ZebraReality 0.1 - 0.3

Forrest Eli Hurley

December 2017

ZEBRACORNS

FRCTEAM900

A Zebracorn Labs Paper

1 Introduction

Team 900 has lots of programmers. A positive cornucopia of programmers. Unfortunately, this often leaves us short on the mechanical side of things. Even after we get our robot built, the lack of a convenient practice field makes it difficult to test all of our robot's functionality or train drivers.

We decided to program a solution.

To test some robot functionality (or perform early prototyping) and train drivers, we built a virtual reality simulation of our robot and the FRC game. This document contains some of the lessons we learned in the process.

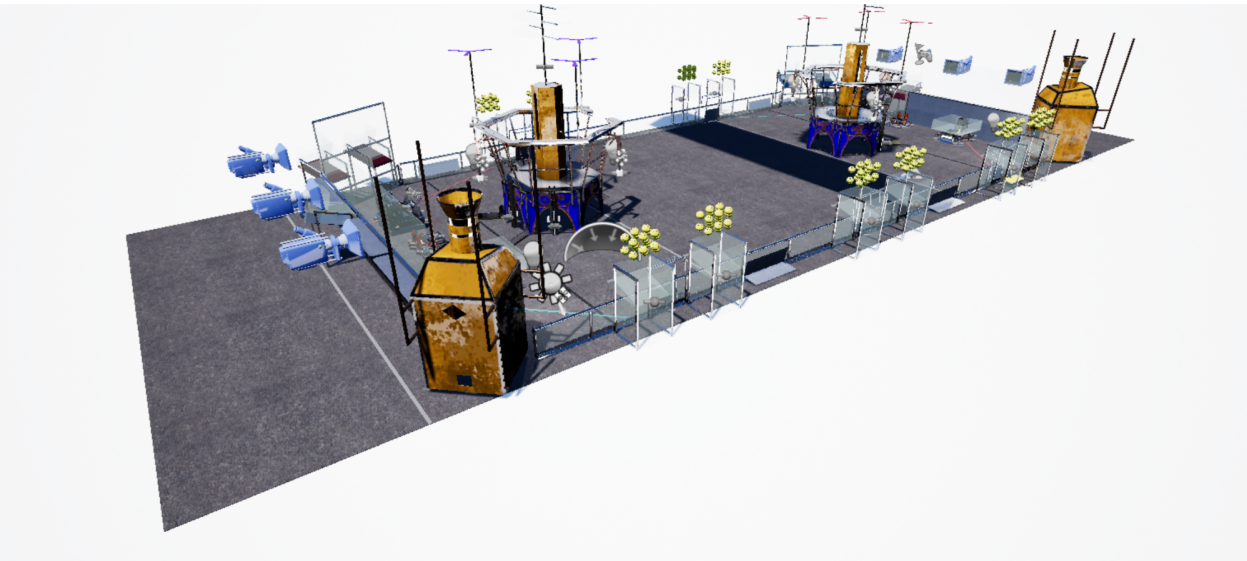

Figure 1.1: The field with robots in starting positions. This was the final appearance of our simulation, with all functionality fully implemented.

2 Drivetrain Modeling

When simulating FRC robots, we decided to focus on making the drivetrains as realistic as possible. The primary purpose of robots in nearly every FRC game is to move things from point A to B, do tasks in multiple locations, or just to move around really fast. The basic FRC robot is often a drivetrain with no additional mechanisms. As such, our top

priority when simulating FRC games was to simulate the drivetrains realistically.

To build these accurate drivetrains we iterated through several different methods, starting with raw physics simulations of convex hulls and eventually moving to the NVIDIA PhysX vehicle model. This model natively supports any type of drivetrain without omni or mecanum wheels. It is extensible to support these as well, but this requires adding custom physics shaders to calculate the forces on the wheels. We have not yet implemented these into the simulation.

Tank and Swerve drive, our team's preferred drivetrains, behave accurately when implemented in this framework. We were able to adjust friction and robot masses, drive into other robots and push them around, and get stuck on obstacles in the same manner as a real robot.

