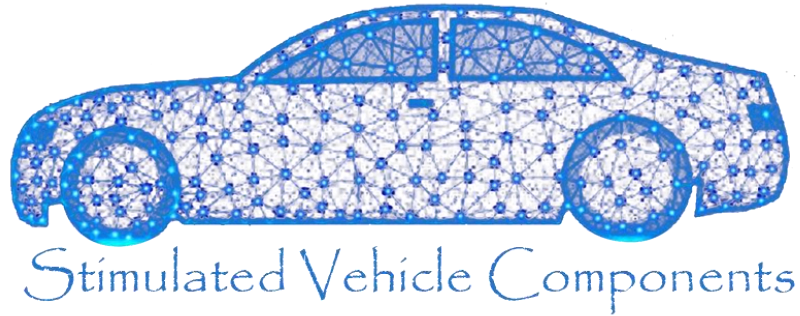


Simulated Vehicle Components



Senior Capstone Notebook

Jose Garza

225008812

Spring 2020

Overview

This document will function as my journal to document my weekly progress, my team contributions, and the overall accomplishments I have made. Additionally, I will add any valuable documentation and tutorials that could aid the future of this course. Given that many of the technologies mentioned in this team are not common, it is important to clearly document proper uses for these tools to allow for a more efficient development process for the future semesters of this program.

Weekly Scheduled Notes

Class 1: (Jan 14)

In this first lecture, Dr. Song discussed the different projects we can apply. We discussed the AutoDrive competition and what each team will be working on and what responsibilities they will hold. We were made to rank our most interesting choices, but beyond that, we did not work on any assignments.

Class 2: (Jan 16)

We had to have had the CV and portrait submitted by Jan 17. In class, we talked further into ROS and its capabilities, yet we had not yet been assigned to a specific team.

Class 3: (Jan 21)

We were assigned into teams, where I was placed in the Simulated Vehicle Components team. In this class period, we chose the team leader, talked to Aaron

about what hardware requirements were needed for us to be able to run Matlab, Unreal Engine, Simulink, and Ros. Given that I did not have a windows computer, which is needed for the Simulink/Unreal connectivity, I went to the software center to get Windows.

After several hours of exploring possible options, I found it easiest to get a new laptop give that my current one was out of memory to install windows.

Class 4: (Jan 23)

In this class period, I believe, we took our teamwork quiz over the book *The Five Dysfunctions of a Team* by Patrick Lencioni. Overall, the quiz was fairly direct, given I read the book, it was a fair quiz.

Class 5: (Jan 27)

We talked about Ros integration with Aaron and also saw a demo of how ROS and Unreal worked together. Although we didn't have a system of our own to test out the environment, we followed Aaron and his demonstration to understand what we would be working with.

Got Quiz grade (94).

Class 6: (Jan 30)

I had received my new laptop, so I was finally able to sit down with Aaron and still all the needed components to be able to start working on the project.

The followed the

documentation for installing the Vehicle Dynamics Blockset Interface for Unreal Engine.

Following this link:

<https://www.mathworks.com/help/releases/R2019b/vdynblks/ug/support-package-for-customizing-scenes.html>

we were able to set up the foundation for the AutoVrtlEnv project.

As noted in the documentation:

1. To install all the components, you first need to have a windows OS, and install MATLAB.
2. On the MATLAB Home tab, in the Environment section, select Add-Ons > Get Add-Ons.
3. In the Add-On Explorer window, search for the Vehicle Dynamics Blockset Interface for Unreal Engine 4 Projects support package. Click **Install**.

4. Following the code provided, we simply had to make subtle adjustments to make it run correctly on our machines:

- a. Update, if necessary, the location of the Unreal Engine's location
- b. Additionally, once you run the program for the first time, you can comment out step 4 since it no longer needs to copy the support files to the project's directory
- c. Lastly, step 5 will open the Unreal Engine, so after this step, the Matlab set up is complete.
- d. Future note: Matlab will be used now to open Unreal and/or add scene images and maps to the valid scene names.

Class 7: (Feb 4)

We had struggles getting the whole team on the same page, mainly due to memory constraints or improper set up.

Key notes would be:

- a. Aim to have at least 100 GB allocated for the Unreal 4.19/Matlab/etc.
- b. Follow the steps in their proper order

We came together as a team to see how we could overcome each member constraints.

Robert and Xiaoyu get an external hard drive while Robert and I used our window's machine.

Class 8: (Feb 6)

We now all had the right hardware, memory available, and all had installed Matlab and Simulink. The next step was to install:

- Ubuntu
- SSH
- ROS
- Catkin

Ubuntu Installation

Follow this link: <https://www.osboxes.org/ubuntu/>

Make sure to choose version 16.04

Follow this link: <https://www.illuminiastudios.com/dev-diaries/ssh-on-windows-subsystem-for-linux/>

SSH Installation

SSH on Windows Subsystem for Linux (WSL)

To get the ssh server working properly, you must uninstall and then reinstall it using the following command:

- `sudo apt remove openssh-server`
- `sudo apt install openssh-server`

- Check the status of the ssh service:

```
Service ssh status
```

- If you see: * `sshd` is not running
 - Then run this command:

```
Sudo service ssh start
```

-
- If you see: * `sshd` is running
 - Then run this command:

```
Sudo service ssh –full-restart
```

Can use 'sudo !!' to generate new host keys

ROS Installation

Following this link: <http://wiki.ros.org/kinetic/Installation/Ubuntu>

Configure your Ubuntu repositories

Setup your sources.list

Setup your computer to accept software from packages.ros.org.

- `sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'`

Set up your keys

- `sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654`

Installation

- `sudo apt-get update`

There are many different libraries and tools in ROS. We provided four default configurations to get you started. You can also install ROS packages individually.

In case of problems with the next step, you can use following repositories instead of the ones mentioned above ros-shadow-fixed

- ROS, rqt, rviz, robot-generic libraries, 2D/3D simulators, navigation and 2D/3D perception

- `sudo apt-get install ros-kinetic-desktop-full`

To find available packages, use:


```
apt-cache search ros-kinetic
```

Initialize rosdep

```
sudo rosdep init
```

```
rosdep update
```

Environment setup

It's convenient if the ROS environment variables are automatically added to your bash session every time a new shell is launched:

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

With this, you have installed what you need to run the core ROS packages

Catkin

Tutorials: <http://wiki.ros.org/catkin/Tutorials>

More Information: http://wiki.ros.org/catkin/conceptual_overview

catkin is the official build system of ROS and the successor to the original ROS build system, rosbuilt. Catkin was designed to be more conventional than rosbuilt, allowing

for better distribution of packages, better cross-compiling support, and better portability. Catkin is included by default when ROS is installed. Catkin can also be installed from source or prebuilt packages. Most users will want to use the prebuilt packages but installing it from source is also quite simple.

Install Prebuilt Package

If you are using a ROS binary distribution on Ubuntu then you can install catkin with apt-get:

```
sudo apt-get install ros-melodic-catkin
```

Most ROS installations will include this by default.

Catkin Workspace

A catkin workspace is a directory where files and packages can be created/modified to be used for ROS. Catkin workspaces make it easier for a user to build certain files and run them.

Sourcing Setup

```
source /opt/ros/kinetic/setup.bash
```

Creating/Building Catkin Workspace

```
mkdir -p ~/catkin_ws/src  
  
cd ~/catkin_ws/  
  
catkin_make
```

Sourcing devel and setup

```
source devel/setup.bash
```

To make sure your workspace is properly overlayed by the setup script, make sure `ROS_PACKAGE_PATH` environment variable includes the directory you're in.

```
$ echo $ROS_PACKAGE_PATH  
  
/home/youruser/catkin_ws/src:/opt/ros/kinetic/share
```

joseunix@LAPTOP-7DG5D5F4: ~/catkin_ws

```
1 sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -cs).list.d/ros-latest.list"
2 sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-keys 042ED6FBAB17C654
3 sudo apt-get update
4 sudo apt-get install ros-kinetic-desktop-full
5 sudo rosdep init
6 rosdep update
7 echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
8 source ~/.bashrc
9 git clone https://github.com/tamu-autodrive-common/Fall2019
10 pwd
11 ls
12 source /opt/ros/kinetic/setup.bash
13 mkdir -p ~/catkin_ws/src
14 cd ~/catkin_ws/
15 catkin_make
16 source devel/setup.bash
17 ls
18 cd ..
19 ls
20 cd Fall2019/
21 ls
22 cd local_planning/
23 ls
24 cd ..
25 ls
26 cd local_planning/
27 ls
28 ln -rs autonomous_vehicle/ ~/catkin_ws/src/
29 cd ..
30 ls
31 cd Fall2019/
32 ls
33 cd ..
34 cd catkin_ws/
35 ls
36 cd src/
```

General flow of command line while installing packages

Class 9: (Feb 11)

