ETCH A SKETCH PROGRAM REPORT

Thu Hang Nguyen
BACATA Level 4
S5080488

SUMMARY

Etch a Sketch program is a graphical programming project produced in C language, utilizing SDL libraries to emulate the Etch a Sketch toy by imitating its drawing operations. In the real-life Etch a Sketch, we switch the buttons to move up and down, right and left. However, in our program, this is done with keyboard (A and D to move vertically, left and right arrows to move horizontally). Moreover, the program adds new features to the game, for example, adjusting line color, line-width, drawing speed and choosing background. It also allows user to save and load files for further editing.

The program has simple, straightforward and easy-to-use user interface with a tool bar, a column of trays, and a main drawing board. However, user must use terminal to run the program and type input such as file name. Saving operation can only store files in the same directory as the executable program.

The program has only one module (one source file) and extensively implements procedural, structural programming paradigms, and iterations.

BACKGROUND

1. Goal of the program

The main goal of the Etch a Sketch program is to allow keyboard user interface to draw lines on the screen, with some additional features (adjusting line color, line width, speed, changing background, saving and loading files). The program should also allow user to start from scratch, or refresh to delete all the line drawings (shaking the toy in real life). User must use terminal to run the program and type input data.

2. Idea

The idea is to utilize SDL libraries to produce graphical context (a window that can be quit), with clickable buttons and functional A and D, left arrow and right arrow key pads, which need a *main event loop*. The source code should handle three layers of graphic separately:

- The user interface (which will be stored in variable *texture*).

- The main drawing board (variable *boardtexture*, background will be loaded on *boardsurface*).
- The line drawings (drawn and loaded on *pixelsurface*, which will be created into *pixeltexture*).

Furthermore, the code should make easy for editing and changing variables and parameters with *struct*. It should apply procedures, structures and recursions, as well as using meaningful variable names to keep the code well-arranged and readable.

Beside the idea for overall algorithm flow, the design idea for user interface must achieve the “straight-forward, easy to use” goal, and it must have good visual impact. Therefore, I quickly sketched a rough design of user interface, created it in Adobe Illustrator, and handle the bars and buttons separately for the realization stage.

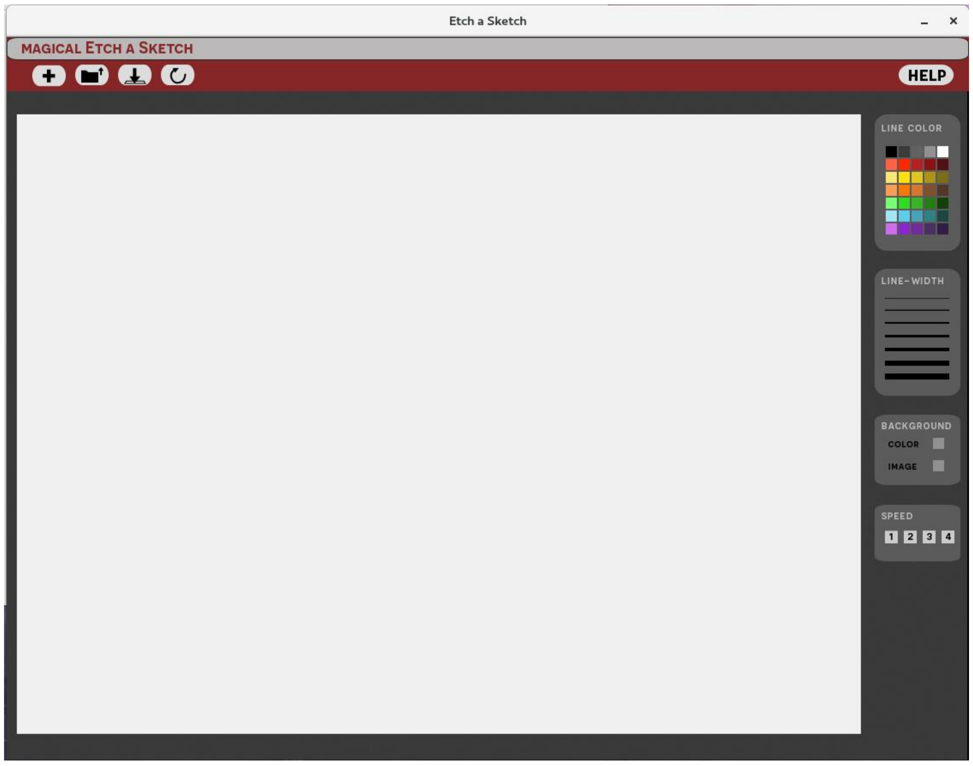
The programming process must be kept track carefully using prototype model (design, implement, then debug). It is a good practice to test the code for individual functionality of the program before actually using it, to avoid affecting the source code, and avoid confusion.

