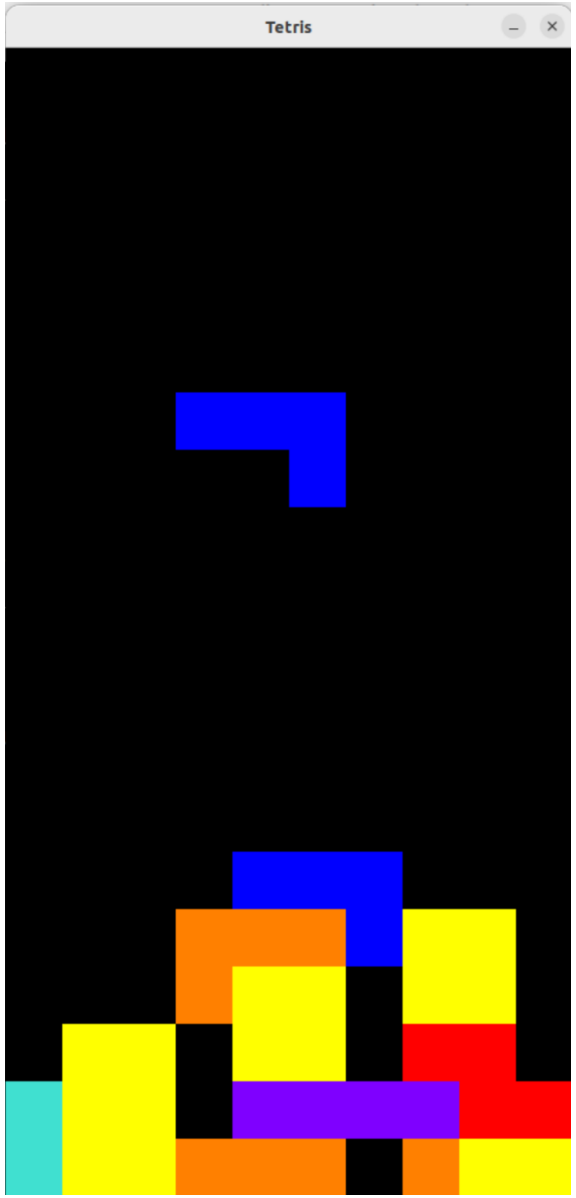
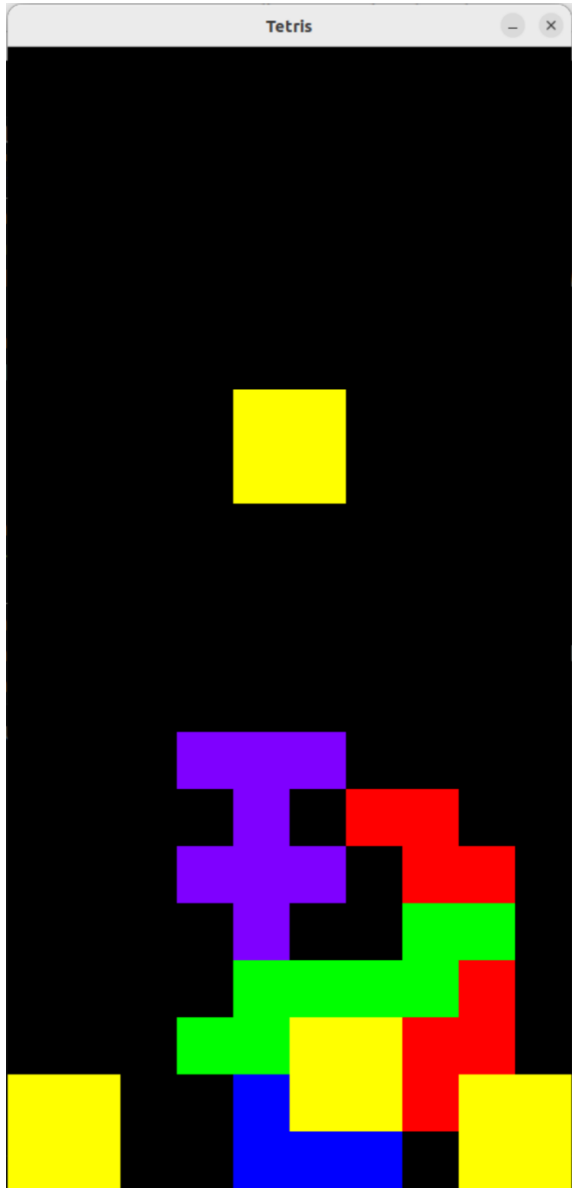
2D Tetris Game

