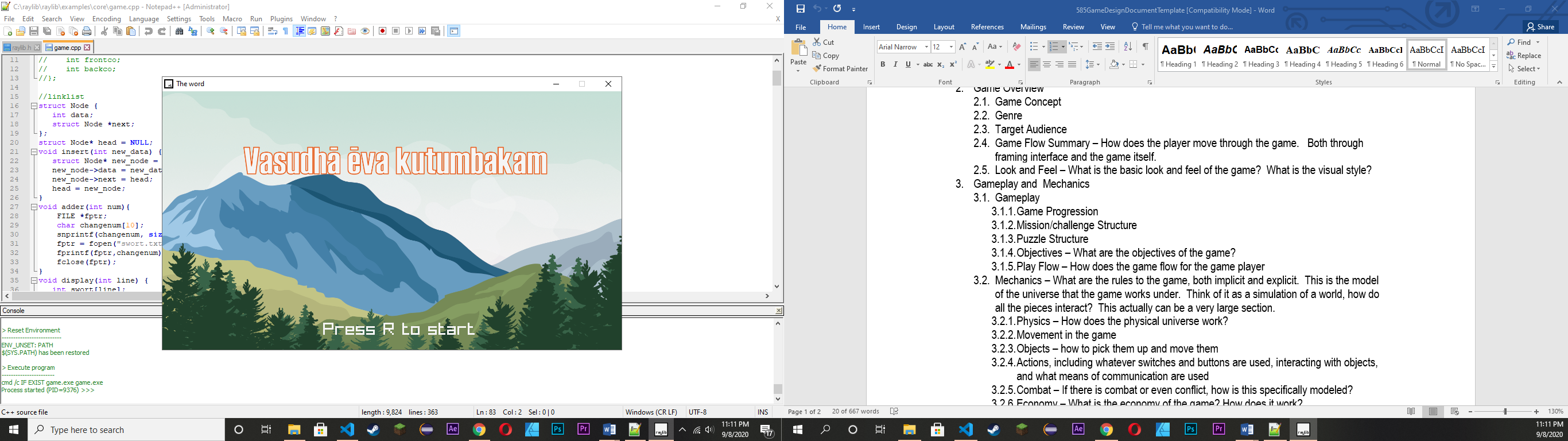
**The World (Avoid Asteroids Game)**

**Introduction**

It is a game about a boy who got teleports into space from thin air. It is said that our universe is actually 11 dimension, space is 4 dimension, 3 normal directional axis and 4th one in space time. It is said there are a lot of gravity anomalies which happen in our space. Gravity can change space time. There is a myth about Bermuda triangle where gravity concentration is high so due to anomalies, stuff or even whole plan get teleported into unknown coordinate in the universe.

**Game Overview**



**Game Concept:**

It is about a boy on a school field trip. Whose bus suddenly get teleported into unknown corner of the universe and to survive he need to avoid getting hit by an asteroid.

**Genre:**

Classic Space Arcade game.

**Target Audience:**

Simple causal gamer.

**Game Flow Summary – How does the player move through the game. Both through framing interface and the game itself:**

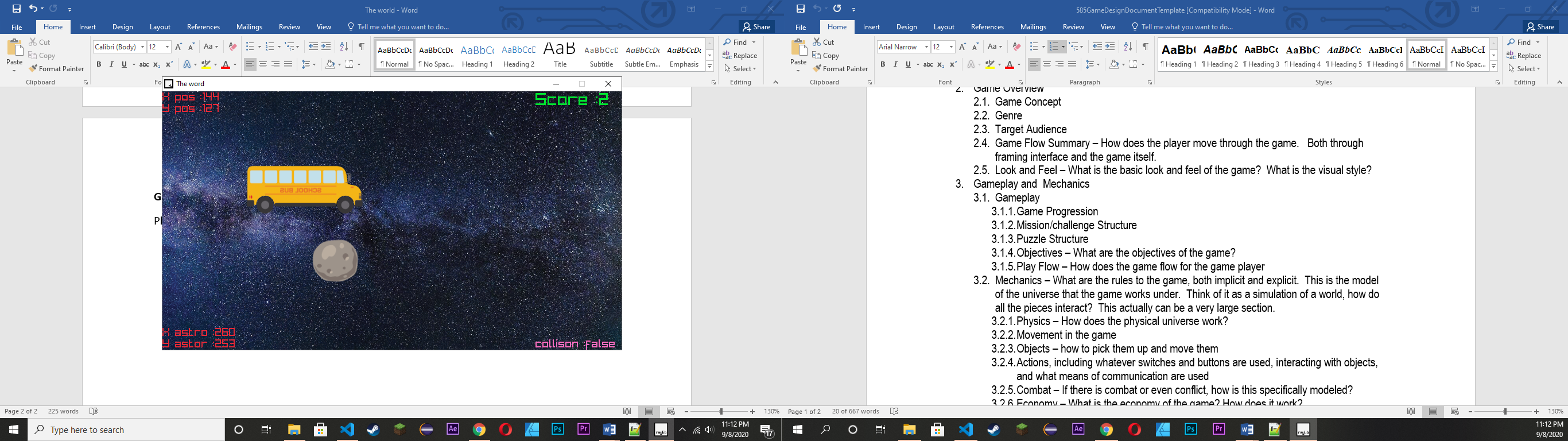
After story player can just need to press “S” to start game. Player need to press Keyboard UP, Down, Right and left to control the bus.

The background music automatically start playing and so to stop it player can press “P” either pause or play the music.