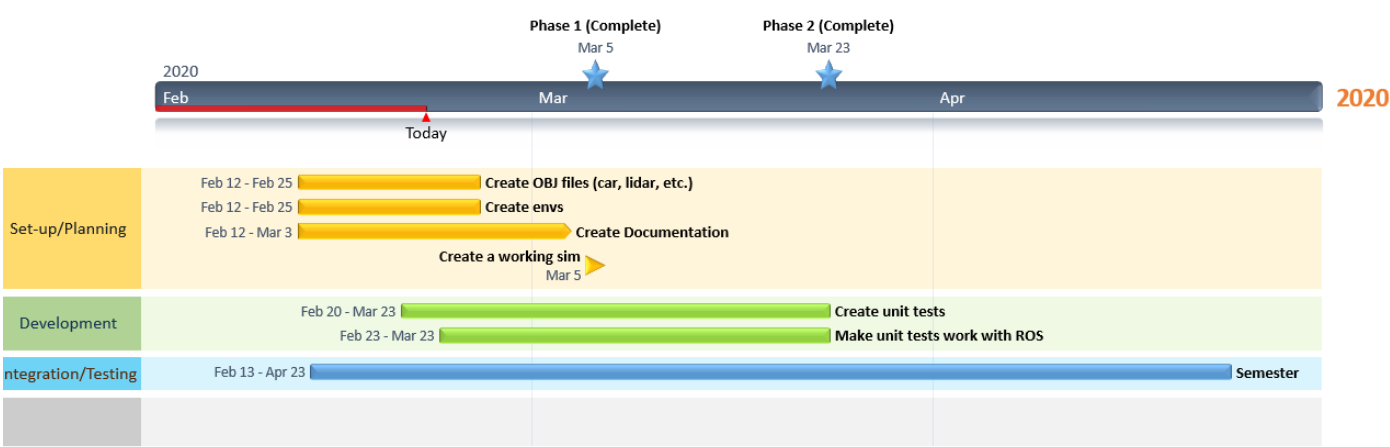
Proposal presentations (other teams)

Cleaned up project proposal.

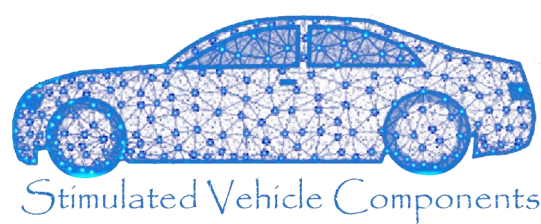
I created more sketches for team management purposes:

For instance:

Simple Gantt Chart:

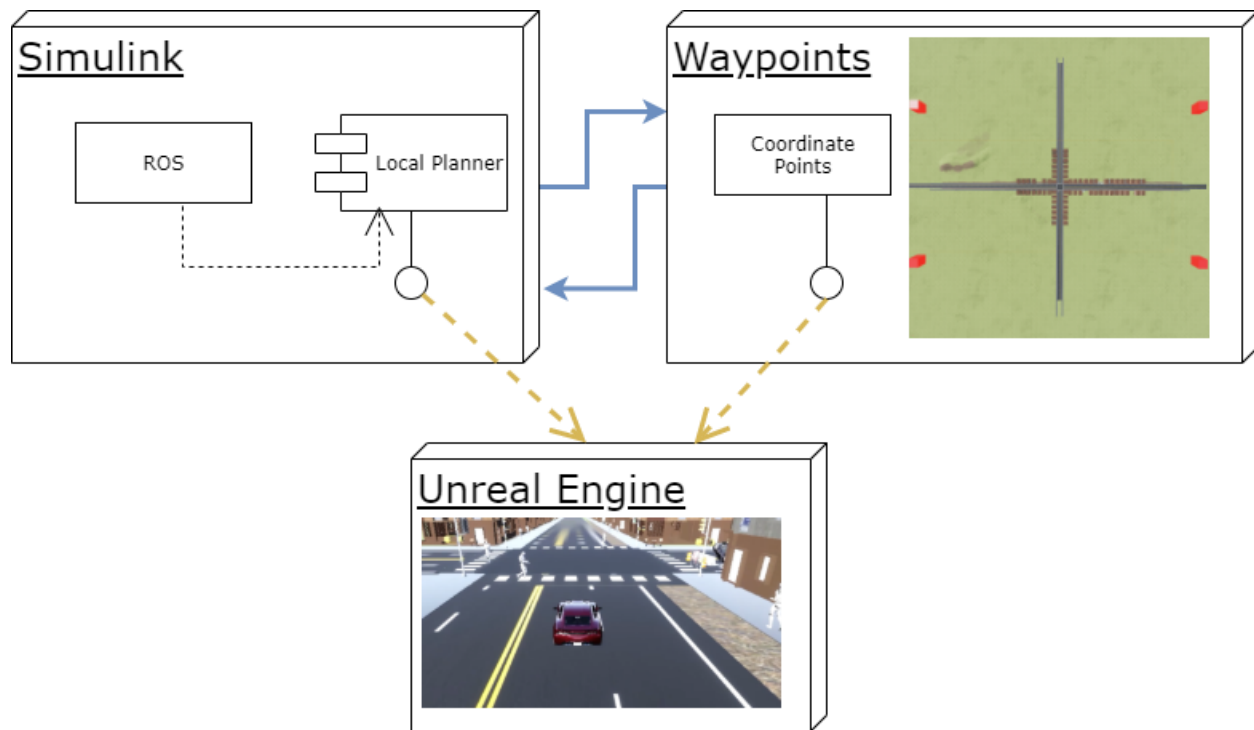


The Team Logo:



Virtual Car with Team Name

System Level Design:



Functional Flow Diagram

Class 10: (Feb 13)

Proposal presentations (our team)

Project proposal

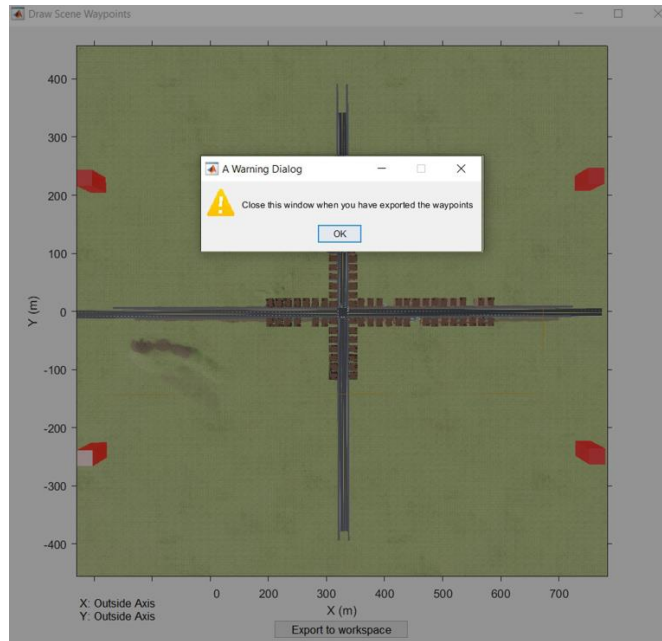
Class 11: (Feb 18)

Having Ubuntu (to run catkins), Catkins (to run ROS) and Ros (the AI of vehicle) set up, I was now able to properly run the Simulink program and run an environment in the Unreal Engine.

This class period we worked with Aaron to have a clearer concept of the interaction between all of the components. There were several behaviors you needed to know in order

to not run into problems. The errors I was able to record were as follows:

- Don't run the Unreal Simulation unless the Simulink has compiled and is *initializing*
- Don't close out of the waypoints window



Map greyed out for clarity

This window displays a pop-up warning; however, it is important to not close it until you have created the way points.


To do this, click anywhere on the street and the pop-up will hide behind the display.

Class 12: (Feb 20)

In this period, and at home, I decided to make a clear guide on the set up of the system from the start.

ROS

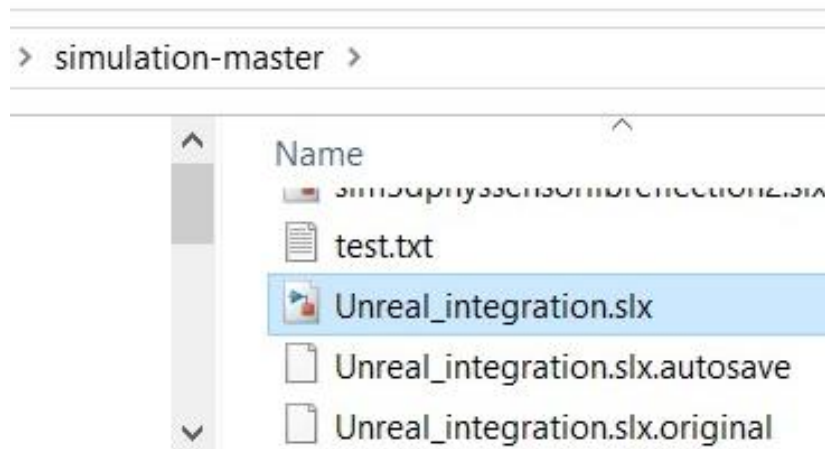
- Start Ros
 - Open up two Ubuntu windows
 - Ubuntu window 1:
 - `cd catkin_ws/` (this will nav to catkin workspace)
 - `sudo ./additional/local` (access local and request global path)

A terminal window screenshot showing the steps to set up ROS. The user is in a directory named 'catkin_ws' and runs 'dir' to show its contents, which include 'Fall2019'. Then, they run 'cd catkin_ws/' to change to the workspace directory. Finally, they run 'sudo ./additional/local', which outputs two INFO messages: '[1588661917.282809100]: Spinning planning module' and '[1588661920.206368400]: Requesting new global path'.

