INT-1

Alexandre DIDIER
Samuel WEISTROFFER

C PROJECT

efrei
PARIS PANTHÉON-ASSAS UNIVERSITÉ

ABOUT THE PROJECT

The goal of the whole project is to code a drawing software using pixels to display shapes.

In the first part, the code was just really about defining structures and using pointers effectively in order to pass shapes in parameters.

The "user interface" in this part was the input of coordinates and the resulting output using a simple menu.

In the second part the code was harder with difficult functions using a drawing algorithm.

We both think that dividing the project into two parts, respectively the structure and the display, was very useful to get organized, so that every new function could easily be implemented.

SUMMARY

1. THE MAIN FILE

The main function of our project is concise and focused on obtaining input from the user. We have used an infinite while loop in order to stay in the program until the user chooses to quit it.

We start by defining an area and a simply linked list `l` as `NULL` that will store each shape in the order they are created.

Then we enter the infinite loop and call the functions defined in menu.c to get the input of the user.

The input obtained from the user is then passed to the compare_string function, which interprets the user's input and calls the appropriate functions to create, modify, or delete shapes within the draw_zone area.

2. Presentation of the FUNCTIONS

MANDATORY FUNCTIONS
- `Point* create_point(int px, int py)`
- `void delete_point(Point* point)`
- `void print_points(Point *p)`
- `Line* create_line(Point *p1, Point *p2)`
- `void delete_line(Line *line)`
- `void print_line(Line *line)`
- `print_line(Line *line)`
- `Square* create_square(Point *point, int length)`
- `void delete_square(Square *square)`
- `void print_square(Square *square)`
- `Rectangle* create_rectangle(Point* point, int width, int height)`
- `void delete_rectangle(Rectangle* rectangle)`
- `print_rectangle(Rectangle* rectangle)`
- `Circle* create_circle(Point *center, int radius)`
- `void delete_circle(Circle* circle)`
- `void print_circle(Circle* circle)`
- `Polygon* create_polygon(int n)`
- `void delete_polygon(Polygon* polygon)`

- void print_polygon(Polygon* polygon)
- Shape *create_empty_shape(SHAPE_TYPE shape_type)
- Shape *create_point_shape(LIST*, int x, int y)
- Shape *create_line_shape(LIST*, int x, int y, int z, int w)
- Shape *create_square_shape(LIST*, int x, int y, int z)
- Shape *create_rectangle_shape(LIST*, int x, int y, int z, int w)
- Shape *create_circle_shape(LIST*, int x, int y, int z)
- Shape *create_polygon_shape(LIST*, int n)
- void delete_shape(Shape * shape)
- void print_shape(Shape * shape)
- unsigned int get_next_id()
- Area* create_area(unsigned int width, unsigned height)
- void add_shape_to_area(Area* area, Shape* shape)
- void clear_area(Area* area)
- void erase_area(Area* area, LIST l)
- void delete_area(Area** area)
- void draw_area(Area* area)
- Pixel* create_pixel(int px, int py)
- void delete_pixel(Pixel* pixel)
- void pixel_point(Shape* shape, Pixel** pixel)
- void pixel_line(Line* line, Pixel** pixel)
- void pixel_square(Square* square, Pixel** pixel)
- void pixel_rectangle(Rectangle* rectangle, Pixel** pixel)
- void pixel_circle(Circle* circle, Pixel** pixel)
- void pixel_polygon(Polygon* polygon, Pixel** pixel)
- Pixel** create_shape_to_pixel(Shape* shape)

- `void read_input(Char* input)`
- `int compare_string(char* str, Area* draw_zone, LIST *l)`
NEW FUNCTIONS
- `NODE * create_node(Shape)`
- `bool empty_list(LIST)`
- `LIST add_tail_list(LIST , Shape)`
- `void print_list(LIST)`
- `shape* get_shape_from_node(NODE *)`
- `void delete_pixel_shape(int k, Area* area)`

Summary of functions

The functions below are explained and ordered based on the functionality of the code.

`NODE * create_node(Shape)`
`bool empty_list(LIST)`
`LIST add_tail_list(LIST , Shape)`
`void print_list(LIST)`
`shape* get_shape_from_node(NODE*)`

Those are the basic functions in order to create, modify and display a simple linked list and we add the last one to allow us to get the shape from a node.

