

SDL Game - White Noise

By James Robertson

Introduction

Welcome to the documentation for my game “White Noise”. The game is a top down shooter with a very simplistic art style. I used only black and white primitives as the graphics to make my game stand out from the rest and I hope that you agree that it does. The game has mouse input for those without a controller but was initially designed with input from an Xbox 360 Controller for Windows.

Research

Style

There are many top down and side on shooters that I have played during my time playing games. I will concentrate on one for this report. The first is a game called Empire of Steel. This game was on the Sega Mega Drive and was one of the first shoot-em-ups that I played. This title had an influence on many games that i created because of the themes that it used and the unique art that is used in that game.



I went the other way with this, making my game seem peaceful with the music and the gameplay not too hectic.

Gameplay

The game has a very simple objective which the player is not really told about because it is so natural to any player of a game. Get points, get kills, and there maybe a little surprise in store for the player! The shooting is quite basic but then the player is given this opportunity to get a power up. A floating "P" this power up allows the ships bullets to go from your normal bullets to super penetration bullets that will plough through hordes of enemies with ease! This adds some variety to the gameplay and ensures that the experience will be different each time you are in the triangle ship.

Enemies also play a huge part into this replayability. I have got preset paths at which the enemies will fly in from but which path and which spawn location is something that is set to a random number generator. The enemies will sometimes move erratically due to this, which makes that game even more enjoyable! Leaving the player on edge every time there is an enemy on screen as you may not know what happens next.

Mathematics

Making a 2D shooter you have to do maths to get something as simple as a bullet to fly upwards. I will go into depth about the maths I have used throughout the program and how I overcame some problems throughout the development of my game.

Problem	Solution
Bullets	So to get bullets to fly upwards towards the top of the screen was some simple vector subtraction. I would get the velocity (Which was 14) and deduct that for each frame of gameplay.
Movement	<p>I had to do this twice due to having two different control methods. The mouse just followed where the cursor was at the time. So that would simply be (In Psuedocode)</p> <p>“PlayerPosition.x = Mouse.x” “PlayerPosiiton.y = Mouse.y”</p> <p>The Controller was a different issue though. The control sticks would output a value of 32676 which is the max range for an 32 bit integer. Adding this the the vector positions would cause the player character to fly off screen quite rapidly.</p> <p>This is why we have division. i would divide the value I get from the controller by 100?? and get smooth analogue movement!</p>
Collision	<p>The maths behind the collision in the game is we have two boxes that are overlapping then we will get a collision and then do something with the result. This is called AABB (Axis Aligned Bounding Box).</p> <p>To do this you have 2 boxes, one called “Bullet” and one called “Enemy” You have to check the top of the bullet to the bottom of the enemy for each frame. If the top of the bullet is ever higher than the enemy then one of the conditions for the collision. However you have to meet two to get a collision, this would check if the right or left of the bullet has hit the opposite side of the enemy.</p> <p>If both those conditions have been met then the collision has occurred!</p>

Programming Logic

Setup

At the start of the program we have to initialise all of the SDL functions that are needed to ensure that the program will run with all of the SDL module ready. This includes setting up the window and the renderer at run time to ensure the game will run.

While setting up the window I made the game run in full screen as I feel like this makes the game much more immersive than others that are in windowed mode. The renderer is initialised to work with VSync to ensure that the game runs smooth at 60FPS. I then initialise all of the classes that I am going to use before the game loop in main. This is mostly to load all the images in and get them ready to draw in the future.

Checks are done in this time to check for a controller being plugged in and what to do with that when you go into the game loop. This is important as it gives the player an option on how they want to play the game on start-up. After the check is done I execute code for XINPUT to enable raw Xbox controller input.

Classes

While creating my game I designed many classes to all work in tandem with one another while often leaning towards optimisation of my code. This would ensure a smooth experience for the player, resulting in no dropped frames and lower minimum and recommended specification to then gain a larger potential audience if this game was to go on sale.

Image Loading

The class I created for this was a simple and really small to avoid code bloat. There are four public functions with two private.

