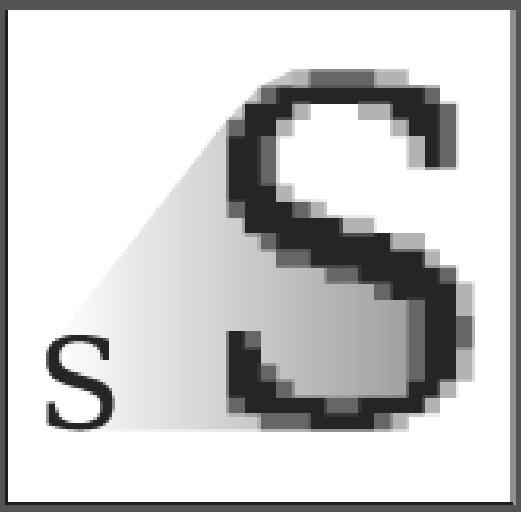
Peinture.com

EFREI Paris - TI202 - L1 S2
Projet en C

Projet réalisé par Idir NAIT MEDDOUR et Fares DARGHOUTH



Créateur d'images vectorielles

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Groupe A
Promo 2027

Introduction

Au cours de ce deuxième semestre, nous avons eu l'occasion de découvrir un nouveau langage informatique, celui qui a d'ailleurs servi de base pour créer Python, le langage C.

Dans le cadre de notre projet en binôme, nous avons été chargés de concevoir et développer un logiciel de dessin vectoriel dans ce langage de programmation.

Au cours de ce rapport, nous décrirons les différentes étapes de développement, nous présenterons également les principales fonctionnalités du logiciel, telles que la création, l'affichage ou encore la suppression des objets vectoriels.

Nous aborderons également les choix techniques que nous avons faits pour la conception du logiciel, tels que l'utilisation de structures de données appropriées pour stocker les informations des dessins, la manipulation des coordonnées mathématiques pour les affichages et les calculs, et l'optimisation des performances.

Enfin, nous discuterons des résultats obtenus, des difficultés rencontrées et des perspectives d'amélioration pour le logiciel. Nous sommes convaincus que ce projet nous a permis de développer nos compétences en programmation, notamment dans un nouveau langage de programmation que nous venons de maîtriser, et en gestion de projet.

Sommaire

1. Présentation fonctionnelle

Fonctionnalités du logiciel :

- Choix de créer un nouveau dessin, d'afficher une description, ou de quitter le programme
- Choix de la taille du dessin
- Commande "help" permettant d'afficher les commandes disponibles, qui sont évidemment toutes sécurisées

- Commandes permettant de créer un objet, soit un point, une ligne, un cercle, un carré, un rectangle, ou encore un polygone :

```
      - point x y : ajoute un point
      - line x1 y1 x2 y2 : ajoute un segment reliant deux points (x1, y1) et (x2, y2)
      - circle x y radius : ajoute un cercle de centre (x, y) et de rayon radius
      - square x y length : ajoute un carre dont le coin superieur gauche est (x, y) et de cote length.
      - rectangle x y width height : ajoute un rectangle dont le coin superieur gauche est (x, y), de largeur width et
de longueur height
      - polygon x1 y1 x2 y2 x3 y3 ... ... : ajoute un polygone avec la liste des points donnes
```

- Commandes permettant la gestion générale du logiciel après la création des objets du dessins (afficher le dessin, afficher la liste des formes présentes sur le dessin, les supprimer à l'aide de leur identifiant, effacer l'écran, supprimer toutes les formes ou encore quitter le programme) :

```
- plot : affiche le dessin
- list : liste les formes du dessin
- delete id : supprime une forme d'identifiant (id) si elle existe
- clear : efface l'ecran
- erase : supprime toutes les formes du dessin

- help : affiche la liste des commandes disponibles

- quit : quitte le programme
```

2. Présentation technique

Cette deuxième partie rapport va brièvement expliquer l'architecture de notre programme Peinture.com

Le programme utilise une méthode de gestion d'état pour déterminer quel menu afficher à l'écran.
Par exemple, nous avons l'état MAIN_MENU, CHOOSE_SIZE, ON_CANVAS, etc.

