**Programming of C language**: Let’s Jump

**Abstract**

This program is a small game called***Let’s Jump***. We have this idea from the popular Wechat small program. ***Let’s Jump***was prevail several months ago. Young people like it because it is easy to play but difficult to get high scores. We just wanted to imitate it and create our 2-D version ***Let’s Jump****.* Our aim is to compile a similar game in C and the game should be played by using the button of players’ mouse only. Because of the limitation of our skills in C, we can not create the 3-D version. But other function like moving a little man, getting the time players spent pressing the button and recording the scores can be done by ourselves. And we also spent time decorating the game interface to make it players friendly.

1. **Introduction and problem statement**

In this project, we are going to imitate the Wechat small program. So this program has the same function as the Wechat version. There should be a timer to record how long players spend pressing the left button of mouse. And then transfer this time variable into a distance variable. And we need to move the man and bricks after getting the distance variable. Also, as a game, ranking is needed all the time. So we create a windows to show the highest 5 scores in history.

What problems shall we solve to finish the project? First, how to get the time is the most important and prior. Second, how to move the man and bricks, or maybe we can say how to move the pictures in the windows is worth paying attention to. Third, for a game, a good interface is half of the program. We should think how to make the game interface attractive.

**2.Analysis**

Before we state this part, let us see the picture of the Wechat version first.



The left one shows us the location of the man and the boxes (in our program we changed the boxes into bricks ). In that picture, we can see a special box, if the man jump onto that box successfully, player can get extra 30 marks.

The right one shows the movement of the man. We do not think we can implement the rotation. But we need to make the man move in the track of a para-curve.

In our project, there are seven main parts, which are recording time, the jump of the man, the movement of the bricks and man, generating new bricks, background music, ranking and restart.

First, how long the left button of mouse is pressed is needed, because the track of the man’s jump is determined by it. We need to get the time and transfer it into the distance.

To make the game more playable and realistic, the man needs to jump in a curve which likes a parabola and the shape of the curve is depended on the length of time that was spent in pressing the left button of mouse. Besides this, how to implement the movement is also a problem. Because we know it is difficult for us to implement the movement of a picture in C language.

Let us think about the Wechat version of the Let’s Jump. The bricks should move left with the man after a successful jump. We should achieve moving the combination of the man and the second brick in the interface to the position where the first brick is. The purpose of the moving is to make room for the next brick and create a dynamic feeling for the game. Compared with the former problem, this one looks more difficult because we need to move a combination of several pictures.

After movement, a new brick should be rightly drawn. The new brick needs to appear in a proper time and at a proper location. If it appears early, The visual effects will be poor. But if it appears later, it will lead to the wasting of time and the delay of game process. And the location of the new brick should be determined at random within a certain range.

Background music is needed to optimize player experience. More importantly, the music played when the left button is pressed can make the game easier. Without music, it is difficult to judge how long the left button should be pressed to make the man jump to next brick.

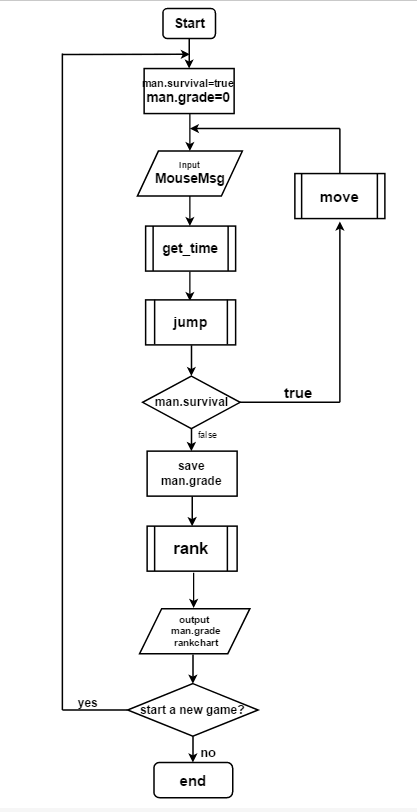


This picture is the ranking chart of the Wechat version. Our program should have a simple chart to show the highest several scores in history.

Ranking is the reflection of a player’s proficiency in playing this game and it will encourage the player to play it more times. So we need to record data and show the rank.

The last one is the restart part. We are going to give the player a chance to choose whether to start a new game or not. it should appear after showing the rank.

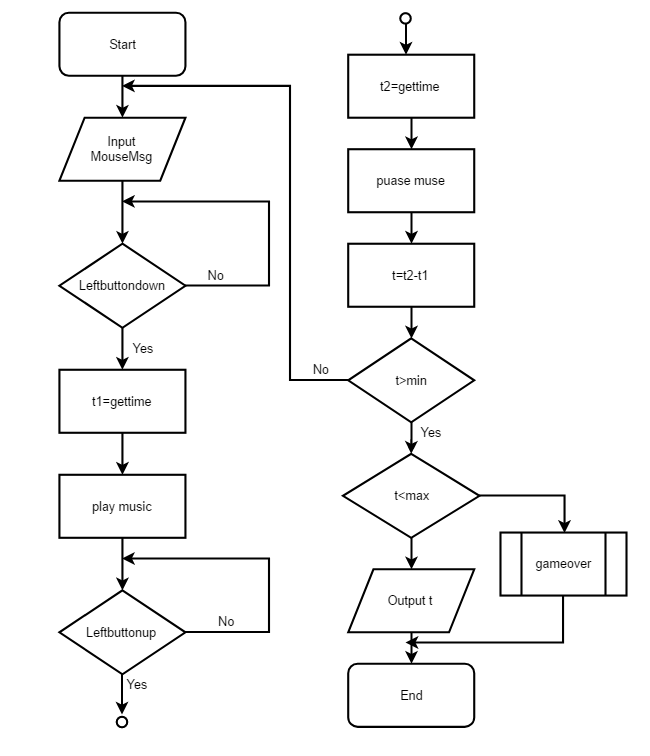
**3.Design**



Picture 3.1 The Design for Whole Program

As is shown in the picture, our program is basically linear. At first we define some important variables, such as man.survival and man.grade. Man.survival is to judge whether the man is survival or not, man.grade is to record grades. Then we get the message of mouse,including it’s position and state. We use a function to get the time that player spend on pressing the left button of mouse. In the jump function, we transform time to length. The man will jump, and how far and how high he’ll jump depends on how long the button is pressed. If the man jumps to the next brick successfully, he will survive. But if he fails, man.survival will become false. Next it’s the judging part. If man.survival is true, the program will go to the move part. In this function, bricks and man will move back and new brick will be drawn. Then the program will go to get\_time function again. If man.survival is false, game will be over. It should save the final grade and then go to the ranking part. In the rank function, it will open the record file, using the data in it to do ranking. Then the coming part is to show the rank and players’ final grade. Finally program is end.

