

Assignment Two Report

Name: YANG LINBIN

Student ID: 0540137

Student email address: 2830406835@qq.com

1. Introduction

This is my second homework for the course Assembly Language. This assignment requires us design and build a simple game project. This project has 5 function modules. The user should be able to choose from those options that decide which function module to perform. Compared to the first assignment, this one is more complex. It is not the various instructions that make this assignment difficult to solve, but the design methodology behind it. Not only do we need strong basis for assembly language but also we should have a main frame in our mind before code work. This report introduces to you the main structure of my program and the methodology of how I build it.

2. Structure Chart

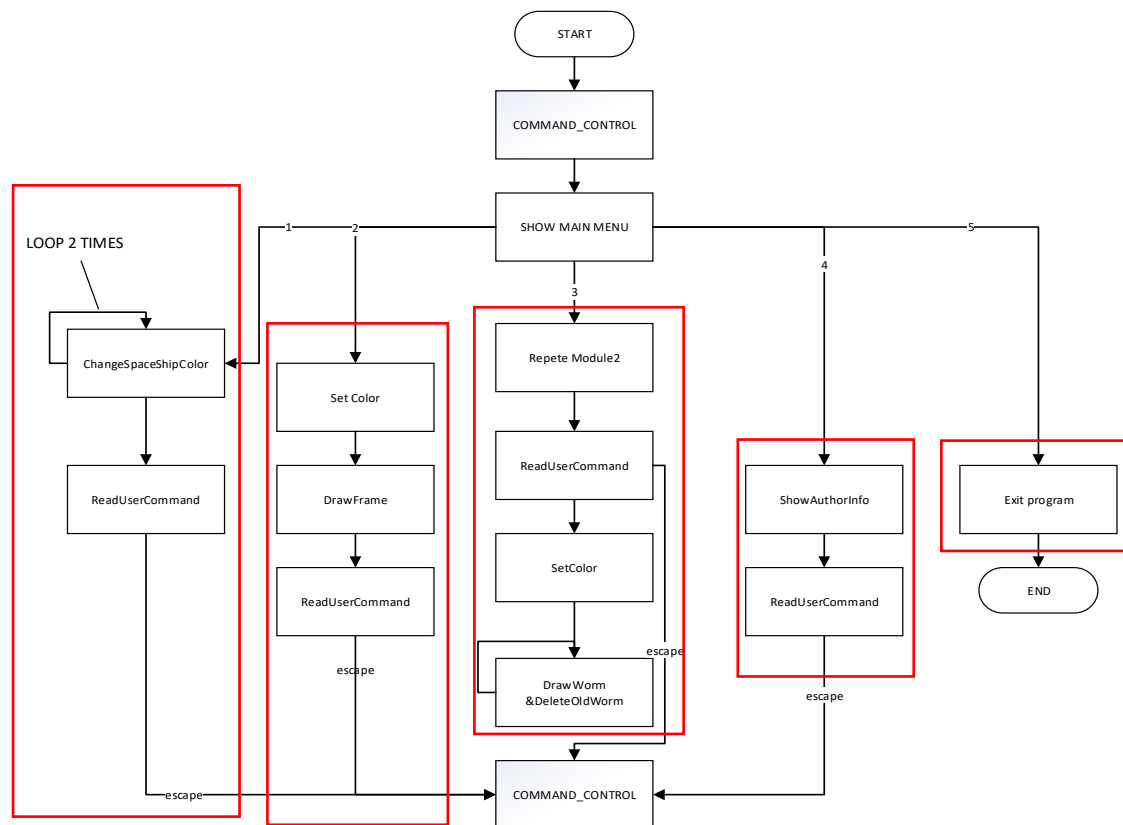


Figure 1

Altogether, there are 10 main components for the structure of my program:

- **CommandControl:** Main Function in program, it decides which function module to perform.
- **ReadUserCommand:** Read User's choice through monitoring keyboard event
- **ChangeSpaceShipColor:** Main function for module one, draw three boxes
- **DrawFrame:** Automatically draw a frame around the command screen.
- **ShowMenu:** Show main Menu for user
- **ShowAuthorInfo:** Print the author's basic information
- **DrawWorm:** Draw worm using the current location parameters
- **DeleteOldWorm:** Clear worm use old location parameters
- **SetColor:** Change the background and font color (Blue Yellow Green)
- **ExitProgram:** Exit the whole program

3. Flow Chart

This chart shows the main structure of my program. It leaves out the details of function modules.

