PARIS PANTHÉON-ASSAS UNIVERSITÉ

C Programming Project

Vector Text-based Editor

Mailinh TA - Manon GAUTIER

L1 Int 3 – promo 2027

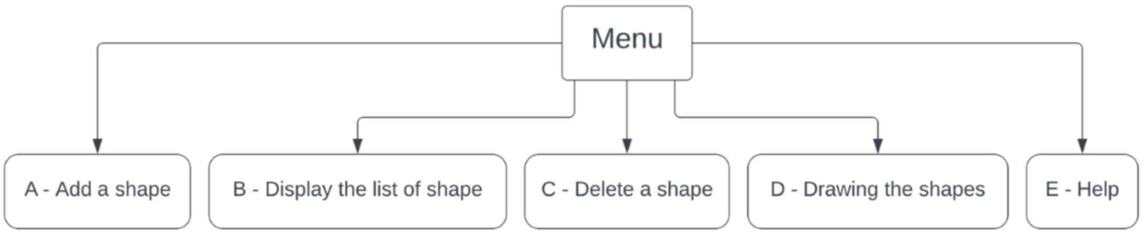
Kais KLAI, Hanen OCHI

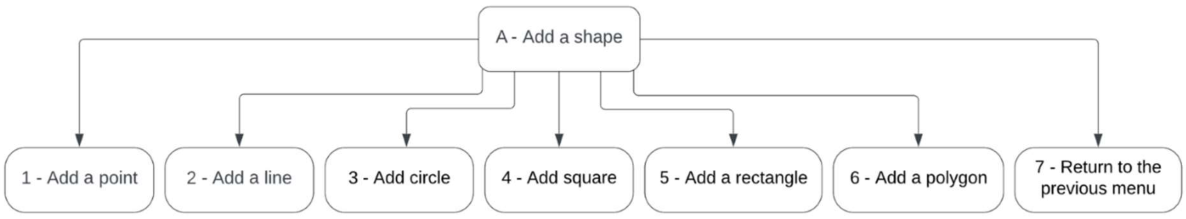
Plan

Introduction

Presentation of the project

In this project, we have the ability to create various shapes such as squares, rectangles, polygons, and more using vector images. Vector images represent the image data from a mathematical perspective, which offers several advantages. One major advantage is the convenience of manipulating lightweight files, and when resizing these images, there is no loss of quality or pixelation effect.
The user can use these different actions:

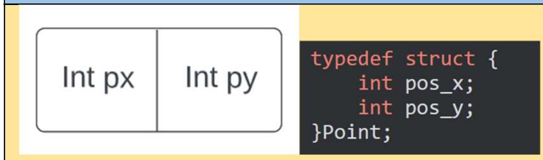




Technical Aspects

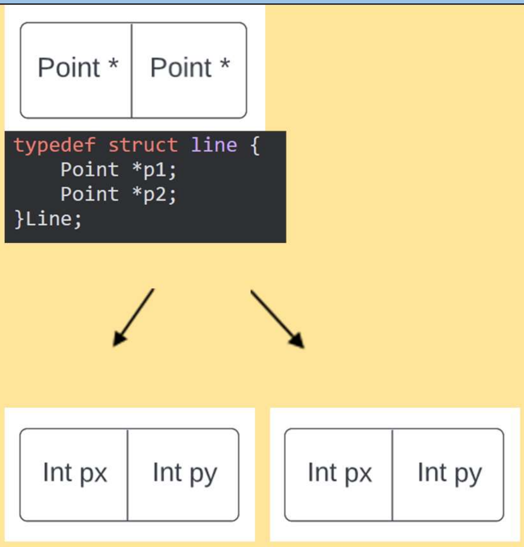
Representation of some structures in memory