1. Record the length of time that left button of mouse is pressed

Picture 3.2 The Design for Timer Function

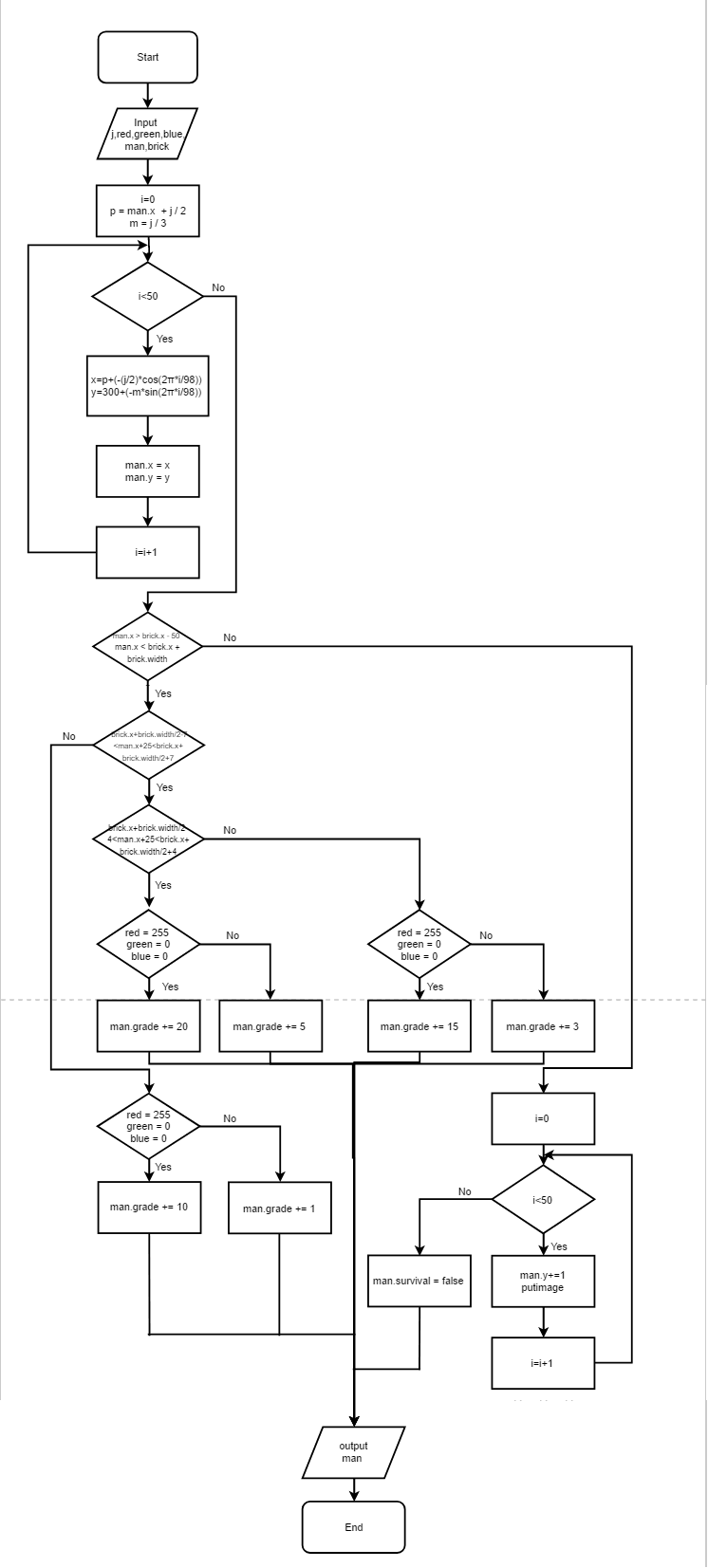
To get how long the player presses the left button, the first thing is to get the message of the mouse, knowing when the button is pressed and when it is freed. Use a function to get the message of mouse, including when the button is pressed and freed, and record these two time.

The second step is to do the calculation. By dividing them, we can get the time difference and it is the length of time that we want. If the length of time is shorter than the minimum we set in advance, the function will go to get mouse message again. It won’t move to next step until it gets a value bigger than the minimum. Meanwhile, if the length of time is so long that it will cause the man jumping out of the screen, this function will call another function, which is gameover, meaning the program will go to the ending part directly.

1. Realize the curve jump of the man

In this part, there are two important things we need to solve. First, we need to let the man jump along the curve. Second, The shape of the curve trajectory should be determined by the length of time when the left button is pressed.

To make the jump more like a parabola, we apply trigonometric function into abscissa and ordinate instead of using linear function in ordinate. And we add length variable, which is transferred from time variable, to the expression of the trigonometric function. In this way, the shape of the

Picture 3.3 The Design for Jump Function

curve trajectory can vary with time. And to realize the jump process, we need to cover the previous figure and add the figure to it’s new position. A cyclic structure is needed in this jump function. The for loop will run 50 times, which is controlled by a variable called i. Every time the loop runs, the man will move a distance.

There is another important part in the jump function. After jumping, recording grade is followed closely. There are several conditions that may appear, so we design some bonus.

（1）The man jump to the middle point of the brick. If it’s a red brick, add 20 points. Otherwise, add 5 points.

