eFrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Vector Text-based Editor

Oscar Masdupuy – Group 14

2022-2023

L1 – INT1 – Halim Djerroud, Fabien Calcado, Asma Gabis

Objectives of the project

In this project, we want to develop a vector drawing application in text mode. The application to be realized must at least allow a user to create a simple vector image using a set of user commands and to display this image in textual mode. The application must therefore be able to manage at least one window and one layer containing all the geometric shapes created by the user. To do this, we need to look at three main aspects:

• The memory representation of all the information of our application. The way to save the information of the geometrical shapes allowing to display them.
• A command line interface (As a menu) allowing the user to perform the actions permitted by the application.

• The display on the screen in text mode of the image created by applying the drawing algorithms described in the corresponding sections. From this minimal version, you will be asked to implement various improvements such as:

• The creation of more complex shapes: Bézier curves; 8

• Code optimization for adding, deleting, or moving a shape in the hierarchy (use of linked lists).

• A system of multiple layers (nested or not).

• A multiple window system; You will of course be able to implement other improvements such as the ability to: apply transformations on the shapes of the image (rotation, translation, etc.), zoom in/out, move a shape from one layer to another, move layers in the image, save/load an image in a text file, offer commands to undo or redo an old command (Undo / Redo).

Functions realized in the project :

```
Point *create_point(int px, int py);

void delete_point(Point * point);

void print_point(Point * p);
```

```
Line *create_line(int x1, int y1, int x2, int y2);
void delete_line(Line * line);
void 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);
void print_rectangle(Rectangle * rectangle);
```

```c
Circle *create_circle(Point * center, int radius);
void delete_circle(Circle * circle);
void print_circle(Circle * circle);
```

```c
Polygon *create_polygon(int n);
void delete_polygon(Polygon * polygon);
void print_polygon(Polygon * polygon);
```

```c
Shape *create_empty_shape(SHAPE_TYPE shape_type);
Shape *create_point_shape(int px, int py);
Shape *create_line_shape(int px1, int py1, int px2, int py2);
Shape *create_square_shape(int px, int py, int length);
Shape *create_rectangle_shape(int px, int py, int width, int height);
Shape *create_circle_shape(int px, int py, int radius);
Shape *create_polygon_shape(int n);
void delete_shape(Shape * shape);
void print_shape(Shape * shape);
```

```c
Pixel* create_pixel(int px, int py);
void delete_pixel(Pixel* pixel);
int min(int v1,int v2);
int max(int v1, int v2);
void pixel_point(Point* point, Pixel*** pixel_tab, int* nb_pixels);
void pixel_line(Line* line, Pixel*** pixel, int* nb_pixels);
void pixel_square(Square* square, Pixel*** pixel_tab, int* nb_pixels);
Pixel** create_shape_to_pixel(Shape * shape, int* nb_pixels);
void delete_pixel_shape(Pixel** pixel, int nb_pixels);
void pixel_rectangle(Rectangle* rectangle, Pixel*** pixel_tab, int* nb_pixels);
void pixel_polygon(Polygon* polygon, Pixel*** pixel_tab, int* nb_pixels);
```

```c
Command* create_command();
void add_str_param(Command* cmd, char* p);
void add_int_param(Command* cmd, int p);
void free_cmd(Command* cmd);
int read_exec_command(Command* cmd);
void read_from_stdin(Command* cmd);
```

```c
Area* create_area(unsigned int width, unsigned int height);
void add_shape_to_area(Area* area, Shape* shape);
void remove_shape_from_area(Area* area, int ID);
void clear_area(Area* area);
void erase_area(Area* area);
void delete_area(Area* area);
int in_list(Pixel** p1, int nb_pixels, int x, int y);
void draw_area(Area* area);
void display_shapes(Area* area);
void print_area(Area* area);
```

Technical uses of functions in the project :

This main program implements a shape management system that allows users to perform various actions on different geometric shapes. Here is a description of the implemented algorithms:

Initialization:

The necessary header files and libraries are included.
Variables are declared, including direction and action to store user input, and selected to control the program loop.
An area pointer a1 is created using the create_area function, with initial dimensions of 10 rows and 30 columns.

Main Menu Loop:

The program enters a while loop that continues if selected is true.
The user is presented with a menu of actions and prompted to select an action by entering a character representing their choice (direction).
If an invalid action is entered, the user is prompted again until a valid action is selected.

Action Selection:

If the user selects 'A' or 'a' (Add a shape), another menu is presented with different shape options, and the user is prompted to select an action (action).
If an invalid action is entered, the user is prompted again until a valid action is selected.

Shape Addition:

Depending on the selected shape option, the user is prompted to enter the required parameters for that shape.
The corresponding shape object is created using the appropriate create_*_shape function.
The created shape is added to the area using the add_shape_to_area function.
The area is drawn and printed using draw_area, print_area, and print_shape functions.

