Victor Collecting Points and Displaying on Screen

Haunted Harbour C++

Anikeith Bankar

Student Game Version: Saahil Kapasi

Real Programming For Kids

August 2019

This document is to implement the scoring points feature for Victor as he is hitting the enemies. This feature is an important feature as it will give the player of the game more of an incentive to destroy the enemies. The steps shown below is configured on the Visual Studio 2019 version. Other versions of Visual Studio are also sufficient. To implement the feature outlined in the document, the student must complete the health bar on screen portion of the game.

The student will want to implement the scoring feature from the beginning, but the student will not know how to get started until all the classes are created, in specific the onHit() method.

Implementing Score Functionality

Ask the student where the scoring implementation should go. In what class specifically, and if we would need to create more members or add another method or none is necessary within classes and it can all be done within the main file.

Creating a member is necessary within the PlayerObject class as we would want all the players to keep track of a score.

PlayerObject.h

```
class PlayerObject: public WeaponsObject
{
public:
    PlayerObject(void);
    int totalHealth;
    int currentHealth;
    int score; // the new member that needs to be added
    int speedx;
    int speedy;
    ...
```

We can initialize our new member to be 0 as it would be wise to do so as the game restarts, the score would as well.

PlayerObject.cpp

```
PlayerObject::PlayerObject(void): WeaponsObject("../Pics/
    ViktorTesla.bmp", 100, 100)
{
setStandRight();
numBullets = 10;
speedx = 0;
speedy = 0;
state = STANDRIGHT;
stoppedLeft = true;
stoppedRight = true;
totalHealth = 100;
currentHealth = 100;
score = 0;  // setting it to 0
...
```

Next we would need to utilize this member of the PlayerObject class and increment it when Victor's bullets hit the flying enemies and the ground enemies. We will start off with the flying enemies because the PlayerObject victor is already being passed into the constructor of the FlyingEnemyObject class. Ask the student what method within the FlyingEnemyObject class we would need to increment the score. If the student says onHit() then the student is correct!

PlayerObject.cpp

```
void FlyingEnemy::onHit(BulletObject* b)
{
    currentHealth -= b->damage;
    if (currentHealth <= 0)
    {
        startCell = 0;
        width = 32;
        height = 28;
        endCell = 6;
        curCell = 0;
        xPic = 0;
        yPic = 156;
        loopCells = false;
        isDead = true;
        viktor->score += 200; // victor receives 200 points for
    hitting a flying enemy
    }
}
```

We also need to implement this for the ground enemies as well. The problem arises here is that we need to implement this with victor hitting the ground enemies but victor is not being passed into the constructor like how FlyingEnemyObject is. To solve this problem, we can always change around the constructor for GroundEnemyObject so let's do it.

GroundEnemy.h

```
class GroundEnemy : public EnemyObject
{ public:
    GroundEnemy(PlayerObject* v); // add PlayerObject *v to the
     constructor
    PlayerObject* viktor; // add a PlayerObject member to the
    class
    int prevX;
    int prevY;
    void move();
    void onHit(BulletObject *b);
    void checkCollisionWithBlock(GraphicsObject* block);

    ~GroundEnemy();
};
```

GroundEnemy.cpp

```
GroundEnemy::GroundEnemy(PlayerObject* v): EnemyObject("../Pics
    /SkullCrab.bmp", 200, 200)
// add victor to the constructor
{
    viktor = v;  // setting victor to the passed in victor
    isDead = false;
    hitWidth = width;
    hitHeight = height;
    ...
```

Now we have to increment the score by hitting the ground enemies. As suggested by the student, we will add that line of code in our onHit() method within the GroundEnemyObject class.

GroundEnemy.cpp

```
void GroundEnemy::onHit(BulletObject* b)
{
    currentHealth -= b->damage;
    if (currentHealth <= 0)
    {
        x += b->speedx;
```

```
        curCell = 0;
        endCell = 0;
        if (speedx < 0)
        {
            yPic = 70;
        }
        else
        {
            yPic = 105;
        }
        int delay = 0;
        isDead = true;
        speedx = 0;
        viktor->score += 100; // adding 100 points to the final score by destroying ground enemy.

    }
}
```

By changing the constructor of the GroundEnemyObject class we will need to update our constructor call in our main file.

