Assignment Three Report

Menu

Introduction

This homework requires us to use assembly language to call C procedure and achieve simple image processing. This task is very challenging, for it involves some fields that we are not familiar with. For example, we need to call C and load picture to our database and print it on the screen. We should know some basic knowledge about the way image is stored in memory so that we can know how to realize such goal. Also it is very important to know about how assembly language cooperates with C, it would help us to know the running mechanism of the template. Once we understand the function of each part of the program, it is easy for us to meet the homework requirements using our own way. In sum, this is a tough work. Below is my report of this programming work, it can help you to know how I design and build my program.

Structure Chart

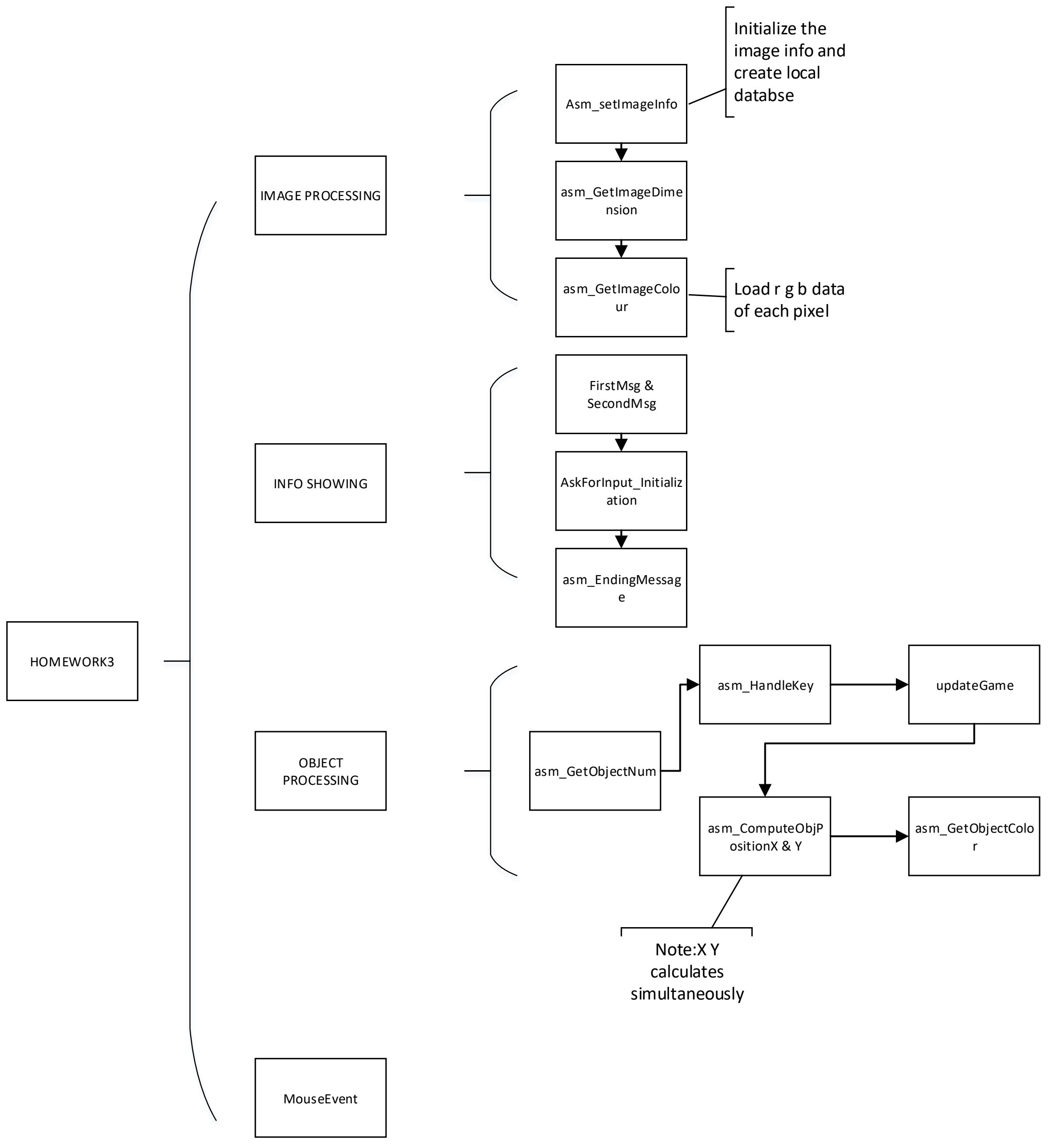


Figure 1

I divide this program into four main parts with 12 components. Mastering the structure of the program is so important because it helps you to know the running mechanism.

- asm_SetImageInfo: It initializes the picture we want to load into our local database including the length, height, RGB info and so on.
- asm_GetImageDimention: We need this because height and width are two important variables and commonly used in other components.
- asm_GetImageColor: After loading data of the image to our local database, we should divide each pixel into RGB data, so that it can show on the screen.

- AskForInput_Initialization: Some info that helps user to know the rules of operating this program shows at the very beginning of the program.
- First & Second Msg: This component asks user to input the speed and lifecycle of sphere or it would use the default value.
- asm_EndingMsg: After user press ESC, it show a Msgbox and then the program exits.
- asm_HandleKey: It monitors the keyboard event, and directs this program to different parts according to different keys it receives.
- UpdateGame: This is the most important part for this program, it change the XY position of sphere and record the position of each sphere.
- asm_ComputePositionX&Y: It gets position data from the local database, so that OpenGL system would move the sphere.
- asm_GetObjectColor: Use random color or rainbow color to beautify our sphere.