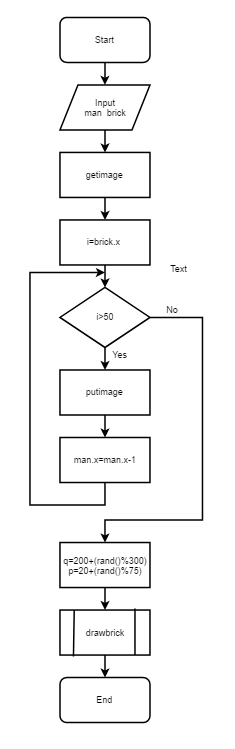
（2）The man jump to the middle area of the brick. If it’s a red brick, add 15 points. Otherwise, add 3 points.

（3）The man jump to the edge of the brick.If it’s a red brick, add 10 points. Otherwise, add 1 points.

1. Move

After the man jumps to next brick, these two bricks should move left with the man, to achieve moving the combination of the second brick and the man to the position where the first brick was, leaving space for the formation of new brick.

To make it, we choose to take a particular area as whole. When it’s time to move, the whole area will move left till the position of the second brick coincides with the original position of the first brick. We use a for loop to realize the moving process.In this process, man.x keeps changing. Every time the loop runs, the image of this area will be put on the interface in the position that is decided by current coordinate. When it stops moving, the current position of the man is the initial coordinate of his next jump.

Picture 3.4 The Design for Move Function

The reason for taking a particular area as whole is to simplify the jump process. Meanwhile, choosing a particular area instead of the whole interface, we can avoid the problem that the patterned part of the background picture moves with the bricks and makes the background incomplete and bad-looking.

Then the program will generate new data for the next brick and call drawbrick function to draw it.

1. Draw new brick

After the former bricks move left, new brick should be drawn on the right. An important thing is that the abscissa, width and color of the brick are determined randomly in advance.

1. Music

There should be five kinds of music played in different time and different occasions:

①Start music played in the welcome interface;

②Background music played as the game progresses;

③Prompt music played when the left button keeps being pressed;

④Ending music played when showing the final grade. If the grade is higher than a certain number, it will play a clapping music. Otherwise, the music will be an encourage one.

To achieve this function, we need to control the start time and end time of music, and pay attention to it’s different kinds.

1. Ranking

When game is over, the rank, which is based on all the grades that previous players have got, will be shown after showing the grade.

In order to show the rank, the first thing we need to do is to record the data from previous games, putting them in a file to make them callable when ranking. When a game is over, it’s grade will be recorded in the file, and then we rank the grades in the file, including the one recorded just now. Finally, we need a interface to show the rank and let players click OK to continue.

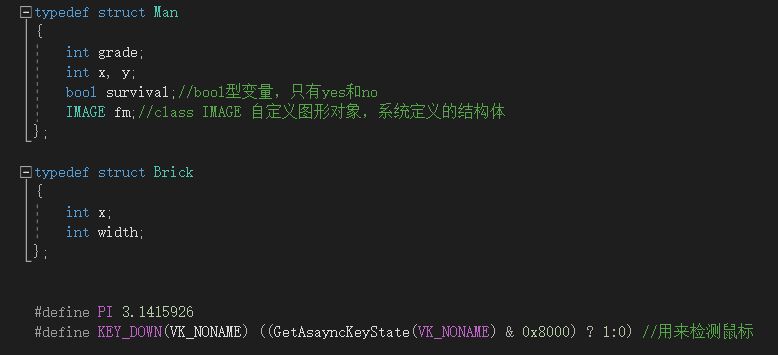
1. Restart

To enhance the playability of the game, the restart function is needed. After showing the rank, there will be a messagebox saying “Yes to retry, No to exit”, to let players choose whether to start a new game. If the player choose Yes, game will restart from the game interface, not the welcome interface. If No is clicked, the game is completely over.

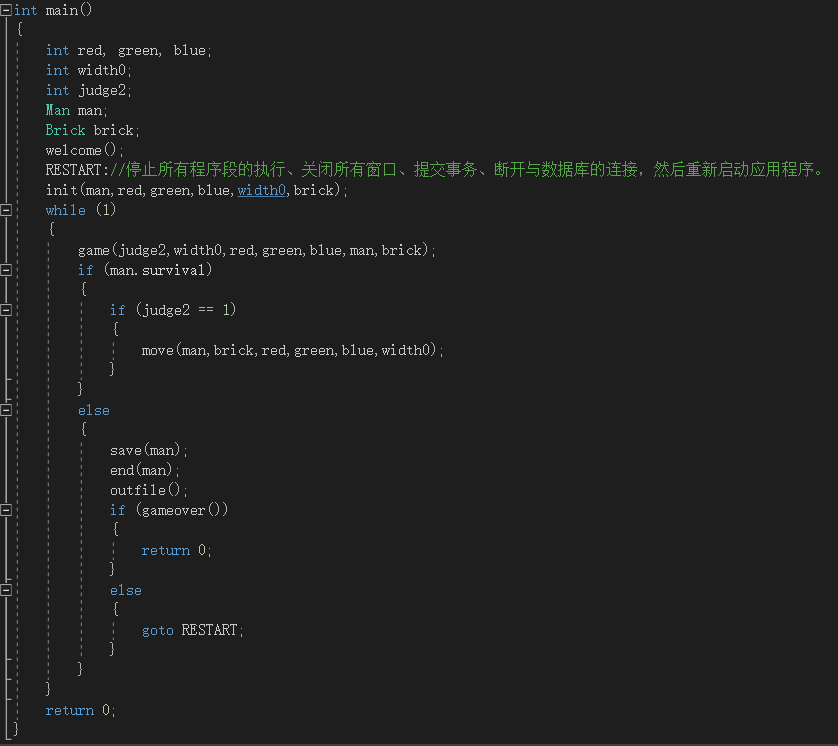
|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Real Result |
| MouseMsg | t1 t2 |  |
| t1 t2 (t2-t1=140) | t |  |
| t1 t2 (t2-t1=80) | Return to while |  |
| t1 t2 (t2-t1=100000) | gameover |  |
| man.x=447 brick.x=400 brick.width=50 | man.grade=man.grade+1 |  |
| man.x=387 brick.x=400 brick.width=50 | man.grade=man.grade+3 |  |
| man.x=397 brick.x=400 brick.width=50 | man.grade=man.grade+5 |  |
| man.x=137 brick.x=400 brick.width=50 | man.survival=false |  |
| Man.grade | record |  |

**4.Implementation**

In this part, we will show some important parts of our code and explain them.