`void delete_pixel_shape(int k, Area* area)`

This function is used to delete a shape based on its ID.

All the other functions are mandatory functions explained in the project.

3. Difficulties Encountered

We did not encounter any major difficulties in the first part of the project, on the other hand the second part posed more problems notably the use of the algorithm of drawing lines of the instruction that we did not succeed in using and that we thus modified.

The drawing and deleting of a shape based on its ID was also difficult to implement because of the problem of memory allocation. Moreover, in some scenarios, the program can crash. For example when we try to erase, then place 2 shapes, then delete the second one. It is a very specific scenario that we didn't succeed in fixing. But still the program is working fine.

Also, we did encounter several problems when trying to implement Nicolas Flasque's algorithm for the line. In fact, we decided to change and use another method than the one that was given. After a week, we got an email concerning this algorithm that actually had problems, but as our method was working, we chose to keep our algorithm. Finally, the polygon has also been a problem for the same reason (use of the lines).

For the menu, we were finding it way simpler and memory-lighter to just detect the input of the user, convert it into a string and from that with if statements take the information directly into the string.

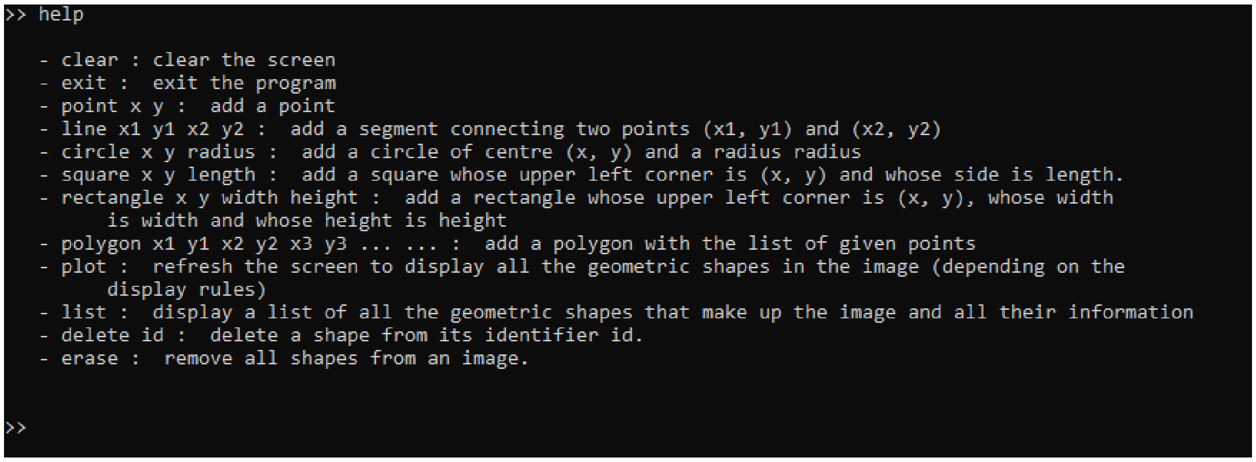
In conclusion, this second part was quite difficult because we were not used to using this kind of memory allocation in big projects.

4. Test and User INTERFACE

Here's the user interface :

```
>> _
```

You can tap help to get all the commands you can use :