FlowChart

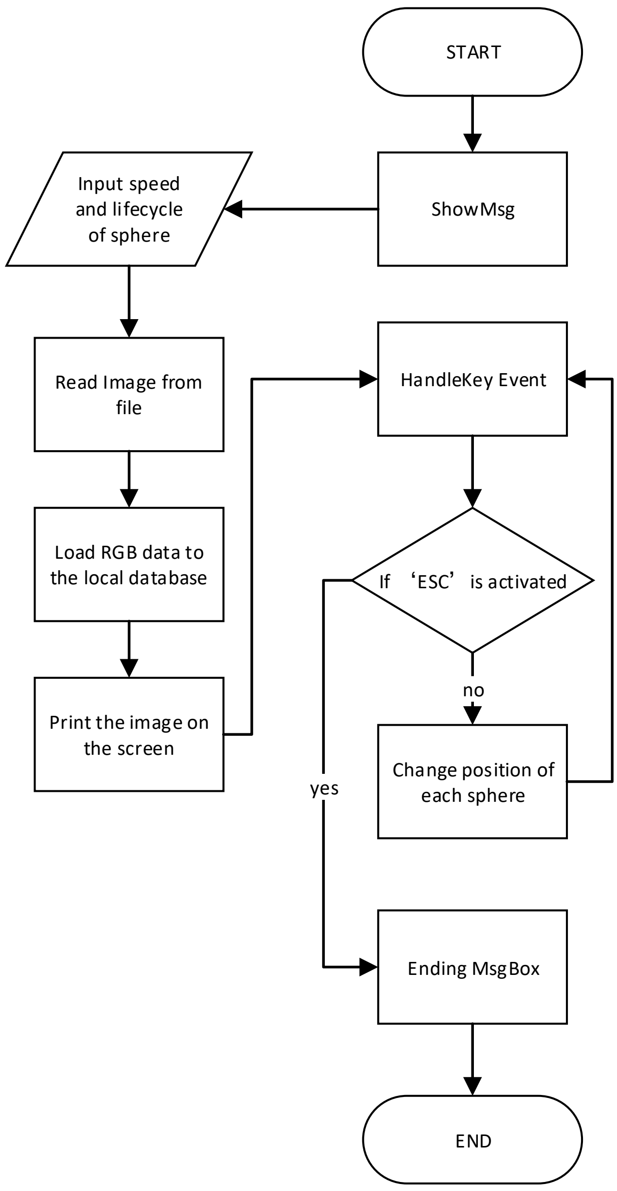

START
ShowMsg
Input speed and lifecycle of sphere
Read Image from file
Load RGB data to the local database
Print the image on the screen
HandleKey Event
If ‘ESC’ is activated
no
yes
Change position of each sphere
Ending MsgBox
END

Figure 2

Figure 2 shows the main flowchart of this program, it illustrates how the image presents on the screen and the way we operate the sphere.

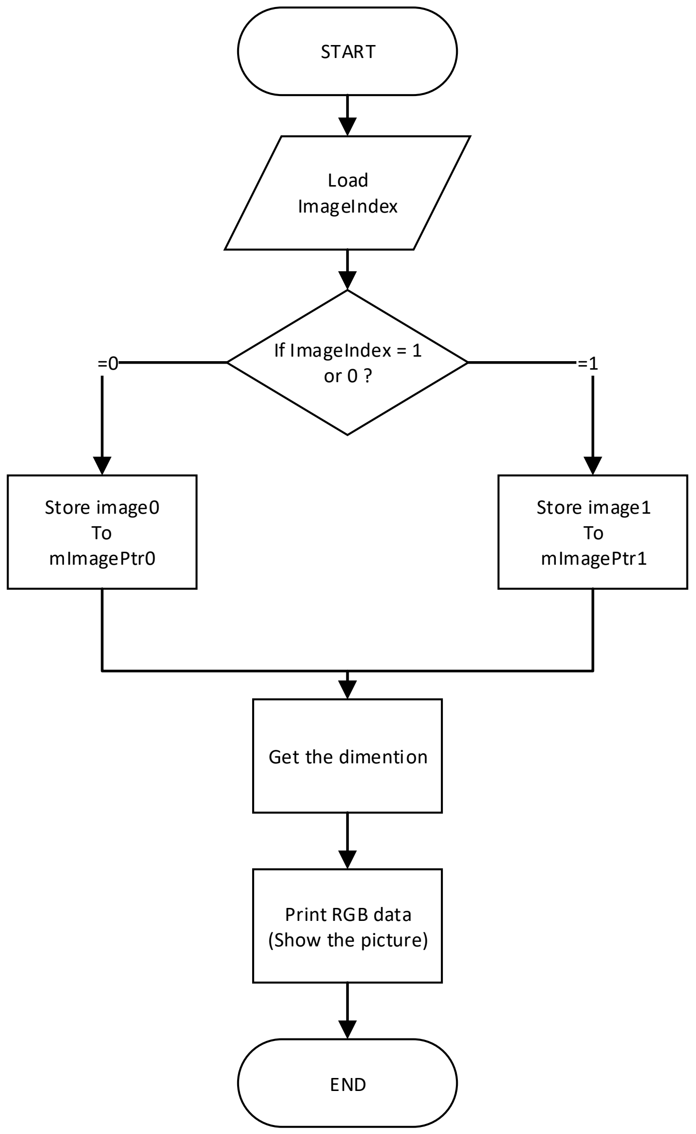

START
Load ImageIndex
If ImageIndex = 1 or 0 ?
=0
=1
Store image0 To mImagePtr0
Store image1 To mImagePtr1
Get the dimention
Print RGB data (Show the picture)
END

Figure 3

This program actually reads both image0 and image1 and it loads any of them according to the imageIndex we set. After the RGB data is stored, we just return them to the OpenGL subsystem and the image shows on the screen.

