**C Language Report**

**STICK HERO**

**BY:**

**张越YueZhang**

**牟唯佳 WeijiaMu**

**汪博宇 BoyuWang**

**王上毓 ShangyuWang**

****

**·Abstract**

The project objective is aimed to create a game that has similarities to the one we had played, but with our own ideas and innovation.

We all come to an agreement that clearly understanding that making a complete C language project successfully and cooperating with teammates well is more vital than just simply finish it as a class assignment. It is obvious that we have the confidence to achieve our goal and meet all the challenges arise in the process.

A complete problem life cycle has been done during our program designing.

We began with raising the idea of creating such a game. Stick hero(Stick hero is a game produced by ertan akgul in 2011. It is an English application on cell phone.) was chosen to be the original model for our project after several discussions because the succinct operating interface and the basic idea of creation are significantly attractive to all the group members. Then we did the analysis of the feasibility and divided the program in to five relatively independent periods that we were going to achieve. Next step, we did the previous design of the program and finished the work division. Each one of us took the responsibility of several main functions so that we could do our own job more efficiently. Simultaneously, we did the document referring and function searching. It was obvious that we were all green hands in this field, but we tried to use our book and the Internet as tools to assist us realize our goal. There is no doubt that the implement period is the most difficult part. We had experienced several hard occasions and got lost in some situations. Fortunately, we found the solutions. As for debugging, we tested our code and corrected those points to make sure that the game could work fluently. Coming to the end of our work, we tried to make improvements to perfect our program.

**1.Introduction/Problem Statement**

The main function that our program can realize is trying to use C language to create an interface that allows operator to play the “stick hero” game. Players manipulate the 2D game character (hero)by touching particular keys(space key & enter key). The character will go right forward, there are randomly generated obstacles on the way, the character goes forward using the stick which is able to lengthen while pressing space key. The stick will fall by pressing enter key. By determining whether the edge of stick falls within the range of the obstacle, the game either continues or ends. If the stick is shorter than the left border of the pillar or longer than the right border of the pillar, then game over.

That is the basic mode for operating the game. What’s more, eating cherries is an advanced mode that will also be realized in our game if time is abundant, which means players can use another to keys to control the character to go up and down to get cherries as an extra bonus in the game. The hero can get bonus through passing every levels and eating cherries. In addition, the beautification of interface like to add pictures and background music are also needed to be brought out.

The technique that we used is based on what we have learnt in class and through reading the C kiss by ourselves. Besides, we searched for several resources like main functions on the Internet.

Comparing to other similar project，there are plenty of unique advantages that our application contains. First of all, we believe that our app fits the majority’s taste. Our interface is more succinct without any ads or complex pictures and patterns. Players are always tired of dealing with glaring screens and our interface can let them directly get to the information they need. Light music is chosen as the background music to make players release their pressure and enjoy the atmosphere of taking adventure in our game and have fun, which is their original aim of playing. Secondly, there is no difficulty for anyone to get started because rules are easily to be understood. Complex operating is not needed to be learnt during playing and all they need is to focus on the pressing time and take care of the structures that delivers randomly. It is suitable to plays among all ages. Thirdly, challenges are not missing. We can control the width of structures to make more troubles for players to pass successful, which means if they wanted to get higher scores, more practice is required. Individuals can easily be attracted by such challenges. In addition, extra bonus rises the degree of difficulty and make it more competitive. The perspectives on which players need to focus increased. They need to assess the length、pressing time and extra bonus at the same time, which dramatically increase the risk of failure. Also, players can keep refreshing the record to get a pleasant game experience.

The problems that we have solved is the basic playing mode. Pillars of different length will be generated randomly in different place. By touching space key to produce the stick, and it will become the "bridge" between the pillars, making the hero pass. By controlling time that you touch the enter key, the stick will have different length (the character waits at the edge of the post to wait for the player to operate). Improving our interface、inserting background music & BGM when the hero dies、adding calculating-grade function and solving the flicker problem when playing are all done through our hard work. We are still trying to add the interesting cherry-eating mode.

**2.Group Division**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student Number | Major | Work Proportion |
| 牟唯佳 | 12170905 | Chemistry | 25% |
| 张越 | 12170901 | Chemistry | 25% |
| 汪博宇 | 12170903 | Chemistry | 25% |
| 王上毓 | 13170707 | Biology | 25% |

1. 牟唯佳 Weijia Mu

Void presskey()

Void login()

These are the main functions that Lily has been working with. She draws a circle track when the stick falls & interface. Besides, she takes the responsibility of debugging and testing.

1. 张越 Yue Zhang

Void move()

Void fall()

These are the main functions that Alice has been working with. She took the responsibility of hero moving and did the freely falling when the hero dies. Besides, she also takes the responsibility of debugging and testing.

1. 汪博宇 Boyu Wang

Int max\_score()

Void change\_score()

Void game()

Void game over()

Vice T&D

These are the functions that he has used. He was working at the extension of the stick part.

1. 王上毓 Shangyu Wang

Int main()

Void UI()

OB ob\_rnd()

Void draw(OB)

These are the main functions that Young has used through his designing. He was taking the generation of random columns part.

**3.Analysis**

The algorithm of the current online version “Stick Hero” is extremely legible to interpret. As most of the games commonly have, the game has a login interface, and then after multiple game functions using API of the C language, the game continues smoothly without a hitch. After judging certain conditions, the game may come to an end. The most important algorithm or function of the “Stick Hero” can be divided as the following 8 parts which are as follows.

3.1 Login Interface

When compiling the codes, an initial interface should be shown. The interface shall convey the message of the name of the game which is STICK HERO. And as our conceives for those operators who are not well aware of the game we are going to print the instructions as “Playing strategy: Press space to lengthen the stick; Press enter to fall the stick.”

P3.1 Initial Interface

There should also be an assigned region, in this case, the red circle Only when mouse is clicked within this region, the game can start.

We also have to mploy a function to set the background which is to put an image on the scene.

3.2 Obstacle creating

P3.2Obstacles Creating

In this part, we need to put two obstacles within the assigned region, which is that the obstacles should be at the same horizontal plane.

The distance between two obstacles and the length of the obstacle are randomly oriented. The obstacles should be filled with black in these cases. Further more the coordinates of the left and right edge of the obstacles should be recorded for the further judging part.



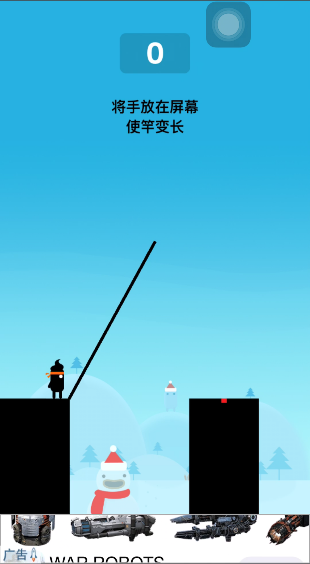
P3.3 Stick lengthening

3.3 Stick lengthen

In this case, the constraint of this case is that space is the only valid input, when inputting the invalid instruct, an error shall be reported.

The stick should be elongated at a constant speed and this width of the stick should be settled. Actually, the stick itself is an obstacle that has a very thin width and filled with black.

What we need to do is that we need to employ a function to simulate the process of stick elongating when pressing space.

3.4 Stick Falling

After the stick elongated to an assigned length, we need to let the stick fall, of course in the shape of quadrant.

Because of after all the researches of documents we have done, we have found no API functions in C language that allow us to implement our idea that lengthen the stick while inputting space and when not inputting space, the stick automatically falls down. Sadly, it can’t be realized.

First of all, we need to employ a function to simulate the dynamic process of stick falling when pressing enter. In doing so, we planned to employ an equation of a circle and retain only a quarter of it to describe the edge of the stick which is theoretically moving as a circle.

P3.4 Stick Falling

After a series of series of wonderful operations which we will talk about in the implementation part, we need to describe the moving formula of the stick. These are the trickiest part that we managed to do in our C program.

The other important thing is that we need to make the screen not flashing and make it look like the stick falling, not the dots moving.

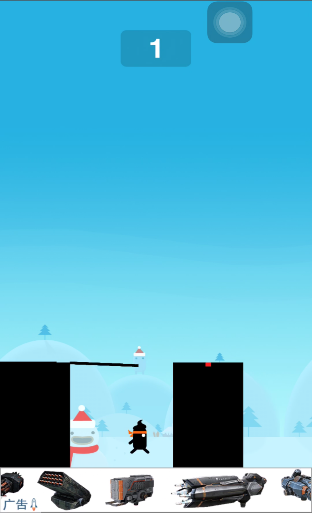
P3.5 Hero Moving

In this part, the constraint is that the hero must move to the edge of the stick horizontally with a constant speed, it starts at the right side of the first obstacle and end at the left side of the second obstacle. It is of course using the coordinates to be realized.