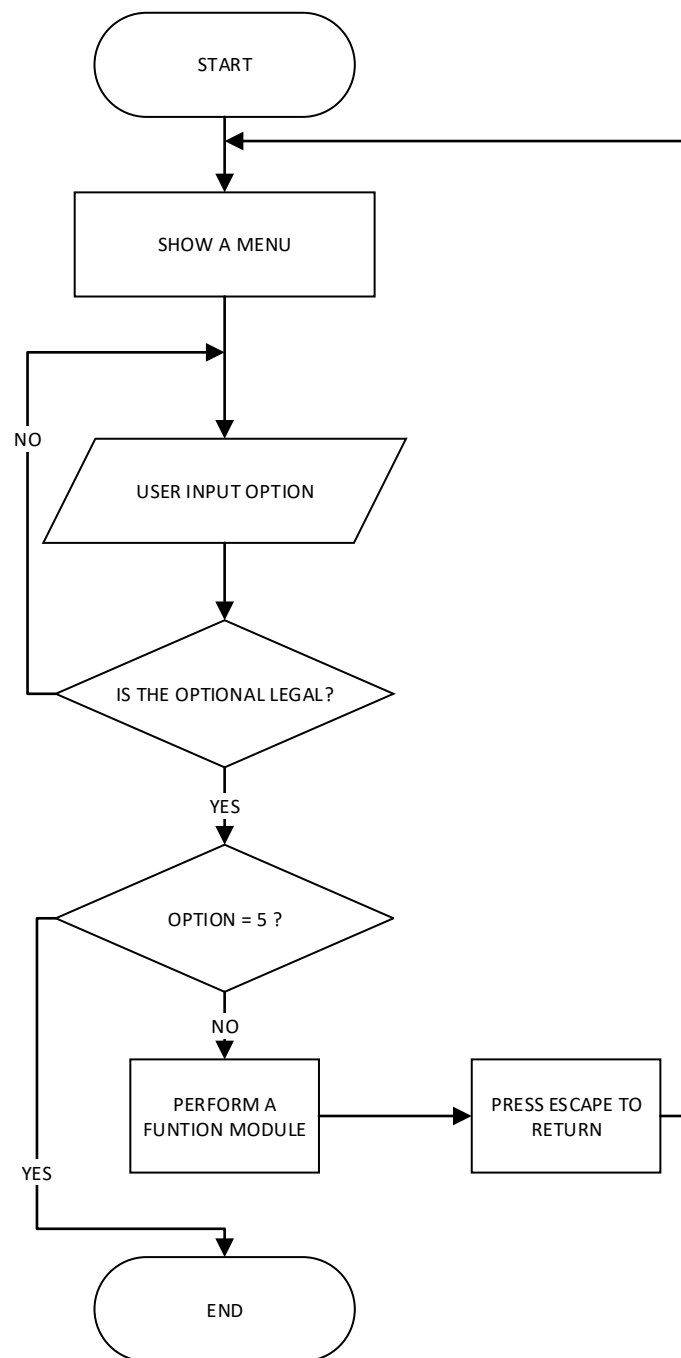


Figure 2

This chart below shows how a frame is draw in command window, it uses four loops to draw each side of the triangle. We should note that in order to show the process of draw the frame we must set time delay for each loop.