Point



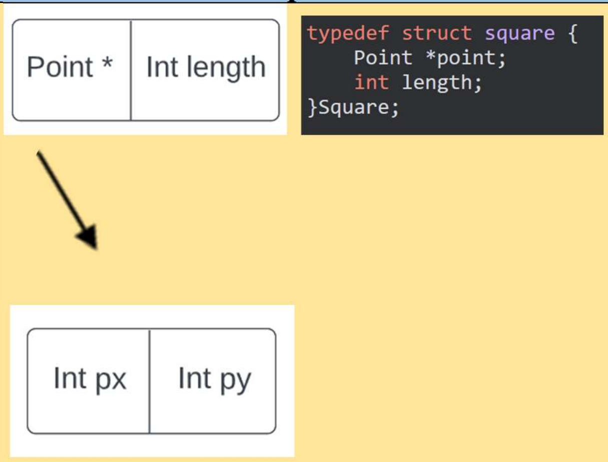
```c
typedef struct {
    int pos_x;
    int pos_y;
}Point;
```

Line



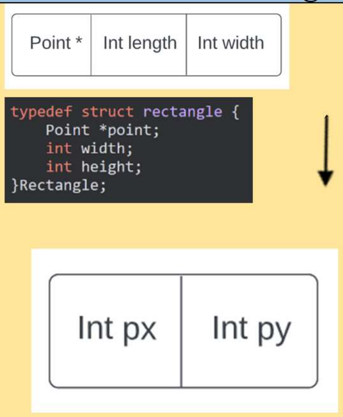
```c
typedef struct line {
    Point *p1;
    Point *p2;
}Line;
```

Square



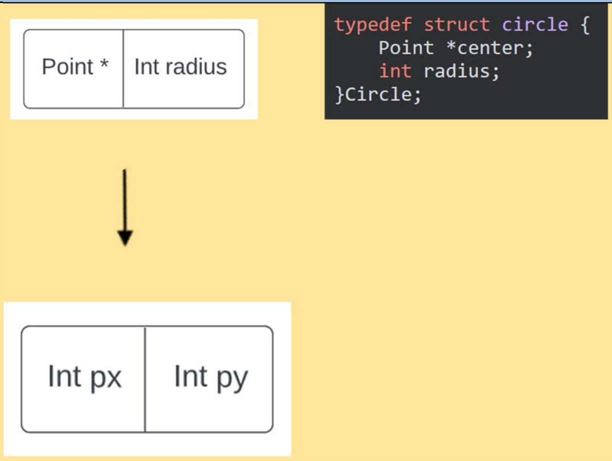
```c
typedef struct square {
    Point *point;
    int length;
}Square;
```

Rectangle



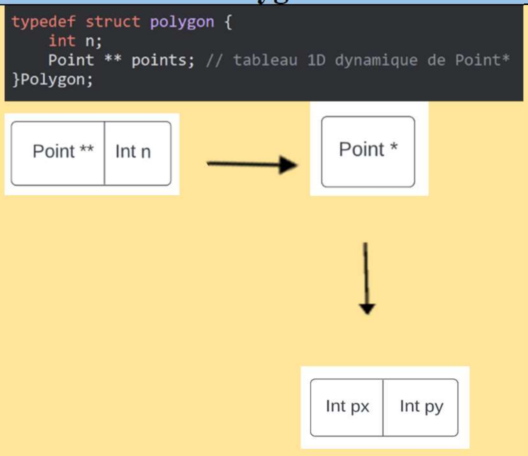
```c
typedef struct rectangle {
    Point *point;
    int width;
    int height;
}Rectangle;
```

Circle



```c
typedef struct circle {
    Point *center;
    int radius;
}Circle;
```