APPLICATION DESIGN

1. Rough design

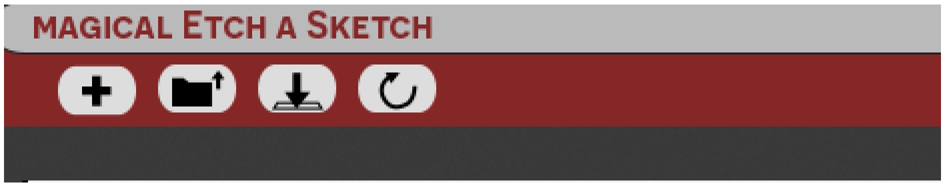
In this part I would describe the design process for the Etch a Sketch program in details.

A. User Interface



A navigatable user interface created in Adobe Illustrator with the main drawing board, a tool bar (from left to right: New, Open, Save, Refresh buttons), and the tray column on the left.

The tool bar handles board (New and Refresh buttons), and the saving and loading features (Open and Save buttons).

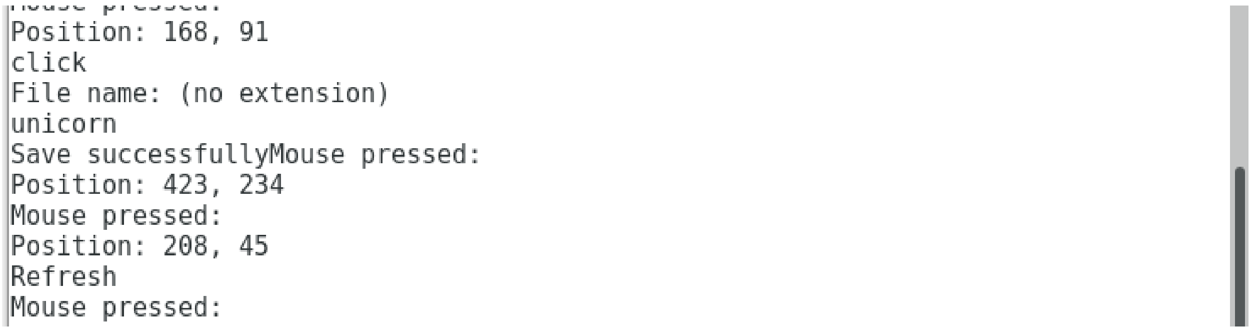


The trays manipulate drawing process that allow user to pick color, choose line-width, change background and adjust speed.

B. User Input (keyboard and mouse)

User input is handled in the main event loop (the while loop) with switch statement: SDL_KEYDOWN for keyboard input interface, and SDL_MOUSEBUTTONDOWN for mouse input.

User can choose their own file name for saving, however, this must be done in the terminal since the program does not allow to put text-based data into the graphical context (SDL libraries do provide functions to achieve this, however, it is beyond my coding ability and time permitted). The file name asked for saving data must not have extension, so that in the source code, I could concatenate the extensions (bmp, png, esc) separately.



```
Position: 168, 91
click
File name: (no extension)
unicorn
Save successfullyMouse pressed:
Position: 423, 234
Mouse pressed:
Position: 208, 45
Refresh
Mouse pressed:
```

C. User Output (screen display and storing files)

The input (keyboard and mouse) should produce data onto the screen.

For drawing operation, every algorithm must only access *pixeltexture,* which is transparent. I can use SDL_FillRect to draw the points on a surface in different positions, sizes and colors. Keyboard input would only change the data stored positions, while mouse input would change the rest. For Speed setting, I use SDL_Delay to delay the event loop, thus making the drawing process onto screen appear slower. However, in realization stage SDL_Delay proved to have its downsize and this was addressed accordingly.

