Projet : Puissance 4 Contre la Machine

Contre-rendu 5

Nous avons une première version de l'algorithme qui fonctionne : elle fait des coups à peu près logiques, mais laisse parfois l'adversaire gagner car elle n'arrive pas à l'anticiper. Avant, l'algorithme ne marchait pas à cause de quelques erreurs dans les fonctions de base du programme, tellement que la fonction qui vérifie si un élément est un array qui renvoyait true à tous les coups.

Il me reste à :

- Faire en sorte que l'algorithme ne joue pas à un endroit où l'adversaire peut gagner
- Eviter les blocs inutiles (typiquement, cela ne sert à rien de bloquer une colonne de 2 jetons dans une colonne où il reste un seul emplacement, même chose pour les lignes)
- Régler les problèmes finaux (mauvaise détection entre les jetons adverses et les siens, jouer au hasard au début ou dans la colonne 4 plutôt que toujours dans la première colonne etc..)

D'un point de vue mémoire, il reste amplement assez de mémoire pour le reste des fonctions que nous devrons implémenter (capteurs, servo-moteurs et éventuellement buzzer et écran LCD) puisque seul 16% de l'espace de stockage de programmes et 30% de la mémoire vive sont occupées.

Pour ce qui est de la rapidité, le programme mets environ 1 seconde par coup, ce qui ne sera rien par rapport au temps que le bras mettra pour se déplacer.

Après ça, il me restera à coder les capteurs. Cela devrait être simple puis que je n'aurais qu'à faire en sorte que l'algorithme attende les valeurs renvoyées par les digitalRead des capteurs des 7 colonnes.

Voici la nouvelle version de la fonction loop de l'algorithme :

```
void loop() {
  // put your main code here, to run repeatedly:
  int prio[7] = {0,0,0,0,0,0,0};
  for (int i = 0; i < 7; i++){

    if (canPlay(i) == true){
      columns[i] += 1;
      int posTempo[2] = {i, columns[i]};
      Serial.println(posTempo[0]);
      Serial.println(posTempo[1]);
      histoC[getNbMoves()][0] = posTempo[0];
      histoC[getNbMoves()][1] = posTempo[1];

      if (checkWin(posTempo, histoC) == true){       //Si l'ordi peut gagner en jouant dans cette colonne, prio absolue
        prio[i] = 100;
      }

      else if (checkLine3(posTempo, histoJ) == true || checkColumn3(posTempo, histoJ) || checkDiago3(posTempo, histoJ)){ //Si l'ordi peut bloquer une victoire adverse
        prio[i] = 99;
      }

      else if (checkLine3(posTempo, histoC) == true || checkColumn3(posTempo, histoC) == true || checkDiago3(posTempo, histoC)){ //Si l'ordi peut aligner trois jetons
        prio[i] = 10;
        if (columns[i] < 6){
          int posTempo2[2] = {i, columns[i] + 1};
          if (checkWin(posTempo2, histoJ) == true){    //...mais que cela permet au joueur de gagner
            prio[i] -= 100;     //prio = -90
          }
          if (checkLine3(posTempo2, histoJ) == true || checkColumn3(posTempo2, histoJ) || checkDiago3(posTempo2, histoJ)){ //ou d'aligner 3 jetons
            prio[i] -= 1;      //prio = 9
          }
          else if (checkLine2(posTempo2, histoJ) == true || checkColumn2(posTempo2, histoJ) == true || checkDiago2(posTempo2, histoJ)){ //ou d'aligner 2 jetons
            prio[i] -= 2;      //prio = 8
          }
        }
      }
```

```
      else if (checkLine2(posTempo, histoJ) == true || checkColumn2(posTempo, histoJ) == true || checkDiago2(posTempo, histoJ)){ //Si l'ordi peut bloquer un alignement de 2 jetons adverses
        prio[i] = 5;
      }

      else if (checkLine2(posTempo, histoC) == true || checkColumn2(posTempo, histoC) == true || checkDiago2(posTempo, histoC)){  //Si l'ordi peut aligner deux jetons
        prio[i] = 3;
        if(columns[i] < 6){
          int posTempo2[2] = {i, columns[i] + 1};
          if (checkWin(posTempo2, histoJ) == true){    //...mais que cela permet au joueur de gagner
            prio[i] -= 100;    //prio = -97
          }
          else if (checkLine3(posTempo2, histoJ) == true || checkColumn3(posTempo2, histoJ) || checkDiago3(posTempo2, histoJ)){ //ou d'aligner 3 jetons
            prio[i] -= 1;     //prio = 4
          }
          else if (checkLine2(posTempo2, histoJ) == true || checkColumn2(posTempo2, histoJ) == true || checkDiago2(posTempo2, histoJ)){ //ou d'aligner 2 jetons
            prio[i] -= 2;     //prio = 3
          }
        }
      }
      columns[i] -= 1;
      histoC[getNbMoves()][0] = -1;
      histoC[getNbMoves()][1] = -1;

    }

    else{
      prio[i] = 0;
    }
  }

  int prioFin = -999;
  int columnFin = -1;

  for (int i = 0; i < 7; i++){
    Serial.print("Prio : ");
    Serial.println(prio[i]);
  }
```

```
for (int i = 0; i < 7; i++){
  if (prio[i] > prioFin){
    prioFin = prio[i];
    columnFin = i;
    Serial.println(prioFin);
    Serial.println(columnFin);
  }
}


if (columnFin == -1){
  int colPlay = random(0,7);
  while(canPlay(colPlay) == false){
    colPlay = random(0,7);
  }
  play(colPlay);
  Serial.print("L'ordi a joué en colonne ");
  Serial.println(colPlay + 1);
}

else{
  play(columnFin);
  Serial.print("L'ordi a joué en colonne ");
  Serial.println(columnFin + 1);
}                                              //Fin tour ordi
for (int i = 0; i < 7; i++){
  Serial.println(columns[i]);
}

Serial.println("Où voulez-vous jouer?");

while (Serial.available() == 0){                  //On attend que le joueur joue
}

int columnJ = Serial.parseInt();
Serial.println("Vous avez joué dans la colonne" + columnJ);
play(columnJ);
```