Shape Display:

If the user selects 'B' or 'b' (Display the list of shapes), the display_shapes function is called, which prints the list of shapes in the area.
Shape Deletion:
If the user selects 'C' or 'c' (Delete a shape), the user is prompted to enter the ID of the shape they want to delete.
The remove_shape_from_area function is called with the provided ID to remove the shape from the area.

Area Drawing:

If the user selects 'D' or 'd' (Drawing the shapes), the draw_area function is called to draw the shapes in the area.

Area Printing:

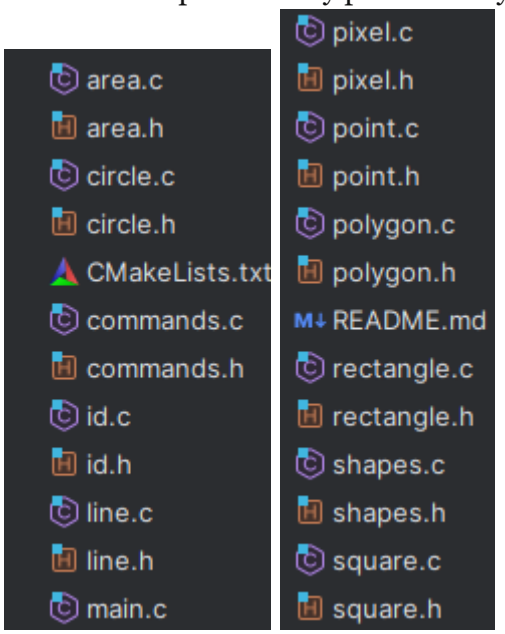
If the user selects 'E' or 'e' (Help), the print_area function is called to print the current state of the area.

Exit:

If the user selects 'F' or 'f' (Exit), the selected variable is set to false, and the program loop is terminated.

The program continues to loop until the user selects the 'F' or 'f' option, allowing them to perform multiple actions on shapes in the area.

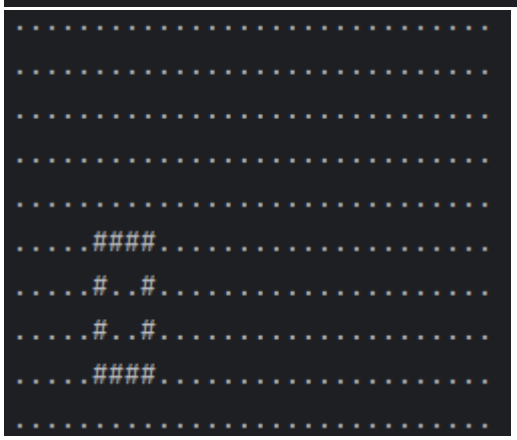
I choose to separate every part in many new files to have a very ventilate code.



Results presentation :

```
Please select an action:
   A- Add a shape:
   B- Display the list of shapes:
   C - Delete a shape:
   D- Drawing the shapes:
   E- Help:
   F- Exit:
----------Your action----------
```

```
Please select an action:
   1- Add a point
   2- Add a line
   3- Add circle
   4- Add a square
   5- Add a rectangle
   6- Add a polygon
   7- Return to the previous menu
----------Your action----------
4
Enter the coordinates of the left corner of your square and the size : x1 y1 size
5 5 3
```



You can reproduce the test by downloading the .zip here and try the program in this order :

Step 1 : Run the main
Step 2 : “A”
Step 3 : “4”
Step 4: “5 5 3”

Conclusion :

Throughout the course of this project, I embarked on a personal journey that encompassed not only technical growth but also valuable insights into work organization, and time management. Here are some personal reflections on the difficulties faced and the significant lessons learned:

Technical Learning:

Dynamic Memory Allocation: One of the challenging aspects was managing dynamic memory allocation using functions like malloc() and free().
Pointer Manipulation: Working extensively with pointers required a deeper understanding of memory addresses, dereferencing, and pointer arithmetic. Overcoming initial difficulties in correctly manipulating pointers significantly improved my understanding of this powerful feature in C.

Work Organization :

Being alone for this whole project were a difficult task, I needed to do the time work twice to complete the project. It was challenging for me to learn by myself a lot of new possibilities in coding without the possibility to ask someone to help me doing a task.

Problem Solving and Debugging:

Debugging Skills: I encountered various bugs and logical errors throughout the project. This provided ample opportunities to develop and refine my debugging skills, including the effective use of debugging tools, systematic problem-solving approaches, and thorough testing practices. Over time, I became more proficient in identifying and resolving issues efficiently.

In conclusion, this project offered a rich learning experience that extended beyond technical skills.. These lessons have equipped me with valuable insights and skills that extend far beyond the project itself, preparing me for future endeavors in software development.