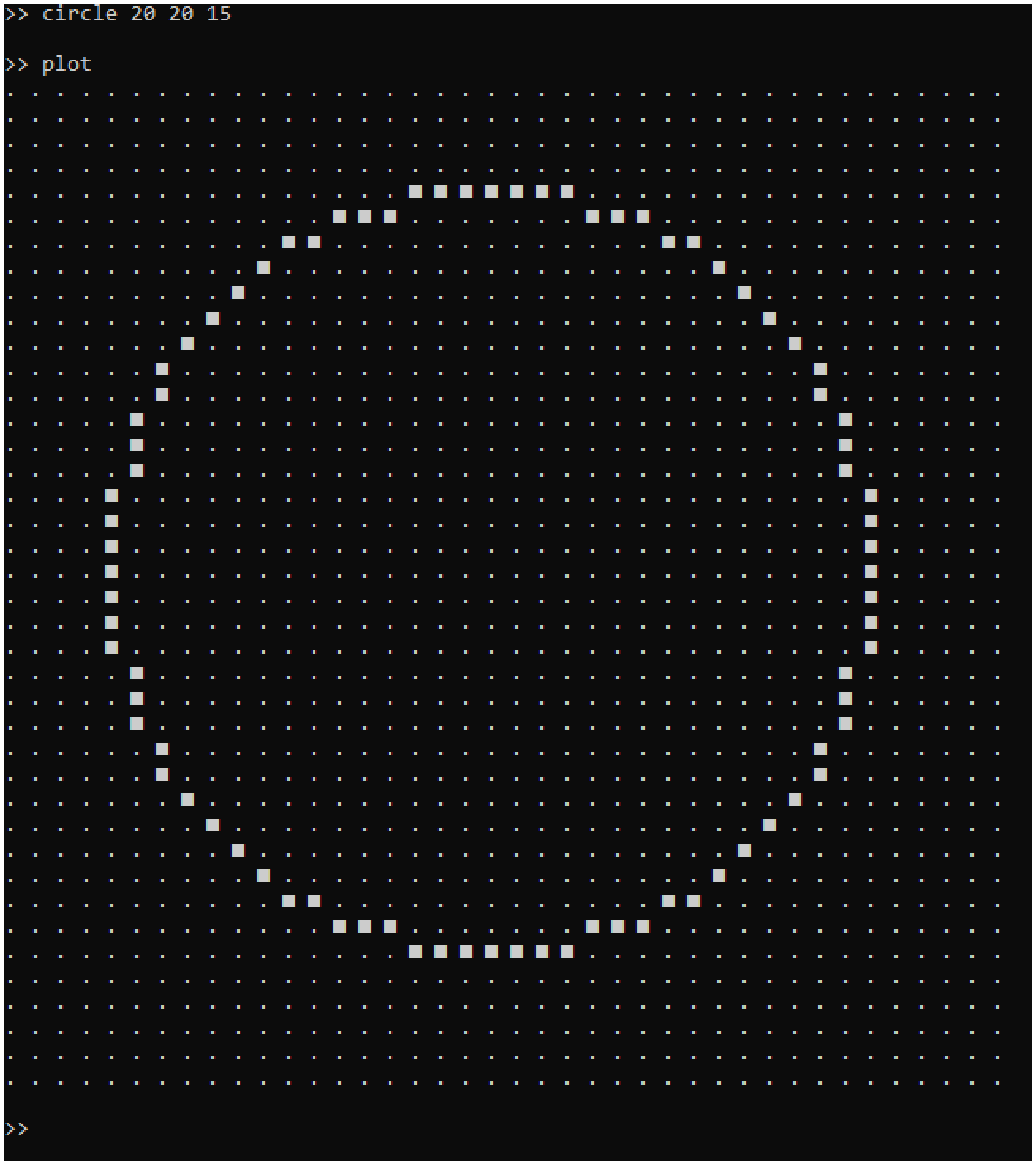


```
>> help

   - clear : clear the screen
   - exit :  exit the program
   - point x y :  add a point
   - line x1 y1 x2 y2 :  add a segment connecting two points (x1, y1) and (x2, y2)
   - circle x y radius :  add a circle of centre (x, y) and a radius radius
   - square x y length :  add a square whose upper left corner is (x, y) and whose side is length.
   - rectangle x y width height :  add a rectangle whose upper left corner is (x, y), whose width
       is width and whose height is height
   - polygon x1 y1 x2 y2 x3 y3 ... ... :  add a polygon with the list of given points
   - plot :  refresh the screen to display all the geometric shapes in the image (depending on the
       display rules)
   - list :  display a list of all the geometric shapes that make up the image and all their information
   - delete id :  delete a shape from its identifier id.
   - erase :  remove all shapes from an image.


>>
```

Let's run a quick test and try to draw a smiley : first, let's draw a circle :

