**C PROJECT REPORT:***Smart Flappy Bird*

* **Abstract**

At the present, artificial Intelligence is a very prospective field in computer science, and it is well-known by our students. Now that we have learned C language, we come up with the idea to apply basic arithmetic of AI in our project by designing a simple game and add certain arithmetic to make the computer perform better than human beings. We choose the popular game “Flappy Bird” and Q-learning arithmetic to achieve our goal. Our main problem is to realize the ordinary game and appropriately put the arithmetic into the game. Once we accomplish this imagine, players will feel the power of artificial intelligence.

* **1.Introduction**

In recent days, we often hear about the AI (Artificial Intelligence). AI is a subject which makes the computer to simulate human thought and behavior (like studying, reasoning, thinking and planning). Including principle to realize intelligence and making computers with brain-like intelligence, so that the computer will be able to realize high-level applications.

AI is an inferior subject of computer science, and it is regarded as one of the three cutting-edge technologies (space technology, energy technology and Artificial Intelligence) in the world in 21th century. Since it has developed quickly in the past thirty years, it has been applied in many subjects and fields, and achieved fruitful progress. AI has gradually become an independent branch of computer science.

Doing research in AI requires a strong basis of mathematics and computer knowledge. Now that we have learnt C language, we can achieve it in a basic game. Therefore, we chose Flappy Bird and Q-learning arithmetic.

Q-learning is a reinforcement learning technique used in machine learning. The technique does not require a model of the environment. Q-learning can handle problems with stochastic transitions and rewards, without requiring adaptations. For any finite Markov decision process (FMDP), Q-learning eventually finds an optimal policy, in the sense that the expected value of the total reward return over all successive steps, starting from the current state, is the maximum achievable. Q-learning can identify an optimal action-selection policy for any given FMDP.

Flappy Bird is a popular game designed by DongNguyen (Vietnam) with a high degree of difficulty. So, we come up with the idea to drive the bird to play the game by itself. With the addition of Q-learning arithmetic, the score will be much more than human players. In addition, we combined the traditional mode and self-playing mode, so that players can have a better experience.

Compared to the classic game, our game focus on realizing the intelligence, and our attempt is original and challenging, while other groups only contribute the ordinary game. Since there is no source code at all, we must range the task on our own and deal with the problem with reasonable imagine. The main problem is to accomplish the basic game and add the certain arithmetic to the game, and our goal is to complete the whole game, including the normal game and self-learning game.

* **2.Group Division**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | student ID | College | Work percentage |
| Qinglin Zhang | 13170705 | Biology | 22% |
| Hongming Zhang | 12170906 | Chemistry | 25% |
| Liqin She | 12170916 | Chemistry | 25% |
| Yinghao Ge | 12170917 | Chemistry | 28% |

1. Qinglin Zhang

She takes charge of interface optimization. She managed to put pictures and background music into the game, in order to make the simple game change into the ordinary game. In details, she changes some data to print the pictures, and adds beginning and ending interface.

1. Hongming Zhang

He created the single game. In the project, he set up most of basic functions of the simple game. In addition, he built many creative functions to make the game better, such as the locomotive ground and the restart of the game. What’s more, he has provided logical thinking for some other work, such as the adjustment of the speed.

1. Liqin She

He modified the simple game to make it adaptive for inserting pictures. During the process, he was in charge of designing an accurate judgement function for the ordinary mode and solving some problems such as averting screen flicker. Additionally, he optimized the game like coordinating the movement of the barriers and bird and the interval to guarantee players a comfortable experience.

1. Yinghao Ge

He is the leader of our group to arrange deadline of finishing staged tasks and the way we did our project. To begin with, he attempted to apply Q-learning arithmetic to the simple version and he made it. Then he designed Q-learning arithmetic mode in formal version, wrote relative code to achieve the bird learns by itself and did some tiny adjustment on the basic data, normalizing the whole game and judgement to meet the requirement of Q-learning arithmetic. Finally, he integrated our project.

* **3.Analysis**

In the recent time, we heard a lot about the AI. Artificial Intelligence is a subject which focuses on how to engage the computer capacity to deal with human’s job. It searches the rule of brain activities and contributes a system which partly possesses intelligence.

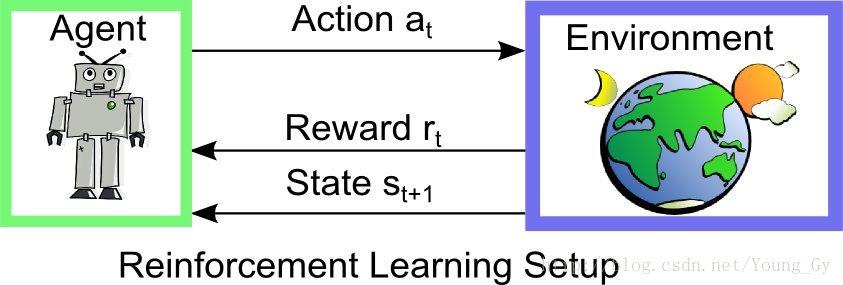
If we insert certain arithmetic into the program, we can make the computer think and behave like human beings. And computers can do many works better and more quickly than human. Since we have learnt C language, we can realize a project to illustrate the strength of Artificial Intelligence.

Flappy Bird is a well-known computer game, whose operation is extremely easy. There are only a bird and barriers on the screen. The bird descends in a certain speed, while barriers are generated in the right side and move to the opposite. If you press the “space”, the bird will immediately go up for a distance. There is a gap between two parts of barriers. Only when the bird flies over the blank, can it survive. The principle of the game is not complex, but the original game accelerates the speed of declining a lot, which rapidly increases the difficulty of game. As for human players, 10 scores is already a very good mark.

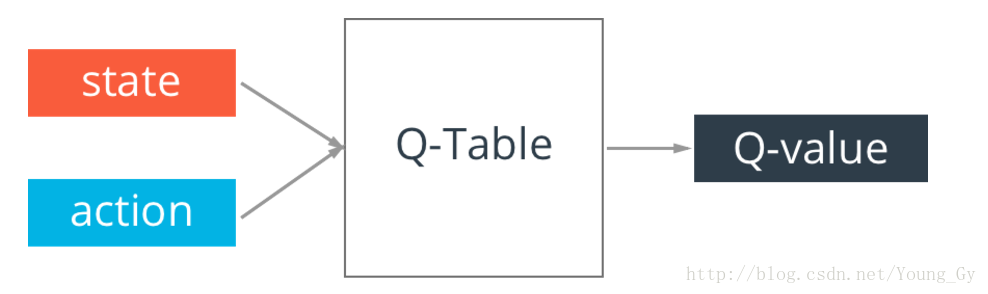
So, in our opinion, this game could be an appropriate carrier of our intent to realize artificial intelligence. Above all, the game is not so difficult for our beginners to accomplish, since it owns a very clear and easy operation, and most of our group members are quite familiar with it. In addition, it has a very high difficulty degree. If we choose the arithmetic properly, and execute it in the game successfully, there will be an obvious difference in results between normal players and computer player, for example, the score is probably much higher, and the movement of bird will change far more flexibly.

