Web Tron IA

Delphine Millet - Mathieu Spegagne

Sommaire

- Projet initial
- Moteur graphique
- Intelligences Artificielles
  - Line Dist
  - Snail AI
  - Minimax
- Heuristiques
- Pistes d’amélioration

1.

PROJET INITIAL

Implémenter une intelligence artificielle

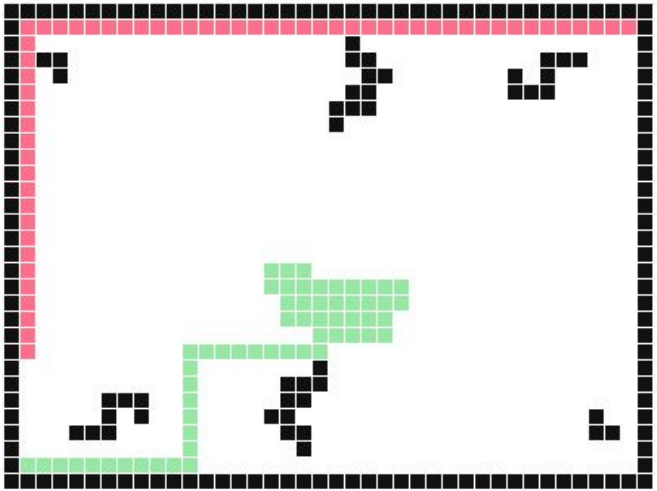
Google AI Challenge - Tron

- Gagné par *a1k0n_* en 2010
- Règles du jeu
  - 2 joueurs laissent une trace lumineuse
  - éviter les murs et traces lumineuses
  - percuter un mur ou une trace ⇒ perdu !
- Comment jouer ?
  - dans un terminal
  - spécifier les IA utilisées + la carte

Google AI Challenge - Tron

- Web Tron AI ⇒ version simplifiée
- Ce qu'on a conservé
  - la génération du jeu et des joueurs
  - le contrôle manuel d'un joueur
- Ce qu'on a ajouté
  - 3 intelligences artificielles
  - des composants graphiques

TRON GAME



PLAYING...

Game | Mode | Credits

PLAYER vs AI  AI vs AI

Both the green and the pink snakes are controlled by the computer

2.

MOTEUR GRAPHIQUE

Générer le jeu

Création de la carte

```
this.draw = function(){
  var canvas = document.getElementById('canvas');
  var ctx = canvas.getContext('2d');
  for(var j=0;j<this.h*this.w;j++){
    var x = j%this.w;
    var y = Math.floor(j/this.w);
    //Coloration
    if(this.map[j] != 0){
      switch(this.map[j]){
         case -1:
             ctx.fillStyle='#111';
             break;
         // Coloration différente pour chaque cas
      }
      ctx.fillRect(x*10, y*10, 9, 9);
    }
  }
}
```

```
this.init = function(){
     //Init players
     this.p2.init(1,this.h,this.w);
     this.p1.init(2,this.h,this.w);
     //Init map
     for(var j=0;j<this.h*this.w;j++){
       this.map[j] = 0;
     }
     this.addBorders();
     this.addBlocks();
   }
```

Création des joueurs

```
var Player = function(id, ai, move){
   this.ai = ai;
   this.id = id;
   this.move = move;

   // Positionnement du joueur dans la carte

   this.updatePosition = function(x,y){
     this.x = x;
     this.y = y;
     this.idx = this.x+this.y*this.w;
   }
}
```

```
var Game = function(){
   this.map = [];
   this.w = 0|(canvas.width/10);
   this.h = 0|(canvas.height/10);
   this.p1 = new Player(
        1,
        document.getElementById('switch').checked,
        2
   );
   this.p2 = new Player(2,true,0);
   this.gameover = false;
}
```

Moteur de jeu

```
function init(){
   game = new Game();
   game.init();
   game.draw();
   tmr = setInterval(frame, FRAMEDELAY);
 }

function frame(){
   keyboardManager();
   gameManager();
   game.draw();
 }
```

```
this.addPlayer = function(player){
   var adv = (player.id === 1 ? this.p2 : this.p1);
   if(this.isWall(player.x,player.y)){
      this.map[player.idx] = player.id+2;
      this.gameover = true;
      if(player.idx === adv.idx){
         player.draw = true;
      }
   }
   else{
      this.map[player.idx] = player.id;
   }
}
```