Figure 4. 1

In our program, we need a picture to represent the little man, so we declared a pointer to a the picture we need as showed in Figure 4.1.

Figure 4. 2

Picture 4.2 shows our main function. Just as what said in the design part, this function was coded in the thinking of design part. At first, we created a welcome interface with the logo of TAQ Honors Program In Science. Then we initialize the game interface and start the jump game.

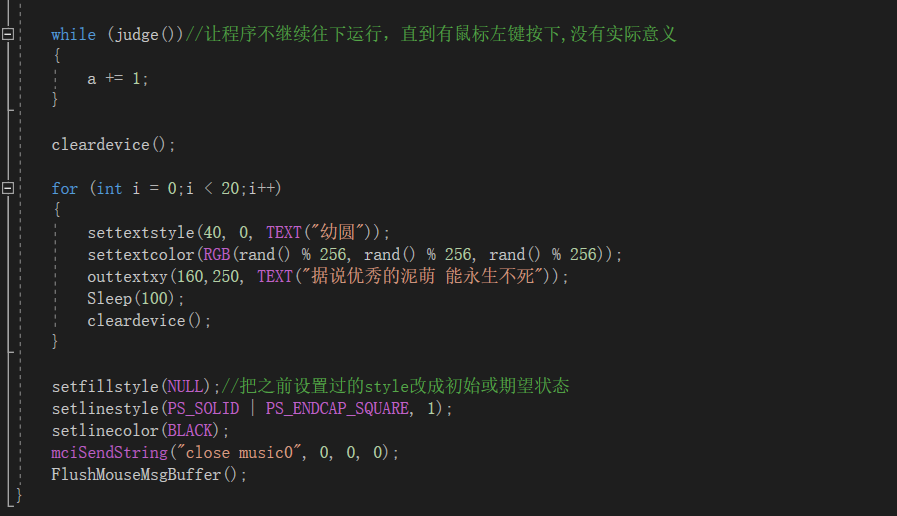
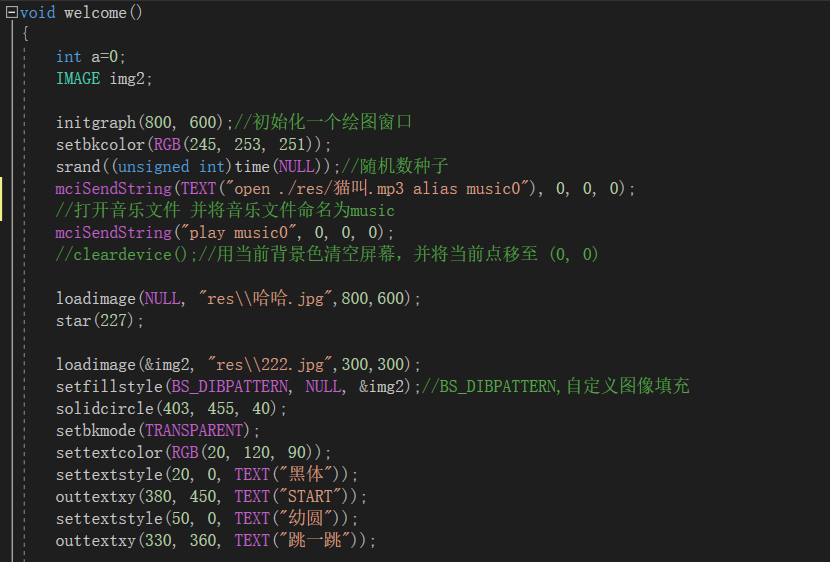


Figure 4. 3

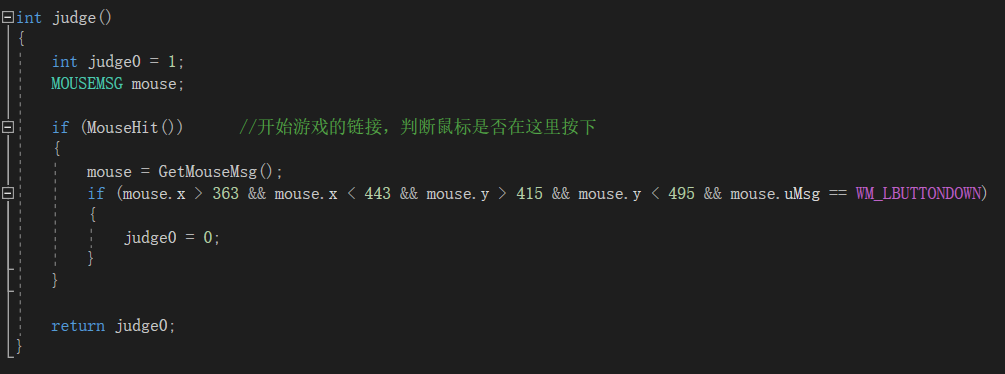


Figure 4. 4

As shown above, it is our welcome function. In this function, we set a background picture and background music. Also we set a button with the word “start game” to judge whether the player have press the button. Once the button has been pressed, the program will go to the game mode. And the while circle is used for this judgement. The judge function is coded as showed in Picture 4.4. And we will not explain the judgement in details.

