## C Program Report: The plane war

## Abstract

a. Problem Statement:

The classic “airplane fight game” can relaxe people and exercise their reflexes. After referring to the classic game rules and mechanisms, we tried to use the C language program to achieve a simple aircraft game. And we plan to insert some personalized element in the game, such as music, pictures, etc., which let the players experience a little fun from this game.

After seeing some existing airplane games on the computer, video games and mobile games, the interface of the game is really significant. In order to make good interface in the game, we may find some useful head files and libraries. Finally, the game is presented in 2D form for convenience and invoking the known knowledge. Thinking that 2D shooting games may require the use of photo resources, drawing tools, and so on, so there is a need to process the images like using The Photoshop Software as well.

To sum up, our project is about the simple 2D shooting game, which may include necessary function and beautiful interface.

b. The brief goal:

our project is about the simple 2D shooting game, which may

include necessary function and beautiful interface.

The game can run fluently, and there is a scoring system, also, the game will not be endless, and there are conditions for ending the game.

There are many pictures and music that you can see or hear in the game.

## Introduction

### The mainly quality of the project:

The project is about the simple 2D shooting game, which will include necessary function and beautiful interface. By making the interface, we have used many special functions from “EasyX Library”. This graphical library

### The mainly technology:



Compiling environment: Visual Studio 2019 Preview

The programming language: C/C++ (The special head file but it is mainly the program about the C)

The “EasyX”:

“EasyX” is for C++ graphics library, can help C++ language beginners quickly get started graphics and game programming.

For example, we can use VC + EasyX very quickly with geometric graphics to draw a house, or a mobile car, and you can write tetris, snake, black and white chess and other small games, you can practice graphics of various algorithms, and so on.



The way to install the EasyX. It is easy to find the EasyX library from the Internet. And we can install them in our computers easily.

 While programming, we also use some basic ways to revise the figure and make the figure suitable in the interface of the game.

### Advantages of the project

We learned about some basic but useful functions from the EasyX and put them into the use in the program.

We found a way to use transparent graphics to embed the image into the background so that can leave the good experience on the player.

The interface in the program is simple and practical for people to play on.

## **Group Division**

|  |  |  |
| --- | --- | --- |
| Name | student ID | College |
|  |  |  |
|  |  |  |
|  |  |  |

### Person1

Use special functions to make the interface in the game and make the special boss levels in this game. Find related pictures and musical resources for the game. Achieve the “how to control my plane ” function for the game

### Person2

Complete the function “how to create the bullet “ and “how to control the speed of planes”. Also, he fixed several key bugs that were serious to this game and optimized the program.

