《现代C++编程实现》电子版：https://leanpub.com/c01

控制台游戏-Pong游戏

YouTube频道： hwdong

博客： hwdong-net.github.io

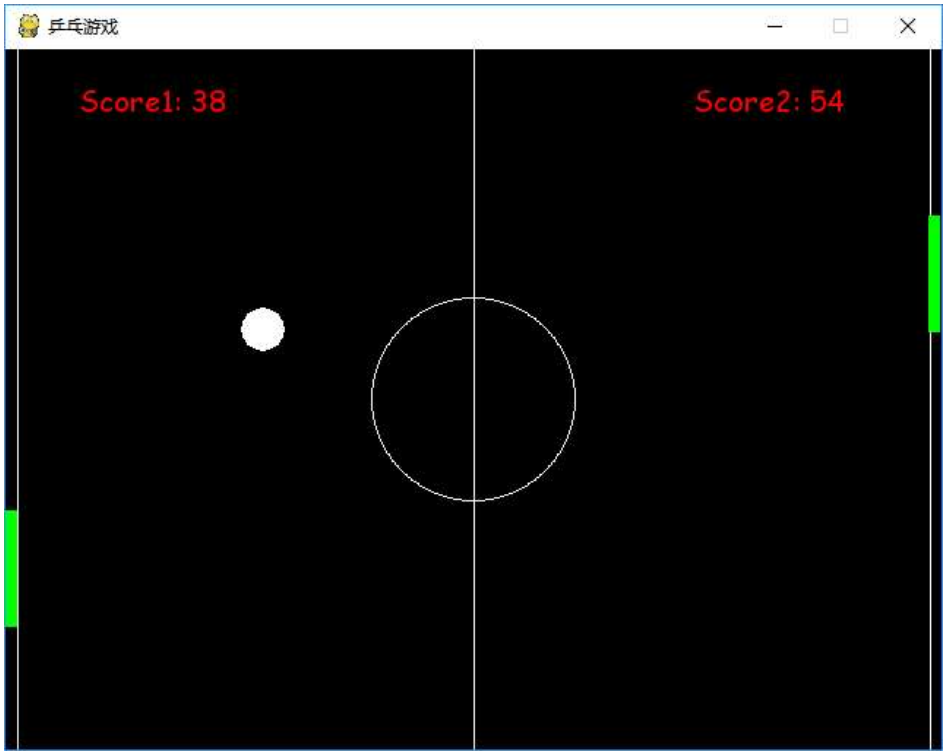
B站: hw-dong

Pong游戏

- 1972年，艾尔·奥尔康（Al Alcorn）为雅达利（Atari）公司开发的模拟乒乓球的《PONG》街机游戏。
- 从酒吧开始风靡美国，很快传到日本。
- 日本的娱乐公司Taito开发了一个类似的克隆产品《Elepong》，成为了日本的第一款电子游戏。