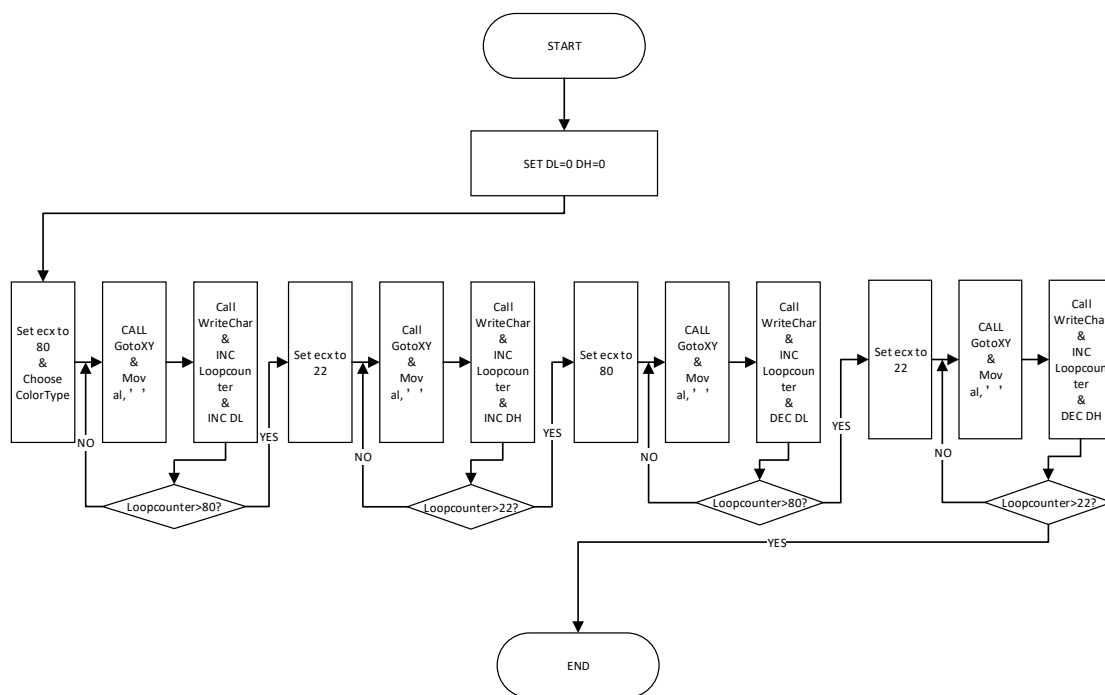


Figure 3

This chart below shows the third module which allows user to use 'e' and 'c' button on the keyboard to control the 'worm' in the screen and the worm itself can automatically move from left to right.