P3.5 Hero Moving

We also need to employ certain functions to make the screen not flashing and make it seems like the process of hero moving.

Furthermore the hero must move automatically after the stick falling process and after the ending of the hero moving process it comes to next scene with two newly oriented obstacles. The first new obstacle shall inherit the same length of the second old obstacle.



3.6Hero Falling

P3.6Hero Falling

The constraint of this part is that only when after judging the edge of the stick didn’t fall within the region of the second obstacle will the hero do the free fall, this is an accelerated process.

We need to employ a function to judge whether the hero passed the current custom pass or not, therefor determine whether the game will proceed or not. The detailed design shall be shown in the implement part.

Also we need to make the screen not flashing at all.

3.7 Game-over Interface

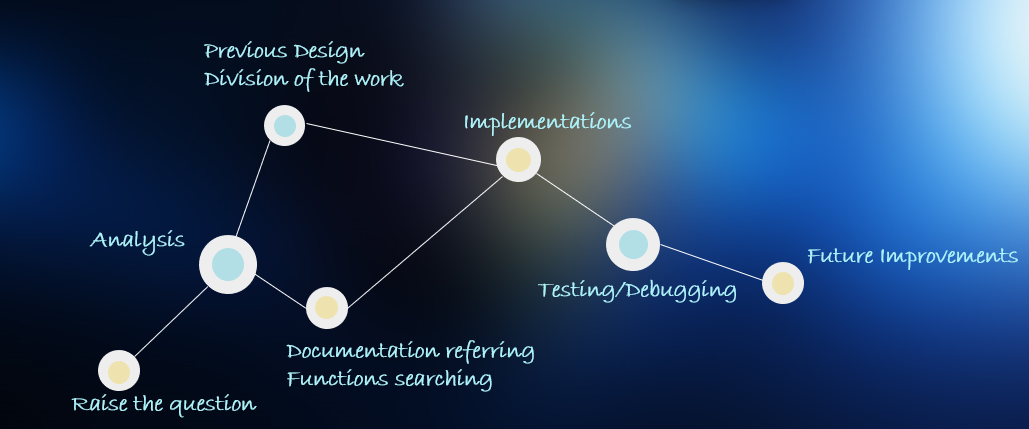
P3.7 Game-over Interface

In this section, after judging the hero didn’t pass the condition, the game automatically goes to the Game-Over interface.

The final score of the game that is just played will be shown on the center of it as well as the highest score that has been recorded. The image also has an assigned region that you can either choose to play again the game with a single mouse click or end the game process immediately.

As is shown above, these are merely the core functions of this game, there are severe other designs we have made in order to make the game operable and playable. For instance, we added background music when the game processes. It is the “Time Runner” by Keega. We also added the game-over music when judging the game ends. We also need to print the image of the hero itself to make it actually look like a hero and find the appropriate background pictures of the game.

**4.Design**



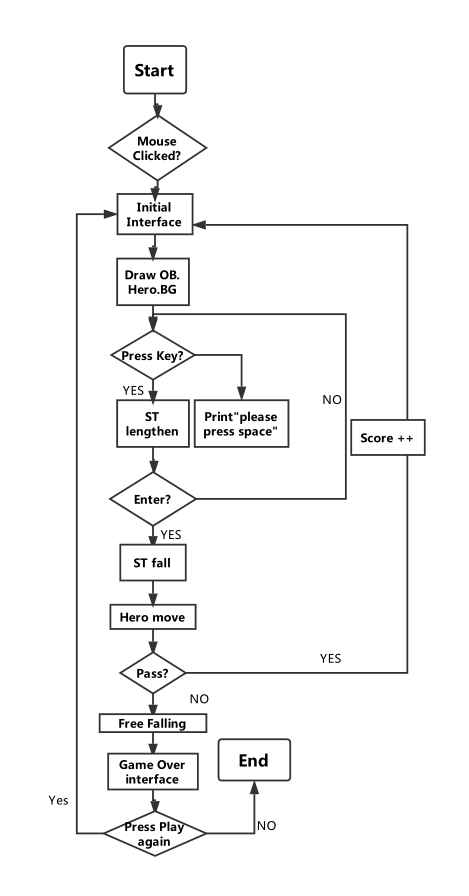
P4.1 Overall Design

As is shown above, it is our Overall Design.

The initial proposal of the program is raised by a group member. After the group discussion, we found the game quite intriguing and amusing. We searched the relative documents, the prospect of imitating this online game seems rather promising. Because the algorithm is quite clear to be seen.

We first raised the question. And then we did the analysis over the whole topic, which made us determined to devote to the program.

Later, we did previous design, division of work, documentation referring and functions searching part simultaneously. And then we each wrote the assigned codes as our group work division, we helped each other through tough times, during the period when someone’s computer accidentally broke down or a couple of our group members were seriously ill.



P4.2 Flow chart of the whole game process

As shown above is the flow chart of our whole program.

After compiling the codes, the game started, if the assigned region is clicked, an initial interface is shown.

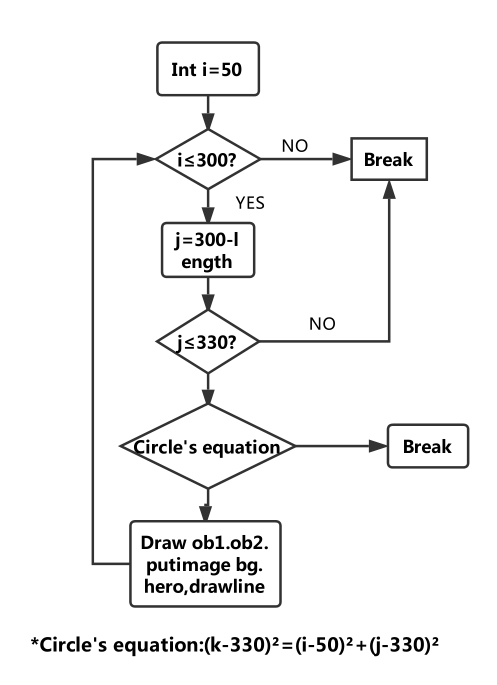
Then, the obstacles will be randomly generated as we designed before, the hero, the background shall also be drawn.

If the assigned key, in this case the space key is pressed, the stick shall elongate, if it is not pressed, an error :”Please press space” is reported.

Then, if enter is pressed, the stick shall fall down according to the equation of the quadrant. If enter is not pressed, the process will go back to the Press Key part.

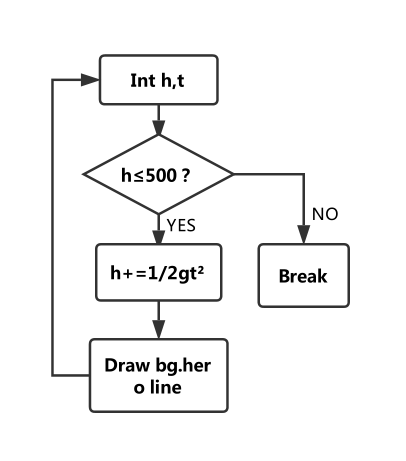
After the stick falling down, the hero shall move on the surface of the stick, if the judging function verdict that the edge of the stick has fallen within the region of the obstacles, the score will be added by 1 and recorded. Later it shall come back to the initial interface of the game. However, if the stick doesn’t fall within the length of the obstacle, the hero will do free falling at the edge of the stick. Then, a game-over interface will be shown.

If the operator pressed the region that says play again, the process will go back to the initial interface while the score is eliminated. If not, the game comes to an end.

“i” is inputted for the x-coordinate of the center of the circle.

P4.3 Flow Chart of the stick falling

“j” is the y-coordinate of the center of the circle.

As is shown above, if i ≤300，the value of ”300-length”will be given to j, then if j≤330 (330 is the maximum height of the program’s scene), the function will imitate the stick falling down process as the equation. Then, obstacles will be drawn and the background image shall be printed, etc. This part of the function goes back to the first state, which is judging the value of i.

P4.4 Flow Chart of the hero free falling process.

