The following describes how to use the car racing game that I have submitted as my final project. The goal of the game is to beat the other cars around the racetrack. The user-controlled car is the blue one. If the blue car hits any of the other cars or the edge of the world, the car will stop. The computer-controlled cars can overlap each other and will keep going if the user car hits one of them.

**Keyboard Commands:**
Up Arrow = Increase Speed
Down Arrow = Decrease Speed
Right Arrow = Turn Right
Left Arrow = Turn Left
Num Keypad 4 = Turn Turret Left
Num Keypad 6 = Turn Turret Right
Num Keypad 8 = Make Turret Larger
Num Keypad 2 = Make Turret Smaller
Num Keypad 1 = Move Turret Left
Num Keypad 3 = Move Turret Right

**Mouse Commands:**
Mouse Wheel - Zoom In/Out

**Panel Controls:**
Use the left and right buttons under **Car Controls** to turn the car left and right. Use the **Speed** slider to change the car's speed. Use the **Stop** button to stop the car.

Use the left and right buttons marked **Turn** under **Turret Controls** to turn the turret left and right. Use the **Size** slider to change the size of the turret. Use the left and right buttons marked **Move** to move the turret left and right.

Press the **Pause** button to pause the game. Press the **Reset** button to reset the game.

**Project Description**

**User Interface -**
This user interface for this project was implemented using MFC. In order to use OpenGL with MFC, the device context that is used to create the application's windows must be "tied" to an OpenGL rendering context. When the Windows operating system is ready to draw the window, it passes control over to OpenGL which draws the window. One of the main problems I had was a separate rendering context needs to be created for each window/viewport in the application. Since my user interface contained 2 viewports that showed the same thing, I needed to explicitly tell Windows to share the rendering resources between the two. This took me quite a while to debug. Texture maps would show up in one window and not the other.

**TCar Classes**
The cars in the program are implemented as TCar objects. A TCar object is comprised of graphics primitives. TCar objects contain a TMovingProperty object that controls all of the movement of the car based on its velocity and its direction. The TCar class contains a Move method that when called tells its TMovingProperty to move it. The TCar class can also return all 4 points that define its perimeter based on its rotation, scale, and translation. The class for the hero car, THeroCar, is derived from the TCar class. It adds a turret to the TCar class. The class for the villain cars, TVillainCar, is derived from the TCar class also. TVillainCars contain a pointer to an list of points that define their path around the racetrack. When the TVillainCar reaches or passes a point on the path, it calculates its new direction based on the next point in the path.

**Basic Program State**

```
Initialize
User Interface

Initialize
Scene

Service Windows
Messages

Service Keyboard

Check For Collision

Move Cars
```