'Cross The Street' 2D Arcade Style Game

Name: Jamie Cropley
P Number: P15188432
Module: IMAT1212- C++ Programming
For attention of Lab Tutor: Mr Mohammad Al-Omari (If not available then hand to: Dr Armaghan Moemeni)
Date: 09/05/2016

Summary:

This report entails a detailed description and analysis of the a clone of popular game Frogger but titled Cross Road inherently, it utilises the language of C++ and the SFML media library. The report focuses on the game mechanics, design and testing, the report then goes on to detail personally achieved aspects throughout and reflection on what could have been done better. Concluding with a conclusion and aspects that influenced the design and programming of the game overall.

- The game in its compiled form can be downloaded and played at the following address, however it is quite old now so you may face compatibility issues in Windows: https://jamiecropley.itch.io/crossroad

- Full code for this project can be viewed at: https://github.com/JamieCropley/cpp-university/blob/main/year1/CrossRoadGameInSFML.cpp

Contents:

Heading number	Heading title	Page number
1	Introduction	1
2	The Game	3
2.1	Game Design	3 & 4
2.2	What I have achieved	5
2.3	How the game works	6 & 7
2.4	Tests	8,9,10 & 11
3	Reflection	12
3.1	How I feel I have done	12
3.2	What I would have done differently	12 & 13
4	Conclusive items	14
4.1	List of software used	14 & 15
4.2	Conclusion	15
4.3	Bibliography	16

Total page count: 16

1. Introduction

This is the final assignment for the C++ module taking into account the aspects of object orientated programming and C++ programming throughout, the goal was to create a computer game that was a 'Cross The Street' 2D Arcade Style Game in a similar to the popular arcade game known as Frogger where the goal is, you as a player needs to get across a road and a river with various obstacles put in your path as well as going against a timer all entwined within a scoring system. The SFML (Which stands for Simple and Fast Multimedia Library) documentation was my mostly used source throughout this project especially where I had to look for further details on how such events work within this media library.

2. The Game

2.1 Game Design

I started the game design based on the original version of the game Frogger, I attempted to break down all elements of the game e.g. scoring system, player movement etc…

- *Frogger game Konami 1981*

The initial aspect that stuck out to me at first was the screen size, I decided the lock the game screen size in at 800 x 600 pixels because this is the lowest possible resolution on my computer, therefore it seemed a suitable size so that the game will run OK across many different computers.

Next I decided to divide the screen up into the following areas and try and implement and replicate their behaviour into my game but not in their entirety.

- Screen size: 800 x 600 pixels
- Sound; Music and Sound effects
- Life system
- Player Name
- High score system
- Score system
- Levelling system
- Enemy system (Cars)
- Turtles and logs to get across water
- Timer
- Player
- Holes to initiate a game won system
- Tiles, road, pavement water etc....
- Animations