**Features**

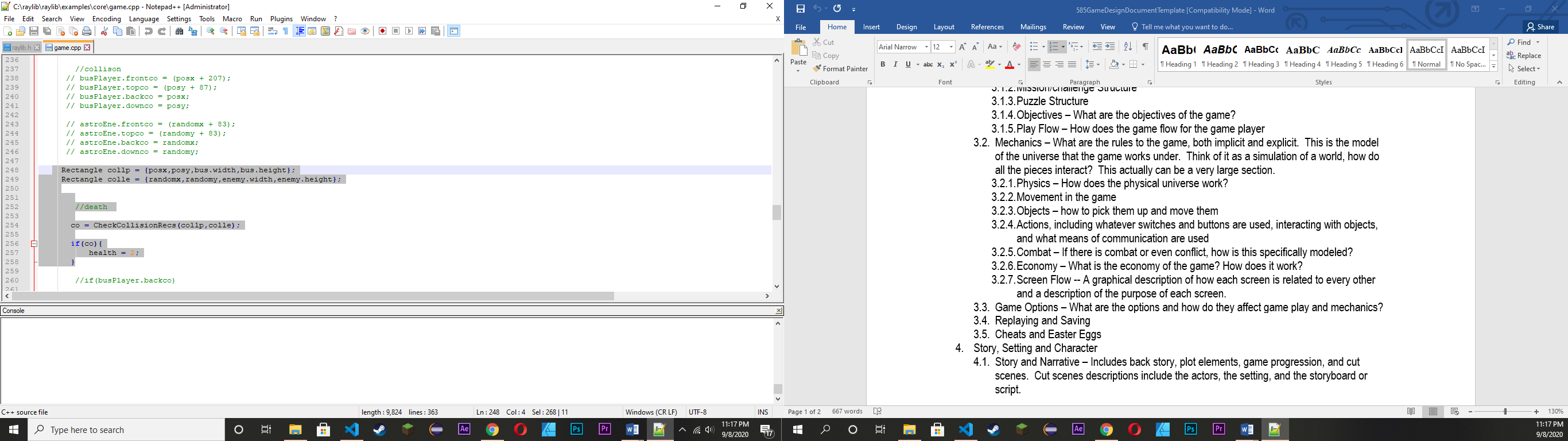
**Game Progression:**

Player just need to avoid asteroid and gain as much high score as possible.

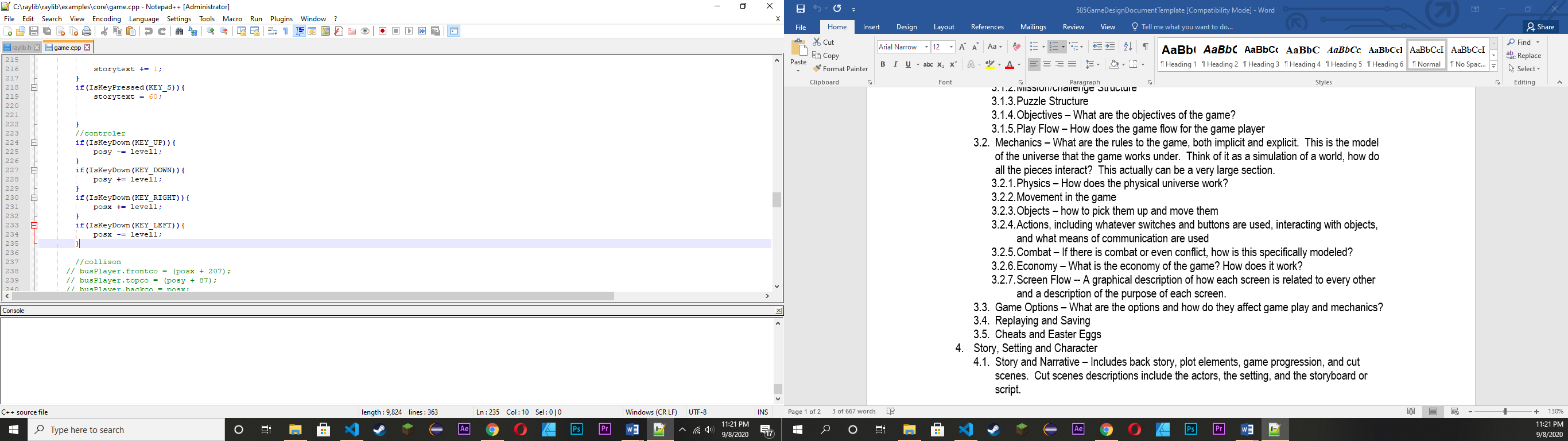


**Physic System:**

It is about a game in space so physic system is pretty straight forward. Player can move in this 2D space freely. For collision we draw a rectangle around both player and asteroid and when the two rectangle collide we pass true and the player die.



Player movement.



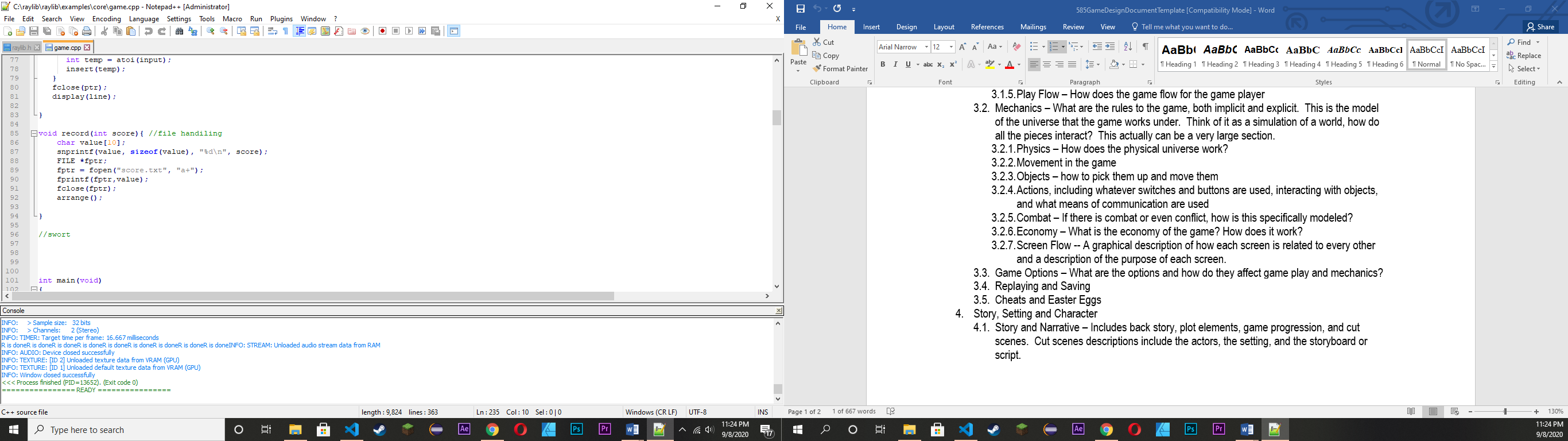
**Game Options:**

There are quit few option like

* “P” Key to pause and play background music.
* “H” Key to save your score in file.
* “S” Key to start game.
* “SPACE\_BAR” Key to go to next page of the story.
* “M” Key to restart the game after you die.