Since we have confirmed the carrier game, we need to choose a proper arithmetic. As we know, the location of bird and barrier is changing all the time, so the arithmetic must be able to deal with a large amount of data, and save them in an appropriate way. Besides, the speed of updating must be quick enough, or it will take too long to achieve the learning process. According to information we found, we preliminary choose two arithmetic, they are sarsa and Q-learning. They are almost the same, for their theories are both rewards and punishment, and it is liable to combine with the game. However, when it comes to data updating, they show an obvious difference. While the object has to take an action, Q-learning will choose the action with the biggest Q, while the sarsa chooses the same Q and update the information at last. As a result, Q-learning shows a quicker learning, so we finally choose it as the arithmetic we adopt.

The principle of Q-learning will be clarified in the following. Q-learning is virtually a kind of reinforce learning. Generally it includes two objects, agent and environment.



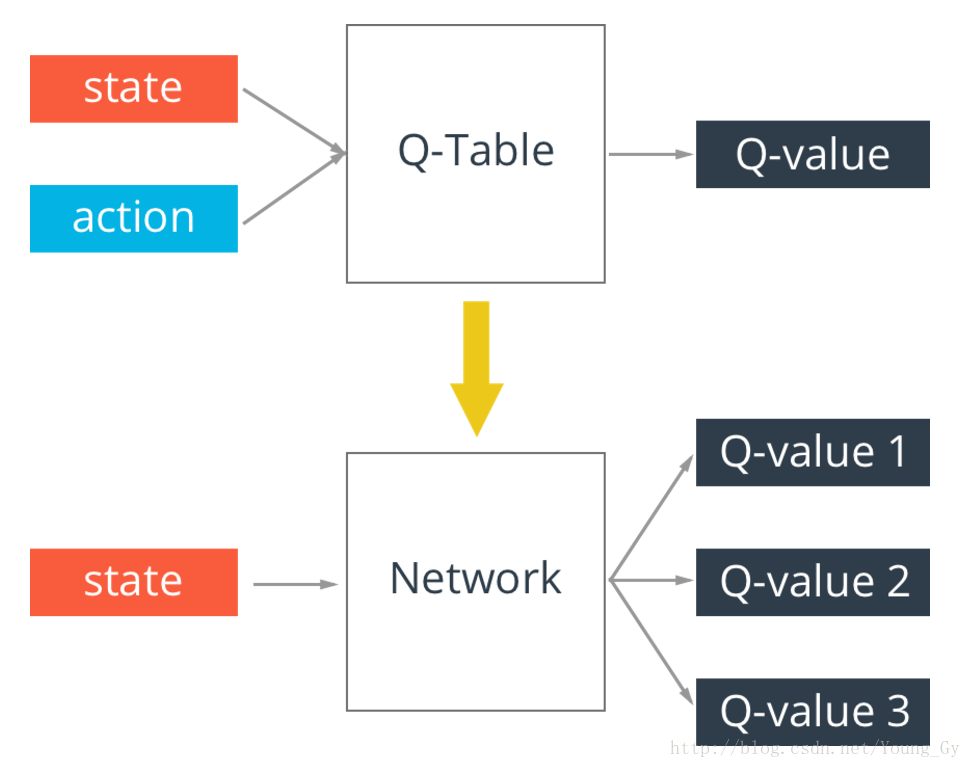
In one state of environment, the agent will adopt an action, subsequently obtain a reward, and come into the next state meanwhile. It usually has following features. First, different actions lead to different rewards. Second, the reward has a trait of delaying. Third, the reward referring to certain action is based on the present state. The core of Q-learning is Q-table. Its horizon line and vertical line refer to the value of state and action.

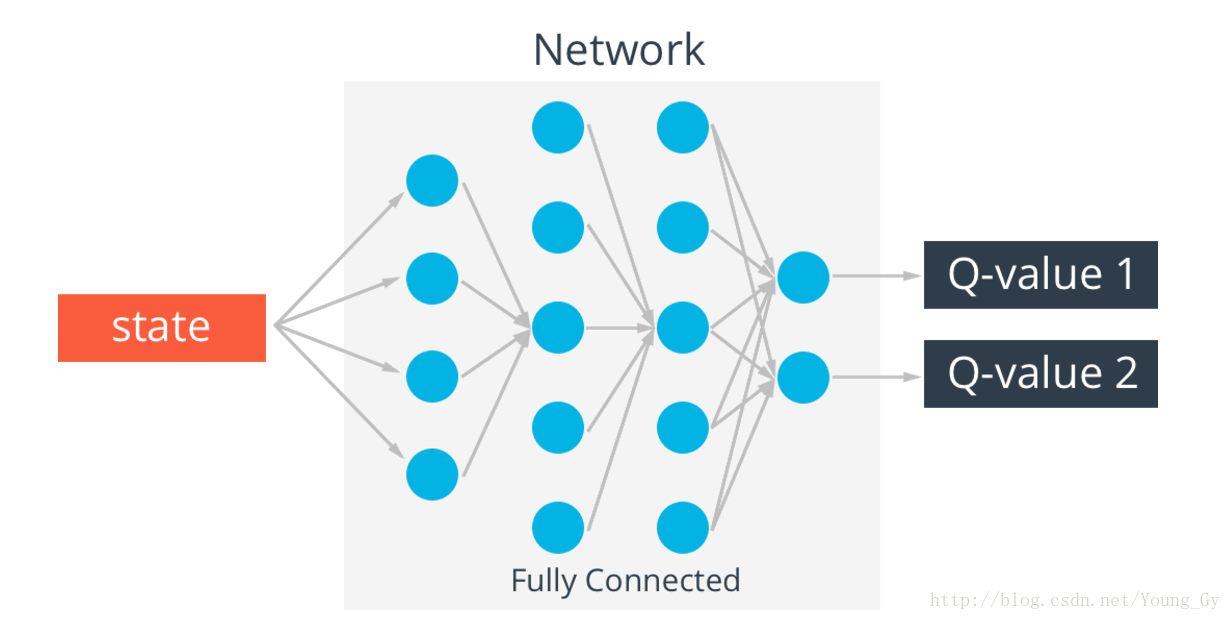


Values in Q-table estimate the goodness of present state and action. In the training process, we use Bellman Equation to update Q-table.

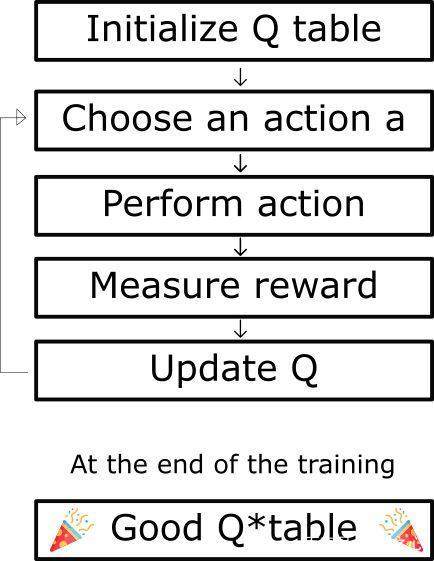
Q(s, a) = r + γ (MAX(Q(s’, a’)))

Q-table has a problem that the state in real situation can be infinite, and the Q-table will be infinite as well. The solution is to add Network. We input state, and the computer output the Q-value of different actions.