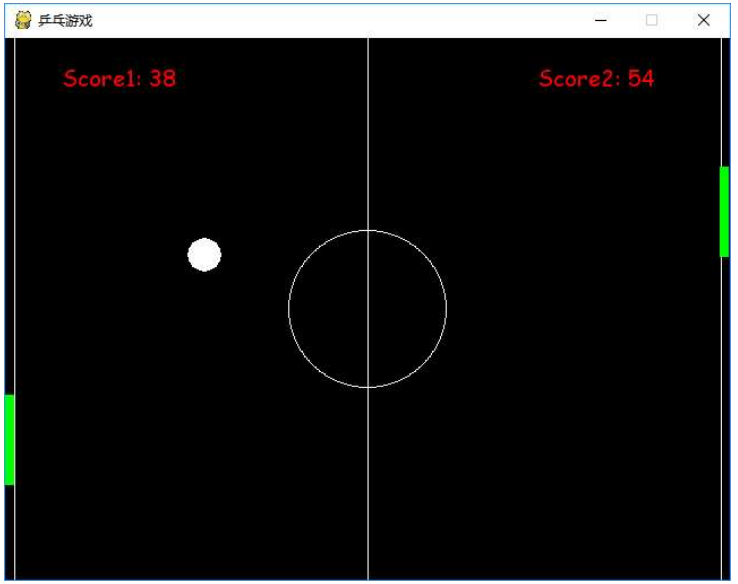
Pong游戏画面

- 1个乒乓球（Ball）和左右两个挡板（Paddle）



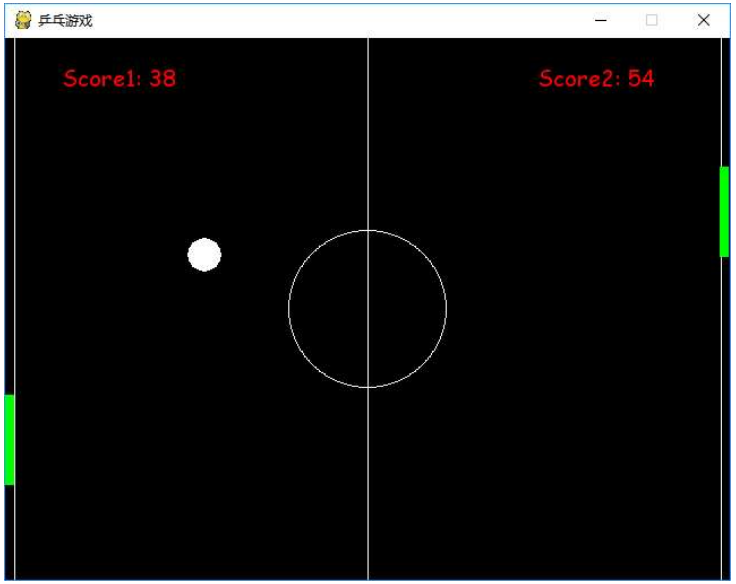
初始化

- 游戏画面有一个窗口，窗口里除了一些背景（如Pong游戏中的分隔线），主要运动物体是一只球（Ball）和2个挡板（Paddle），分别用1个圆和2个矩形表示。



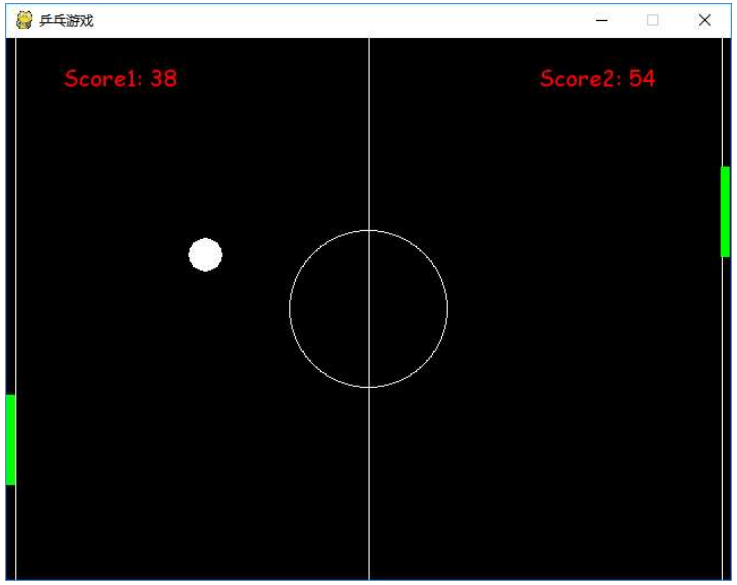
初始化

- 球有位置、大小（半径）、颜色、速度等属性，而挡板也有位置、大小（长宽）、颜色、速度等属性。

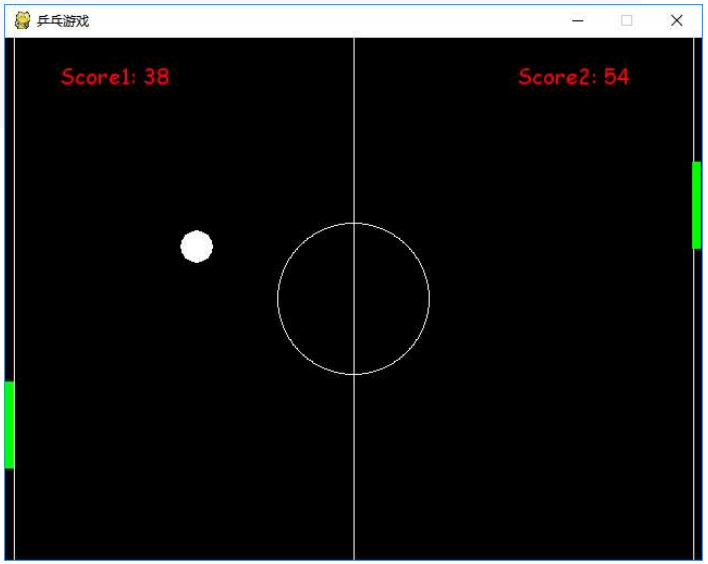


初始化

- 游戏还有记录各自分数的变量。当然，游戏窗口也有长宽、标题、背景颜色等属性。

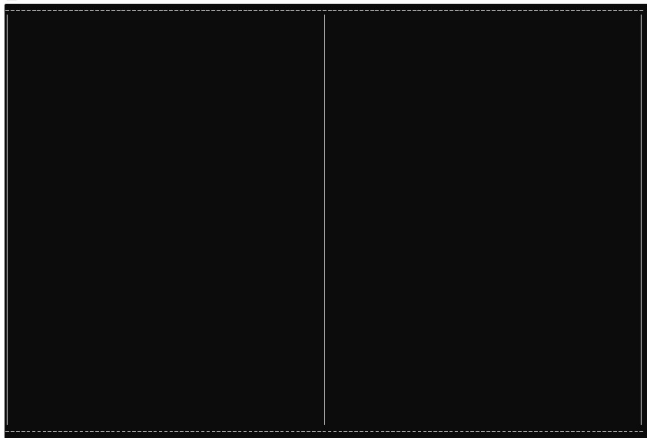


```cpp
#include <iostream>
using namespace std;
int main() {
	//1. 初始化游戏中的数据
	auto WIDTH{ 120 }, HEIGHT{ 40 }; //窗口长宽
	auto ball_x {WIDTH/2}, ball_y{HEIGHT/2}, ball_vec_x{0}, ball_vec_y{0}; //球位置及速度
	auto paddle_w{4}, paddle_h{10};   //挡板的长宽
	auto paddle1_x{0}, paddle1_y{HEIGHT/2-paddle_h/2}, paddle1_vec{3}; //挡板1位置及速度
	auto paddle2_x{ WIDTH - paddle_w },
		paddle2_y{ HEIGHT/2 - paddle_h/2 }, paddle2_vec{3}; //挡板2位置及速度
	auto score1{ 0 }, score2{ 0 };  //双方的得分
	return 0;
}
```

