Heilmann G4

Hugo

Compte-rendu séance du 09/01/2023

Lors de cette séance, j’ai tout d’abord lancé l’impression 3D de l’intégralité des pièces d’échec car celle-ci risque de prendre un long moment car il y a un total de 32 pièces 3D à imprimer.

Ensuite, nous avons décidé avec Camilia des dimensions finales de notre jeu d’échec en fonction des pièces en cours d’impression. Le plateau de jeu fera donc 46x46 cm ; il aura des cases de 5cm de côté afin de permettre au pièces les plus larges (les rois et reines avec 2cm de diamètre) de passer entre les cases pour se déplacer sans bousculer les autres pièces ; ainsi qu’une marge de 3cm tout autour du plateau qui servira à recueillir les pièces mortes.



Enfin, j’ai consacré le reste du temps à l’amélioration du code est plus spécialement du void loop dont chaque boucle représentera le déroulement d’un tour.

// Pour le déplacement des pièces avec le bras

String A,B,C,D,E,F,G,H;

const int moteurX=3; // Moteur pour les déplacements en ligne

const int moteurY=5; // Moteur pour les déplacements en colonne

const int aimant=2; // Aimant pour attraper les pièces

int x=0; // Position x de l'électroaimant

int y=0; // Position y de l'électroaimant

int px=0; // Position x de la pièce à déplacer

int py=0; // Position y de la pièce à déplacer

boolean white\_turn=true; // Pour savoir à qui est le tour

int jeu[8][8] = { // Instance du tableau de jeu

{20,30,40,50,60,40,30,20}, // Ligne 1 pion=1

{10,10,10,10,10,10,10,10}, // Ligne 2 tour=2

{0,0,0,0,0,0,0,0}, // Ligne 3 cavalier=3

{0,0,0,0,0,0,0,0}, // Ligne 4 fou=4

{0,0,0,0,0,0,0,0}, // Ligne 5 reine=5

{0,0,0,0,0,0,0,0}, // Ligne 6 roi=6

{1,1,1,1,1,1,1,1}, // Ligne 7 vide=0

{2,3,4,5,6,4,3,2}, // Ligne 8 piece noire=piece blanche\*10

};

void setup() {

// put your setup code here, to run once:

Serial.begin(9600); // Accès au moniteur série

pinMode(moteurX,OUTPUT); // Moteur ligne comme receveur

pinMode(moteurY,OUTPUT); // Moteur colonne comme receveur

digitalWrite(moteurX,HIGH); // Moteur ligne éteint

digitalWrite(moteurY,HIGH); // Moteur colonne éteint

digitalWrite(aimant,HIGH); // Aimant desactivé

int line\_possibilities[8] = {

1,2,3,4,5,6,7,8

};

String column\_possibilities[8] = {

A,B,C,D,E,F,G,H

};

}

void loop() {

// Demander la ligne de la pièce à déplacer;

Serial.print("Ligne de la pièce à déplacer ? ");

if (Serial.read()!= "") {

target\_line=Serial.read();

continue;

}

if (Serial.read() not in line\_possibilities){

Serial.print("Inexistant");

break;

}

else {

Serial.println(target\_line);

}

// Demander la colonne de la pièce à déplacer

Serial.print("Colonne de la pièce à déplacer ? ");

if (Serial.read()!= "") {

target\_column=Serial.read();

continue

}

if (Serial.read() not in column\_possibilities) {

Serial.print("Inexistant");

break;

}

else {

Serial.println(target\_column);

}

// Vérifier que la case ne soit pas vide

if (jeu[target\_line][target\_column]==0){

Serial.println("This hut is a no man's land");

}

// Demander la ligne de la case à atteindre

Serial.print("Ligne de la case à atteindre ? ");

if (Serial.read()!= "") {

wanted\_line=Serial.read();

continue;

}

if (Serial.read() not in line\_possibilities){

Serial.print("Inexistant");

break;

}

else {

Serial.println(wanted\_line);

}

// Demander la colonne de la case à atteindre

Serial.print("Colonne de la case à atteindre ? ");

if (Serial.read()!= "") {

wanted\_column=Serial.read();

continue

}

if (Serial.read() not in column\_possibilities) {

Serial.print("Inexistant");

break;

}

else {

Serial.println(wanted\_column);