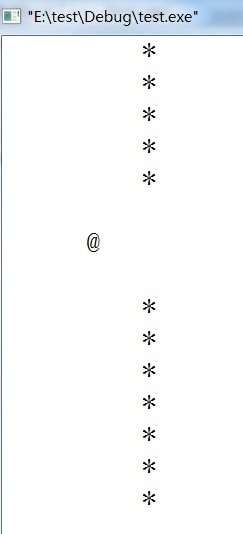
When the training has just begun, we need to add certain randomness in order to see more situations. As the training becomes deeper, we could gradually decrease the randomness. Following is the whole structure of Q-learning.



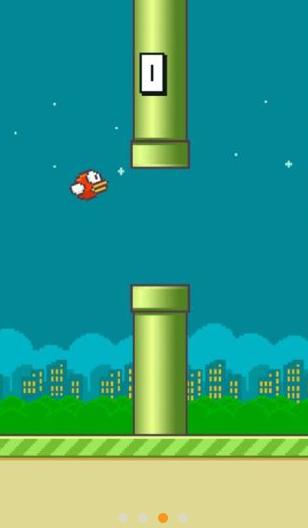
In this game, we choose the horizon distance of bird and barrier, and vertical horizon as the state. There are only two actions which can be adopted, fly up or do nothing. The method of reward could be positive when the bird is alive, or fly over the barrier, and negative when the bird dies.

When the bird updates the state, the present profit(r) and memory profit(Q(s’, a’)) will be take into consideration. We could find that the bigger γ is, the more memory profit the bird will pay attention to.

Above is all about Q-learning and its apply. The following is about the development of basic game. The first part is to exploit a symbol-like game. First, we suppose to exploit a bird which declines easily, and able to go up when we press the “space”. Next, we need to print the bird and the barrier at the same time. We need to add several variables to achieve the goal. Then the barriers are made to move from the right side to the left side. Following is to judge if the bird has passes the barriers by flying over the blank, and add a variable to record the score. Finally, after the barriers disappears in the left side, they should appear in the right again, which is achieved with a circulation. In addition, we will use random variable to realize the different location of the blank.



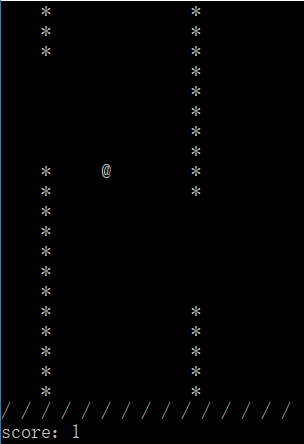
The final part is to do the interface optimization. It is based on the last part. The material should be collected before using. First of all, the picture of background, birds, and barriers should be added to the screen. The symbol can be deleted then. What’s more, since the score is simple figure in the first version but pictures in the basic game, the score recording mode should be rewritten. Finally, the music, beginning interface and ending interface will be added. If possible, we would like to add creative animation effect.



* **4.Design**

1. Simple game creation

We want to realize a symbol-designed Flappy Bird, which is called the ‘ordinary mode’ in the following passage. In the ordinary mode, the bird looks in the shape ‘@’. And the barriers made up of ‘\*’ have only one line. To make sure that the window of the game has a border at the bottom, we used ‘/’ to create the ground. The computer prints all the appointed symbols to take on a simple interface. As a result, the background is black and the abstract images are white by contrast. (4-1)

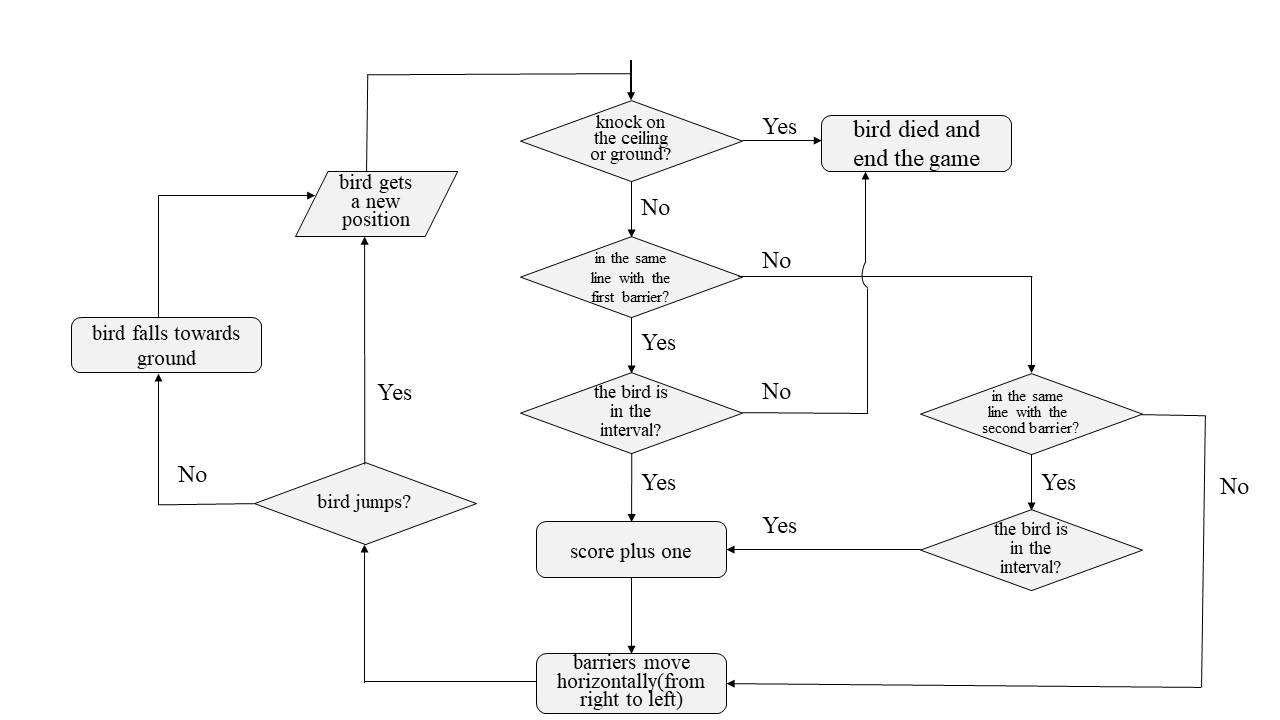


(4-1)

As for the general design for the ordinary mode, we designed a big cycle. The cycle mainly has three fundamental parts: the judgment, the score record and the movement. When the game operates, it comes to the judgment first. And the results of the judgment determine the score record. At the same time, the results have a direct effect on the movement. Once the position of the bird changed, it goes to the head of the cycle——the judgment again.(4-2)

ⅰ) Judgment: First, it will test whether the bird knocks on the ceiling or ground. If the bird knocks on the ceiling or the ground, it means the bird died and the game will be brought to an end. If the bird is alive after the first judgment, it will come to the second judgment on condition that the bird is in the same line with the barrier. If so, the computer will test whether or not the bird is in the interval of the barrier.

ⅱ) Score record: Scores will plus one if the bird passed one barrier successfully. Otherwise the bird died and the game ends. And the scoreboard keeps refreshing.

