



ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE ET D'ANALYSE DES  
SYSTÈMES - RABAT

---

**Projet C :  
Robot Intelligent**

---

*Réalisé par :*

Anas L'HICHOU  
Rida TAZI

*Encadré par :*

Mr. Mohamed RADOUANE



## Résumé

Ce projet est le fruit des conseils et critiques bienveillantes d'un grand nombre de personnes. Nous tenons à les remercier ici et leur faire part de toutes nos gratitude, pour avoir été à l'écoute et toujours d'une aide précieuse.

Plus particulièrement, nous souhaitons adresser à travers ces courtes lignes nos remerciements les plus sincères au Professeur Mr. Mohamed Radouane, qui a eu le soin d'examiner et encadrer ce travail et ainsi nous offrir une véritable opportunité d'apprentissage.

Nous aimerais aussi gratifier les efforts de nos professeurs des modules de « Structures de données » : Monsieur Abdellatif EL FAKER , « Algorithmique » : Monsieur Ahmed ETTALBI et de « Techniques de programmation » : Monsieur Hatim GUERMAH pour sa disponibilité, son savoir-faire et les méthodes de programmation et raisonnement qu'il nous a transmis.

Enfin, nos remerciements s'adressent aux professeurs et au corps administratif de l'Ecole Nationale Supérieure de l'Informatique et de l'Analyse des Systèmes-ENSIAS.

# Table des matières

<b>1 Présentation du projet</b>	<b>1</b>
1.1 Sujet . . . . .	1
1.2 Outils utilises . . . . .	2
1.2.1 SDL2 . . . . .	2
1.2.2 Les environnement de programmation utilises . . . . .	3
1.2.2.1 CodeBlocks 17.12 . . . . .	3
1.2.2.2 Linux . . . . .	3
1.2.3 Photoshop . . . . .	4
<b>2 Derolement du programme</b>	<b>5</b>
2.1 Menu . . . . .	5
2.1.1 Menu console . . . . .	5
2.1.2 Menu graphique . . . . .	5
2.2 Interface d'acceuil . . . . .	6
2.2.1 Principe de fonctionnement du bouton . . . . .	7
2.3 Deplacement du Robot . . . . .	8
2.3.1 Mouvement du Robot . . . . .	8
2.3.1.1 La fonction Move() . . . . .	9
2.3.1.2 La fonction verifier() . . . . .	9
2.3.1.3 La fonction moveObject() . . . . .	9
2.3.2 Le compteur . . . . .	10
2.3.3 Le chemin . . . . .	11
2.4 Interface du relancement . . . . .	12
2.4.1 Robot intelligent 2.0 . . . . .	12



# Chapitre 1

## Présentation du projet

### 1.1 Sujet

Le projet consiste à développer une application en C pour commander un Robot. Il s'agit de déplacer à l'aide de commandes des objets de différentes formes géométriques. Le robot doit se déplacer à la zone 1 et vérifie la forme de l'objet déposé. Si c'est un cercle, le robot va le transporter vers la zone 2, sinon l'objet va garder sa place.