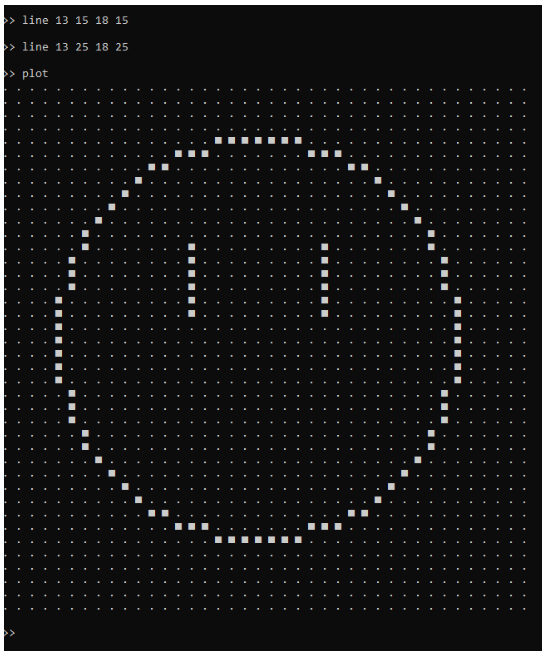


```
>> circle 20 20 15

>> plot
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . ■ ■ ■ ■ ■ ■ ■ . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . ■ ■ ■ . . . . . . . ■ ■ ■ . . . . . . . . . . . . . .
. . . . . . . . . . . ■ ■ . . . . . . . . . . . . . ■ ■ . . . . . . . . . . . .
. . . . . . . . . . ■ . . . . . . . . . . . . . . . . . ■ . . . . . . . . . . .
. . . . . . . . . ■ . . . . . . . . . . . . . . . . . . . ■ . . . . . . . . . .
. . . . . . . . ■ . . . . . . . . . . . . . . . . . . . . . ■ . . . . . . . . .
. . . . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . . . .
. . . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . . .
. . . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . . .
. . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . .
. . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . .
. . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . .
. . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . .
. . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . .
. . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . .
. . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . .
. . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . .
. . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . .
. . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . .
. . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . .
. . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . .
. . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . .
. . . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . . .
. . . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . . .
. . . . . . . ■ . . . . . . . . . . . . . . . . . . . . . . . ■ . . . . . . . .
. . . . . . . . ■ . . . . . . . . . . . . . . . . . . . . . ■ . . . . . . . . .
. . . . . . . . . ■ . . . . . . . . . . . . . . . . . . . ■ . . . . . . . . . .
. . . . . . . . . . ■ . . . . . . . . . . . . . . . . . ■ . . . . . . . . . . .
. . . . . . . . . . . ■ ■ . . . . . . . . . . . . . ■ ■ . . . . . . . . . . . .
. . . . . . . . . . . . . ■ ■ ■ . . . . . . . ■ ■ ■ . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . ■ ■ ■ ■ ■ ■ ■ . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

>>
```

Then add the eyes :

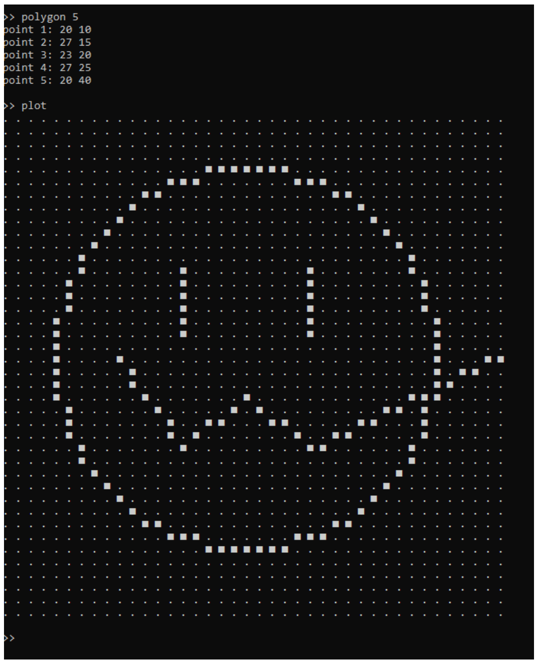


```
>> line 13 15 18 15

>> line 13 25 18 25

>> plot
```