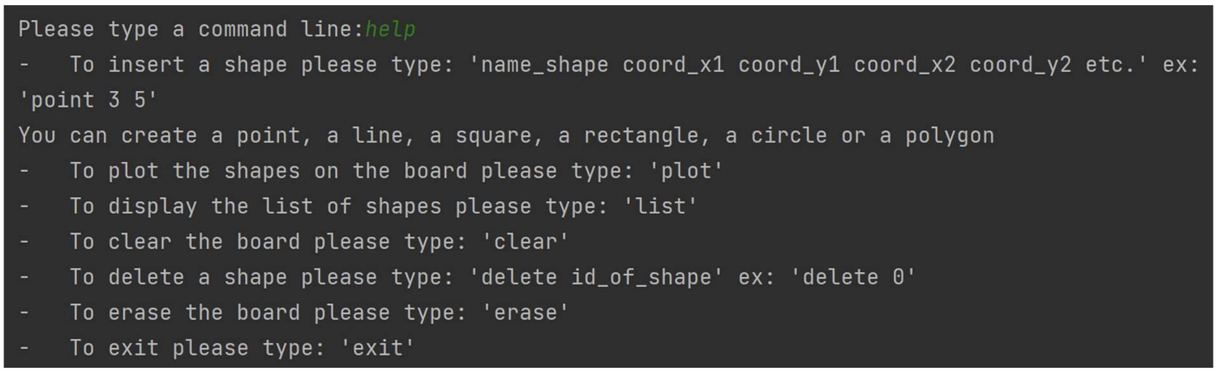
Polygone



```c
typedef struct polygon {
    int n;
    Point ** points; // tableau 1D dynamique de Point*
}Polygon;
```

