arrête plus rapidement que dans le cas de la simulation #1. La simulation effectuée suit donc la logique car l'accélération dans le cas #2 est toujours plus importante vers le bas (à moins d'une vitesse initiale de zéro) que par rapport à la simulation #1. Dans le cas de la simulation #2, deux méthodes numériques ont été implémentées, celle d'Euler et celle de Runge-Kutta d'ordre 4. Dans les deux cas, le résultat de la simulation est sensiblement le même. Il ne semble donc pas y avoir de problème de stabilité numérique dans le cadre de ce devoir #1.