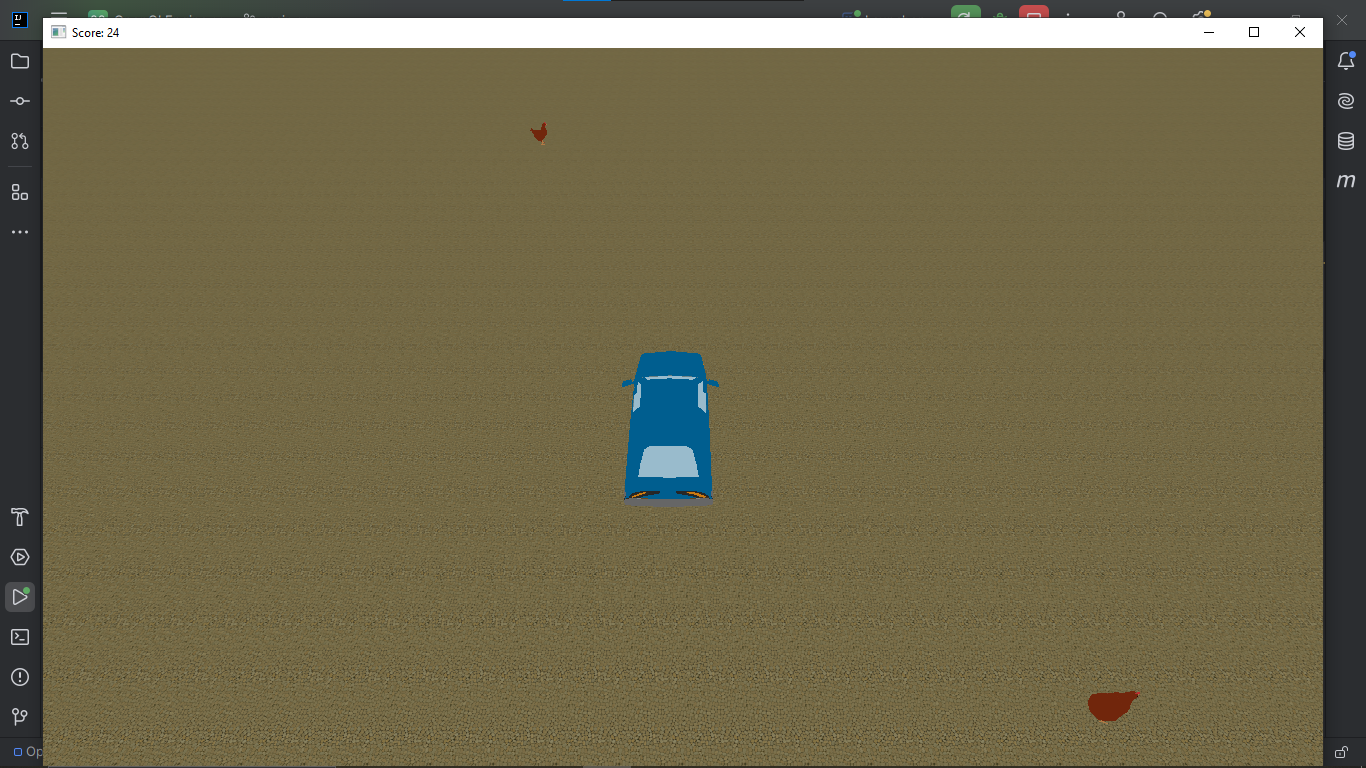
**Saqlain Haider (SP24-BSE-109)**

**OpenGL Engine**

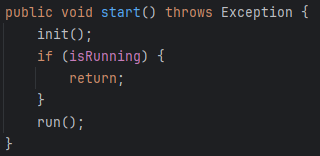
**Overview:**



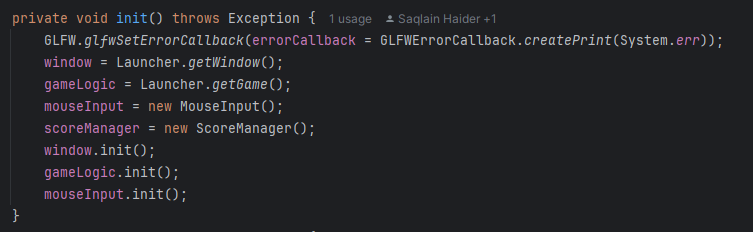
1. The engine is capable of rendering 3D graphics and Directional Light.
2. A simple game is made inside the engine.
3. Ability to detect key events, mouse events etc.
4. Collision detection between valid entities.