Name	Public/Private	Type	Operation
setFilename	Public	Method / Function	This takes the string from the call in main and sets in the the private string called "filename" to the file location
loadBMP	Public	Method / Function	The method takes a file after you have set the filename and loads it in. Sets the background colour key to clear the light blue to create a transparent image.
freeSurface	Public	Method / Function	This frees the surface from the memory to save space.
convertToTexture	Public	Method / Function	Converts the surface to a texture to get ready for displaying.
filename	Private	String	This data type holds a string for the file that will be loaded for the image.
surface	Private	SDL_Surface *	The temporary surface to load the image onto.

Enemy

The enemy class is where I create the resizable vector where I can spawn, delete, move and test for collision with the enemies that come on screen.

Name	Public/Private	Type	Operation
enemy	Public	Method / Function	This initialises all of the patterns and the variables when the class is initialised.
spawnPattern	Public	Method / Function	The method rolls a random number and spawns a pre-set pattern based off it.
spawnEnemy	Public	Method / Function	This takes in two arguments, the x and y, and spawns an enemy based off them.
moveEnemy (1,2,3,4)	Public	Method / Function	Loops through each enemy in the array and moves them based off the velocity.
drawEnemy	Public	Method / Function	Draws the image for the enemy based on all of the positions in the Vector.
checkCollisions	Public	Method / Function	This takes in one bullet from the main loop and tests it to the enemies.
eraseEnemies	Public	Method / Function	Erases the enemies at the start of the vector. This is called when they
enemies	Private	Vector (SDL Rect)	This is where the enemies positions are placed. This is in a vector so we can manage the memory more effectively.
velocity	Private	Unsigned 8 Bit Integer	Set velocity for the bullets. This is never changed.
positionIncrement	Private	Unsigned 16 Bit Integer	Increment for each spawned enemy on the screen.
randomPattern	Private	Unsigned 8 bit integer	Random value between 0 - 3 for pattern decision making
patternID	Private	Unsigned 8 bit integer	ID to decide which movement to use

Bullet

This class is fairly similar to enemy. The bullets shoot from the ship.

I will go through the differences with enemy.

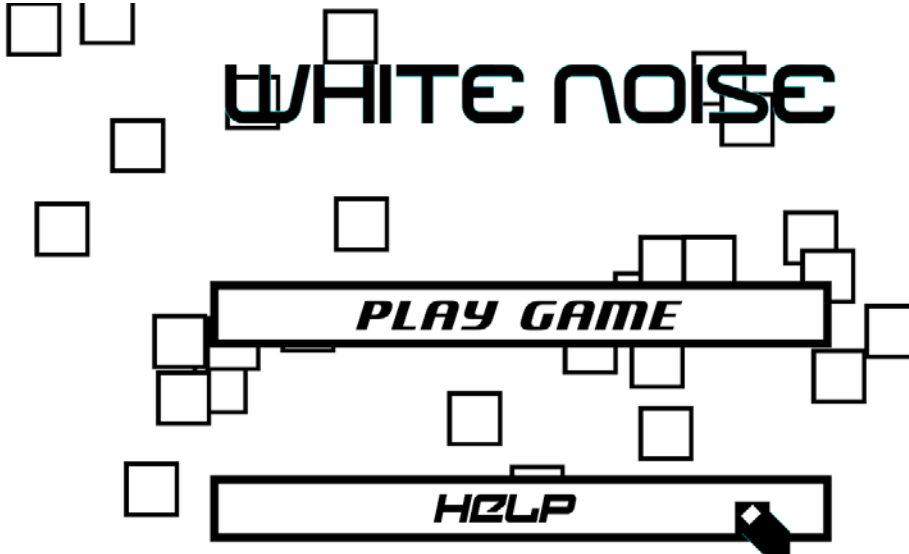
Name	Public/Private	Type	Operation
throwBullets	Public	Method / Function	This throws the bullet when a collision has been met with the enemy
getBullet	Public	Method / Function	This gets the current bullet in the vector.
amountOfBullets	Public	Method / Function	Gets the amount of bullets that a currently in the vector.
bullets	Private	std::vector <SDL_Rect>	Holds the position of each bullet.