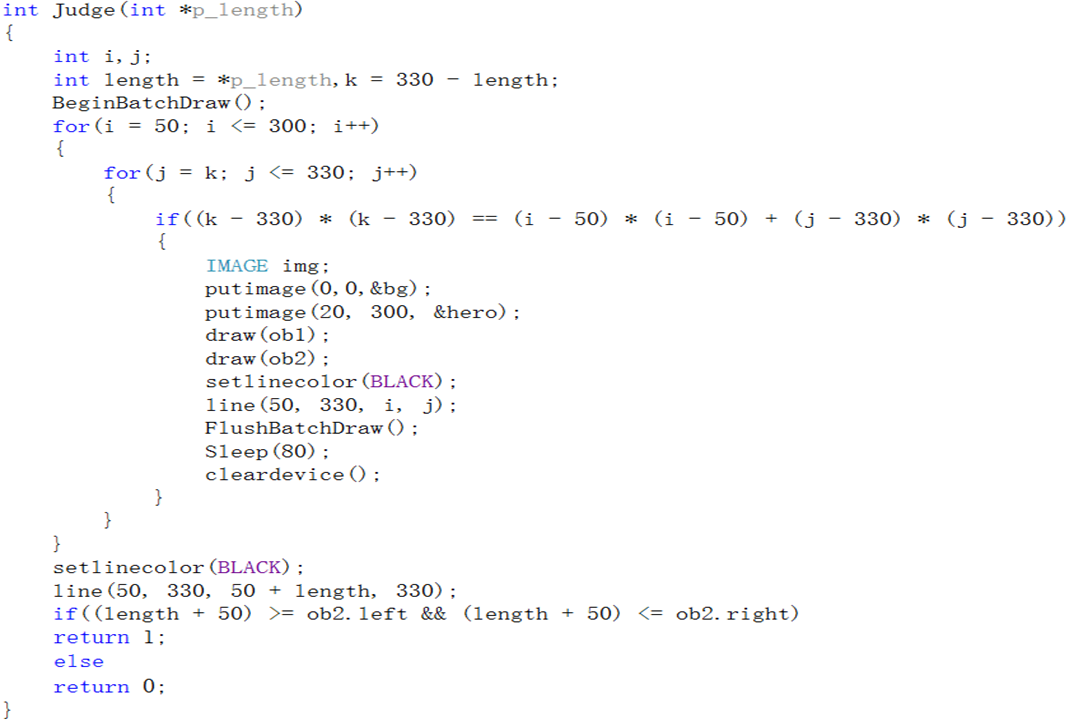
First, we input h and t, which respectively represent the height and time of the hero falling.

If h≤500, h+=0.5\*g\*t². Then, the drawing process shall proceed, and later go back to the part that inputting variables

If not, it will come to a break.

**5.Implementation**

During the implementation part, we put our design and analysis into real codes with the help of all the information we have obtained about different header fills and different function names. But most importantly, we used the techniques we have learned from the C kiss to think about our program. Since the code is way too long, we would display 3 most important part of the code to show our methodology. However, since the codes have been clearly displayed, further illustration would seem tedious and unnecessary. We would label the important parts of the codes, therefore benefit the readers.



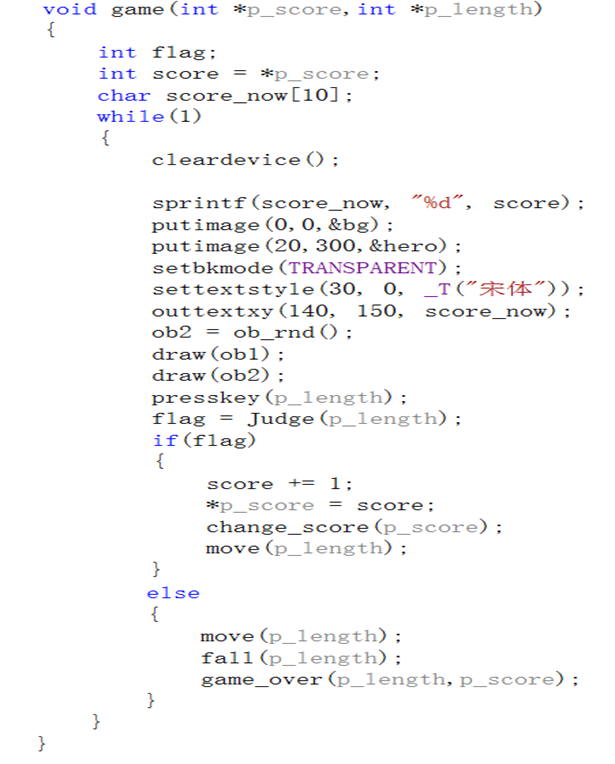
P5.1 The code of the stick falling process

--Use batch drawing to prevent screen flashing.

--Use a function to describe the drawing process

--Simulating the process of the falling stick

--Judging whether edge of stick falls within the lengthes of the obstacles.

****

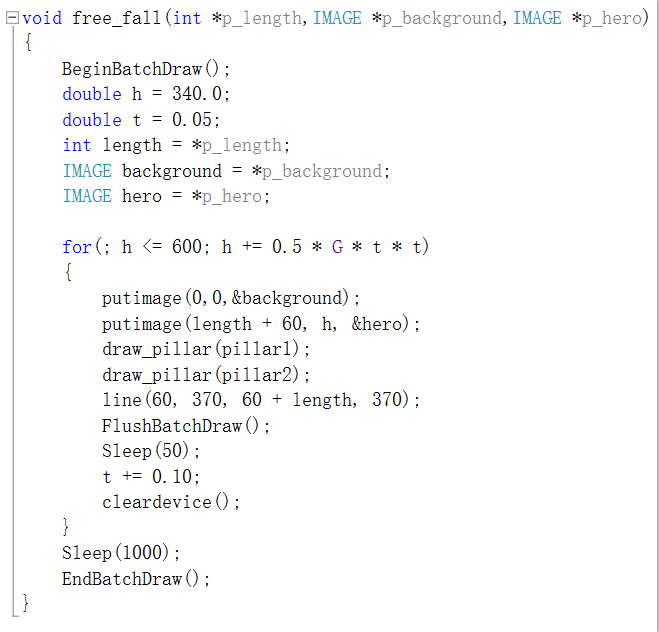
P5.2 The code of the Game function.

--Save current score.

--Put image in specified location.

--Drawing and putting the images of the obstacles.

--Judging whether continue or game over.



P 5.3 The code of free falling.

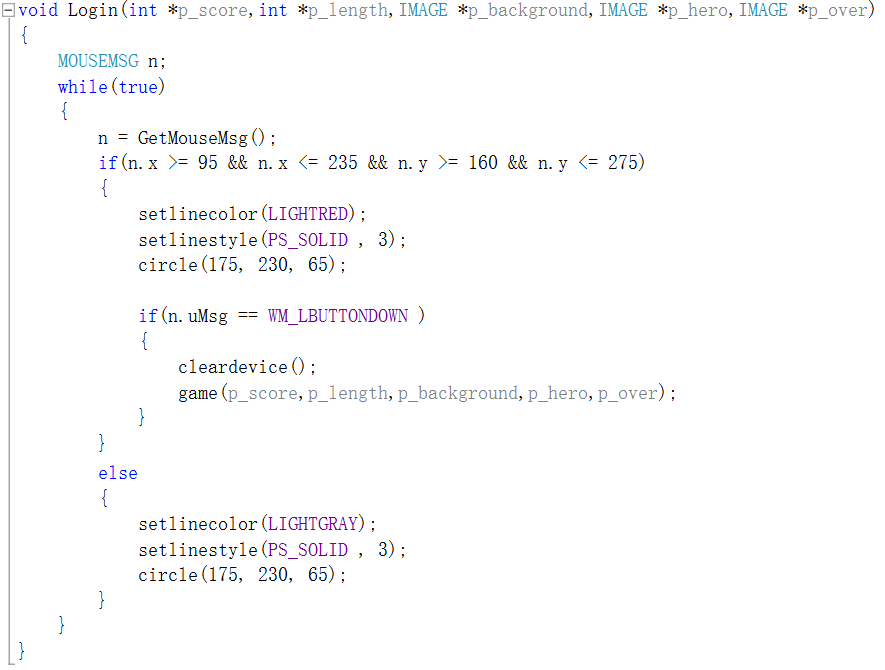
--Imitate the process of the falling stick.

--Put the image of the background &hero &pillars.

--Use Flash Batch Draw to draw line.

There are the main functions of our program.