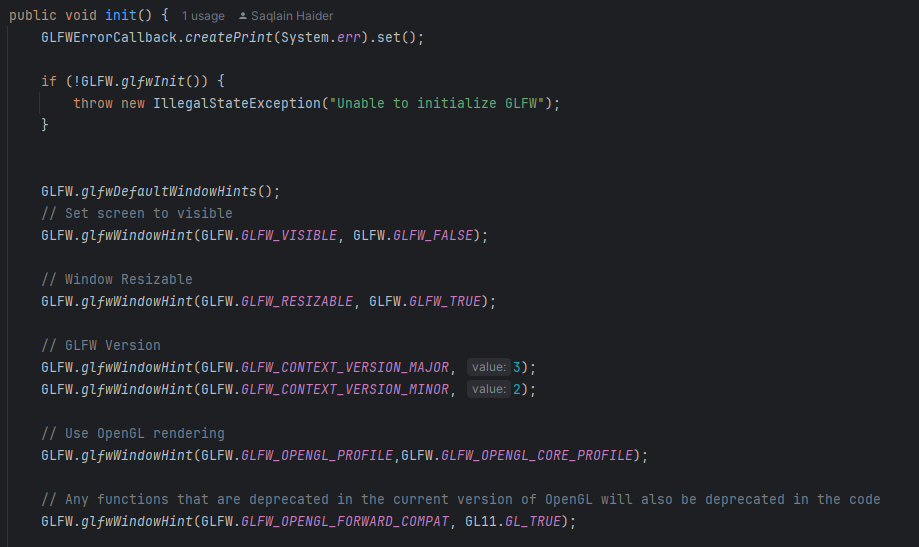
The launcher class creates the display and the instance of the game but the EngineManager class is responsible for displaying the window and the game. The start method is called inside a try catch block as it throws an exception in the EngineManager Class.

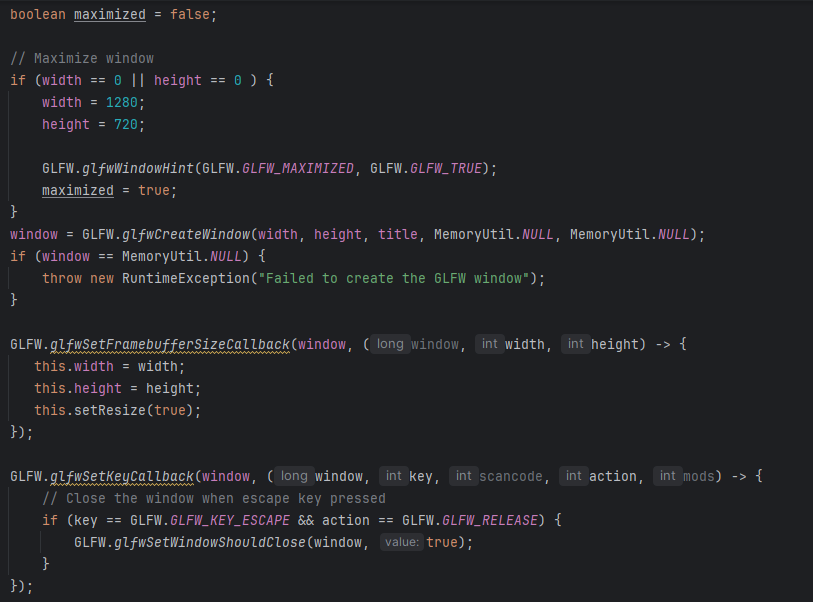


The start method calls the init method to initialize the engine and later runs the engine is not already running



The methods now calls the init method in window (to show the window) the game logic to run the game and mouseInput to enable mouse input.

