

RESSOURCES COMMUNES

Sommaire

- 0 – Boucle principal de gameplay
- 1 – Structures du code
- 2 – Carte de jeu
- 3 – Apprentissage par renforcement
- 4 – Actions possibles pour l'agent / IA
- 5 – Perceptions et états de l'agent
- 6 – Apprentissage par renforcement
- 7 – Visuel de notre jeu

0 – Boucle principal

Un match de tennis. Un joueur contre une intelligence artificielle entraînée à renvoyer la balle. La condition pour renvoyer la balle est d'être dans une des quatre zones de récupération.

1 – Structures du code

```
| - Res
|     | - [différentes images]
| - src
|     | - gest_event.c
|     | - main.c
|     | - map.c
|     | - player.c
|     | - render.c
| - map.txt
```

Détails :

gest_event.c	: gestion des événements du clavier et souris
main.c	: code principal
map.c	: gestion de la carte de jeu
player.c	: gestion du joueur
render.c	: affiche les éléments du jeu en 3D (raycasting)

2 – Carte de jeu

[illegible]

Détails :

Les valeurs 1 correspondent à la foule et représentent les limites du terrain.

Les valeurs 2 correspondent aux filets.

Côté droit : joueur.

Côté gauche : IA.

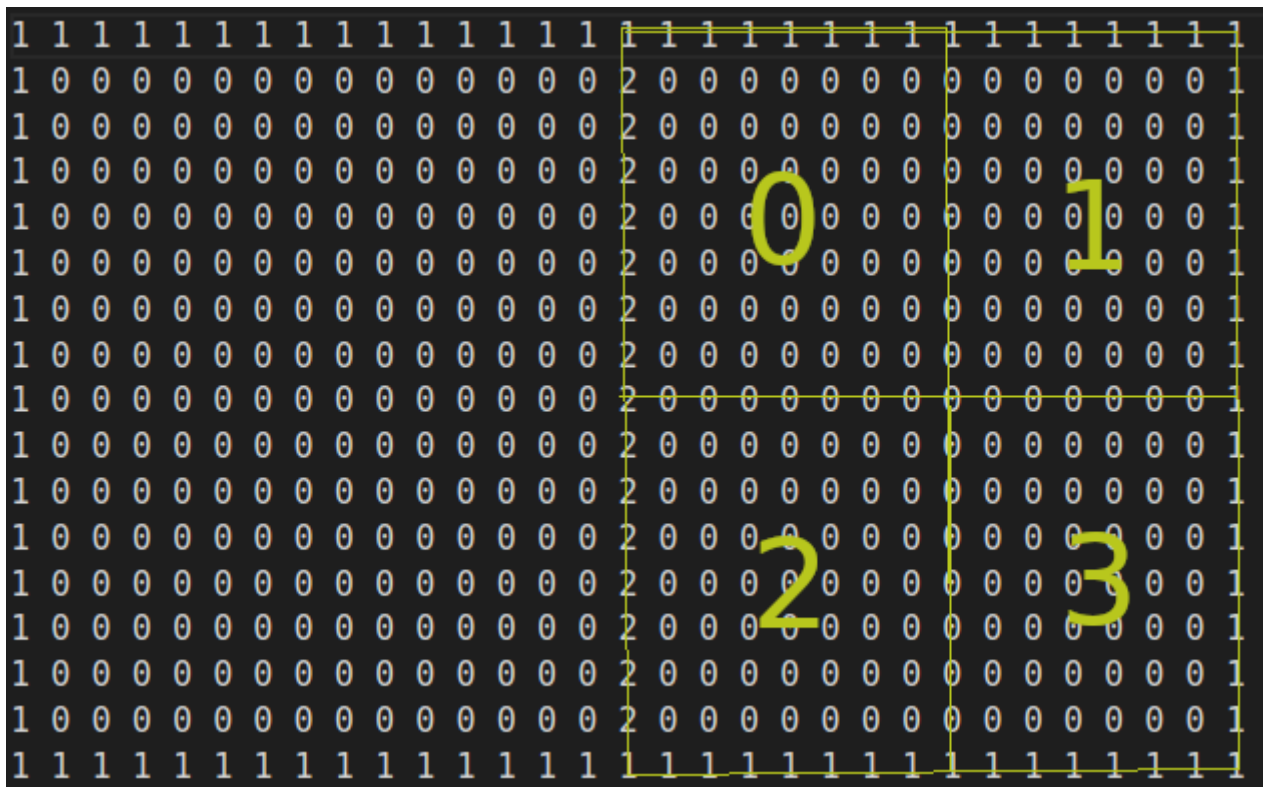
L'axe horizontal est l'axe des X.