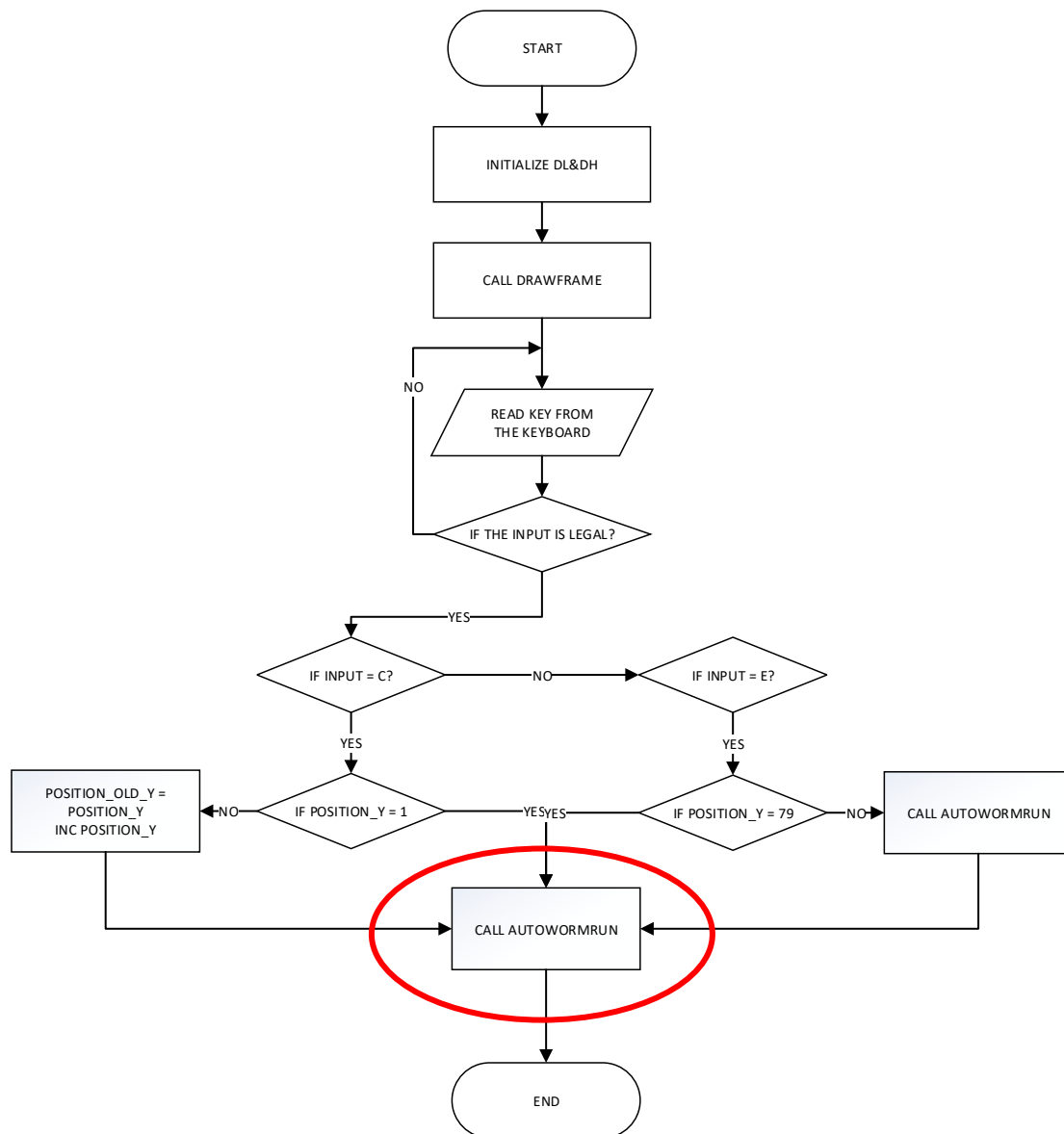


Figure 4

For the AutoWormRun function, the flow chart of it is showed in the next page.

This is what AutoRunWorm function does in the whole program, it change the current position of worm on X-axis. Together with module3 (Shows in Figure3), position_y is changed according to the control button you press. Then the current position info and past position info is sent to DeleteOldPosition and DrawWorm, then the worm “move” on the screen!