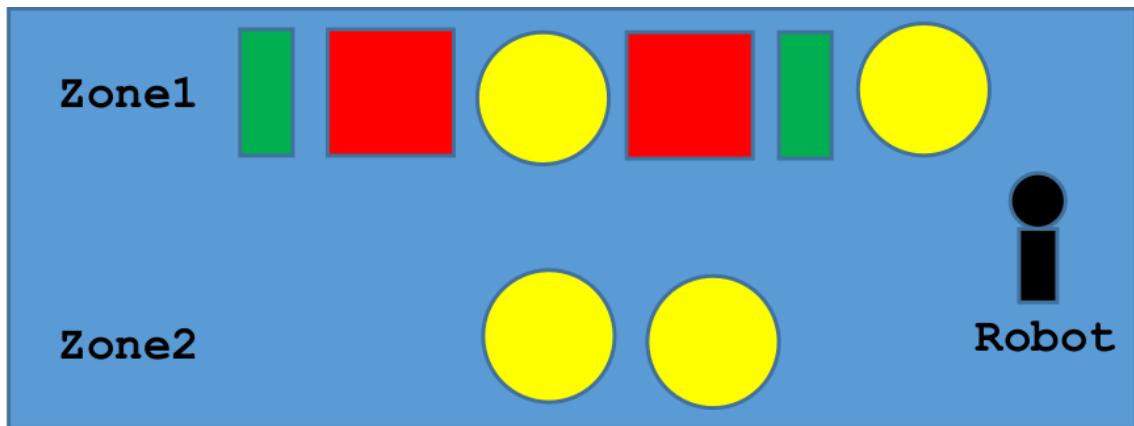


FIGURE 1.1 – Schéma descriptif

## 1.2 Outils utilisés

### 1.2.1 SDL2

Pour l'interface graphique on a opter pour la "SDL2 (Simple Directmedia Layer)". Simple DirectMedia Layer (SDL) est une bibliothèque logicielle libre. Son API<sup>1</sup> est utilisée pour créer des applications multimédias en deux dimensions pouvant comprendre du son comme les jeux vidéo, les démos graphiques, les émulateurs, etc. Sa portabilité sur la plupart des plateformes et sa licence zlib, très permissive, contribuent à son succès.