Edouard CLOCHERET and Louis PAILLONCY

Contents

Result



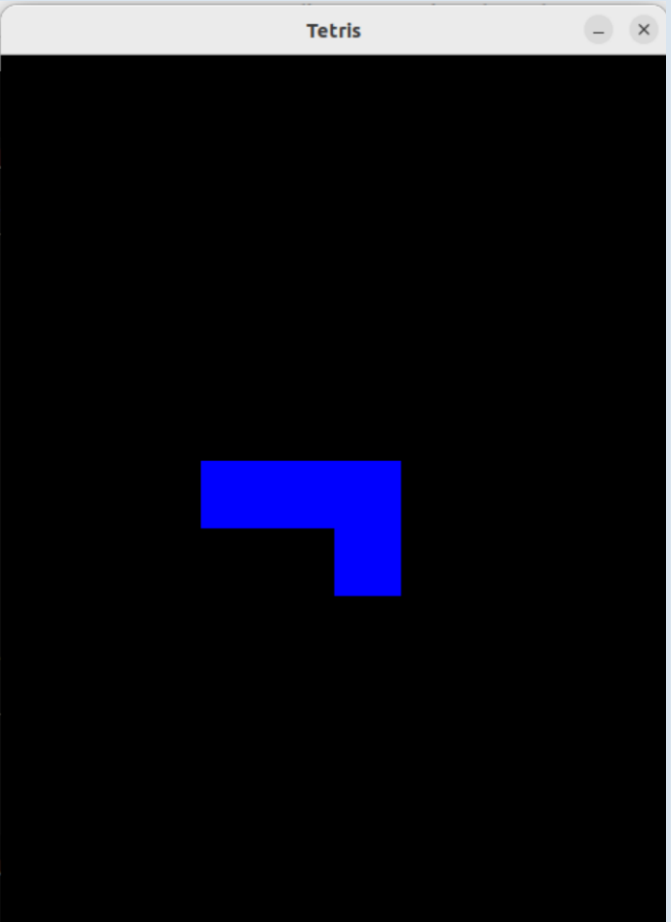


What is the main loop ?

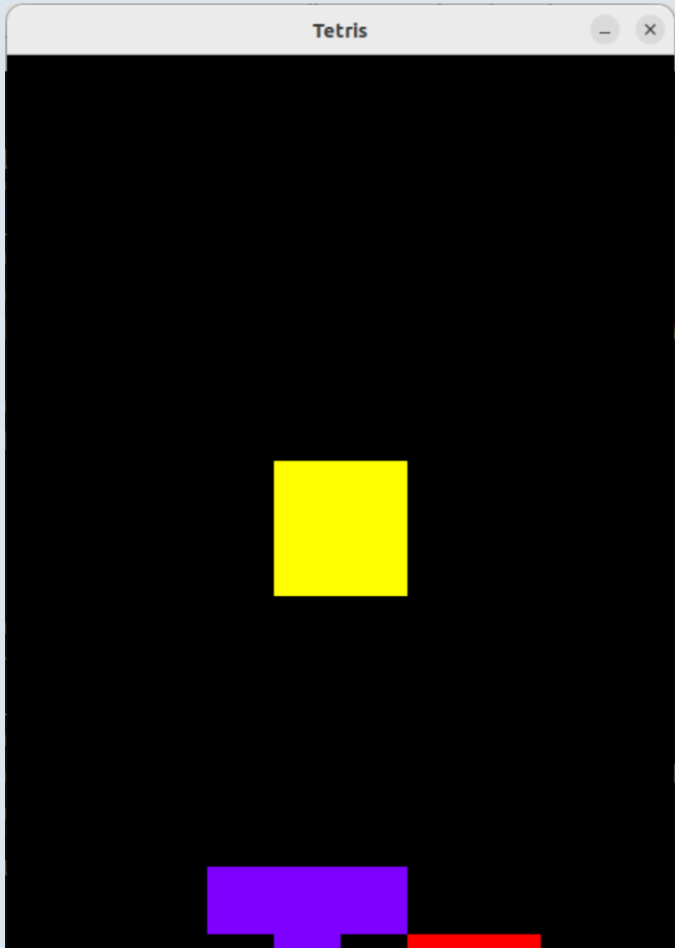
Gaming loop

```
//on utilise ce bloc jusqu'à ce qu'il touche le sol
while((!quit) &&(collision(&falling_meteor, matrice)==0 )&& (!game_over(matrice))){
    printf("boucle\n");

    //faire descendre le bloc
    falling_meteor.y=falling_meteor.y+going_down;

    //Mettre à jour l'affichage
    pre_render(&pRenderer,&falling_meteor, matrice);

    //Mettre à jour le jeu selon le joueur
    entree_clavier (&falling_meteor, &event, &quit, matrice);
    test_ligne_complete(matrice);
    SDL_Delay(16);

    printf("position du bloc : %d , %d\n",falling_meteor.x, falling_meteor.y);
}
```