3 Immersion and Visual Realism

The realistic appearance of the simulation takes a far backseat to the realism of the physics. However, there are several situations in which it turns out to significantly improve the simulation as a method for driver training. For example, the alliance station wall should be clear. Really. In case you aren't convinced without evidence, see below.

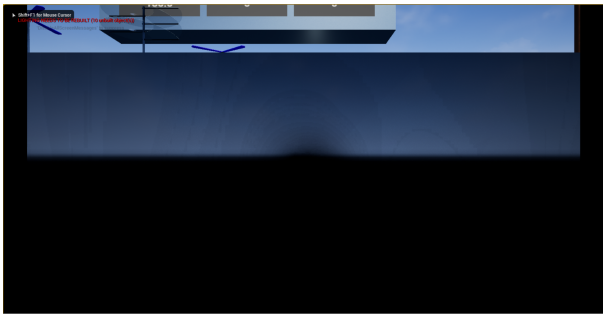

Figure 3.1: Opaque alliance station wall

Figure 3.2: Transparent alliance station

Several recent FRC games have had many potential obstacles to the driver's field of view. Virtual reality can be a great tool for understanding the visibility of a game field before setting foot on one. In 2017, the airships were directly in the middle of the field, but they had clear regions along their base. It could be possible for a driver to see through that

region by standing in a slightly different location, or by crouching. As such, if a team's strategy or mechanism is dependent on having high visibility across the entire field, they need to have that visibility. If the airships are more substantial than expected, driver(s) would likely have trouble maintaining effective control. While static images of the field can also help with understanding the obstruction of obstacles, it is difficult to know how a driver or coach will be able to move to be able to see around obstruction, or what short object will actually entirely eclipse small robots behind them.

The second area where visuals are vital is in improving immersion for driver training. A realistic scene can make a driver feel like they are actually on the field. However, the most important factor for immersion isn't detail or realism of objects. Instead it is frame rate and response time. It is generally agreed within the VR community that if a virtual reality experience has under 90 apparent frames per second, it is more likely to cause nausea in a user and feels less responsive.[1] The responsiveness of the simulation to head movements is probably more important for preventing nausea and custom rendering pipelines have been built in game engines such as UE4 or Unity to allow today's hardware to support the pixel throughput necessary. FRC games with many objects or 3d models with huge vertex counts can all slow down the frame rate and responsiveness of VR simulations to a crawl. It is vital to optimize 3d models and find ways to simplify the physics and appearance of repeated objects to effectively train drivers.

[1] https://www.polygon.com/2016/3/17/11256142/sony-framerate-60fps-vr-certification
https://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/