Powerup

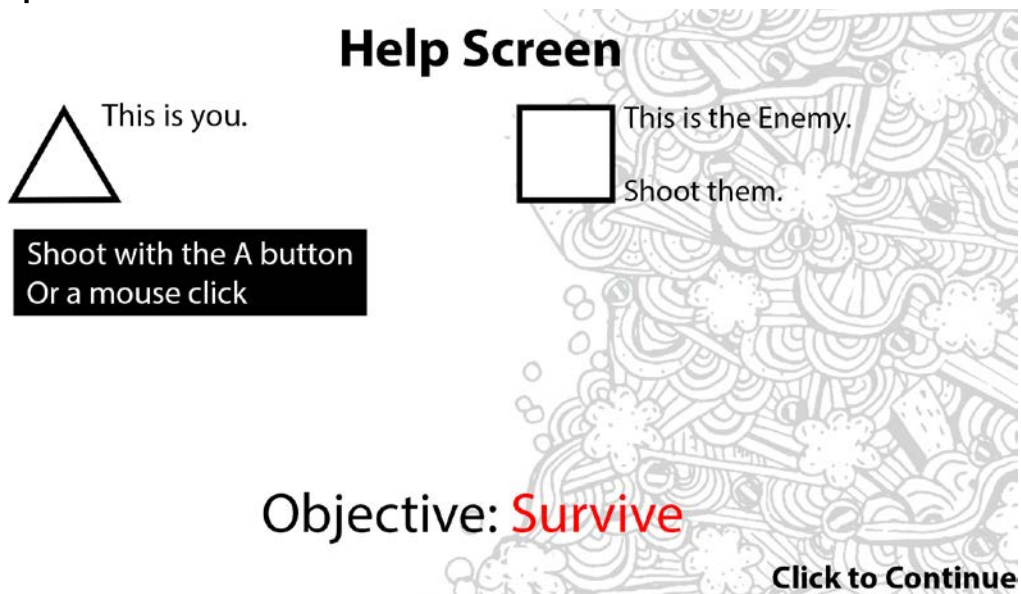
The Powerup class handles the powerups that spawn on screen. The class mirrors the enemy class that was made previously.