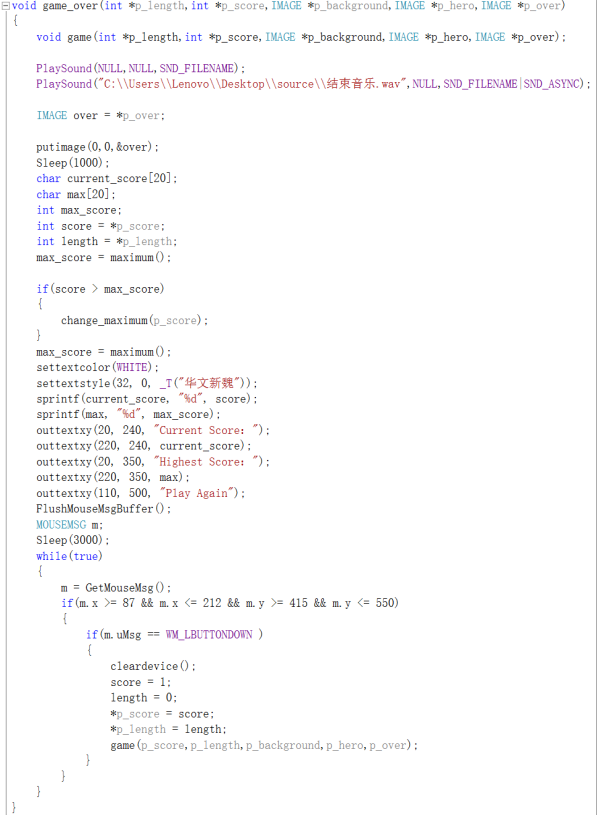
Now, some associated functions will be showed.



P5.4 Creating the initial interface that recognize the mouse click



P5.5 main function of the whole game



P5.6 Main function of the gameover

**6.Testing & Debugging**

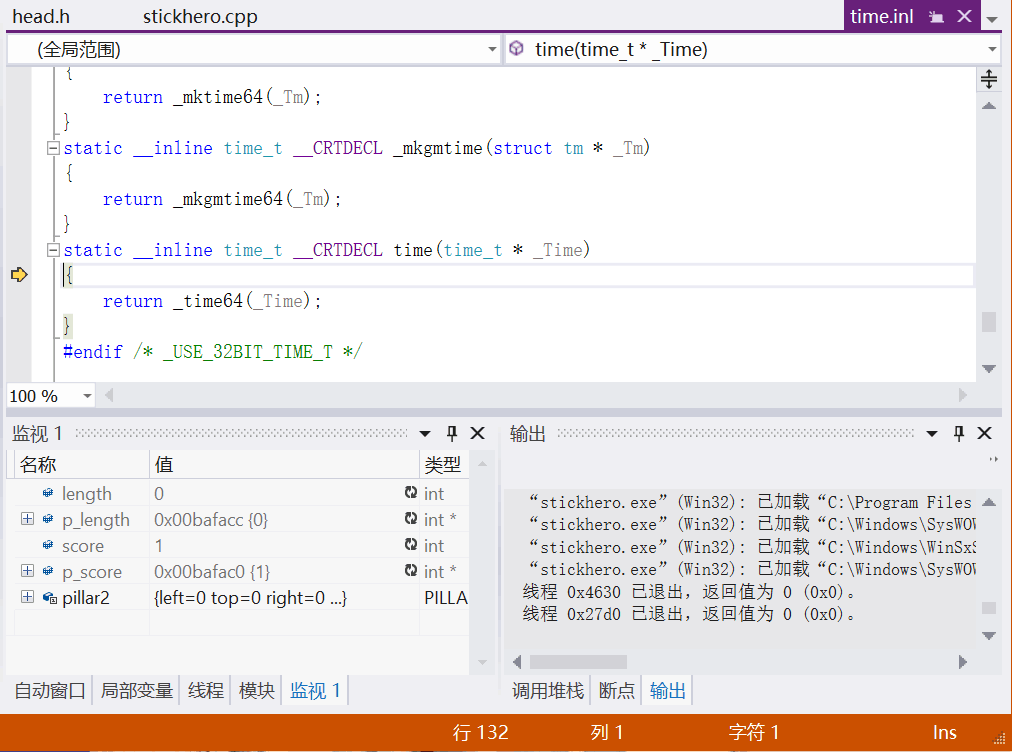
The main focus of this phase is on verifying whether the written code meets the problem requirements under known constraints.

We use testing and debugging together to ensure our project is correct at any time. In testing part, we should make sure that 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 .In debugging part, we put breakpoints in every circulation which has possible failure scenarios, and put the variable we want to watch in the watch table, pressing F10, F11 , along with the project running, we realize our watching target.

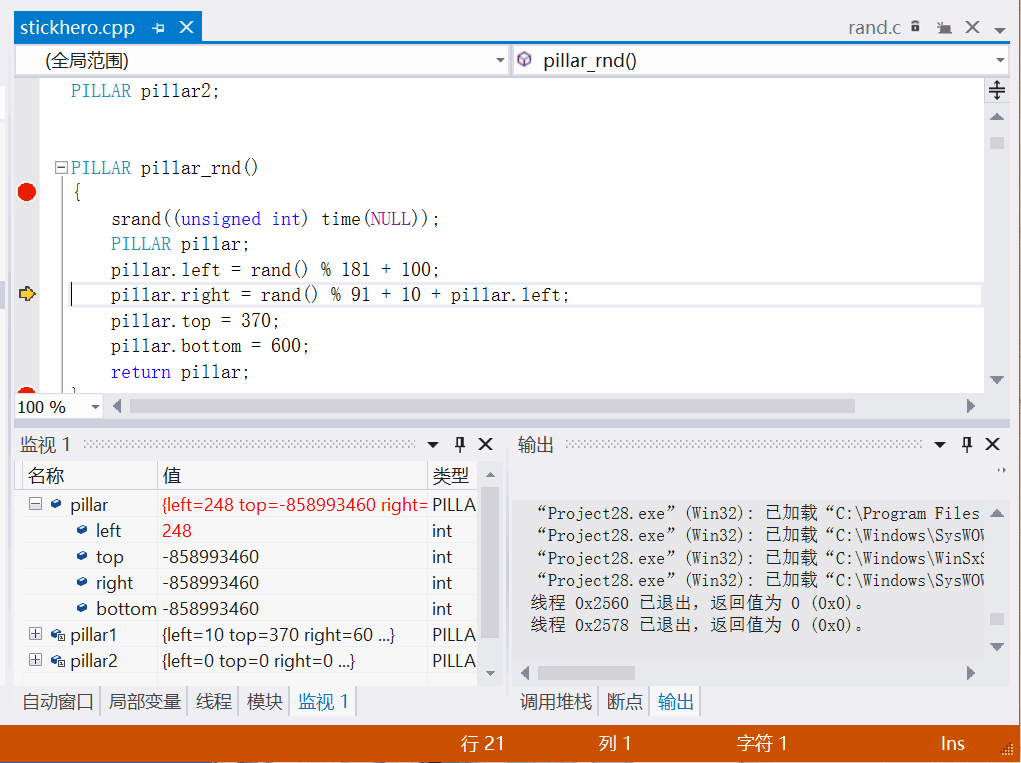
We test and debug to the function order. Our specific testing and debugging process are as follows.

1. **PILLAR pillar \_rnd()**

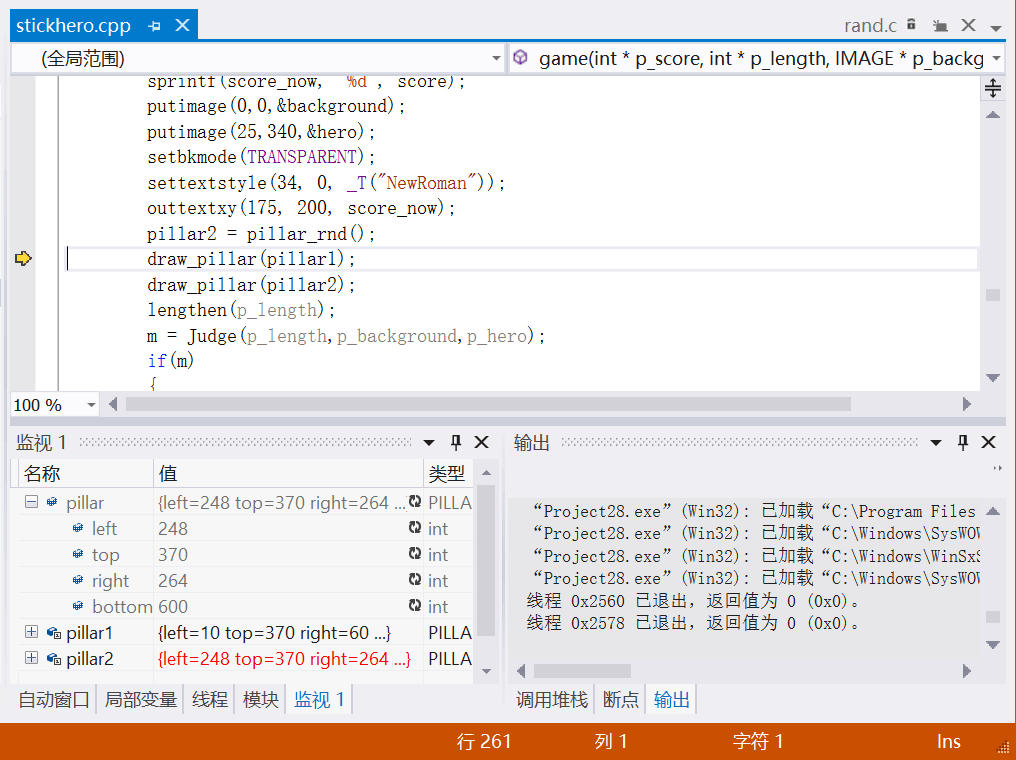
Before calling the function PILLAR pillar \_ rnd() ,the struct pillar’s four parameters(left ,top ,bottom ,right) are initialized to garbage numbers(Picture 2). When calling the function, it’ll generate random numbers (P1)and assign satisfactory random numbers to its four parameters (pillar. left = rand() % 181 + 100; pillar. right = rand() % 91 + 10 + pillar. left; pillar. top = 370; pillar. bottom = 600;)(P3). Through watching the variable pillar. We make sure that this function can randomly generate pillars.



