**Assembly language and system programming**

Assignment \_\_\_\_03\_\_\_\_

Student Name:\_\_\_\_\_\_\_\_\_楊家安\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student ID: \_\_\_\_\_\_\_\_0516021\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student email address:\_\_\_\_\_\_\_\_\_gxxygxxy@gmail.com\_\_\_\_\_\_\_\_\_\_\_

**[10%] Introduction [ at least 100 words]**

**WORD COUNT:\_\_\_\_\_\_162\_\_\_\_\_\_\_ [ Must be filled or zero score]**

This is homework 03. To practice many things in chapter 9, such as movsd, lobsd, stosd.

The homework program controls a snake to move automatically. You need to input the speed and the life cycle of snake at first. The default is 100 and 25 respectively.

You could press ‘w’, ‘a’, ‘s’, ‘d’ to move up, left, down, right respectively.

You could press ‘p’ to change the snake into rainbow colors.

You could press ‘r’ to change the snake into random colors.

You could press ‘c’ to clear the current spheres of the snake.

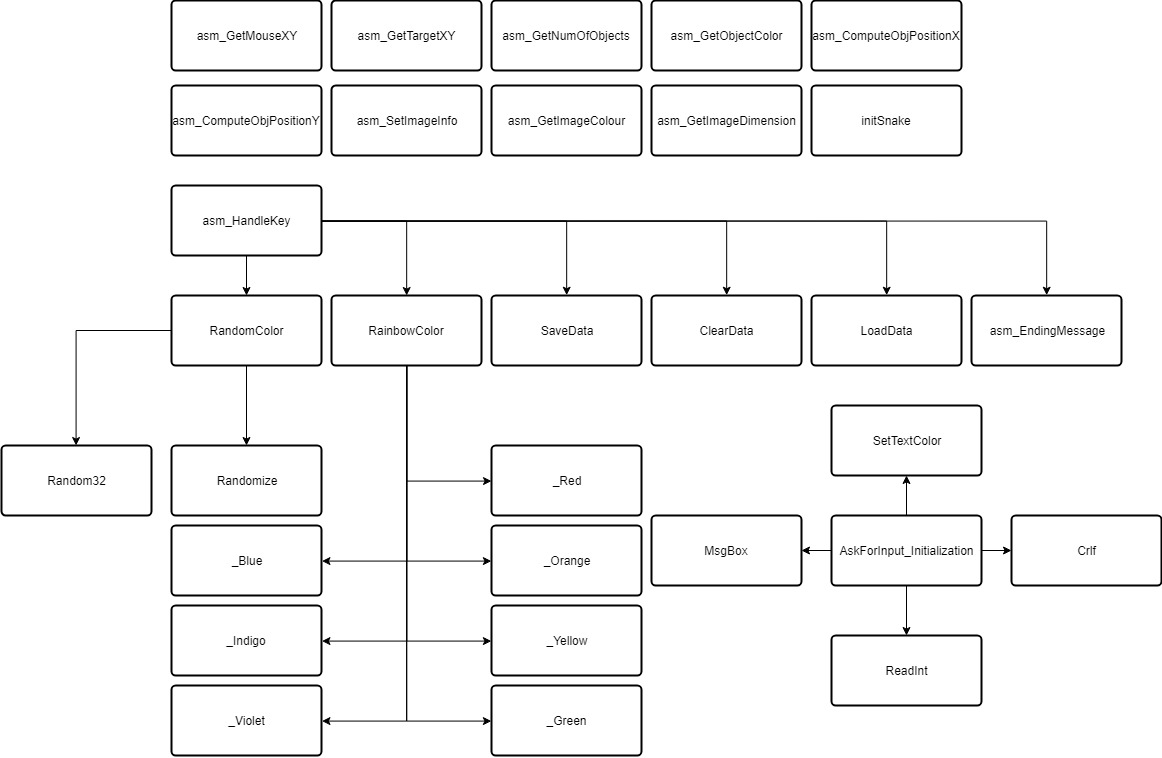
You could press ‘v’ to save the color and position of the current state of the snake. Then you could press ‘l’ to load them.

You could click the left mouse button to set the target. The head will move to the target. After hitting the target, the snake would stop moving.