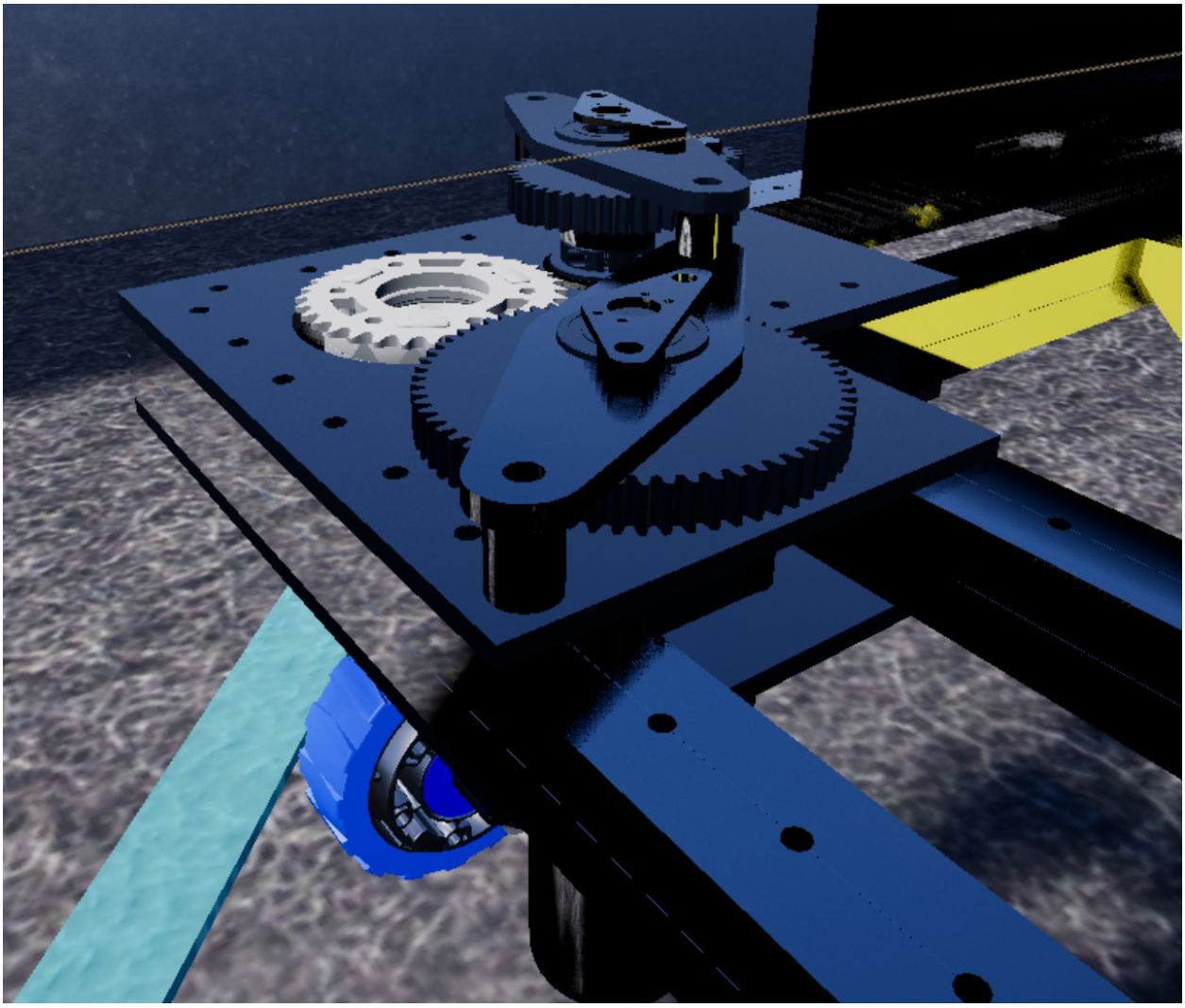

Figure 3.3: A closeup of the swerve module on the simulated robot. Notice that there are no encoders or wires, as these are impossible for a driver to see clearly. The gears and sprockets do not turn or mesh and exist to imply structure to a distant observer.

4 Importing CAD

To pull in accurate models of the field or the robot, we started with digital CAD files in the SOLIDWORKS format. Other software options exist as well and are just as viable as SOLIDWORKS. However our team builds all robot cad in this software, so it is our easiest starting point for importing 3D models into VR. Most CAD formats save lists of operations

applied to primitives or sketches. These can be things such as lofting or extruding. The file formats used in describing models for VR are quite a bit different. They consist of a set of vertices and edges forming polygons in a mesh which is placed in 3D space to construct a visible surface.

The models for visuals can be generally be generated by importing CAD into mesh modelling software such as 3DS Max. However, this shortcut leaves the models with far too many vertices and dense meshes. The high and unnecessary detail reduces the possible frame rates in scenes including the model and are completely unusable for physics calculations. Collisions in physics are calculated using convex meshes. Convex meshes would include sphere and rectangular prism shaped objects, but not say, a torus. As the collision mesh is not seen, it can be relatively inaccurate to the robot and be constructed manually from primitives.

To prepare the robot model to move and support any moving parts in the field, we suggest using skeletal mesh techniques. The separate moving portions of the robot are each attached to a virtual support or bone. Major approximations must be made when doing this. Ignore the small gears and any part that does not have a direct impact on the interactions between the robot and its environment. It is likely that the smaller parts aren't even visible to a driver. The portions of the model can be individually moved in the final simulation with physics affecting the results.