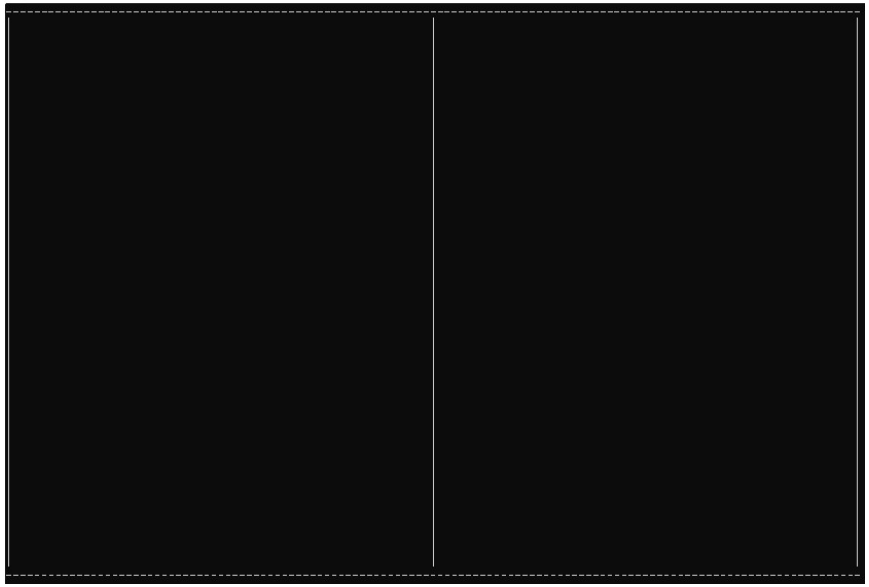


绘制场景

- 绘制场景包括绘制背景和游戏中的运动物体。可以先绘制背景中的上下墙壁和3条竖线。
- 也就是通过在控制台窗口中输出一些特殊字符如|或=分别表示3条竖线和墙的图案。



```
int main() {
        //…

        //2. 绘制场景
        //2.1绘制背景
        //2.1.1 先绘制背景中的顶部墙
        for (auto x = 0; x <= WIDTH; x++)
                std::cout << '=';
        std::cout << '\n';
        //2.1.2 绘制背景中的3条的竖线
        for (auto y = 0; y <= HEIGHT; y++) {
           for (auto x = 0; x <= WIDTH; x++)
             if (x == 0 || x == WIDTH / 2 || x == WIDTH)
                std::cout << '|';
             else std::cout << ' ';
                std::cout << '\n' ;
        }
        //2.1.3 绘制背景中的底部墙
        for (auto x = 0; x <= WIDTH; x++)
           std::cout << '=';
        std::cout << '\n';
}
```



绘制场景

- 挡板可以用一个矩形表示，球用大写字母O表示。

绘制场景

- 为了将球和挡板绘制在窗口画面中，需要在前面的绘制背景的循环中判断哪些位置是球和挡板，然后在这些位置绘制代表球和挡板的特殊图案字符。

```
//2.1.2 绘制背景中的3条的竖线
for (auto y = 0; y <= HEIGHT; y++) {
   for (auto x = 0; x <= WIDTH; x++)
     if (x == 0 || x == WIDTH / 2 || x == WIDTH)
        std::cout << '|';
     else std::cout << ' ';
        std::cout << '\n' ;
}
```

绘制场景

- 因此，需要修改前面的if...else语句，在球和挡板的位置绘制球和挡板。

```
for (auto y = 0; y <= HEIGHT; y++) {
   for (auto x = 0; x <= WIDTH; x++)
     if(x==ball_x&&y==ball_y)                                    //球的位置
        std::cout << 'O';
     else if (y>= paddle1_y &&y < paddle1_y + paddle_h
                &&x>= paddle1_x && x < paddle1_x + paddle_w) {    //左挡板位置
        std::cout << 'Z';
     }
     else if (y >= paddle2_y && y < paddle2_y + paddle_h
                && x >= paddle2_x && x < paddle2_x + paddle_w) {    //右挡板位置
        std::cout << 'Z';
     }
     else if (x == 0 || x == WIDTH  / 2 || x == WIDTH)          //竖线位置
        std::cout << '|';
     else std::cout << ' ';
        std::cout << '\n' ;
}
```

让球动起来

- 游戏中是通过不断绘制新的画面让游戏画面动起来的。
- 游戏实际是如下的一个循环过程，

```
初始化游戏数据
循环（直到游戏结束）{
    处理事件（如用户输入、定时器）
    更新游戏状态（游戏中的数据）
    绘制游戏画面
}
```

让球动起来：清除画面

- 为了绘制新的画面，必须先清除原来的画面，在Windows平台上，可以用如下的gotoxy函数（需要包含windows.h头文件）。

```
#include <windows.h>
void gotoxy(int x, int y){
    COORD coord = {x, y};
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
```

让球动起来：清除画面

- 只要在每次绘制新的画面前，调用这个函数gotoxy(0,0)将光标定位在左上角就相当于清除了屏幕的内容。

```
#include <windows.h>
void gotoxy(int x, int y){
    COORD coord = {x, y};
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
```

让球动起来：隐藏光标

- 为了防止画面刷新时出现闪烁光标，可以使用下面的函数隐藏掉光标：

```
void hideCursor(){
    CONSOLE_CURSOR_INFO cursor_info = {1, 0};
    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cursor_info);
}
```

让球动起来：给球一个随机的初始速度

- 用随机数生成的相关函数，生成随机的初始化速度。

```
srand((unsigned)time(0));                    //生成随机数种子
    ball_vec_x = rand() % 3 + 1;                    //生成一个随机整数，表示x和y方向的速度大小
    ball_vec_y = rand() % 3 + 1;
    if (rand() % 2 == 1) ball_vec_x = -ball_vec_x;   //随机改变初始的速度方向
    if (rand() % 2 == 1) ball_vec_y = -ball_vec_y;
```