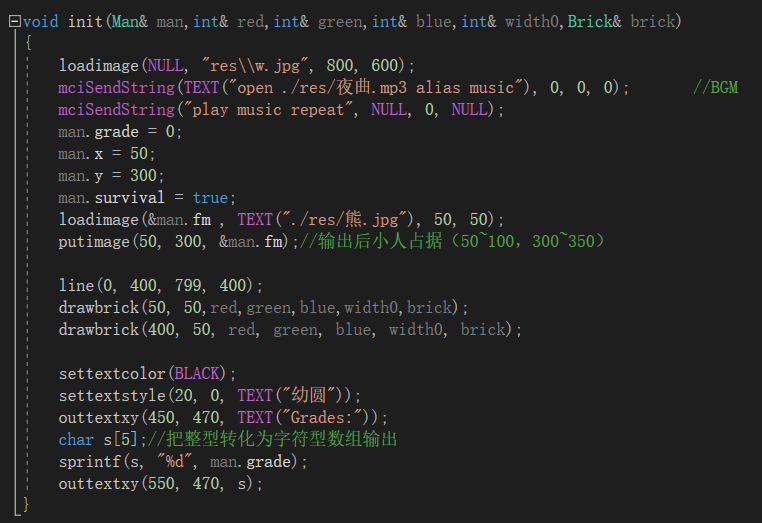


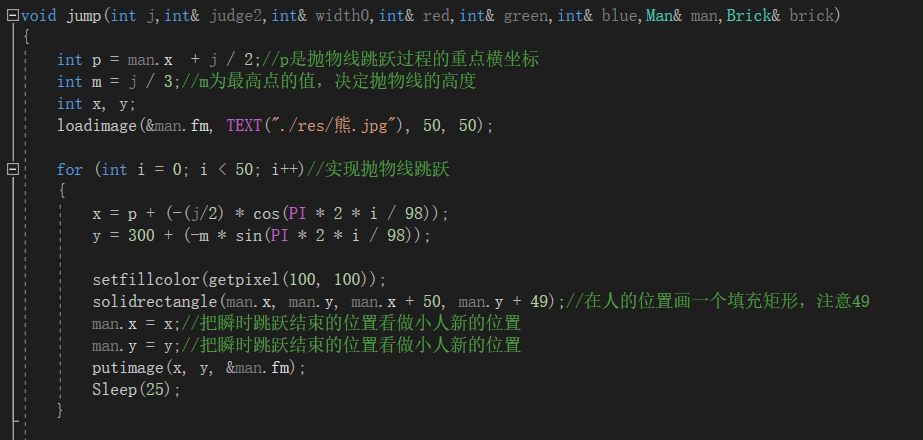
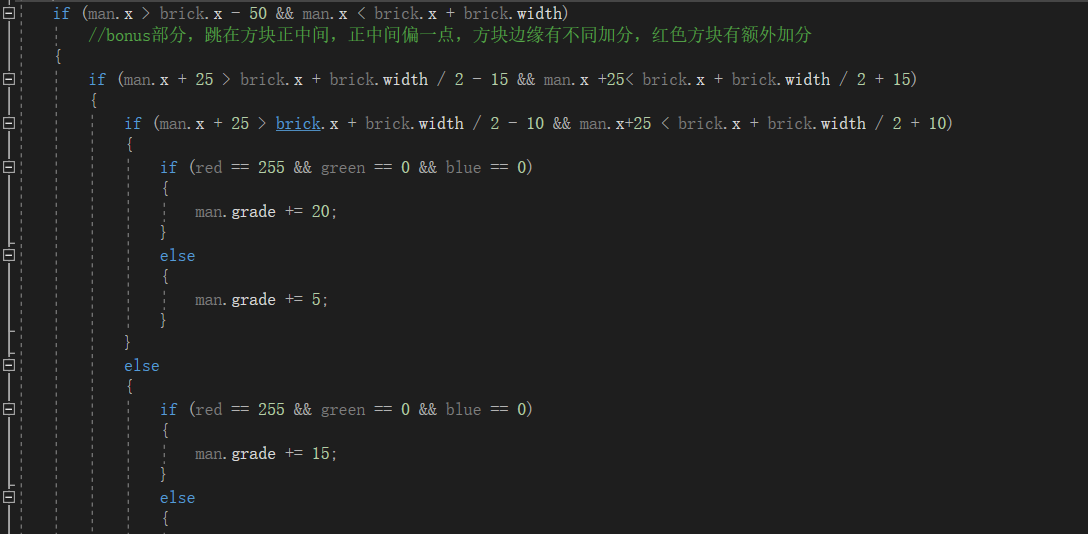
Figure 4. 5

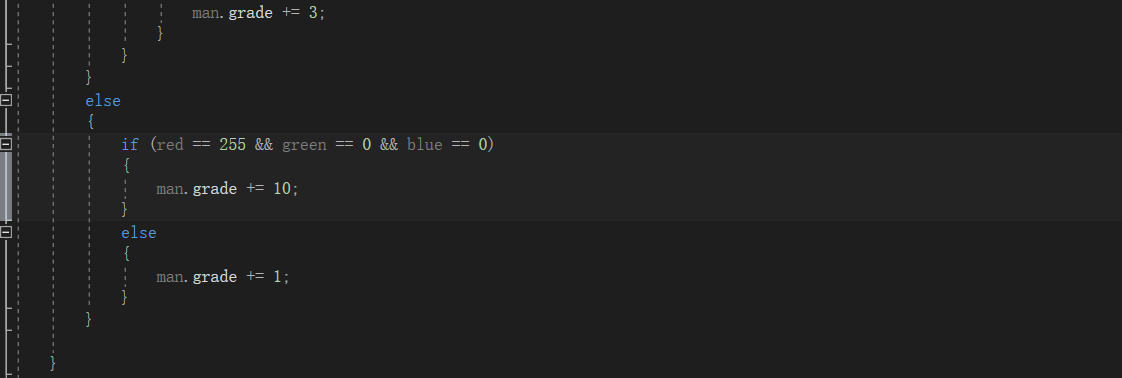
This is the initialization of game interface, which is the beginning of the game. We set man.grade as 0 and the initial coordinate of the man is decided. To let the players know their grades, we put a text at the lower right corner. It can show the grades as game goes on.



Figure 4. 6

The soul of our program——the timer function is showed in Picture 4.6. This function implement the switch between time variable and distance variable. We need to record how long the player press the left button of the mouse. To make the game playable, this operation is very important. So we call it as the soul of the program. There is one thing worth paying attention that is the “(if t2-t1>100)” . The sensitivity of getting mouse message is not as high as we expect. So we set the judgement here to ensure the time variable the computer get is valid.