**Saving:**

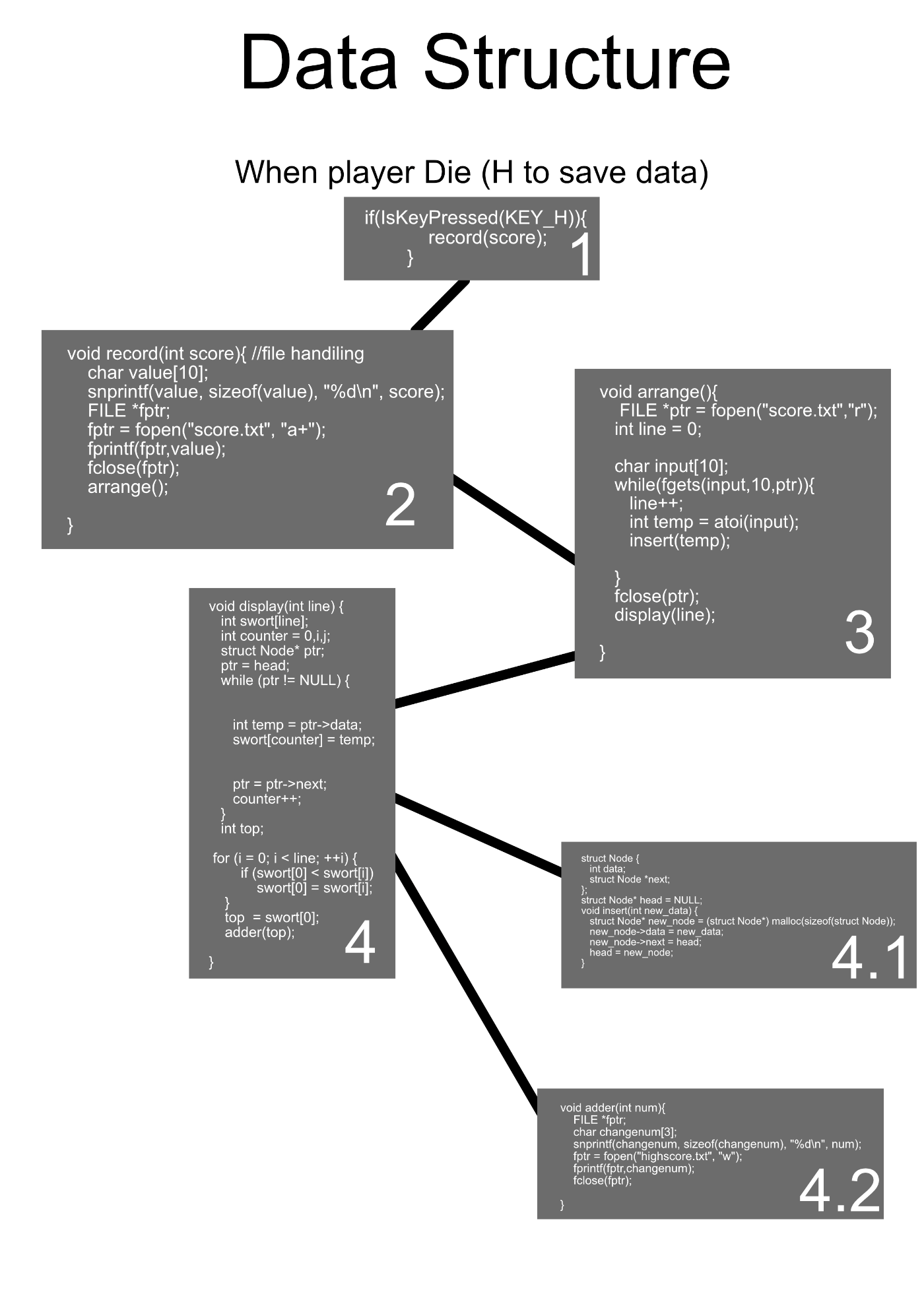
When you press “H” your score will automatically save in score.txt file.



**Data Structure**

For this project we are using Simple link list to store our data and we use that value again to find the highest score and print it into our file system.

First when player Die we ask them to “Press H to save Score”. If player press H , the score will save in a variable which will now put in another function call to save in it file format “score.txt”.



1. When player Die, This function called another no 2 function to save data.
2. This function write score data into score.txt file.
3. This function reopen the score file and retrieve data and line value. Its call function 4.1 to store data into linked list form.
4. This function re-output linklisted data and sort highest value at one array 0 or first index.
5. It store incoming data into Linklist node.
6. This function write highest value form link list into another text file “highest.txt”.

our record function, store it in file format and again call another function. Which reread the file and insert our value in linked list format. After our data are in linked list we find the highest value and rewrite it in another file name “highscore.txt”.