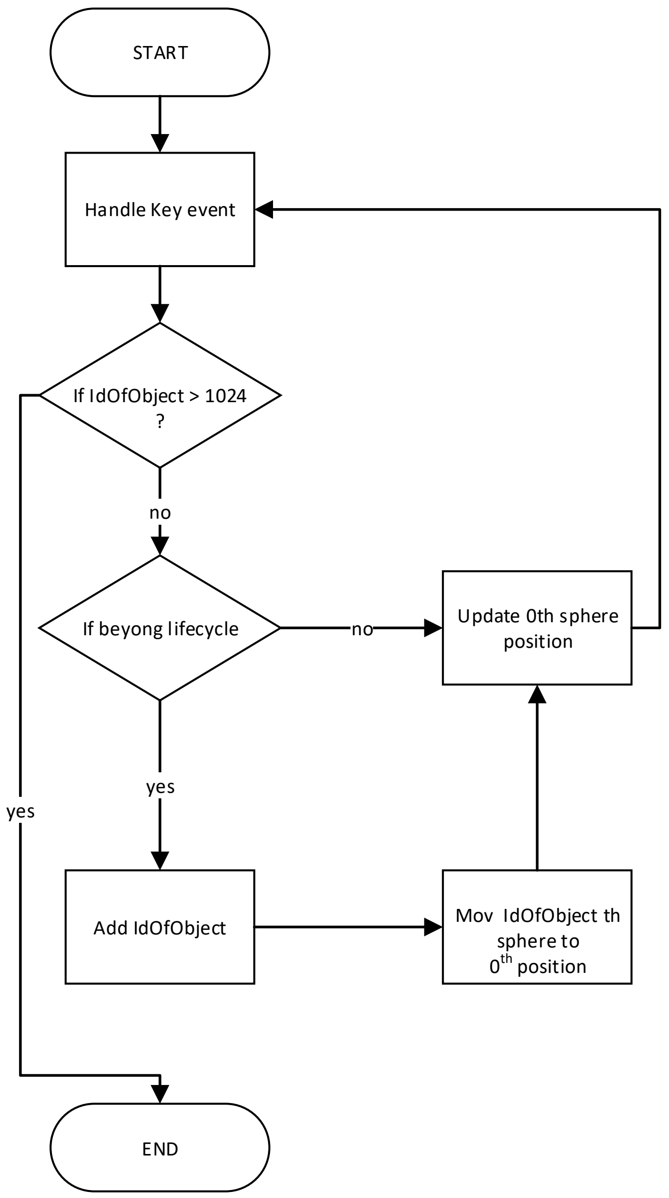


Figure 4

This figure shows how to control the sphere on the screen and I think it is the most complicated part for this assignment, because you must adjust some variables like speed and lifecycle to make the sphere run just like one by one on the screen. Besides, you can only control the first sphere and you must always update the position of other spheres to make the whole spheres run like a "snake". Precisely, at the end each lifecycle, we move the i^{th} sphere to the location of 0^{th} sphere.

System Architecture

The OpenGL system involves some subsystems to cooperate with. Just like the structure diagram shows in PartⅡ.

First, the system shows message on the screen indicating some basic rules for operating this system. Then it asks user to input the speed and lifecycle of the sphere or it would use the default value. (In my program, the default value is Speed: 65 lifecycle: 30).

Second, image is loaded into our local database according to the imageIndex we set. The local database stores RGB data and then we call the C program to print them on the screen so that the image is presented.

Third, Handle_KeyEvent monitors input from the keyboard, according to the input, the system changes the position of the sphere and realize other functions such as change random color to rainbow color. Note that we only directly modify the position of the 0th sphere. Other spheres just follow the 0th sphere on the screen.

Finally, we press ESC when we want to exit the program, the program first shows a MSG box and then exit.

The approach

- How does the sphere generate? How to control each sphere?

Solution:

At first, you should set the number of objects to 1024 (any number you want), it actually sets the number of "ball" that would show on the screen. The positions of all spheres are stored in ObjPosX and ObjPosY.