2.2 What I have achieved

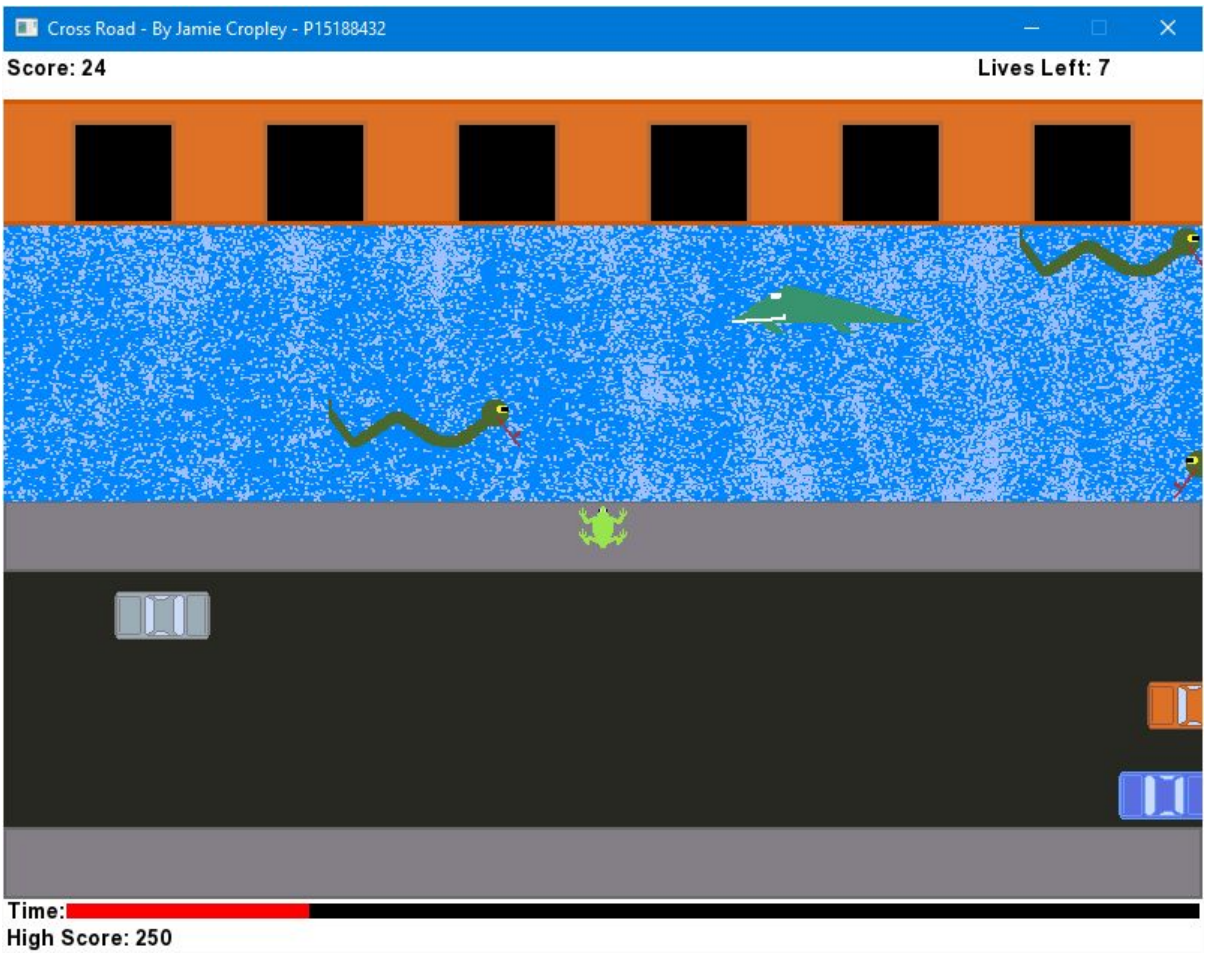
As per the aforementioned in 2.1 I managed to successfully achieve the following elements in my game overall by utlising object oriented programming, c++ and the SFML media library as well as the software known as Aseprite to create all my artwork for my game:

- Screen size: 800 x 600 pixels: Achieved successfully by fully locking into this screen size and ensuring different computers try not to scale up the game in any sense.
- Life system: Achieved successfully but redesigned in replacement of a levelling system and game over system.
- High score system: Achieved successfully which increments as long as the player as lives.
- Score system: Achieved successfully which increments only when the player moves upwards through the level
- Enemy system: Cars, snakes and crocodiles: Achieved successfully but due to collision problems with making the logs and lilies work successfully I ended up replacing these with two different types of new enemies crocodiles and snakes which worked the same way as cars. I also had to make the water not kill the frog in this respect.
- Timer: Achieved successfully.
- Player: Achieved successfully.
- Holes to initiate a game won system: Achieved successfully.
- Tiles, road, pavement water etc...: Achieved successfully but did not use tiles, or grid dynamics for the game area, instead precise positioning of predetermined sized images on the screen, I found this beneficial in the long run when setting game boundaries, I believe a tiled grid system would of been more complex.