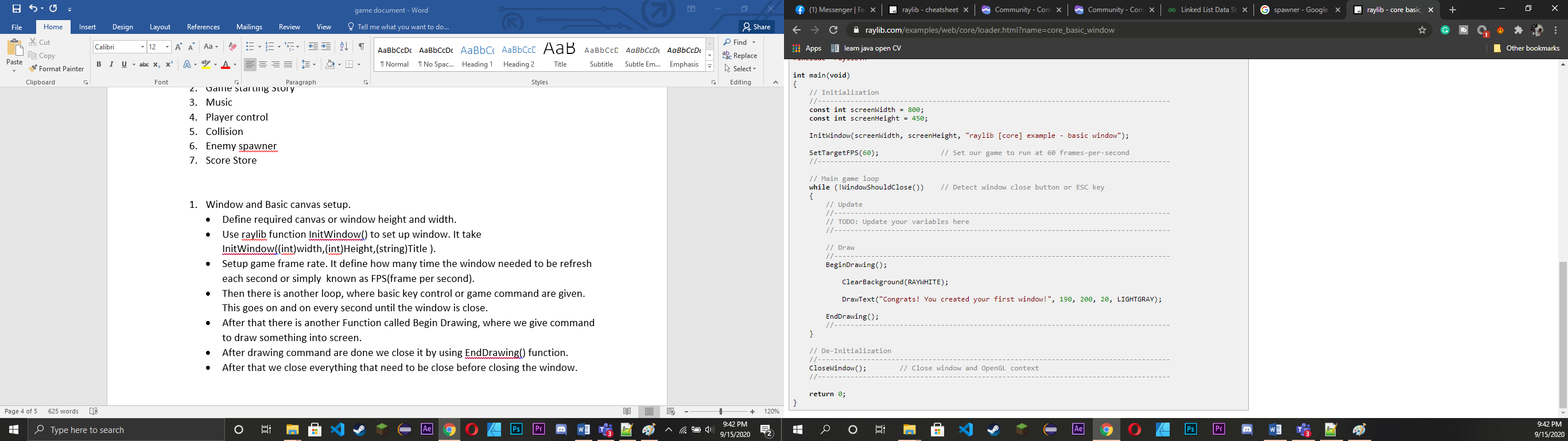
**Algorithm**

For, this game we are using a C/C++ game framework called Raylib.

For simplicity we are going to divided it into different section.

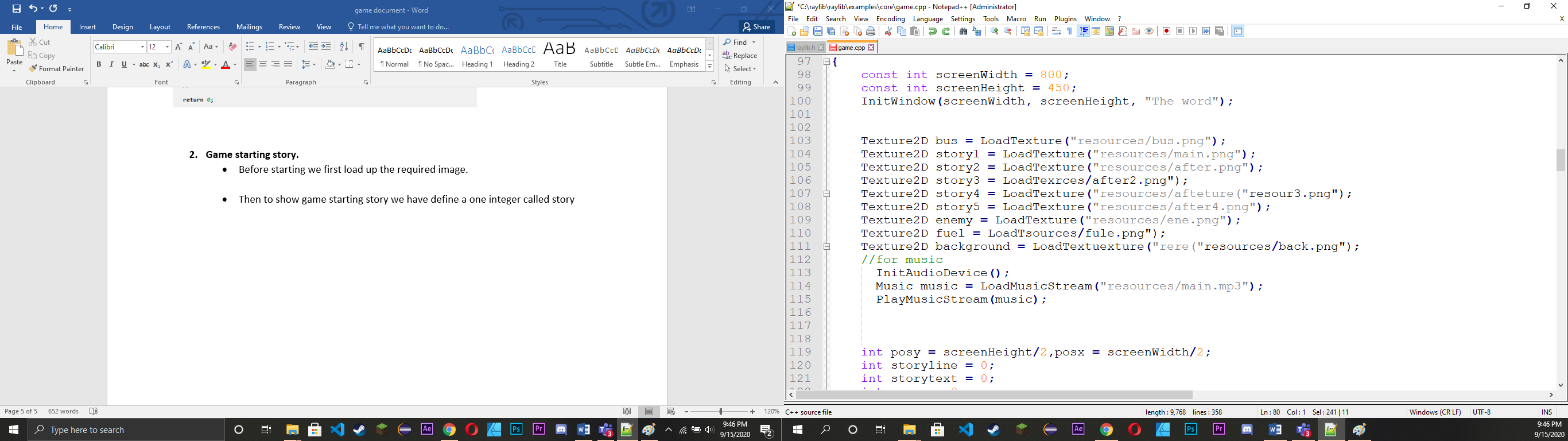
1. Window and basic canvas setup
2. Game starting Story
3. Music
4. Player control
5. Collision
6. Enemy spawner
7. Score Store
8. Game restart
9. **Window and Basic canvas setup.**

* Define required canvas or window height and width.
* Use raylib function InitWindow() to set up window. It take InitWindow((int)width,(int)Height,(string)Title ).
* Setup game frame rate. It define how many time the window needed to be refresh each second or simply known as FPS(frame per second).
* Then there is another loop, where basic key control or game command are given. This goes on and on every second until the window is close.
* After that there is another Function called Begin Drawing, where we give command to draw something into screen.
* After drawing command are done we close it by using EndDrawing() function.
* After that we close everything that need to be close before closing the window.