}

// Vérifier que la couleur de la pièce à déplacer soit raccord avec la couleur de la personne dont c'est le tour

if ((white\_turn==true) and (jeu[target\_line][target\_column]>=10)){

Serial.println("This is not your color.");

}

if ((white\_turn==false) and (jeu[target\_line][target\_column]<10)){

Serial.println("This is not your color.");

}

// Vérifier si le coup demandé est légal; (boolean true)

if (legality()==false){

Serial.println("Illegal move !");

}

// Vérifier que la case sur laquelle déplacer la pièce est libre

// false : si la pièce présente est ennemie : élimination;

// sinon : coup illégal;

if (jeu[wanted\_line][wanted\_column]!=0){

if ((jeu[target\_line][target\_column]<10) and (jeu[wanted\_line][wanted\_column]>=10)){

continue;

}

else if ((jeu[target\_line][target\_column]>=10) and (jeu[wanted\_line][wanted\_column]<10)){

continue;

}

else {

Serial.println("You can't kill your teammates!");

}

}

// Déplacer l'électroaimant sous la case de la pièce à déplacer

aller\_case(target\_line, int target\_column);

// Déplacer la pièce

deplacer\_piece(wanted\_line, int wanted\_column);

// Changement de joueur

change\_turn();

}

// Ensemble des fonctions utiles

// Fait revenir le bras à l'origine

void revient\_origine() {

avancer\_caseX(-x);

avancer\_caseY(-y);

x=0;

y=0;

}

// avancer/reculer jusqu'à la colonne ax

void avancer\_caseX(int ax) {

int nbPas=ax\*50; // Demi-case environ 50px ?

for (int i=0;i<nbPas;i++) {

digitalWrite(moteurX,LOW); // on allume le moteur ligne pendant nbPas

}

digitalWrite(moteurX,HIGH); // on éteint le moteur ligne

x=ax;

}

// avancer/reculer jusqu'à la ligne ay

void avancer\_caseY(int ay) {

int nbPas=ay\*50; // Demi-case environ 50px ?

for (int i=0;i<nbPas;i++) {

digitalWrite(moteurY,LOW); // on allume le moteur colonne pendant nbPas

}

digitalWrite(moteurY,HIGH); // on éteint le moteur colonne

y=ay;

}

// Déplace l'électroaimant

void aller\_case(int goX, int goY) {

int diffX=goX-x;

int diffY=goY-y;

avancer\_caseX(diffX);

avancer\_caseY(diffY);

x+=diffX;

y+=diffY;

px=x;

py=y;

}

// Déplace une pièce

void deplacer\_piece(int i, int j) {

if (already\_taken(i,j)==true) {

dead(i,j);

aller\_case(i,j);

}

digitalWrite(aimant,LOW); // active l'aimant pour attraper la pièce

avancer\_caseX(i); // se déplace sur la ligne voulue (avec la pièce)

avancer\_caseY(j); // se déplace sur la colonne voulue (avec la pièce)

digitalWrite(aimant,HIGH); // désactive l'aimant pour lacher la pièce

int piece=jeu[px][py]; // Conserve la valeur de la pièce à déplacer

jeu[px][py]=0; // Rend la case de la pièce que l'on déplace vide

jeu[i][j]=piece; // Donne la valeur de la pièce déplacée à la case d'arrivée

}

// gérer ici l'endroit où sont envoyés les pièces mortes

void dead(int i, int j) {

jeu[i][j]=0; // rend la case de la pièce morte vide

// Déplace la pièce dans la zone morte

digitalWrite(aimant,LOW); // active l'aimant

aller\_case(-20,-20); // les nombres négatifs représentent la zone morte

// il faut modifier cette partie pour que les pièces mortes soient côte à côte

digitalWrite(aimant,HIGH); // désactive l'aimant

}

// Passe le tour à l'autre joueur

void change\_turn(){

if (white\_turn==true){

white\_turn = false;

}

else {

white\_turn = true;

}

}

// Pour vérifier si un coup est légal

boolean legality() {

// if (le coup est légal) {

if (3+3=6){

return true;

}

else {

return false;

}

}

// Pour vérifier si une case est occuppée

boolean already\_taken(int i, int j) {

if (jeu[i][j]==0){

return false;

}

else {

return true;

}

}