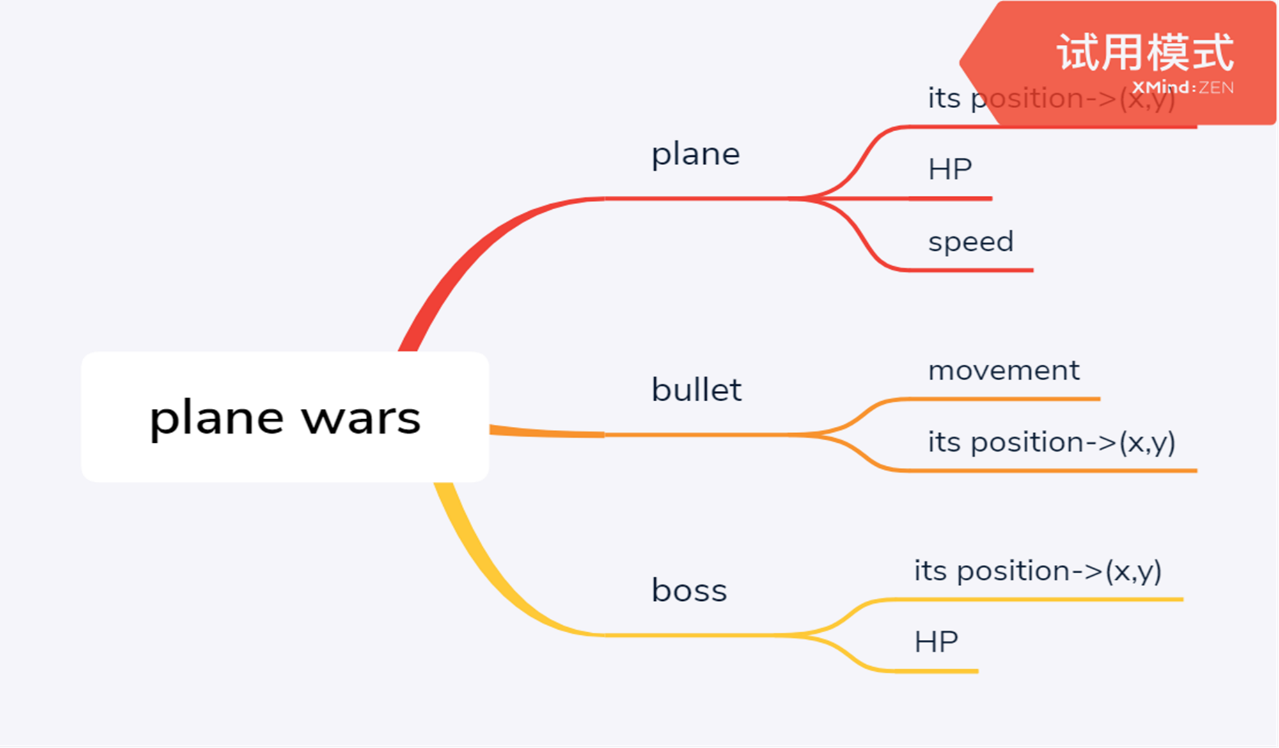
### Person3

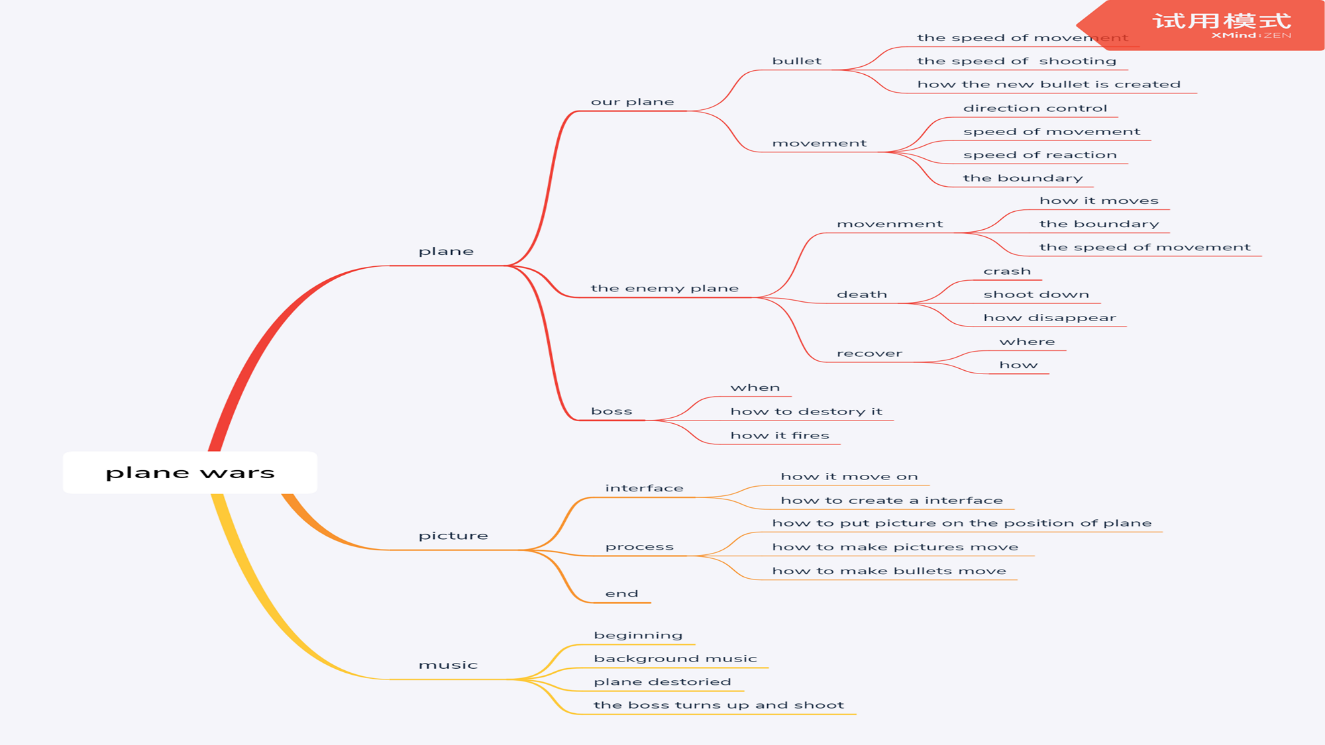
Complete the function ”how to make the enemy plane move” and “how to make the enemy plane update”