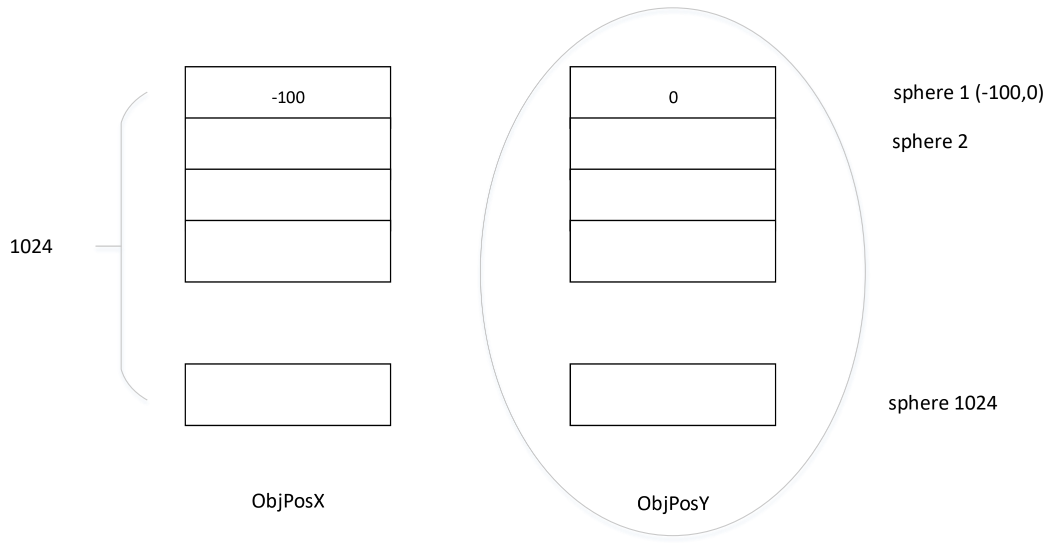


Figure 5

In order to make it move like a "snake", we should set two flags: lifecycle and speed. The function UpdatePosition is circularly called. We modify the position of the first sphere and if the calling time reaches the lifecycle of the 0th sphere, then the next sphere would be automatically move to the present location of the 0th sphere. After that, we continue to move the 0th sphere. So the following 1023 spheres just play a role recording the running path of the 0th sphere.

- How can we show the image on the screen?

Solution:

First, we should know some basic knowledge of images. We know that image can be divided into w*h pixels and each pixel has three color elements: Red, Green, Blue. So what we should do is to extract the RGB data from the image and then let the OpenGL system to print them on the screen. Then we should have the image as the background of our system.

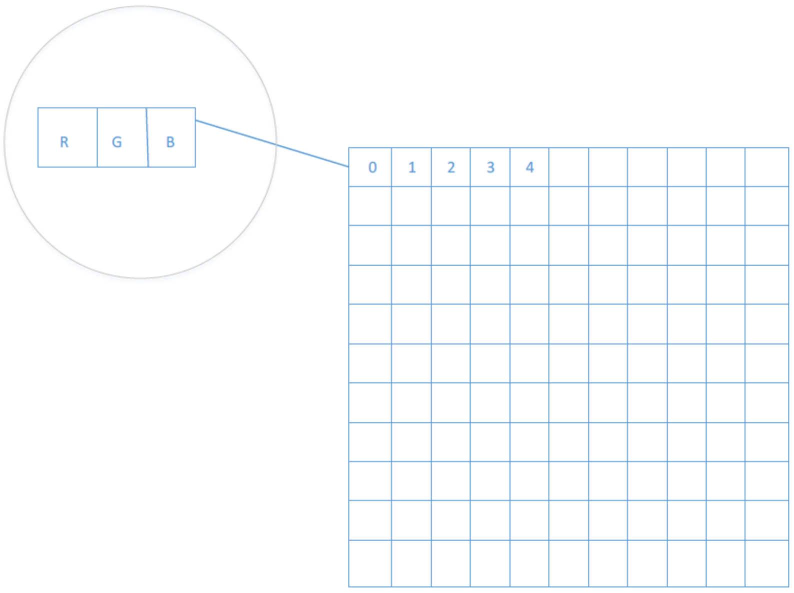


Figure 6

Another point I want to stress is that we must have a deep understanding of how to deal with two-dimension array using assembly language, because the image data is stored using this form (See Figure 6). Also note that RGB data is store in general registrations, and we must extend intermediate variables to proper size before we transfer them to the OpenGL system.

- How to let the sphere move in the direction of your mouse click?

Solution:

When I try the example exe file in the folder, I find every time when I click on the screen, it shows an axis position in the backend. However, the sphere does not move in this axis. Actually, the axis for the spheres on the screen which help us to locate each of them is like this (See Figure 7):

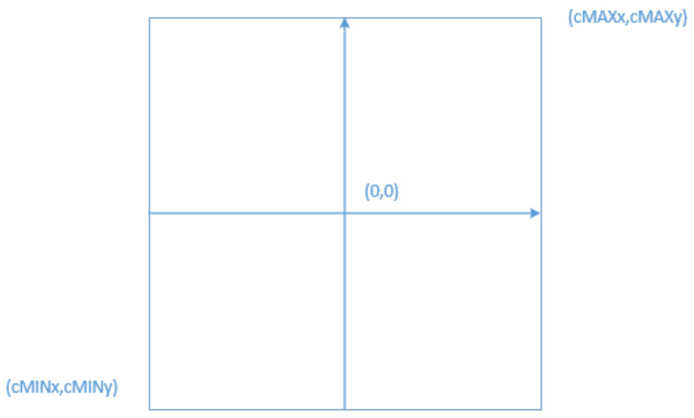


Figure 7

Here, our click sets the destination position for the 0th sphere. However, we must map this location to location in the real axis.

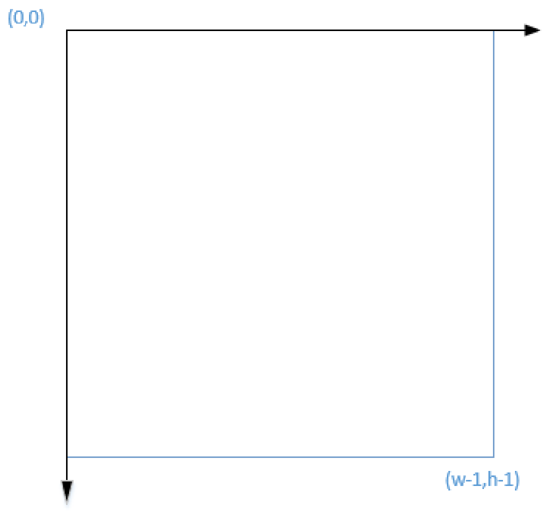


Figure 8 The “fake” axis

For example:

$$X_{REAL} = \frac{x_f}{w} * (cMAXx - cMinx) - \frac{(cMAXx - xMinx)}{2}$$

Where:

