Rapport de programmation :

Arène

Pailler Gaëtan
Pasco-Castillo Marc
Subirats Johann
Mouyebe Téo
Ngo Jean-Etienne

Introduction :

Nous devions réaliser une arène dans le cadre de notre projet d'informatique. Cette arène devait pouvoir contenir 32 joueurs afin de réaliser un tournoi entre différentes IA codées par les autres groupes de notre promotion.

Nous avons donc décidé de créer un labyrinthe dans lequel les IA doivent échapper à un Pac-Man. L'arène est composée d'un labyrinthe avec des murs que les joueurs ne pourront pas franchir, d'un Pac-Man qui se déplace de façon aléatoire avant que ce-dernier ne détecte un joueur et décide de le pourchasser pour le manger. Par contre, le pac-man lui peut passer les murs. Une fois que le joueur est mangé il réapparaîtra de manière aléatoire sur la carte en tant que Pac-Man. Ainsi, plus il y a de joueurs mangés plus il y aura de Pac-Man dans l'arène.

Ensuite, les joueurs ne se déplaçant pas ou restant toujours dans la même zone de l'arène se retrouveront infectés et éliminés par la même occasion.

Au début de la partie les joueurs vont se retrouver sur les bords de l'arène et Pac-Man au centre de cette dernière. Pour échapper aux monstres de l'arène les joueurs pourront utiliser des bonus afin de les avantager dans leur survie et améliorer leur expérience de jeu.

Lors des duels, la règle ne sera pas modifiée. C'est le dernier survivant qui l'emporte.

Par contre, il n'y aura plus un mais plusieurs pacmans qui vont apparaître aléatoirement sur la carte au début de la partie.

Présentation de notre code

```c
#include <stdio.h>
#include <stdlib.h>
#include <SDL/SDL.h>
#include  <time.h>

#define LARGEUR_TILE 16  // haute
#define HAUTEUR_TILE 16

#define NOMBRE_BLOCS_LARGEUR 108
#define NOMBRE_BLOCS_HAUTEUR 44
```

Tout d'abord, avant de se lancer dans le code, nous avons réalisé notre jeu à partir de la SDL. Il faut donc inclure les fichiers SDL sinon notre code ne pourra être lue et exécutée par la suite.

Ensuite nous définissons le nombre de pixels de nos carrés qui sont de 16 par 16.

Puis, on définit la taille de notre arène.

```
char* table[] = {
"11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111",
"10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001",
"10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001",
"10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001",
"10001111111000000111111111111111111000111111111111110001111111111111100011111111111111111111000001111111000111",
"10001222221000100012222222222222210001222222222221000122222222222210001222222222222221000100012222210001",
"10001222221000110001222222222222210001222222222221000122222222222210001222222222222221000110001222210001",
"10001222221000121000122222222222210001111111111111000111111111111110001222222222222210001210001222210001",
"10001222221000122100012222222222210000000000000000000000000000000001222222222221000122100012222210001",
"10001222221000122100012222222222210000000000000000020000000000000001222222222221000122210001222210001",
"10001222221000122221000122222222210000000000000000000000000000000001222222222210001222210001222210001",
"10001222221000122222100012222222210001111111111111000111111111111110001222222222100012222100012222210001",
"10001222221000122222210001222222210001222222222210000000000000122222222210001222222100012222221000122210001",
"10001222221000122222221000122222210001222222222210000000122222222221000122222210001222222210001222210001",
"10001222221000122222222100012222210001222222222100000000122222222221000122222100012222222100012222210001",
"10001222221000122222222210001222210001222222221000000000122222222221000122221000122222222210001222210001",
"10001222221000111111111111110001111000111111100000000100000000111111100011110001111111111111000122221000 1",
"10001111111000000000000000000000000000000000000000011100000000000000000000000000000000000000011111110001",
"10000000000000000000000000000000000000000000000000111110000000000000000000000000000000000000000000000001",
"10000000000000000000000000000000000000000000000000111000000000000000000000000000000000000000000000000001",
"10000000000000000000100011111111111111100011000000010000000011000111111111111110001000000000000000000001",
"10001100011111111110001100012222222221000122100000000000000001221000122222222210001100011111111110001100 1",
"10001210001222222210001210001222222221000122221000000000000012222100012222222210001210001222222210001210001",
"10001221000122222210001221000122222210001222222100000000000012222210001222222100012210001222221000122100 1",
"10001222100012222210001222100012222100012222222100000000001122222221000122221000122210001222221000122210001",
"10001222210001222210001222210001221000122222222211000000012222222222100012100012222100012222100012222100 1",
"10001222221000122210001222221000110001222222222222110000012222222222221000100012222210001222210001222210001",
"10001222222100012210001222222100000012222222222222100012222222222222210000001222221000122100012222221000 1",
"10001222222210001210001222222221000122222222222222210001222222222222222100011222222100012100012222222100 1",
```