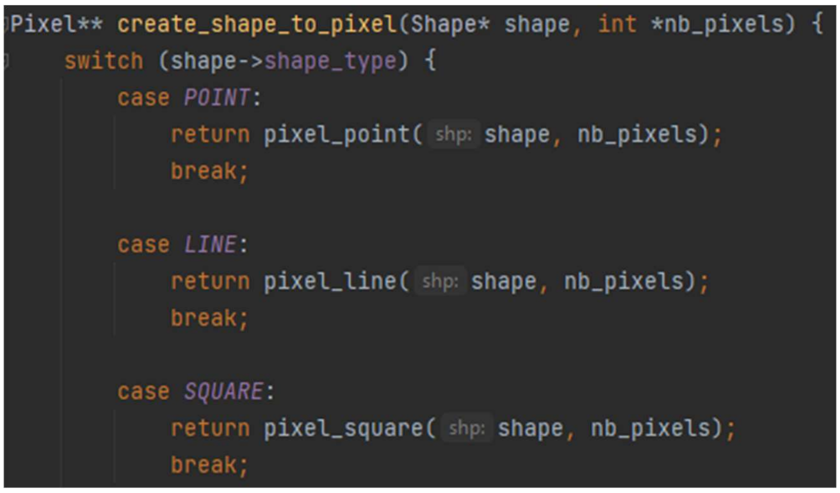
Main implemented algorithms

Interface for Help

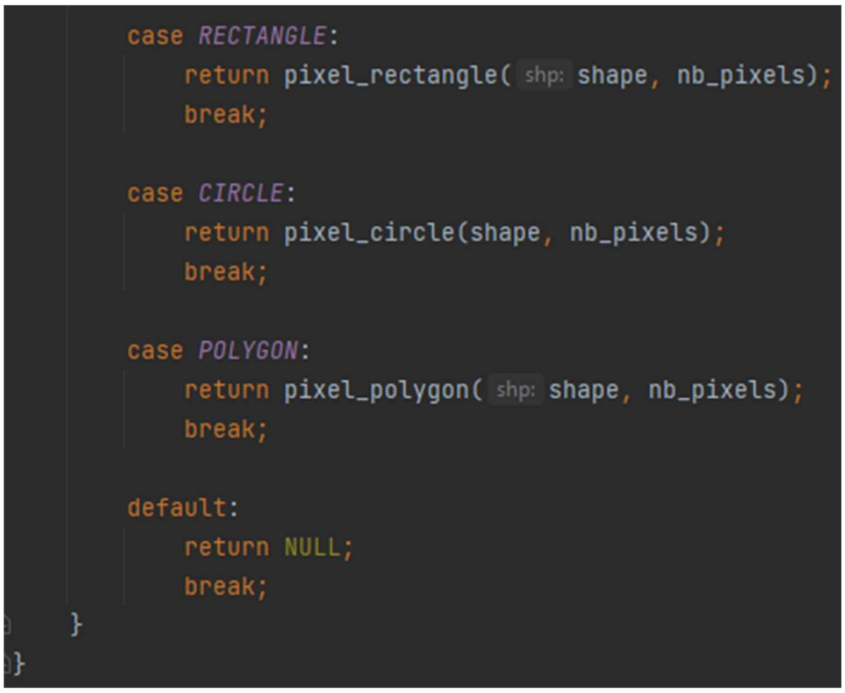


```
Please type a command line:help
-   To insert a shape please type: 'name_shape coord_x1 coord_y1 coord_x2 coord_y2 etc.' ex:
'point 3 5'
You can create a point, a line, a square, a rectangle, a circle or a polygon
-   To plot the shapes on the board please type: 'plot'
-   To display the list of shapes please type: 'list'
-   To clear the board please type: 'clear'
-   To delete a shape please type: 'delete id_of_shape' ex: 'delete 0'
-   To erase the board please type: 'erase'
-   To exit please type: 'exit'
```

Functions who transform the shape into pixel

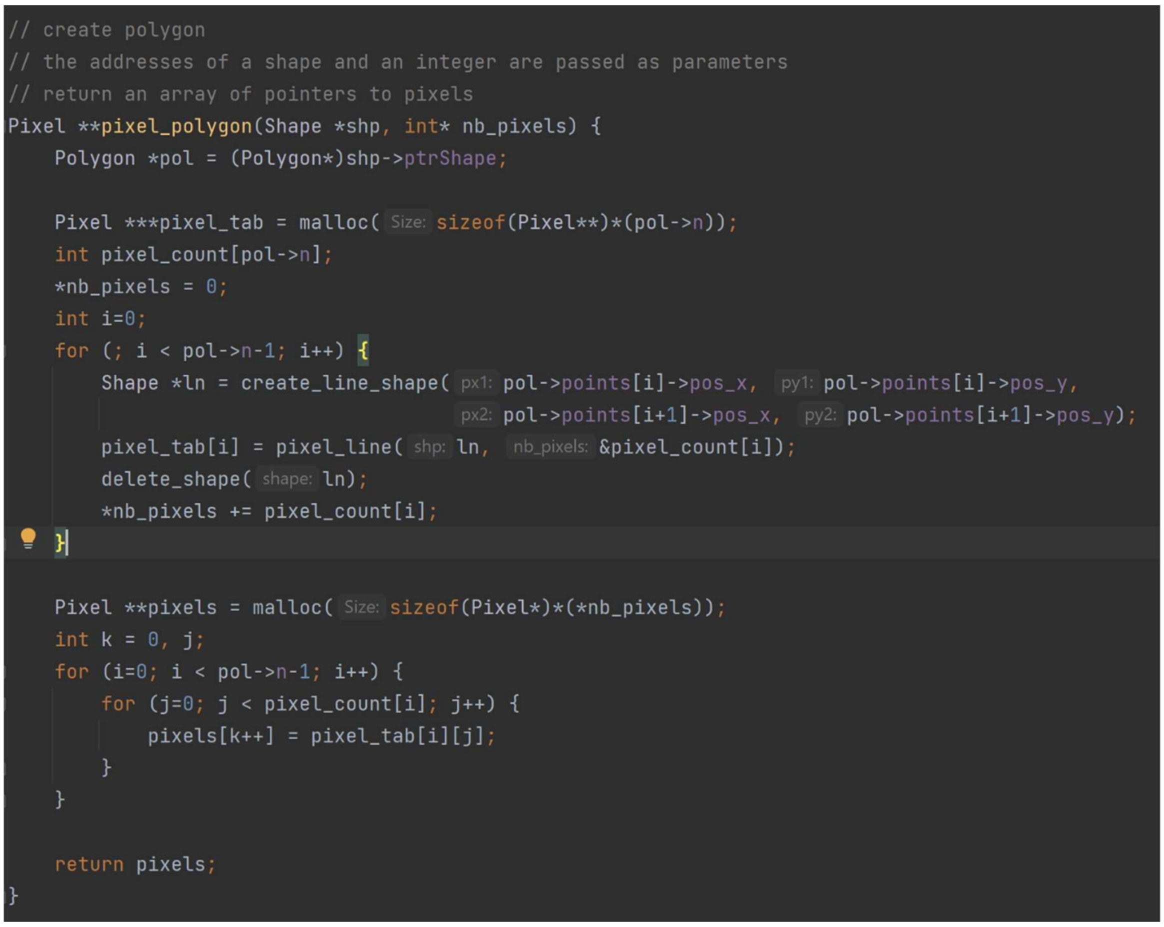


```
Pixel** create_shape_to_pixel(Shape* shape, int *nb_pixels) {
    switch (shape->shape_type) {
        case POINT:
            return pixel_point( shp: shape, nb_pixels);
            break;

        case LINE:
            return pixel_line( shp: shape, nb_pixels);
            break;

        case SQUARE:
            return pixel_square( shp: shape, nb_pixels);
            break;
```



