# C Project Report: *BREAK THE BRICks*

1. **Abstract**

Our C program is a game called bricks breaking. In the program, we aimed at creating a game that the players can press “A” and “D” to control the board moving to catch the ball. If the player didn’t catch the ball, you will lose the game. When the player breaks all the bricks,the game will be over. And you will win the game.

Sometimes, you can encounter some tools in the game, not all the tools are helpful to the player. Sometimes,the tool will make your board shorter or accelerate the speed of the ball which makes your game harder to continue.

## Introduction/Problem Statement

Our project is about the game called the bricks breaking. The rules of the game is described on the part “Abstract”.

Compared to other game which is similar to our game, the adventure of our project is that we have made a lot of innovation such as tools and a new model that the player can make the maps by themselves.

About the tools that appear in the game, they have different functions, one of the functions is change the length of the board, the other function is that change the speed of the ball. When you get the tool, you will get board longer or shorter. Sometimes, when you get the tool, you will get ball faster or slower.

Besides,we created a new model that the player can make their own map. The player should input two numbers, one number is x coordinate and the other number is y coordinate. When you feel the map is perfect, then you can end making the map and use the map to continue your game. The new model will give the player more fun and enjoyment.

The requirements of the C project is that we should write a function that making maps, making board and ball moving and so on.And finally, we come out of the solution of these requirements.And the solution will be talked in the part “Implementation”.

## **Group Division**

### Member1\_

Complete the functions of the interface of the C program and making a new model that the player can make the maps of their own .

### Member2\_

Complete the functions of the maps which include 5 maps such as the and doing the combination of the functions.

### Member3\_

Complete the functions of the board and ball moving,and realize the function

of the rebound of the ball and when the player can’t get the ball, game will be over.

### Member4\_

### Complete the functions of the tool of the game and when the player get the tool ,the change of the board and the ball will appear.

## **Analysis**

## First,I will explain the solutions’ constraints of our C program.

**Solutions’ constraints:**

First of all, about the ordinary model, the player can only press “A” and “D” to keep the board moving and the other buttons can’t work.

About the new model, the player can only press the numbers between zero and nine, the other numbers can’t work.

Secondly, the player should get the ball all the time, if the player can’t get the ball, the game will be over.

Thirdly, if the player can break all the bricks, game will be over and the player will win the game.

Fourthly, the map should be the same size as the other maps to make the game more clear.

Finally, the new model should give an output after the player gives the inputs that are legal and if all the inputs are given, the process will be over and the map will be used to continue the next part of the game.

Basically, we need several functions to write the C program.

**Functions we need:**

1. We need a function about the map making. The map should include the bricks,the board and the ball.
2. We need a function that can make the board and ball moving.When the player input “A” and “D”, the board will move.
3. We need a function that can make the game be over.When the player can’t get the ball or the player can break all the bricks ,the game will be over.The only difference is that you win or lose.
4. We need a function that can make tools which have a lot of functions such as change the length of the board or the
5. We need a function that can make the new model that the player can make their own maps.

Next part is our detailed analysisabout the functions we need.

**Detailed analysis:**

About the **first function** that makes maps which include the bricks, the board and the ball. Firstly, we tried to use the function “printf” to make the bricks and the broad and the ball. But as time goes, we find that the way is too simple and crude. So we used the Easy-X function to make the bricks, the broad and the ball seems more beautiful than the direct ways. About the bricks making, the “for” cyclic sentence are used for many times to simply the process of the bricks making. About the board and ball making,we simply plan to use the Easy-X function to make such input.The way is very easy.

About the **second function** that the board and ball moving.The function is combined by several of functions. The movement of the board is very easily achieved by the changes of the coordinates of the board. About the movement of the ball. Firstly, we tried to use the function that can directly achieve the rebound of the balls. But later, we found that there is no such functions. Next, we find that the movement of the ball is closely related to the removal of bricks. So we plan to make the function that if the ball touches the upper wall, the left wall and the right wall, its direction will be changed. So from the theory part, we has made the movement of the ball.