让球动起来：给球一个随机的初始速度

- rand()函数用于生成一个随机数种子，然后用rand()函数生成一个整数，通过%运算，使得代表速度的整数不至于过大。

```
srand((unsigned)time(0));                    //生成随机数种子
    ball_vec_x = rand() % 3 + 1;                    //生成一个随机整数，表示x和y方向的速度大小
    ball_vec_y = rand() % 3 + 1;
    if (rand() % 2 == 1) ball_vec_x = -ball_vec_x;   //随机改变初始的速度方向
    if (rand() % 2 == 1) ball_vec_y = -ball_vec_y;
```

让球动起来：根据球的速度更新球位置

- 在游戏循环中根据速度不断更新球的位置，并绘制游戏画面，就能让球动起来。

```
ball_x += ball_vec_x;    //根据速度改变位置
ball_y += ball_vec_y;
```

完整代码

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <windows.h>
using namespace std;

void gotoxy(int x, int y) {
    COORD coord = { x, y };
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
void hideCursor() {
    CONSOLE_CURSOR_INFO cursor_info = { 1, 0 };
    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cursor_info);
}
```

```
int main() {
    //1. 初始化游戏中的数据
    auto WIDTH{ 120 }, HEIGHT{ 40 }; //窗口长宽
    auto ball_x {WIDTH/2}, ball_y{HEIGHT/2}, ball_vec_x{0}, ball_vec_y{0};
    auto paddle_w{4}, paddle_h{10};
    auto paddle1_x{0}, paddle1_y{HEIGHT/2-paddle_h/2}, paddle1_vec{3};
    auto paddle2_x{WIDTH-paddle_w},
        paddle2_y{HEIGHT/2-paddle_h/2},paddle2_vec{3};

    srand((unsigned)time(0));  //生成随机数种子
    ball_vec_x = rand() % 3 + 1; //生成一个随机整数
    ball_vec_y = rand() % 3 + 1;
    if (rand() % 2 == 1) ball_vec_x = -ball_vec_x;
    if (rand() % 2 == 1) ball_vec_y = -ball_vec_y;

    //游戏循环
    while (true) {
        // 1. 处理事件
```

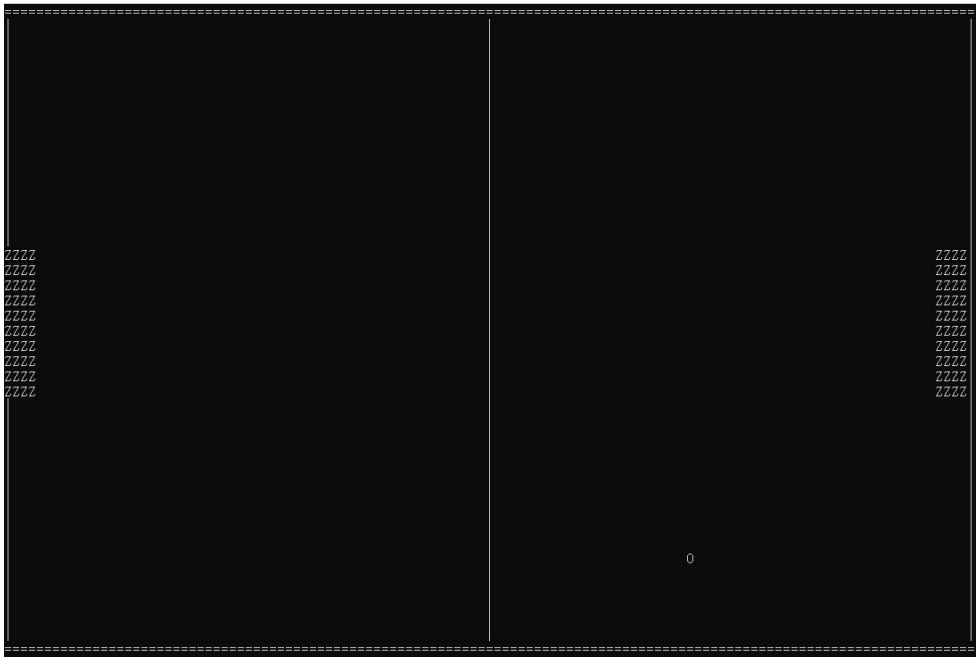
```
//游戏循环
while (true) {
    // 1. 处理事件