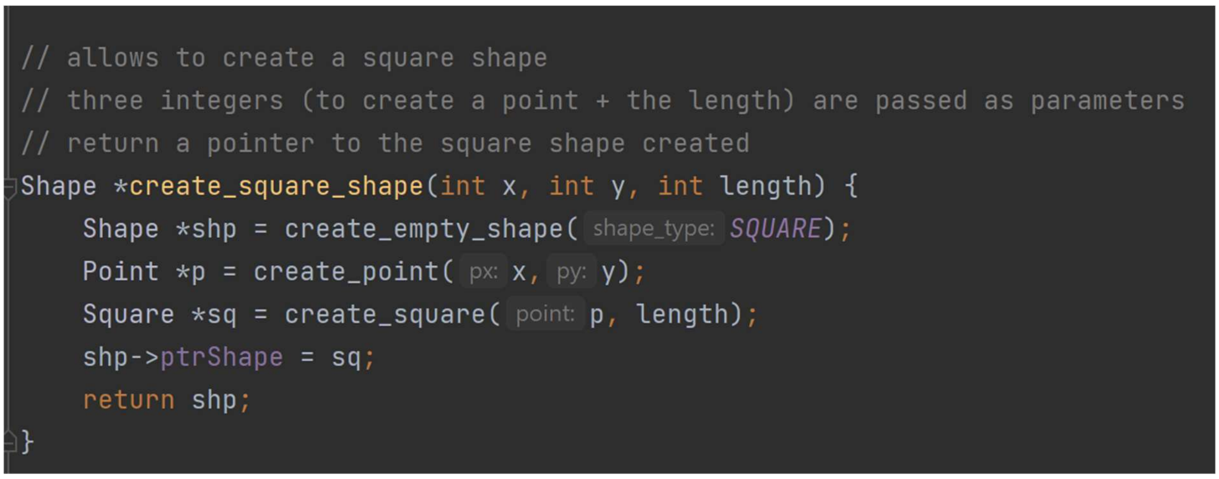
```
        case RECTANGLE:
            return pixel_rectangle( shp: shape, nb_pixels);
            break;

        case CIRCLE:
            return pixel_circle(shape, nb_pixels);
            break;

        case POLYGON:
            return pixel_polygon( shp: shape, nb_pixels);
            break;

        default:
            return NULL;
            break;
    }
}
```