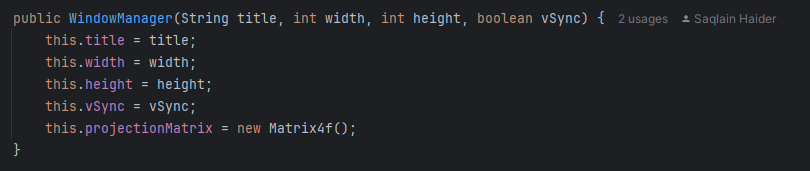




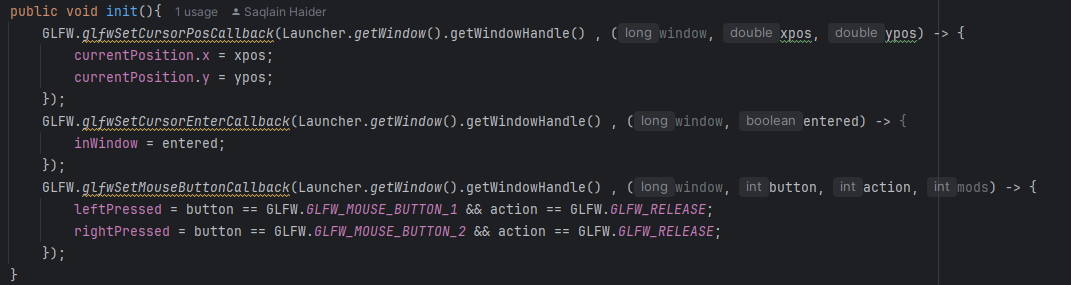
A screen shot of a computer program

Description automatically generated

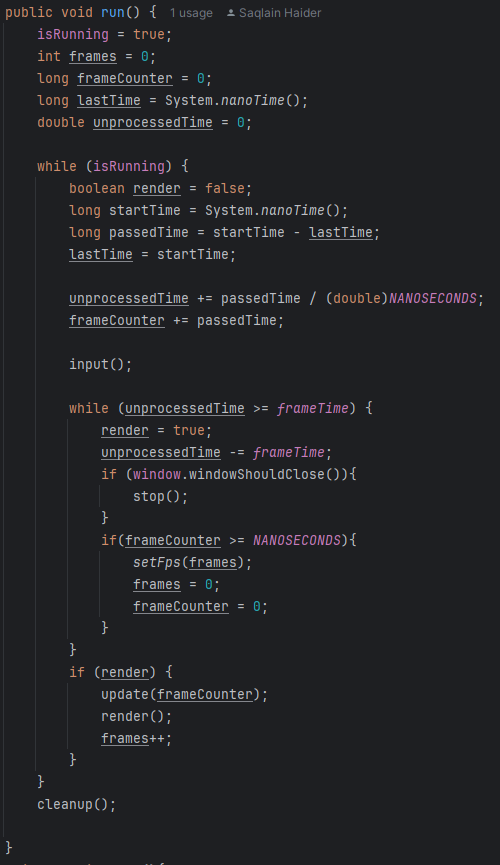
The init method in WindowManager is responsible for displaying the window basic constructor to set the height, width etc.



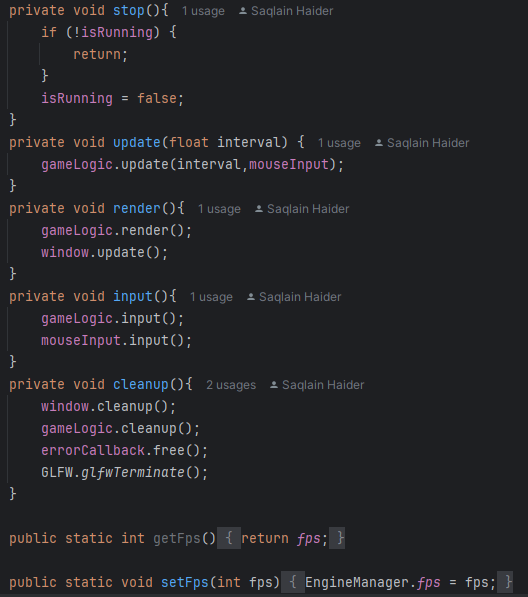
The init method now sets the window to visible, resizable, sets the OpenGL versions the upper bound and the lower bound etc. If the height and width is 0,0 the screen is maximized. Later creates the screen and throws and exception if unable to create screen. Calls events when a key is pressed, and the window is resized. If screen is not maximized, it centers the screen using vidmode.