Ici, nous avons notre arène compose de 0, de 1 et de 2. Chaque chiffre correspond à un bloque de couleur différent ce qui va permettre d'avoir une jolie image en arrière-plan.

```
void Afficher(SDL_Surface* screen,SDL_Surface* tileset,char** table,int nombre_blocs_largeur,int nombre_blocs_hauteur)
{
    int i,j;
    SDL_Rect Rect_dest;
    SDL_Rect Rect_source;
    Rect_source.w = LARGEUR_TILE;
    Rect_source.h = HAUTEUR_TILE;
    for(i=0;i<nombre_blocs_largeur;i++)
    {
        for(j=0;j<nombre_blocs_hauteur;j++)
        {
            Rect_dest.x = i*LARGEUR_TILE;
            Rect_dest.y = j*HAUTEUR_TILE;
            Rect_source.x = (table[j][i]-'0')*LARGEUR_TILE;
            Rect_source.y = 0;
            SDL_BlitSurface(tileset,&Rect_source,screen,&Rect_dest);
        }
    }
    SDL_Flip(screen);
}
```

Cette fonction comme son nom l'indique permet d'afficher notre arrière-plan.

Le flip permet d'actualiser notre fenêtre.

```
SDL_Surface *screen = NULL ,*tileset = NULL,*pacman1 = NULL
SDL_Rect positionTileset, positionPacman1;
SDL_Event event;

int continuer = 1;
```

Nous avons, par la suite, initialisé nos différentes variables. Le « int continuer = 1 » permet de savoir uniquement s'il faut fermer le programme ou non.

```
positionPacman1.x = 500;
positionPacman1.y = 260;
```

Nous initialisons les variables positions.

```
}
SDL_Init(SDL_INIT_VIDEO);
```

La SDL_INIT_VIDEO permet d'initialiser la SDL au niveau de l'affichage graphique

```
screen = SDL_SetVideoMode(LARGEUR_TILE*NOMBRE_BLOCS_LARGEUR, HAUTEUR_TILE*NOMBRE_BLOCS_HAUTEUR, 32,SDL_HWSURFACE|SDL_DOUBLEBUF);
```

C'est ce permet de paramétrer la surface qui servira de fenêtre qui prend quatre valeurs : l'axe des x, l'axe des y, le nombre de bits auquel le programme doit tourner et le type de mémoire utilisé.

```
tileset = SDL_LoadBMP("tileset1.bmp");

pacman1 = SDL_LoadBMP("pacman1.bmp");
SDL_SetColorKey(pacman1, SDL_SRCCOLORKEY, SDL_MapRGB(pacman1->format,255,255,255));
```

Nous faisons afficher notre arrière-plan et nous copions l'image sur la SDL du pacman ensuite. Ici, notre srect est nulle donc toute la surface est copiée.