Random draw of blocs



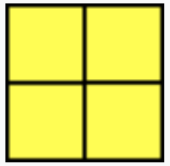
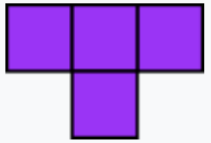
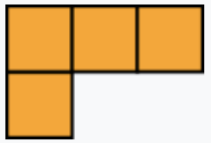
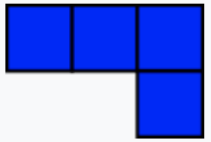
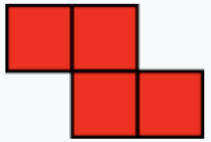
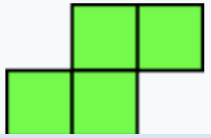

```
//tirage d'un bloc
//tirage au sort du bloc qui va apparaître :
int etat_tire = rand() % 7 + 1;
//int etat_tire = 1;
if (etat_tire ==1) {falling_meteor.son_nom = I;
else if (etat_tire ==2) falling_meteor.son_nom = O;
else if (etat_tire ==3) falling_meteor.son_nom = T;
else if (etat_tire ==4) falling_meteor.son_nom = L;
else if (etat_tire ==5) falling_meteor.son_nom = J;
else if (etat_tire ==6) falling_meteor.son_nom = Z;
else if (etat_tire ==7) falling_meteor.son_nom = S;

falling_meteor.x = taille_carreau*3;
falling_meteor.y = 0;
falling_meteor.rotation = 0;
```



What is a Tetris Bloc ?

Definition of a bloc

```c
//A name is given to each tetris bloc
enum shape {I, O, T, L, J, Z, S};

//Refer to bloc_names.png to see every b
typedef struct bloc {
    enum shape son_nom;
    //position of the most left and high
    int x;
    int y;
    //number of clockwise rotations from
    int rotation;
} bloc ;
```

Image	Forme
	I-Tetrimino
	O-Tetrimino
	T-Tetrimino
	L-Tetrimino
	J-Tetrimino
	Z-Tetrimino
	S-Tetrimino

How to test collisions ?