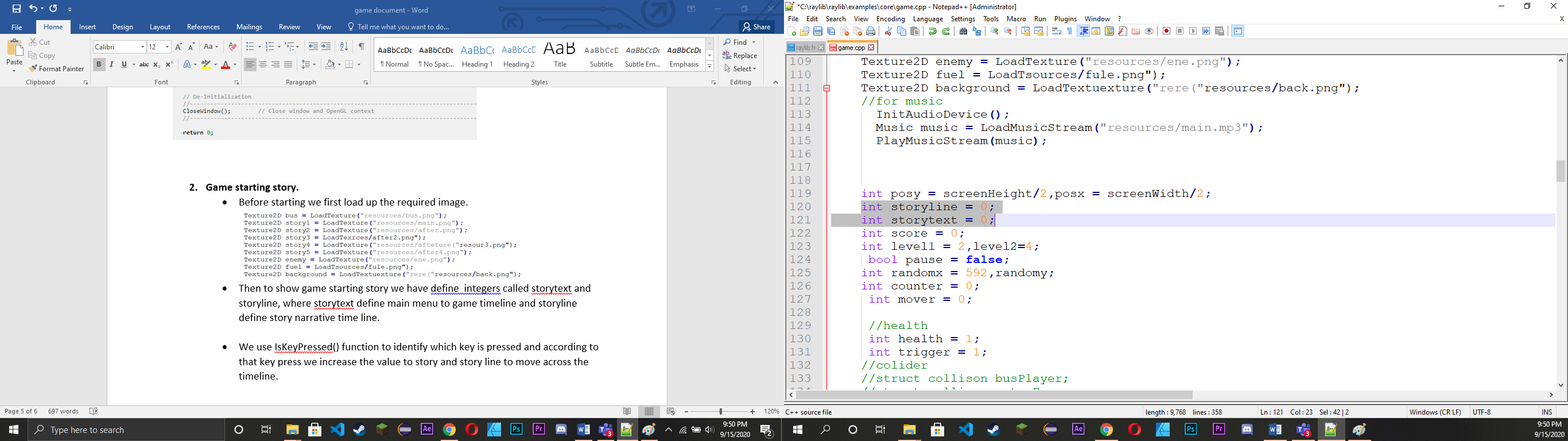


1. **Game starting story.**

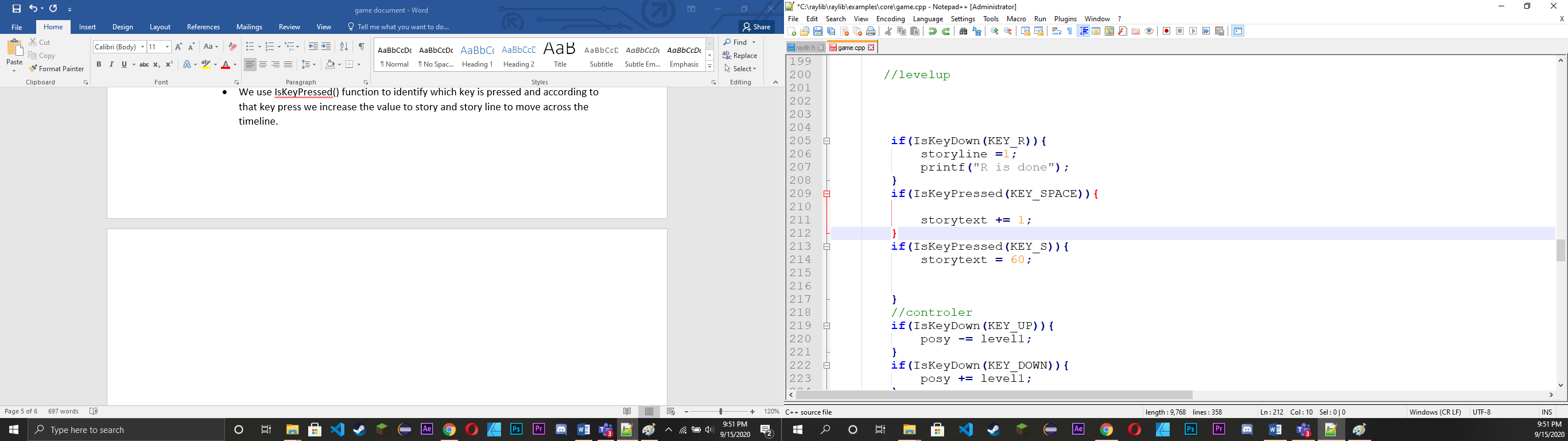
* Before starting we first load up the required image.



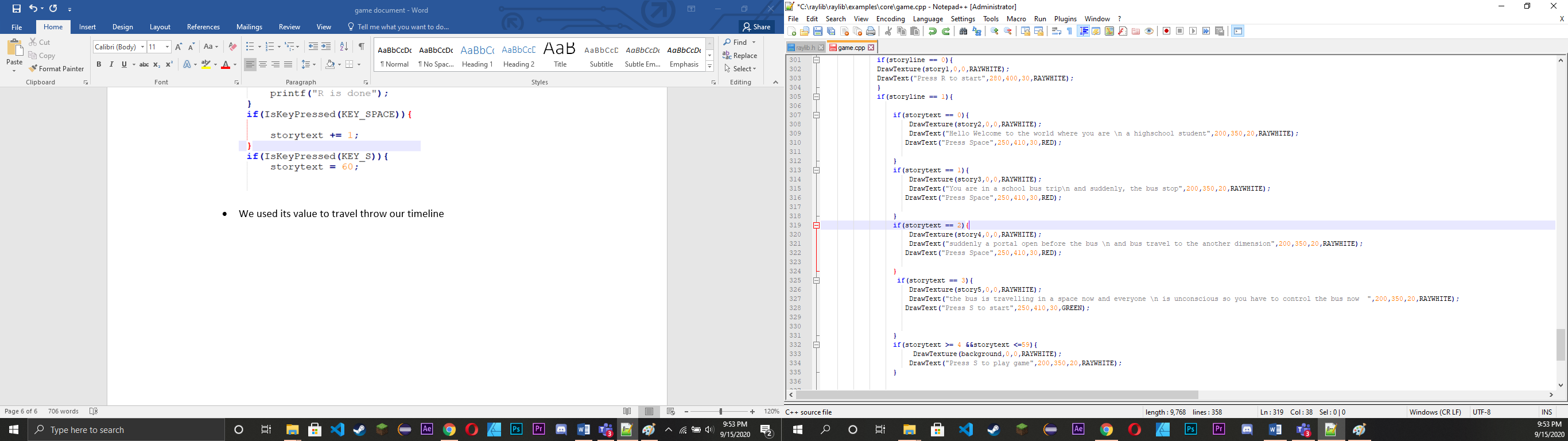
* Then to show game starting story we have define integers called storytext and storyline, where storytext define main menu to game timeline and storyline define story narrative time line.



* We use IsKeyPressed() function to identify which key is pressed and according to that key press we increase the value to story and story line to move across the timeline.