## Analysis

we make three kinds of analysis. The first is conceptual analysis, we come up with three main components and relative elements. The plane, bullet, and boss. Then we go to more specific, the demand analysis, which is what we should do next. We divide games into three parts, each parts has different requests. For the plane, especially our plane, we should figure out how it fire(the bullet speed , bullet produced…..) For the enemy plane, we should think about how they can be destroyed and update in the new position.

a. The first one is the plane including “my plane” and “enemy plane”





b. The second one is about the ”boss”. And there are two bosses in this game so that there are two levels during the game.

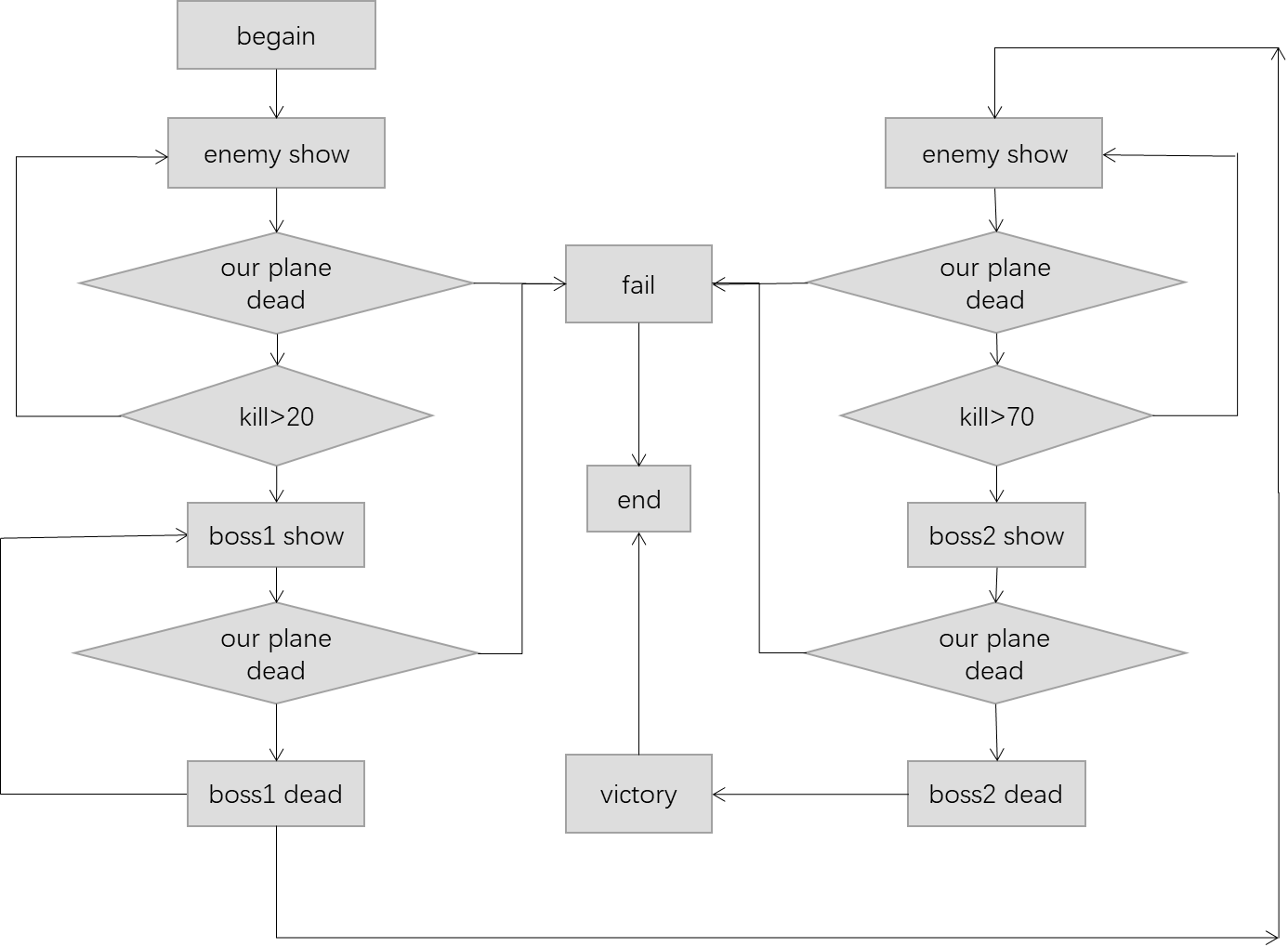
c. The third one is that the interface will be transferred in the different period and condition.