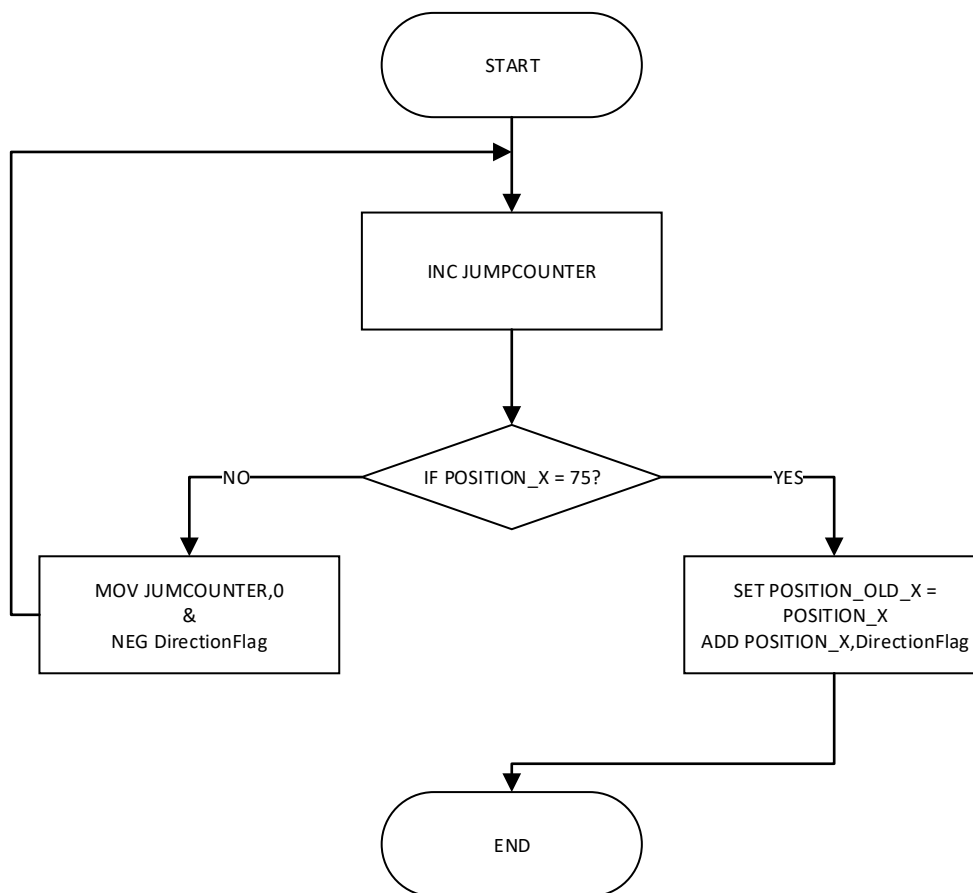


Figure 5

This chart shows how I draw the window of the module1 (See Figure 5). Also we must note that in order to draw three 3 X 2 boxes, we must run the program circled in the black box for two times.