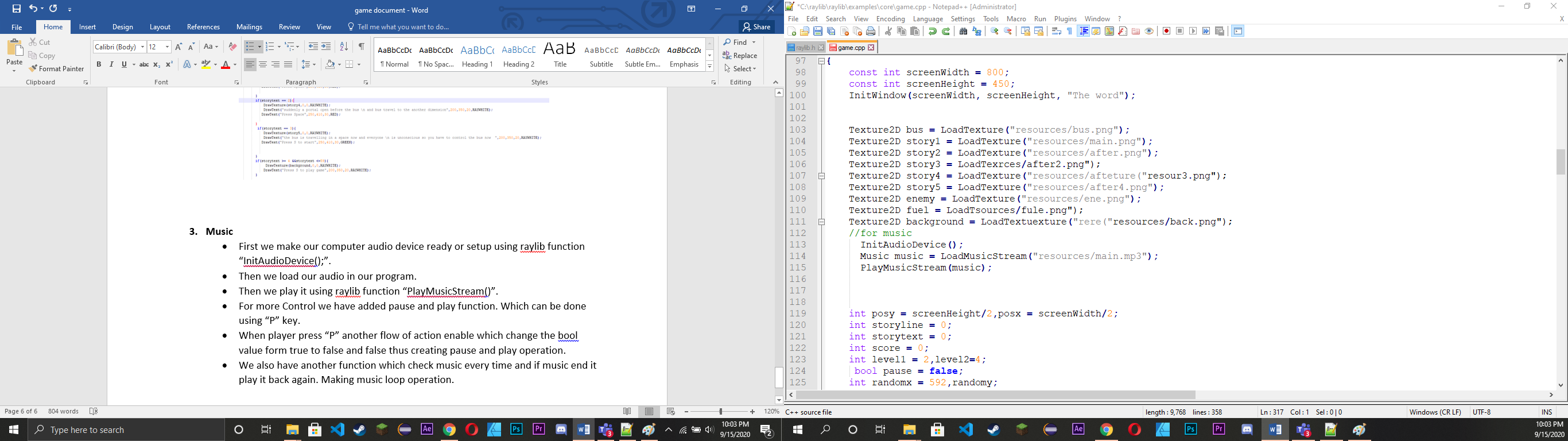


* We used its value to travel throw our timeline

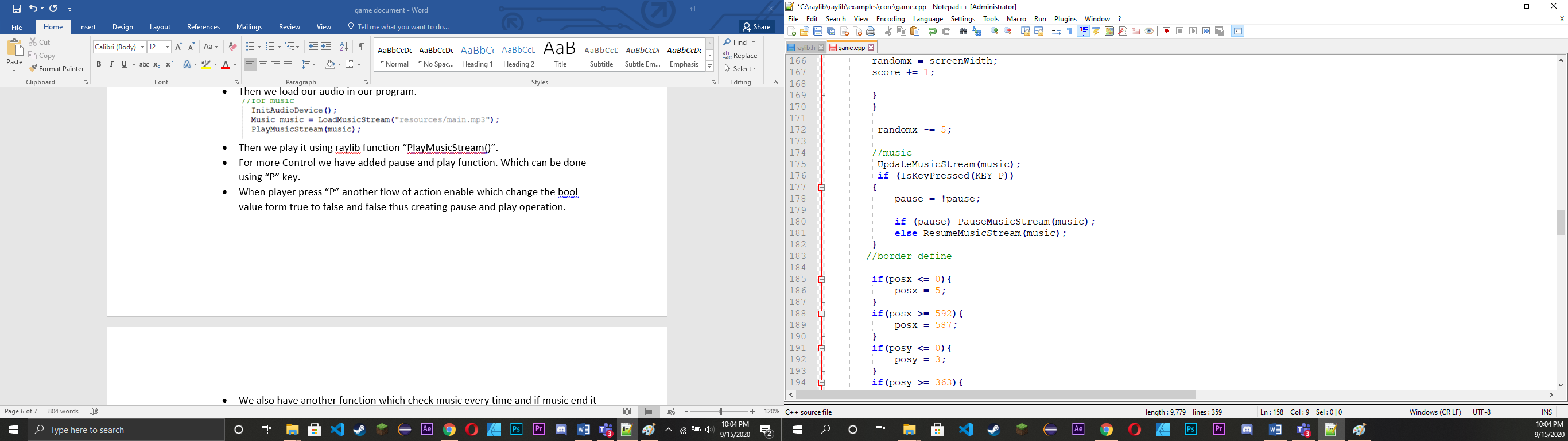


1. **Music**

* First we make our computer audio device ready or setup using raylib function “InitAudioDevice();”.
* Then we load our audio in our program.

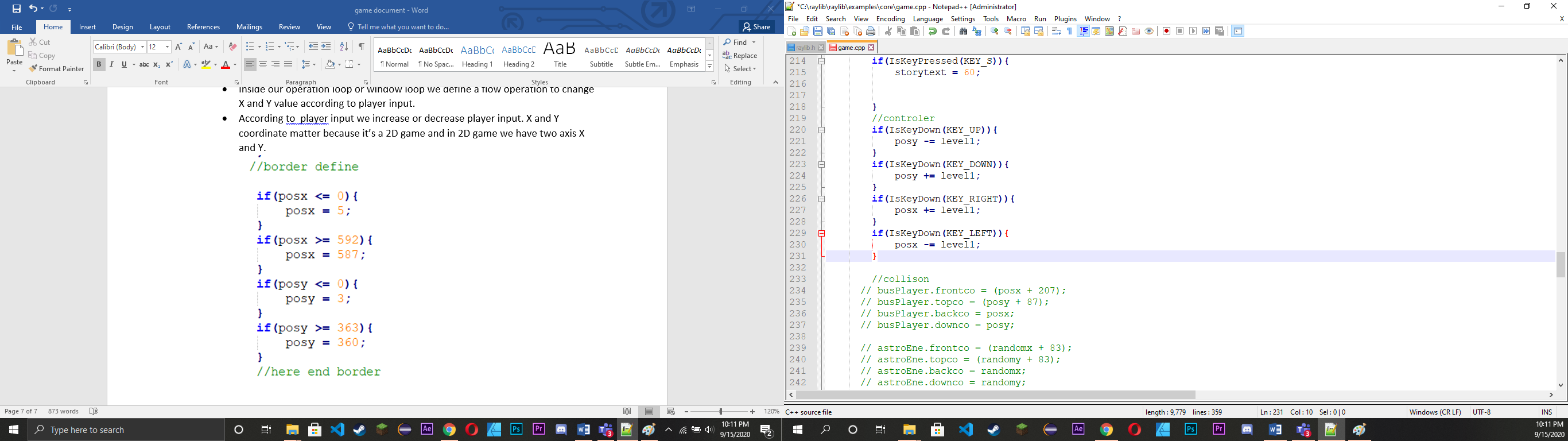


* Then we play it using raylib function “PlayMusicStream()”.
* For more Control we have added pause and play function. Which can be done using “P” key.
* When player press “P” another flow of action enable which change the bool value form true to false and false thus creating pause and play operation.
* We also have another function which check music every time and if music end it play it back again. Making music loop operation.



1. **Player control**

* First we define player X and Y coordinate . It is usually int.
* Inside our operation loop or window loop we define a flow operation to change X and Y value according to player input.
* According to player input we increase or decrease player input. X and Y coordinate matter because it’s a 2D game and in 2D game we have two axis X and Y.