P6.1Generating random number



P6.2The struct pillar’s parameters are garbage numbers at first.

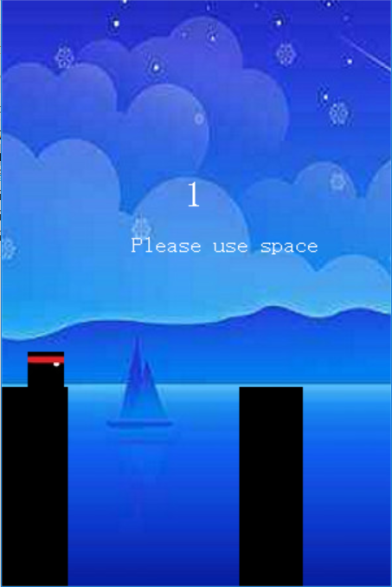


P6.3Struct pillar’s four parameters have been assigned.

1. **void lengthen(int \*p\_length)**

In this function, we need to test that only when press’’, can lengthen the stick, any other invalid input will cause reporting an error ’please press space’(P4), and when press enter, it will break from current circulation ,getting into next circle layer.

|  |  |  |
| --- | --- | --- |
| Input | Expected output | Real output |
| ’ ’ | Stick lengthen |  |
| ‘a’ | Print ’Please press space’ |  |
| 2 | Print ’Please press space’ |  |
| ‘+’ | Print ’Please press space’ |  |
| ‘\r’ | Break |  |

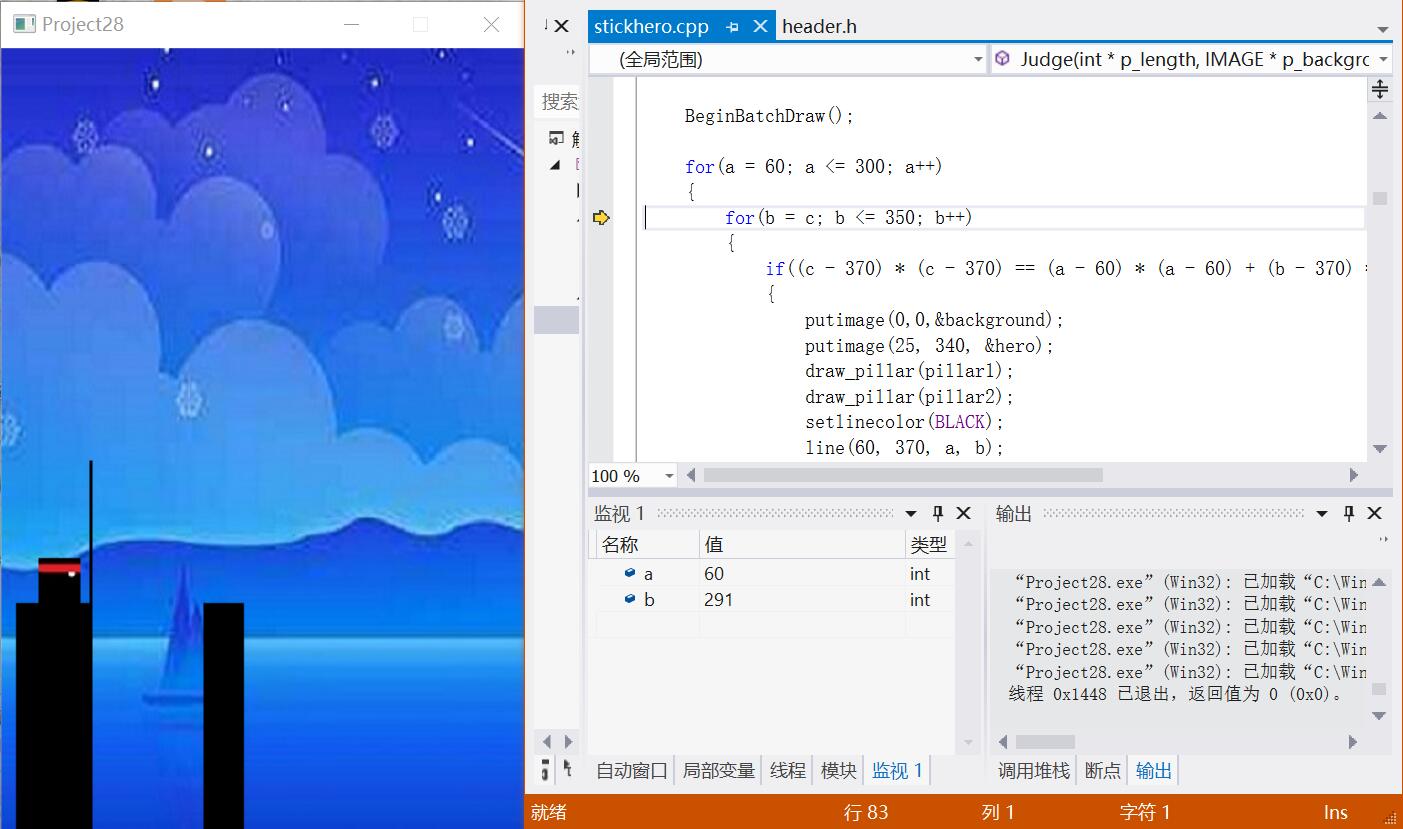


P6.4Report an error as print ’please press space ’when pressing invalid input.

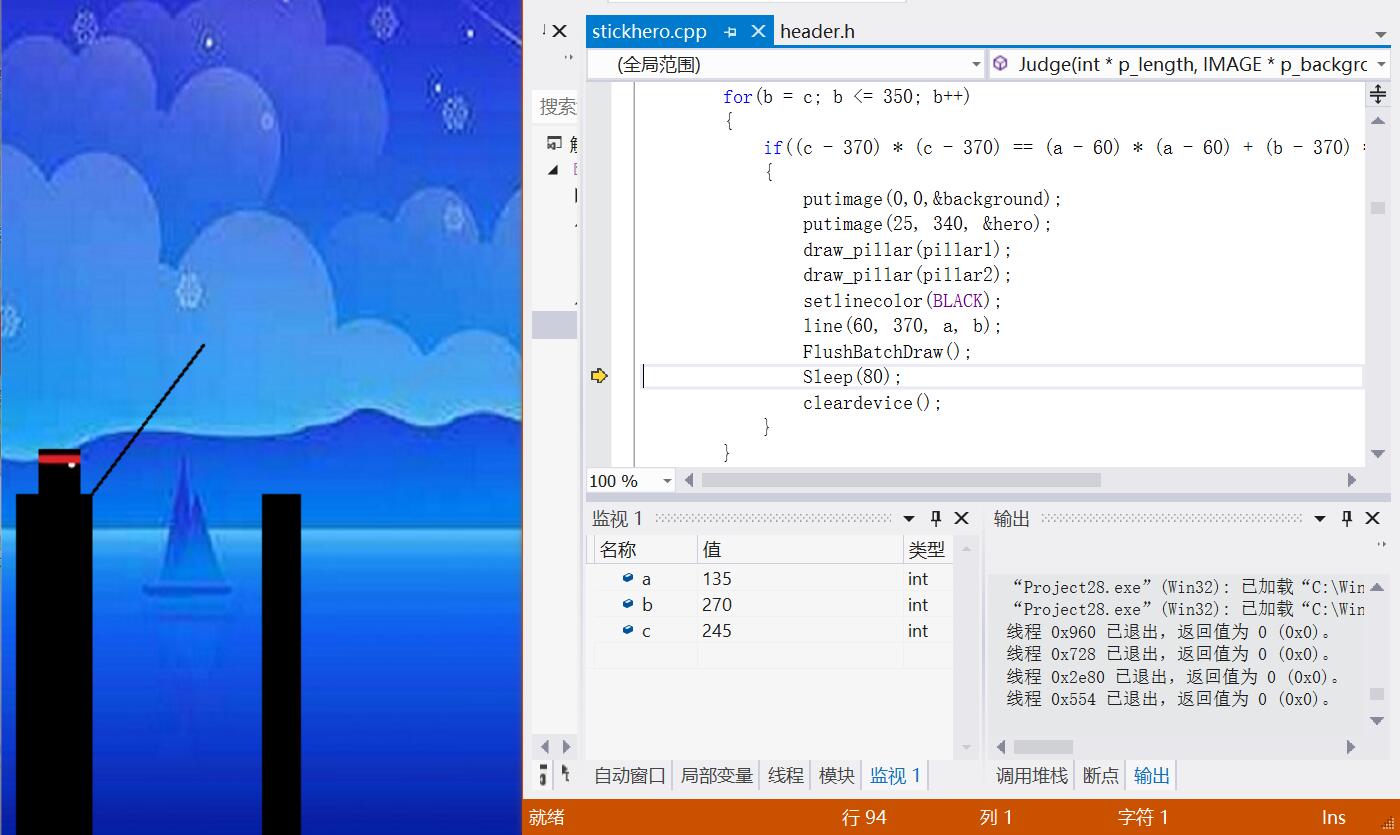
And for the debugging part, we defined a variable l, along with the circulation, l is reduced by stick speed=10. So we put the variable l in the watching table to observe its variation.