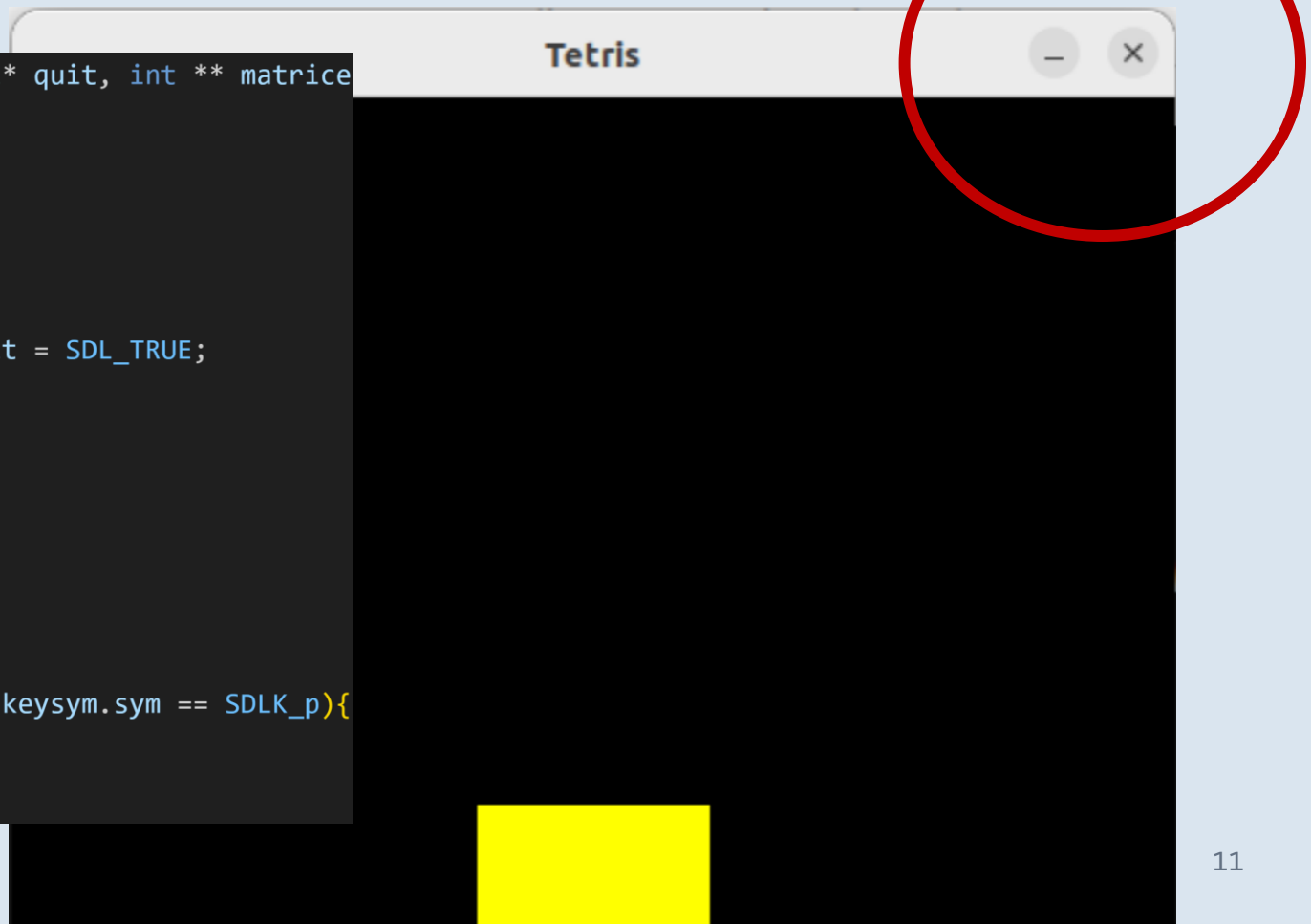
```
void translate_bloc_to_positions(bloc* falling_meteor, int*x1,int*y1, int*x2,int*y

    switch (falling_meteor->son_nom){
        case I :
            if (falling_meteor->rotation ==0||falling_meteor->rotation ==2){
                *x1 = falling_meteor->x/taille_carreau;
                *x2 = *x1+1;
                *x3 = *x2+1;
                *x4 = *x3+1;
                //on prend la partie entière parce qu'il n'y a pas de collision
                // si on occupe une demie- case
                *y1 = falling_meteor->y/taille_carreau;
                *y2 = *y1;
                *y3 = *y1;
                *y4 = *y1;
            }
```

Translate a bloc into the 4 positions to test

How to interact with the program ?