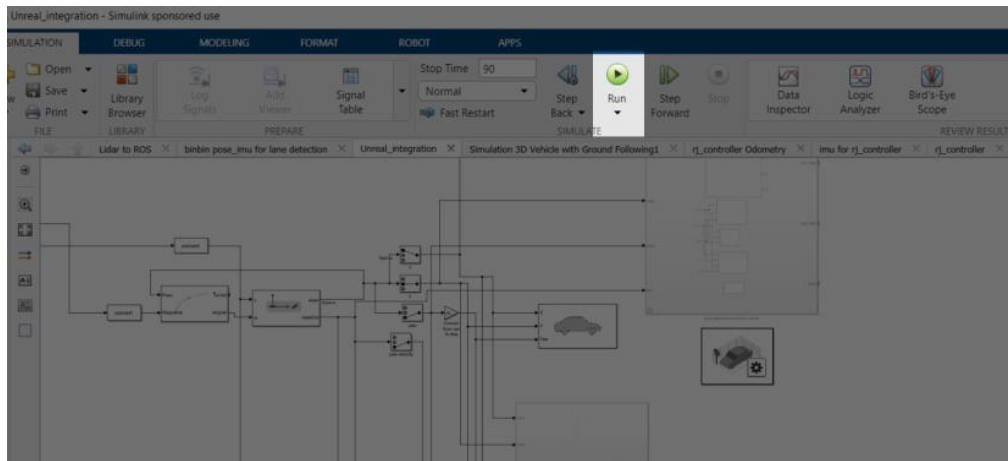
- Ubuntu window 2:
 - `roscore` (note: this is no longer necessary with updated Simulink)


```
joseunix@LAPTOP-7DG5D5F4: ~  
joseunix@LAPTOP-7DG5D5F4:~$ roscore  
... logging to /home/joseunix/.ros/log/6c1fc446-8e9e-11ea-899c-0  
Checking log directory for disk usage. This may take awhile.  
Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage is <1GB.  
  
started roslaunch server http://LAPTOP-7DG5D5F4:61194/  
ros_comm version 1.12.14  
  
SUMMARY  
=====  
  
PARAMETERS  
* /rostdistro: kinetic  
* /rosversion: 1.12.14  
  
NODES
```

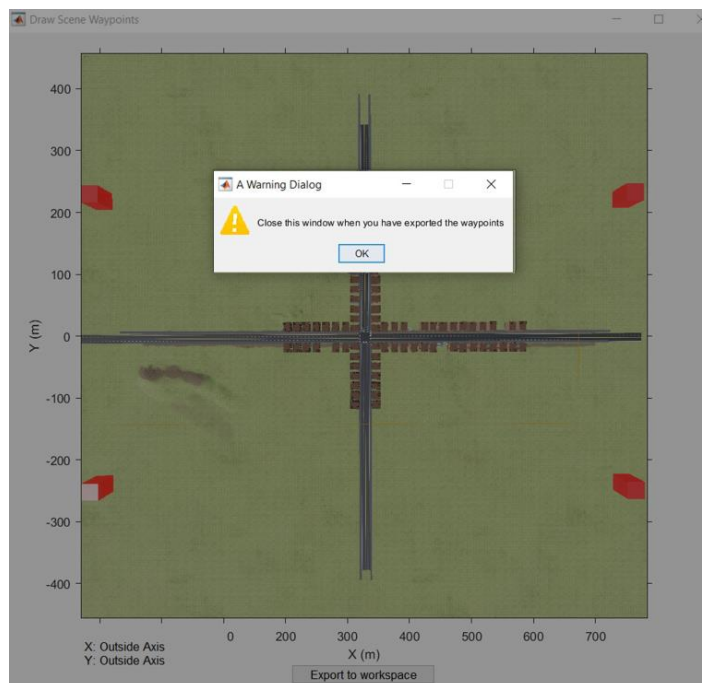
Running Simulink



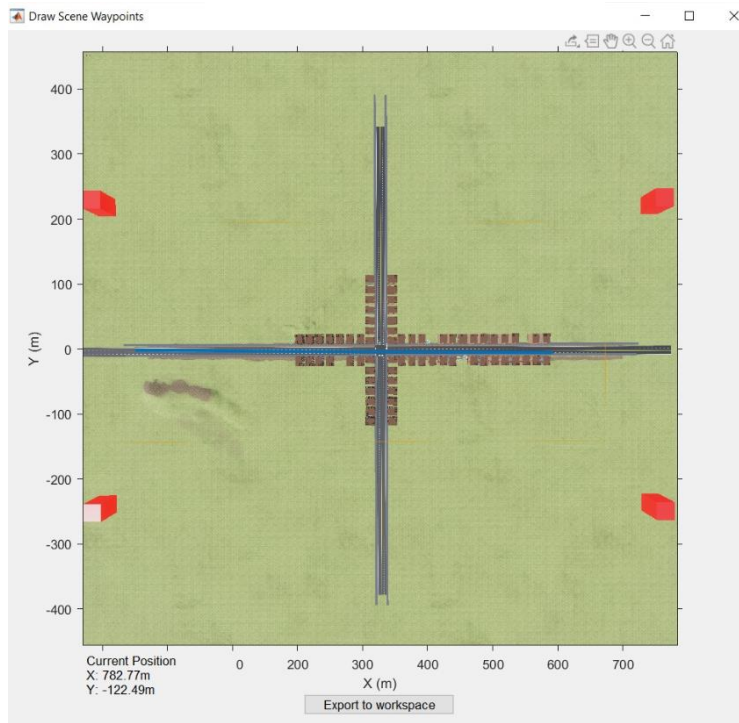
Open up Simulink (Found in Simulation-master repository)



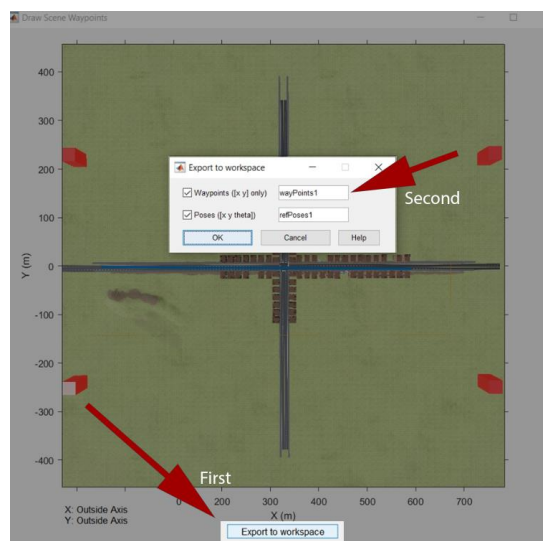
Click Run while in Simulation Tab



Two Displays will pop up. Make sure to not click 'ok'. Click on Map directly



Once you have clicked on the map, you will then be able to generate the way points.



Once you have created the way points, then you can click 'Export to workspace'.

A new pop up will appear. Make sure the waypoints variable name is:

Waypoints: wayPoints

Poses: refPoses

Make sure to remove trailing numbers

Example: Waypoints: wayPoints1 → Waypoints: wayPoints

Additional Notes:

We made improvements to our proposal, so we ran it through Aaron to confirm we were on the right track. We also assigned each other roles, given that we were not too structured during our presentation...

Class 13: (Feb 25)

Set up ROS:

```
joseunix@LAPTOP-7DG5D5F4:~$ cd catkin_ws/
```

```
joseunix@LAPTOP-7DG5D5F4:~/catkin_ws$ ./additional/local
```