L'axe vertical est l'axe des Y.

3 – Apprentissage par renforcement

Nous entraînons une IA à se positionner dans la bonne zone pour récupérer la balle.

Il y a 4 zones.



L'IA doit apprendre à se positionner dans la bonne zone pour attraper la balle.

Le déplacement de l'IA est libre dans toute sa partie du terrain.

Le déplacement n'est pas discret : l'IA bouge en X et Y.

4 – Actions possibles pour l'agent / IA

Notre agent peut se déplacer dans sa partie du terrain.

Il peut :

- avancer sur l'axe des X
- avancer sur l'axe des Y
- reculer sur l'axe des X
- reculer sur l'axe des Y
- rester immobile

Tableau des actions :

←	↑	→	↓	immobile
---	---	---	---	----------

5 – Perceptions et états de l'agent

L'agent a plusieurs perceptions pour prendre ses décisions.

Il connaît :

- la zone dans laquelle il se trouve actuellement (zone_agent)
- la zone de départ de la balle qu'il va devoir attraper (zone_balle_départ)
- l'angle à plat de départ de la balle (angle_plat_balle_départ°)
- l'angle en hauteur de départ de la balle (angle_hauteur_balle_départ°)

Il y a :

- 4 zones d'agent possible
- 4 zones de départ possible
- 5 angles à plat
- 3 angles en hauteur

Tableau des états :

zone_agent	zone_balle_départ	angle_plat	angle_hauteur
0	0	1	1
1	1	2	2
2	2	3	3
3	3	4	1
1	2	5	1
...

Nous avons donc :

$$4 * 4 * 5 * 3 \text{ états} = 240 \text{ états}$$

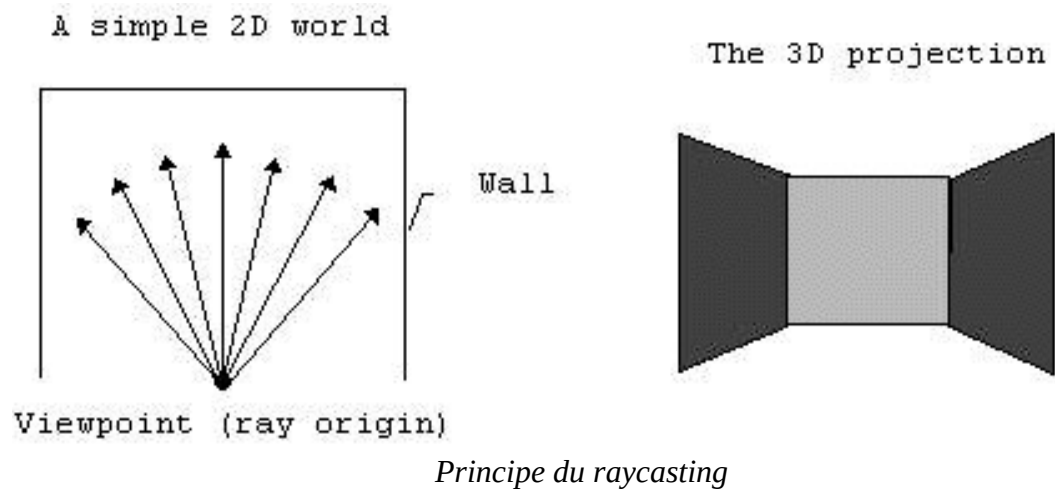
Avec les 5 actions cela nous fait un tableau de 240 lignes et 5 colonnes donc 1200 cases.

6 – Apprentissage par renforcement

Nous utilisons le Q-learning pour entraîner notre IA.

7 – Visuel de notre jeu

Nous avons décidé de faire toute la logique de notre jeu en 3D.



Nous utilisons exactement cette technique pour faire le rendu de notre jeu.

Exemple de visuel « in game » :