**(3) int Judge(int \*p\_length,IMAGE \*p\_background,IMAGE \*p\_hero)**

In this function, we want to simulate the stick falling process and judge whether the hero can pass or not. Considering we’ve initialized int a and b in his function, so we put a and b in the watching table, we expect that if the variable a and b are suitable for the circle formula, the stick is drew on the screen(P5,P6), and a++,b++.And it will break from this circulation until a is bigger than 300, b is bigger than 350.



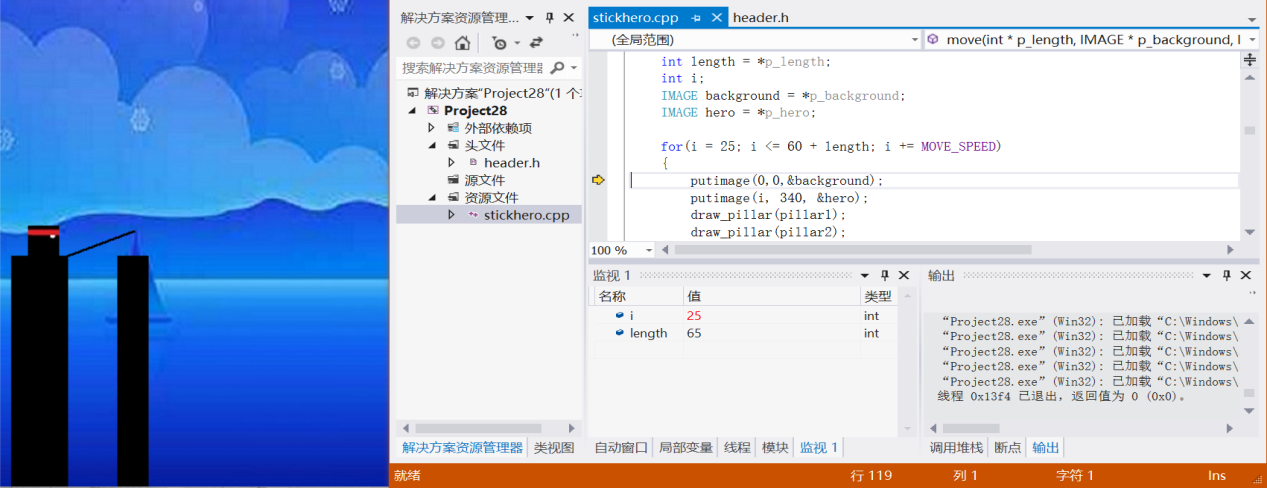
P6.5When a, b are not suitable for the circle formula, it won’t draw a circle



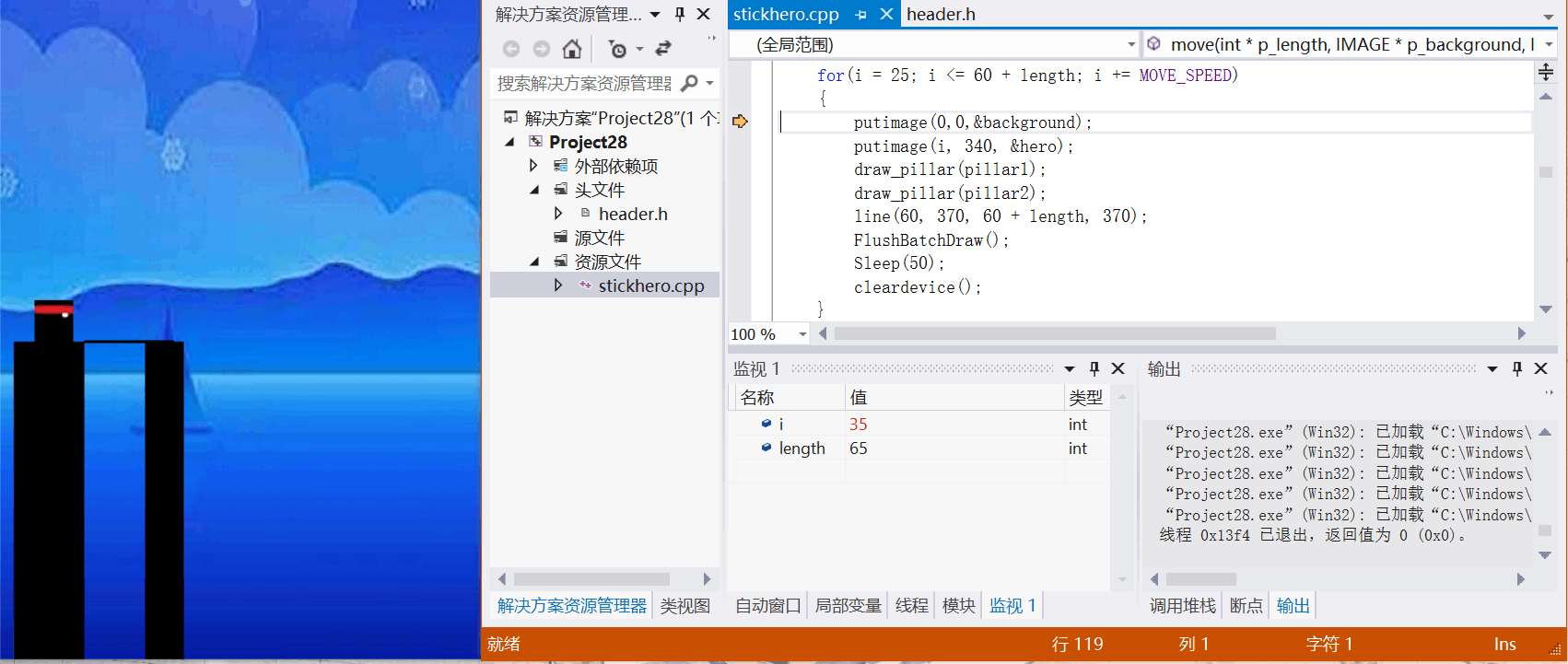
P6.6When a, b are suitable for the circle formula, it will draw a circle