About the **third function** of the judgement of the game over. The function is much easier than the two functions mentioned above. When the ball's center is below the board, we end the game and give you a pop-up window, which says “you lose!” through the MessageBox. When all the bricks were destroyed, we also let the game end and pop up a window ,saying “you win!”

About the **fourth function** of the tools of the game which can change the movement of the board and the ball. We firstly define three cases in the main function to make three kinds of the running speed of loops. When the prop changes the running speed of the ball, we can realize it by changing the kind of speed of recyle.The length of the board is expressed by parameters. When the length of the board is changed by the prop, the effect can be achieved by changing the value of parameters.

About the **last function** of the new model that the player can make their own maps. The function was improved on the previous function. When the player input two numbers, it will be stored.And we use the “if” sentence to judge whether to print the bricks. And the function will be easier using it. And when the player input all the numbers, the game will start and the player can enjoy playing their own maps.

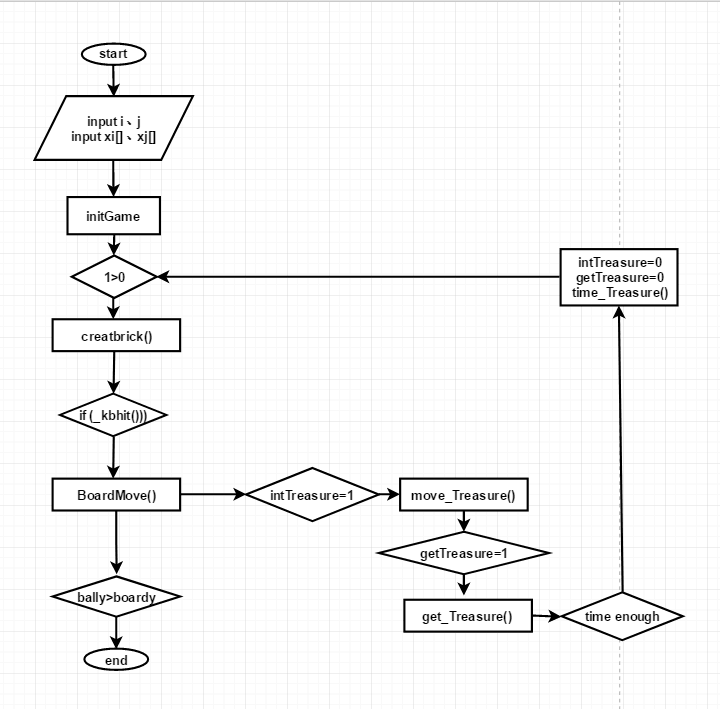
After considering all the functions we need, we can easily make all the functions in a very simple way. Finally, we need a function to make the interface when the game begins and the game ends. The function use the function “gotoxy” to define where to output the words we want to say and the game help we want the players know.

Planing all the functions, next job is to combine all the functions into together. In this part, we tried to combine the functions follow by follow. But later we find that there are too many global variables in the C program project. So we changed another way of doing this. As for how to integrate each single map, we mainly use the switch function to divide different maps into different cases. in order to make the ball motion function correspond to each map, we insert each ball motion function into the brick making function and transfer the brick coordinates into the ball motion function in the form of parameters. We combined all the functions into together.

## From the things I mentioned above, we can analyse the whole process of designing the game easily, from the beginning to the ending.