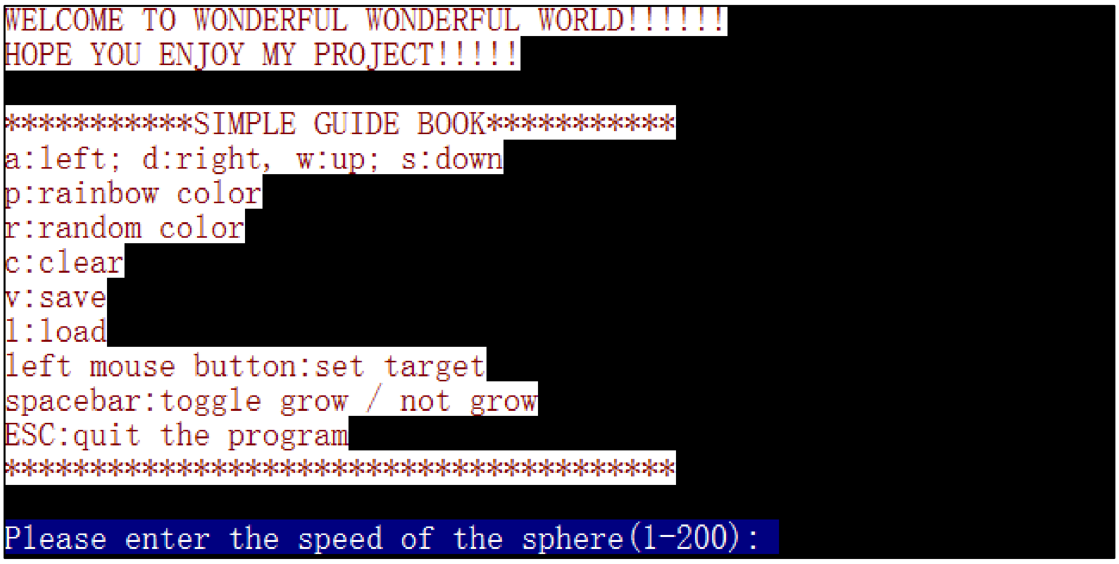
- X_{REAL} denotes x position in ‘real’ axis
- X_f denotes x position in ‘fake’ axis
- w denotes width of the image
- cMAXx largest x position in canvas
- cMINx smallest x position in canvas

● How to clear other sphere and leave only 0th sphere?

Solution:

I believe there are many ways to achieve this. For me, I just hide another 1023 sphere behind the 0th sphere everytime when I the user move the 0th sphere. It gives you a feeling that 1023 spheres are removed.

Discussion/Experiments



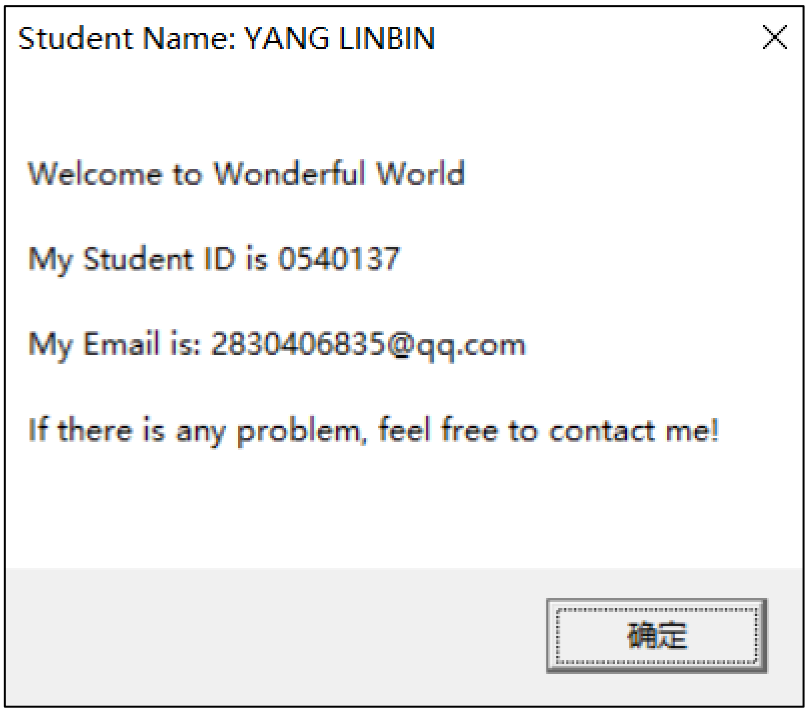
```
WELCOME TO WONDERFUL WONDERFUL WORLD!!!!!!
HOPE YOU ENJOY MY PROJECT!!!!!

***********SIMPLE GUIDE BOOK***********
a:left; d:right, w:up; s:down
p:rainbow color
r:random color
c:clear
v:save
l:load
left mouse button:set target
spacebar:toggle grow / not grow
ESC:quit the program
***************************************

Please enter the speed of the sphere(1-200):
```

Figure 9

Figure 9 shows the very beginning of this system. It asks user to input the speed and lifecycle of the sphere or the system would use the default value. In the blue area, the system outputs the information one character each time and there is a time delay so it creates the effect of a typewriter.



Student Name: YANG LINBIN

Welcome to Wonderful World

My Student ID is 0540137

My Email is: 2830406835@qq.com

If there is any problem, feel free to contact me!