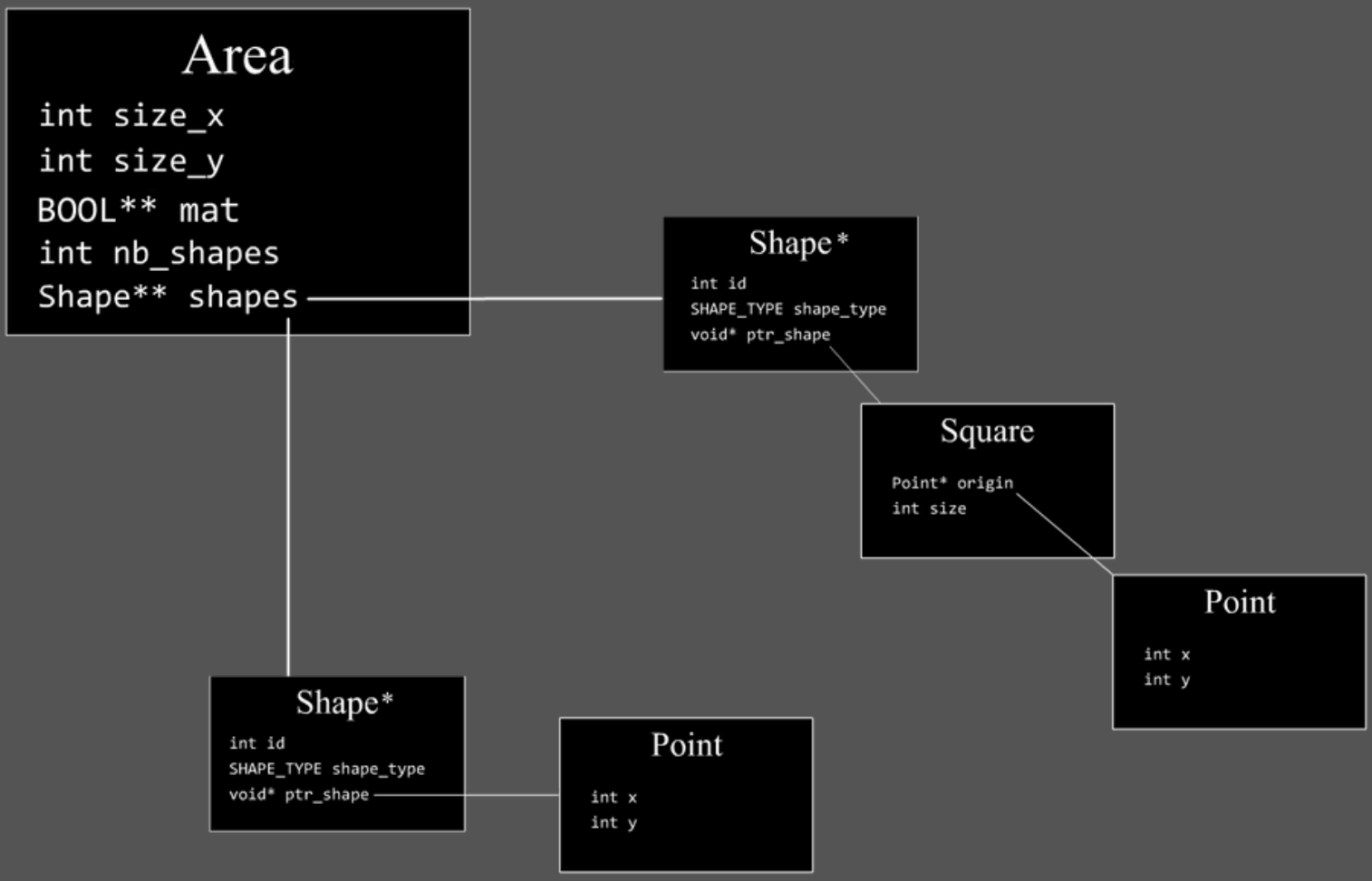
```
typedef enum { MAIN_MENU, CHOOSE_SIZE, ON_CANVAS, ADD_SHAPE } STATE;
```

Cette méthode a l'avantage d'éviter du code trop imbriqué, c'est-à-dire, que presque toutes les fonctionnalités du programme se retrouvent peu réparties, et permet de faire gérer les différents états du programmes de manière indépendante (puisque chaque état est géré par ses propres fonctions), et l'inconvénient de forcer l'usage de quelques variables globales (une pratique déconseillée), en l'occurrence l'état actuel du programme.
Les états sont tous regroupés dans un enum.

Concernant la structure des struct, les éléments les plus simples sont les formes : Point, Line, Square, Rectangle, Circle et Polygon.
Ces formes sont contenues dans des Shape qui contiennent également un identifiant unique. Pour l'instant, ces Shape sont contenues dans un struct nommé Area, qui représente le projet tout entier.

2. Présentation technique

Schéma explicatif du squelette de notre programme.



Ici, le dessin possède 2 Shape, une qui contient un Point, l'autre qui contient un Square.

2. Présentation technique

La gestion des commandes se fait grâce à la structure de données Command :

```
typedef struct {
    COMMAND_TYPE type;
    int int_params[10];
    int int_size;

    char* str_params[10];
    int str_size;
} Command;
```

Ici, str_params et str_size ne sont pas utilisés.

