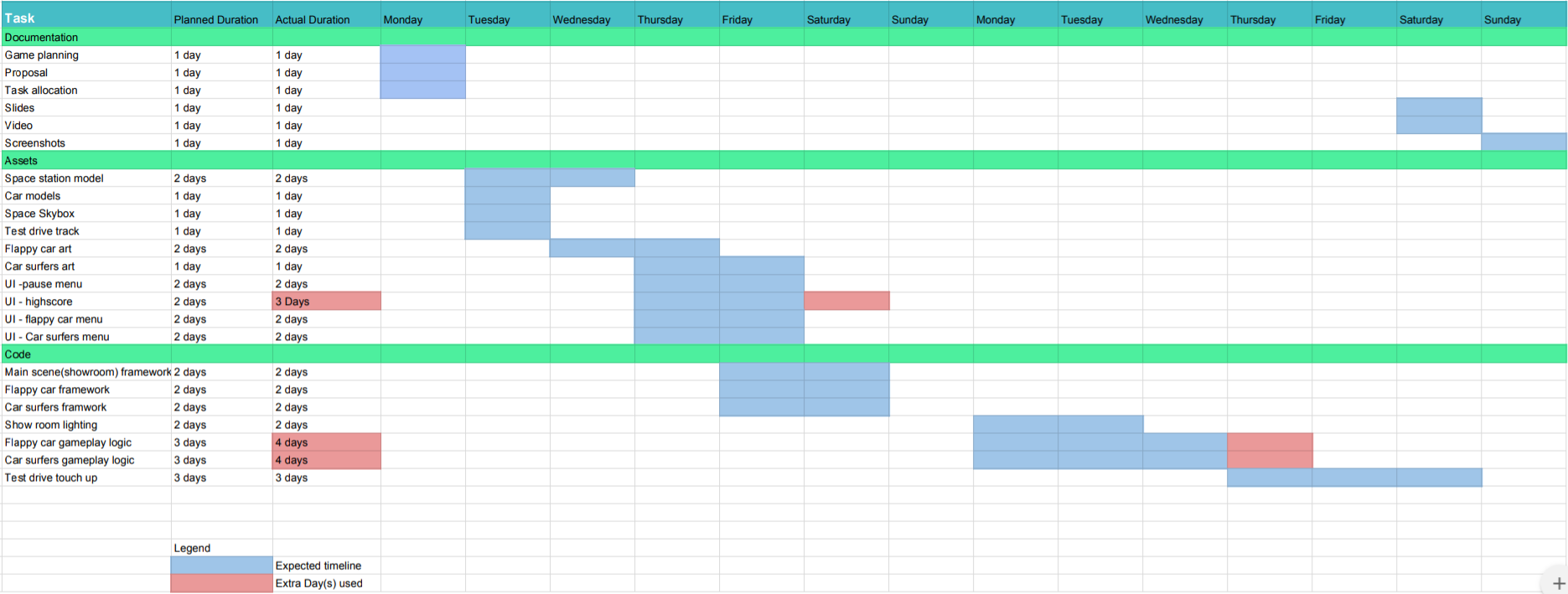
Project Report

Project Schedule



Implemented Features :