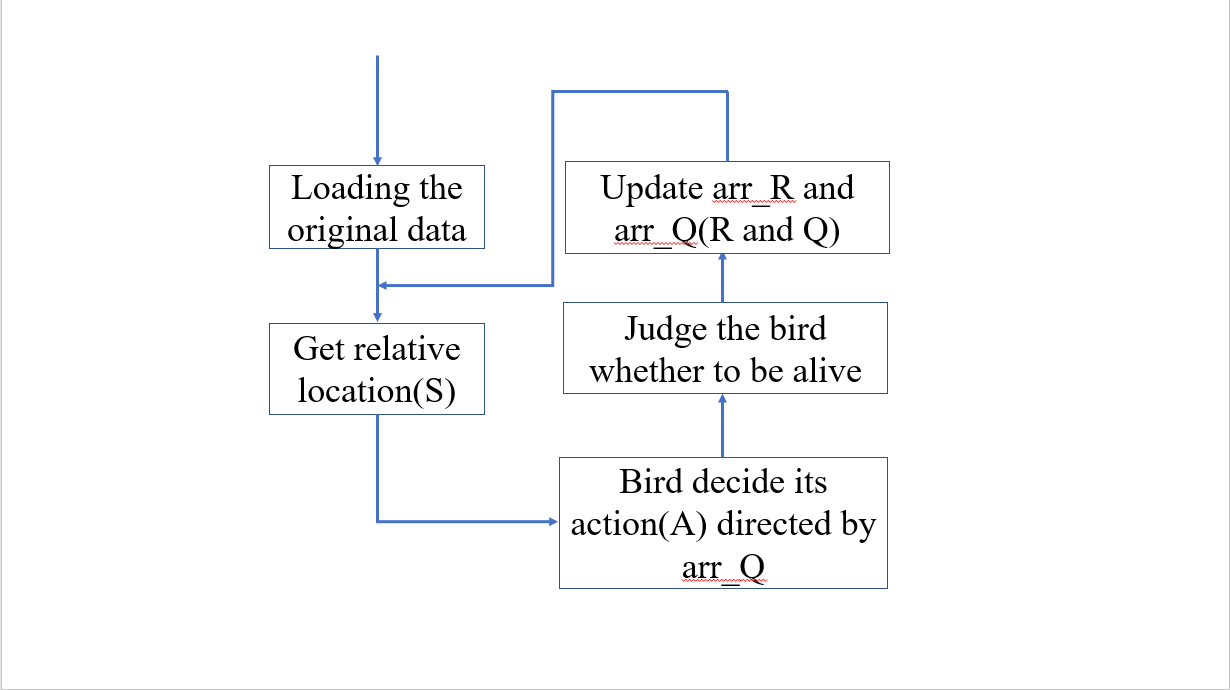
ⅲ) Movement: Judgment and score record convey critical information to movement. If the bird gets one score or haven’t met barriers, barriers will move horizontally from right to left. Also, we still need to make the bird move if we want. The bird can only move vertically, which can be called jump. If it jumps, it will move towards the ceiling. Otherwise, it falls towards the ground.

1. Q-learning arithmetic

To begin with, Q-learning arithmetic has four elements, just state(S), action (A), reward(R) and Q. The method it works is that we design the reward firstly, then every time the machine gets to a state, it will have a Q’s number in terms of the central equation and the reward number, and this process will be repeated over and over again. Finally, the numbers in Q will be steady and depending on the Q, machine can get the most efficient and correct way to finish its word.

The problem statement of Q-learning arithmetic applied to Flappy bird is to make sure the learning cycle. Firstly, computer needs to know the state of the bird. According to the state, the bird will get an action from the computer. In this way, the bird can learn and fly at the same time. Now, the bird finished one step, so the computer will get a reward number in terms of bird’s life-state, which will be saved in array R. After all of these, computer will deal with the array Q. In term of the equation of Q-learning arithmetic, there will be a number inputted to array Q. Then it will go back to get the new state. As we can see, there is no way to get out from this cycle, so the bird can learn constantly. This structure will be testing and debugging by the bird’s learning process and its whole performance.

Content of the above is the general structure of Q-learning arithmetic in Flappy bird. So now we want to use a flowchart to show this structure clearly.



* **5.Implementation**

1. Simple game creation

I. The establishment of bars

As we all knows, there are endless bars whose gap are in random position in “Flappy bird”. This part is talking about how we implement it.

First, we need lots of bars which can move left automatically. As a matter of fact, it’s easy to control its move. We can just use a while cyclic sentence to do that. But it’s really difficult to print infinite bars. In view of this and another fact that there are only two bars appearing at the same time, so we set up a function to make the bar which reach the left side of the interface return to the right one. The code is shown in figure 1.

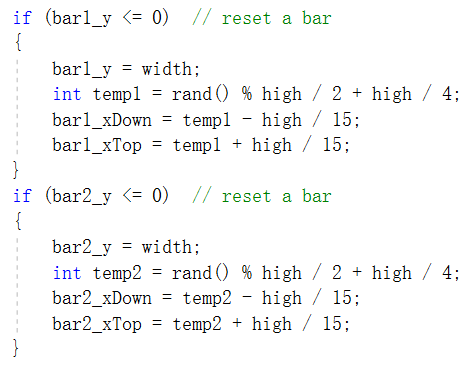


Figure 1

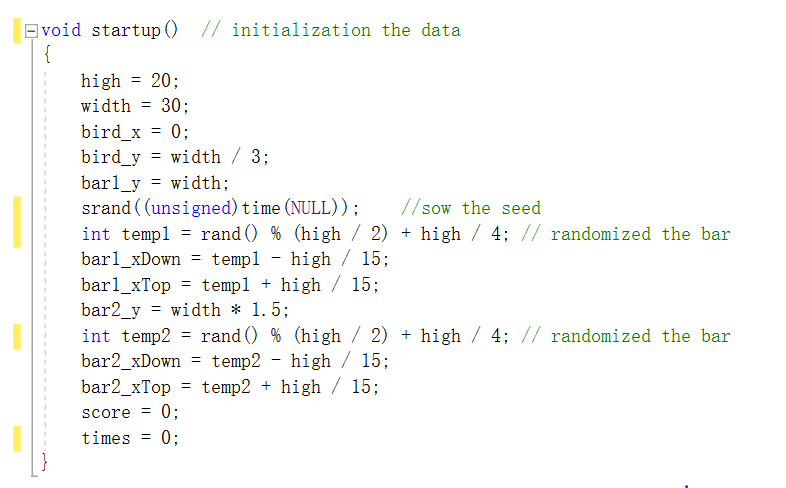
Besides that, we need make the gap be in the random position. We use random number to do it, just like figure 2.

Figure 2

What’s more, it is noteworthy that sometimes the bird will be bound to be dead because of the numerical value of many variables such as the speed of its falling and the distance between two bars. In these situations, the gap of the former is in a high position, while the gap of the latter is in a low position, so that the bird will be dead even if we don’t press space bar to rise it.

To avoid this situation, we limit the scope of “temp” to prevent the gap being too high or too low. In the figure 2, we make “temp” between one fourth “high” to three fourth “high” (“high” means the height of the game interface). Through our tests, it’s a great numerical value for the game.

II. Initial interface

It’s no doubt that the game needs an initial interface. This part will do the job that introducing the implementation of the initial interface and its mechanism.

Above all, let’s have a look at the code and the result.