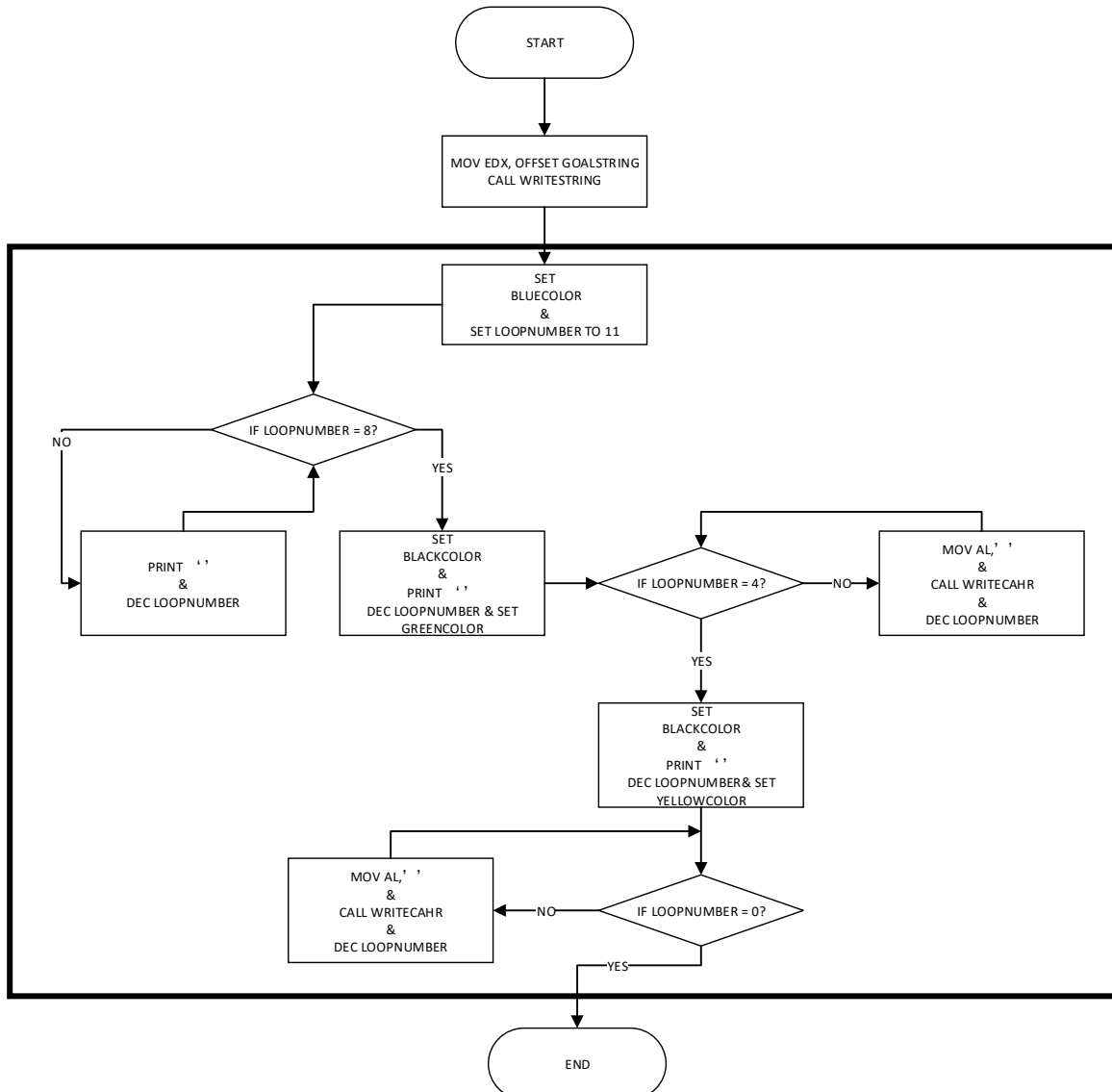


Figure 6

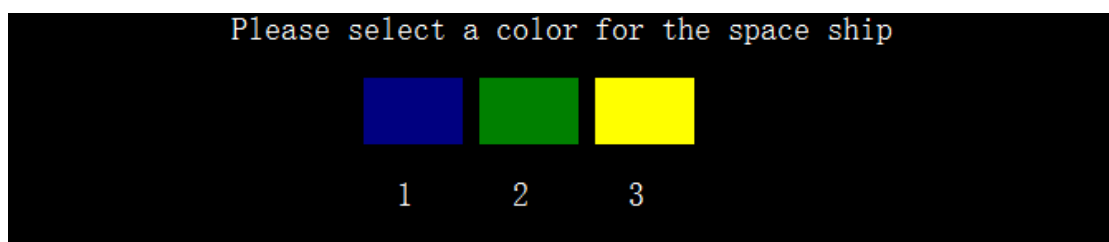


Figure 7

4. System Architecture

As is can be seen from the structure chart (See Figure 1), this program achieves a small game system. It shows the main menu and let user input their choice then it perform corresponding function module.

If the user does not want to play it in sequence (input 1-2-3-4-5), the program would set ColorType to 1 and this means the default color of frame and spaceship is blue. However, if the program runs as the designer expects, it should go like these:

First, the module one asks user to input their choice for the color of frame and spaceship. After making choice (let out a sound when finished), it automatically returns to the main menu.

Second, it draws a frame automatically around the command screen of which the color is already set in module one. The direction of the running label is positive rotation.

Third, the program repeat module 2 without time delay, then it draws the worm in the frame. The user can press 'e' and 'c' to move the worm up and down and the worm itself can automatically move on the X-axis.

Fourth, clear the screen and show the Author's basic info, user can press 'Escape' to return to the main menu.

Last, when user press 5, the whole program exits.

5. The approach

For this part, I will pick some typical problems I encountered while building this project and explain my solutions to these problems.

- How to combine the discrete modules together?

I use two ways to fix this problem:

1. Use intermediate variable to link different modules.

As we know, there are logic relations between some modules, for example, module one decides which color to use in the next two modules. They also exists between some components mentioned in Chapter 2, take DeleteOldWorm and DrawWorm as an example, both of them must share the position information of the worm.

I use variable ColorType to record the choice user makes in module one. Then the next two modules choose their color according to the value of ColorType. (1-BLUE 2-GREEN 3-YELLOW).

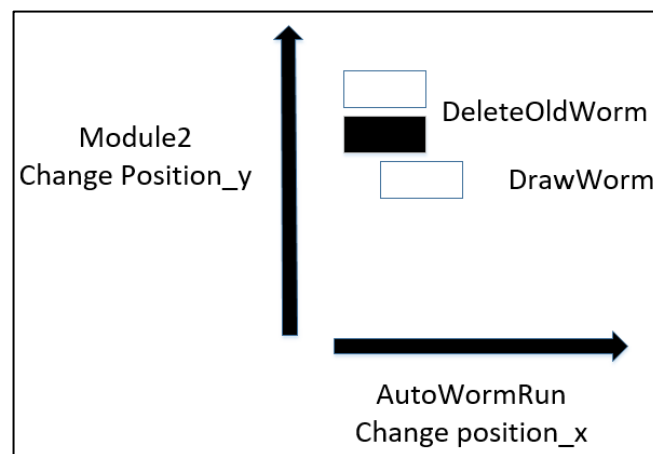


Figure 8

In order to control the position of the worm and make different components cooperate well, I use position_x (y) to record the current position of the worm and Position_Old_X (Y) to record the previous position. Module two and AutoWormRun share and change the variable of these four variables to control the position of the worm (See Figure 8).

