Test Document for Space Invaders Clone.

# The Barriers

In order to test if the barriers work, I used:

cout << iNoOfHitsBarrier(no. of Barrier) << endl;

This allowed me to check whether the bullets and the invader bullets have collided with the given barrier and find out whether the barrier has decreased, allowing the sprite to change their textures. During the initial creation, I used a vector for the barriers in order to cut down the coding. I realised while testing this that the vector was only destroying the first barrier and not destroying any others. I then made the change to separate the barriers into individual ones to make it easier for the interactions.

# The Player Bullets

In order to test the if the players bullets were moving in the right direction and if they appear in the right place, I used:

cout << sprite.getPosition().x << “ ” << sprite.getPosition().y << endl;

This allowed me to check if every bullet made by the player has been created and is moving in the correct way. First, I didn’t have a speed variable but whilst testing for the first time I saw that the bullets were moving very slowly towards the invaders. This made me create the variable fBulletSpeed that was passed on from the constructor as an argument. This allows me to change the speed as needed and allows me to check whether the bullets are moving in the intended way.

# The Game

## Spawning the Invaders

In testing, even with the two for loops to spawn the aliens, it would run through the subroutine twice producing 110 invaders, 2 layers in the same position. I introduced two more variables to limit the subroutine to only run once. To check that it was working, I outputted the value of fSpawnTimer with:

cout << this->fSpawnTimer << endl;

I put the variables in an if statement to stop it looping.

I did initially have all my update functions in one single function, but for simplicity and to make sure that the program ran a little smoother, I separated the functions out and then called them in the main update function.

## Player Bullets

During the testing of the game, I found that the more bullets that were shot, the slower and slower the game would move. In order to combat this, I culled the bullets at the top of the screen. To show that they were being culled, I used:

cout << this->bullets.size() << endl;

I changed the bullet firing time to shoot more a second and got the following result.

|  |  |
| --- | --- |
| Bullet Number | Output |
| 1 | 1 |
| 1, 2 | 2 |
| 1, 2, 3 | 3 |
| 2, 3 | 2 |
| 3 | 1 |
|  | 0 |

This helped with the lag that was created in the game.

## Rendering

I had problems rendering the invader bullets. I spent a couple of days trying to figure out why they wouldn’t render in my Game.cpp, and then I realized that I was creating all the invader bullets in Invaders.cpp rather than Game.cpp therefore I needed to move my code to Invaders.cpp in order to render these bullets. I moved the code to the render function of Invaders.cpp so when game called the invaders to render, it would render both the aliens and their bullets.

# The Invader Bullets

During the rendering problem, I thought the bullets were moving the wrong way or even not moving. To make sure I was moving them I used:

cout << sprite.getPosition().x << “ ” << sprite.getPosition().y << endl;

This allowed me to check where the bullets were within the window and see where they were moving every frame as this was called for every frame. This allowed me to see I was in fact moving the bullets downwards, but they were a bit fast so I could lower this speed. I did make this class after the bullet class hence they are the same. I did initially have a Boolean value for the bullet to show if the bullet was the players or the invaders, however they would not render the invader bullets. I decided to make two separate classes so I could render both.

# The Invaders

## The Speed

To add a little more to the program, I gave the invaders a speed to move at depending on the number of invaders left. First, I used the equation . This would give the following: 

This made me realise that the less invaders the lower they would move. This was the opposite of what I needed. So instead, I used . This game me: 

This fixed the problem and gave the invaders an exponential increase in their speed relative to the number still on screen.

## The Invader Shots

To make sure the bullets were spawning in the right place, I used:

cout << fPosX << “ ” << fPosY << endl;

This allowed me to check in the console where the bullets were spawning and if they were spawning where the invader that shot it. This didn’t need any alteration.

## Update the Invaders Bullets

There were 3 major problems with this function.

### Bullet and Player Collision

I had problems getting the players position. I coded for the collisions which worked but only if the player didn’t move. If the player moved, I saw that randomly the player would die. After this, I used:

cout << player.getBounds().left << “ ” << player.getBounds().top << endl;

This showed me in the console that the player was not moving and constantly stood at 500, 900. This made no sense as putting the same code in Game.cpp gave me the actual position of the player. After some searching, I realised I had made a mistake by creating an instance of Player in Invaders.h which is completely different to the instance of the player that was created in Game.h. I therefore had to pass in the player instance that was created in Game.h to the update function of Invaders.h which in turned passed it to the updateBullets function. This allowed me to get the players real position and both pieces of code outputted the same position for the player.

### Bullet Culling

I forgot to add the line

delete this->vInvaderBullets.at(iCounter);

This led to a slowdown of the game as the vector would never decrease but only would increase and overtime the game would become unplayable. Although I did not know why it was happening, I ultimately found the problem by using:

cout << this->invaderBullets.size() << endl;

This showed me that the vector would never decrease, and I had not put in a line of code. I quickly rectified this.

### Bullet and Barrier Collisions.

I had the same problem I had with the player where I created a new instance of Barriers, separate to the one in Game.h. This led to me to the same fix.

# The Main program

Here, I needed to test the randomness of the seed I had chosen. Using:

int iRandomNumber = rand();

cout << iRandomNumber << endl;

I found that certain seeds were producing very high numbers, and as this randomness directly impacts the invaders bullets, I tweaked the seed until I got a reasonable random, but small number generator.

# The Player

## Allowing the player to shot.

I had problems with the player not being able to shoot, or the timing between shots. In order to troubleshoot this, I included:

cout << “True” << endl; or cout << “False” << endl;

This helped me see when these functions were working and when the player could shoot. It helped me make the code for the bullets to shot at the right time.

## Is the player dead?

To make sure the death subroutine was working, I used:

cout << “IsDead is happening” << endl;

This allowed me to see when the subroutine was called. I had to make a couple of tweaks to the code to make sure this happened as soon as the player was hit by the bullet.

# Sound Engine

My main problem here was that my instance would not change. This meant that sometimes the sound effect may continuously play. It happened at random with no noticeable pattern. To stop this, I found the assert.h library that can be used to verify assumptions made by the program and print a diagnostic message if this assumption is false. This helped find problems and help me fix the problem.