The mouseInput init() the events exits which are called when ever the mouse position changes on screen, enters the screen, left or right button is pressed.



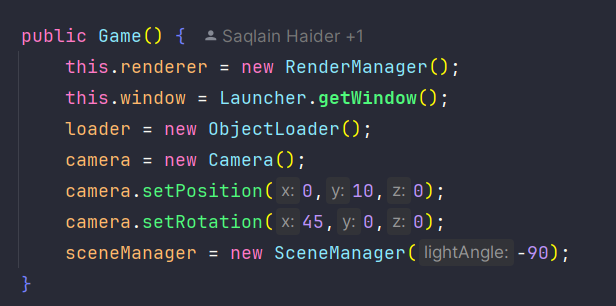
The run function sets isRunning to true and handles the main gameloop. Input() is called every iteration of the of the while loop. A basic timer judges the time elapsed between each frame. And only runs the next frame when the time passed between frames is greater or equal to the time each frame should run after (frame time). Frame time is calculated by 1s / FPS (how many frames we desire to show in 1 second) eg. If frames are 60 so 1/60 = 0.01667 is the delay between each frame this is the time the loop waits to call the next iteration of update() and render(). Stop function is called when escape key is pressed (checked from window class). Calls cleanup when breaks out of the loop i.e. engine is closed and not running engine runs first but closes in the end.



Update(), render(), input(), cleanup() calls the update/ input/ cleanup/ render in gameLogic, mouseInput and window

Renderer:

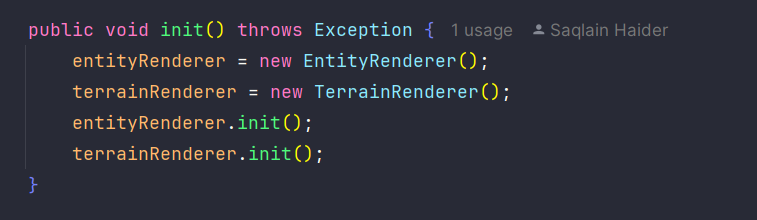
The Game class initializes the RenderManager class by calling the init() .



A screen shot of text

Description automatically generated

RenderManager init(); initializes the two render managers i.e. EntityRenderer and Terrain Renderer.



The entity renderer contains the HashMap of the Model of each Entity;



Creates and instance of the ShaderManager class and createsUniforms



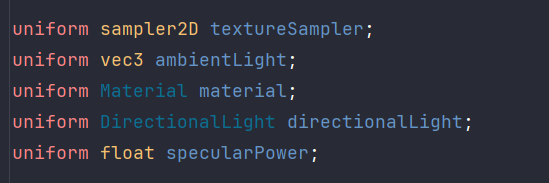
ShaderManager contains a Map of he uniforms String is the shader code and the integer is the shader code id.

A screen shot of a computer code

Description automatically generated

CreateUniform creates a uniform with uniform name and the appropriate ID. If the uniform of the same name doesn’t exist.

Uniforms are set for each given value type. These uniforms are than used by GLSL shader code to calculate textures and draw/ render them on screen.

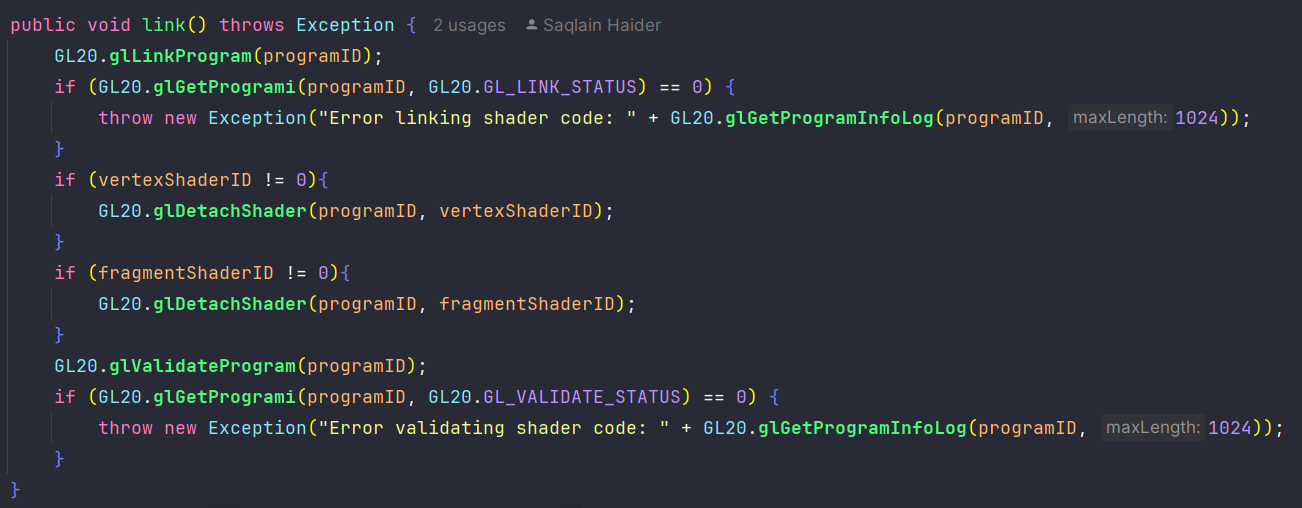
 A blue and white text on a black background