Figure 10

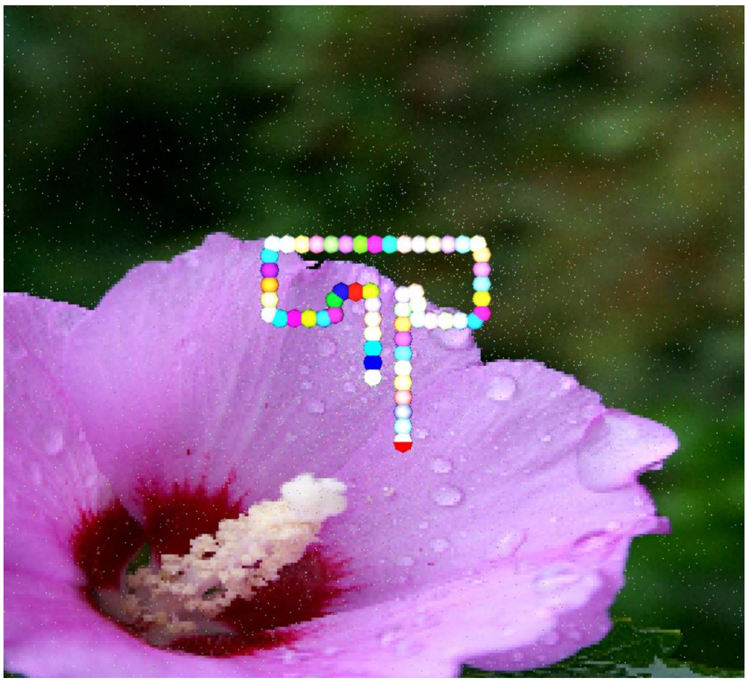


Figure 11

Figure 11 shows running state of my program. It loads the picture onto the screen. And the user can use WASD to control the movement of the sphere and it runs like a "snake" on the screen.

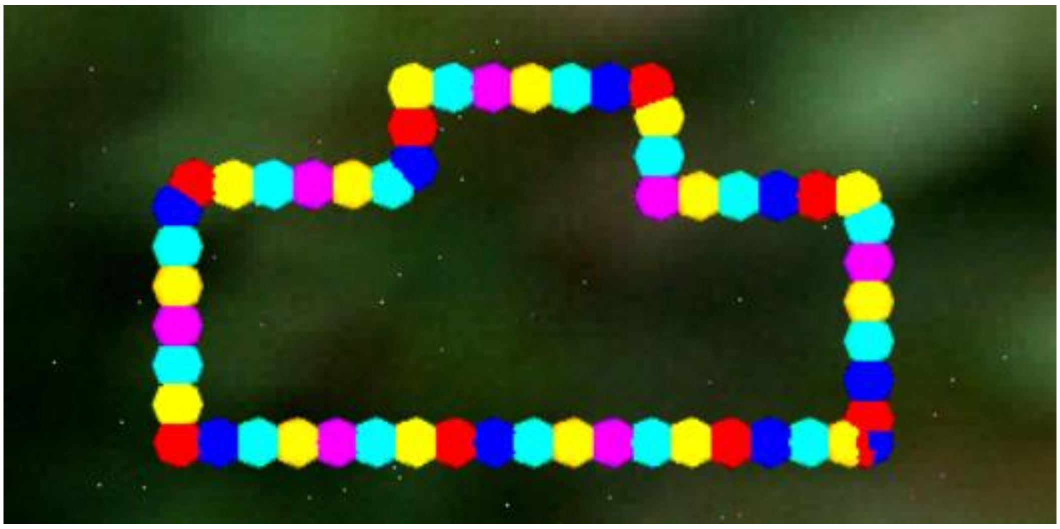


Figure 12

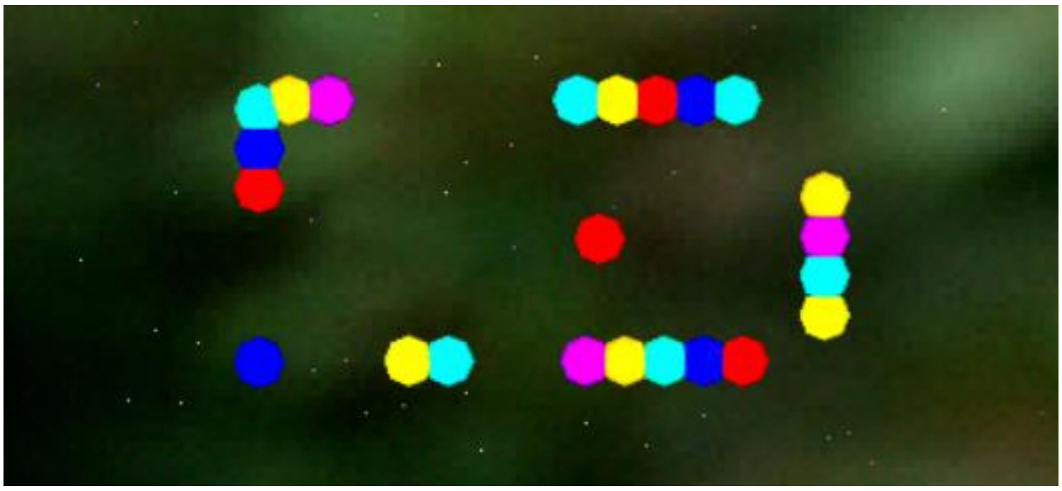


Figure 13

It would change color to rainbow when I press ‘p’ and those balls that follow the 0th ball would immediately stop when I press ‘spacebar’. (Show in Figure 12)