In other words, these requirement comes from the true plane war game, but we just do some simplifying about the elements in the true game

## Design

In this part, we will divide it into three parts(the interface, the boss, and the plane) and the flow chart to introduce.

### The flow chart



This flow chart is to show how the game is running. The important clue is the “kill” (score). That means when you get different scores, you will get the game into the different position. For example, if your kill reaches to 20, the first boss will appear and attack you.

### The interface

We think about what the gaming interface should be like.

It should include:

a. the interface of loading and starting the game

b. the window to operate your game

c. the information bar

d. the interface of ending the game

The interface of loading and starting the game is to imitate the true game. When you open a game, you will find the loading interface and then you can see the instruction and press some keys to begin the game.

The classical game about the plane war has the special operating interface, which means your plane and enemy plane will not fly out side of this interface. On both sides of this interface are the information bar. The information bar can provide players with the important information and the scores.

There are two ways to end this game, including the winning and losing. In different ways to end this game, the game will play different the interface including the relevant music.

### The plane

This part includes “my plane” and “the enemy plane”.

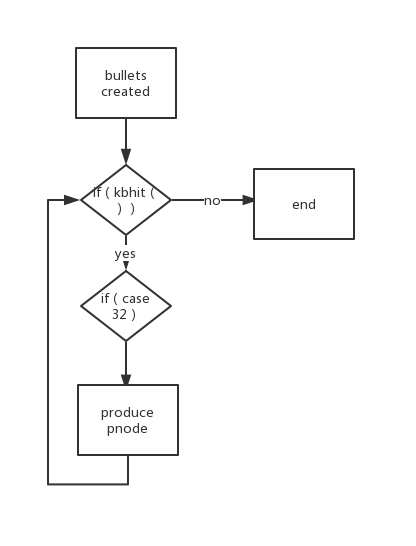
1. my plane:

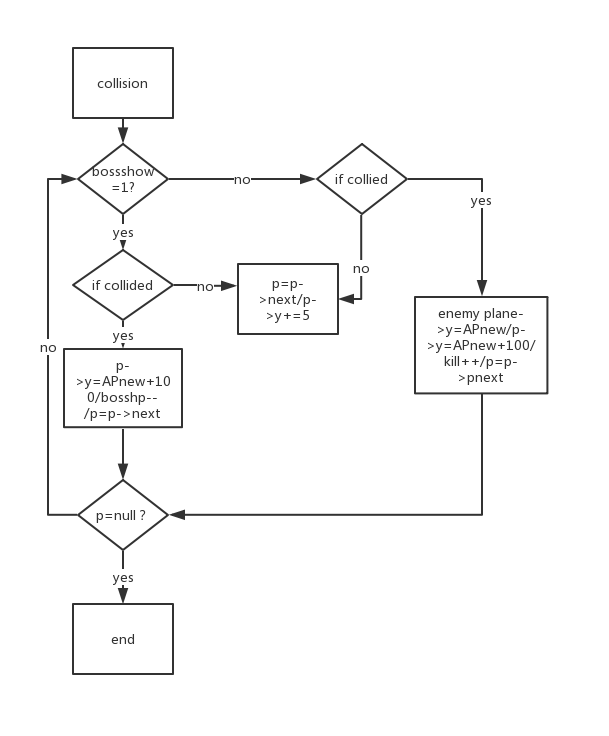
after thinking, this part should include:

how to control “my plane”