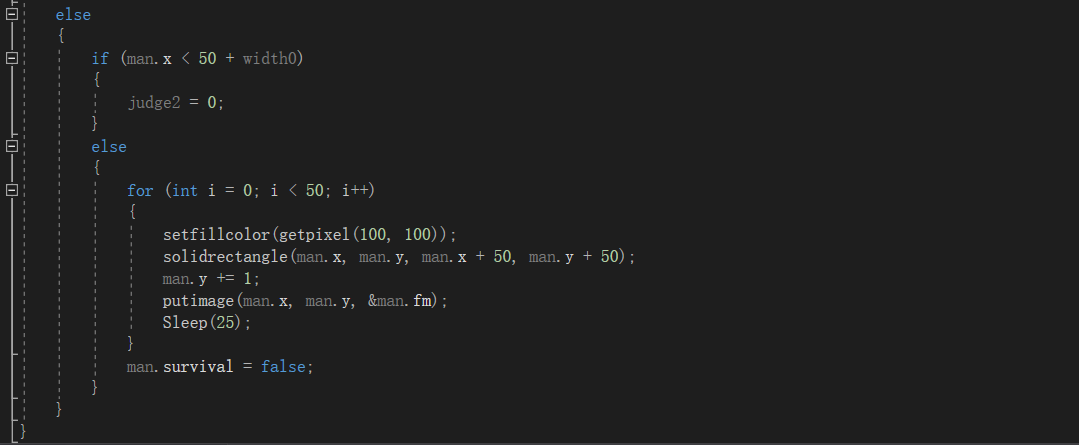


Figure 4. 7

Picture 4.7 shows our jump and grading system. To move the picture of the man, we make a circle to display the picture one by one and let a rectangle with the color of the background picture go with the man. So the latter one can cover the man. Then to players or observers, it looks like the move of the picture. Then about our grading system. To make the game more playable and attractive, we created a bonus part. For example, when the man jump onto the canter of a brick, you can get extra 10 marks. And if the man only jump onto the edge of the brick, only 1 mark is offered. Besides this, red bricks can bring players another 10 marks. The judgement of death is also coded here. If the man does not get the edge of the next brick, go to the end function. And later I will explain the end function. So we have a big “if” nest here.

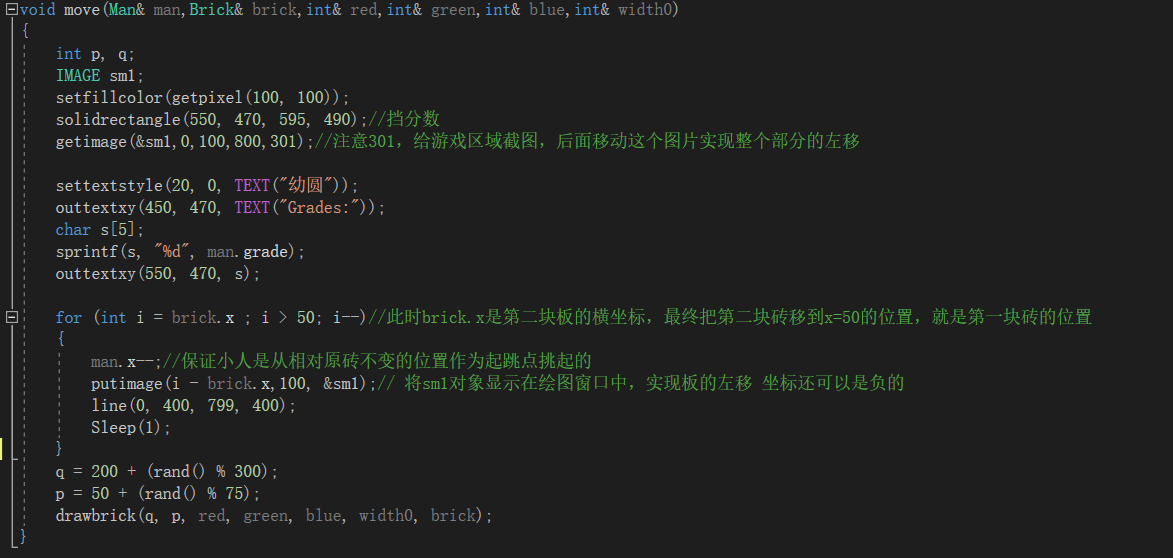
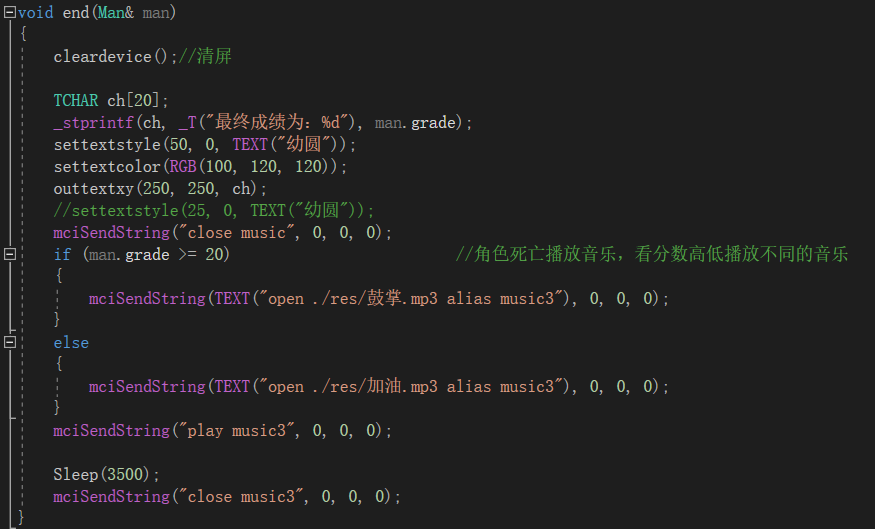


Figure 4. 8

After jumping to the next brick, we need to move the game part to the left side of the interface. So here comes the move function. Here we select the central part of the interface and move the screenshot to the left side. After this the next brick can have enough room to appear.

The move function only plays its role when the man jumps onto a next brick successfully. So here comes the end function which plays its role after the players fail to operate successfully.

Figure 4. 9

Picture 4.9 shows the end function. The game interface will change into an end interface. The players’ final grade will be displayed here and depends on how many grades they get, the program will play different music.