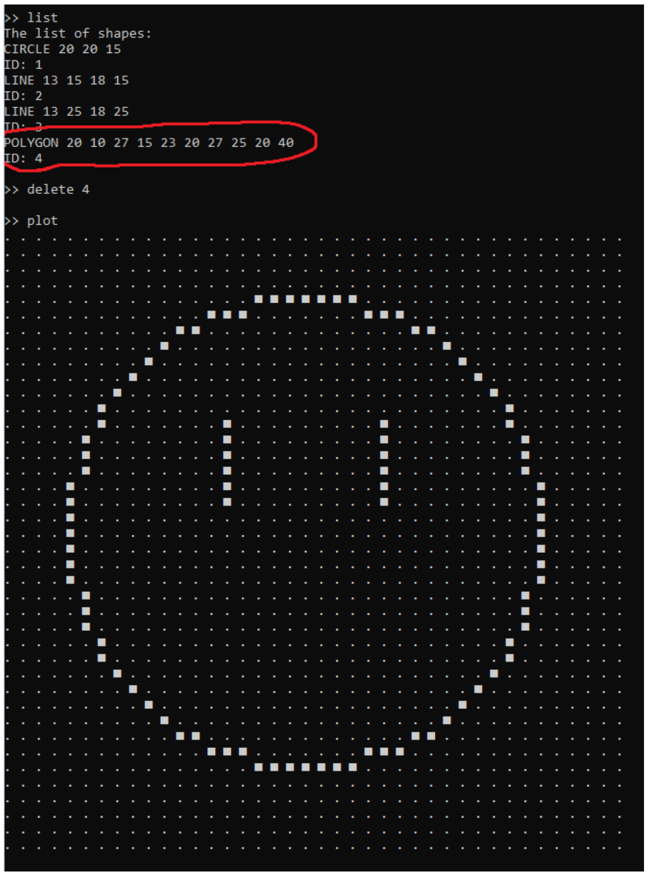
Then add the mouth :



```
>> polygon 5
point 1: 20 10
point 2: 27 15
point 3: 23 20
point 4: 27 25
point 5: 20 40

>> plot
```

Oh no ! We made a mistake. Let's fix it !

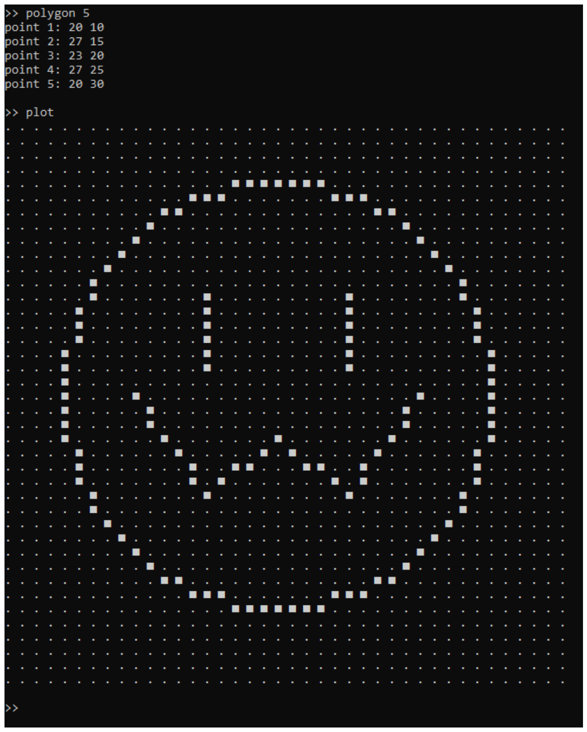


```
>> list
The list of shapes:
CIRCLE 20 20 15
ID: 1
LINE 13 15 18 15
ID: 2
LINE 13 25 18 25
ID: 3
POLYGON 20 10 27 15 23 20 27 25 20 40
ID: 4

>> delete 4

>> plot
```

Let's draw the mouth again :