how to make the bullet

how to destroy the enemy plane

how to be attacked by the enemy plane

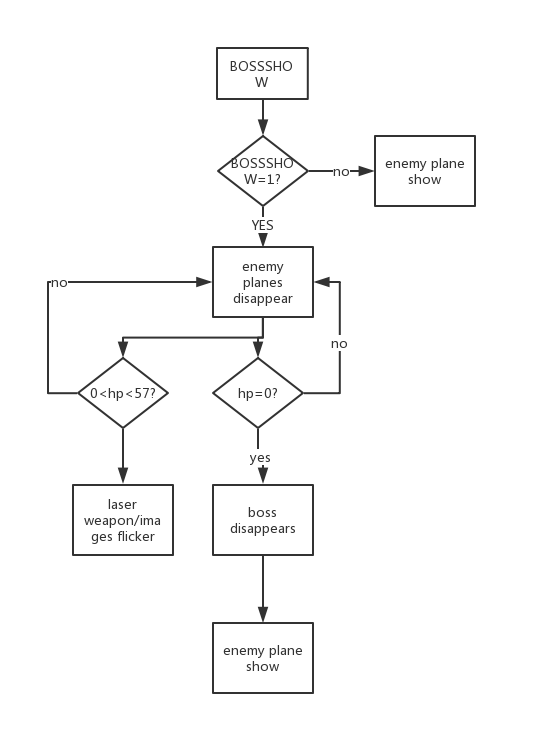


After we put the picture on the dynamic coordinates, we can realize the flight of the aircraft, so that the planes can move on the operating screen.

We decided to use special buttons to trigger the bullet. Also, the bullet is enough almost.

The bullets will create and collide with the enemy planes and make different effect.

### The boss

There are two kinds of boss function. When your scores reach to 30, the first boss will appear, and when your scores reach to 70, the second boss will appear. 

After your plane destroy the boss, you will receive lots of scores.

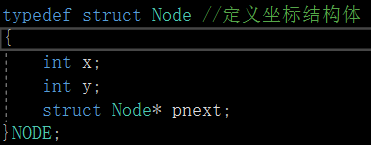
All in all, you have to destroy the small enemy plane to get enough scores and then make the boss appear. You only have to use the bullet to destroy the enemy plane, and the game will continue.

## Implementation

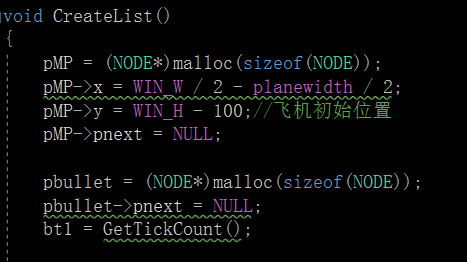
In this part——implementation, we will share some details about how to implement some function in this game.

### Coordinate structure

Define structure



Using the structure (example)



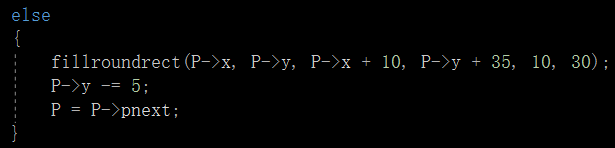
For example, pMP is defined as a global variable in the code, and is not placed with subsequent member assignments. Defining the pointer type and using the heap for pMP allocation instead of directly defining it in the stack are both meant to be in the same format as defining bullets because bullets need to use linked lists.

### Create List

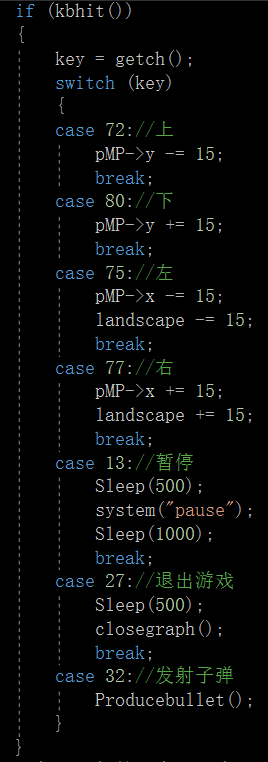
Create a linked list to control the creation of bullets and the movement of bullets. Linked lists are chosen because they are easier to insert and modify than arrays.