## design

### Main



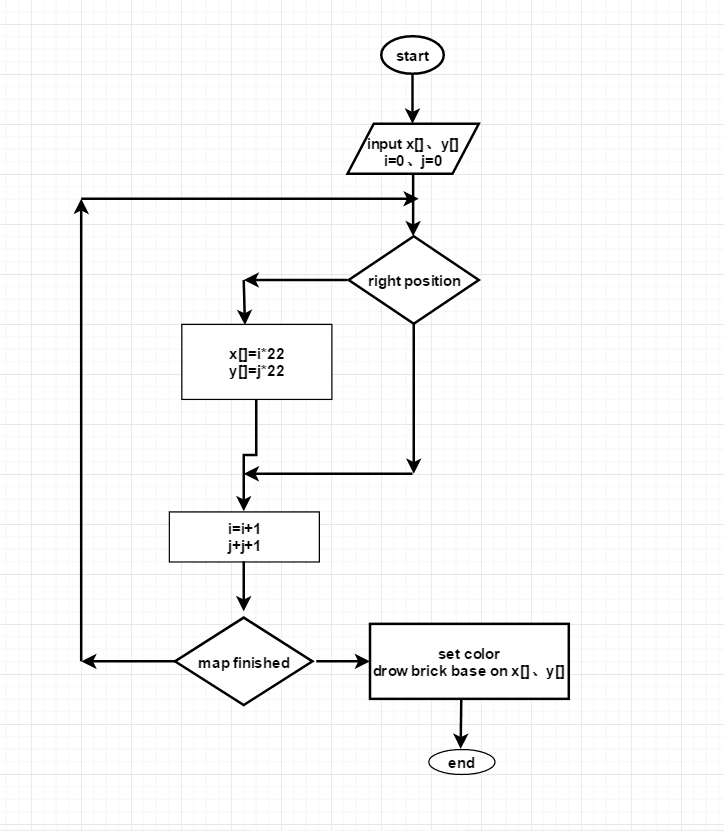
In a cycle, firstly draw the map and if there is any input on the keyboard the both board and the ball will move, the game begin. If the ball touch the brick, the brick will disappear, and it maybe generate props. If generate props, intTreasure = 1 and running move\_Treasure (). If board caught the props, getTreasure = 1 , intTreasure = 0 and running get\_Treasure () to make effection of props. After a some time, the effection disappeares, getTreasure = 0 and running time\_Treasure props () to make effection disappeared. And the program continues do until all the bricks are broken or the board doesn't catch the ball,the game will be over.

### Ballmove

### 

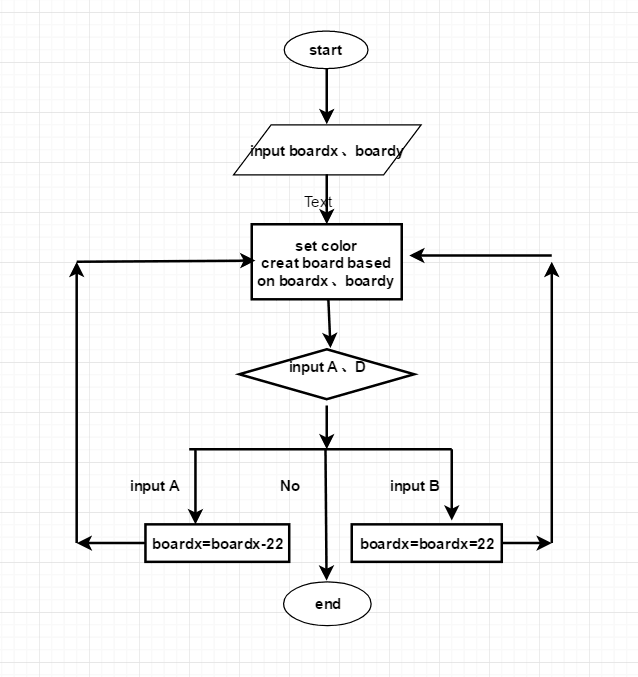
Vx and Vy represent the direction of the ball’s moving. When Vx> and Vy>0, the ball moves to the lower right. First, a black ball is drawn in the position before the displacement of the ball. And then a new ball is drawn in the position after the displacement to give a visual sense of the movement of the ball. When the ball comes into contact with the obstacles on the map, the direction of the ball’s moving changes(Vx and Vy change sign), and then the ball moves until the end of the game.