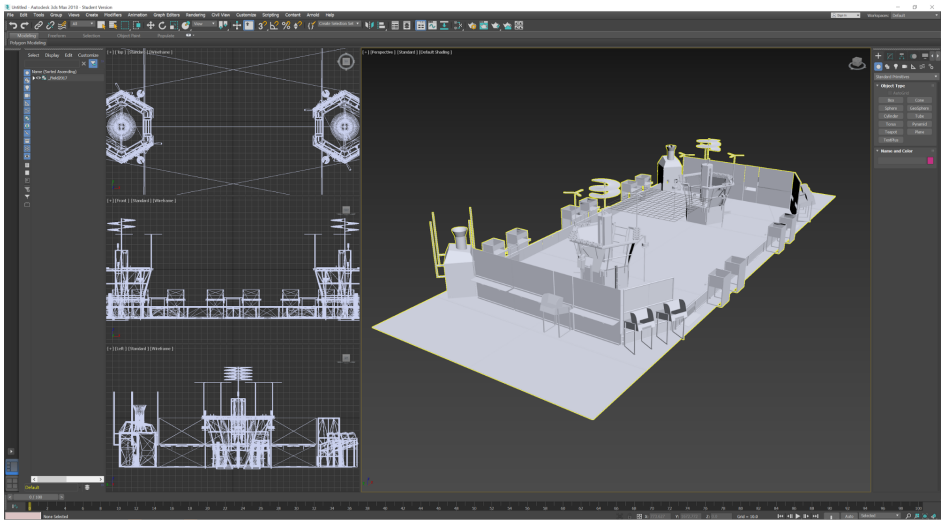

Figure 4.1: The full field immediately after import into 3DS Max.

Within this context of importing 3d objects into VR, one of the biggest areas of difficulty we had was in importing the field CAD provided by FIRST. Because of the way that SOLIDWORKS handles multiple sets of assemblies, some parts of the field were duplicated almost at random. Additionally, other parts were flipped backwards or rotated and moved away from their proper location. This all required careful manual editing and several days of puzzling over how all of the pieces were supposed to fit together. We are not certain if these issues somehow occurred while importing the field CAD into 3DS Max or if it was an intrinsic issue with the CAD released by FIRST.