For saving *esc files (configuration file), I used file access to gain access to binary files.

2. Technique

The overall programming technique for this Etch a Sketch program is mainly procedural (void functions that return none or multiple values), which makes the code easier to navigate. Procedural programming proves to be very effective, as I can call a repeated algorithm multiple times, for example, the algorithm for loading image with SDL image library, which may turn into unnecessary large blocks of code with multiple textures and surfaces, or the algorithm for picking colours.

Sparsely I apply functional programming, only for returning a value of 1 or 0 if the command fails and return NULL pointer.

To “Open” or “Save” the *esc (binary) file, I must pass multiple values by reference to access all the settings in one function.

I also use iterations, such as while loop for event loop, and for loop for repeating patterns such as the color swatches, line buttons and speed buttons, instead of writing the codes for them many times.

3. Macro design (algorithm flow)

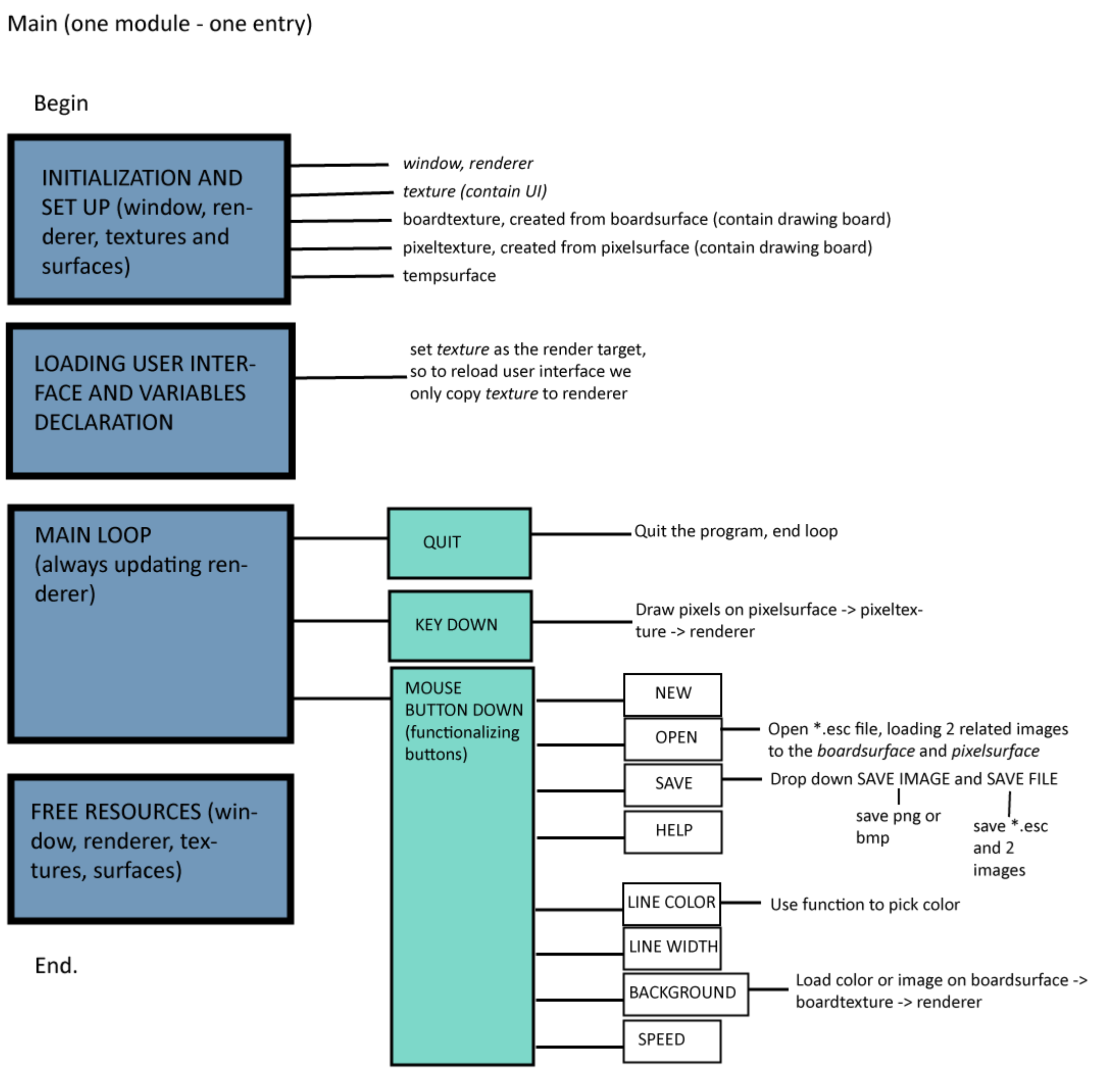
The Etch a Sketch program is not a large project, so the macro design for it is very simple. The source code only contains one C file, with one entry (main function).