Execution

Main Menu

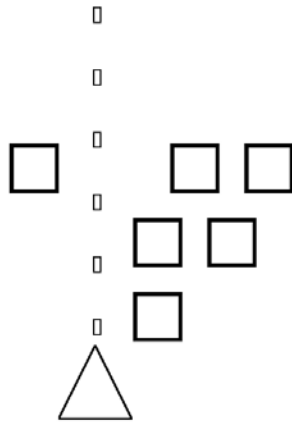


Help Screen



Shooting - Start of game

PRESS ESC TO PAUSE



Game Over



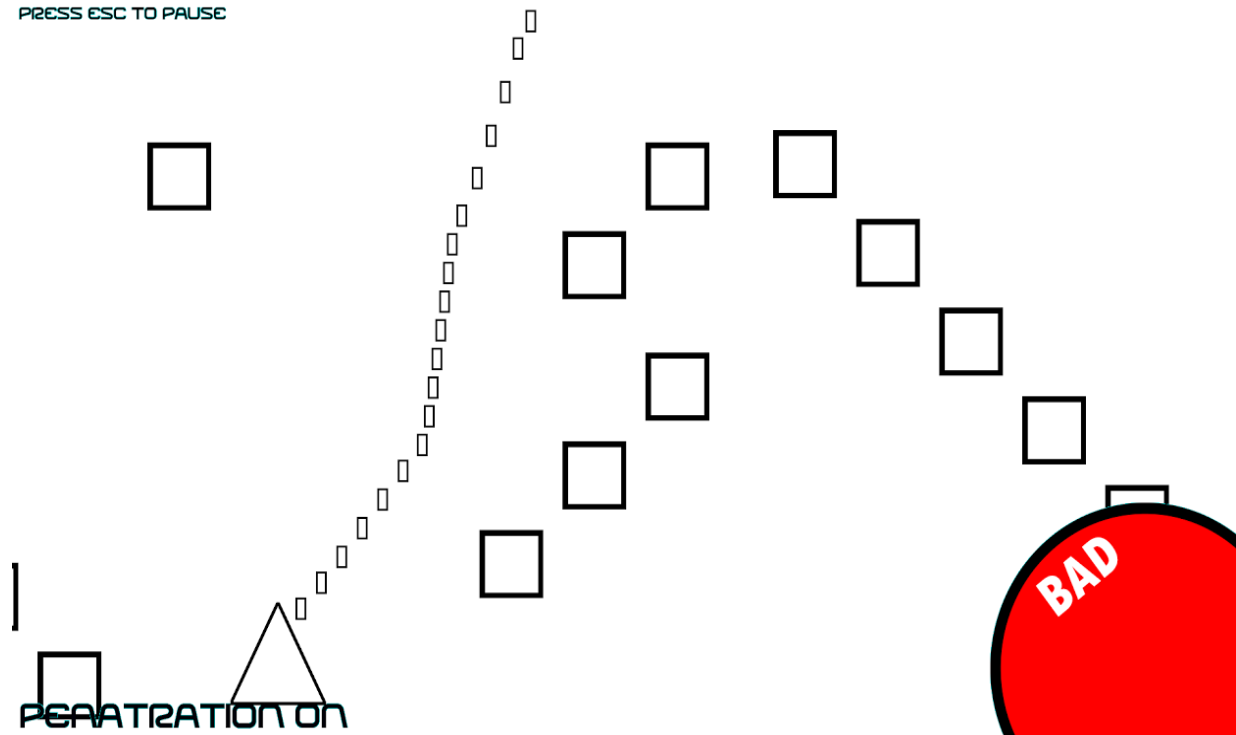
THE SQUARES KILLED YOU.