La fonction qui lit les commandes est celle-ci :

```
void read_from_stdin(Command* cmd);
```

Elle sera utilisée pour modifier l'objet Command, qui va ensuite être utilisé par la fonction suivante :

```
int exec_command(Command* cmd);
```

Elle utilise la fonction *add_shape_to_area* qui va ensuite appeler les fonctions suivantes, en fonction de la commande saisie par l'utilisateur :

```
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(const int lst[], int n);
void delete_shape(Shape * shape);
void print_shape(Shape * shape);
```

3. Présentation des résultats

Lorsqu'on lance le programme, il est affiché cela :

```
##################################
##                              ##
##          PEINTURE.COM        ##
##                              ##
##                              ##
##                              ##
##################################
Bonjour ! Que souhaiteriez-vous faire ?
        1. Nouveau dessin

        2. A propos

        99. Quitter
```

Il faut ensuite que l'on choisisse une taille :

```
Choisissez la taille de votre dessin (entre 10,10 et 50,50) :10,10
```

Voici l'affichage après avoir créé certaines formes, et les avoir affichés

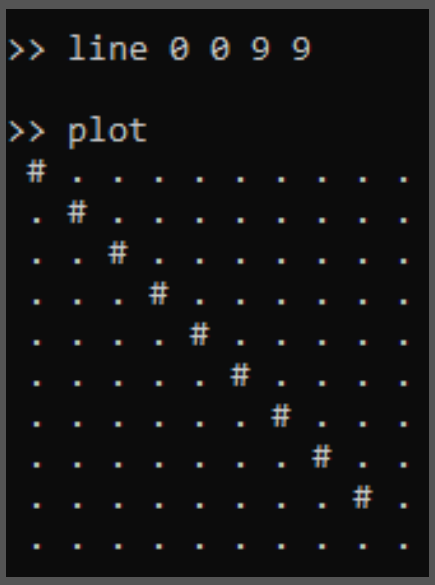
```
>> point 4 6

>> plot
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . # . .
. . . . . .
. . . . . .
. . . . . .
```

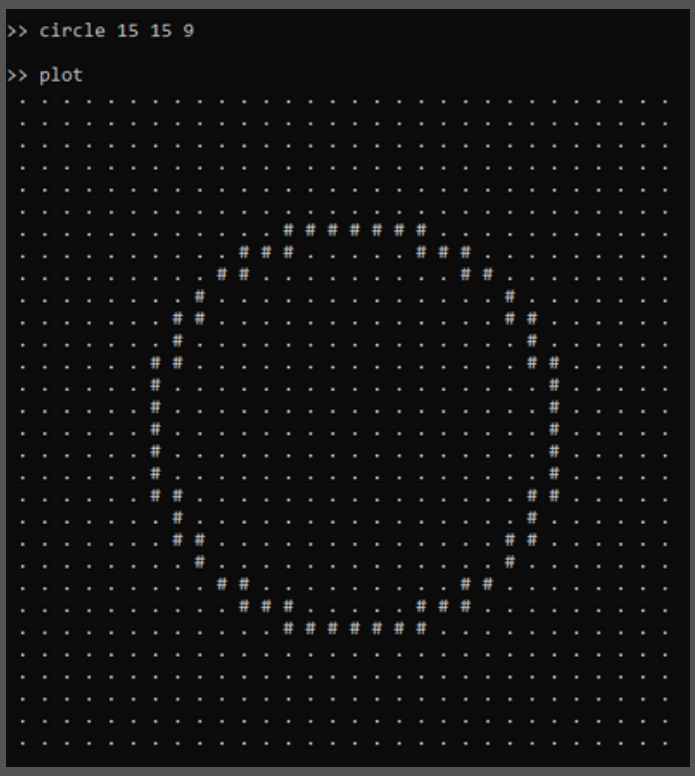
```
>> square 1 1 7

>> plot
. . . . . .
. # # # # # # # . .
. # . . . . . # . .
. # . . . . . # . .
. # . . . . . # . .
. # . . . . . # . .
. # . . . . . # . .
. # # # # # # # . .
. . . . . .
. . . . . .
```

```
>> rectangle 3 4 5 2

>> plot
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . # # # # # . .
. . . # # # # # . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
```

```
>> polygon 1 1 0 9 9 9 1 1

>> plot
. . . . . .
. # . . . . . .
. . # . . . . . .
. . . # . . . . . .
. . . . # . . . . .
. . . . . # . . . .
# . . . . . # . . .
# . . . . . # . . .
# . . . . . # . .
# . . . . . .
```

3. Présentation des résultats





4. Conclusion

En conclusion, notre projet de création d'un logiciel de dessin vectoriel en langage C a été une expérience enrichissante. Nous avons ainsi appris à mieux utiliser ce langage de programmation, en utilisant notamment les structures, les fonctions (nous avons d'ailleurs appris à utiliser le principe de généricité avec void*), les switch, etc. Nous avons réussi à développer un logiciel de dessin vectoriel fonctionnel sur console en C, en utilisant des commandes saisies par l'utilisateur ce qui facilite grandement son utilisation.

Nous avons également appris à collaborer en équipe, en utilisant de nombreux moyens de partage de fichier et de contrôle de fonction, tel que Github ou encore Discord pour communiquer, ou même la fonctionnalité *Code With Me* proposé par JetBrains sur CLion.

Malgré les défis rencontrés (comme les nombreuses erreurs de segmentation), nous sommes fiers des résultats obtenus. Ce projet nous a permis de renforcer nos compétences en programmation et en collaboration. Nous sommes reconnaissants de cette opportunité et restons ouverts aux commentaires pour améliorer davantage notre logiciel.