1. **Collision**

**Here we have two type of collision**

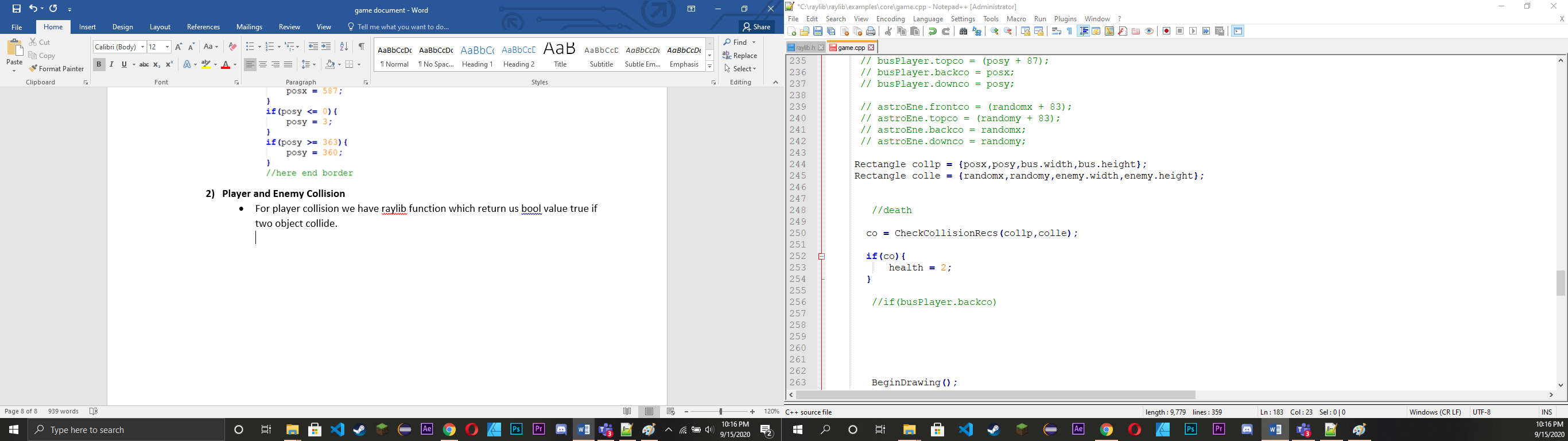
1. **Border collision**

* For border collision, we basically track player X and Y coordinate and when its outside of window area , we reinitialized it value to its border value. Thus creating border collision effect.



1. **Player and Enemy Collision (Death)**

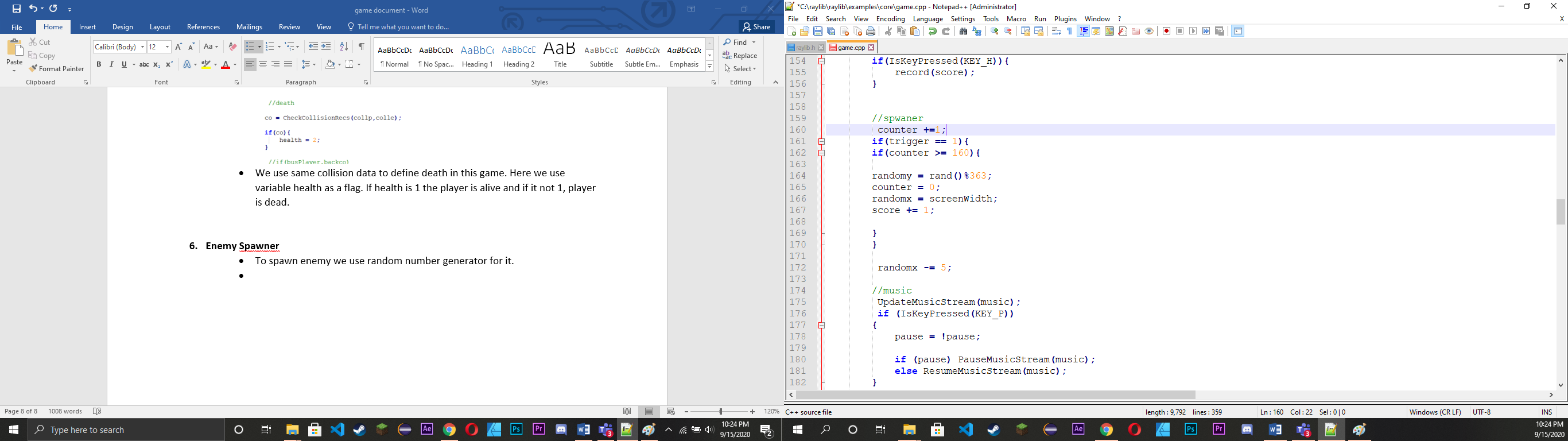
* For player collision we have raylib function which return us bool value true if two object collide.
* We draw rectangle around both enemy and player here and if two rectangle collided , The function return true value.



* We use same collision data to define death in this game. Here we use variable health as a flag. If health is 1 the player is alive and if it not 1, player is dead.

1. **Enemy Spawner**

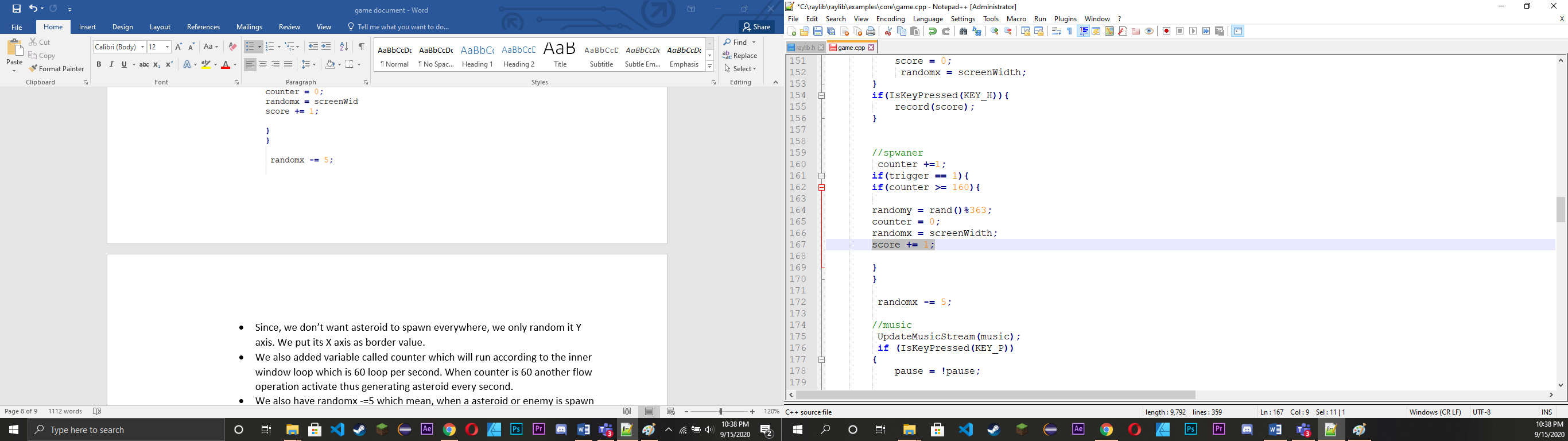
* To spawn enemy we use random number generator for it.