Description automatically generated

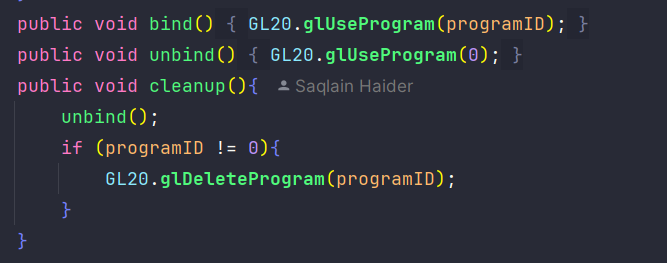
A screen shot of a computer program

Description automatically generated

The createShader() reads the shader code / compiles it and attaches the programID with the shaderID.

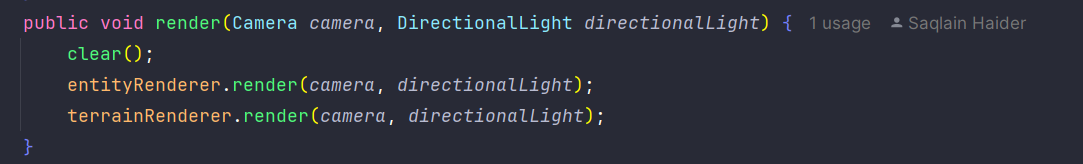


Checks if shader is vertex or fragment and links the shader code to the programID.



Other functions in ShaderManager to bind the shader with the entity // Terrain or what ever thing that might be using it. Bind to called when rendering and unbind is called when unrendering.

Back to the rendering class



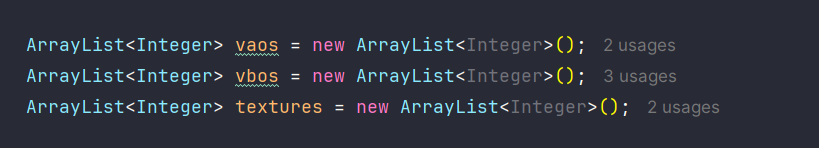
Entity renderer and terrain renderer is separate to enable future modifications. But the RenderManager is only responsible for Rendering Lights and creating/ calling the constructor or EntityRenderer and TerrainRenderer.