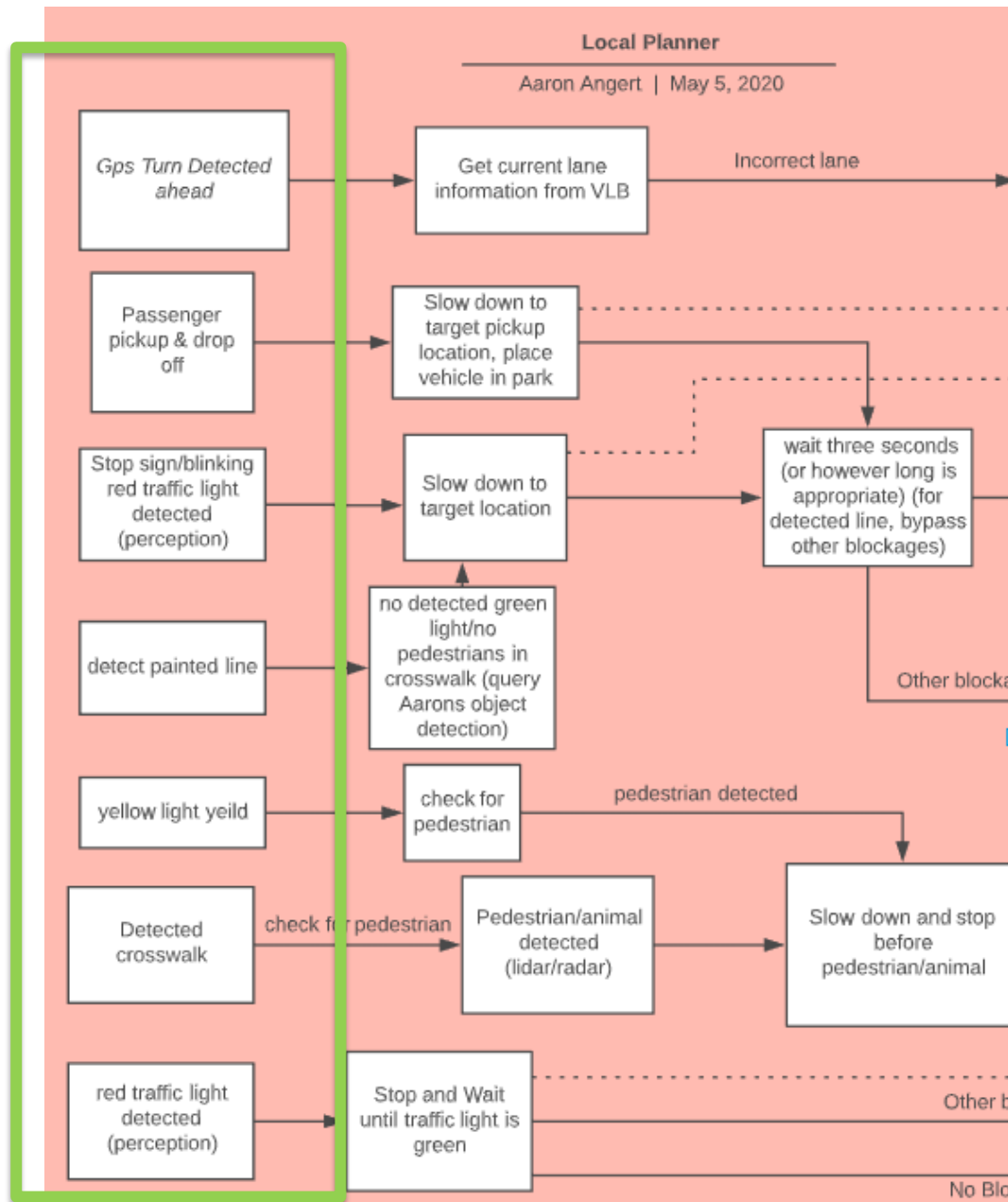
This week we finally began creating environments in the Unreal Engine. We, as a team, decided what we each needed to create to start making clear progress.

Additionally, we focused on making team progress. For instance:

- Show problems we meet in iteration 0 and discuss solutions
- Make sure we have a good idea of how to move forward with the new objectives
- Discuss duties of each member for iteration 1

Map Developments:

1. Red light map setting up
2. Left/Right turn map setting up, but material problem needs to improve
3. Roundabout map setup, but some details like the angle problem need to be fixed
4. Pedestrian map setup



Local Planner by Aaron Angert

The choices for our maps came from the general behaviors that a vehicle would likely undergo while being tested, which are in the green section. We decided to chose popular scenarios and work our way to more specific examples.

Class 14: (Feb 27)

Sources:

Physical Animation / Rag doll TUTORIAL in Unreal Engine 4 latest version

(<https://www.youtube.com/watch?v=Xnx9L6DgfE>)

Life Of A Crash Test Dummy - Ragdoll Crash Dummy Prototype In Unreal Engine 4

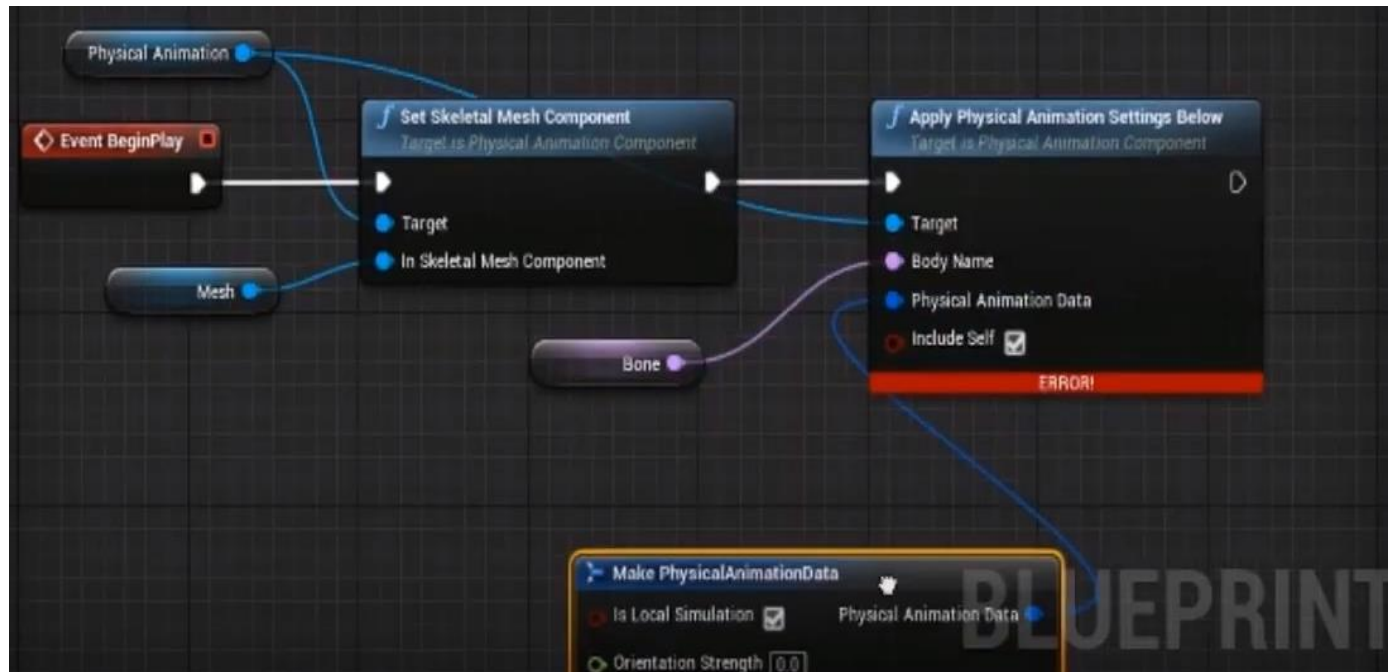
(<https://www.youtube.com/watch?v=ZqjMCAZLiRc>)

Unreal Melee Combat - Animations with Root Motion - UE4 Tutorials

(<https://www.youtube.com/watch?v=iLm6MzJ2ckY>)

Ue4 Tutorial - Moving an Object along a path using a Spline Track

<https://www.youtube.com/watch?v=bWXI91FdMtk>



This week, I focused heavily on making sure I had the environment assets working correctly. It was challenging getting a animations embedded into the actors. After many tutorial videos, I was able to generate walking animations for the pedestrians and add these pedestrians to my map.

Class 15: (Mar 3)

Following the previous tutorials, I was able to develop a cart that allows a mannequin mesh to move around the world. The developer would create a spline path which then the skeletal mesh would follow. This spline can be parented to any object, for instance a car to move other cars, or bikes how I have in my demo.

When the env is ran, the pedestrians will follow the red line. Each pedestrian has a unique offset which will allow them to be scattered throughout the line, which will give a more realistic feel.

Class 16: (Mar 5)

I was having problems allowing the Mesh Components react to physical environments and physical interactions such as gravity. What I ended up doing was making the pedestrian a skeletal mesh component and added the walking animation directly.



The pedestrian is a Skeletal Mesh Component to be interactive

Class 17-18: (Mar 10/12)

SB

Over spring break, I was able to add more meshes to my sketch, make the scenery more realistic and watched many tutorial videos as to help me add buildings and roads.

Class 19-20: (Mar 17/19)

Cancelled

Virtual Semester Start

Class 21: (Mar 24)

CDR presentations

Peer review 1

Reschedule all CDR presentations on the Tuesday of Mar. 24. Our GM advisor joined the presentations via Zoom. The sequence of the presentations remained the same.