FIGURE 1.2 – Logo SDL

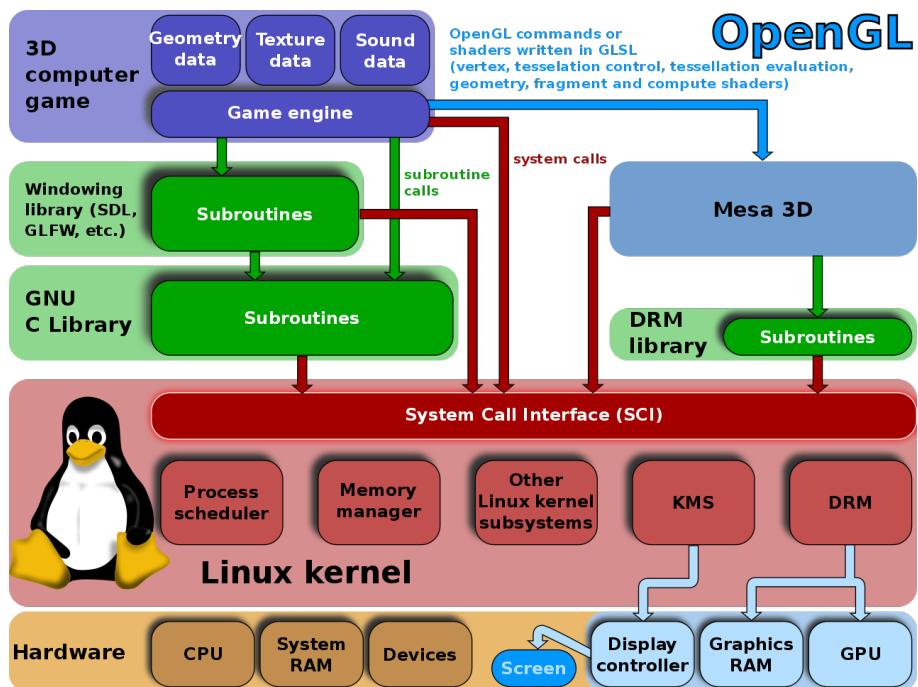


FIGURE 1.3 – Simple Directmedia Layer

1. Application Programming Interface

## 1.2.2 Les environnement de programmation utilisés

### 1.2.2.1 CodeBlocks 17.12

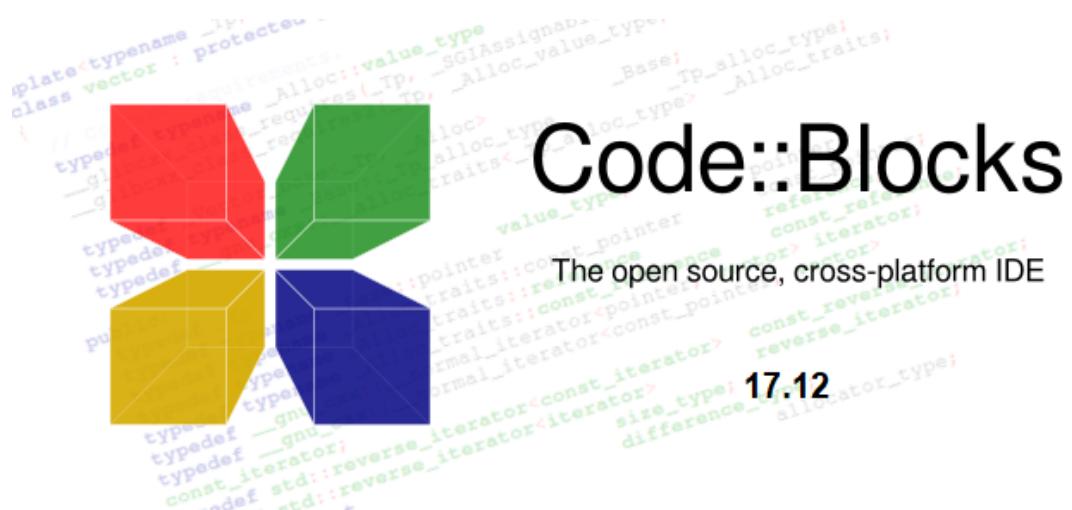


FIGURE 1.4 – CodeBlocks Logo

### 1.2.2.2 Linux

On a notamment travaille avec le distro Elementary Os, qui s'est avéré stable et plus efficace du côté utilisation.



FIGURE 1.5 – Elementary OS Logo

Le Makefile utilise :

```
1 CC := clang
2
3 CFLAGS := `sdl2-config --libs --cflags` -ggdb3 -O0 --std=c99 -Wall -lSDL2_image -lm
4
5 HDRS := fonctions.h
6
7 SRCS := main.c fonctions.c
8
9 OBJS := $(SRCS:.c=.o)
10
11 EXEC := game
12
13 all: $(EXEC)
14
15 $(EXEC): $(OBJS) $(HDRS) Makefile
16     $(CC) -o $@ $(OBJS) $(CFLAGS)
17
18 clean:
19     rm -f $(EXEC) $(OBJS)
20
21 .PHONY: all clean
```

Listing 1.1 – Makefile

### 1.2.3 Photoshop

On a concu la plupart des images utilises a l'aide de ce programme.



FIGURE 1.6 – Photoshop Logo

# Chapitre 2

## Déroulement du programme

Voici comment l'utilisateur va interagir avec le robot.

### 2.1 Menu

#### 2.1.1 Menu console

A l'exécution du programme dans sa première version, on se trouve devant la page d'accueil du robot et un menu console où l'utilisateur doit choisir l'ordre des formes, puis on lui demande de cliquer sur le bouton start.

```
Choisissez l'ordre de vos formes avant de commencer :  
0 = Rectangle  
1 = Cercle  
2 = Triangle  
3 = Carré  
4 = Random  
5 = Tout Random ou continuer Randomly  
6 = Quitter  
  
Forme 1 : 1  
Forme 2 : 1  
Forme 3 : 1  
Forme 4 : 1  
Forme 5 : 5  
  
CLIQUEZ SUR START
```

FIGURE 2.1 – Menu Console

#### 2.1.2 Menu graphique

Puisque le fait d'avoir à manipuler la console et l'application peut sembler fatigant pour quelque utilisateur, on a dû penser à tous intégrer dans l'interface graphique tout en l'adoptant au fait que l'utilisateur soit capable de changer les formes géométriques juste avec un tout petit clique.

Après que les figures ont été choisies l'utilisateur est invité à cliquer sur le bouton "CONTINUE" pour lancer le Robot.



FIGURE 2.2 – Menu Graphique

```

1 if (event4.button.x>=230 && event4.button.x<=531 && event4.button.y>=434 && event4.button.y<=
2   535 ){
3   if(event4.type == SDL_MOUSEBUTTONDOWN)
4     dd=1;
5   Cont.w = 400;
6   Cont.h = 400;
7   Cont.x = 180;
8   Cont.y = 279;
9   SDL_RenderClear(render);
10  SDL_RenderCopy(render, texChoix, NULL, &back);
11  SDL_RenderCopy(render, texRand, NULL, &Rand);
12  SDL_RenderCopy(render, texCont, NULL, &Cont);
13  showShapes(TabForm,render,texCircle,texRect,texTriangle, texCarre,rect);
14  SDL_RenderPresent(render);
}

```

Listing 2.1 – Code Source du bouton CONTINUE

## 2.2 Interface d'accueil

L'interface d'accueil est munie d'un bouton "START" permettant le lancement du programme. Le bouton s'agrandit quand le curseur survole celui-ci.