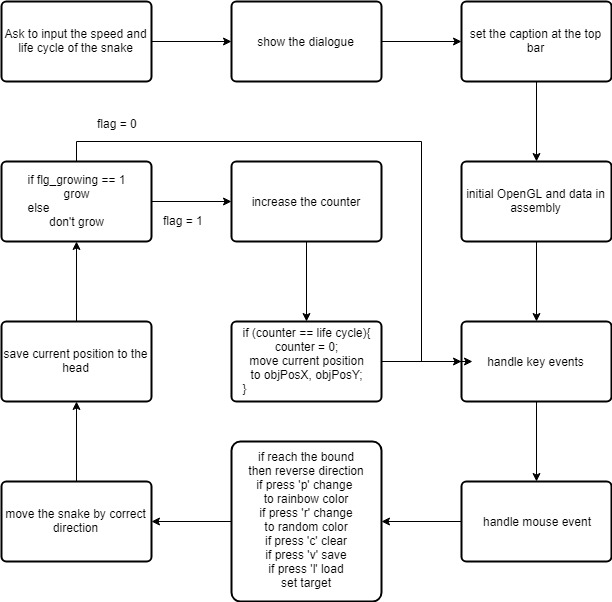
You could press spacebar to switch between growing and non-growing.

Press ESC to quit the program.

**[10%] System Chart** [at least 10 components]



[**10%] Flow Chart [**at least 10 components**]**



**[10%] System Architecture** [**at least 100 words]**

**WORD COUNT:\_\_\_\_\_\_\_127\_\_\_\_\_\_ [ Must be filled or zero score]**

There are many procedures in the assembly program, which is not modified:

asm\_ClearScreen, asm\_ShowTitle, asm\_InitializeApp, asm\_EndingMessage, asm\_updateSimulationNow, setCursor, asm\_GetNumParticles, asm\_GetParticleMaxSpeed, asm\_GetParticleSize, asm\_handleMousePassiveEvent, asm\_handleMouseEvent, asm\_SetWindowDimension, asm\_GetObjectType, asm\_ComputeRotationAngle, ASM\_setText, asm\_ComputeParticlePosX, asm\_ComputeParticlePosY, asm\_GetImagePixelSize, asm\_GetImageStatus, asm\_getImagePercentage, asm\_getStudentInfoString, asm\_GetImagePos,

There are some procedures that I have to modify:

asm\_GetMouseXY: no call

asm\_GetTargetXY: no call

asm\_HandleKey: call RandomColor, RainbowColor, SaveData, ClearData, LoadData, asm\_EndingMessage

asm\_GetNumOfObjects: no call

asm\_GetObjectColor: no call

asm\_ComputeObjPositionX: no call

asm\_ComputeObjPositionY: no call

asm\_SetImageInfo: no call

asm\_GetImageColour: no call

asm\_GetImageDimension: no call

AskForInput\_Initialization: call SetTextColor, Crlf, ReadInt, MsgBox

initSnake: no call

updateSnake: call GotoTarget

My own procedure:

RandomColor: call Randomize, Random32

RainbowColor: call \_Red, \_Orange, \_Yellow, \_Green, \_Blue, \_Indigo, \_Violet

\_Red, \_Orange, \_Yellow, \_Green, \_Blue, \_Indigo, \_Violet: no call

SaveData: no call

LoadData: no call

ClearData: no call

GotoTarget: no call

**[30%] The approach [ at least 300 words]**

**WORD COUNT:\_\_\_\_\_\_\_541\_\_\_\_\_\_ [ Must be filled or zero score]**

First, to read the image and show it at the background, I have to modify the asm\_SetImageInfo and asm\_GetImageColour, asm\_GetImageDimension.

asm\_SetImageInfo:

The image total size is w\*h\*3(for R, G, B of each pixel). I use movsb to move imagePtr(esi) to mImagePtr(edi). And move w, h, 3, 3 to mImageWidth, mImageHeight, bytesPerPixel, mBytesPerPixel respectively.

asm\_GetImageColour:

To return the RGB of the pixel, I lodsb the mImagePtr at the position (mImageWidth\*(mImageHeight-iy)+ix)\*3 for R,