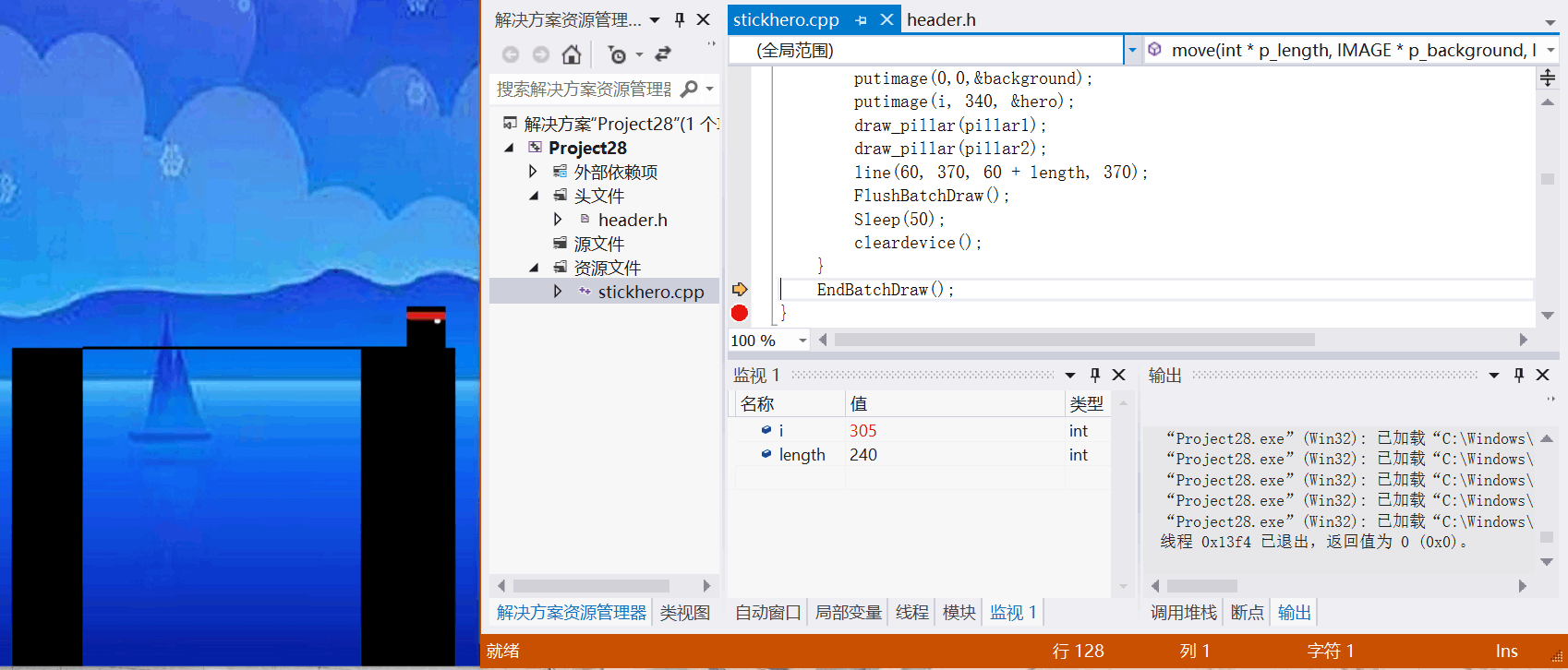
**(5) void move(int \*p\_length,IMAGE \*p\_background,IMAGE \*p\_hero)**

In this function, we want to simulate hero moving process. What we need to test is that our hero can move from the left edge of the stick to the right edge of the stick precisely. So we put the variable I in the watching table, we see that I is initialized to 25 at first(P7), and than i is increased by the move speed(P8), and when i is bigger than 60+stick length, it break from this circulation(P9).



P6.7 i is initialized to 25 at first

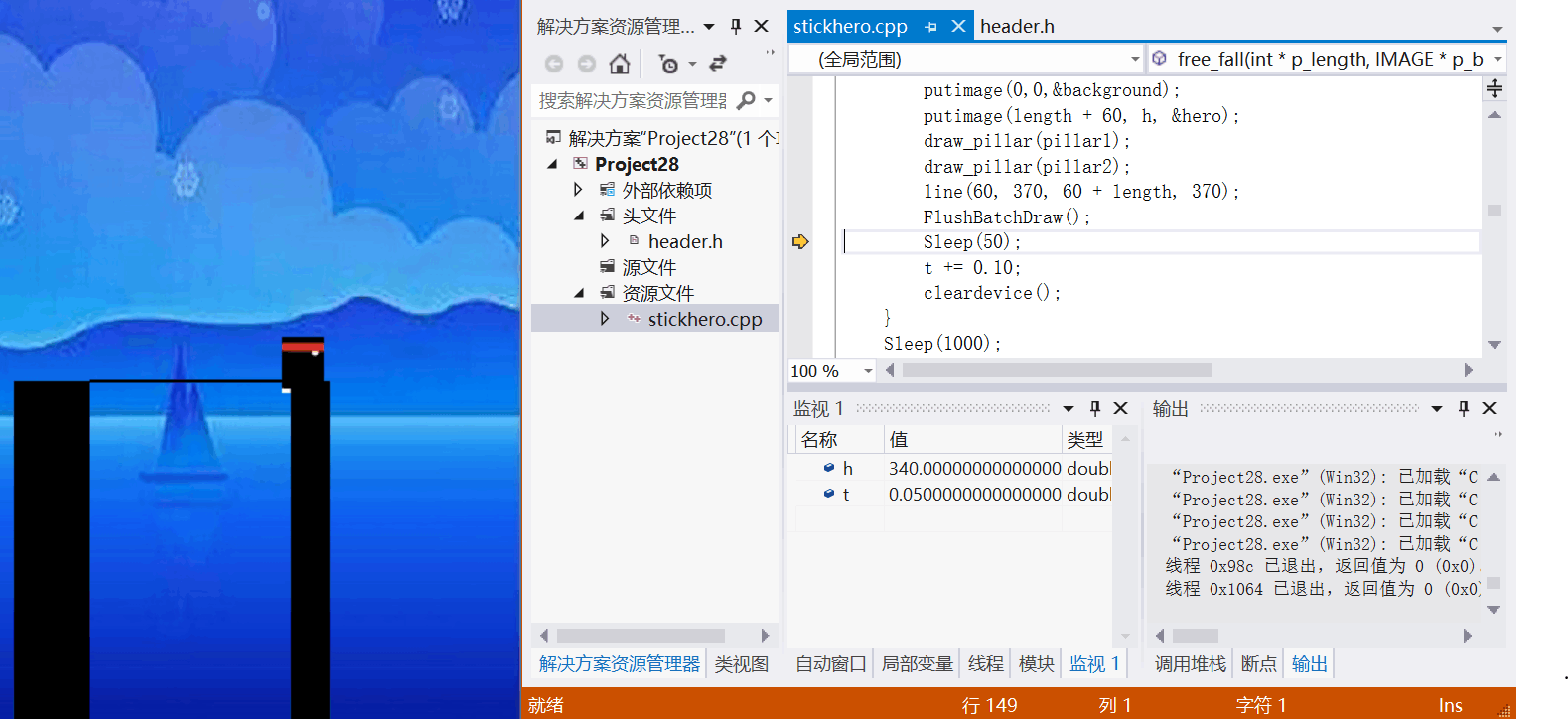


P6.8 i increased by 10(move speed) 