### 2.2.1 Principe de fonctionnement du bouton

Le bouton s'agit d'un rectangle "SDL", son fonctionnement se traduit par une boucle qui attend l'événement du clique gauche de la souris quand le curseur se trouve encapsule dans les dimensions (en pixel) du bouton.

```
1 while(dd)
2 {
3     while(SDL_PollEvent(&event2)){
4         if (event2.button.x>=230 && event2.button.x<=531 && event2.button.y>=434 && event2.button.
5             y<= 535)
6         {
7             button.w = 400;
8             button.h = 400;
9             button.x = 180;
10            button.y = 279;
11            SDL_RenderClear(render);
12            SDL_RenderCopy(render, texMenu, NULL, &menu);
13            SDL_RenderCopy(render, texButton, NULL, &button);
14            SDL_RenderPresent(render);
15
16            if (event2.type==SDL_MOUSEBUTTONDOWN) {
17                m_vel = SPEED;
18                dd=0;
19            }
20        }
21    else {
22        button.w = 301;
23        button.h = 301;
24        button.x = 230;
25        button.y = 329;
26
27        SDL_RenderClear(render);
28        SDL_RenderCopy(render, texMenu, NULL, &menu);
29        SDL_RenderCopy(render, texButton, NULL, &button);
30        SDL_RenderPresent(render);
31    }
32 }
```

Listing 2.2 – Code Source du bouton

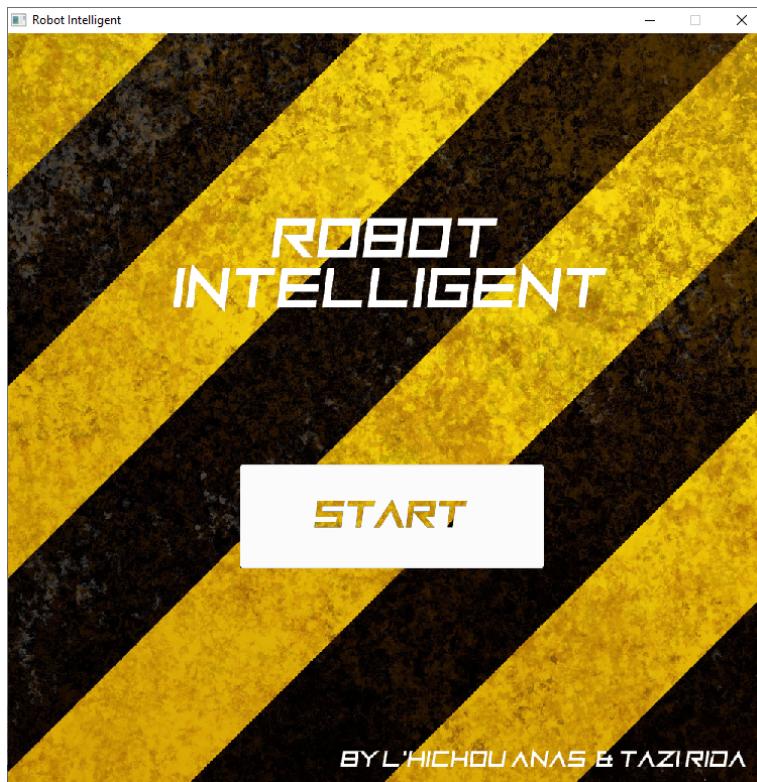


FIGURE 2.3 – Interface d'accueil

## 2.3 Déplacement du Robot

Après le clique sur le bouton "CONTINUE", le Robot s'affiche à sa position initiale ainsi que les formes géométriques selon l'ordre choisi par l'utilisateur.