* Powerful UI
  + Pause menu
    - Makes use of alpha blending in the tga to make it translucent and to make it look cooler
    - Reuse of the indicator tga(the arrow) to how which option the player is at
    - The menu also displays the names of the car player has selected to try out minigames and test driving with
    - The options are linked to different values in the enum which can be edited to have more options on the pause menu
      * When the player chooses an option, depending on which option the player chooses can toggle the pause menu(resume), open the highscore board for the minigames, exit from the test drive area back to the showroom or exit the game entirely
  + Car Stats menu
    - When the player is nearby the stat menus of each car will pop up and show the player the individual stats that each car has
    - Using a quad with texture, it will render at a fixed position but the rotation is depending on the player as it will calculate the rotation orthogonal to the player so that the player can see the stats no matter the angle they are standing at
    - When interacted with, the player can use the same car for test driving or to play minigames with
      * Each car has different stats to make gameplay more fun or more challenging
  + Highscore menu
    - Displays the top 3 highscores of each minigame that is accessible anywhere in the game
    - It reads the text files that the scores are saved in and renders them on screen
  + Speedometer
    - The speedometer will rotate based on the velocity of the car to show how fast the car is going in the test drive area
* Selection Screen
  + Collision detection for interactions and spotlights for the car
    - Something similar to test drive
* Test Drive (3D)
  + Cars control (Cars movement)
    - The Cars’ turning and speed is base on the Handling and the Acceleration of various car
    - 3rd Person Camera is applied
    - Changes in Shader to make car flashing white when respawn
      * Controlled by local boolean and pass it into a new Render Mesh function to render the white color when it's needed
      * Boolean check also added into the fragment shader script
    - Collision Detection applied
      * The track is a round shape so we check by comparing the length (player’s position to the origin) with the radius of the round track
        + (Player.pos.x - adjacent of radius)^2 + (Player.pos.z - opposite of radius)^2 <= radius^2

* Mini Games
  + Flappy Car
    - Orthogonal view camera
    - Minigame1obj class
      * It is a class that stores and returns data for objects
        + Coordinates
        + Velocity for x & y
        + Type (For powerups and player)
        + Next address
    - Main Menu
      * The main menu is what will appear when going from the main scene to minigame1 scene
      * The options in can be chosen by up and down arrow keys and the enter button
      * The main menu options are
        + Start game
        + Highscore

View top 5 highscores saved in the highscores txt file (ifstream)

* + - * + Controls (View game controls)
        + Select Car

Press 1/2/3/4 to select the car they want

All 4 cars can be viewed at the same time with their animations

* + - * + Return to motorshow

Change scene to motorshow

* + - * Control, Car selection and Highscore menu size are tracked by their own floats which will increase when opening them, enlarging the menu, and decreasing then to 0 when when returning to the main menu
    - Endless Generated Background
      * Backgrounds data is saved in a Minigame1obj singly linked list
      * Backgrounds are generated when there is going to be a gap on screen (Background.x <= 400)
      * They are deleted to save space when there are off screen (Background.x <= -400)
      * The background is moved up update based on its velocity which is based on the gamespeed (float)
    - Randomly Generated Walls
      * Walls data are saved in a Minigame1obj singly linked list
      * Walls are generated at the end of the list according to the game update speed
      * When they are generated they are randomly assigned a y coordinate to create a gap ((rand() % 580) + 10)
      * Wall collision with the player is checked using AABB collision check (eg. (Player->returnlocationx() + 25 >= WallMid->returnlocationx() - 25 && Player->returnlocationx() + 25 <= WallMid->returnlocationx() + 25))
      * If the starting wall is too far left, it is deleted and the next wall becomes the new starting pointer (WallStart.x <= -20)
      * When a wall is deleted, the player gains 100 points
      * The walls are moved every update based on its velocity which is based on the gamespeed (float)
    - Car Control
      * Car data is stored as a Minigame1obj pointer
      * Car control is determined by which car the player is using
        + Weight (y downwards velocity)
        + Floatiness (y velocity)
        + Speed (maximum game speed (x velocity))
      * Cars can be changed in the car selection menu
      * When the player press spacebar, the velocity of the car is set to a value based of the car
      * The car is moved every update based on its y velocity
      * The car cannot move below the screen (y < 25) or above it (y > 575)
    - Powerups
      * Powerup data is stored as a Minigame1obj pointer
      * There are 4 different power ups able to be collected
        + Slowdown (Reduce game speed and increases wallsize)
        + Wall breaker (Deletes the wall pointer closes to the front of the player on the right when pressing enter)
        + Gain +100 points
        + Random (Randomly Selects a power up from above)
      * The powerups are stored in boxes and the player must touch them to gain their effects, which is check by AABB collision checker
    - Sounds
      * Sounds are played using the irrklang library which is linked to the file (SoundEngine->play2D("audio/sound\_crash.wav", GL\_FALSE);) (GL\_TRUE is for looping the sound)
      * Wav and MP3 files can be played
      * Sounds are player when
        + Menu Cursor is selected
        + Car is selected
        + Menu is selected