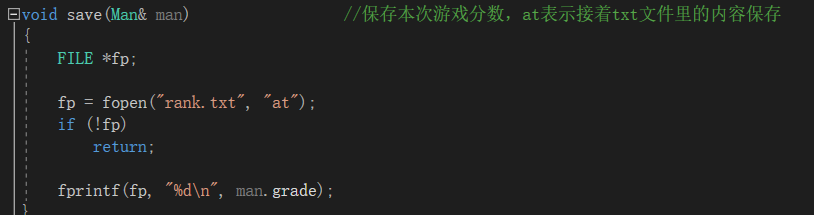


Figure 4. 10

Here we can see a “save” function. It just save the players’ grade into a text file. “at” means that put the new number at the last in the text tile.

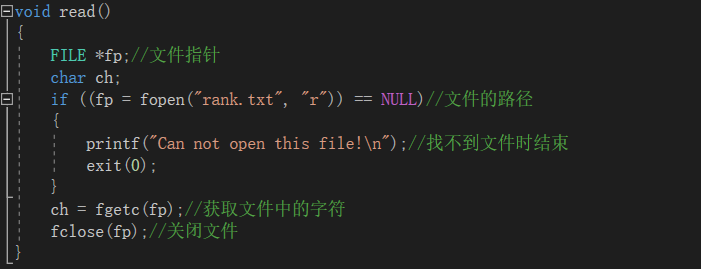


Figure 4. 11

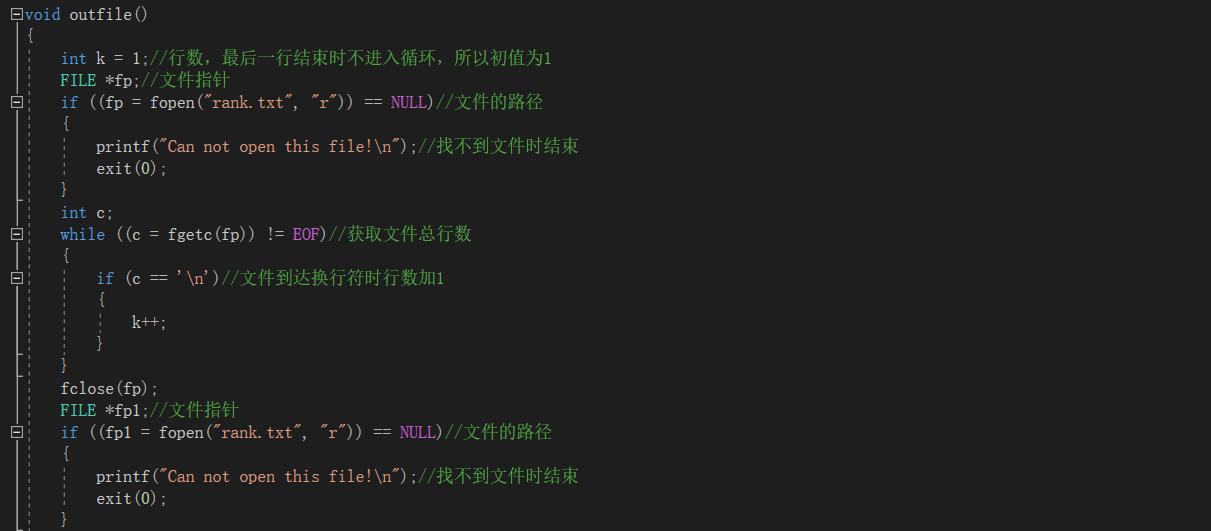
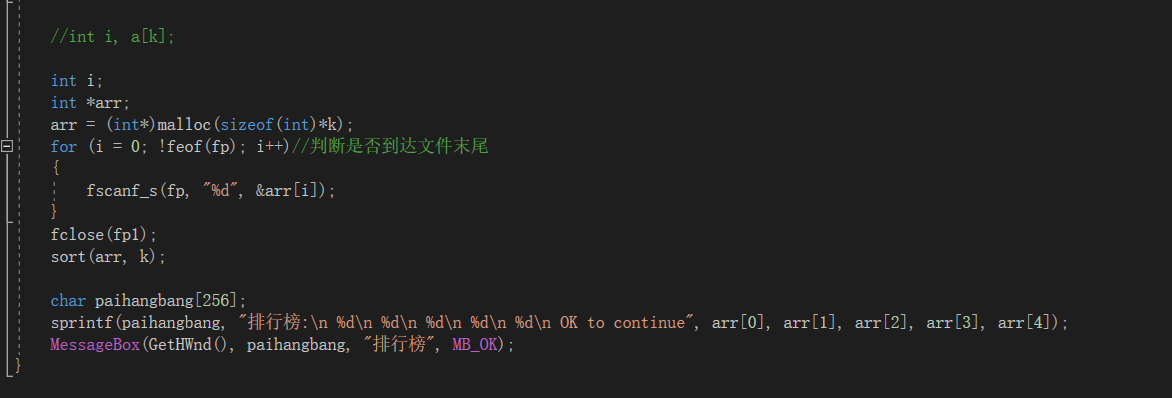
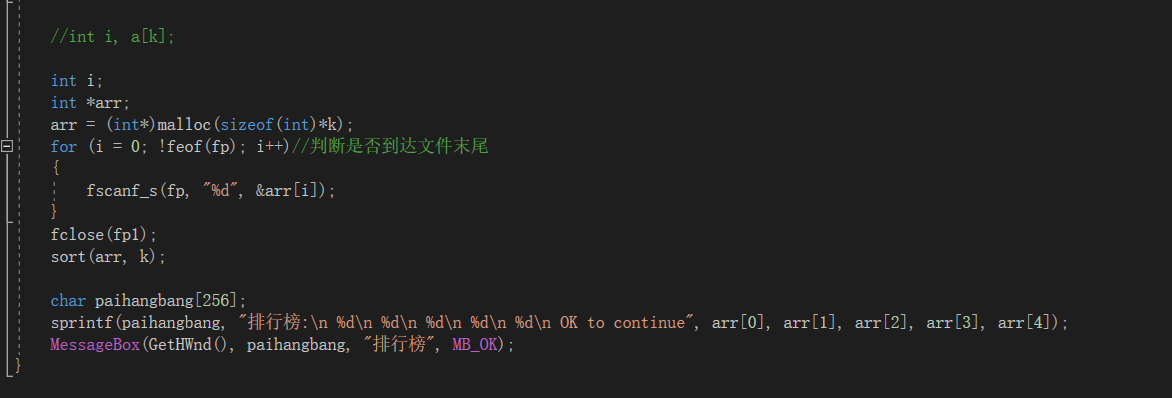


Figure 4. 12

Picture 4.11 and 4.12 figure implement a ranking system. After showing the final grades, it will create a windows with the highest 5 grades in history. And the players can know their ranking. To be specific, we have “read” and “outfile” two functions to implement this. The read function is used to open the text file in which we saved the grades. And the outfile function is used to get and show the ranking. But because we do not have dynamic array in C, so we need to set a circle to know the number of the grades. Then another circle implements the output.