### Creat brick

.

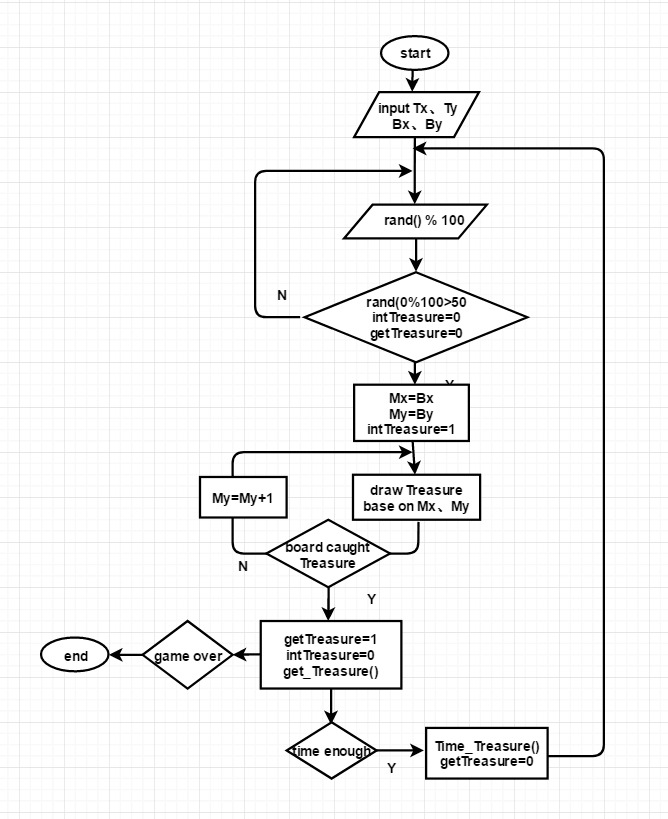
First, set up two arrays x [], y[] to store the number about the bricks’ position, and then use the i and j said bricks’ rows and columns. When i and j represents the ideal position in the map, according to the value of the i j, define the two arrays. When all the bricks are assigned in the two arrays, according to the data in both arrays, it is easy to drow all the bricks.

### boardmove



Input the initial position coordinates of the board and draw a board. According to the keyboard input A or D, change the abscissa boardx. In the original position draw a new board, whose color the same with background color. and then draw a new board in the new location, implementation of moving board. But if the new board is out-of-range (boardx<=0 or boardx>=mapx), the board won’t move. If there isn’t a keyboard input, function over and wait for the next round of cycle to running this function again.

### The props



Generate a random number, if the remainder of a random number to satisfy it divided by 100 more than 50, intTreasure = 0 and getTreasure = 0, there will have a prop in the position where the last brick broken and intTreasure = 1. Increasing My ,according to Mx and My, in the original position, draw a black props and in the new position draw a new props to realise the moving of props. If the board catches the prop, intTreasure=0, getTreasure=1 and run the get\_Treasure () to make a since about prop. After some time, run the Time\_Treasure () invalidate the prop and getTreasure=0. Then a new random number is generated to continue the cycle until game over.

## Implementation

In the process of making the game, we encountered three major problems. These three problems are about the three most important functions in this game.

1. The first problem is how to draw bricks.

**Codes:**

case 6:**//Fixed map 5--“AT”**

{

int X6[28], Y6[28];

setfillcolor(GREEN);

for (int q = 1; q<12; q = q + 2)

{

int w;

w = q / 2 + 6;

int x = (w + 2) \* 22;

int y = q \* 12;

X6[(q - 1) / 2] = x;

Y6[(q - 1) / 2] = y;

}

for (int w = 1; w<12; w = w + 2)

{

int e;

e = 6 - w / 2;

int x = (e + 2) \* 22;

int y = w \* 12;

X6[5 + (w + 1) / 2] = x;