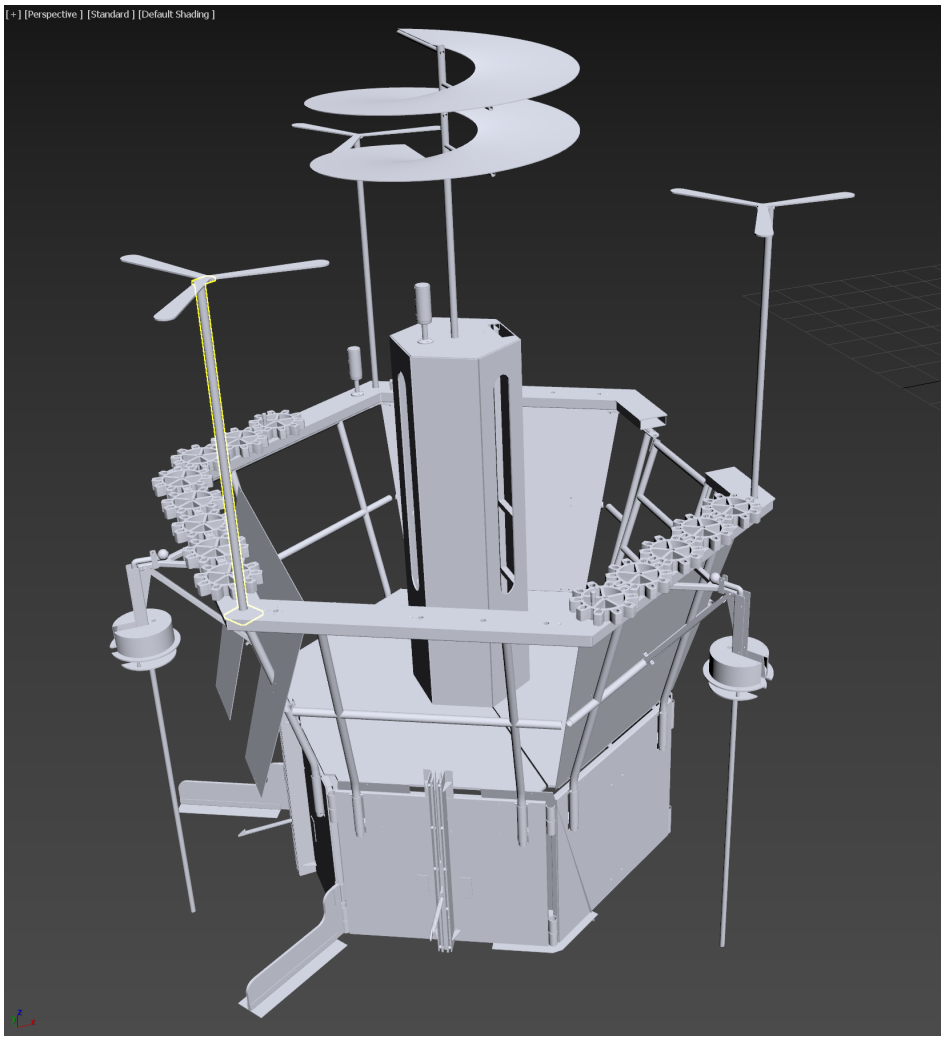

Figure 4.2: The airship in the field CAD provided by FIRST. The polycarbonate panels on its left side are missing or flipped into odd orientations.

5 Closing Remarks and Future Work

Our first year exploring VR ended relatively successfully. Our driver quickly adapted to a variety of field conditions and his clever maneuvering improved our score in several cases. We implemented and did rudimentary tests of every game mechanic and even applied textures created with Allegorithmic's Substance Painter to make the simulation appear more realistic. We went through three iterations, primarily rebuilding the robot drivetrains, finally reaching a stable model.

There are a variety of ways this work could be expanded upon. A multiplayer simulation would allow a driver to actually compete in matches and would allow the sim to approach the effectiveness of actual competitions at training drivers. Alternatively, or perhaps additionally, AI controlled robots could challenge the player to do more than just drive from point A to B, instead coming up with innovative strategies. It could also serve as a platform for testing high-level AI components for future implementation on actual robots. Another possible area for expansion is connection with ROS nodes or logged data. This incorporation would allow ZebraReality to be used as a possible alternative to Gazebo or to visualize past FRC matches for analysis.

6 Outside Links

For more details on UE4 and its vehicle implementation follow the following links:

- https://www.unrealengine.com/
- https://docs.unrealengine.com/latest/INT/
- https://docs.unrealengine.com/latest/INT/Engine/Physics/Vehicles/VehicleUserGuide/index.html

Below is a link to the PhysX documentation on vehicles as well as the 3D modelling and texturing softwares we used:

- http://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/Vehicles.html
- https://www.allegorithmic.com/
- https://www.autodesk.com/products/3ds-max/overview

The following videos were created with and around various versions of our robot simulation. The teaser video features a simulation of the pilot role, involving picking up and placing gears on pegs. The second video is of the very first prototype of the robot simulation code.

- https://www.youtube.com/watch?v=5QQnIibMcqs
- https://youtu.be/RPwyd24WwTM

7 Acknowledgements

Thanks to our mentors for encouraging this project and helping us to acquire the resources to see its possibilities.

Thanks to Allegorithmic and Autodesk for providing student licenses to Substance Painter and 3DS Max respectively. This software was vital for preparing our robot models to be placed into VR.

Thanks to Dassault Systemes for providing the copies of SOLIDWORKS used to build digital models of our robot.

Thanks to Cooler Master for donating a full tower case to hold all of the components we need to run VR. We used this for one of our main development computers and brought it to competitions to run demos.

Thanks to Epic Games for providing Unreal Engine 4 free of charge for non-profit purposes. It abstracted away much of the lower-level programming and allowed us to focus on creating realistic dynamics and immediately place players into VR.

Thanks to NVIDIA for their ongoing support of Team 900. We appreciate the hardware support they have given us in the form of discrete high-end desktop computing and software support. The physics simulations were primarily built upon the NVIDIA PhysX platform. Everything from collision calculations to the vehicle dynamics were designed around these libraries.

And a final thank you to Lucid Dream VR and Mike McArdle for introducing VR to our FRC team and providing internship opportunities to students. The experience we gained was invaluable in the completion of this project.

ZEBRAREALITY

Have questions? Contact support@team900.org

http://team900.org