Starting with the first bullet, subtract a number from its ordinate value, which together with Sleep determines the speed of the bullet, and then point P transfer from the first bullet to the second bullet until P is empty. Because the pointer field of the last bullet is empty. The first bullet here is the latest bullet, and the last bullet is the meaning of the first bullet.



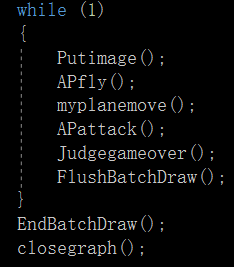
### Control the direction



This khbit is the library function in the conio.h header file that reads from the input buffer. If we hit the up button, we just change the plane's y-coordinate, and the degree of change determines how much the plane presses forward, along with the Sleep function in the while loop (which determines how many times you can press per unit of time) to determine the speed of the plane (the distance per unit of time).

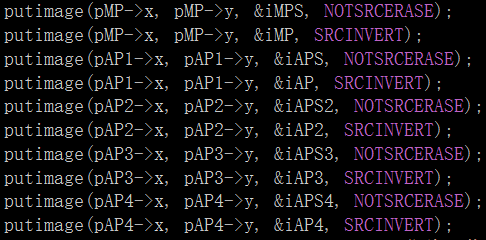
### The drawing system

we use EasyX library drawing system. of course, we can also use WIN32API drawing system.



### Aircraft and bullet pictures following the movement

We can post the picture of the plane to the moving position of the plane, instead of a fixed position, so as long as the coordinate value of the plane was changed, the picture naturally followed the relevant position.



Tips:

Use Sprite and mask to create a transparent effect.

 Sprite picture: this is the image you want to draw. You want some areas of the image to be transparent.

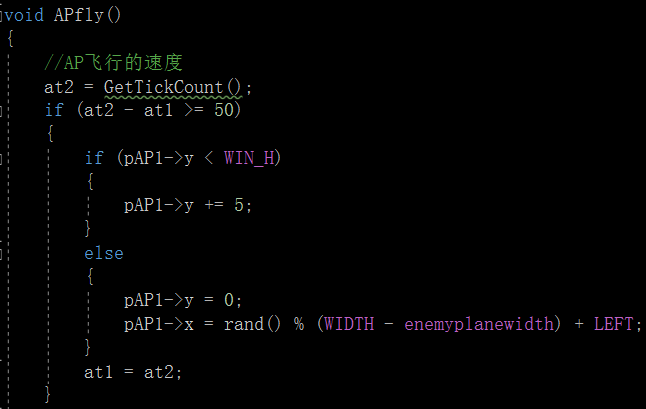
 Mask Picture: the area to be displayed is white and the transparent area is black.

putimage(x, y, Mask, NOTSRCERASE);

putimage(x, y, Sprite, SRCINVERT);

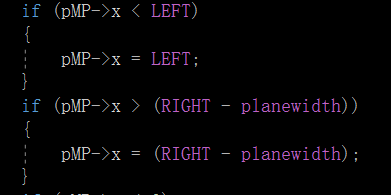
### Speed and time difference

Here I use the time difference to control the speed of the aircraft. Every time, check to see if the time difference is correct. If it is correct, increase the bullet and assign the value of at2 to at1 to update the time difference.