    // 2. 更新数据
    ball_x += ball_vec_x;
    ball_y += ball_vec_y;

    gotoxy(0, 0);  //定位到(0,0)，相当于清空屏幕
    hideCursor();  //隐藏光标

    // 3. 绘制场景
    //3.1绘制背景
    //3.1.1 先绘制背景中的顶部墙
    for(auto x = 0; x <= WIDTH; x++)
        std::cout << '=';
    std::cout << '\n';
```

```
//3.1.2 绘制背景中的3条的竖线、球、挡板
for (auto y = 0; y <= HEIGHT; y++) {
   for (auto x = 0; x <= WIDTH; x++) {
      if (ball_x == x && ball_y == y)
          std::cout << 'O';
      else if (y >= paddle1_y && y < paddle1_y + paddle_h
                  && x >= paddle1_x && x < paddle1_x + paddle_w) {
          std::cout << 'Z';
      }
      else if (y >= paddle2_y && y < paddle2_y + paddle_h
                  && x >= paddle2_x && x < paddle2_x + paddle_w) {
          std::cout << 'Z';
      }
      else if (x == 0 || x == WIDTH / 2 || x == WIDTH)
          std::cout << '|';
      else std::cout << ' ';
   }
   std::cout << '\n';
}
```

```
        //3.1.3 绘制背景中的底部墙
        for (auto x = 0; x <= WIDTH; x++)
            std::cout << '=';
        std::cout << '\n';
    }
    return 0;
}
```

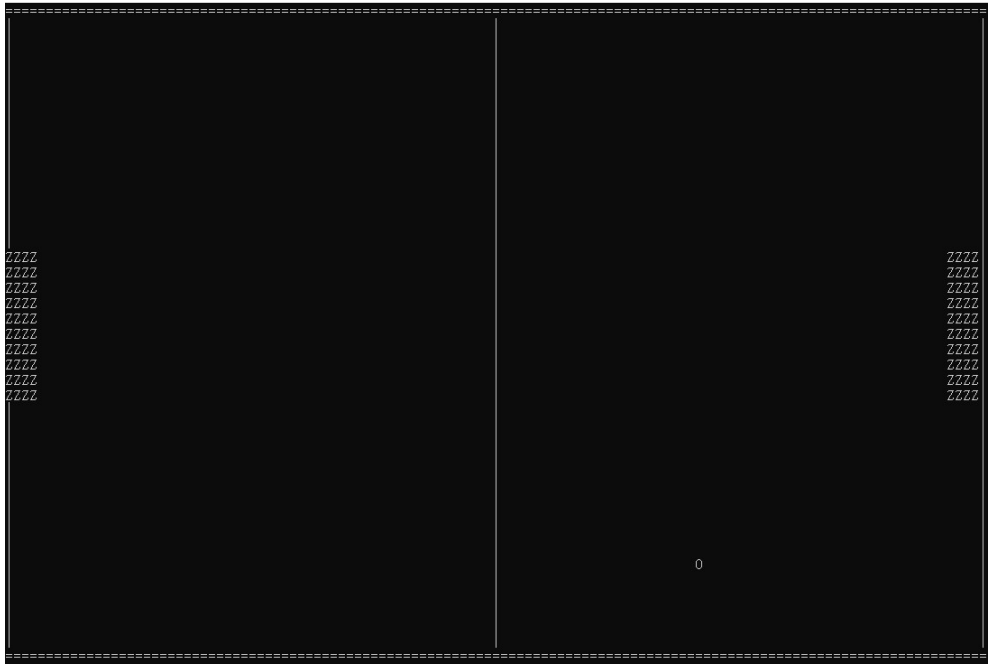
让球动起来：碰撞检测、更新数据

- 球如果和上下墙壁发生碰撞会反弹回来，而遇到挡板也会发生反弹，如果越过左右沟渠而没有遇到挡板，则球跑出画面，另外一方的得分将增加，然后球重新从新位置以随机速度出发。



让球动起来：碰撞检测、更新数据

- 先检测球是否和墙壁或挡板碰撞，如发生碰撞，通过改变球的相应的速度方向，而使球发生反弹，和上下墙壁碰撞，球的垂直速度方向变成反方向，和左右挡板碰撞，球的水平速度方向变成反方向。



让球动起来：碰撞检测、更新数据