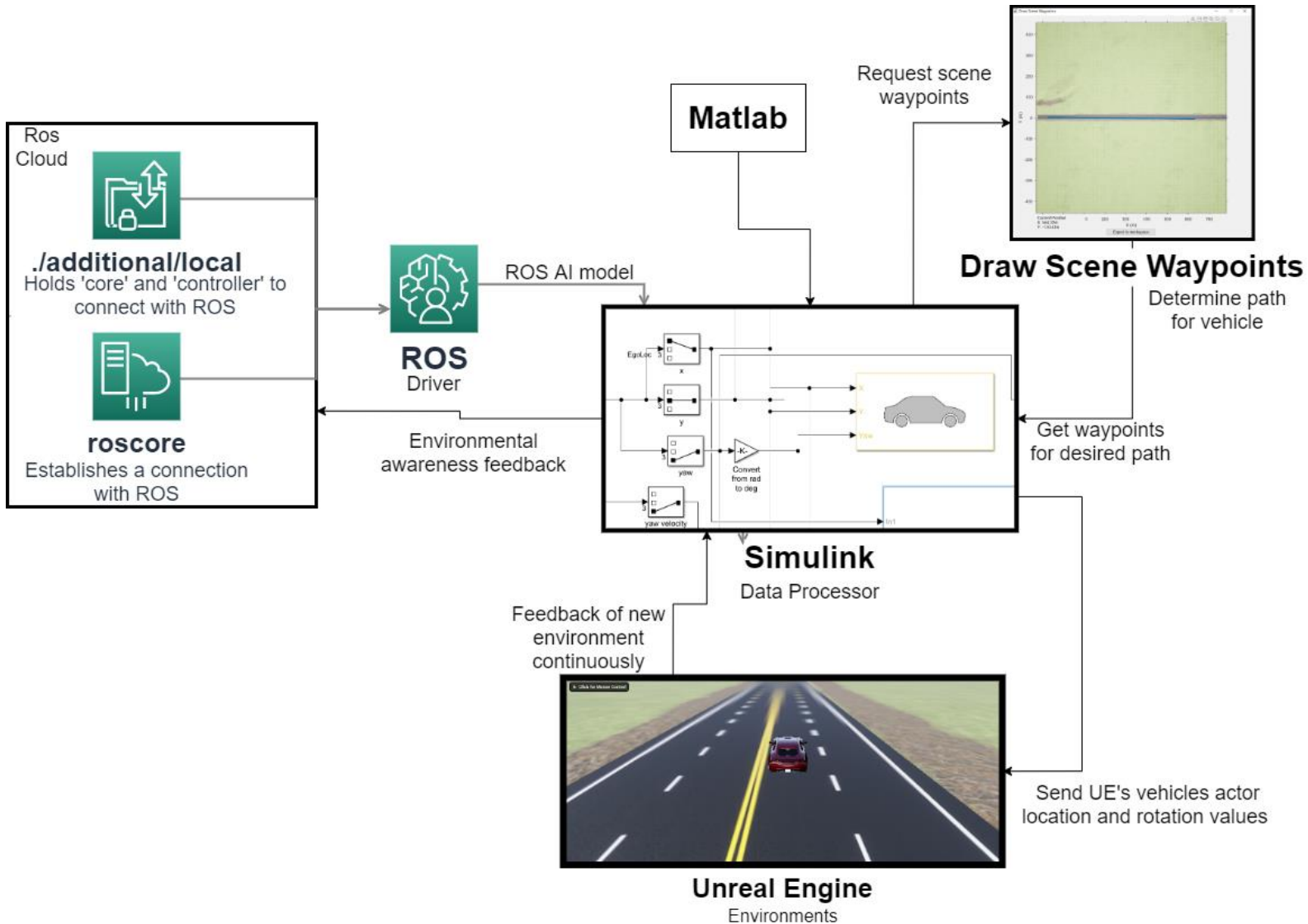
Moved the CDR report and slide deadline to Mar. 25.

Unreal Engine 4 - Basic Movement Input #1

<https://www.youtube.com/watch?v=Lmhbb0ROiqM>

We presented our Critical Design Review PowerPoint to our GM advisor.

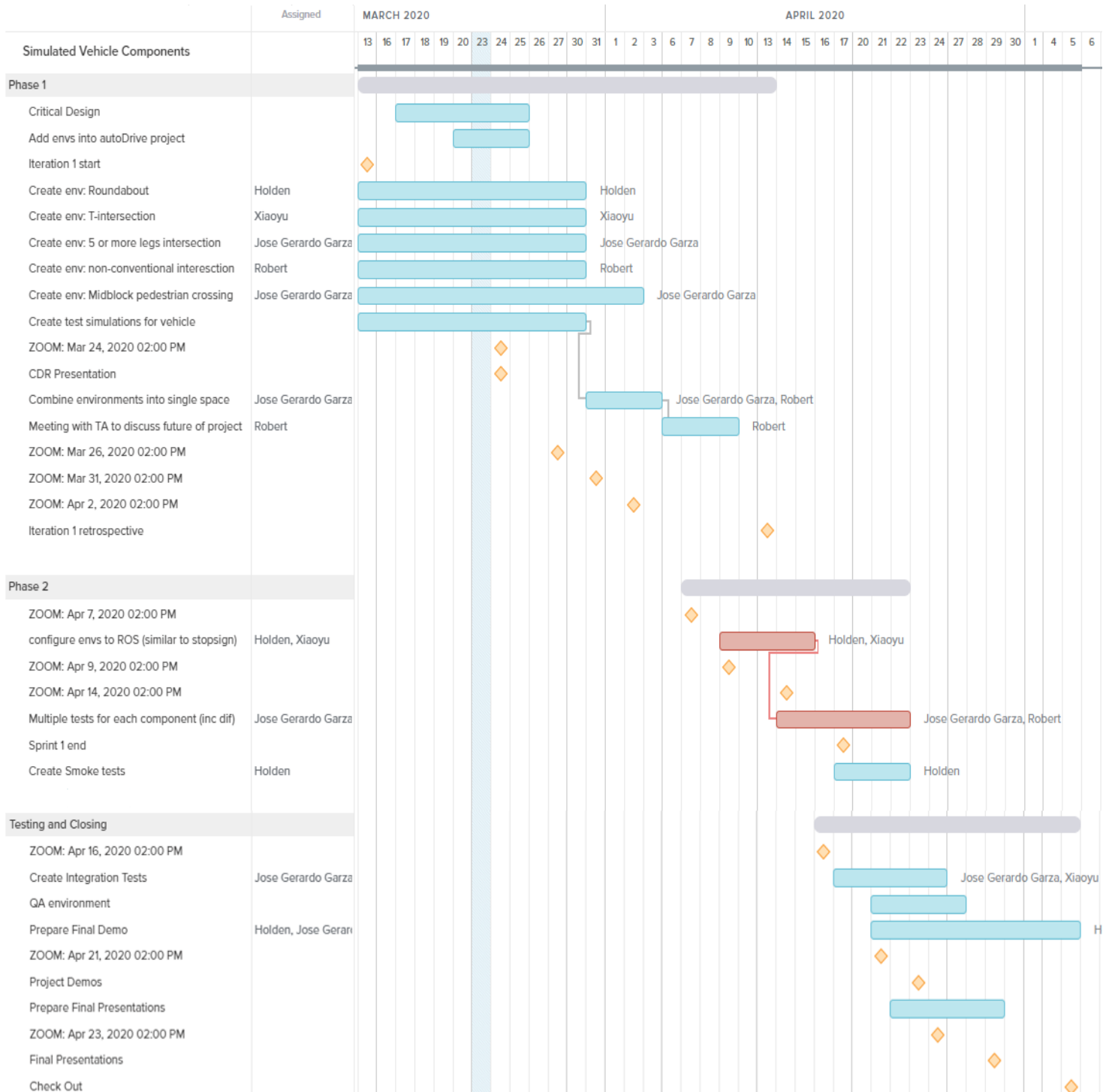
Within this report, I created many illustrations to help visualize the overarching connection of the framework.



Data Flow Diagram of the System

Additionally, to better prepare our project's management. I decided to create a detailed Gantt chart. I used the website (<https://prod.teamgantt.com/gantt/schedule>) to create an aesthetically pleasing yet detailed chart.

TEAM GANTT



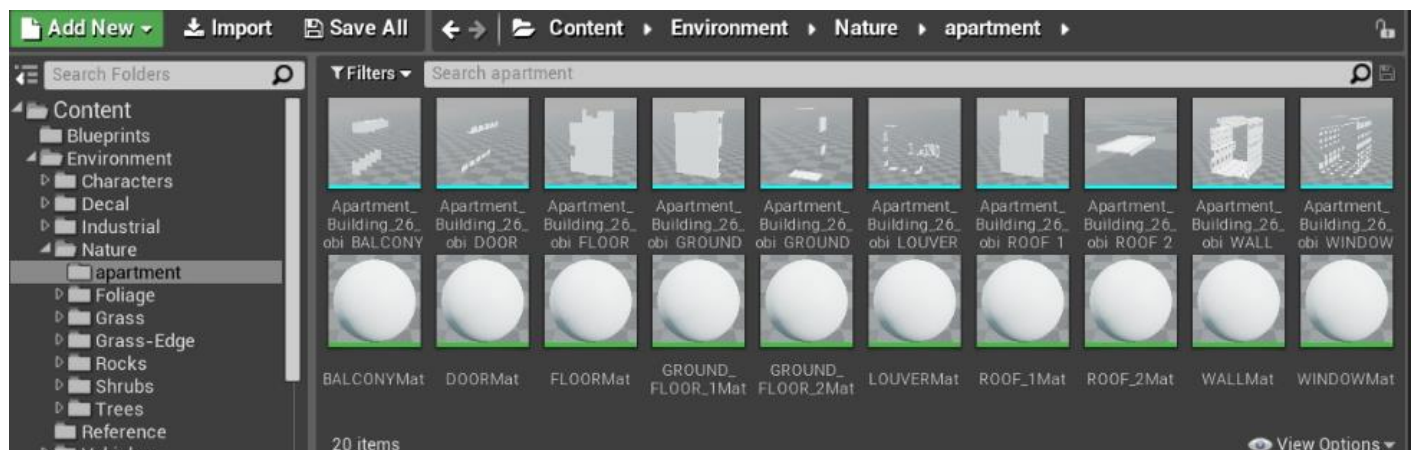
Also, to better depict the general roles in a more deliverable-like manner, I also made a table to illustrate the tasks that were assigned for each member of the team.