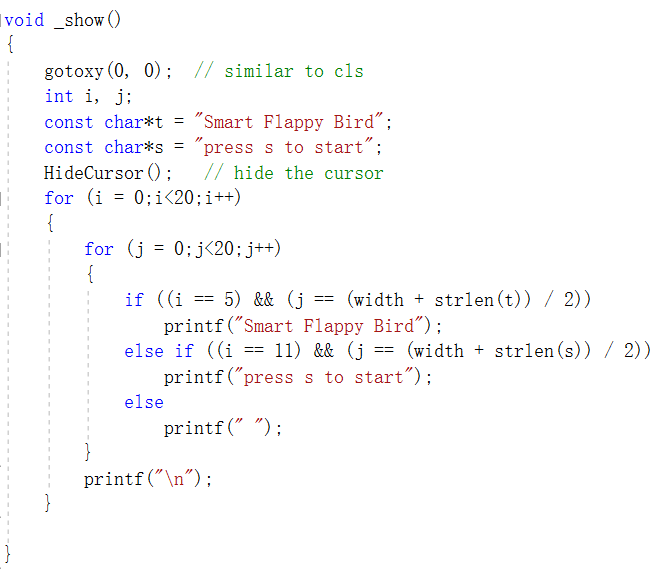


Figure 3

Figure 4

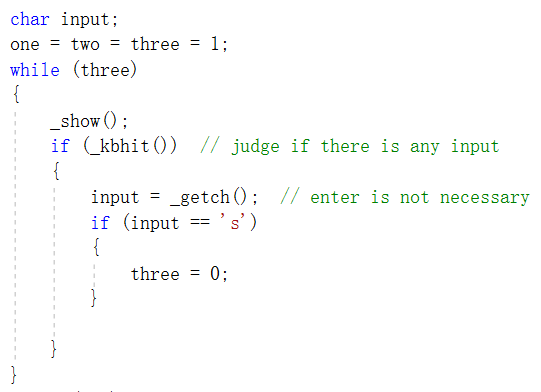


Figure 4



Figure 5

As we can see, the print of the initial interface is ordinary, but we use strlen() specially. It can help us centralize the words.

As for the switch of the interfaces, we choose while cyclic sentence. In fact, the initial interface will keep being printed until we press “s” to jump out of the loop and go on.

III. Locomotive ground

As we all know, the ground in Flappy bird is locomotive and we realize it mainly by the white and green horizontal stripe which looks like conveyor belt. We also want to do it in our simple game.

Known from the first implementation, endless characters is really hard to come true. And we can’t use the method in the first implementation because there are many “/” in the horizontal stripe and it’s too intricate to reset these “/”. Under the circumstances, we came up with a creative way to solve the problem.

Consider that “/” and spacing alternate in the belt, which means that there are only two situations here: “/” is the first one, or spacing is the first one, we decide use the odevity to solve it.

As you know, our game is a big while cyclic sentence in fact. So we define a value “times” to count the number of iterations. Every time we go through the loop, it adds 1 to “times”. So when “times” is an odd number, one of “/” and spacing will be printed first. When “times” is an even number, the other will be printed first. The core code is shown in figure 6.

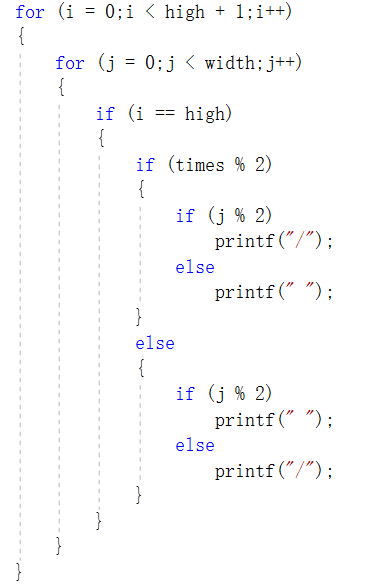


Figure 6

IV. Restart the game

As you know, both our program and game are while cyclic sentences, just like the code in figure 7.

We use “one” and “two” to control the program and the game. As the code in figure 4 shows, both they are 1 at first. When the bird dies in the game, “two” will be changed to 0. That means game over. Now if we press “r”, which means restart, we can make “two” become 1 again to restart the game. But if we press other keys, “one” will be changed to 0, and that will finish the program and exit it.

What’s more, considered that we need to use spacing bar to play the game, we set that press spacing bar will do nothing to avoid the accidental press.

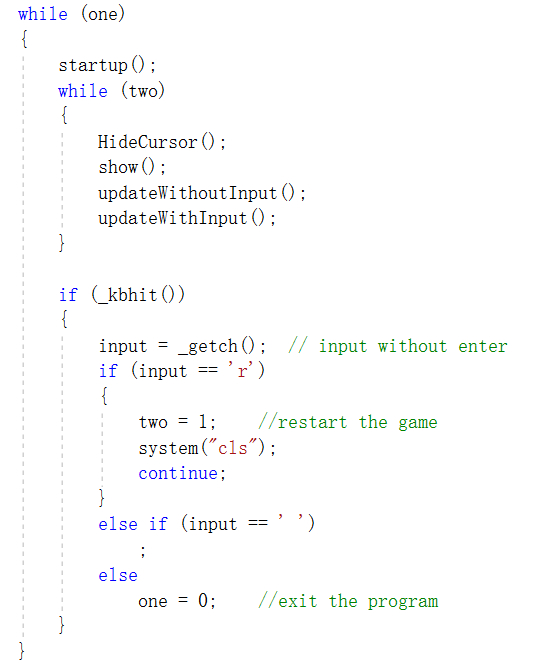


Figure 7

V. Speed regulation

In Q-learning mode, it often cost much time for the bird to learn well and get a high score. To reduce the time, we need to make it speed up. But it’s not appropriate to start the game with a high speed. So we create a function to adjust the speed.

One thing we need to know is that it’s very fast for computer to execute these simple statements. It spends most time on Sleep function. In the other word, Sleep function is the rate-determine step, so we can control the speed of the game by control the parameter of Sleep function, just like figure 8.

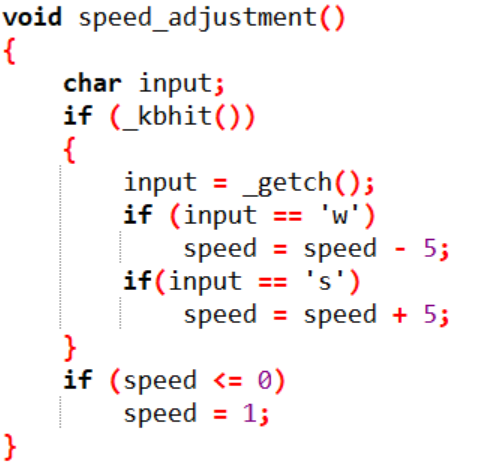


Figure 8

In the program, we define a value named “speed” to be the parameter of Sleep function, and we can change it by press “w” or “s”. In addition, for the parameter of Sleep function must be a positive number, we make speed become 1 when it’s less than zero.

1. Q-learning arithmetic

In order to achieve it, we need to define the four elements in Q-learning arithmetic.

State(S): In Flappy bird, what can influence the game’s result should be expressed by state, so using the relative location between the bird and pillars can meet this requirement. Because when the relative location is in a certain range, the bird can pass the gap safely, I define the relative location as state. Our game uses x and y to express the high and width respectively. In this way the delta x and delta y can express the relative location.