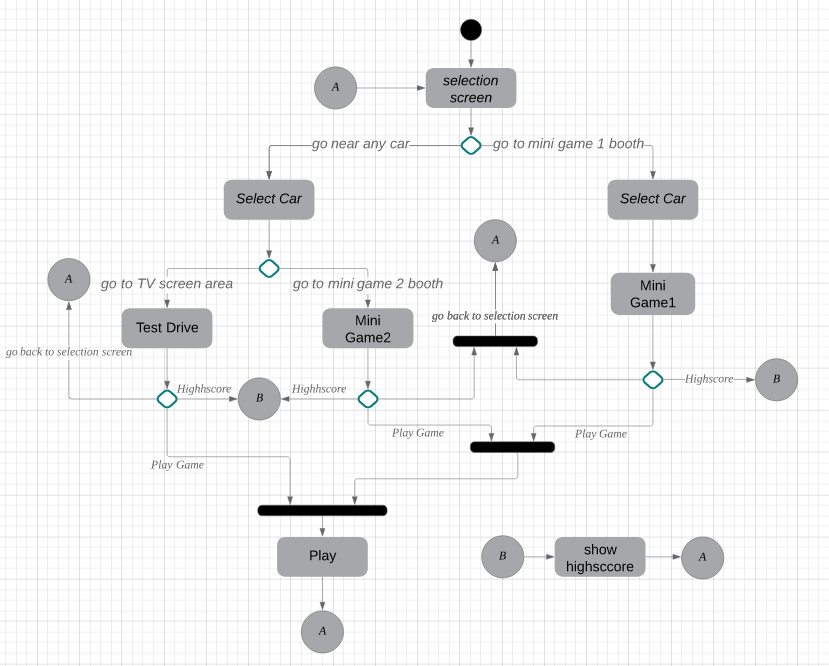
Highscore menu

Car selection menu

All other menus

* + - * + Game is started
        + Player Jumps up (with a delay in between each jump)
        + Wall is destroyed by wall breaker
        + Collision with a wall and obtaining a new highest score
        + Collision with a wall nothing obtaining the highest score
    - Score
      * Score is tracked by a int
      * 100 score is gain when a wall is deleted or a 100 points powerup crate is collected
      * After the player loses (collision with a wall) the points are checked against the scores in a highscore txt file (ifstream) and if a new high score is obtained, saved into the highscore txt file (ofstream)
      * The top 5 highest scores are saved in the txt file
    - Texture/Sprites/Animation
      * All textures are saved as tga files
      * All sprites used are 2x2 with no spacing in between
      * Car animation
        + Animation state is tracked by an int where it is form 0 - 3, each number being 1 animation state
        + Animation state number increase or decrease is tracked by a bool (true is increase, false is decreasing number)
        + Animation state change is changed on a timer where the animation state number is changed based on the bool
    - Restart
      * A restart function is activated when the player goes from the game over menu to the main menu
      * The restart function would
      * Delete all pointers except sound engine
      * Reset score
      * Reset Timers
      * Create a starting background pointer
      * Create a player pointer
  + Car Surfers
    - Orthogonal View Camera
    - Endless Road Background (until player dies)
      * Road is moving downwards through the use of shaders
    - Rocks
      * Rocks are randomly generated and will appear in one of the three lanes
      * Rocks will go down the lane. When the Rock goes out of the screen, it is deleted. A new rock will be randomly generated on the top of one of the three lanes.
      * Rocks will move faster as the score become higher
    - Car Controls
      * Players are able to move left of right to avoid the rocks coming down the lane
    - Nitro Boost (Power-Up)
      * Nitro Boost will increase over time
      * When Nitro Boost reaches a 100, player can press <space> to active boost
    - Collision Detection
      * If the player hit a rock, they lose
      * When Nitro Boost is used and the player hits a rock, rocks will be bounced back.
    - Score System
      * If Player is alive, the score will continue increasing
      * While Nitro Boost is being used, the score will increase by 3 times the speed.
      * All time high score is tracked via text file
        + When there is a new high score, it will be updated into the text file so it will be saved
    - Sound System
      * Sounds are played using irrklang library
      * Sounds are used for the sound effects of nitro boost and when the car hits the rock