Deliverables		
Task	Sub-task	Member
Deliverable 1 - Project Environmental Planning	Sub-task 1 – Create environments:	Pedestrian Crossing Jose Garza Due: 3/25 T- Intersection Xiaoyu Wu Due: 3/25 Roundabout Holden Batte Due: 3/25 Non-Conventional Intersection Robert Osonma Due: 3/25
	Sub-task 2 – Work on Report and Presentation	All members Due: 3/25
	Sub-task 3 – Add all environments in one AutoDrive workspace	Jose Garza, Robert Osonma Due: 4/05
	Sub-task 4 – Zoom meetings	All members Every class periods
Deliverable 2 – Configure ROS	Sub-task 5 – Integrate Stop light env With ROS	Holden Battle, Xiaoyu Wu Due: 4/15
	Sub-task 6 – Create multiple tests for each component	Jose Garza, Robert Osonma Due: 4/20
	Sub-task 7 – Create smoke test to see if all tests work	All members Due: 4/25

Deliverable 3 – QA and Final Demos	Sub-task 11 – Create Integration tests And make sure all components work	All members Due: 4/25
	Sub-task 12 – Prepare Final Demos Presentation and report	All members 4/29
	Sub-task 13 – Prepare for checkout	All members 5/5

Class 22: (Mar 26)

During this week, I began making my pedestrian map more realistic my first adding apartment buildings. After several hours of research, I realized making the apartment buildings might not be ideal, but instead downloading an obj file and added them instead. By doing so, however, the files did not come tied with a color/material. I then needed to either create materials or add my own from other objects. After digging online for free licensed objects, I was able to find clean and simple apartment buildings.



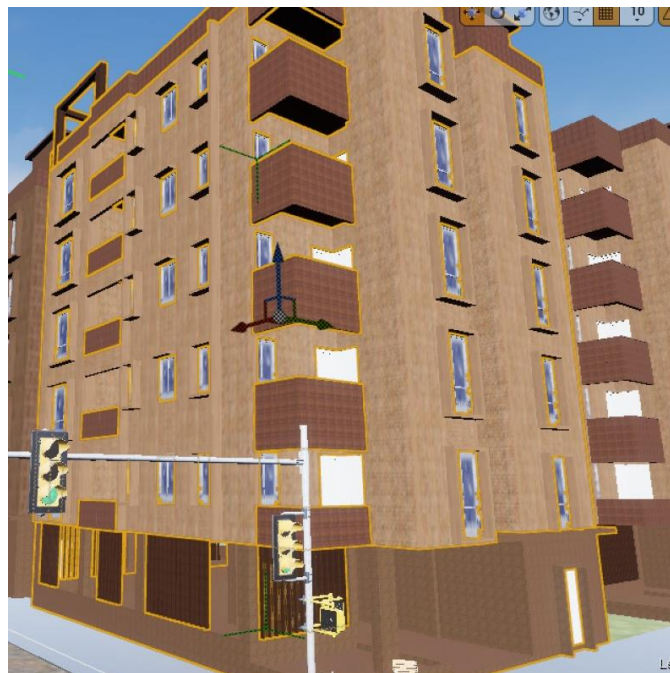
Mesh and Material of Apartment Buildings

Class 23: (Mar 31)

I had to build the apartments using the templated objects. After playing around with the different materials and objects, I was able to generate realistic apartments. I created about 6 different styles to not feel too repetitive.

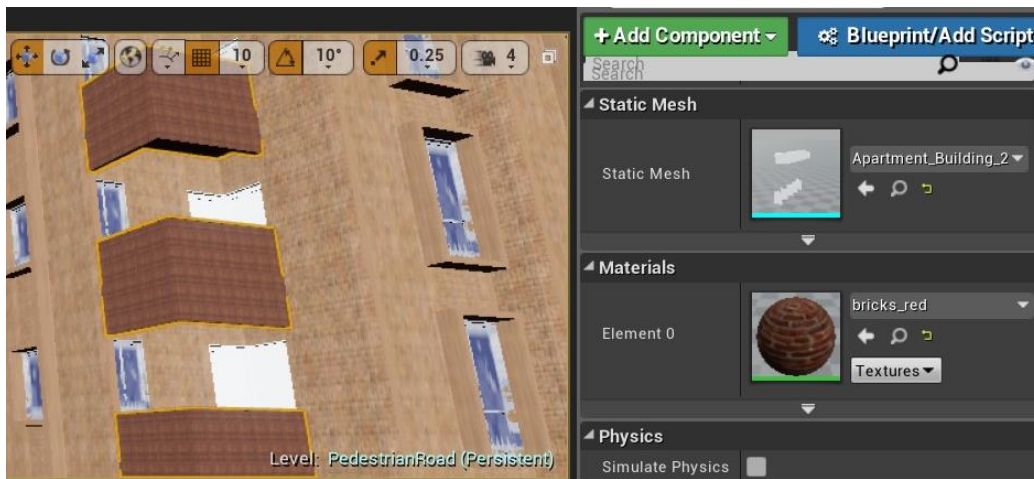


Apartment Building Layout



Overall View of Building

Meshes and Materials



Apartment building balconies with red brick



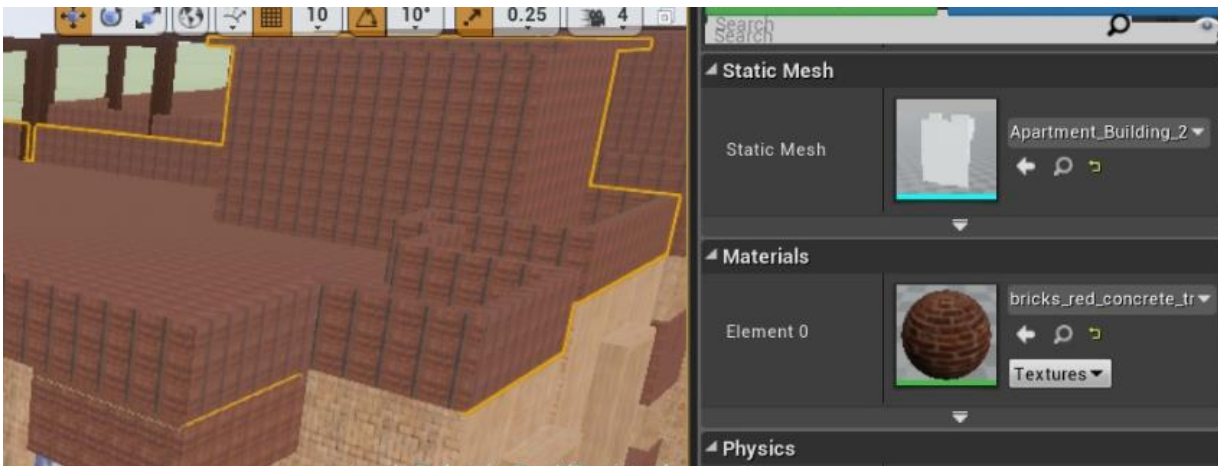
Building framework with orange wall bricks



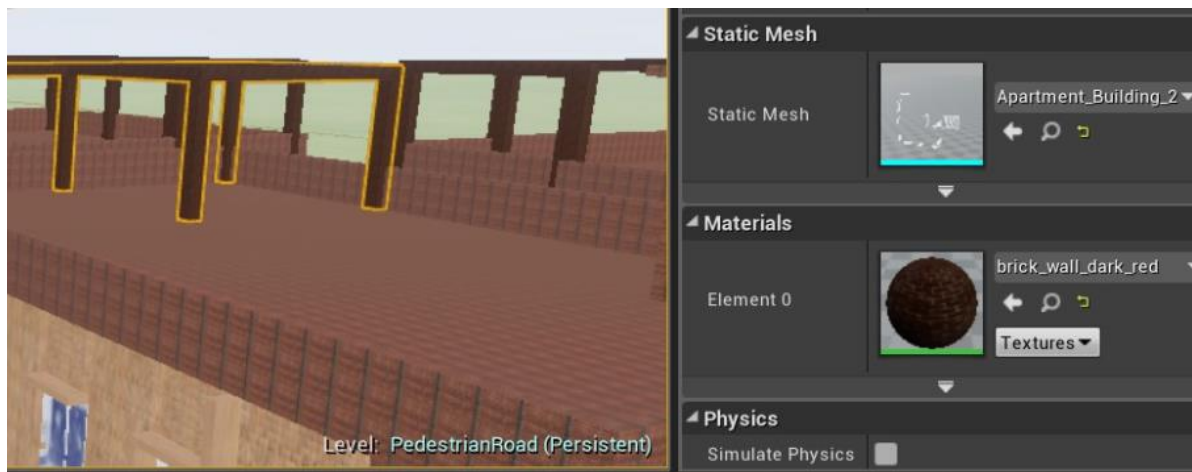
Used crystal blue material for windows



Black or white material for doors



Red concrete brick roof, one of many examples



Dark red brick wall for pavilion roofing

Class 24: (Apr 2)

After I was able to properly add apartment buildings, I then grouped the building into one folder, then duplicated that folder many times to have many apartments throughout my map. This then gave the environment a very lively look.

Additionally, I began adding more street-like properties to my map.