GAME
OVER

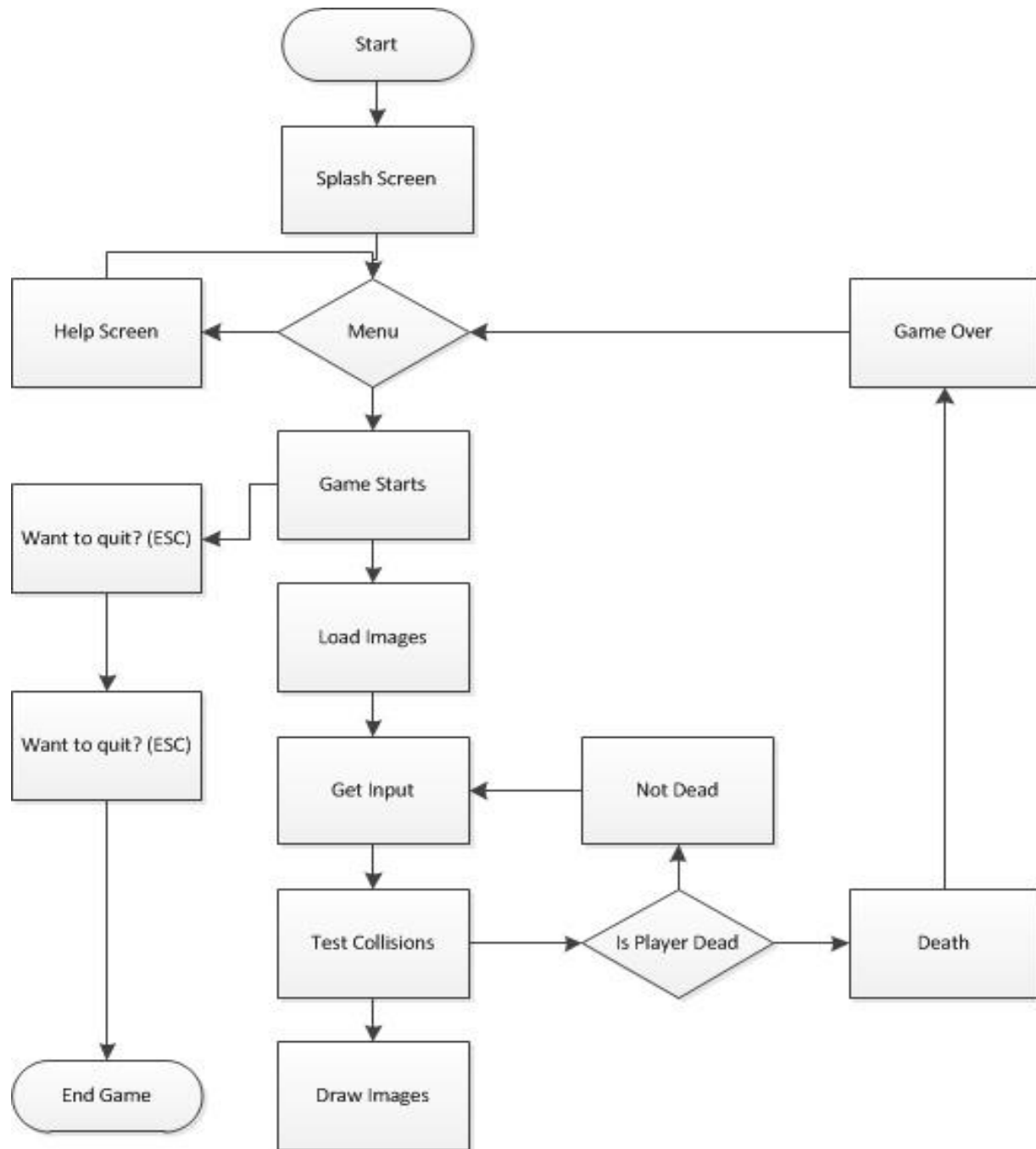
LIFE WILL NEVER BE THE SAME

Boss Rush

PRESS ESC TO PAUSE



Flow Diagram



Diary

White Noise Change Log

This will be the document that will detail all of the changes that have happened with my SDL game that I am creating named "White Noise"

- 17/11/14 - Initialization of window and render complete
 - Changed warning level of Visual Studio to level 4 and fixed all warnings
- 18/11/14 - Image displayed
- 23/11/14 - V-Sync enabled on the renderer
- 26/11/14 - Begun optimising and modulating the start up of the game
- 28/11/14 - Beginning implementation of Xinput
- 02/12/14 - Completed the optimisation
- 04/12/14 - Xinput buttons, triggers and left thumbstick functional
 - Image moving with the thumbstick
 - Vibration added to the thumbstick movement.
- 05/12/14 - "Walls" added to the sides of the levels so the player can not escape
 - Transparent Images
 - Broken Rotation of the gun turret (Microsoft, 32,767 is not easy to convert to 360 degree movement.)
- 06/12/14 - Optimised image loading system is partially implemented
 - Class based structure working smoothly - may need to change string loading but apart from that it is working perfectly
- 07/12/14 - Unstable build from class initialisation, Possible Cause = Null Pointers :(
- 10/12/14 - Roll back to 06/12 version and rewrite classes.....It works! :D
- 11/12/14 - Capped the FPS to 60. - "Mystery Screen Tear" in Room 226 only.
 - Bullet class made!
 - Vector that is dynamically assigning and reassigning bullets depending on if the bullet is active - done!
 - No Level 4 warnings, all fixed
- 22/12/14 - Enemy Loading in place
 - Image loaded
- 28/12/14 - Enemies move and many turn up
 - Collision still a problem due to many enemies and many bullets
 - Not anymore! Fixed using the SDL_HasIntersection() function
 - All level 4 warnings fixed again. The game is fully optimised at this point
- 05/01/15 - Enemies now teleport off screen when attacked.
 - Bullets also now fly off screen when they are done with
 - Commenting and formatting being redone to improve on readability.
 - Doxygen style has been implemented.
 - No level 4 compiler warnings - All fixed.
- 11/01/15 - Music has been added to the game on startup for the menu
 - Tried to improve this inside a class but failed, will retry at a later date.