Y6[5 + (w + 1) / 2] = y;

}

for (int z = 3; z<10; z = z + 2)

{

int x = (z + 2) \* 22;

int y = 7 \* 12;

X6[11 + (z - 1) / 2] = x;

Y6[11 + (z - 1) / 2] = y;

}//"A"

setfillcolor(RED);

for (int a = 13; a<25; a = a + 2)

{

int x = (a + 2) \* 22;

int y = 24;

X6[15 + (a - 11) / 2] = x;

Y6[15 + (a - 11) / 2] = y;

}

for (int b = 2; b<13; b = b + 2)

{

int y = (b + 1) \* 12;

int x = 418;

X6[21 + b / 2] = x;

Y6[21 + b / 2] = y;

}**//“T**”

**//Store brick coordinates as array elements**

for (int tt = 0; tt<28; tt++)

{

if (arr[tt][tt] == 0)

solidrectangle(X6[tt], Y6[tt], X6[tt] + 20, Y6[tt] + 10);**//Draw the map**

}BallMove(X6, Y6, 28);**//Pass the bricks as parameters to the next function**.

}break;

This is one of the maps we made. as you can see, firstly we use mathematical knowledge to calculate the coordinates of the bricks we want to create, and then store the abscissa and ordinate of these bricks in two arrays. Finally, specify the size of each brick, and then use the solid rectangle function to draw these bricks.

In this problem, We added an innovative part: how to draw your own map model

**Codes:**

if(number==2)

{

for(int p=0;p<10;p++){

scanf("%d%d",&t[p],&m[p]);**//The first number must be less than 40 and the second number must be less than 25. But both must be greater than 0.**

solidrectangle(m[p]\*22-22,t[p]\*12-12,m[p]\*22-2,t[p]\*12-2);}

}

CreatBrick(number,t,m);

If you choose this mode, our solution is to store the abscissa and ordinate of each input brick in an array, And draw the model when inputting each brick coordinate, and transfer it to the CreatBrick function to draw after all brick coordinates are input.

1. The second problem is how to move the board.

**Codes:**

void BoardMove()

{

setfillcolor(BLACK);

solidrectangle(boardx, boardy, boardx + length, boardy + 10);

char c = \_getch();

switch (c)

{

case 75:

case'A':

case'a':

boardx -= 15;

break;

case 77:

case 'd':

case 'D':

boardx += 15;

break;

default:

break;

}

if (boardx <= 0)

boardx = 0;

if (boardx >= 550 - length)

boardx = 550 - length;

setfillcolor(BLUE);

solidrectangle(boardx, boardy, boardx + length, boardy + 10);

}

This problem is actually relatively simple. we only need to change the color of the board to the background color. then when inputting " a" or " d", the coordinates of the board will be reduced or increased by a certain unit length. finally, the board will be repainted according to the new board coordinates.

1. The third problem is how to solve the problem of ball movement and how to remove the bricks that were hit.

**Codes:**

void BallMove(int \*X, int \*Y, int len)**//The coordinates of the bricks are passed into this function as parameters**

{

iscatch = 0;

setfillcolor(BLACK);

solidcircle(ballx, bally, 10);

BeginBatchDraw();

if (ballx >= 540 || ballx <= 10) **//Left wall and right wall**

{

velocity\_x \*= -1;

}

if (bally <= 10) **//Upper wall**

velocity\_y \*= -1;

int flag = 0;

for (int m = 0; m != len; m++)

{

int x = X[m];

int y = Y[m];

if (arr[m][m] == 0 && ballx >= x - 10 && ballx <= x + 32 && bally <= y + 22 && bally >= y - 10)