Scene with Tables, Rocks, and Trees



*Scene with Traffic cone, an accident, streetlights
pedestrian sign, pedestrian buttons, bike sign*



Simple Rock and Tree Setting



Different Angle of Accident



*Scene with Pedestrians, Bike, Accident
Buildings, Tables. Stoplights, Crosswalk*



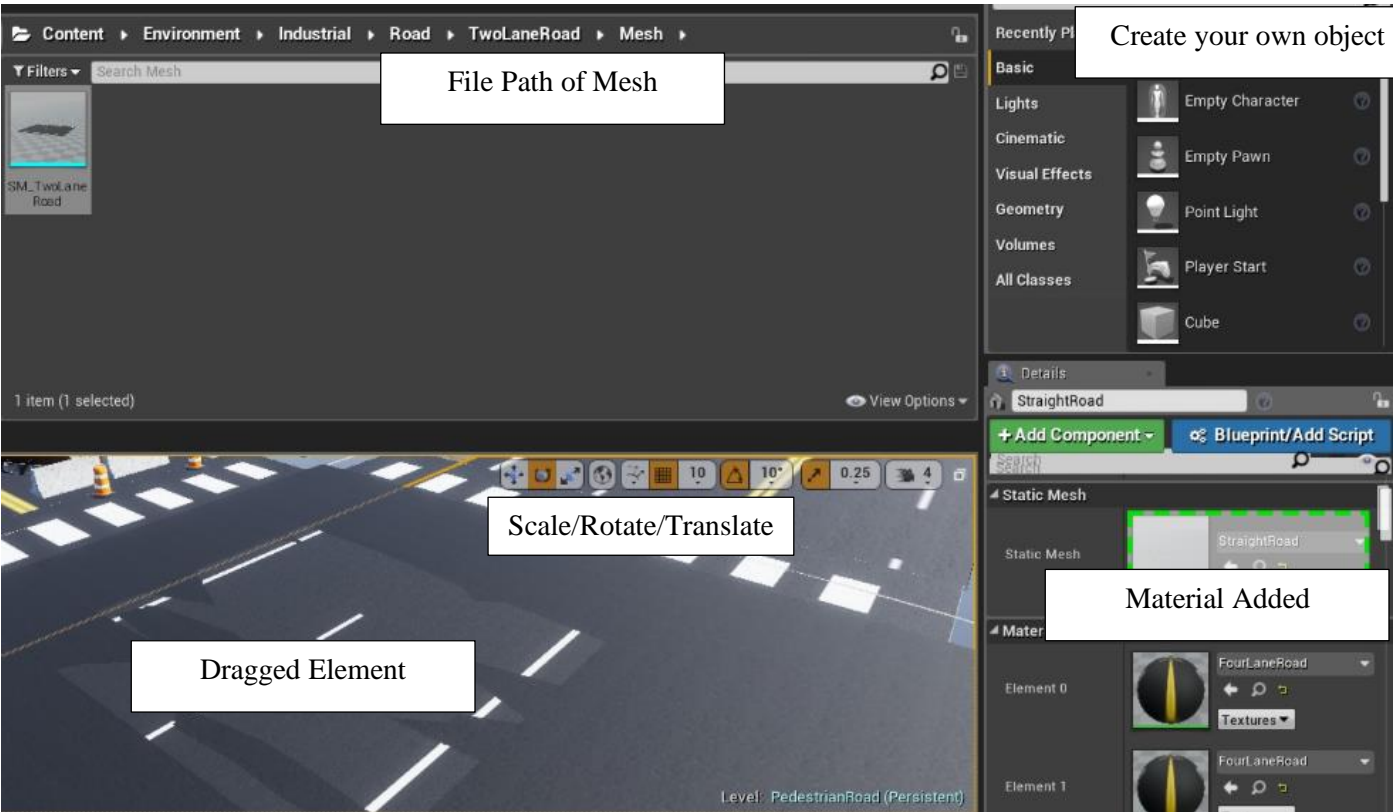
Don't Walk Crossing Sign



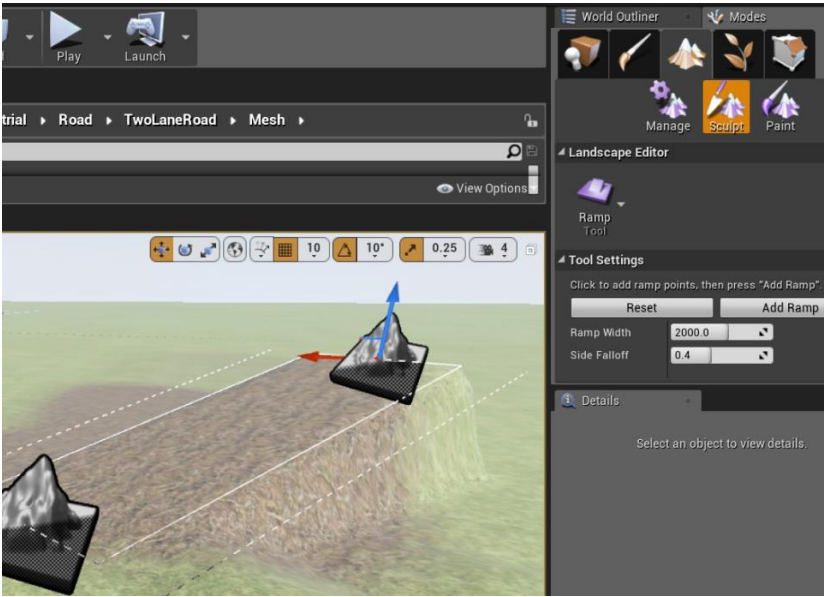
Walk Crossing Sign

Class 25: (Apr 7)

Documentation for simple tutorials:



Simple Labeled Setup



General Model to add landscapes



Pedestrian Controls

The Pedestrian Controls tell the pedestrian skeletal mesh component how to behave.

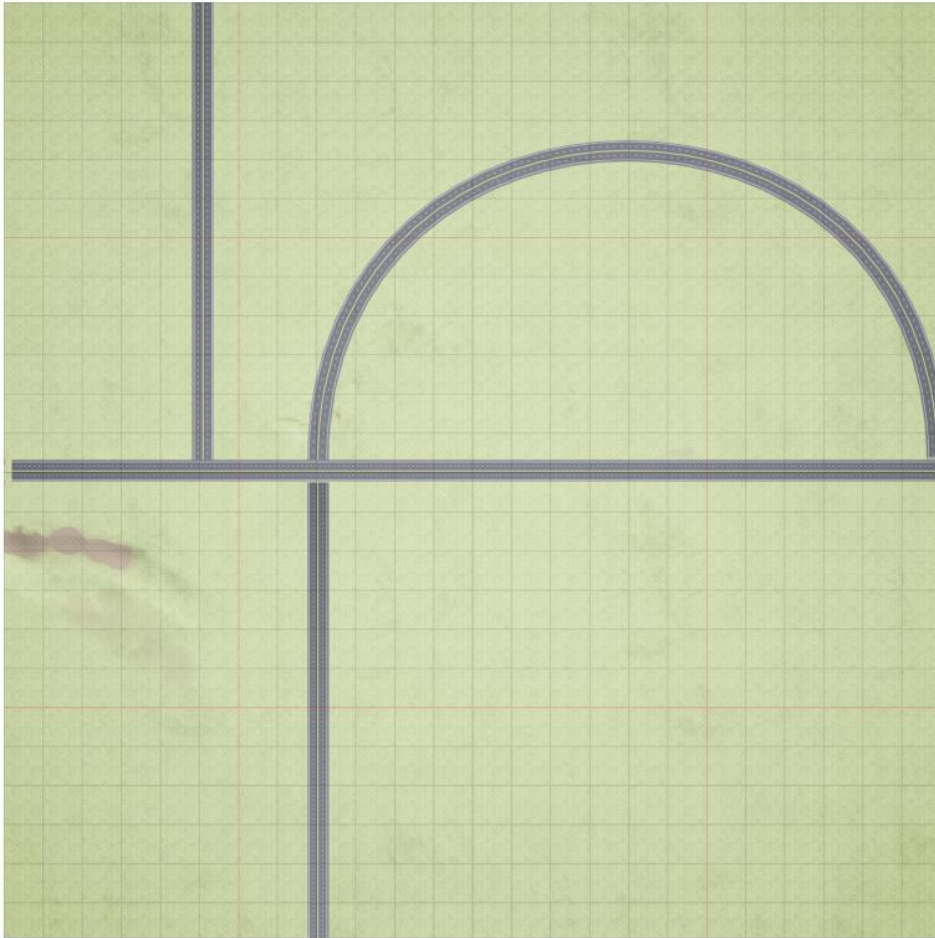
First, you must determine if the pedestrian is active, second, you must determine which spline path to follow. In the pedestrian map, I have three spline paths, one for each sidewalk and one for the crosswalk cycle. Additionally, you can determine what the duration of each cycle is as well as where to start within the spline path. The spline path's length is set as $0 \rightarrow 1$ so an offset of 0.5 will place the pedestrian at the middle of the path at the start of the environment.

Class 26: (Apr 9)

Began discussing screen capture capabilities for way points interface, with Aaron.

I had initially thought of having a drag and drop interface where a user can imitate their map with a software like Photopea or Photoshop. I had created a psd file where there was straight roads, curved roads, and landscapes. Aaron, however, mentioned that it would be ideal if we were able to somehow take a screenshot of the engine and send that

directly to Matlab's tools for the way points.



Initial Design

We then moved forward with the idea of a screen capture. Xiaoyu mentioned we can use to take screenshots. I did some research on this tool and the website (<https://docs.unrealengine.com/en-US/Engine/Basics/Screenshots/index.html>) was helpful for this task.