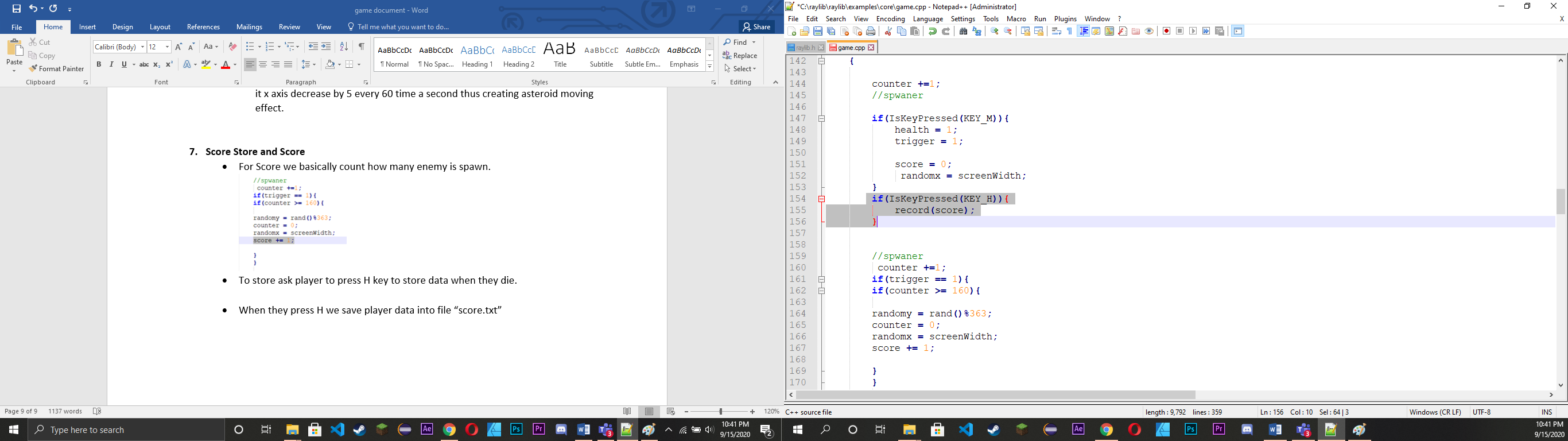
* Since, we don’t want asteroid to spawn everywhere, we only random it Y axis. We put its X axis as border value.
* We also added variable called counter which will run according to the inner window loop which is 60 loop per second. When counter is 60 another flow operation activate thus generating asteroid every second.
* We also have randomx -=5 which mean, when a asteroid or enemy is spawn it x axis decrease by 5 every 60 time a second thus creating asteroid moving effect.

1. **Score Store and Score**

* For Score we basically count how many enemy is spawn.



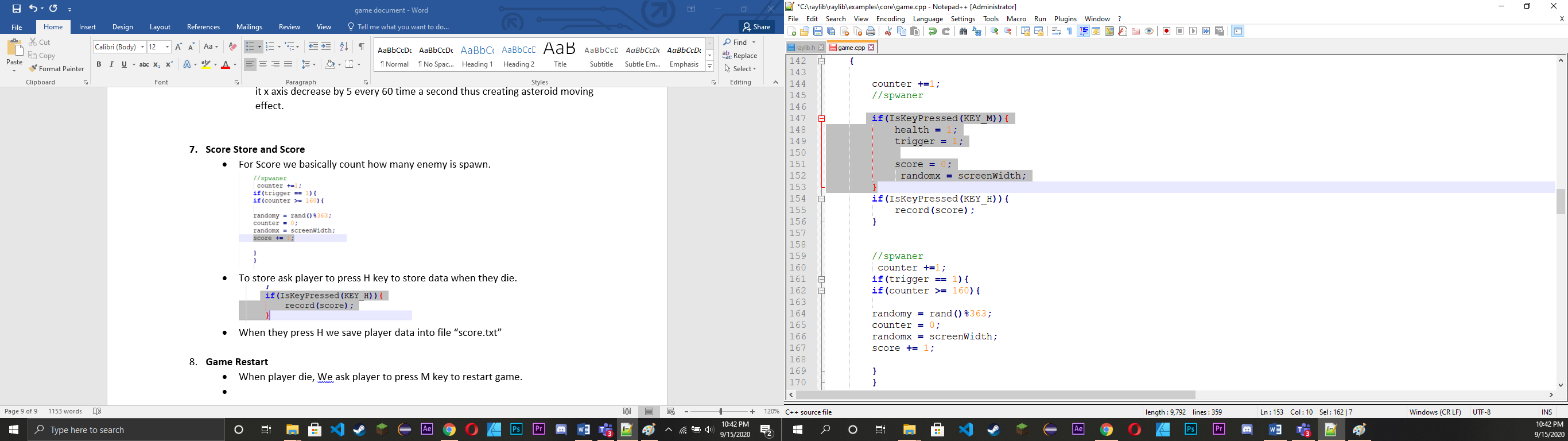
* To store ask player to press H key to store data when they die.



* When they press H we save player data into file “score.txt”

1. **Game Restart**

* When player die, we ask player to press M key to restart game.



* When game is restart we reset everything that are related to game operation like health which define player health, Trigger which is flag which stop and start enemy spawner and Score value.

**Conclusion**

Learning and making game of our own has always been a dream and by this project we understand many things about game design and creation.

By making this game, we have learn a new framework “Raylib”. This project also helped us to understand team work and time management as well. Our goal for this project was to make graphical and 2d game, by looking at the end result we accomplish our goal.