HauntedHarbourAB.cpp

```
...
if (strcmp(section, "[Ground-Enemy]") == 0)
        {
            while (true)
            {       fscanf(file, "%d %d %d", &index, &x, &y);

                if (index == -1)
                {
                    break;
                }
                enemies[numEnemies] = new GroundEnemy(&viktor); // passing in victor into the new constructor
                enemies[numEnemies]->x = x;
                enemies[numEnemies]->y = y;
                numEnemies++;
            }
        }
...
```

Displaying Score Next to Healthbar

To display the score on the screen we will need to work with case WM_Paint statement in our main file, like we did with our healthbar. Instead of making a class like we did with the healthbar, we will just implement within the Paint case. We will be utilizing the WIN32 C++ function TextOut to display text to our screen.
TextOut takes in 5 arguments: (HDC screen, int xstartpoint, int ystartpoint, LPCWSTR text, int lengthtext)
LPCWSTR is just a wide a string, do not need to worry too much about this.
To implement this we need to first make a "Score" text. This will be done at the top of the main file where all the declarations of the variables are. **Make sure to include <string >in the header declarations at the top of the page**

HauntedHarbourAB.cpp

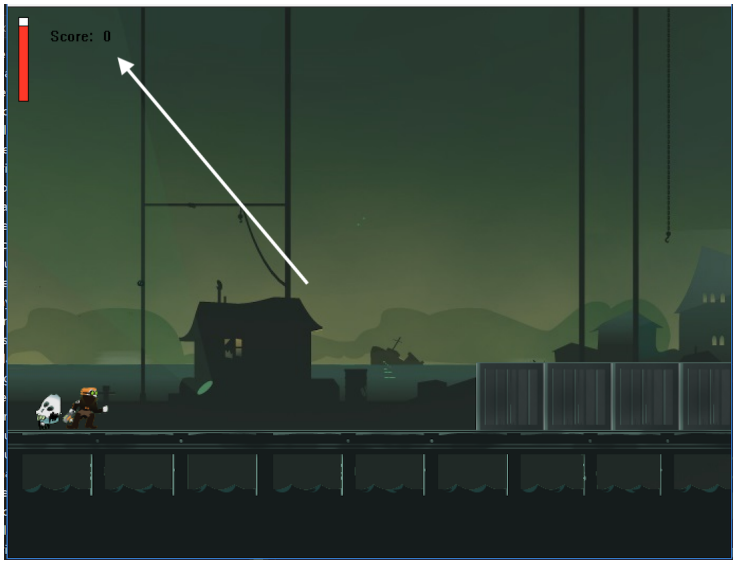
```
...
BackgroundObject ground("../Pics/Ground.bmp", 0, Ground, 774, 128, 1);
BackgroundObject startScreen("../Pics/TitleScreen.jpg", 0, 0, 700, 550, 0);
bool startGame = false;
int mapPosition = 0;
HDC offScreenDC;
int score2 = 0;
int delay = 0;

wchar_t scoreText[] = L"Score:  "; // don't forget the L infront of the string
LPCWSTR wide_scoreText = scoreText ; // wchar_t is the same data-type as LPCWSTR but TextOut only takes LPCWSTR
...
```

Now we will move on to the Paint case and display the score over there.

```
...
if (startGame)
        {
            background.draw(offScreenDC);
            ground.draw(offScreenDC);
            viktor.draw(offScreenDC);
            std::wstring scoreString; // declaring a wide string
```

```
8             LPCWSTR scoreString2; // to change wide string to a
    LPCWSTR
9             scoreString = std::to_wstring(viktor.score); //
    to_wstring function takes in an int
10            scoreString2 = scoreString.c_str(); // transforming
    wide string to LPCWSTR
11            SetBkMode(offScreenDC, TRANSPARENT); // setting the
    background of the text to transparent, default it is an
    ugly white
12            SetTextColor(offScreenDC, RGB(0, 0, 0)); // setting
    the text colour to black
13            TextOut(offScreenDC, 40, 20, wide_scoreText, std::
    wcslen(wide_scoreText); // displaying "Score: "
14                                                    // if
    std::wcslen(wide_scoreText) is not working then put 8
    instead
15            TextOut(offScreenDC, 90, 20, scoreString2, std::
    wcslen(scoreString2)); // displaying continous changing
    score of victor to the right of the scoreText.
16 ...
```

Congratulations! You have successfully implemented a score feature for Victor. The game screen should look similar, if not the same, to the figure below.



More Information about Wide String: Wide String