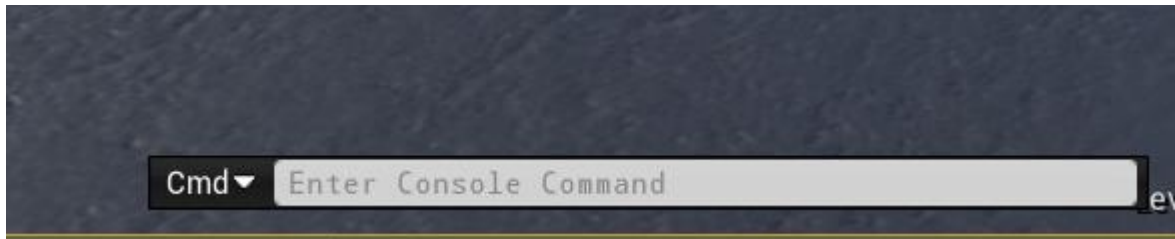
Class 27: (Apr 14)

I tried different techniques and approaches and came to realize that for the easiest way to capture a picture and send it to the Matlabs' toolbox is with the code as follows:


```
HighResShot 4845x4845 filename="G:/location/where/Matlab/toolbox/driving/drivingdata/sim3d_custom_map.jpg"
```

The HighResShot screenshot console command is a great cross-platform tool for taking very high resolution screenshots. It is capable of generating screenshots of any size that are based off of the screen resolution (or window resolution if the game is running in window mode). The high-resolution screenshot console command does this by rendering the game frame multiple times, one tile at a time at full resolution, and concatenating all tiles into a single image file.

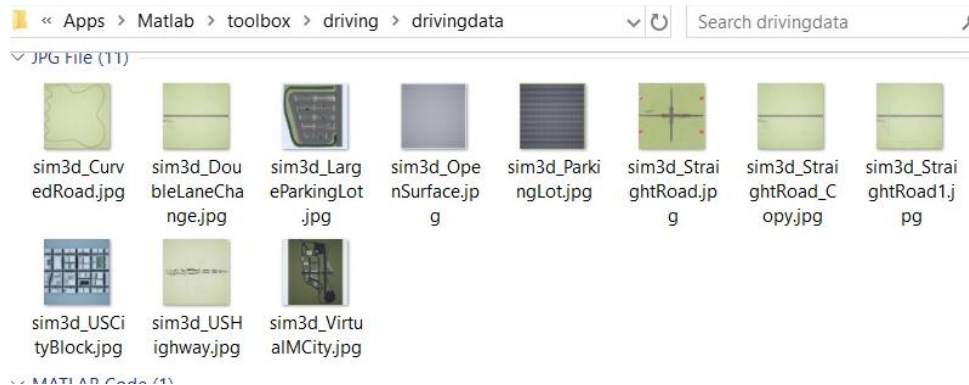
To access Unreal Engines console command, press '~', the tilde, which is to the left of 1. This will bring up the console as follows:



Once the console is shown, enter the command with the specified size (width x height) and path for convenience. Note: The Simulink program requires a 4845x4845 image, therefore, it is important to add this parameter.



Once the developer is ready to take the screenshot, simply click enter. The system will take about 10 seconds which then will proceed with updating the desired directory with a screenshot of the visible environment. As expected, the file is added to the drivingdata, which is the location the Matlab gets the image for the way points. So directly replacing this image with the screenshot will allow the Simulink to be fed the correct file.

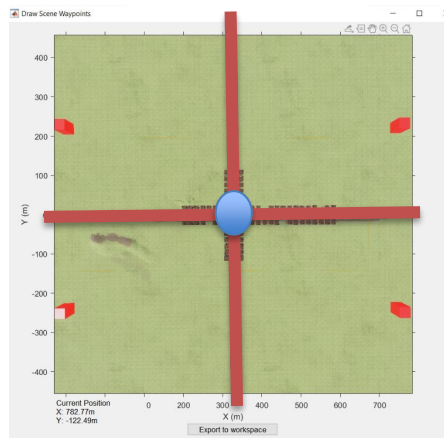


Class 28: (Apr 16)

Peer review 2 due

I did some subtle tweaking and was able to find some ways to improve the accuracy.

For starters, make sure the origin is in the right place. To determine if it is, you can observe the location through the Simulink GUI.

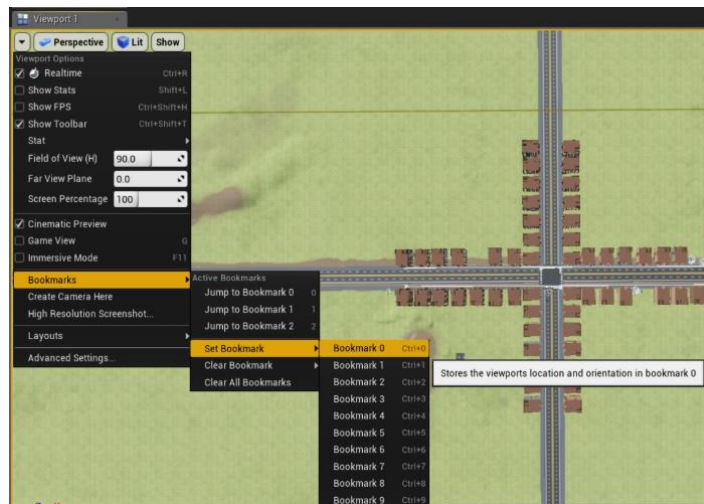


Determine Origin

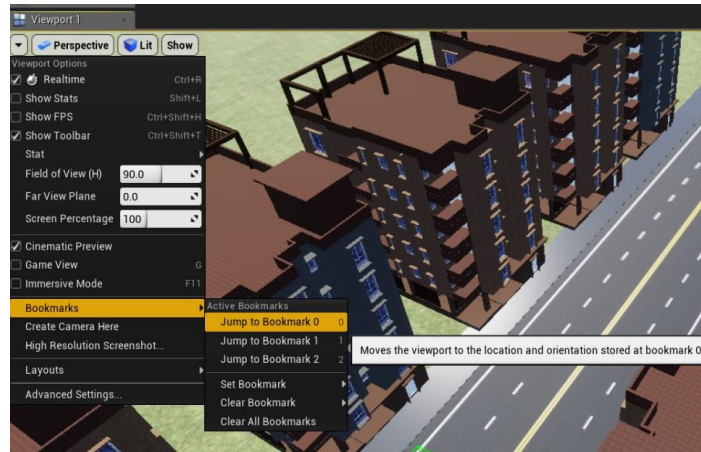
The origin seems to be around the (325, 0) so I added an object in this location in the Unreal engine and it seem to correspond as expected.

An additional feature I decided to add to this system is the capability of bookmarking the camera's location at a certain position (x, y, z) and angle. This then allows for a more precise scale of the environment in comparison with the Simulink position reference.

To set a new bookmark, simply go to a desirable height. Then, make sure your map is aligned as much as possible (I have not found a way to lock the camera in a 90 degree manner), then finally click the dropdown/Bookmarks/Set Bookmark/Bookmark x



Set Bookmarks



Jump to Bookmark x

Class 29: (Apr 21)

Ending the semester, these last couple of days, we also worked on creating demos for the final presentation. I helped adjust the speed of the clips to make sure they were all at similar speeds. Given we are running the system on different machines, some are bound to run slower. This allowed for a cleaner and smoother demo.

Class 30: (Apr 23)

Project Demo

Lastly, I also met up with Aaron to discuss ways to efficiently transfer environments from one machine to another. We first attempted the Export All method, yet it showed to be inefficient since not only would it export them in one folder, meaning it was no longer structures, but it would also take a really long time to load.

The second, and more reliable method, seems to be drag and dropping the environment folder to a shared google drive. When dropping this folder to a new project, it showed to

only add the different files, while also preserving the existing ones. This is a much better and effective method.

Class 31: (Apr 29)

Final presentation 2-5

Class 32: (May 5)

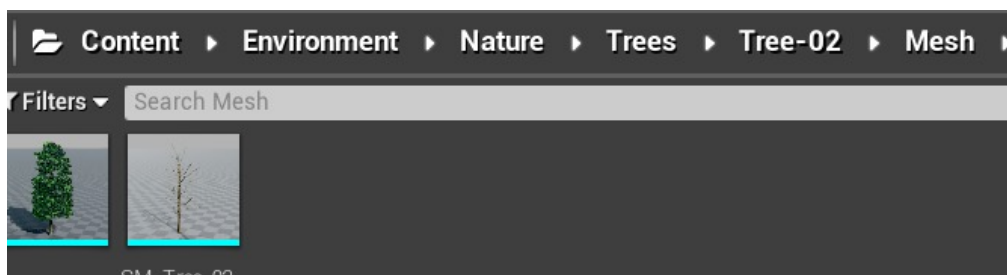
Peer review 3 due

Final Report

All deliverables

Closing Notes:

- There are many elegant and nice meshes already included so getting familiar with these would be good.



- Running Simulink without ROS showed to be a lot easier for the development process, so if a developer's computer is low on memory or RAM, only using Simulink could be promising.
- You can uninstall many UE packages that aren't necessary saving you over

40GBs

- You can draw road spline meshes using the Models/ Landscapes/ Edit spline
- I added an animation and apartments libraries to the drive, for countless of unique animations such as walking, running, sitting, standing, etc. and buildings.