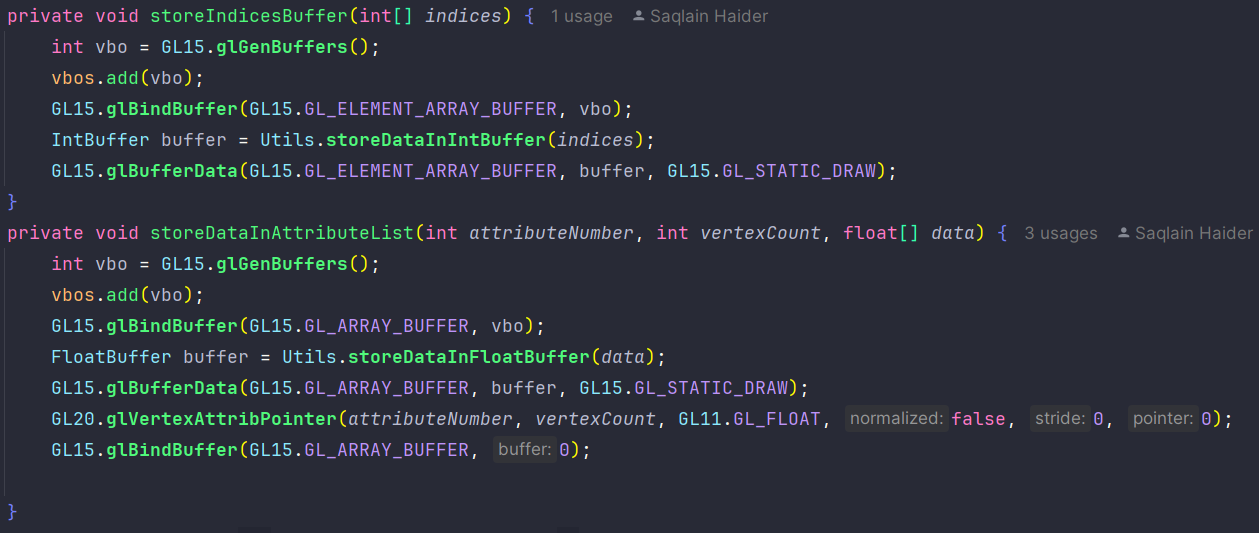
Now the render method calls the shader.bind(); and sets the projections matrix (A orthographic volume box to display shaders). Forward is -z, backwards is z, left is -x, right is x, y is up and down is -y axis. Calculated by the bottom left most vector3 and the top right most vector3 them draws the models on the screen (drawing triangles).



Cleanup(); unbind(); and bind(); performing similar tasks as explained earlier i.e. binding the model with the entity and allocating 3 attribs in Vertex Atribs Array/ Objects // array that stores all the information regarding the drawing of the entity. This information is gained from the ObjectLoader Class when loading the object. Unbind disables all the Array Objects allocated for that model to render.

ObjectLoader:  A computer screen shot of text

Description automatically generated

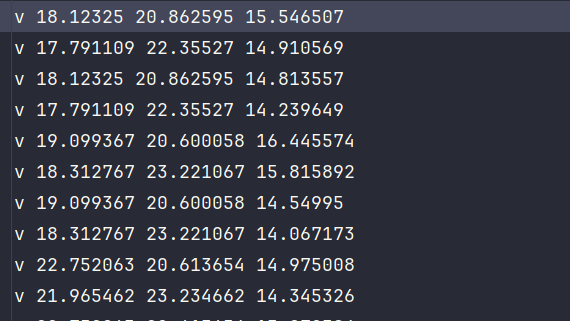
This as explained stores data in buffers (temp storage) to render the faces/ vertexes and textures

 A computer screen shot of text

Description automatically generated A screen shot of a computer program

Description automatically generated A screen shot of a computer program

Description automatically generated

OBJ format support this reads the obj format using file handling (reads line by line) stored in the shown format: A screenshot of a computer

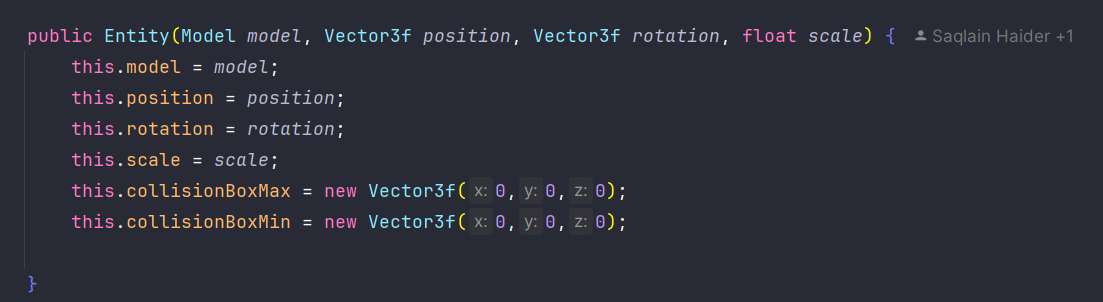
Description automatically generated A screenshot of a computer

Description automatically generated

The object stores date in the array and processes the faces from vertices stores the array data in buffers of the GPU in order for the render the them on the projection Matrix 