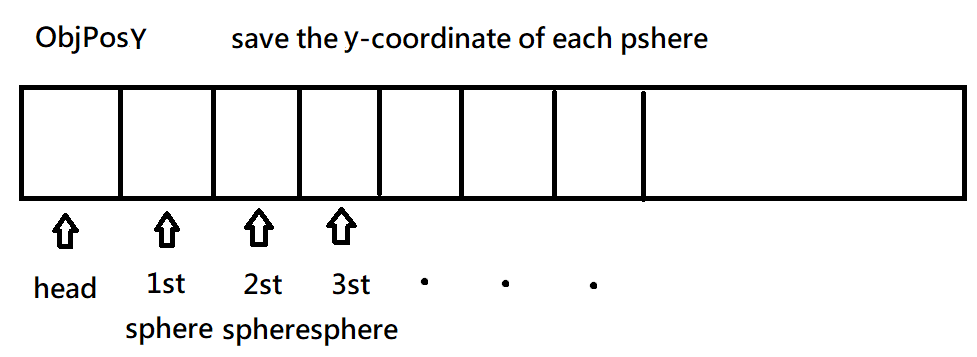
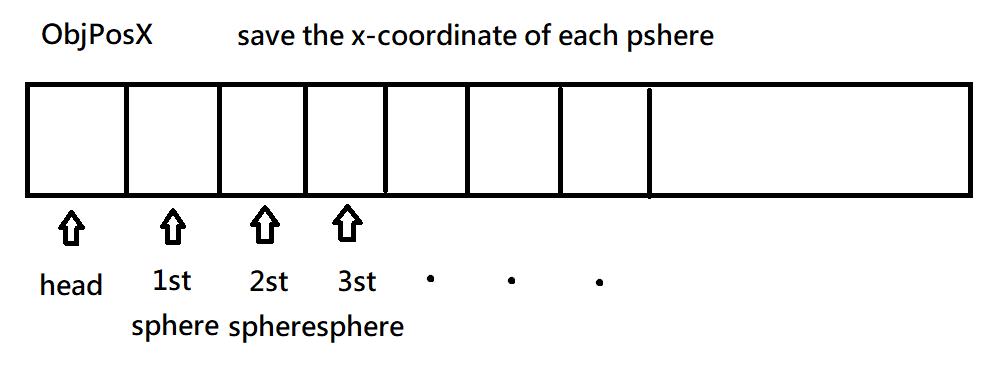
(mImageWidth\*(mImageHeight-iy)+ix)\*3 + 1 for G

(mImageWidth\*(mImageHeight-iy)+ix)\*3 + 2 for B.

asm\_GetImageDimension:

Move mImageWidth, mImageHeight to iw, ih respectively.

Second, to make the snake move, I need to complete asm\_ComputeObjPositionX, asm\_ComputeObjPositionY , and updateSnake.



asm\_ComputeObjPositionX:

Move [OFFSET objPosX + objID\*4] to eax (eax will return to C++ there).

asm\_ComputeObjPositionY:

Move [OFFSET objPosY + objID\*4] to eax (eax will return to C++ there).

updateSnake:

According to snakeMoveDirection(0,1,2,3,4) to do the corresponding way(add/sub snakeSpeed to cur\_snakeObjPosX/cur\_snakeObjPosY or stop(jump to L\_Quit)).

Use a counter, if the procedure updateSnake runs snakeLifeCycle times, then the program will print the sphere (move cur\_snakeObjPosX to [OFFSET objPosX + numObjects\*4], which numObjects means how much spheres are on the screen currently. Y-coordinate is as the same way. To judge whether updateSnake runs snakeLifeCycle times, I increase the counter every updateSnake, use compare when the counter is equal to the snakeLifeCycle, it prints and make the counter zero again.

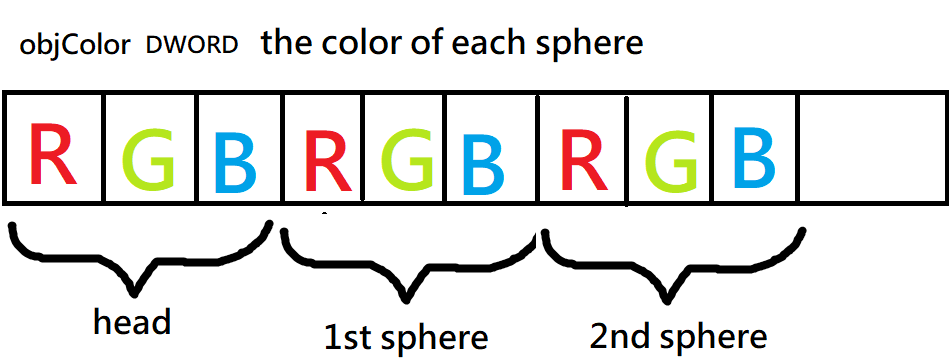
Third, to colorize the spheres, I need to complete asm\_GetObjectColor.

asm\_GetObjectColor:

Move [OFFSET objColor + objID\*3\*4] to r.