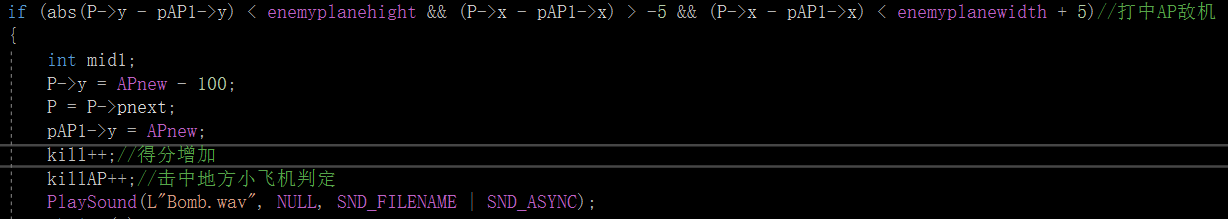
### My plane has a flight boundary

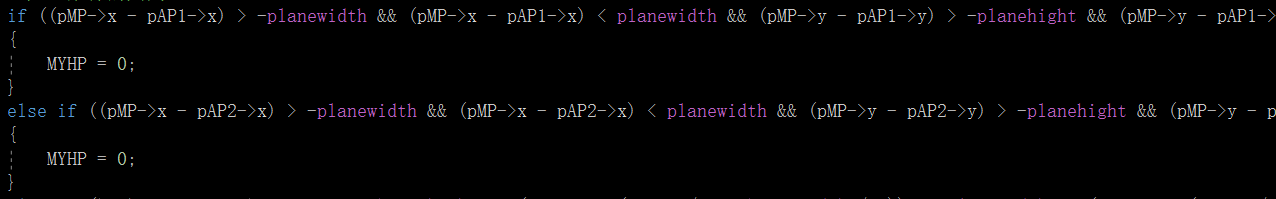
This is an example of the left boundary. We use the if function to keep the plane from going to the negative coordinates. Also, use the if function to keep other planes in the operating screen.



### The judgement for destroying enemy plane and collision

A kill is the special difference between the coordinates of the bullet and the coordinates of the plane within a certain range. Also, the collision is the same reason.





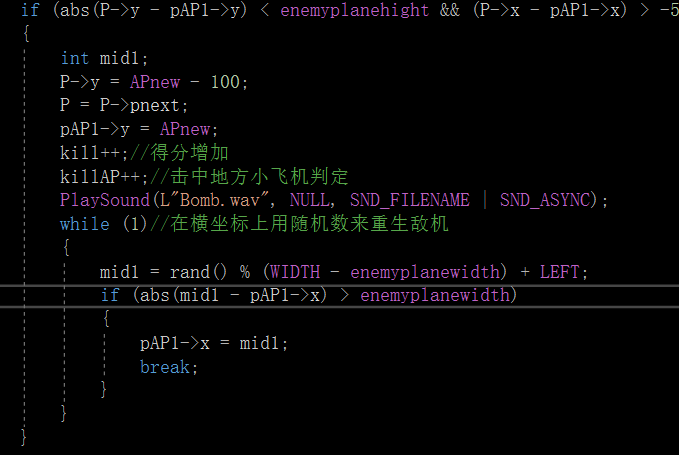
Since we are using bullet coordinates, I put the decision part in the p-incrementing list. The collider decided to leave the bullet, so I could put it outside the p-increment list.

This requires understanding the specific meaning of window coordinates

### Coordinates of enemy regeneration

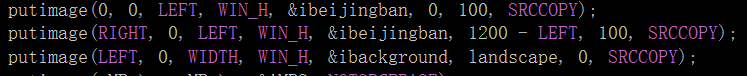
The enemy planes were reborn in two ways: first, they were killed by our planes, and second, they flew out of the lower boundary.

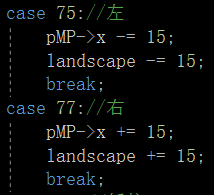
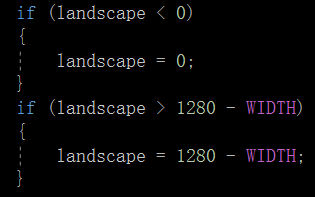
First, the enemy plane was destroyed:



### the effect of moving background image

Using the overloading function of putimage, the key point is the “landscape” variable, which determines the selection area from where the image starts, then we press left and right in the khbit section to change the value of landscape, and finally specify its boundary.

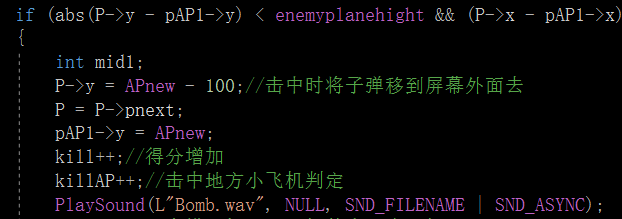


Similarly, the opening animation of the first image from the middle to open the same way.

### The bullet disappeared when it hit the enemy plane

When the decision is passed, just move the ordinate of the current bullet out of the screen to create the effect that the bullet disappears.



### Bosses’ behavior

We use the b1show and b2show integer variable to distinguish between boss combat and normal enemy combat when the boss appears and when the bullet increments.