User interaction

```c
void entree_clavier (bloc* falling_meteor, SDL_Event* event, SDL_bool* quit, int ** matrice

    while (SDL_PollEvent(event)!=0) {

        switch(event->type) {

            case SDL_WINDOWEVENT:
                if (event->window.event ==SDL_WINDOWEVENT_CLOSE) *quit = SDL_TRUE;
            break;

            case SDL_QUIT :
            if(event->type == SDL_QUIT) *quit = SDL_TRUE;
            break;

            case SDL_KEYDOWN:

                if (event->key.keysym.sym == SDLK_ESCAPE||event->key.keysym.sym == SDLK_p){
                    *quit = SDL_TRUE;
                }
```



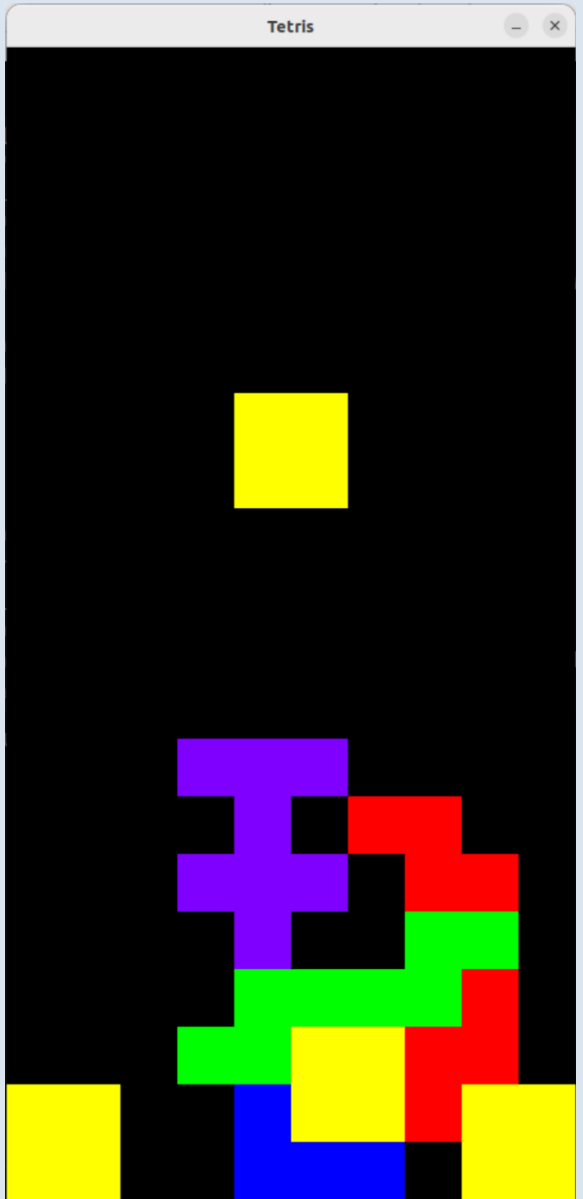
Keyboard entries & tests

```
if (event->key.keysym.sym == SDLK_q){

    rotation(-1, falling_meteor);
    //on vérifie que le bloc ne sorte pas du cadre
    int x1; int y1;
    int x2; int y2;
    int x3; int y3;
    int x4; int y4;
    translate_bloc_to_positions(falling_meteor, &x1,&y1, &x2,


    //si l'action n'est pas autorisée on l'annule
    if (x1<0||x2<0||x3<0||x4<0
    ||x1>9 || x2>9 || x3>9 || x4>9
    ||y1>19 ||y2>19||y3>19||y4>19
    ||y1<0||y2<0||y3<0||y4<0){
        //on annule si on sort du cadre
        rotation(1, falling_meteor);
    }
    else if (matrice[y1][x1]!=0 ||matrice[y2][x2]!=0
    ||matrice[y3][x3]!=0||matrice[y4][x4]!=0){
        //on annule si la case est déjà occupée
        rotation(1, falling_meteor);
    }
```

What about the display ?