2. Write a command control function using which...case structure.

In order to make the structure of this program more clear and link five function modules with the main menu, I apply which...case structure in CommandControl function to achieve this. The structure is easy and very practical (See Figure 9):

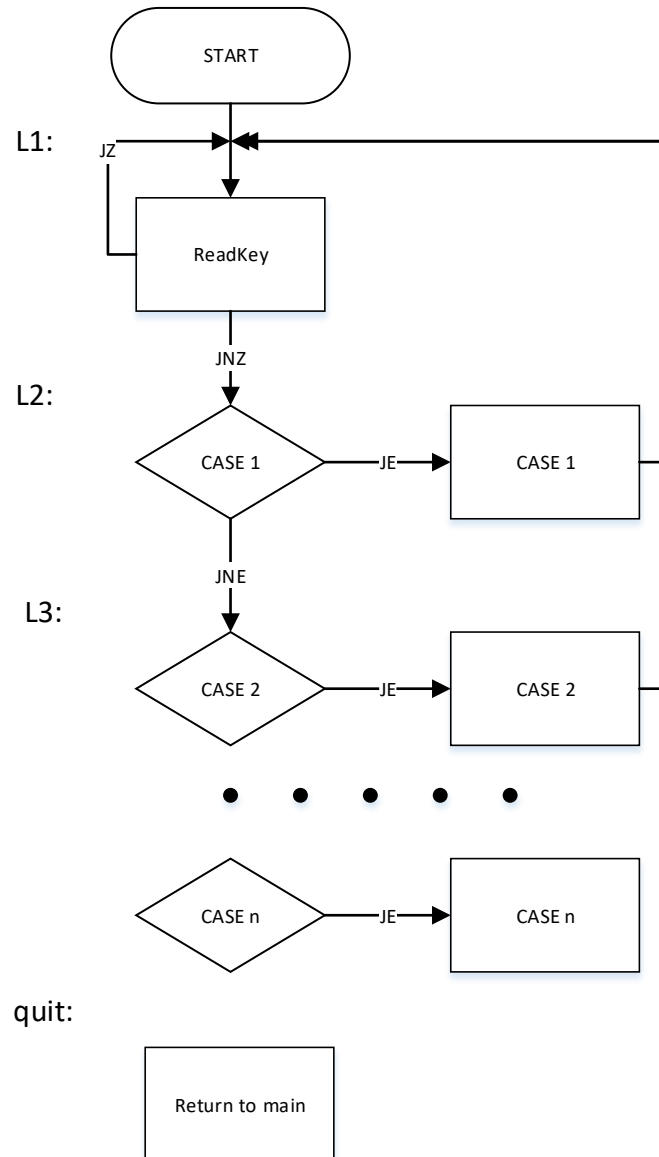


Figure 9

- How to avoid blink effect while move the worm?

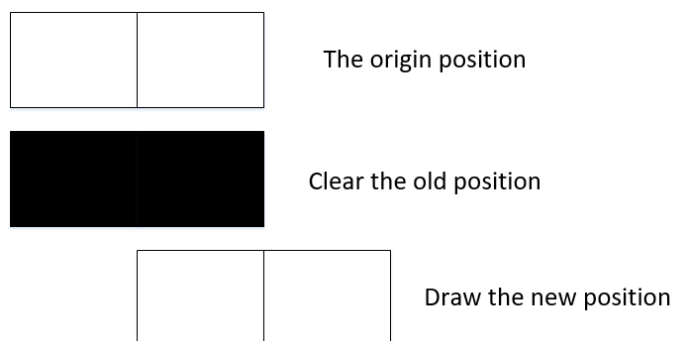


Figure 10

We can see from Figure 10 that we do not clear the screen every time the worm changes its position. We use `Position_Old_X (Y)` to record the previous position of the worm and before worm changes to new position we use shade to cover the original position then draw the worm on the new position, so the worm starts to “move”. This method skillfully solve the problem of blink of the screen.

- What does “ReadKey” do?

