**Hover Racing Game – Report**

I believe I will get a mark of 80+

Level editor:

* Load up the level editor and move around the map;
  + If no map is provided “map.txt” will be made and write all the basic objects to it
* Wasd moves around
* Q and E rotate the camera
* Arrow keys move the camera
* Z and X to cycle objects
* 0 to place an object
* 1 and 2 to increase the steps by which the object moves
* Collision boxes assigned based on Type

I first made an Array of all the meshes that could be used label them using Enums. A function is called that writes (t, x, y, z, r) where t = type, xyz = co-ordinates and r = rotation. A function reads the list and create a vector that stores all the information. A vector is created that stores the Imodels. This info is passed to a class to handle the information (bounding boxes) and the class is then saved to another vector to later be manipulated.

State Transitions:

* Games states use Enum
* Due to level editor, checkpoints use ints not enums to allow for multiple checkpoints
  + They do work in order this way
* When at end state, Reset option available that resets values to defaults.

At the beginning of each states it contains statements that could change the state for easy readability.

Collision Detection:

* Sphere to box
* Sphere to Sphere
  + Checkpoint legs
  + Other player

Using a for loop, I loop though every object created in the Imodel vector, It then compares the type with an Enum to either call S2S or S2B and it passes the player class and the Object class. If they return true then momentum is reverse and the car bounces and slows down.

S2S sqrt( pow( (Sphere.Model->GetX() - ObjSphere.x),2)

+ pow( (Sphere.Model->GetZ() - ObjSphere.z),2)

)- Sphere.Radius - ObjSphere.Radius < 0

S2B Sphere.Model->GetX() > (Box.XMin) - Sphere.Radius &&

Sphere.Model->GetX() < (Box.XMax) + Sphere.Radius &&

Sphere.Model->GetZ() > (Box.ZMin) - Sphere.Radius &&

Sphere.Model->GetZ() < (Box.ZMax) + Sphere.Radius

Car:

* moves using vectors (momentum, drag, thrust)
* leans into corners
* bounces smoothly off objects and loses some momentum
* boost
  + Uses frameTime as an int and checks how long the space bar has been held using bools
* Basic damage model
  + Uses Bools, when collision detected set true, remove 1hp, set false.

Using the car matrix, I created structures to determine the resultant force of the car depending on which direction it faces. The calculation of the Drag as a co efficient of the momentum put an un-official speed limit on the car. The Car its self is attached to a dummy so the camera can move separate from it and so it can rotate locally without causing problems.

NPC:

* Second Player object Follows Dummy nodes that can be placed in the level editor