I believe I achieved the game design I initially set out to however with some aspects of it I had to come up with more creative solutions because I could not figure out the right code to implement and make it work in the game.

I achieved knowledge of how a library can work and interact with the c++ language and therefore how it can extend it as such.

2.3 How the game works



- *Cross Road - Jamie Cropley - P15188432 - 2016*

- Screen size: 800 x 600 pixels: This works by utilising the SFML event for Render window, it draws a gui screen to the computer which then contains all the games elements therein. Other various code sets the scaling of such elements and manipulates them where and when necessary as such, mostly ‘window.method’ was used towards the end of my code to put everything in its place and draw it the screen as necessary towards the end of my code in the game.
- Life system: This works by using logic and math, you as in the player gain one extra life if you reach the black windows at the top of the screen but if you die upon collision with a crocodile, car or snake you get one lesser life, if you end up with zero lives the high score resets along with other aspects of the game. Consistently upping your lives by getting to the top of the game keeps your high score intact therefore creating an addictive method to replace the levelling system in the original game.

- High score system: On each play through and death your score gets added to the high score, however if you have no lives left this then becomes reset.
- Score system: You gain 2 points per press of the up arrow key on the keyboard to a maximum of 54. You gain 200 points if you the player collide with any of the black boxes at the top of the screen.
- Timer: This counts down in seconds from 30 seconds by using the SFML event of Clock and represented graphically through a progress bar, the player, you lose a life upon the time running out.
- Player: The up, down, left and right keys on the keyboard move the player, I disabled the ability to be able to hold the key down so that you have to keep pressing / tapping the key, this I believe is a simple thing to achieve that makes the game that much more challenging.
- Holes to initiate a game won system: These are drawn as separate squares under a transparency layer of the image ‘topmud.png’ they act similar to the way that the collisions work in the rest of the game, where it increments the scores and such therefore resetting the player back to the beginning of the game but also giving the player one additional life as well as 200 points, this is all in place of a levelling system where the lives act as a levelling initiative.
- Tiles, road, pavement water etc...: These have no effect on gameplay other than preventing the player going off screen through the setting of player boundaries.

2.4 Tests

Bug fixing:

Start of game:

1.) Does the player always start in the same position?
2.) Do lives always start at 6?
3.) Does the progress bar start not full?
4.) Does the Score start at 0?
5.) Does the High Score start at 0?

Gameplay:

1.) If you press up arrow on keyboard does the player move upwards by specified movement amount?
2.) If you press down arrow on keyboard does the player move downwards by specified movement amount?
3.) If you press left arrow on keyboard does the player move to the left by specified movement amount?
4.) If you press right arrow on keyboard does the player move to the right by specified movement amount?
5.) Are all top, left, right and bottom boundaries intact?
6.) Does the scoring and high score system work in their entirety?
7.) Does the timer work?
8.) Do the player and enemy collisions work?
9.) When the black squares are collided with by the player at the top of the screen does everything reset and an additional life added to the player?

End of game:

1.) Can you end the game by closing the game window successfully?

Start of game:

Game Mechanic	Does it work as intended?	If no what happens?	Expected fix?
Does the score start at zero?	YES	N/A	N/A
Is the progress bar empty?	YES	N/A	N/A
Does lives left start at 6?	YES	N/A	N/A
Does the high score start at 0?	YES	N/A	N/A
Does everything draw to screen especially in terms of layer ordering?	YES	N/A	N/A
Does the player start in the same position?	YES	N/A	N/A

Gameplay:

Game Mechanic	Does it work as intended?	If no what happens?	Expected fix?
If you press up arrow on keyboard does the player move upwards by specified movement amount?	YES	N/A	N/A
If you press down arrow on keyboard does the player move downwards by specified movement amount?	YES	N/A	N/A
If you press left arrow on keyboard	YES	N/A	N/A

does the player move to the left by specified movement amount?			
If you press right arrow on keyboard does the player move to the right by specified movement amount?	YES	N/A	N/A
Are all top, left, right and bottom boundaries intact?	YES	N/A	N/A
Does the scoring and high score system work in their entirety?	YES	N/A	N/A
Does the timer work?	YES	N/A	N/A
Do the player and enemy collisions work?	YES	N/A	N/A
When the black squares are collided with by the player at the top of the screen does everything reset and an additional life added to the player?	YES	N/A	N/A