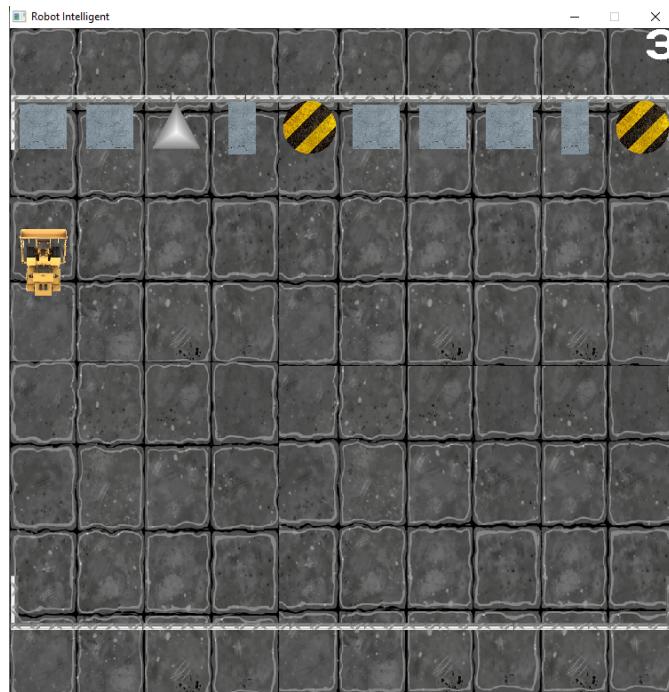


FIGURE 2.4 – Etat initial du robot

### 2.3.1 Mouvement du Robot

A l'état initial le Robot se trouve entre les deux zones avant de se déplacer à la zone 1, il parcourt ainsi les différentes formes en vérifiant s'il s'agit d'un cercle ou pas. Ainsi s'il détecte un cercle il le déplace vers la zone 2, et passe à la vérification de la forme suivante.

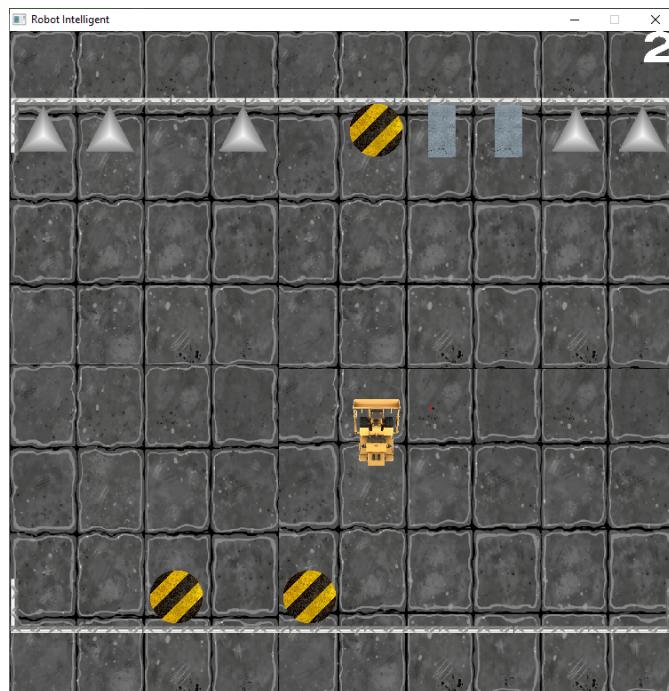


FIGURE 2.5 – Robot en déplacement

Pour ce fait trois fonctions ont ete utilise :

### 2.3.1.1 La fonction Move()

La fonction Move() permet le deplacement vertical du robot de la zone 1 a 2 ou de la zone 2 a 1 ceci ce fait avec le parametre z ( pour zone )

```
1 void Move(float* y, int* yvel, int* z){  
2     switch(*z){  
3         case 1 :  
4             *yvel=-SPEED;  
5             if (*y<=2*wid/N){  
6                 *y=2*wid/N;  
7                 *yvel=0;  
8             }  
9             break;  
10        case 2 :  
11            *yvel=SPEED;  
12            if (*y>=wid-wid/N){  
13                *y=wid-wid/N;  
14                *yvel=0;  
15            }  
16            break;  
17        default :  
18            *yvel=0;  
19    }  
20}
```

Listing 2.3 – Code Source de la fonction Move()

### 2.3.1.2 La fonction verifier()

La fonction verifier() detecte la forme de l'objet actuel, elle prend en parametre un tableau de taille N remplie par l'utilisateur; ce tableau est de type int, ou est stockee des valeurs de type shape, definie par une enumeration. Elle retourne 1 si elle detecte un cercle sinon renvoie 0.

```
1 //creation variable forme  
2 typedef enum{  
3     RECT,CERCLE,TRIANGLE,CARRE  
}shape;  
5  
6  
7 int verifier(int T[],int k){  
8     if (T[k]==CERCLE) return 1;  
9     else return 0;  
10}
```

Listing 2.4 – Code Source de la fonction verifier()