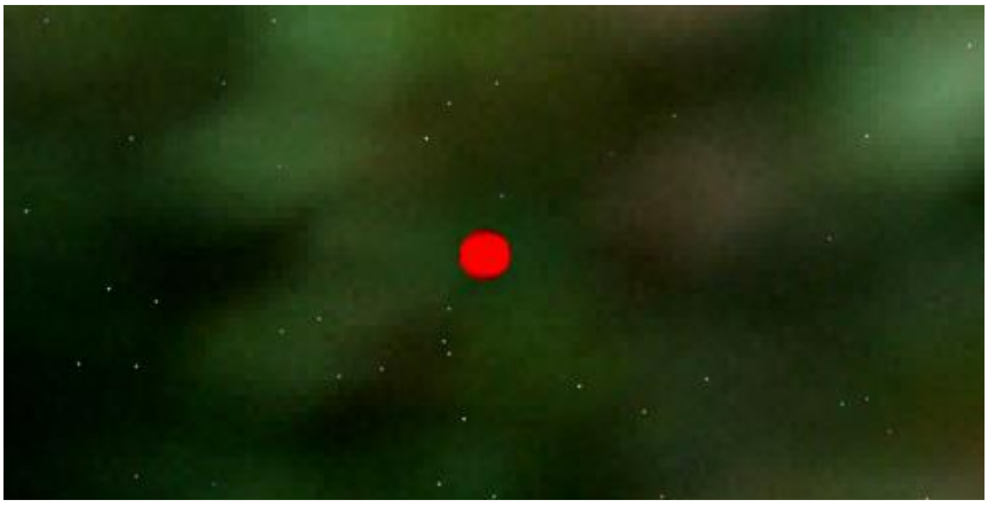


Figure 14

When I press ‘c‘, all other spheres is hidden behind the 0th sphere which make it feel like other spheres are removed from the screen. (Show in Figure 14)

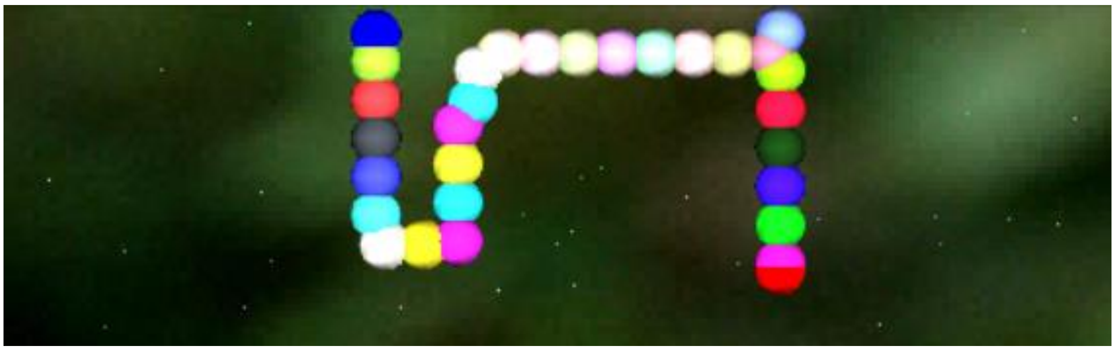


Figure 15

Figure 15 shows that when the sphere hits the bound, it would immediately goes in the diverse direction. This is because I change the variable moveDirection when the 0th sphere hits the bound.

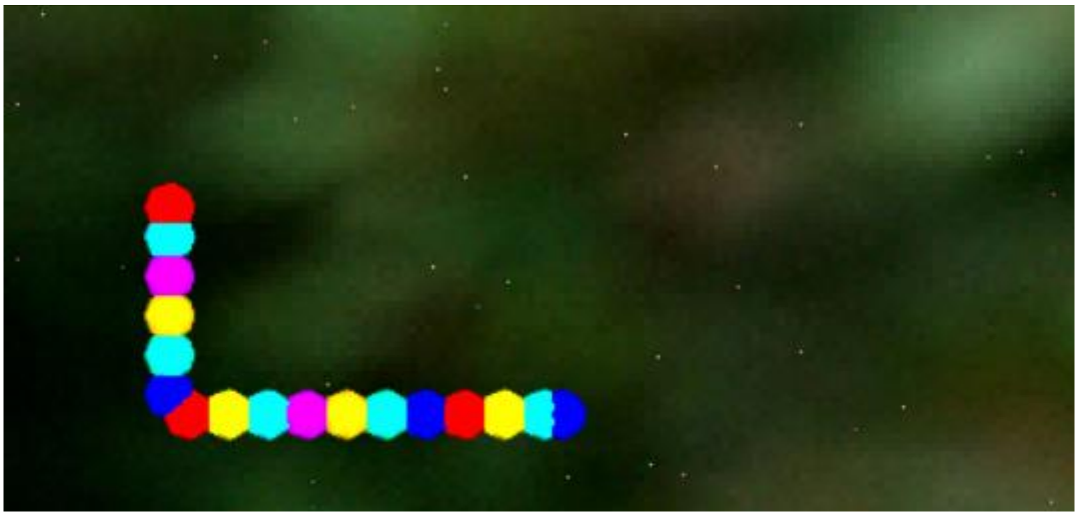
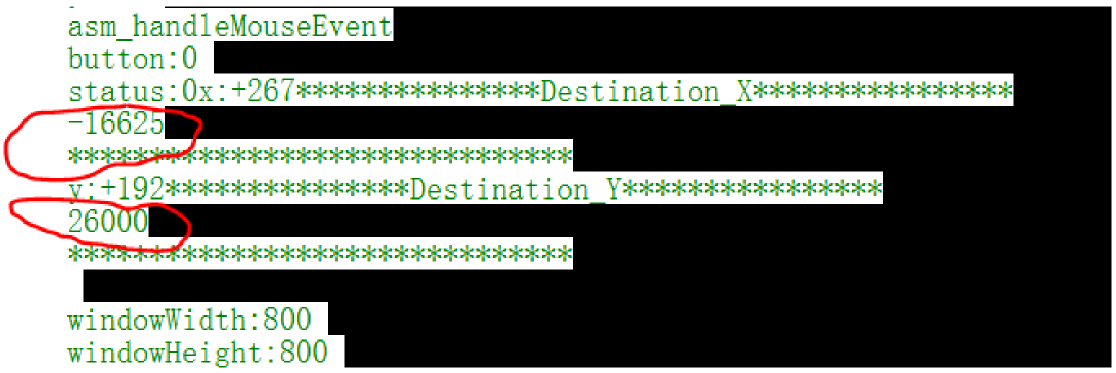