Move [OFFSET objColor + objID\*3\*4 + 4] to g.

Move [OFFSET objColor + objID\*3\*4 + 8] to b.



To randomize the color, just random each column start from [OFFSET objColor + 3\*4] to the end of the array.

To make the color rainbow, just make each sphere start from [OFFSET objColor + 3\*4] in order red(255,0,0), orange(255,128,0), yellow(255,255,0), green(0,255,0), blue(0,127,255), indigo(0,0,255), violet(139,0,255), and loop.

Fourth, to set the target with the mouse left button, and make the snake move to the target, I need to complete asm\_GetMouseXY, asm\_GetTargetXY.

asm\_GetMouseXY:

Move mouseX to [out\_mouseX]. Move mouseY to [out\_mouseY].

asm\_GetTargetXY:

Move target\_X to [out\_mouseX]. Move target\_Y to [out\_mouseY].

Move eax, flg\_target. (return)

flg\_target = 1 means there is a target for the snake head to go. 0 means no.

To make the snake head go to the target, I first use two variables x\_difference, y\_difference, which means the distance of the target and the cur\_snakeObjPos.

x\_difference = abs (target\_x – cur\_snakeObjPosX).

y\_difference = abs (target\_y – cur\_snakeObjPosY).

First, compare x\_difference and y\_difference and decide to move first.

Second, if either x\_difference or y\_difference is smaller than snakeSpeed, it means the head has reached the target, and make the flg\_target zero.

The program is written in the procedure GotoTarget.

Other procedures:

asm\_GetNumOfObjects:

Move maxNumObjects to eax (return maxNumObjects).

initSnake:

Move 1 to flg\_growing (it means it is growing)

Make the head red. objColor[0] = 255 objColor[1] = 0 objColor[2] = 0

asm\_HandleKey:

Handle the input and do the right thing according to the input.

updateSnake:

If the current snake position is out of the bound, reverse the moving direction.

According the snake moving direction to add or subtract the current snake position by snake speed.

If flg\_growing is equal to 0, don’t print, else print.

Use the counter and current snake position to move and print (as above-mentioned).

**[20%] Discussion/Experiments [ at least 200 words]**

**WORD COUNT:\_\_\_\_\_\_\_231\_\_\_\_\_\_ [ Must be filled or zero score]**

I think the hardest part is at the beginning for showing the background image and moving the snake, because there are many variables and procedures which are unknown to me to make a bridge between the C++ and assembly.

At first, I completed a few procedures (asm\_SetImageInfo and asm\_GetImageColour, asm\_GetImageDimension) about the background image. Unfortunately, if there was a small mistake in one of them, then the background won’t appear correctly. It made the debug more and more difficult.

Then, to move the snake with procedures (asm\_ComputeObjPositionX, asm\_ComputeObjPositionY , and updateSnake). At the beginning I misunderstood the asm\_ComputeObjPositionX. Later I encountered the same problem as before. Debugging cost me a big part of the time at the beginning in doing homework 03.

After showing the background image and moving the snake successfully, I first found what is in the objColor , that is RGBRGBRGB……. It became easier and easier, making the color random or rainbow just cost me little time.

The last part cost me some time was to set target and make the snake move to the target. First compare the x and y of the distance between target and snake head, and move in the correct way. However, I was thinking how to determine the snake had reached the target. I finally came up with an idea, that is, the distance is smaller than the speed of the snake.

**[10%] Conclusion [ at least 100 words]**

**WORD COUNT:\_\_\_\_\_\_\_130\_\_\_\_\_\_ [ Must be filled or zero score]**

The homework 03 uses lots of things in chapter 9, such as stosd, movsd, and lodsd. Before doing the homework 03, I was unfamiliar with them. Now I can use them in the correct way anytime. And this is really a cool project, it combines OpenGL and assembly, it’s hard to imagine.

The homework also enhances what I have learnt, especially the usage of ebx, esi, edi.

And other skills such as USES. The homework is well-module, readable, and easy to modify.

I haven’t touch the OpenGL before, that’s really cool. I seldom do such an interesting homework, the last time I do is to do the website in the course database.

In the final project homework 04, I think there will also be interesting and challenging on the road.