Action(A): This element is easy to define, because the bird only has two choices. Action only have two choices, they are flying and falling. However, how the bird can fly by itself? No matter where the bird is, the bird always has its relative location and computer can find two values in array Q. One represents flying, the other represents falling. Therefore, the bird can choose its action in terms of the two values.

Reward(R): According to the definition of state and action, the array R should have three variables to confirm one number in array R (The three variables are delta x delta y and action), so array R is a three-dimensional array. Although I ensure the type of this array, I still need to rule how to give the bird reword. This is the conversion I make to apply this arithmetic better. Because it is too difficult to give all numbers which should be in array R, I want make the array R updated by the bird’s state. In this game, our goal is to make the bird fly longer. Therefore, the first important thing is to live, the second one is to pass the pillars, but obviously the second one is more important. Finally I designed that if the bird is alive, the number is 5. If the bird passes a pillar, the number is 250 and if the bird is dead, the number is -5000. In this way, the bird can get the relative reward value to know how to decide its action. In the definition of R, we finish to apply Q-learning arithmetic to Flappy bird basically.

Q: This element is the same order array as R, so Q is also a three-dimensional array. The difference between R and Q is the way to get the number in array.

After solving the basic definition about the four elements, when we really using this to the game directly, we will find this is difficult to judge if the bird passed a pillar. To make it possible and convenient, I adjust most of data in this game to make this game several units, just using a 10 units’ width and 22 units’ height as a basic unit for the bird. At the same time, other relative data are changed to be multiple of 10 or 22 (Figure 9). It really increases the speed of bird’s learning, because the number of different state is reduced.

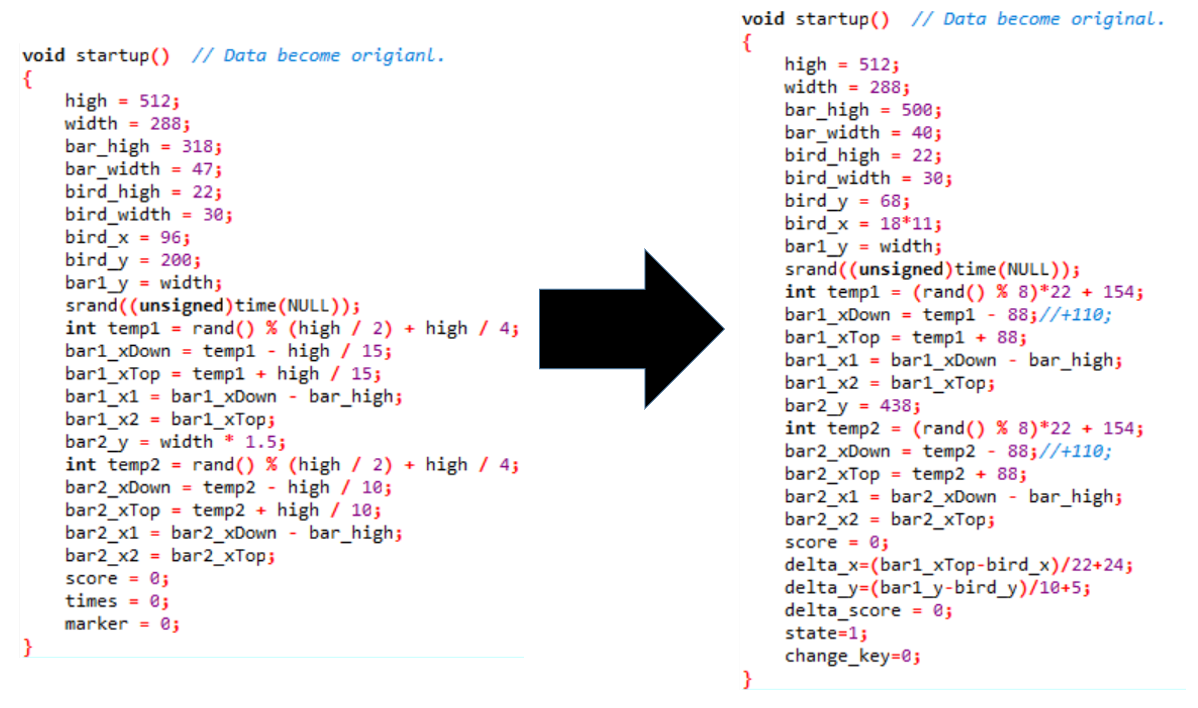


Figure 9

After changing these, I also need to adjust the way to judge whether the bird can get score. Only when the delta y is an ensured value, the bird will get score (Figure 10). In this way, it’s simple to define which location the bird will get its score and it’s convenient to get the change of score.

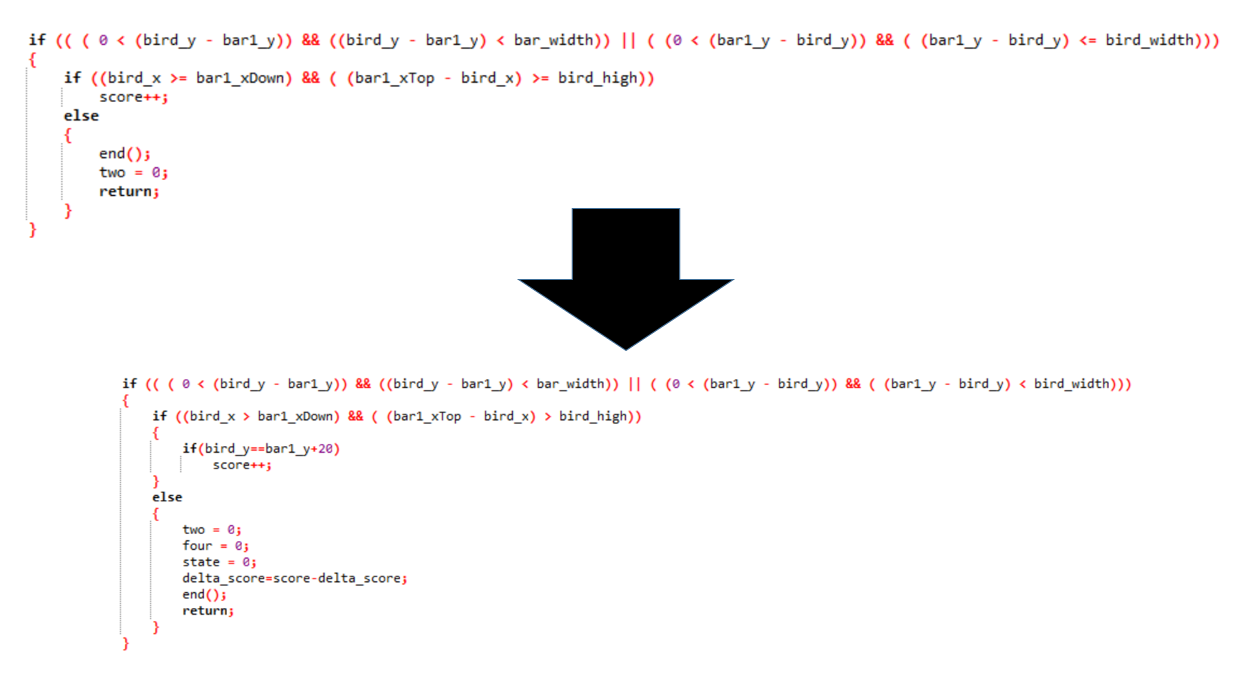


Figure 10

In this game, the screen always has two pillars, which are bar1 and bar2 respectively. When one pillar is gone, it will appear at the most proper location of the interface. Therefore, it will make disorder on confirming the relevant location. Thereby this problem needs to deal with. I wrote a series of function to change this disorder (Figure 11). When the bird passed a pillar completely, state will be the relative location between bird and pillar. In this way, computer can get right information to update R and Q.