End of Game:

Game Mechanic	Does it work as intended?	If no what happens?	Expected fix?
Can you end the game by closing the game window successfully?	YES	N/A	N/A

3 Reflection

3.1 How I feel I have done

I feel like I could of done a lot better on this assignment especially in terms of implementing a full on clone of the Frogger game, additionally I do not believe my code was well organised as I ended up putting all my code in one main function due to time limitations, I did plan to at least separate it into different functions and classes therein. I feel like I have still exhibited Object Orientated Programming through

Overall I would say I have successfully completed the majority of the assignment requirements although not completely, I have especially completed most of the requirements specified under ‘Tasks to be undertaken’ on the assignment brief.

3.2 What I would have done differently

The following elements I would of added to the game if I had more time:

- Animation, in way of utilising the software Aseprite to create framed animated and possibly skeletal like movements, e.g. if key pressed up on player then animate legs moving forwards.

- Sound effects, I created these sound effects which are in the asset folder, however I ran into to many problems when trying to implement them into my game in terms of the code and SFML events.

- Soundtrack / music score, I would of liked to take the time to write a musical soundtrack to the game, in terms of making the game more engaging especially in regard to a potential levelling system, where the music would of got more intense the harder each level got.

- Game over and pausing system in lieu of a menu system, I believe a game over system would of been a good idea to implement because the game does not give the player a break from playing, in terms of a menu system it would of been good to also be able to pause the game, the player can currently only stop playing if they close down the game completely, I did find minimising the game to the task bar seemed to pause it however.
- Levelling system that added more ways to die and increased speed of enemies and such, this again linking to the music I wanted to implement into the game. I would

have also made the enemies in the game move faster to make the game more challenging per each level up in the game.

Two elements I wanted to add to the game that I did not get time to do:

- Oxygen system for the water, I wanted to make it so the player could stand on the water but only for a limited time, so there would of been a countdown timer that acted like an oxygen tank perhaps in the way of a blue progress bar that went down the longer the frog or player was in the for therefore dying when it reached 0 / progress bar went fully down.

- I want to add a random UFO to the game much like a bouncing ball that the player had to avoid, the UFO would randomly appear say every 10 levels or so adding another layer of difficulty to the game overall.

Everything else I would have done better:

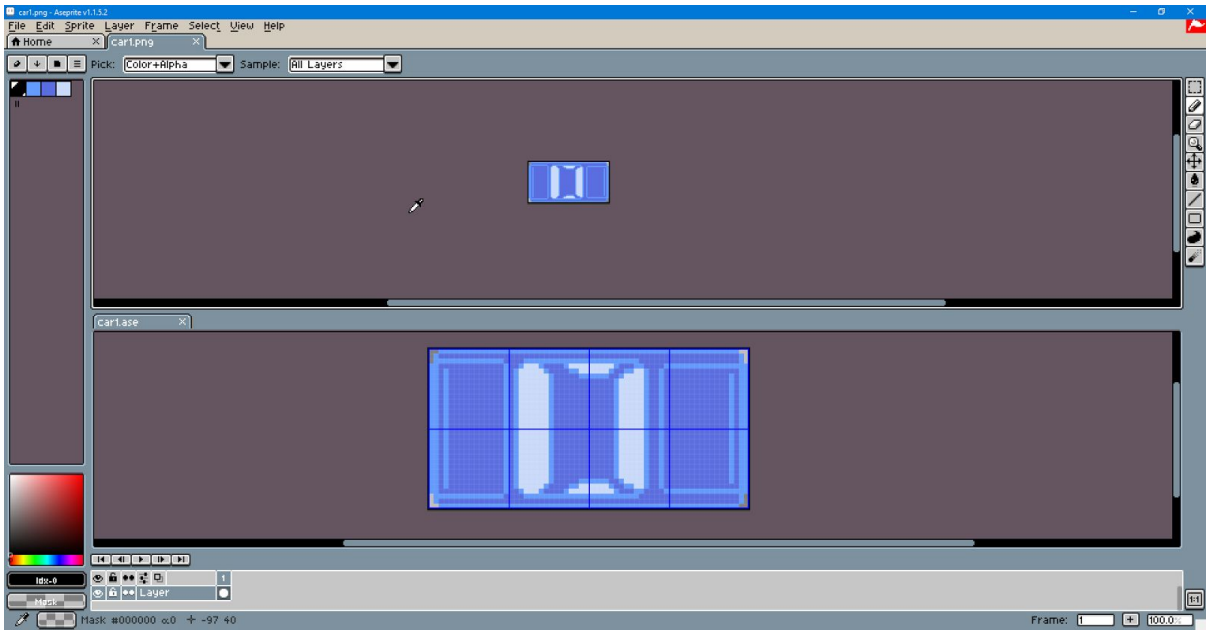
- Organisation of code and file structure, as well as classes, this was very lacking in my code although I have still tried to make it organised by grouping similar things together and commenting it appropriately.