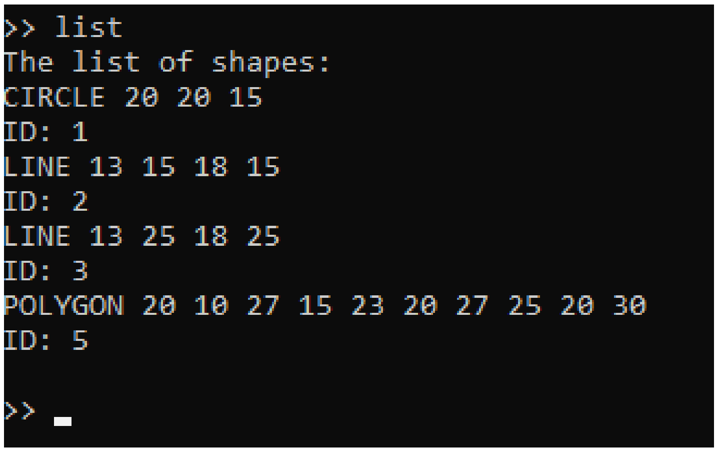


```
>> polygon 5
point 1: 20 10
point 2: 27 15
point 3: 23 20
point 4: 27 25
point 5: 20 30

>> plot
```

```
>>
```

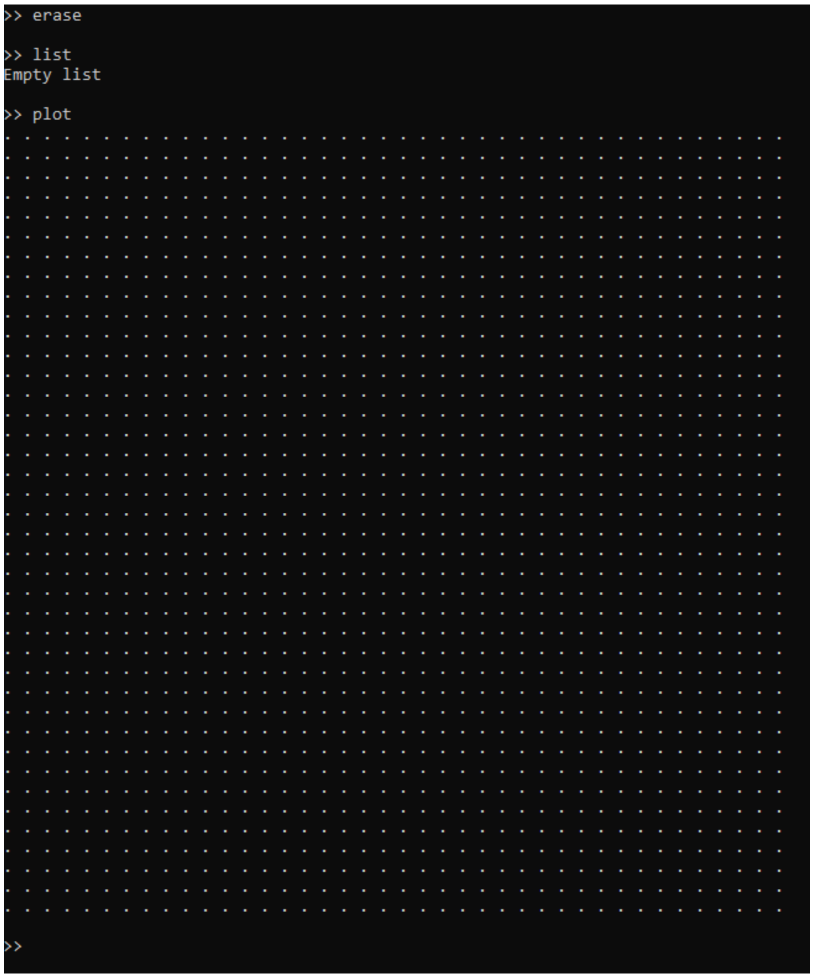
Now let's check all the shapes :



```
>> list
The list of shapes:
CIRCLE 20 20 15
ID: 1
LINE 13 15 18 15
ID: 2
LINE 13 25 18 25
ID: 3
POLYGON 20 10 27 15 23 20 27 25 20 30
ID: 5

>> 
```

We can erase the canva and start a new drawing if we want to ! :



```
>> erase

>> list
Empty list

>> plot
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

>>
```

5. CONCLUSION

This first C project was not very simple but very enriching. We learned a lot about the use of structures, pointers and dynamic arrays in particular.

We were also able to improve our way of working in groups with github in particular which simplified things compared to the first project where we used discord.

We found that C was a very useful language for this kind of project with its system of pointers, structure and dynamic array in particular and we hope to deepen the use of this language

Thank you !