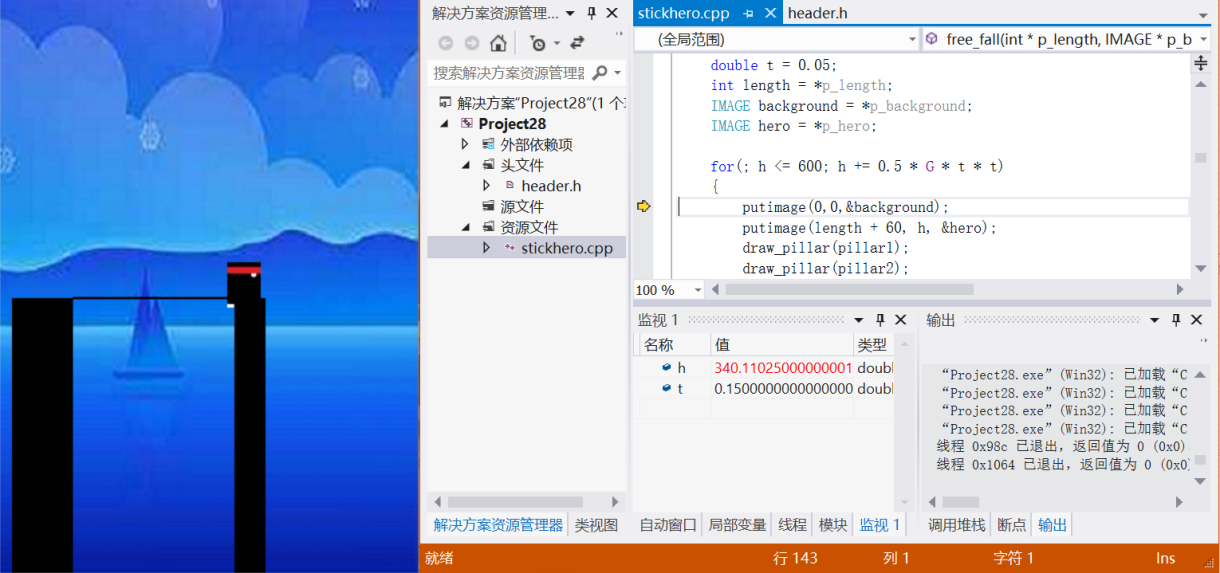
P6.9 Break when i is bigger than 300

**(6) void free\_fall(int \*p\_length,IMAGE \*p\_background,IMAGE \*p\_hero)**

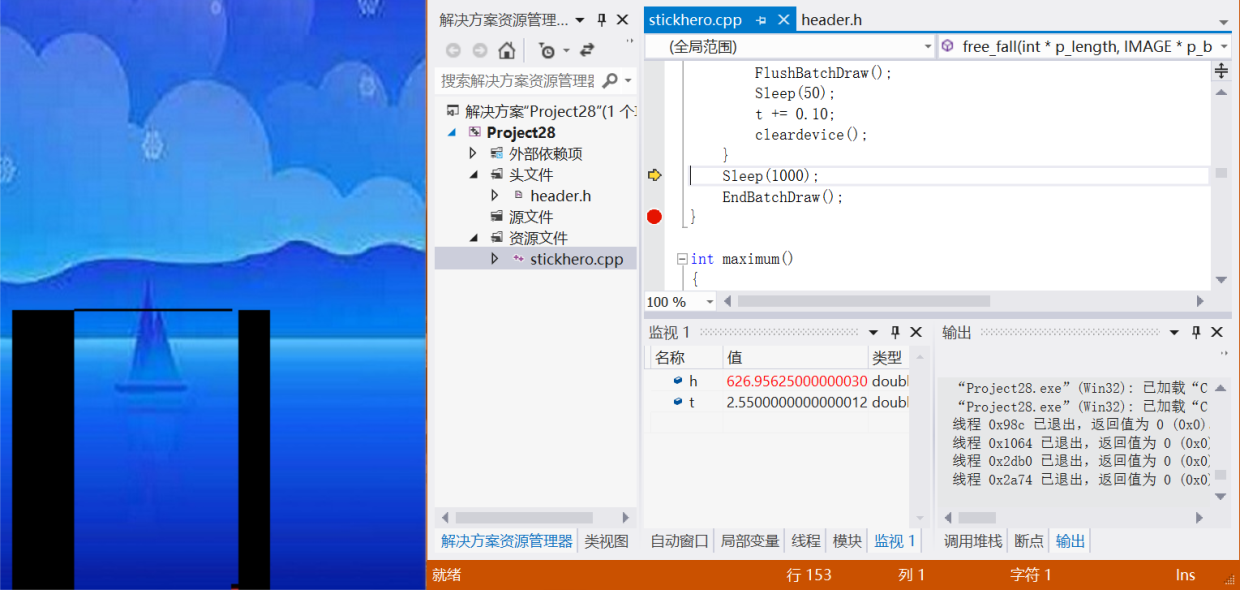
In this function, we want to simulate the free falling process. We need to make sure that hero is falling from the initial height to the ground precisely. So we watch the variable h, we watched that h is 340 at first(P10), and it changes according to the formula h += 0.5 \* G \* t \* t(P11), and break this circulation until h is bigger than 600(P12).



P6.10 h is initialized to 340 at first.



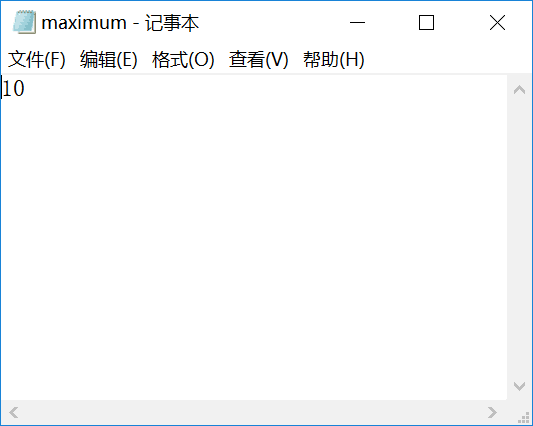
P6.11 h is changed according to the formula h += 0.5 \* G \* t \* t



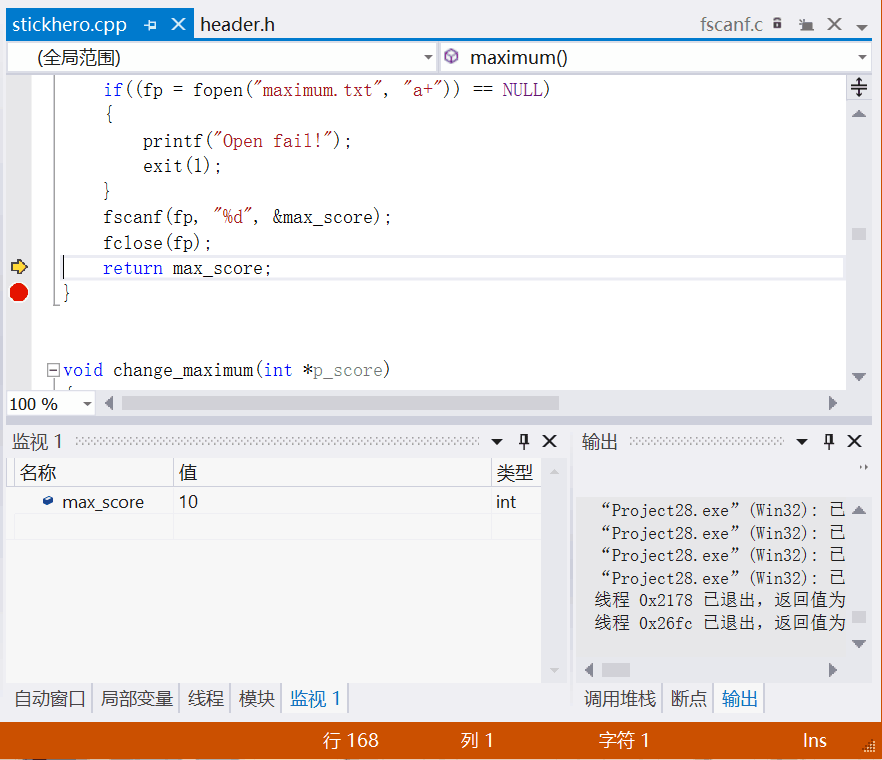
P6.12- break this circulation until h is bigger than 600.

**(7) int maximum()**

In this function, we need to test that it can assign the content of ‘maximum.txt’ to the variable max\_score. So we put max\_score in the watching table, we watched that when running this function, max\_score has the same value as the content of’ maximum.txt’(P13,P14).



P6.13Find and open the ‘maximum.txt’



P6.14-max\_score has the same value as the content of’ maximum.txt’.

**(8) void change\_maximum(int \*p\_score)**

In this function, we need to test that it can change the content of ‘maximum.txt’ by assigning current score to it. So we find and open the ‘maximum.txt’ , and we watched that change successfully.

**(9) void game\_over(int \*p\_length,int \*p\_score,IMAGE\*p\_background,IMAGE \*p\_hero,IMAGE \*p\_over)**

In this function, we want to put max score, current score and ‘Play Again’ on the screen, and when player click the correct position of ’Play Again’ ,it will call game(). So we need to test that when we click the right position, we can play game again, and when we click the wrong position, it will stay current interface, not turning to game start interface.

|  |  |  |
| --- | --- | --- |
| Input | Expected output | Real output |
| Click the mouse in correct position(n.x >= 95 && n.x <= 235 && n.y >= 160 && n.y <= 275) | Draw a red circle on the screen |  |
| Click the mouse in wrong position | Draw a grey circle on the screen |  |
| Not click the mouse | Draw a grey circle on the screen |  |

And through testing, we make sure that this function can achieve the above functions.

**(14) void Login(int \*p\_score,int \*p\_length,IMAGE \*p\_background,IMAGE \*p\_hero,IMAGE \*p\_over)**

In this function, we want to realize the function that when we click the mouse in correct position, it can draw a red circle on the interface. When we click the mouse in the wrong position or not click the mouse any more, it will draw a grey circle.(P15)

|  |  |  |
| --- | --- | --- |
| Input | Expected output | Real output |
| Click the mouse in correct position(m.x >= 87 && m.x <= 212 && m.y >= 415 && m.y <= 550) | Return to game start interface and play again |  |
| Click the mouse in wrong position | Stay current game over interface |  |
| Not click the mouse | Stay current game over interface |  |

After testing, we make sure that our function is correct.

P6.15When pressing in correct\wrong position or not pressing , draw a red\grey circle.

**7. Result and Conclusion**

After a regular basis of programing, compiling and debugging, we can say that we have mostly finished our initial idea of our project, which is to use our C language skills to try to imitate the online-version. It turned out that we have done it quite well. Except of the definition of the background pictures is rather lower than the commercial one.

During our C programming, we have learned to cooperate with our teammates, everyone is an essential part within our system, and we can only make progress when everyone is doing their best for the group.

We have also learned quite a lot about the understanding the basic uses of the C language and we have practiced quite a lot about debugging and other knowledge. While researching the functions and header files, we have gained many techniques about programing. Most importantly, we have learned to think like the denugger.