# Objective

Place as high as possible in each race and claim points in a bid to win the drivers’ championship. Driving the course also has its hazards, beware of drivers trying to clip your car and also oil slicks that can appear randomly. Hitting one of these will cause you to spin and lose time and possibly the race.

Running off the track will cause the car to slow down and trying to cut corners could result in you being penalised and force respawn back at the start / finish line.

Bonus items also appear randomly. There are three types, a “Go faster”, a “Poison” and a “missile”. You must use the item collected if you wish to pick up another.

Go faster – is a short acceleration burst, this could be the difference in winning the race or not.

Poison – This causes the car to slow down temporarily. Why would I want this? You don’t, that’s the risk in picking up the box.

Missile – the one everyone wants, fire this at your competition and show them who owns the track.

Be aware, your competition has these abilities also.

# How to play

W Accelerate

A Steer Left

S Brake / Reverse

D Steer Right

SPACE Use Bonus Item

B Look Behind

# Features

**Color key collision detection to detect track run off**

This was code that was borrowed and modified from an existing 2d racer game and can be used to sample a piece of texture contained on the back buffer and detect if that sample contains any pixels matching the color you want. The major challenge here was how to go about taking code that was applied to a 2d environment and applying it to a 3d environment. This once solved made the creation of this game a lot easier, as the alternative of trying to implement bounding boxes wouldn’t have allowed me to create the game I wanted to create.

Credit and a big thank you due to George W. Clingerman <http://www.xnadevelopment.com/tutorials/theroadnottaken/theroadnottaken.shtml>

**AI / CPU cars**

Cars can navigate around the track themselves, giving a realistic race simulation. Tying into the above, it was now possible to create a “radar” sort of system that would allow the cars to navigate around the track.

Having a left, front and right feeler setup, depending on which feelers are triggered by detecting off track colors, the car will take appropriate action, i.e. left feeler triggered then steer right to stay on track.

**Bonus items**

As described in the game objective, these were placed in the game to make it a little more interesting adding advantages but also risk factors to the game.

The bonus boxes are randomly positioned on track making use of the color key collision detection to ensure that they are actually positioned on the track. They remain in place until they are either collected or 5 seconds have passed in which case they will be repositioned.

**Obstacles**

These again described in the game objective is an oil slick that is randomly positioned on the track. If a car contacts them then that car will be caused to spin and lose time.

This is similar to the bonus item, in that it is using the color key collison detection to ensure that it is actually been positioned on the track.

**Check points**

This was put in place to ensure that all cars complete the full circuit before they are awarded a lap. Failing to pass each checkpoint in sequence will result in that car respawning back to the start / finish line.

Check points are positioned at the start of every race by scanning the texture and detecting colored squares that I positioned there using Photoshop.

To improve performance, I divided up my Photoshop using a grid pattern into 128 pixel blocks to minimise the time it takes to scan the entire texture which is (1024 x 1024).

**Automatic fill race finish table**

I the player car finishes before the other cars, I implemented a method that would end the race and automatically populate the drivers list with the other cars. The purpose of this is improve the players gaming experience as waiting for all cars could possible take an eternity if they were to continue hitting oil slicks, running off track and forced to respawn.

**Custom comparison class**

This is a custom class that was implemented to sort the car/drivers championship points. Originally I was using a method that I found on the internet that used LINQ, but seeing that we covered collection framework in OOP3 recently I felt it would be best to implement some of the theory learnt there.

**Content/Models/Textures created by Author**

Just a quick note on the content that was used in the game, the cars, hoarding, and track were all created in blender. All textures were created using Photoshop and the video clip at the beginning was created using blender and windows movie maker. XNA and blender logos were used to give acknowledgement to the technology used to create this game.