Functions that create the polygon in pixel



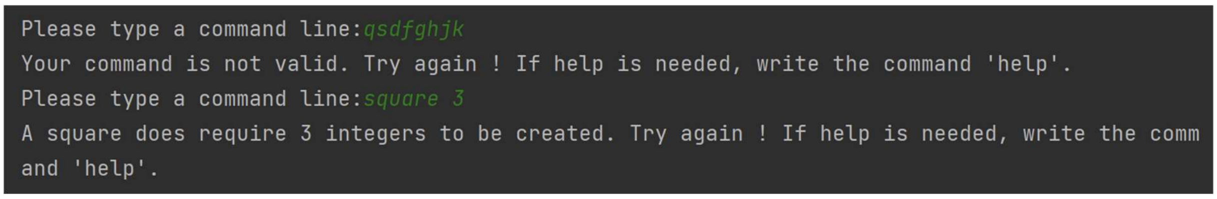
```
// create polygon
// the addresses of a shape and an integer are passed as parameters
// return an array of pointers to pixels
Pixel **pixel_polygon(Shape *shp, int* nb_pixels) {
    Polygon *pol = (Polygon*)shp->ptrShape;

    Pixel ***pixel_tab = malloc( Size: sizeof(Pixel**)*(pol->n));
    int pixel_count[pol->n];
    *nb_pixels = 0;
    int i=0;
    for (; i < pol->n-1; i++) {
        Shape *ln = create_line_shape( px1: pol->points[i]->pos_x,  py1: pol->points[i]->pos_y,
                                       px2: pol->points[i+1]->pos_x,  py2: pol->points[i+1]->pos_y);
        pixel_tab[i] = pixel_line( shp: ln,  nb_pixels: &pixel_count[i]);
        delete_shape( shape: ln);
        *nb_pixels += pixel_count[i];
    }

    Pixel **pixels = malloc( Size: sizeof(Pixel*)*(*nb_pixels));
    int k = 0, j;
    for (i=0; i < pol->n-1; i++) {
        for (j=0; j < pixel_count[i]; j++) {
            pixels[k++] = pixel_tab[i][j];
        }
    }

    return pixels;
}
```

Function create square shape

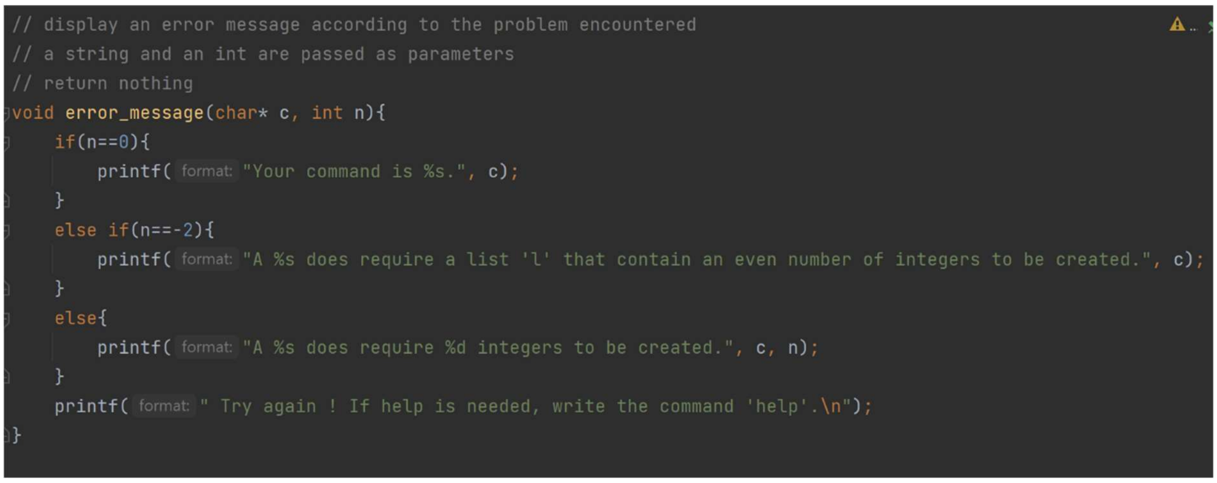


```
// allows to create a square shape
// three integers (to create a point + the length) are passed as parameters
// return a pointer to the square shape created
Shape *create_square_shape(int x, int y, int length) {
    Shape *shp = create_empty_shape( shape_type: SQUARE);
    Point *p = create_point( px: x,  py: y);
    Square *sq = create_square( point: p, length);
    shp->ptrShape = sq;
    return shp;
}
```