13/01/15 - Added enemy patterns into the game, now there are 2 patterns that can be selected at random to add some surprise and replayability to the game.

14/01/15 - Audio fixed.

- Power Ups added to the game
- Rearranged the CPP files and organised more.

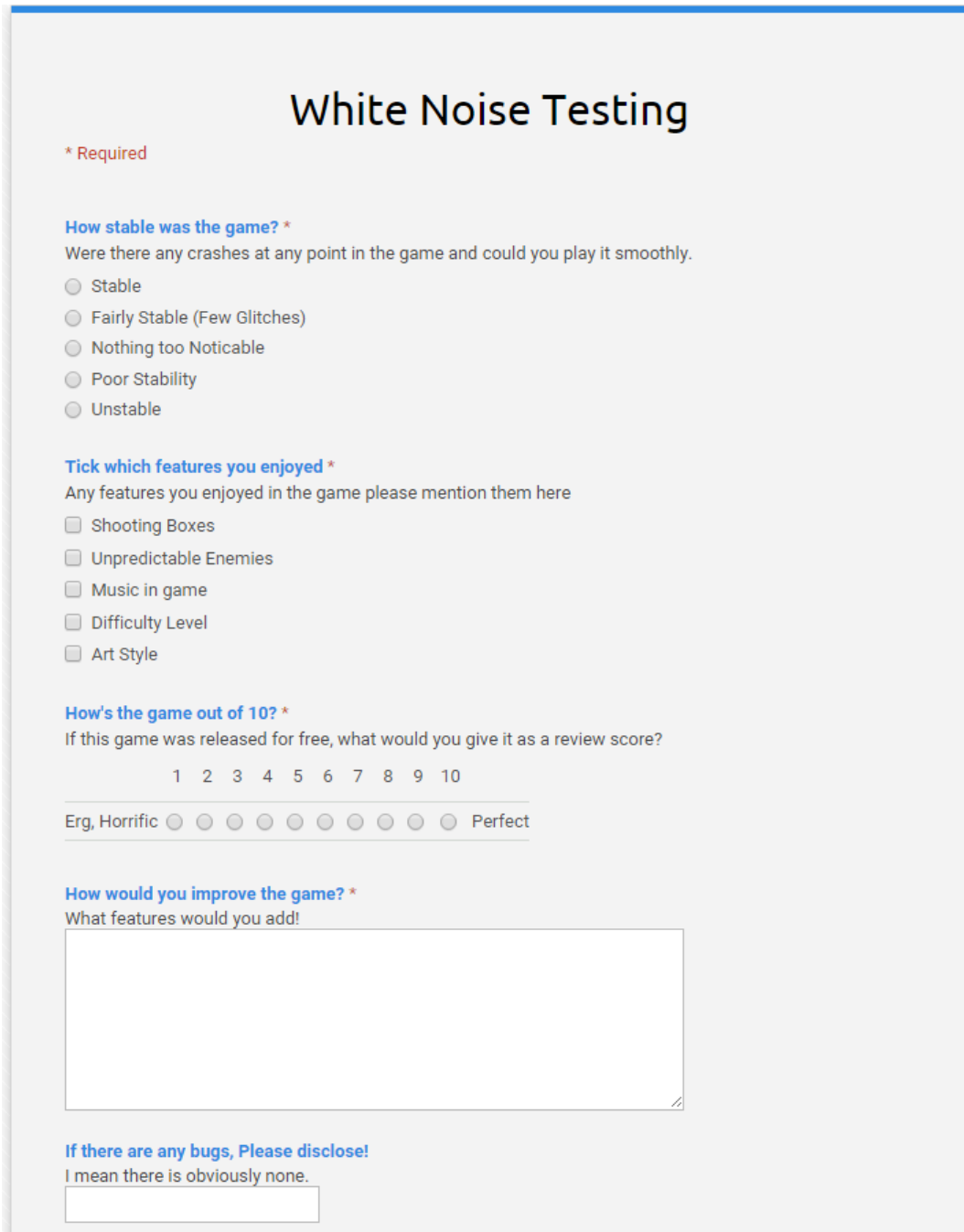
15/01/15 - Doxygen commenting complete.