- Better game design and artwork, if I had more time I would of done this a lot better again with the addition of animations to mostly everything.

4. Conclusive items

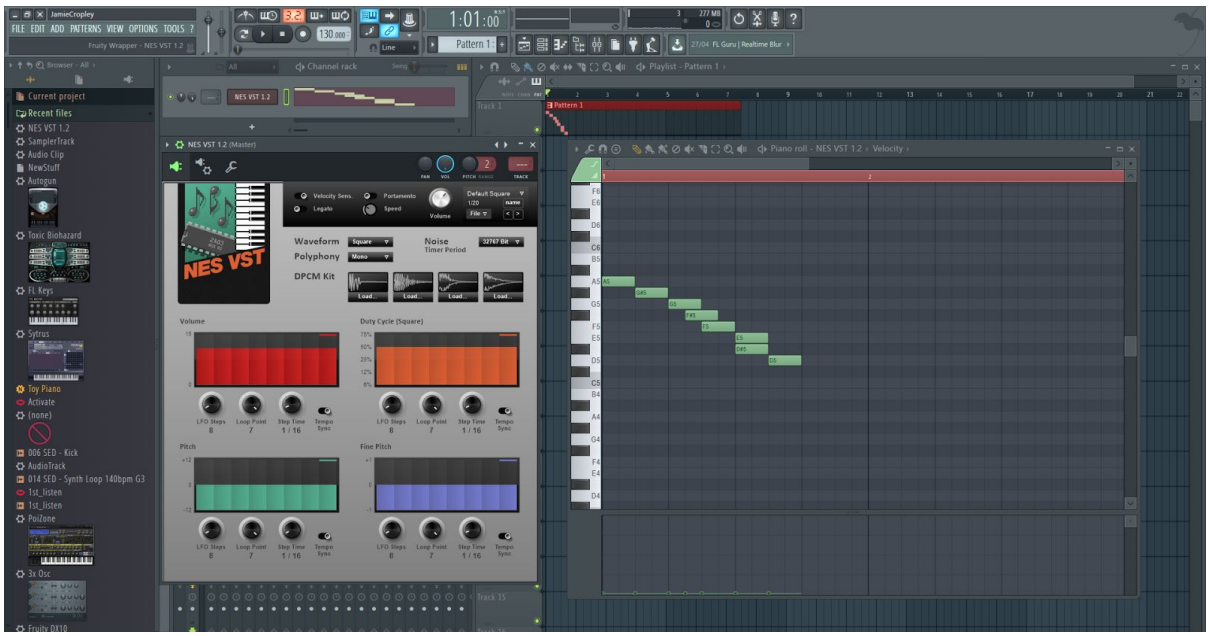
4.1 List of software used

Aseprite:



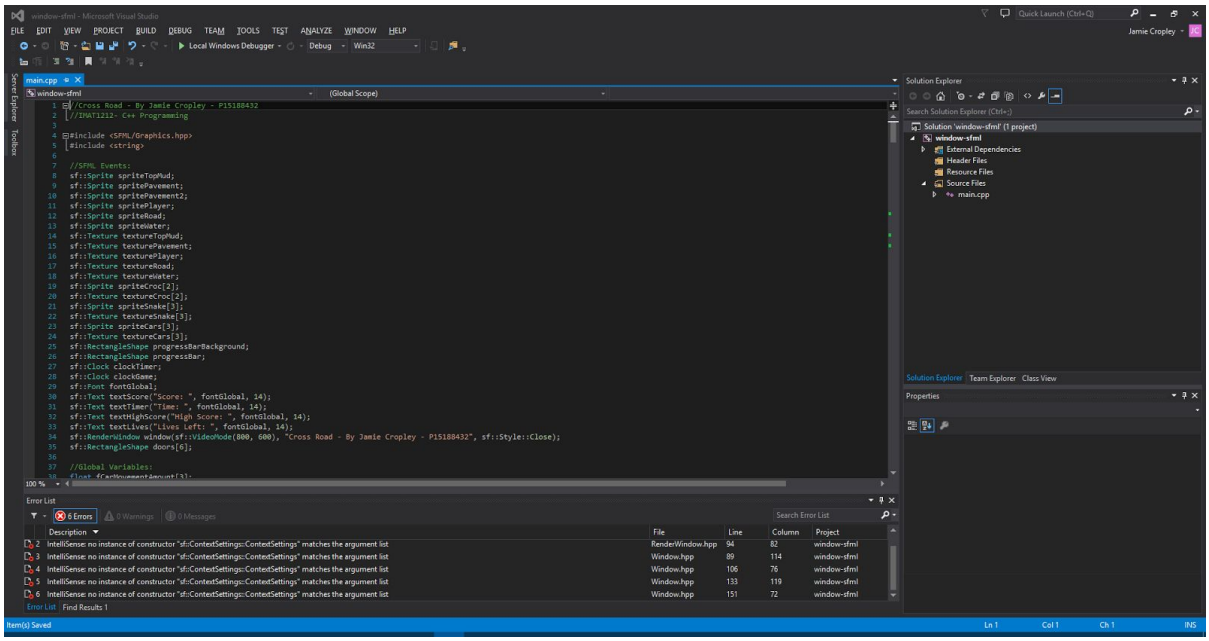
I used this software to draw all my own art work for the game. The above example is how the car were drawn using some formation of pixel art.

FL Studio:



I used this software utilising a virtual instrument within the software which is like a plugin to create basic sound effects and music for my game, however I ended up not using them due to coding problems.

Visual Studio 2013 Professional with Update 5:



I used this software to create the code for my game and link it to and utilise the SFML library therein.

4.2 Conclusion

In conclusion I did not fully implement an exact clone of the original frogger game due to time limitations and coding problems I encountered, however I believe my creative solutions to making some formation of my own fully functional different version of the frogger game turned out very well with everything working and zero bugs.

4.3 Bibliography

Websites:

SFML (2016) *Documentation of SFML 2.2.* [Online] Available from: http://www.sfml-dev.org/documentation/2.2/ [Accessed 08/05/2016].

SFML (2016) *Simple and Fast Multimedia Library.* [Online] Available from: http://www.sfml-dev.org/index.php [Accessed 08/05/2016].

Classikgames.com, Konami, Sega/Gremlin (1981) *Frogger*. [Online] Available from: http://www.classikgames.com/frogger.html [Accessed 08/05/2016].

Software:

David Capello (2016) *Aseprite*. [Software] Version 1.1.5.2

Image Line (2015) *FL Studio*. [Software] Version 12.2

Matthew Montag (2016) *NES VST*. [Software] Version 1.2

Microsoft (2016) *Visual studio 2013 Professional with Update 5*. [Software] Version 4.6.01038