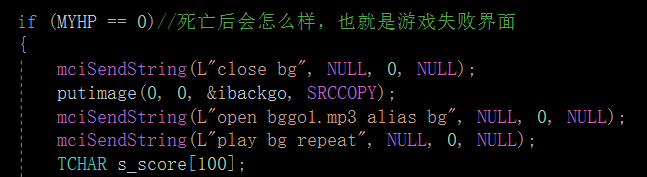
I initialize the value of b1show and b2show to be 0, and assign the value of b1show to be 1 when the kill reaches to 20, and assign the value of b2show to be 1 when the kill reaches to be 70. And then, assign the value of b1show or b2show to be 0 when the bosses are killed.

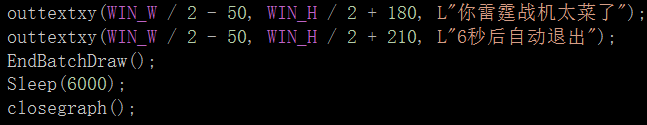


### Lose the game

Define MYHP to make the end screen appear.

When colliding or being hit, MYHP of my plane is assigned to 0. The If judgment shows the end of game screen.



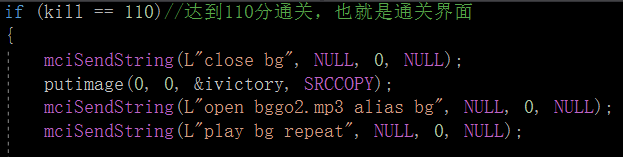


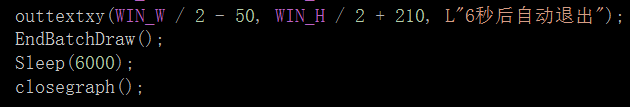
The losing interface lasts for 6 seconds and then the game will be closed.

### Win the game

Define kill(scores) to make the end screen appear.

When the player gets 110 kill, the player will win this game. And the If judgment shows the end of game screen.





The winning interface lasts for 6 seconds and then the game will be closed.

## Testing and Debugging

The main focus of this phase is on ***verifying*** whether the written code meets the problem requirements under known constraints.

1.Basically, the written code, after being compiled, is tested by inputting several data, including corner cases ones (e.g., negative or even numbers), and check if the desired output is generated.

2.Typically, tests should be performed when concluding each function and modules, as well as when integrating functions and modules in the final program.

3.This form should appear in this phases to fully test your project

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Real Result |
| Press space | Bullets created and ahead of plane | Bullets created and ahead of plane |
| Press up /down/ right / left key | Plane move up/down/right/left | Plane move up/down/right/left |
| Press enter key | Game pauses | Game pauses |
| Kill>20 | Boss1 shows | Boss1 shows |
| Kill>70 | Boss2 shows | Boss2 shows |
| Boss hit by bullets | Bosshp--/bullet disappears | Bosshp--/bullet disappears |
| 0<Bosshp<47 | Laser weapon  Images flicker | Laser weapon  Images flicker |
| Bosshp<0 | Boss disappears/play crash music | Boss disappears/play crash music |
| Enemy plane crashed by bullets | Enemy plane and Bullet disappear/play crash music/enemy plane renewed/kill  ++ | Enemy plane and Bullet disappear/play crash music/enemy plane renewed/kill++ |
| Boss2 crashed | Play winning scenes/game ends | Play winning scenes/game ends |
| Plane hp=0 | Play failure scenes | Play failure scenes |
| Press key quickly | React at a constant speed | React at a constant speed |
| Plane and enemy plane collide | Both disappear/game ends/play failure scenes | Both disappear/game ends/play failure scenes |

## Result&Conclusion

Although we just complete our project, there are still some trouble and some great ideas that cannot be achieved. It is some pity that we just achieve some basic ideas and function. This project let me understand some of the important shooting game operating rules. For instance, shooting judgement is related to relationship between object models’ coordinates. And the parameter changes are used to switch the game interface. We also learned from other classmates and the Internet to realize some good functions like the keyboard function. Also, the most important of these is the EasyX library functions, which is really helpful for making the interface. It is quite interesting to implement the running of game characters mainly through coordinate addresses. We programmed this game while learning some useful functions and ideas from other people. And we even went to learn something about photoshop, when we met the problem to make the background transparent in the gaming screen. Doing a project is a good way to learn, but from this project we also learned that we need some basis of data structure. We will learn some good algorithms in the future in order to meet the project content better. That’s all, and thanks for your reading.