```
ball_x += ball_vec_x;
ball_y += ball_vec_y;
if (ball_y < 0 || ball_y >= HEIGHT)            //和上下墙碰撞，改变垂直速度方向
   ball_vec_y = -ball_vec_y;

if (ball_x < paddle_w&& ball_y >= paddle1_y && ball_y < paddle1_y + paddle_h)
{ //和左挡板碰撞，改变水平速度的方向
    ball_vec_x = -ball_vec_x;
    score1 += 1;
}
else if(ball_x >WIDTH - paddle_w && ball_y >= paddle2_y
  && ball_y < paddle2_y + paddle_h)
 { //和右挡板碰撞，改变水平速度的方向
    ball_vec_x = -ball_vec_x;
    score2 += 1;
}
```

让球动起来：碰撞检测、更新数据

```
bool is_out{ false };                    //是否跑出沟渠的bool标志
if (ball_x < 0) {score2 += 1; is_out = true;                }
else if (ball_x >WIDTH-paddle_w) {score1 += 1; is_out = true;    }
if (is_out) {                          //跑出左右沟渠，球回到中心并以新的随机速度出发
  ball_x = WIDTH / 2; ball_y = HEIGHT / 2;
  ball_vec_x = rand() % 3 + 1;
  ball_vec_y = rand() % 3 + 1;
  if (rand() % 2 == 1) ball_vec_x = -ball_vec_x;
  if (rand() % 2 == 1) ball_vec_y = -ball_vec_y;
}
```

让球动起来：事件处理： 用挡板击打球

- 用上下箭头键移动右挡板，而用字母w和s移动左挡板。

让球动起来：事件处理： 用挡板击打球

- 为了得到键盘输入，先用kbhit()函数检测是否存在按键消息，然后通过getch()函数得到按键的字符。这2个函数在头文件conio.h中说明。

让球动起来：事件处理： 用挡板击打球

- 下面代码根据用户输入改变挡板的位置：

```
// 1. 处理事件
  char key;
  if (_kbhit()) {                                    //键盘有输入
     key = _getch();                                 //得到输入的键值
     if ((key == 'w'|| key == 'W')&& paddle1_y > paddle1_vec)
        paddle1_y-=paddle1_vec;
     else if ((key == 's' || key == 'S')&& paddle1_y + paddle1_vec + paddle_h < HEIGHT)
        paddle1_y += paddle1_vec;
     else if (key == 72&& paddle2_y > paddle2_vec)
        paddle2_y -= paddle2_vec;
     else if ((key == 80)&& paddle2_y + paddle2_vec + paddle_h < HEIGHT)
        paddle2_y += paddle2_vec;
  }
```

让球动起来：如何显示分数?

- 首先定义分数以及显示分数位置的变量：

```
auto score1{ 0 }, score2{ 0 }, score1_x{ paddle_w+8 }, score1_y{ 2 },
                    score2_x{ WIDTH- 8- paddle_w }, score2_y{ 2 };
```

让球动起来：如何显示分数?

- 要将int类型的分数转化为字符串才能通过输出字符串的方式显示分数，可以使用c++标准库的字符串类型string的to_string()函数将一个整数转化为一个字符串。

```
std::string s1{ std::to_string(score1) }, s2{ std::to_string(score2) } ;
```

让球动起来：如何显示分数?

- 最后在绘制场景时，在显示分数的位置输出这个表示分数的字符串即可：

```
else if (y == score1_y && x == score1_x) { //左分数位置
    std::cout << s1;
    while (x < score1_x + s1.size()) x++;
    x--;
}
else if (y == score2_y && x == score2_x) {//右分数位置
    std::cout << s1;
    while (x < score2_x + s2.size()) x++;
    x--;
}
```

关注我

博客： hwdong-net.github.io

Youtube频道

hwdong

@hwdong · 5.01K subscribers · 558 videos

博客：https://hwdong-net.github.io >

youtube.com/c/4kRealSound and 4 more links