New function : error_message qui indique une erreur lorsque la saisie de données est fausse



```
Please type a command line:qsdfghjk
Your command is not valid. Try again ! If help is needed, write the command 'help'.
Please type a command line:square 3
A square does require 3 integers to be created. Try again ! If help is needed, write the comm
and 'help'.
```



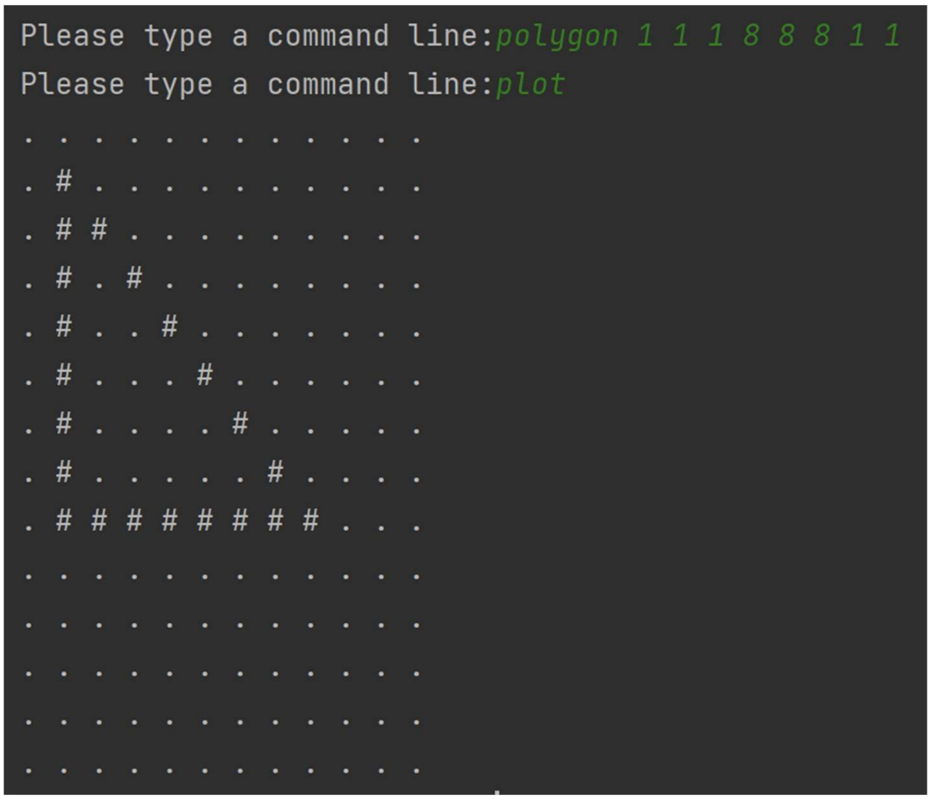
```
// display an error message according to the problem encountered
// a string and an int are passed as parameters
// return nothing
void error_message(char* c, int n){
    if(n==0){
        printf( format: "Your command is %s.", c);
    }
    else if(n==-2){
        printf( format: "A %s does require a list 'l' that contain an even number of integers to be created.", c);
    }
    else{
        printf( format: "A %s does require %d integers to be created.", c, n);
    }
    printf( format: " Try again ! If help is needed, write the command 'help'.\n");
}
```