### 2.3.1.3 La fonction moveObject()

La fonction moveObject() gere le mouvement des objets, vu qu'on a stocke les formes dans un tableau de type "SDL\_Rect" on repere la position de la forme a deplacer a l'aide d'un parametre k qui indique la colonne ou se trouve le Robot, le tableau P quant a lui prend deux valeur : 0 ou 1 indiquant si on va deplacer l'objet ou non (si cercle ou non). La premiere condition permet aux formes de rester dans la fenetre, et la deuxieme decale le Robot a droite.

```
1 void moveObject(SDL_Rect rect[],int P[],int k,int* z,float* x){ // k = colonne actuelle du  
2     Robot  
3     rect[k].y+=SPEED*P[k]/60;  
4     if (rect[k].y>hei-2*hei/N){  
5         P[k]=0;  
6         rect[k].y=hei-2*hei/N;  
7         if (k==9) *z=0;  
8         else *z=1;  
9         if (*x<wid-wid/N)  
10            *x+=wid/N;  
11            SDL_Delay(500);  
12    }
```

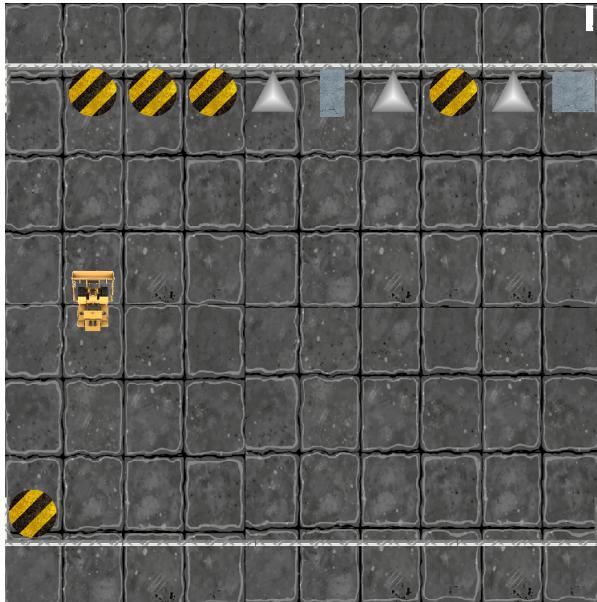
Listing 2.5 – Code Source de la fonction moveObject()

### 2.3.2 Le compteur

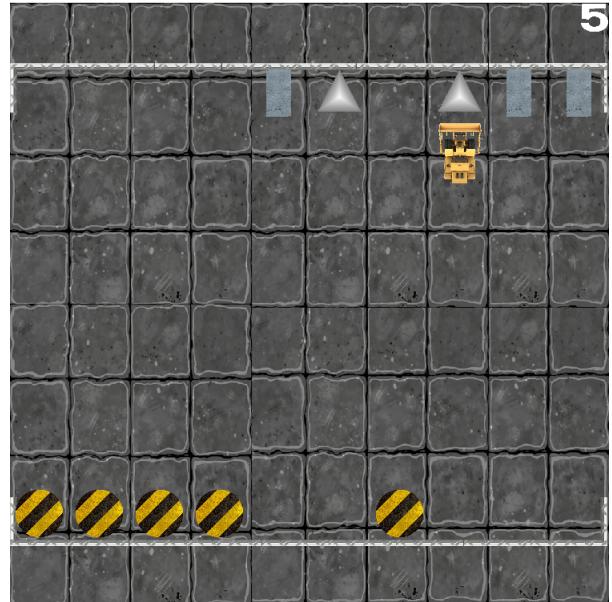
Le compteur permet de compter, au fur et a mesure du deplacement du Robot, le nombre de cercles detecte. A chaque cercle detecte le compteur s'incremente, et grace au switch on place l'image convenante a la valeur du compteur sur la fenetre.

```
1 counter++;
2 switch(counter){
3     case 1 :
4         texCOMPTEUR = SDL_CreateTextureFromSurface(rend, imgN1);
5         SDL_FreeSurface(imgN1);
6         SDL_QueryTexture(texCOMPTEUR, NULL, NULL, &compteur.w, &compteur.h);
7         break;
8     case 2 :
9         texCOMPTEUR = SDL_CreateTextureFromSurface(rend, imgN2);
10        SDL_FreeSurface(imgN2);
11        SDL_QueryTexture(texCOMPTEUR, NULL, NULL, &compteur.w, &compteur.h);
12        break;
13     case 3 :
14         texCOMPTEUR = SDL_CreateTextureFromSurface(rend, imgN3);
15         SDL_FreeSurface(imgN3);
16         SDL_QueryTexture(texCOMPTEUR, NULL, NULL, &compteur.w, &compteur.h);
17         break;
18     case 4 :
19         texCOMPTEUR = SDL_CreateTextureFromSurface(rend, imgN4);
20         SDL_FreeSurface(imgN4);
21         SDL_QueryTexture(texCOMPTEUR, NULL, NULL, &compteur.w, &compteur.h);
22         break;
23     case 5 :
24         texCOMPTEUR = SDL_CreateTextureFromSurface(rend, imgN5);
25         SDL_FreeSurface(imgN5);
26         SDL_QueryTexture(texCOMPTEUR, NULL, NULL, &compteur.w, &compteur.h);
27         break;
28     case 6 :
29         texCOMPTEUR = SDL_CreateTextureFromSurface(rend, imgN6);
30         SDL_FreeSurface(imgN6);
31         SDL_QueryTexture(texCOMPTEUR, NULL, NULL, &compteur.w, &compteur.h);
32         break;
33     case 7 :
34         texCOMPTEUR = SDL_CreateTextureFromSurface(rend, imgN7);
35         SDL_FreeSurface(imgN7);
36         SDL_QueryTexture(texCOMPTEUR, NULL, NULL, &compteur.w, &compteur.h);
37         break;
38     case 8 :
39         texCOMPTEUR = SDL_CreateTextureFromSurface(rend, imgN8);
40         SDL_FreeSurface(imgN8);
41         SDL_QueryTexture(texCOMPTEUR, NULL, NULL, &compteur.w, &compteur.h);
42         break;
43     case 9 :
44         texCOMPTEUR = SDL_CreateTextureFromSurface(rend, imgN9);
45         SDL_FreeSurface(imgN9);
46         SDL_QueryTexture(texCOMPTEUR, NULL, NULL, &compteur.w, &compteur.h);
47         break;
48 }
```