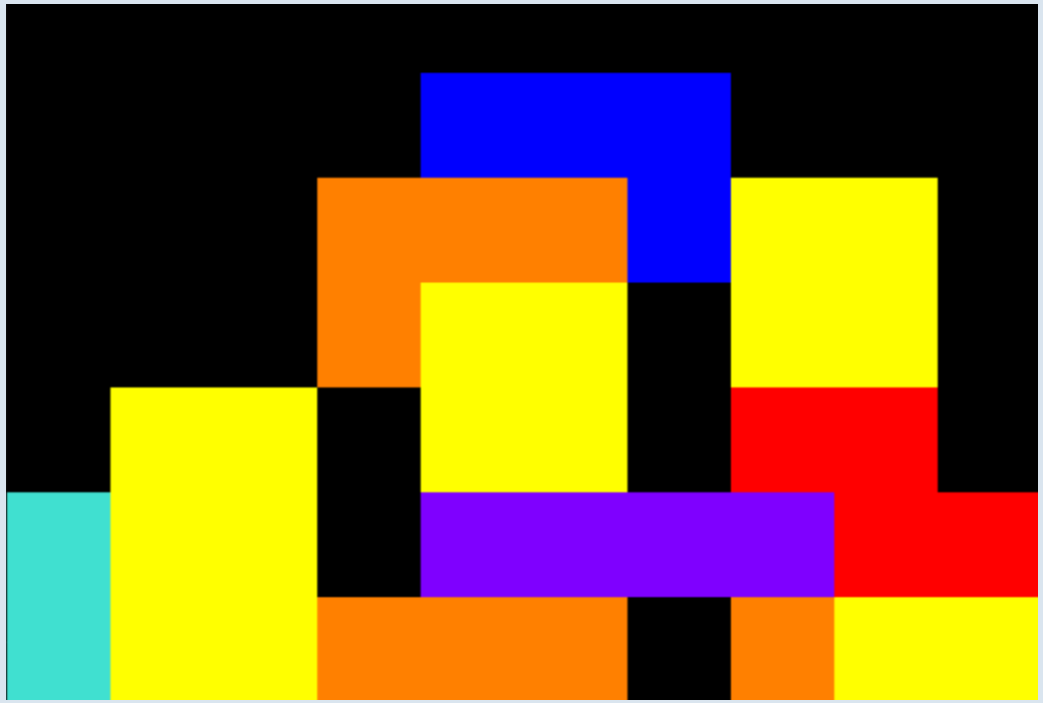
Displaying the falling & still blocs

```c
void pre_render(SDL_Renderer** pRenderer, bloc * falling_meteor,
    printf("remise à zero du rendu\n");
    //On remet d'abort à zero le rendu
    SDL_SetRenderDrawColor(*pRenderer, 0, 0, 0, 255); //noir
    SDL_RenderClear(*pRenderer);



    SDL_Rect rectangle;
    SDL_Rect rectangle2;

    switch(falling_meteor->son_nom){ ···


//dessin des blocs déjà tombés
draw_matrix(pRenderer,matrice, falling_meteor);

// Mise à jour du rendu
SDL_RenderPresent(*pRenderer);

}
```



Testing for points or game over

```
//the falling bloc will be labeled "falling_meteor"
bool game_over(int ** matrice){
        if ((matrice[3][0]!=0)||(matrice[3][1]!=0)||(matrice[3
        ||(matrice[3][5]!=0)||(matrice[3][6]!=0)||(matrice[3][
            return true;
        }
        return false;
}

void test_ligne_complete(int ** matrice){
    for(int i=19;i>3;i-=1){
        if(matrice[i][0]!=0 && matrice[i][1]!=0 && matrice[i][
        && matrice[i][4]!=0 && matrice[i][5]!=0 && matrice[i][
            for(int j=i;j>=3;j-=1){
                for (int k =0; k<10; k+=1){
                    matrice[j][k]=matrice[j-1][k];}
```



Challenges

Algoritmic

Invent how to represent a bloc and structure a multi-file program.

Development Environment

Working in a linux environment, witha graphical interface

Using a virtual machine

Chaînes d’approvisionnement

Learning to use a git repository

Thanks

for your attention !