Technical Difficulties

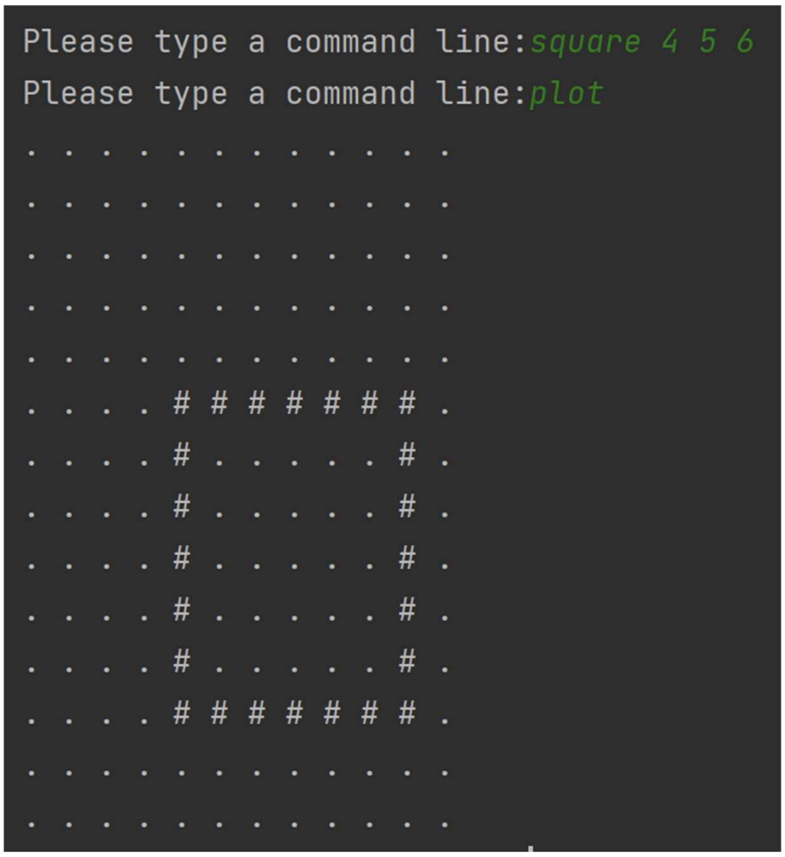
This project requires extensive use of pointers that directly access memory and data structures. Manual memory management is complex, because it is necessary to explicitly allocate the memory used by the variables.

Also, understanding and implementing Bresenham's algorithm for drawing a line is difficult, as it requires a solid understanding of math and logic. The latter requires careful thought to draw diagonal lines using decisions based on approximations.

Result presentation



```
Please type a command line:polygon 1 1 1 8 8 8 1 1
Please type a command line:plot
. . . . . . . . . . . .
. # . . . . . . . . . .
. # # . . . . . . . . .
. # . # . . . . . . . .
. # . . # . . . . . . .
. # . . . # . . . . . .
. # . . . . # . . . . .
. # . . . . . # . . . .
. # # # # # # # # . . .
. . . . . . . . . . . .
. . . . . . . . . . . .
. . . . . . . . . . . .
. . . . . . . . . . . .
. . . . . . . . . . . .
```



```
Please type a command line:square 4 5 6
Please type a command line:plot
. . . . . . . . . . . .
. . . . . . . . . . . .
. . . . . . . . . . . .
. . . . . . . . . . . .
. . . . . . . . . . . .
. . . . # # # # # # # .
. . . . # . . . . . # .
. . . . # . . . . . # .
. . . . # . . . . . # .
. . . . # . . . . . # .
. . . . # . . . . . # .
. . . . # # # # # # # .
. . . . . . . . . . . .
. . . . . . . . . . . .
```

Conclusion

The C language is powerful and flexible, its use can involve difficulties related to the manipulation of pointers and the manual management of memory. Also, the use of mathematical knowledge such as in Bresenham's algorithm to draw a line is interesting but challenging to code.