**5.Testing**

In this part we will test our program to make sure that there is no bug in it. The following figures will show the result of testing. But first of all, let me show you our test plan.

|  |  |  |
| --- | --- | --- |
| Input | Expected result | Real result |
| Jump to the center of a ordinary brick. | Man.grade + 5 | Man.grade + 5 |
| Jump to a the center of a red brick. | Man.grade + 20 | Man.grade + 20 |
| Jump to a red brick but not the center. | Man.grade + 15 | Man.grade + 15 |
| Jump to a ordinary brick. | Man.grade + 3 | Man.grade + 3 |
| Jump to the edge of a red brick. | Man.grade + 15 | Man.grade + 15 |
| Jump to the edge of a ordinary brick. | Man.grade + 1 | Man.grade + 1 |
| Pressing time >5000ms | Return 0 | Return 0 |

First we test the jump function.

|  |  |  |
| --- | --- | --- |
| Input | Expected result | Real result |
| Man.x=71  St=390 | Man.x =461 | Man.x=460 |
| Man.x=108  St=403 | Man.x =511 | Man.x=509 |
| Man.x=139  St=217 | Man.x =356 | Man.x=354 |
| Man.x=127  St=435 | Man.x =562 | Man.x=560 |
| Man.x=71  St=72 | Man.x =143 | Man.x=142 |
| Man.x=50  St=472 | Man.x =522 | Man.x=521 |
| Man.x=50  St=23 | Man.x =73 | Man.x=71 |
| Man.x=50  St=464 | Man.x =514 | Man.x=513 |
| Man.x=50  St=508 | Man.x =558 | Man.x=557 |
| Man.x=50  St=27 | Man.x =77 | Man.x=75 |

This is the testing case of grading system. Man.grade is the coordinate of the man. And st means the distance that the man will jump. We make the st equals 1/5 of the pressing time. As we can see, the real results have vary small error compared with the expected result. The difference between real results and expected results is only less than 2. So we can say it implements our demand successfully. The difference is from the approximation of integer. The difference can be ignored because it do not affect the game.

Then let us see the test case of timer function.

There is a big problem that makes it very difficult to test it. When we debug it, the program will not go on until the pressed left button of mouse is loosed. But the both of the down and up conditions need a break point. So it means we can not debug the two timeGetTime function. And we can not get the value of the two time variable. To test it, we use a simple way——to use other timer in the computer or a phone. The statistics may be not that accurate.

|  |  |  |
| --- | --- | --- |
| Input | Expected result | Real result |
| Pressing time =5.71s | St=0 | St=0 |
| Pressing time=7.20s | St=0 | St=0 |
| Pressing time3.55s | St700 | St=698 |

Next is the test case of ranking system.

Because we created a text file in the game folder, so we can change the statistics in the file to see whether the ranking system works well.

|  |  |  |
| --- | --- | --- |
| Input(the highest 5 numbers) | Expected result | Real result |
| 420, 242, 101,  85, 56 | 420, 242, 101,  85, 56 | 420, 242, 101,  85, 56 |
| 420, 242, 101,  85, a | error | Can not open the file |
| 420, 242, 101,  85, 1001 | 1001, 420, 242,  101, 85 | 1001, 420, 242,  101, 85 |
| 5, 2, 0, 0, 0 | 5, 2, 0, 0, 0, | 5, 2, 0, 0, 0, |

In this chart, we can see the ranking system is good enough for our program. There is a illegal input “a” in the chart, and the file can not be opened. But in fact, while playing the game, there can not exist such illegal input in the text file. If all the statistics are legal, the ranking functions will find the highest 5 numbers in the file and show us them accurately.

Here comes the testing case of grading system.

|  |  |  |
| --- | --- | --- |
| Input | Expected result | Real result |
| man.x=447 brick.x=400 brick.width=50 | man.grade=man.grade+1 | man.grade=man.grade+1 |
| man.x=425 brick.x=400 brick.width=50 | man.grade=man.grade+1 | man.grade=man.grade+1 |
| man.x=387 brick.x=400 brick.width=50 | man.grade=man.grade+3 | man.grade=man.grade+3 |
| man.x=397 brick.x=400 brick.width=50 | man.grade=man.grade+5 | man.grade=man.grade+5 |
| man.x=362 brick.x=365 brick.width=62 | man.grade=man.grade+5 | man.grade=man.grade+5 |
| man.x=328 brick.x=288 brick.width=122 | man.grade=man.grade+5 | man.grade=man.grade+5 |
| man.x=137 brick.x=400 brick.width=50 | man.survival=false | man.survival=false |
| man.x=77 brick.x=400 brick.width=50 | man.survival=false | man.survival=false |
| man.x=102 brick.x=288 brick.width=122 | man.survival=false | man.survival=false |

This chart shows the test of grading system. Because this part is a little complex, we only test the important part of it. For example, in this chart, we can see that the real result is the same as the expected result.

By now, we have tested the jump function, grading function, timer function and the ranking functions. Just as the results showed, these parts work as well as we expected.

1. **Conclusion**

By now, we have finished all the expectation. The game is very playable. We have a logo of Tang Aoqing Honors Program in Science in the welcome interface. After clicking the START button, it goes to the game mode. And while pressing the left button of players’ mouse, it will play a special sound to let the player feel the elapse of time. This makes the game more playable and player friendly. After testing and debugging, we can say that this project is accomplished as successfully as we expect.