**But :**

Le but technique de ce PTI est de mettre en place une solution pour gérer le fonctionnement de notre jeu, en évitant les collisions entre les pions et de contrôler les autorisations de déplacement lors du jeu.

**Principe :**

Au début du jeu, nous instancions un echiquier de 8x8 cases alternées blanches & noir, puis nous y placons des pions sur 2 rangés par joueur(2 max.). Dans une boucle infini, nous demandons la position du pion a bouger ainsi que la position apres deplacement. L’ordinateur vérifiera si la case sélectionner comporte un pion appartenant au proprietaire, si la destination est autorisé (taille deplacement ou presence d’un pion du proprietaire post deplacement).
Le pion adverse sera manger lorsque notre pion arrivera sur la case du pion adverse

**Mise en œuvre :**

Pour mettre en œuvre mon projet, j’ai utilisé divers outils :

* Logiciel RAD (Rapid Application Développement) Dev-C++
* langage C

Pour le fonctionnement du projet, nous avons du créer plusieurs modules :

* Module de gestion de la console(entree, sorties) : Ce sont toutes nos fonctions contrôlant l’affichage console de l’echiquier et des demande de saisies diverses ex : void getpions(struct echiquier \*ech) , void saisiemouvementusr(struct echiquier \*\*ech, position \*posf, position \*posto)
* Module de contrôle de deplacement :ce sont les fonction permettant de controler si le deplacement est autorisé ou non. Ex : int isproprietaire(struct echiquier \*ech,int color,int w,int h)
* Les initialisateur : ce sont les fonctions permettant d’initailisé notre echiquier comme poser les couleur de l’echiquier et les pions. Ces fonctions servent uniquement a initialiser, il n’instancie pas la structure. Ex : void initcolor(struct echiquier \*\*ech), void initpions(struct echiquier \*\*ech),…
* Les Setter & les Getter : ce sont les fonctions qui regarderons notre structure pour nous renvoyer une reponse simple a une demande, pour ne pas avoir a decortiquer la structure et son fonctionnement pour connaître une toute petite information. Ex : void setblack(caseechec \*\*cst), int getcolorcase(caseechec \*cst),…

**Conclusion :**

Ce projet m’a permis de développer mes connaissances dans le langage C et particulièrement le fonctionnement des pointeurs.

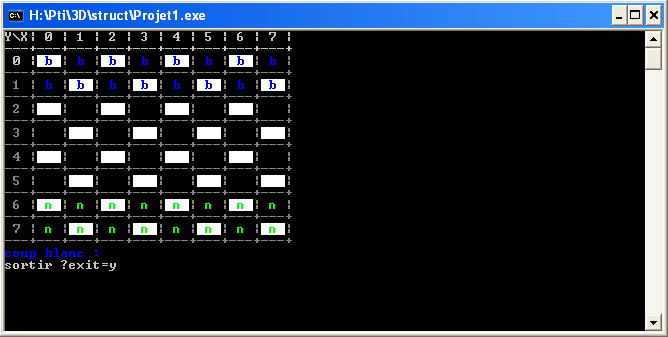


Fig. 1 : Imprime écran du jeu en mode console



Fig. 1 : fichier générer du jeu en mode OpenGL (extension en cours de développement)

Extrait de code :

//structure 2 entier pour les positions
typedef struct poscas{
unsigned short int positionH;
unsigned short int positionW;
}position;
//case a 2 variables 1 0 pour noir 0 1 pour blanc 0 0 pour vide
typedef struct cs {
int n;//noir
int b;//blanc
}caseechec;
///structure de notre echiquier
typedef struct echiquier{
//la liste des pions cpu pour le traitement
pioncpu \* pionsCPU;
//le tableau de case de l'echiquier //soit noir , soit blanc
caseechec caseunit[NBCASE][NBCASE];
//le tableau de contenu des cases de l'echiquier //soit noir , soit blanc
caseechec casecontenu[NBCASE][NBCASE];
// int \*getcolor(int x)=getcol(x);
// void \*remplircouleur()=initcolr();
}echiquiers;
//mise des etats de la structure a 0 1 (blanc)
void setblanc(caseechec \*\*cst){
((\*cst)->n)=0;
((\*cst)->b)=1;
}
//retour rapide des configurations de la structure
// 1->noir
// -1->blanc
// 0->vide
int getcolorcase(caseechec \*cst){
if (cst->n==1) return 1;
else if (cst->b==1)return -1;
else return 0;
}
//constructeur du contenue des cases
void initpions(struct echiquier \*\*ech){
int i,y,last=0;
for (i=0; i<NBCASE;i++){
//pour chaque case sur x
for (y=0;y<NBCASE;y++){
//pour chaque case Y(x)
caseechec \* castmp=&((\*ech)->casecontenu[i][y]);
//si la ligne correspond a l'empl d'un pion a placer il modifie
// l'etat du contenu des cases
if (i<NBLIGNETOFILL) setblanc(&castmp);
else if(i>=NBLIGNETOFILL && i<(NBCASE-NBLIGNETOFILL))setempty(&castmp);
else setblack(&castmp);
}
}
}