Listing 2.6 – Code Source de la fonction moveObject()



(a) Compteur a 1



(b) Compteur a 5

FIGURE 2.6

### 2.3.3 Le chemin

A la fin du parcours du robot, et a l'aide d'un bouton-bascule son chemin s'affiche.

```

1 void draw(SDL_Renderer *rend, int rr,SDL_Point points[],int k, int z,int T[]){
2     if (k==9 && T[N-1] == 1 && z==0){
3         //dessin chemin
4         SDL_RenderDrawLine(rend,wid/N/2,hei/2-hei/(2*N),wid/N/2,5*hei/N/2);
5         SDL_RenderDrawLines(rend,points,rr-1);
6     }
7     else if (k==9 && T[N-1] != 1 && z==0){
8         //dessin chemin
9         SDL_RenderDrawLine(rend,wid/N/2,hei/2-hei/(2*N),wid/N/2,5*hei/N/2);
10        SDL_RenderDrawLines(rend,points,rr);
11    }
12 }
13
14 void storePoints(SDL_Point points[],int *bb,int *rr,int choix){
15     if (choix == 1){
16         points[*rr+1].x=(2*(*bb)+1)*(wid/N/2)+2;
17         points[*rr+1].y=5*hei/N/2;
18         points[*rr+2].x=(2*(*bb)+1)*(wid/N/2)+2;
19         points[*rr+2].y=hei-3*hei/N/2;
20         points[*rr+3].x=(2*(*bb)+3)*(wid/N/2)-2;
21         points[*rr+3].y=hei-3*hei/N/2;
22         points[*rr+4].x=(2*(*bb)+3)*(wid/N/2)-2;
23         points[*rr+4].y=5*hei/N/2;
24         (*bb)++;
25         *rr+=4;
26     }
27     else if (choix == 0){
28         points[*rr+1].x=(2*(*bb)+3)*(wid/N/2);
29         points[*rr+1].y=5*hei/N/2;
30         *rr+=1;
31         (*bb)++;
32     }
33 }
34 }
```

Listing 2.7 – Code Source des fonctions manipulons le tracage du chemin

Le chemin est trace a l'aide d'une fonction SDL qui s'appelle "SDL\_DrawLines()" qui prend en parametres un tableau de type "SDL\_Point" regroupant les diffenrent points du Robot stockes par la fonction storePoints().

## 2.4 Interface du relancement

A la fin du mouvement du Robot, dans sa première version, une interface "RESTART" apparut demandant à l'utilisateur s'il veut recommencer ou pas. Cette interface est munie de deux boutons "YES" et "NO". Son principe de fonctionnement est exactement le même que celui du bouton "START".