Standared unbind and cleanup methods for ObjectLoader

Enitity:



Model:

A computer screen shot of text

Description automatically generated

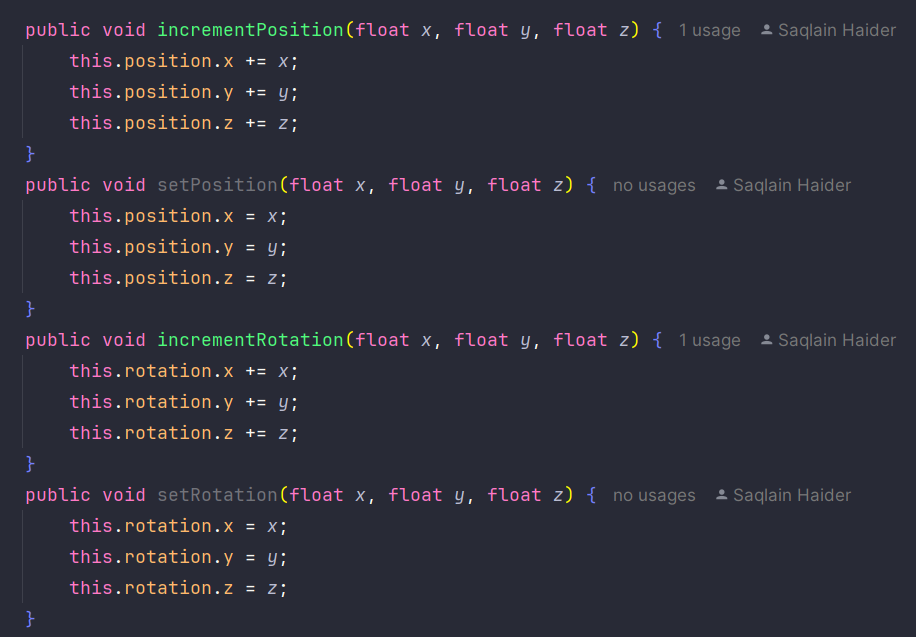
Material:

A screen shot of a computer program

Description automatically generated 

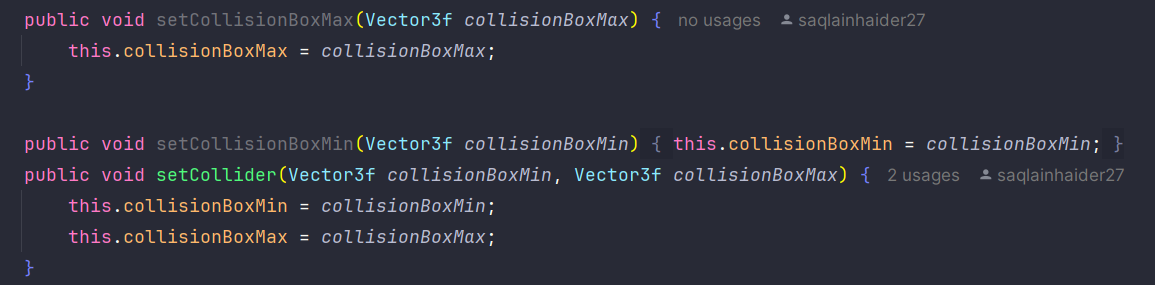
Texture: A screen shot of a computer program

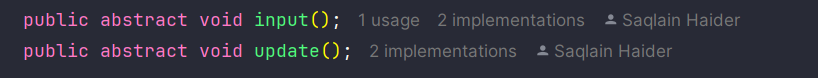
Description automatically generated

Entity class has standard methords to update position and rotation of the entity 

AABB collision with other entity

A screen shot of a computer code

Description automatically generated 

Enitity is an abstract class with 2 abstract functions 

Player: A screenshot of a computer program

Description automatically generated

A computer screen shot of a computer

Description automatically generated

ScoreManager: A screen shot of a computer program

Description automatically generated

SceneManager:

A screen shot of a computer program

Description automatically generated

Has a list of entites and Terrains on the scene adding in this list renders the entity and removing unrenders it   
Game: A screenshot of a computer program

Description automatically generatedA computer screen shot of text

Description automatically generated

A screen shot of a computer program

Description automatically generated

Pause unpause the game.