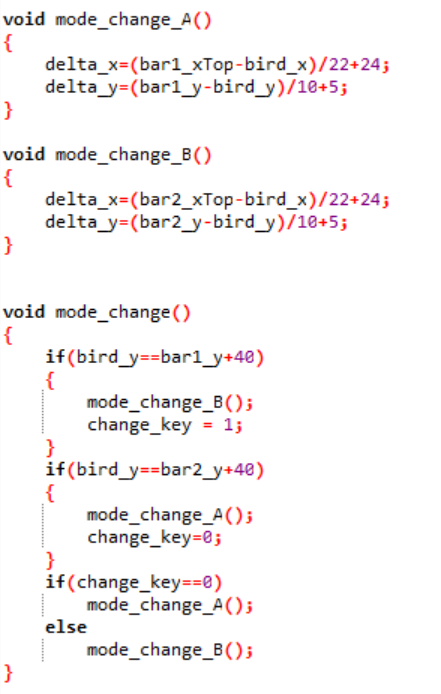


Figure 11

Moreover, there are several values used to ensure computer can get the relative information. For instance, state is used to inform R and Q whether the bird is alive and delta\_score can express whether the bird go through a gap. When four elements are confirmed, there is no large problem to use it.

Now, Q-learning arithmetic can be used in this game and correct information about state and action can be inputted to this array. Generally speaking, this arithmetic can work for our learning goal.

1. Interface optimization

There are two basic aspects in interface optimization. Firstly, we need to plug in pictures and music. Then, some of the pictures must show an animation effect by quickly switch the pictures.

Above all, we should put pictures into the game, so we collect pictures at the beginning. Some materials are downloaded from the Internet, since a few people have tried to realize the game and they remain their resource. Some pictures are made by hand, and a few pictures have been adjusted to be more suitable to the interface at the same time.

At the beginning, we added necessary header file.

#include <graphics.h>

#include <windows.h>

Easy\_X is reported to be widely used in interface design, so we downloaded it for preparation. In order to plug in pictures, we learned two basic functions, they are loadimage() and putimage(), and two sentences are showed as examples in the following. Loadimage() loads the picture, and putimage() prints the picture on certain location.

loadimage(&background, "F:\\game\\background.jpg");

putimage(0,0,&background);

In the function loadimage(), the symbol “&” is used to find the address of file, and the word “background” is the name of it. It must be claimed as a global variable. The detailed address is written subsequently. We must be aware that the whole address could not exceed the limitation of thirty-six characters, or the computer will not be able to find the file. Besides, the format of file is limited. Only bmp, jpg,

In the function putimage(), we should give the point to start printing the picture, so the first two figures is exactly the coordinate point, and the original point is at the top left corner. The horizontal coordinates is in front of the vertical coordinates. The address searching is behind.

When the pictures come to use, we find that some pictures has a white margin, which badly influences the overall effect. As a result, some orders are use to deal with it.

putimage(180,50,&score1[0],NOTSRCERASE);

putimage(180,50,&score2[0],SRCINVERT);

First we invert the color of certain pictures, and save it for use. NOTSRCERASE and SRCINVERT are two kinds of Bilbit function, NOTSRCERASE is used to combine the color of source picture and target rectangle and invert the color, while SRCINVERT only combine the color of two areas. They are put behind the address searching sentence.

The addition of music is almost like the pictures.

mciSendString("open \"F:\\game\\background.mp3\" alias music ", NULL, 0, NULL);

mciSendString("play music", NULL, 0, NULL);

The function mciSendString() needs a header file.

#include <mmsystem.h>

MciSendString can be used to add background music. In the first sentence the audio file is loading, and the detailed address is required. Generally, we don’t need to return anything, so the value is NULL. And the next sentence is to play the music.

Then, we open a window to execute the file.

initgraph(288,512);

It is used to open a window with certain size. The figures also refer to a coordinate point.

In the process of game, the mouse is used to enter either mode of the game. If we press the right button, it refer to the Q-learning mode. If we press the left button, we will enter the ordinary mode.

MOUSEMSG m;

while(true)

{

m=GetMouseMsg();

switch(m.uMsg)

{

case WM\_LBUTTONDOWN:

one=two=1;

Sleep(1000);

return;

break;

case WM\_RBUTTONDOWN:

three=four=1;

Sleep(1000);

return;

break;

}

}

Here the MOUSEMSG is a variable, and GetMouseMsg() is a function to obtain a piece of mouse message. Then, we designed different action when we put different buttons.

Besides, the print orders are also used to print a button.

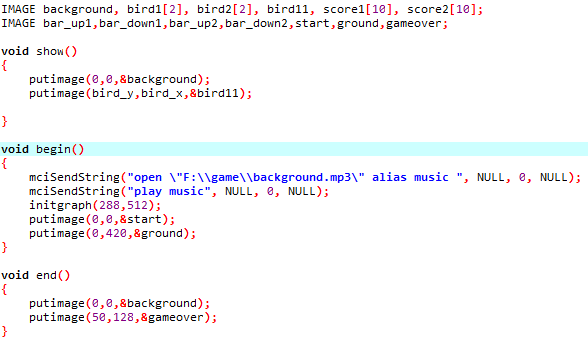
setfillcolor(WHITE);

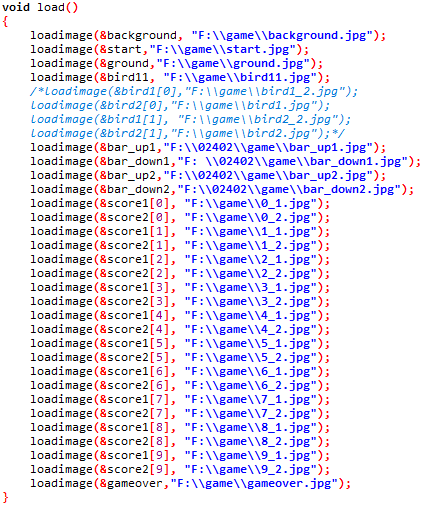
solidrectangle(36,354,108,402);

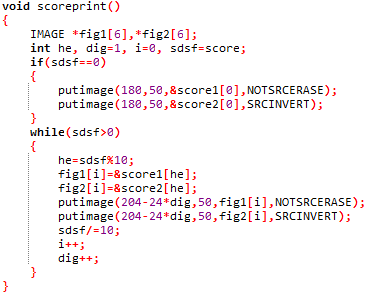
The first sentence is to set the color of graph. Beside red, green, blue and so on, we can also use mixed color by adjusting the value. The following sentence is to create certain form shape, including rectangle, triangle, circle and so on. The parameter of them is required.