La clé de couleur définit une valeur de pixel qui sera traité comme transparent dans un blit. Le Pacman n'a plus de contour blanc autour de lui.

```
SDL_EnableKeyRepeat(10, 10);
```

Cette ligne permet d'activer la répétition des touches du clavier. Le premier 10 correspond aux délais avant de pouvoir réappuyer sur une touche et le deuxième 10 est l'intervalle de temps. Ces deux valeurs sont bien sûr exprimées en millisecondes.

```
while (continuer)
```

Le while fait tourner en boucle le cœur de notre programme.

```
SDL_WaitEvent(&event);
```

C'est une fonction qui permet de prendre en compte les évènements du clavier mais qui ne fait rien si elle ne reçoit pas d'instructions.

```
Afficher(screen,tileset,table,NOMBRE_BLOCS_LARGEUR,NOMBRE_BLOCS_HAUTEUR);

SDL_BlitSurface(pacman1, NULL, screen, &positionPacman1);
```

Cela permet d'afficher notre arrière-plan puis notre pacman.

```
switch(event.key.keysym.sym)
```

C'est un switch qui reçoit les entrées du clavier qu'on a conditionné avec des cases pour aller en haut, en bas, à gauche et à droite.

```
case SDL_QUIT:
    continuer = 0;
    break;
```

Cela signifie que la boucle while sera finie.

```
SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255, 255, 255));
```

Cette fonction permet de colorer la surface écran en blanc.

```
SDL_FreeSurface(pacman1);
```

Cette ligne permet de libérer en mémoire la place utilisée par pacman.

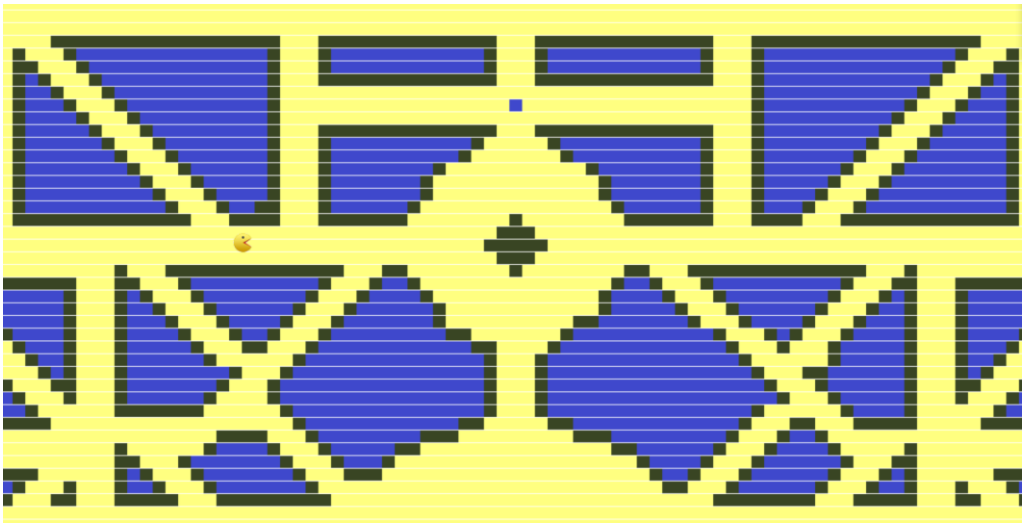
```
SDL_Quit();
```

Cette fonction permet de désallouer tous les éléments de la SDL qui avaient été initialisés précédemment.

```
return EXIT_SUCCESS;
```

Cette dernière ligne montre que le programme s'est bien terminé.

Nous sommes des novices en programmation et nous avons l'intention de mettre des murs et pour cela il faut coder la carte à la main avec des 1 pour passer et des 0 pour être bloquer par le mur. On a essayé avec une boucle if mais le travail sur les murs est inachevé et ne fonctionne pas malgré des tentatives.



Copyrigths : image de Pac-Man
http://www.etaletaculture.fr/wpcontent/uploads/2013/11/pacman.jpg