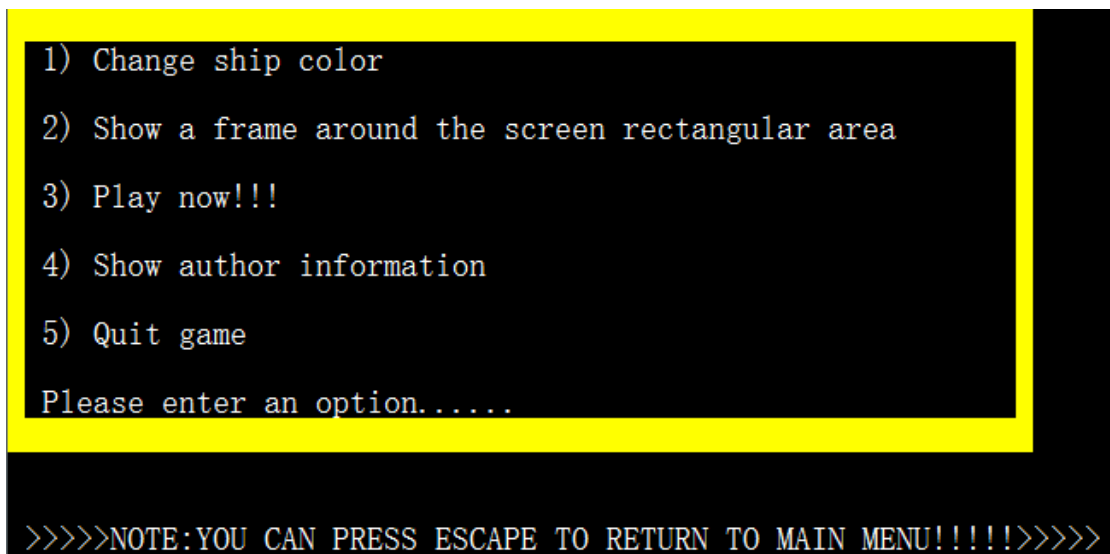
`ReadKey` is an internal function of ASM. It monitors the keyboard events and store the ascii code of the input value. For example, when we press ‘Esc’ on the keyboard, then `Readkey` writes 27 to `al`. Also we must note that when no keyboard event does not exist, the ASM would skip `ReadKey`, so usually we set an infinite loop outside the `ReadKey` to ensure that our input is received by the program.

6. Discussion/Experiments

For this part, I would input 1 to 5 in sequence to show you the performance of my program. I will also point out little problems and give scheme to fix them.

1. The main Menu

I add additional notes (See Figure 11) on it to make my program more friendly.



```
1) Change ship color
2) Show a frame around the screen rectangular area
3) Play now!!!
4) Show author information
5) Quit game
Please enter an option.....

>>>>>NOTE:YOU CAN PRESS ESCAPE TO RETURN TO MAIN MENU!!!!>>>>>
```

Figure 11

2. The first module

This window waits for the user to select from three color. (See Figure 12)



Figure 12

>>>>>>>>>>>>>GREEN>>>>>>>>>>>>>>

A large black rectangular area covers the majority of the page, starting below the header and extending nearly to the bottom and right edges. This appears to be a redaction or a placeholder for content that has been obscured. The word "GREEN" is visible at the top left of this area.

The program automatically draws a green frame for the command screen in positive rotation (See Figure14).

```
>>>>>>>>>>>>>PRESS ESCAPE TO RETURN>>>>>>>>>>>>>
```



```
Student ID: 0540137  
Student Name: YANG LINBIN  
Student Email Address: 2830406835@qq.com  
>>>>>>>>>>>>>>>PRESS ESCAPE TO RETURN>>>>>>>>>>>>>>>
```

请按任意键继续. . .

During test procedure of my program, I found this little bug (See Figure 19). Every time I start again from option 1 which means I want to reset my game. However, when I enter module 3, the position of ship is not on the northwest corner. It is apparent that I forget to initialize dl and dh when user enters module3.



Figure 19

7. Conclusion

After finishing both code work and report for this assignment, not only do I understand more ASM instructions, but also I develop my comprehensive thinking ability. This work requires us design a small game system, we need to have detailed scheme in our mind before start to code or we could confront with a lot of problems and waste a lot time debugging them. As the old saying goes: Sharpening your axe will not delay your job of cutting wood.

We have finished all the materials for ASM language and I feel that we may not have chance to apply it to our work in the future, because high-level language is more easier to write with. However, ASM language is a good tool for us to know how your computer run instruction on hardware level and this would definitely help you to write more efficient program using high-level language. So grasping the knowledge of the ASM language is quite meaningful for us.

In conclusion, although this assignment is a big challenge for me, I really enjoy the process of building it!