Since the name of game is “Flappy Bird”, I come up with the idea to make the bird be able to flap its wing. However, to achieve my goal, the pictures of bird’s movement must be switched all the time. In the end, it results in the splash screen, and it has a bad influence on player’s experience. We failed to solute this problem so this part is deleted at last.

In the last part, I’d like to show all code I wrote (Figure 12).



****

****

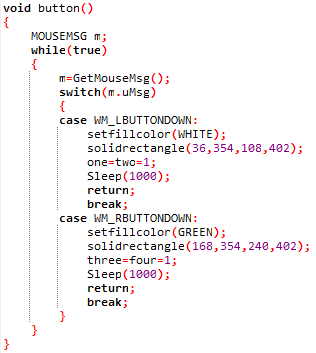
****

Figure 12

* **6.Testing&Debuging**

1. Simple game creation
2. The movement of the barriers

Function name: ‘updateWithoutInput()’

This function controls the movement of the barriers. Since it has nothing to do with what the player inputs, we need to detect its changes. And the ‘bar1\_y’ or ‘bar2\_y’ can reflect its changes. Here we take ‘bar1\_y’ for example.

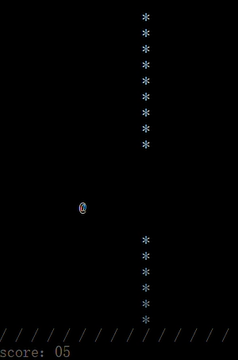
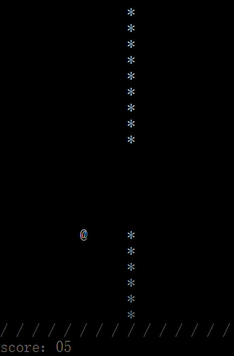
|  |  |  |
| --- | --- | --- |
| Action (Input) | Expected Result | Real Result |
| Game operates | bar1\_y - 1 | bar1\_y - 1 |
| Game ends | No change | No change |

1. The movement of the bird

Function name: ‘updateWithInput()’

This function controls the position of the bird. And the position of the bird can not only be set by the computer but also the player. The ‘bird\_x’ can reflect the changes.

|  |  |  |
| --- | --- | --- |
| Action (Input) | Expected Result | Real Result |
| Press spacebar | bird\_x - 2 | bird\_x - 2 |
| No input | bird\_x + 1 | bird\_x + 1 |
| Other input | No changes | No changes |



No input

(4-3)

(3)Score returns to zero after restarting the game

Functions name: ‘startup()’ and ‘while ()’

We designed the function ‘while ()’ to get the game restarted without exiting the procedure. ‘while (one)’ can start the procedure and ‘while (two)’ can start the game. So, ‘while (two)’ and ‘startup()’ are nested in ‘while (one)’. And every time the procedure is started, all the data will be reset, which is called initialization. What the score board shows reflects the changes.

|  |  |  |
| --- | --- | --- |
| Action (Input) | Expected Result | Real Result |
| Press ‘s’ | 0 | 0 |
| Press ‘r’ | Score turns to 0 | Score turns to 0 |
| Other input | No changes | No changes |

1. Q-learning arithmetic

In our Q-learning arithmetic mode, when I finished the code to run this mode, I found that the bird looked strange. It couldn’t do well to learn how to fly, which really made us upset. After several times to change, test and debug, I found out the main point wasn’t the setting of R, the method to fly or the changing location, but the state inputted to the array R and array Q.

According to the definition about four elements, we can see that for a cycle, there will be change of state. In the simple version, if I use the state after action, the bird can learn well. However, in the formal version, it can’t work as before.

|  |  |  |
| --- | --- | --- |
| The State Inputted Into Array | Expected Result | Real Result |
| Using state after action | The bird can learn well like the one in simple version. After enough time, the bird won’t die. | The bird only can learn little and the highest score it can get is about 40. Besides, the bird always dies at the first pillars. If there is two pillars’ gap that have long distance, the bird couldn’t learn how to pass it. |
| Using state before action | The bird will perform better than before. At least, it won’t look stupid and really can learn how to deal with it. | In this time, the bird really knows how to learn and fly. Even if it still dies easily at the first pillar, it can learn to deal with the hard situation. According to this performance, the bird has the capability to learn. After several minutes, the bird can pass more than hundred pillars. |
| Using state made from delta x after the next action and delta y before action. | Because this kind of state is a mistake of writing code, it won’t do well, and its result is unpredictable. | The bird really learns to fly. In this way, it won’t die. However, it only wants to find the right way to knock into the pillar. It can pass the first pillar or the second one, but it can’t repeat it again. |

Although the bird knows how to fly, it still waste time at the first pillar. Owing to the distance between the bird original location and the first pillar, the bird must die many times to learn how to get a better location to pass it. Therefore, I try to add some different ways to make the birds fly.

|  |  |  |
| --- | --- | --- |
| The Way Bird Decides Its Action | Excepted Result | Real Result |
| Action is decided only by array Q. | The bird will learn how to fly, but at the beginning, the bird must die. | The bird really always died at the beginning, but it still can learn. |
| Action is decided by array Q.  There is 1/1000 possibility to make the bird fly randomly. | The bird will learn faster than before, but it also brings some additional dangerous to bird. | There are no obvious differences. The bird still learnt after several times’ death. However, it can make a little trouble to train it to learn more. |
| Action is decided by array Q.  There is 1/1000 possibility to make the bird fly randomly.  If the values in array Q both are 0, the bird will stay at a steady height. | The bird will fly and fall one by one, so comparing with the last method, the bird will have more opportunities to pass the first pillar without death. | The bird really saves the time to pass the first pillar. In this way, the bird will get the useful value earlier in array Q. Adding the random flying, the bird learns more efficient than before. |

To save more time, we came up with an idea to adjust Sleep(). When we press ‘w’ the value in () will be reduced. If we press ‘s’, the value in the bracket will be increased. So we also test this part.

|  |  |  |
| --- | --- | --- |
| In Q-mode, The key pressed | Excepted Result | Real Result |
| Press ‘w’ in several times | The interface will go more and more fast. | The interface really goes faster. However, if I press too many times, the interface will stop, because the value in the bracket become zero or negative number. |
| Press ‘s’ in several times | The interface will go more and more slowly. | The interface really goes slowly. |
| Press other keys | There will be no change for our game. | No influence on the interface. |

These are the content tested and debugged for Q-learning mode. In the process to apply this arithmetic, there are too many attempts and countless mistakes. However, all of the mistakes and problems have been dealt with. Therefore, no matter how many mistakes and barriers you face, trust debugging and you will solve all of them.

* **7.Result & Conclusion**

Now we have achieved all our imagine. We have developed the basic game in the first period of time. Then, Q-learning arithmetic has been applied in the program. It takes a long time to find matched data and detailed data has been adjusted and tested all the time. After a period, we find the proper data for the arithmetic, and we make these two situations be two modes of the game. In the Q-learning mode, the bird learns how to fly over the barriers at an amazing speed. We accelerate the speed of game and the bird achieved about three hundred scores after two minutes. After three hours, the bird even flew over thirty thousand barriers. In comparison of human players, the learning of computer makes great progress. It implies that the reinforce learning can be easily used in the computer, and it has infinite potential to work better than human. It can even replace a part of brain activity. We combined the ordinary game and Q-learning game together, as a result, players can easily feel the power of learning arithmetic, as well as enjoy the fascinating game.