Figure 16

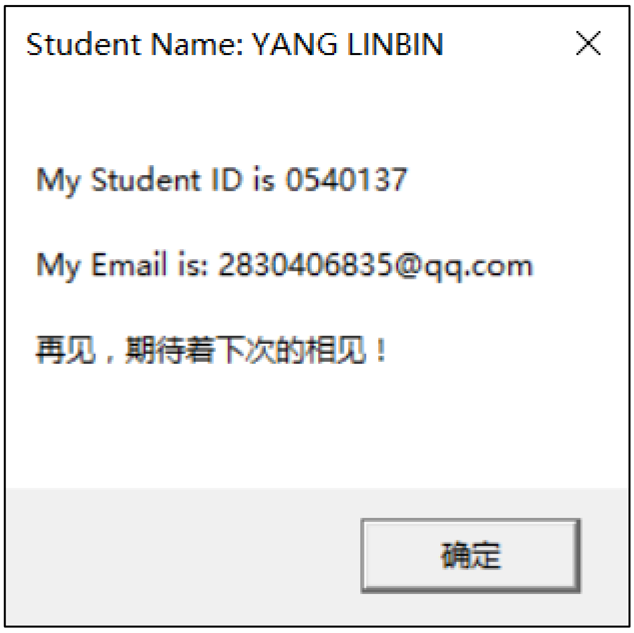
Figure 16 shows that when I click the on the screen, the 0th ball would move to the point where I click. The following picture shows the click position after the mouse click event happens.



```
asm_handleMouseEvent
button:0
status:0x:+267****************Destination_X****************
-16625
********************************
y:+192****************Destination_Y****************
26000
********************************

windowWidth:800
windowHeight:800
```

Figure 17



Student Name: YANG LINBIN

My Student ID is 0540137

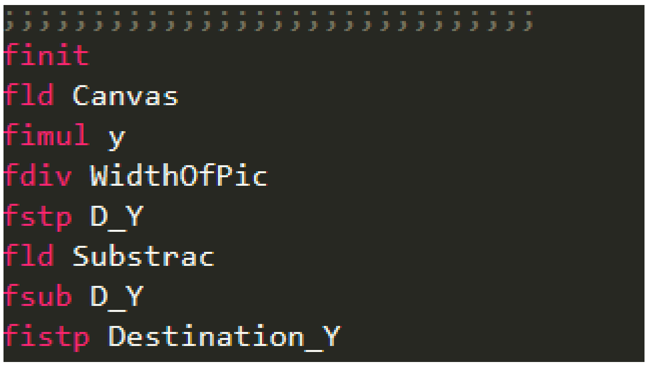
My Email is: 2830406835@qq.com

再见，期待着下次的相见！

Figure 18

As it is show in Figure 18, when I press ESC, it pops out MSG box showing the author’s information and then the whole program exits.

Speaking of the most challengeable part of this program, the first thing that comes to my mind is to monitor the mouse click event and direct the 0th sphere to the click point. The first trouble is how to calculate the exact position of the click point. It involves floating point calculation, what is more, the result would be stored in int memory. So a lot of details must be stressed when try to fix this problem. Below is my way.



```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
finit
fld Canvas
fimul y
fdiv WidthOfPic
fstp D_Y
fld Substrac
fsub D_Y
fistp Destination_Y
```

Figure 19

Use floating point stack, apply instructions like fimul, fidiv and etc. We want to store the final result in int memory, so we use fistp instruction. We set five flags for this part:

- flagFin: Monitor whether or not MouseClickEvent happens. When I click, it is set to 1 and when WASD is pressed, it is set to 0.
- flag1: It is 1 when sphere must move upforward.
- flag2: It is 1 when sphere must move downforward.
- flag3: It is 1 when sphere must move left.
- flag4: It is 1 when sphere must move right.

When the MouseClickEvent happens, everytime when UpdateGame is called, I compare the Desination_X(Y) with Present_X(Y) and make subtraction. When the result is below a certain level, then the program regards it move to the exact position.

Conclusion

This is the most challengeable task for assembly programing we have met so far. This assignment involves a lot of new knowledge and image processing is one typical example. If we do not know how an image is store in a computer system then there is no way for us to solve the problem.

The most impressive thing I felt during the processing of working on this task is that computer engineering is one interdiscipline. For example, how to convert the position of destination of 0^{th} sphere requires strong math background. Your artistic way of thinking can even play an important role in judging how you process your image. So we cannot limit our knowledge within one single field. Instead, we ought to dabble different kinds of knowledge and broaden our horizon.

Apart from this, we should be active and know the essence of every problem. It may require enough patience at the very beginning, but once you pull it through, you will find that you really benefit a lot from it!