3.
INTELLIGENCES ARTIFICIELLES

Doter les joueurs d’une intelligence

Line Dist

- Intelligence la plus faible
- Direction du serpent ⇒ distance maximale avec un mur
- Calcul de la meilleure orientation

```
var bestHeur = 0;
var bestMove = 0;
for(var move = 0; move < 4; move++){
    var heur = 0;
    newX = x + dx[move];
    newY = y + dy[move];
    while(!game.isWall(newX, newY)){
        heur++;
        newX += dx[move];
        newY += dy[move];
    }
    if(heur>bestHeur){
        bestHeur = heur;
        bestMove = move;
    }
}
return bestMove;
```

Snail AI

- Même comportement que LineDist
- Décision prise face à un mur
- Comportement en escargot

```
while(!game.isStuck(newX, newY) && !game.isWall(newX, newY)){
        heur++;
        newX += dx[newMove];
        newY += dy[newMove];

        if(game.isWall(newX, newY)){
                if(newX<(game.w/2) && (newMove%2) === 1){
                        altMove=2;
                }
                if(newX>(game.w/2) && (newMove%2) === 1){
                        altMove=0;
                }
                // Tests sur newY
                newX += dx[altMove];
                newY += dy[altMove];
        }
}
```

Minimax

- Implémentation de Minimax adaptée
- Clonage du jeu pour chaque mouvement possible
- Déplacement du joueur
- Calcul du score depuis les heuristiques
- Alpha-beta pruning
- Iteration sur 5 niveaux

```
var bestValue = -5000;
for(var newMove = 0; newMove<4; newMove++){
  var newNode = clone(node);
  adv = (player.id === 1 ? newNode.p2 : newNode.p1);
  newPlayer = (player.id === 1 ? newNode.p1 : newNode.p2);
  newPlayer.move = newMove;
  newPlayer.updatePosition(newPlayer.x+dx[newPlayer.move],newPlayer.y+dy[newPlayer.move]);
  newNode.addPlayer(newPlayer);
  var value = minimaxAI(newNode,(depth-1),false,adv,init,alpha,beta)[0];
  bestValue = Math.max(bestValue,value);
  if(bestValue === value){
    bestMove = newMove;
  }
  alpha = Math.max(alpha,value);
  if(beta <= alpha){
    break;
  }
}
return [bestValue,bestMove];
```

4.
HEURISTIQUES

Améliorer les intelligences artificielles

Calcul des heuristiques

Elles différent avec la situation du joueur

- Les joueurs évoluent dans le même espace
- Les joueurs sont séparés
- Le joueur va rencontrer un mur
- L’adversaire va rencontrer un mur
- Les deux joueurs vont rentrer en collision

```
this.diffFirstJoinable = function(player,adv){
     var score = 0;
     for(var i=0;i<this.w;i++){
       for(var j=0;j<this.h;j++){
         if(!this.isWall(i,j)){
           var distancePlayer = Math.floor(Math.sqrt(Math.pow((player.x - i),2)+Math.pow((player.y - j),2)));
           var distanceAdv = Math.floor(Math.sqrt(Math.pow((adv.x - i),2)+Math.pow((adv.y - j),2)));
           if(distancePlayer < distanceAdv){
             score++;
           }
           if(distancePlayer > distanceAdv){
             score--;
           }
         }
       }
     }
     return score;
   }
```

```
this.score = function(player){
 var score = 0;
 var adv = (player.id === 1 ? this.p2 : this.p1);
 this.spaceCounting(this.p1);
 this.spaceCounting(this.p2);


 if(player.space !== adv.space){
   score = player.space - adv.space;
 }
 else{
   score = this.diffFirstJoinable(player,adv);
 }
   if(this.map[adv.idx] === adv.id+2){
     score = 2000;
   }
   if(this.map[player.idx] === player.id+2){
     score = -2000;
   }
   if(player.draw || adv.draw){
     score = 0;
   }

   return score;
}
```

5.
DÉMONSTRATION

Un aperçu des possibilités de notre jeu

6.

PISTES D' AMÉLIORATION

Vers une IA plus performante

Pistes d'amélioration

- diffFirstJoinable avec A*
- AI basée sur réseaux neuronaux et apprentissage (Gagnant Google AI Challenge)

Merci !

Des questions?