{

if (bally >= y && bally <= y + 12) **//左右**

velocity\_x \*= -1;

else if (ballx >= x && ballx <= x + 22) **//上下**

velocity\_y \*= -1;

else if (pow(ballx - x, 2) + pow(bally - y, 2) <= 100 || pow(ballx - x - 22, 2) + pow(bally - y, 2) <= 100 || pow(ballx - x, 2) + pow(bally - y - 12, 2) <= 100 || pow(ballx - x - 22, 2) + pow(bally - y - 12, 2) <= 100)

{

velocity\_x \*= -1;

velocity\_y \*= -1;

}

else

continue;

arr[m][m] = 1;

flag = 1;

setfillcolor(BLACK);

solidrectangle(x, y, x + 20, y + 10);

if (starting&&ballx >= boardx - 10 && ballx <= boardx + length + 10 && bally >= boardy - 10 && bally < boardy)

{

iscatch = 1;

if (ballx >= boardx && ballx <= boardx + length)

velocity\_y \*= -1;

else if (pow(ballx - boardx, 2) + pow(bally - boardy, 2) <= 100|| pow(ballx - boardx-length, 2) + pow(bally - boardy, 2) <= 100)

{

velocity\_y \*= -1;

velocity\_x \*= -1;

}

}

starting = 1;

ballx += velocity\_x;

bally += velocity\_y;

setfillcolor(WHITE);

solidcircle(ballx, bally, 10);

FlushBatchDraw();

}

For the problem how to solve the problem of ball movemen , the movement of the ball is closely related to the removal of bricks. we first specify the coordinates of the ball pen to change the movement direction according to the coordinates of the upper wall, the left wall and the right wall of the known game interface. Then the array of brick coordinates stored during brick making is passed into this function in the form of parameters. through this method, we get the coordinates of the rebound when the ball collides with the brick.

As for how to eliminate the hit brick, we mainly change the color of the hit brick to the background color and eliminate it by changing its parameters regarding the existence of the brick.

1. The fourth question is how to make props randomly produce different effects.

**Codes:**

if(rand()%100<60 &&intTreasure==0 && getTreasure==0) **//30 % chance of treasures**

{

Treasure.x=x;

Treasure.y=y;

intTreasure=1;

if(rand()%100<25) **//accelerate**  {

Treasure.type=1;

}

else if(rand()%100<50) **//decelerate**

{

Treasure.type=2;

}

else if(rand()%100<75) **//Boards grow longer**

{

Treasure.type=3;

}

else **//Board shortening** {

Treasure.type=4;

}

}

}

}

As to how to randomly produce props with different effects. We mainly use random numbers to specify the probabilities of different props through the probabilities of random numbers generated in different ranges.

## Testing and Debugging

1. The first test was about board movement.

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Real Result |
| “A” | The abscissa of the board decreases by 15 | The abscissa of the board decreases by 15 |
| “D” | The abscissa of the board is increased by 15 | The abscissa of the board is increased by 15 |
| “A” “A” | The abscissa of the board first increases by 15, then increases by 15 | The abscissa of the board first increases by 15, then increases by 15 |
| “A” “D” “D” | The abscissa of the board first increases by 15, then decreases by 15, and finally decreases by 15 | The abscissa of the board first increases by 15, then decreases by 15, and finally decreases by 15 |
| “S” | There is no change in the abscissa of the board | No response |
| “J” | There is no change in the abscissa of the board | No response |
| “Space” | There is no change in the abscissa of the board | No response |
| “5” | There is no change in the abscissa of the board | No response |
| “7” | There is no change in the abscissa of the board | No response |

From the results of testing and debugging, it can be seen that the code of wood board movement meets the requirements. only when " a" and " d" are input can the wood board move normally.