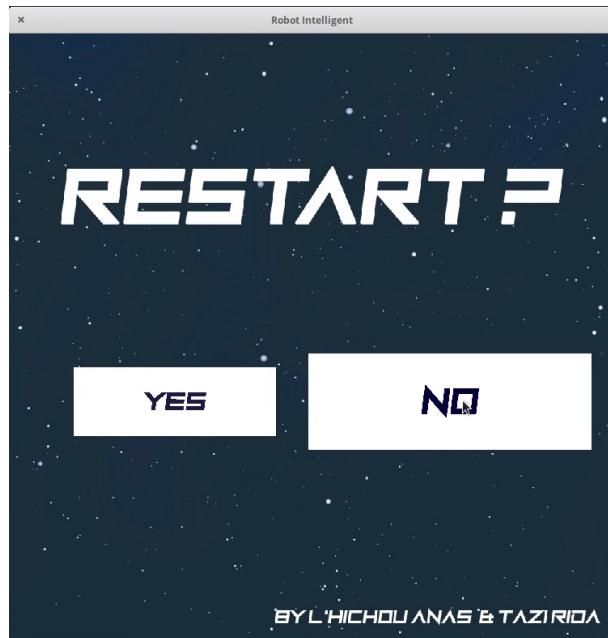


FIGURE 2.7 – menu RESTART

### 2.4.1 Robot intelligent 2.0

Vue qu'après que robot finisse de tracer son chemin le menu "RESTART" apparut immédiatement, on a dû penser à intégrer deux boutons sur la même page juste après l'arrêt du Robot.

Le bouton "EXIT" : Pour quitter le programme.

Le bouton "RESET" : Pour recommencer.

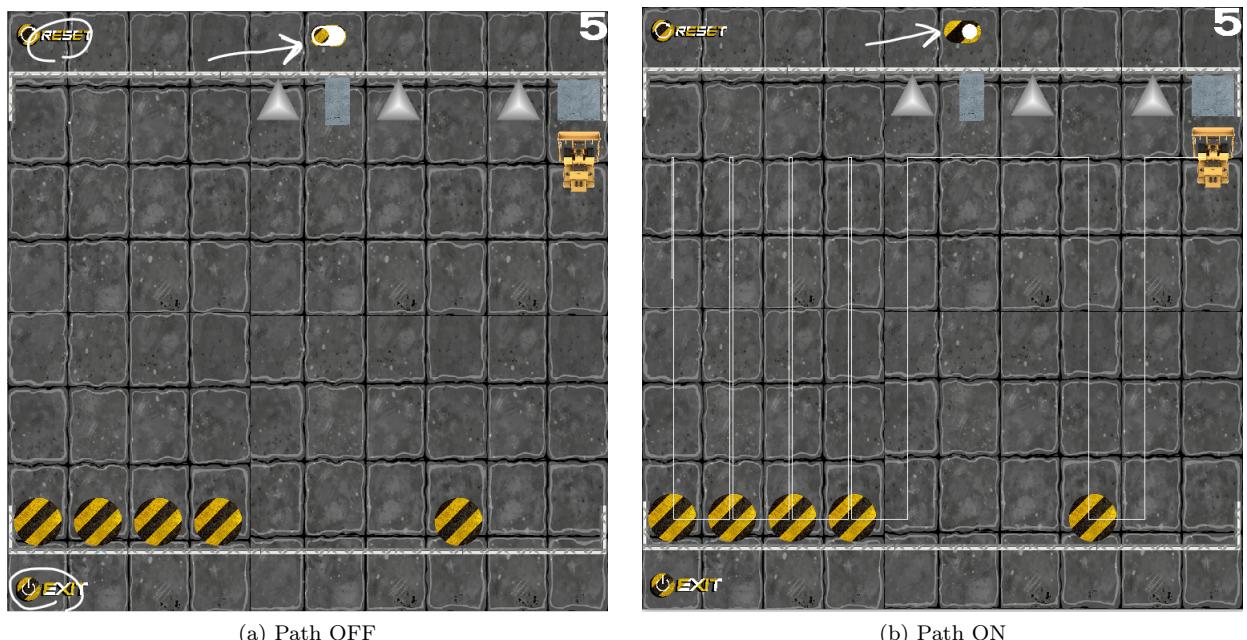


FIGURE 2.8