- Balance to the final boss
- More organisation
- Filters added

Testing

For testing I created a Google Form to the create an interactive way of doing the feedback for the game.

https://docs.google.com/forms/d/1crJu_MR4Fo5afmDtm0mGSbmrSbZAfV7qhf4Ap75FKd8/viewform?c=0&w=1



The screenshot shows a Google Form titled "White Noise Testing". It includes several sections with required questions:

- * Required**
- How stable was the game? ***
Were there any crashes at any point in the game and could you play it smoothly.
 - ☐ Stable
 - ☐ Fairly Stable (Few Glitches)
 - ☐ Nothing too Noticable
 - ☐ Poor Stability
 - ☐ Unstable
- Tick which features you enjoyed ***
Any features you enjoyed in the game please mention them here
 - ☐ Shooting Boxes
 - ☐ Unpredictable Enemies
 - ☐ Music in game
 - ☐ Difficulty Level
 - ☐ Art Style
- How's the game out of 10? ***
If this game was released for free, what would you give it as a review score?
1 2 3 4 5 6 7 8 9 10
Erg, Horrific ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Perfect
- How would you improve the game? ***
What features would you add!
- If there are any bugs, Please disclose! ***
I mean there is obviously none.

I had 4 testers and the top responses were:

How stable was the game? - Stable 100%

Tick the features you enjoyed? - Difficulty Level 100%, Unpredictable Enemies 75%

How's the game out of 10? - Mean Score = 7.75

How would you improve the game? - Best response "I would add a slider for the difficulty (speed of enemies etc.) and the volume. I would add an exit button and menu, as currently there is no easy way to exit it and it is not intuitive."

Any Bugs? - None 100%

Strengths

Luckily I feel like this implementation of a game programming wise is an example of fantastic code. The code I created was constantly tested to be in line with "Level 4" compiler warnings. This means that any performance warnings that the compiler is picked up. Every single one has been fixed which I feel is a fantastic achievement. I feel like my program also did the job of optimising for memory well too. To display images I opted out of going for the computation heavy PNG format and stayed with BMP. PNG's hold information, most of it metadata, which is largely unnecessary for the application that I was creating.

The game also had good gameplay! The game would work with a controller fluently using Xinput which allowed me to add features such as rumble to a really low level API. Reasoning behind using less inheritance in my function was to avoid "Virtual Function Overhead". When looking at assembly code you can see that using virtual functions calls 3 different locations before it finally gets to the correct path. This is less than optimal and decided to weave the classes together with pass by reference and value which seems like a quicker method for such a small program.

Doxygen commenting style has been implemented also in the games code this has ensured my code is really readable in the future and I can access it through a HTML document. this is in (SDL_Game/Docs)

Last but not least the game is fun. One thing with having a limited time to program and create a game you sometimes forget that it has to still fulfill the need of fun. The fun is mainly caused by the erratic nature of the enemies and how they spawn which keeps the experience fresh each time boot up the game.

Weaknesses

The game sadly does not work as fluently with a mouse as I wish it would. The controls do not really translate well and in my pursuit of a different game experience by not using the arrow keys did not work as well. The mouse controls are also a little unresponsive and do take a little wrestling to work.

If I was to redo this then I would use the factory design pattern. Due to time constraints and lack of experience in this pattern be as ambitious.

Conclusion

The game I am over all really pleased with. The game runs smoothly, the controls are intuitive and the gameplay offers a new experience in each playthrough. Although there are places I can identify there are weaknesses in my game I believe this highly optimised code is some of the best I have ever wrote.

Word Count : 2500

References

Third Arcade, 2009. *Steel Empire*. Screenshot. Third Arcade. Available from:
http://www.third.arcadeox.com/g/users/8/7870/genesis_scores/steel-empire-the-1227619497.png
[Accessed 19 January 2015].