1. The second test is about the player's own map drawing mode.

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Real Result |
| “3” “3” | One value in array Y is 3 and one value in array X is 3. And brick appear at corresponding positions | One value in array Y is 3 and one value in array X is 3. And brick appear at corresponding positions |
| “14” ”15” | One value in array Y is 14 and one value in array X is 15. And brick appear at corresponding positions. | One value in the array of abscissa is 15, and one value in the array of ordinate is 14. And brick appear at corresponding positions. |
| “3” “30” | One value in array Y is 3 and one value in array X is 30. But no bricks appeared in the interface. | One value in array Y is 3 and one value in array X is 30. But no bricks appeared in the interface. |
| “0” “3” | One value in array Y is 0 and one value in array X is 3.But no bricks appeared in the interface. | One value in array Y is 0 and one value in array X is 3.But no bricks appeared in the interface. |
| “a” “b” | Error occurred and no bricks appeared in the interface. | Error occurred and no bricks appeared in the interface. The program prompts you to run incorrectly and the game is closed. |
| “space” “3” | Error occurred and no bricks appeared in the interface. | Error occurred and no bricks appeared in the interface. The program prompts you to run incorrectly and the game is closed. |
| “a” “3” | Error occurred and no bricks appeared in the interface. | Error occurred and no bricks appeared in the interface. The program prompts you to run incorrectly and the game is closed. |

From the results of testing and debugging, it can be seen that the codes of the bricks made by the players themselves meet the requirements. only when the coordinates of the specified range are input can the game run normally.

## Result&Conclusion

By the time of the report, we had basically completed our initial goal of completing the basic brick breaking process and adding prop function. After the reporting, we discussed together according to the teacher's evaluation and others groups’ report, and further improved our code. Although the final program was not perfect, some function was not well realized, but everyone in our group was proud and satisfied. We totally relied on our own strength, relied on our own persistence to make our own game.

We designed the initial interface, inserted background music, designed different brick maps, completed the board movement, ball movement and ball bounce, props function and game end interface. Our group also has very good innovation thought, in order to increase the playability of the game, we created a new mode. In this mode, players can make their own maps based on their ideas.

We did run into a lot of problems in the process. When we were drawing brick maps, boards, balls, we wanted to use “printf”, but the graphics were too simple, and then we used Easy-x,using the coordinate mapping, the graphics became beautiful. At the same time, it also be more convenient to realize the function of ball bounce and props. At the ball bounce, we also had a huge problem. Sometimes the boards seemed to catch the ball, but the game was over, and sometimes when the brick was hit by the ball, there was a sense of embeddedness.As we were trying to solve the problems,we find our that there is no need to completely respect the rules of physics, no need to wait until the ball touch other object, and we can leave a little bit of error space on the data, which will make the vision more perfect. In thh implementation of prop function, there are also some problems. And the most difficult thing for us is the falling speed of the props, because we can't make the ball move too fast, too fast, the board cant catch it. We also can't make it too slow, too slow, player would have no game experience. The same is true of props. But if props and ball speed is the same, there will be a phenomenon that we can only catch one of them. And in fact,we can only catch the ball if you want to continue the game.So the prop will be chicken ribs. So at this point, we finally had to choose to let the props drop a little faster.

Of course, for a progress of division of labor, for beginners like us, integration is also very difficult. We have a lot of problems with integration, for example, to eliminate global variables as much as possible (we didn't do very well at this point, but we did do our best); although others have explained his code to you, you may still be a little confused when you integrate, the code is too long for us, we are likely to delete some code by mistake, and then the our mind will be collapsed. This process depends entirely on the solidarity and persistence of the group. Integration phase, there are two weeks from the final report, everyone went to bed after midnight everyday. After integration , everyone will stare at the code at our free time, seeing where we can improve. The improved code, can be send three or four in the QQ group every day.

In short, Rome wasn’t build in a day. Three feet of ice is not a day of cold, the realization of the entire function is the result of our long-term efforts;The Foolish Old Man, who removed the mountains, is not just one person. Everyone division of labor is clear, their respective tasks, burdens, psychological pressure will not be so great. Our team is pleased that we are on track to achieve our goal; we are proud to have made such a program as a non-computer professional. We really think C language is interesting. Thanks for AT and Xu Hao give us such chance to lay the root for it.