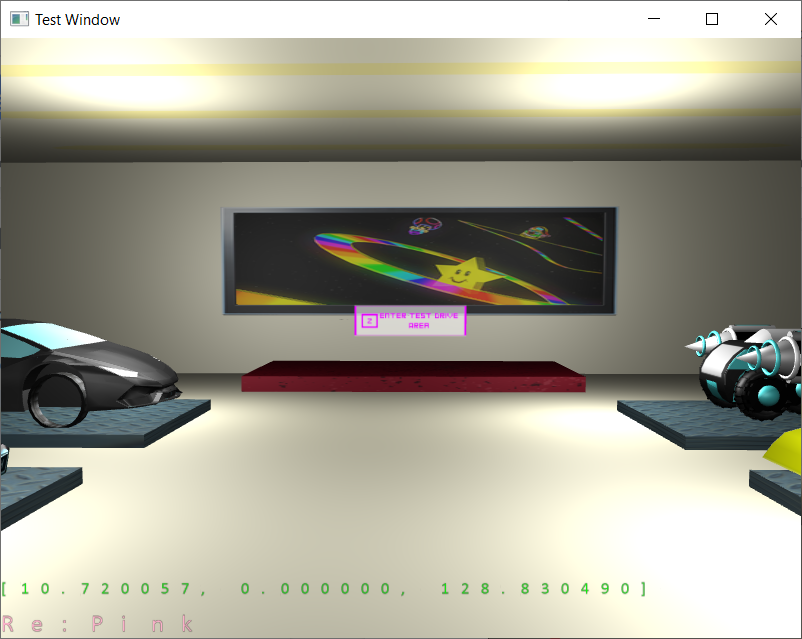
UML Activity Diagram :