Update is run when the current state is play A computer screen shot of text

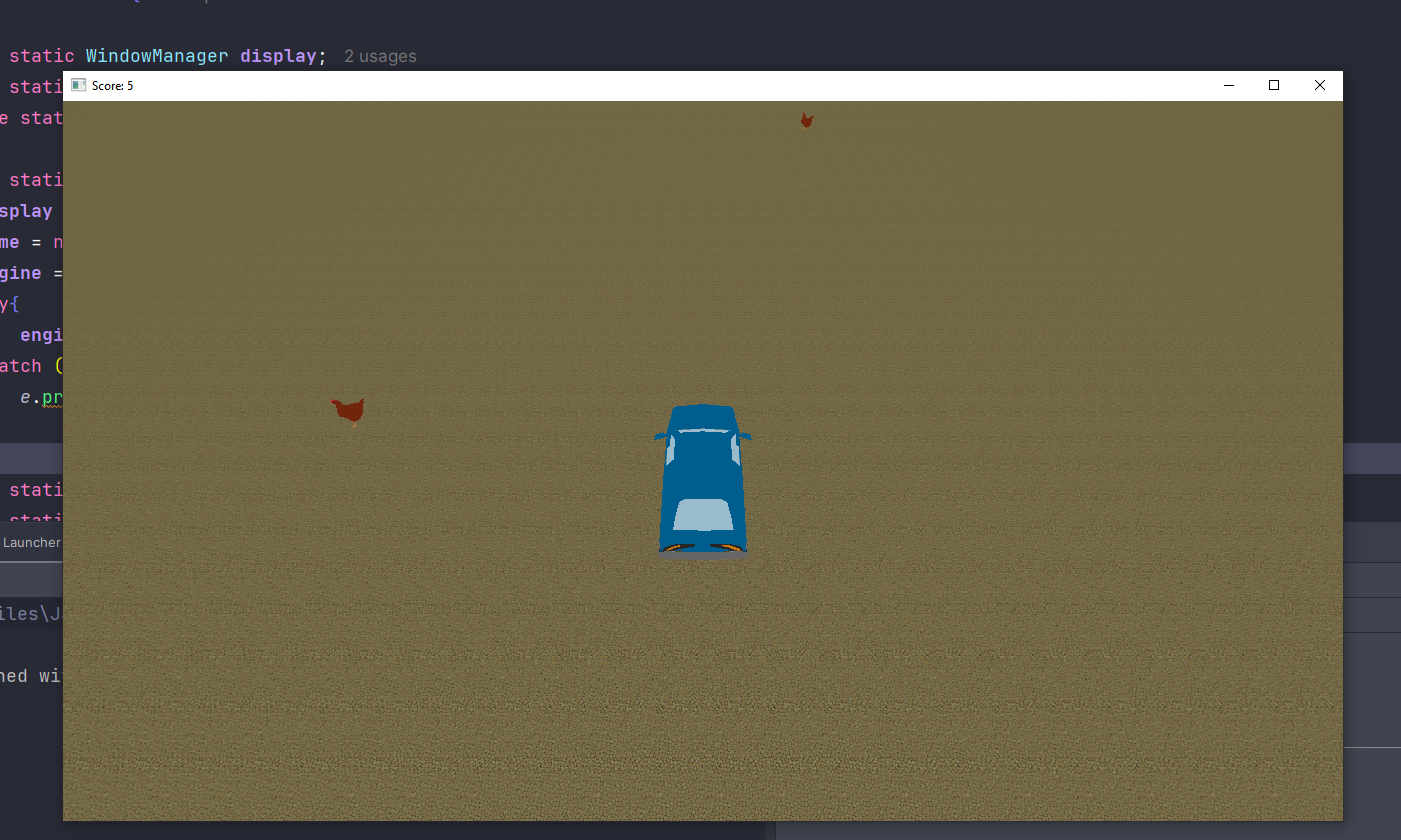
Description automatically generatedA screen shot of a computer program

Description automatically generated

Player only collides with collectables

Score is set on screen title 

Calls render for each entity and terrain in list.



Car moving forward continuedly colliding with chickens. Chicken unrenders itself when car collides with it Score in title of the screen game pause/ stops when score is 100.