Overall, the project contains one source code file, a Makefile to compile the source code with Clang by typing *make all* in the terminal, an image folder that contains all image files to be loaded into the graphical context, a font for SDL to load text-based data, and an executable file called EtchaSketch (after compilation).

The program means to run on Linux operation system by the terminal. For successful compilation, SDL2 libraries (including SDL.h, SDL_image.h and SDL_ttf.h) must be installed beforehand in the computer.

It is supposed to run without any failures in the NCCA First Year lab of Bournemouth University.

4. Micro design (modules)


Main (one module - one entry)
Begin
INITIALIZATION AND SET UP (window, renderer, textures and surfaces)
window, renderer
texture (contain UI)
boardtexture, created from boardsurface (contain drawing board)
pixeltexture, created from pixelsurface (contain drawing board)
tempsurface
LOADING USER INTERFACE AND VARIABLES DECLARATION
set texture as the render target, so to reload user interface we only copy texture to renderer
MAIN LOOP (always updating renderer)
QUIT
Quit the program, end loop
KEY DOWN
Draw pixels on pixelsurface -> pixeltexture -> renderer
MOUSE BUTTON DOWN (functionalizing buttons)
NEW
OPEN
Open *.esc file, loading 2 related images to the boardsurface and pixelsurface
SAVE
Drop down SAVE IMAGE and SAVE FILE
save png or bmp
save *.esc and 2 images
HELP
LINE COLOR
Use function to pick color
LINE WIDTH
BACKGROUND
Load color or image on boardsurface -> boardtexture -> renderer
SPEED
FREE RESOURCES (window, renderer, textures, surfaces)
End.

Macro design for this program includes funtions for color manipulation, loading images, saving and opening configuration files, and a main function to render graphical context and wait for events. In the main function, the program executes all the necessary initializations and set-ups (SDL_Init, SDL_Window, SDL_Texture….), then it loads user interface from Images directory and declares variables for use in the main loop. The program then feeds events in the main loop until the user click the X button on the top right corner of the window, it then gets out of the main loop and release all the resources.

It is very important to release everything that has been initialized. If the program is to further updated, it should be easy to navigate through the codes and change the values.

IMPLEMENTATION

1. REALIZATION

In realization stage, I wrote the source code in a C file and carefully stored it in different hardwares. I quickly commented all execution steps as illustrated above, and wrote the code separately, step by step.

As mentioned above, the SDL_Delay function is executed, but at the same time the program keeps feeding events, resulting the loop still going even if there is no input data from keyboard or mouse. However, with the help from Yannis Ioannidis I was able to fix this problem.

I intended to store surfaces into the binary *.esc files, but this failed. With the help from Unit lecturer Eike Anderson I was able to address the problems by storing two related image files.

2. TESTING

The testing has been done in Linux operation system, on my personal laptop and on computers in the NCCA First Year Lab in Bournemouth University.

A. Opening the program

The program opens successfully all computers and laptop tested. It shows a window and loads graphics without failure.

B. Drawing operation

The drawing operation, even though successfully executed, gets slower the longer the program running, which then may result in the system killing the program. This is tested on my Dell laptop, Linux Centos 7. In the lab the program sometimes gets slower, but the system does not terminate it.

I suggest the problem **may** lie in SDL_FillRect command, which has relatively slow operating time but need to be multiply executed every time the keys are pressed. This can be changed to manipulating individual pixels, however, we would need another drawing algorithm for changing the line-width.

C. User input (file name)

The program successfully allows user to type input data into the terminal without failures, even if the file name is not existent in the directory (it will print out errors).

RESULT