Game Design Document

Version 2017.7.21.2

Table of Contents

A Lengthy Introduction

Taking the opportunity to wax poetic and perhaps write with ostentation, *Ascengine* is a proud member of the resurging anti-gravity racing genre of games. In the last half decade, notable entries within the genre include and are most certainly limited to *Redout* as well as *Fast Racing Neo*. Neither game presents itself as being significantly different from their pioneering predecessors such as *F-Zero* and could yet be considered clones. Speaking with the pretentiousness, as one to pursue innovation at all costs while submerged in the sea of narrative games academia, I hope to offer players a different experience with *Ascengine*.

Surely, *Ascengine* will maintain the defining characteristics of the genre; vehicles will move quickly and game levels will not be flat, but rather varied and twisted to the point of player interest and engagement. However, the world has seen this tried, tested, and true formula presented a good many times now. We force ourselves to step forward and give people the opportunity to experience something they didn't know they wanted. To move forward, we must first look backwards. In 1999 LucasArts released the popular *Star Wars Episode 1: Racer* and its accompanying arcade cabinet *Star Wars Racer Arcade*.

A feature of this release, which I imagine most experienced in an arcade if at all, is player agency to control two vehicle engines independently. While briefly revisited within *Kinect Star Wars*, today we have access to excellent console controllers with similar designs across different platforms. Short of an adventure into virtual reality, there is no need for players to experience this kind of racing title with both arms outstretched, a potential cause for fatigue. There exists a miraculous invention: the joystick. Fortunately, controllers these days have two of them. *Ascengine* promises to be a dual-stick racing title: two engines, two joysticks, one to one input. For better or worse, fun or frustration, this is the path of innovation *Ascengine* will take. This is *Ascengine's* purpose.

With this, avenues of opportunity become available. Learning from the shortcomings of *Ascengine's* predecessors, the physics and forces of classical mechanics can be simulated. In an unfamiliar environment, for the players who brave the frontier I created, I can give them that comfort. Make the game intuitive by replicating behavior that is already known. Making sure to leave as many different knobs exposed to tune the experience to a feeling of perfection, may I chip away at the marble to expose something shaped as excitement and fun. With twin engines of futuristic design, may I ascend and meet this challenge.

An Explanation of Physics

The implementation of physics in *Ascengine* could be called simple and deep. With additional coefficients for tuning, *Ascengine* features principles of force and hence acceleration, torque, drag and perhaps other physics quirks from the world. It all stems from interaction with the engine's rotation. Unless otherwise noted, vectors are in vehicle coordinate space and not world-coordinate space.

The vehicle will accelerate either forward or backwards (note, not move) based on the ratio of both engine's X component vector (horizontal) and the appropriate thrust coefficient (player input). By necessity then, with the relationship of angle and component vectors, the larger the X component vector(s) the smaller the Y component vectors. As an anti-gravity racer, the vehicle will then move closer to or further from the ground or racing surface. An observable effect of lowered suspension on vehicles in the real world is reduced drag. While normally a trade off in practice, going faster is normally quite the advantage in racing games. In our case in particular, there is a negative correlation between vehicle height and X component magnitude, meaning that when you go more quickly, your vehicle has naturally moved closer to the ground. This could lead to an exponential relationship if we used our distance from ground directly as a scalar to drag. It is likely better to add or subtract a small amount of drag, multiplied by said scalar to a base drag value, or something similar.

As the engines are not acting on or attached to a single point, each creates a torque on the vehicle's body (and themselves). Notably, our vehicle will only accelerate forwards (at all) if both torques push from behind the vehicle. If the torques are uneven, the vehicle will turn while accelerating forwards. To what extent the vehicle turns is dependent on the ratio of the two torques and likely a coefficient or two for tuning. Similarly, if the torques are both clockwise or counterclockwise, the vehicle will only experience angular/rotational acceleration. Players will need to reverse both engines to slow down. Of course, there will be a maximum velocity and maximum angular velocity when dealing with all of these forces.

Also of note, there is a real-world relationship between distribution of mass and how quickly something can rotate. As more mass is concentrated to the center, an object is easier to torque. Conversely, in our case as engines apply the torque and likely have mass themselves, it's easier to torque an object if the force is applied further away from the center of mass; moment arm length and the like. While we might have edge cases based on vehicle width, engine distance, and the ratio of engine to vehicle mass- having two relationships in direct opposition to one another, it's possible this system is too complex for proper appreciation in *Ascengine*.

Mechanical Complications

It isn't necessarily possible to simply rotate a vehicle about its vertical axis (yaw?). When adjusting rotational orientation in more than one direction, say adjusting the pitch so the nose of the vehicle is higher and also adjusting the yaw in an attempt to turn the vehicle left or right, that yaw modification is applied to the original position/orientation vertical axis rather than the appropriate direction based on the current and relative orientation after the pitch was changed.

This is fairly important for the sake of accurately traversing sloped terrain on a vehicle, as the rotation always needs to happen relative to what is "up" for that particular vehicle's orientation in that instant. This rotation change needs to **happen after** pitch and roll modifications based on varying levels of the surface(s) beneath the vehicle at different points which is information gained from raycast hit distances.

Furthermore, all of these rotations need to happen before adding to the appropriate acceleration values. However, the way these values interact with the relative space of the vehicle in relationship to the world is complicated. On one hand, we want to maintain the horizontal plane momentum that the vehicle has, and adjust it so it stays horizontal relative to the vehicle's orientation. But on the other hand, by necessity on a curved and sloped surface, these movement vector components will definitely have vertical components as the vehicle's rotation changes. Unless we completely ignore vertical components completely, do we even have acceleration that isn't just visual in that direction? I don't think so.