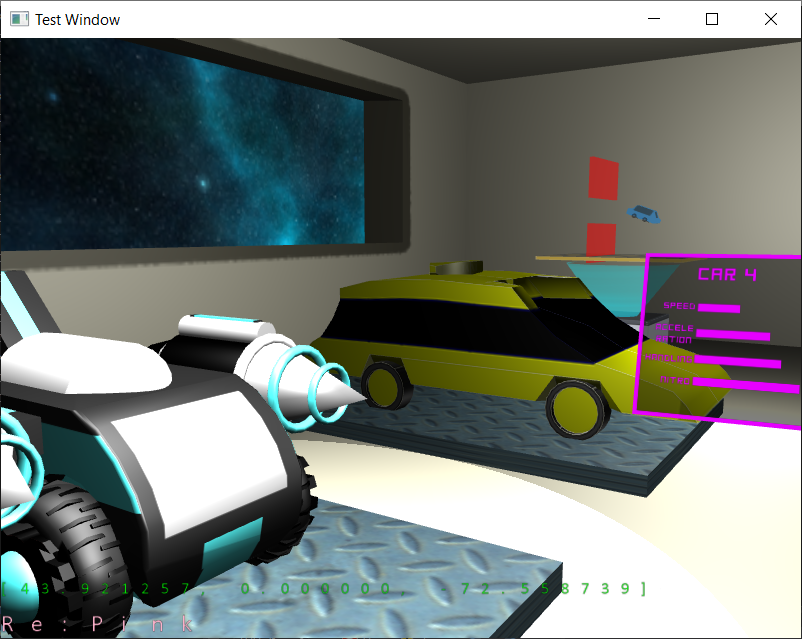


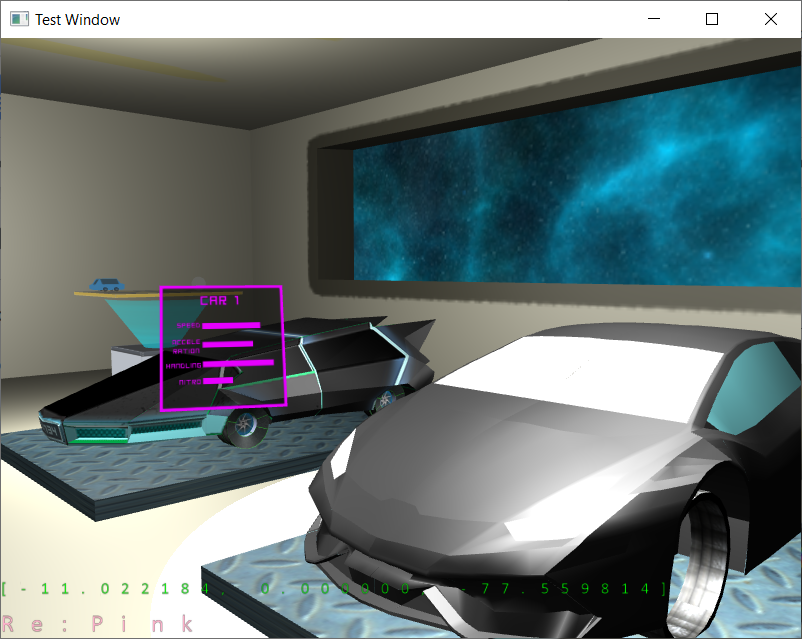
* Since its will always loop back to the selection screen so there is no end point to this game
* Mini game 1 and Mini game 2 have a different way to select car

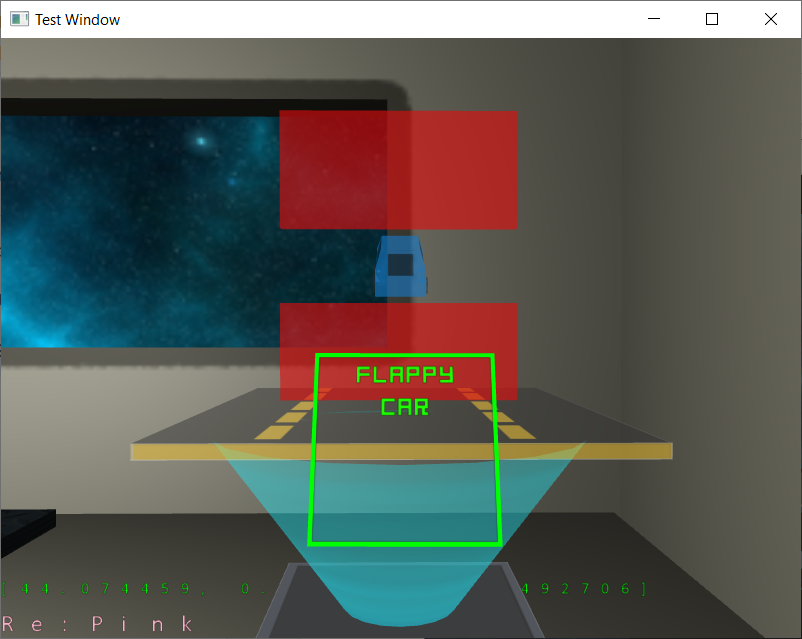
Screenshots:

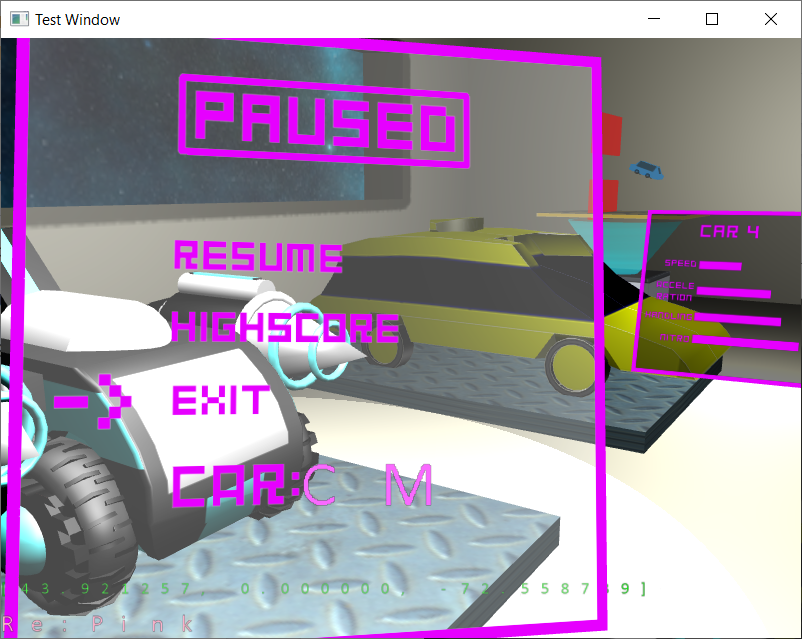
Motorshow:

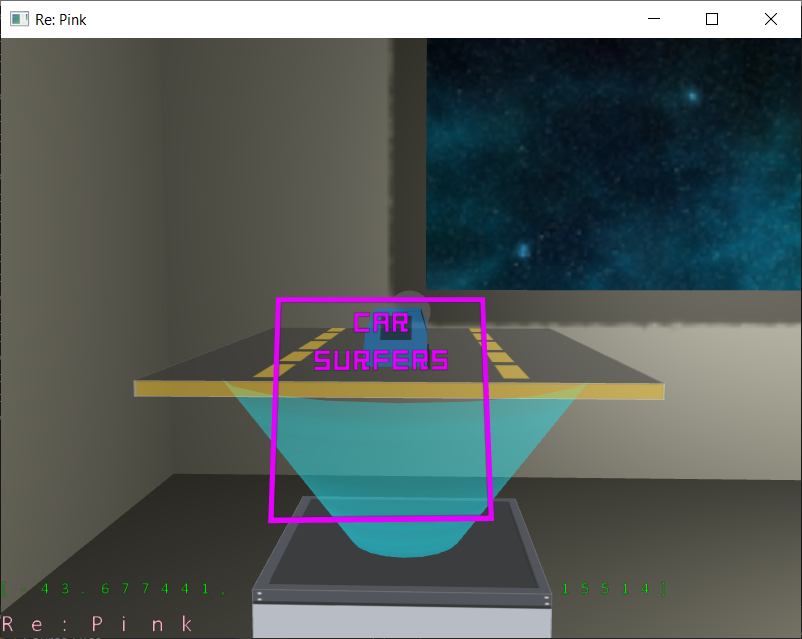


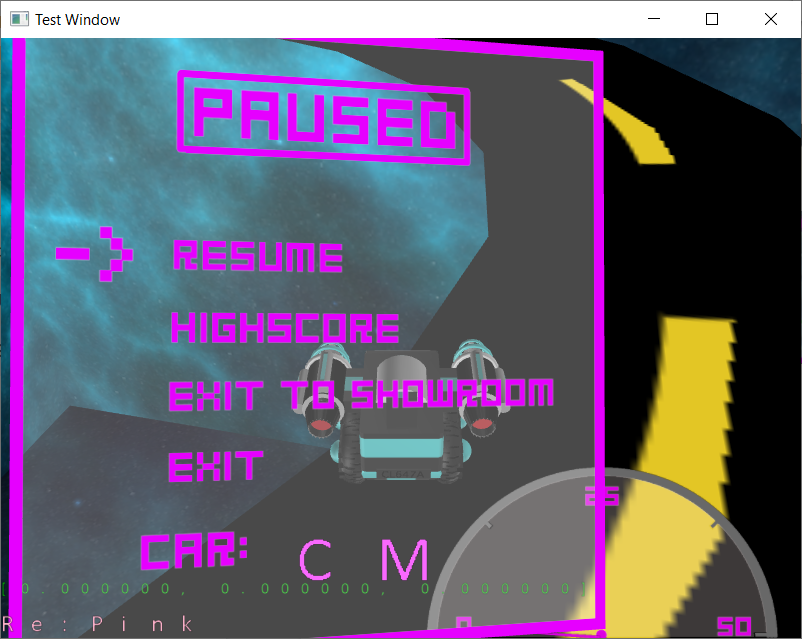








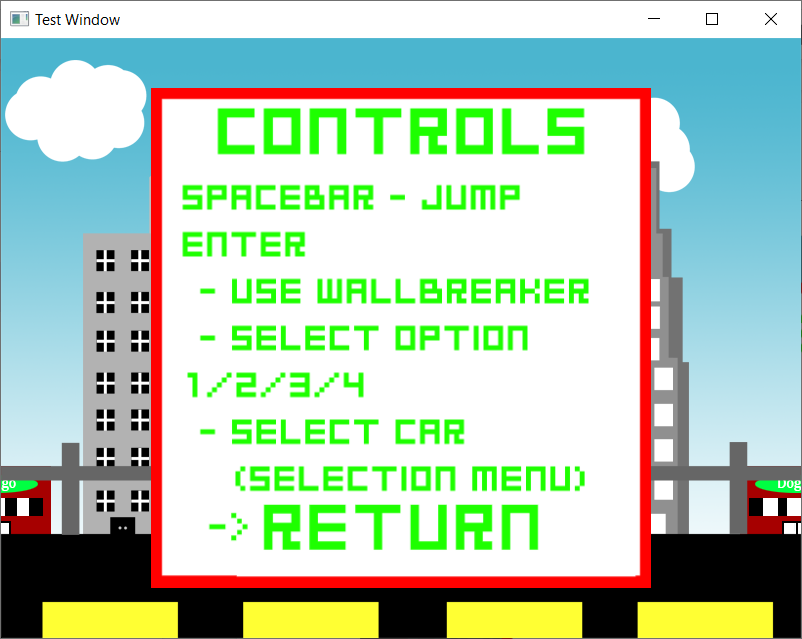


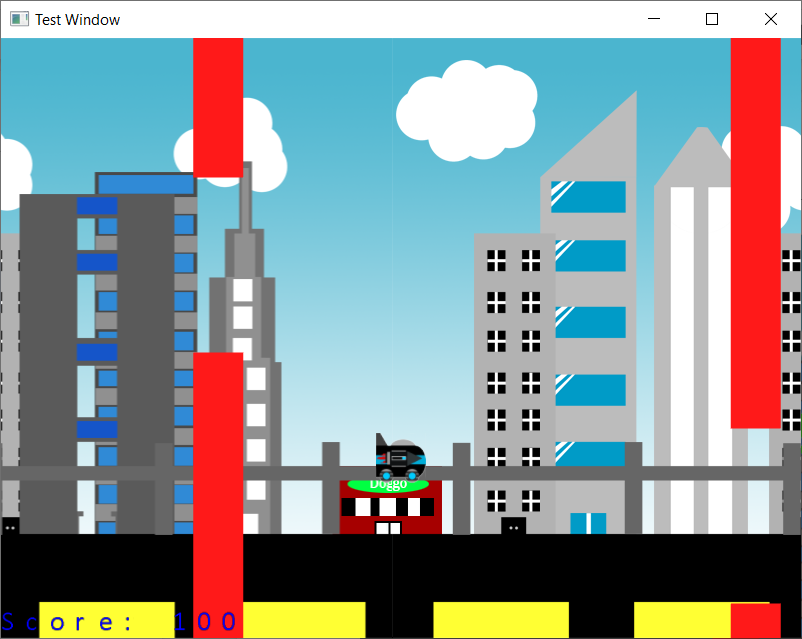
Test Drive:



Flappy Car:







Car Surfer:

