

*Florida International University*  
*School of Computing and Information Sciences*

Software Engineering Focus

# Final Deliverable

Project Title: A Simulation Platform for Autonomous Research and  
System Evaluation (SPARSEv1.0)

**Team Members:** Nicolas Gonzalez, Emmanuel Perez

**Product Owner(s):** Miguel Alonso Jr

**Mentor(s):** Miguel Alonso Jr and Masoud Sadjadi

**Instructor:** Masoud Sadjadi

The MIT License (MIT)  
Copyright (c) 2016 *Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*This document presents the information necessary to gain a good understanding of A Simulation Platform for Autonomous Research and System Evaluation (SPARSEv1.0)*

## Table of Contents

|                                       |          |
|---------------------------------------|----------|
| <b>INTRODUCTION .....</b>             | <b>5</b> |
| CURRENT SYSTEM .....                  | 5        |
| PURPOSE OF NEW SYSTEM .....           | 5        |
| <b>USER STORIES</b>                   |          |
| IMPLEMENTED USER STORIES .....        | 7        |
| PENDING USER STORIES .....            | 10       |
| <b>PROJECT PLAN</b>                   |          |
| HARDWARE AND SOFTWARE RESOURCES ..... | 12       |
| SPRINTS PLAN .....                    | 13       |
| <i>Sprint 1</i> .....                 |          |
| .....                                 | 13       |
| <i>Sprint 2</i> .....                 |          |
| .....                                 | 13       |
| <i>Sprint 3</i> .....                 |          |
| .....                                 | 14       |
| <i>Sprint 4</i> .....                 |          |
| .....                                 | 15       |
| <i>Sprint 5</i> .....                 |          |
| .....                                 | 16       |
| <i>Sprint 6</i> .....                 |          |
| .....                                 | 17       |
| <i>Sprint 7</i> .....                 |          |
| .....                                 | 18       |
| <b>SYSTEM DESIGN</b>                  |          |

|                                                                                                                        |    |
|------------------------------------------------------------------------------------------------------------------------|----|
| ARCHITECTURAL PATTERNS                                                                                                 |    |
| .....                                                                                                                  | 20 |
| SYSTEM AND SUBSYSTEM DECOMPOSITION .....                                                                               | 21 |
| DEPLOYMENT DIAGRAM                                                                                                     |    |
| .....                                                                                                                  | 22 |
| DESIGN PATTERNS                                                                                                        |    |
| .....                                                                                                                  | 22 |
| <b>SYSTEM VALIDATION</b>                                                                                               |    |
| .....                                                                                                                  | 23 |
| <b>GLOSSARY</b>                                                                                                        |    |
| .....                                                                                                                  |    |
| ...                                                                                                                    | 37 |
| <b>APPENDIX</b>                                                                                                        |    |
| .....                                                                                                                  |    |
| ...                                                                                                                    | 38 |
| APPENDIX A - UML DIAGRAMS                                                                                              |    |
| .....                                                                                                                  | 38 |
| <i>Static UML Diagrams</i>                                                                                             |    |
| .....                                                                                                                  | 38 |
| <i>Dynamic UML Diagrams</i>                                                                                            |    |
| .....                                                                                                                  | 40 |
| APPENDIX B - USER INTERFACE DESIGN .....                                                                               | 52 |
| APPENDIX C - SPRINT REVIEW REPORTS .....                                                                               | 69 |
| APPENDIX D - USER MANUALS, INSTALLATION/MAINTENANCE DOCUMENT, SHORTCOMINGS/WISHLIST DOCUMENT AND OTHER DOCUMENTS ..... | 74 |
| <b>REFERENCES</b>                                                                                                      |    |
| .....                                                                                                                  |    |
| ..                                                                                                                     | 80 |

## INTRODUCTION

## **Current System**

There are other platforms that provide some of this functionality, such as OpenAI's gym, CARLA, or Microsoft's AirSim. However, these systems have a number of issues, such as a lack of realism, real-time simulation speed, custom sensor configurability, and readily usable APIs.

We will be starting from scratch, this is SPARSE version 1.0. We began with Unreal Engine's C++ Vehicle template. This gives us a simple car with the physics and logic to control it within Unreal.

## **Purpose of New System**

Design a realistic autonomous AI simulator that addresses the shortcomings of current systems.

Include:

A realistic environment and vehicle.

Real-time simulation speed.

Programmatic input from a python client.

## **USER STORIES**

The following section provides the detailed user stories that were implemented in this iteration of the SPARSE project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

### **Implemented User Stories**

- User should be able to make an API call through RPC so that they can control the simulation environment
- User should be able to send API calls to throttle the car
- User should be able to send API calls to brake the car
- Users should be able to send API calls to steer car left and right
- User should be able to send API calls to reverse the car
- Users should have 3 cameras to get data from their car
- User should be able to send API calls to toggle on and off the front cameras
- Users have a realistic model of a car to drive with, so they can simulate realistic scenarios
- User should be able to control the python client using a Controller to control the car and game hud
- User should be able to see the frame rate of the simulation in the hud, to monitor its performance
- User should be able to reset the platform to restart the pawn to its default position
- Users should have a realistic environment to drive the car on

### **Pending User Stories**

- Users have access to live stream of the visual data from the car
- Users have access to visual data which is being recorded to disk
- Realistic environment and events to test AI
- User's vehicle can be tested with randomly generated events
- Users can switch cameras in application to see different views
- Both Mac and Windows should be able to use this software

## PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

## Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

|                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                              |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <b>Software:</b> <ul style="list-style-type: none"><li>● Unreal engine 4.20.3</li><li>● Visual Studios 2017<ul style="list-style-type: none"><li>○ Workload: Game development with C++</li></ul></li><li>● Rpclib 2.2.1</li><li>● Python 3<ul style="list-style-type: none"><li>○ mprpc</li><li>○ Inputs</li><li>○ keyboard</li></ul></li></ul> | <b>Hardware:</b> <ul style="list-style-type: none"><li>● Gamepad that works on a PC</li><li>● Ideally a decent GPU</li></ul> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|



## Sprints Plan

### *Sprint 1*

Stories Pulled in:

- SPRS-20 - User should be able to make an API call through RPC so that they can control the simulation environment
- SPRS-2 - User should be able to send API calls to throttle the car
- SPRS-3 - Users should be able to send API calls to steer car left and right
- SPRS-19 - User should be able to send API calls to brake the car

### *Sprint 2*

Stories Pulled in:

- SPRS-20 - User should be able to make an API call through RPC so that they can control the simulation environment

### *Sprint 3*

Stories Pulled in:

- SPRS-20 - User should be able to make an API call through RPC so that they can control the simulation environment

### *Sprint 4*

Stories Pulled in:

- SPRS-20 - User should be able to make an API call through RPC so that they can control the simulation environment

### *Sprint 5*

Stories Pulled in:

- SPRS-2 - User should be able to send API calls to throttle the car
- SPRS-3 - Users should be able to send API calls to steer car left and right
- SPRS-19 - User should be able to send API calls to brake the car

### ***Sprint 6***

Stories Pulled in:

- SPRS-8 - Users should have 3 cameras to get data from their car
- SPRS-10 - Users have access to live stream of the visual data from the car
- SPRS-21 - Users have a realistic model of a car to drive with, so they can simulate realistic scenarios

### ***Sprint 7***

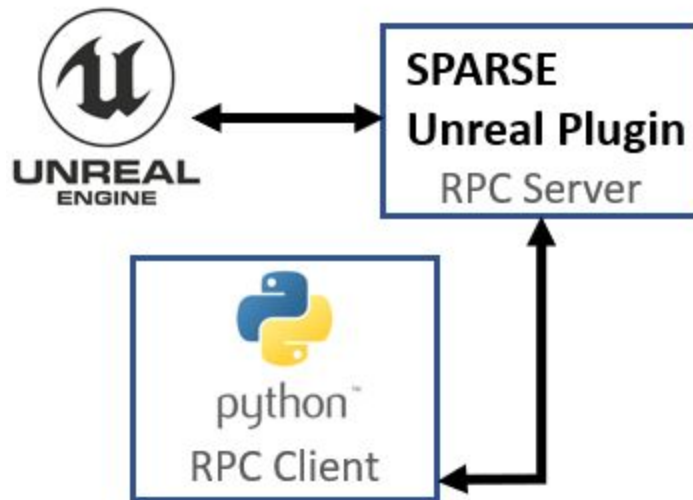
Stories Pulled in:

- SPRS-13 - Users should have a realistic environment to drive the car on
- SPRS-22 - User should be able to send API calls to reverse the car
- SPRS-23 - User should be able to reset the platform to restart the pawn to its default position
- SPRS-24 - User should be able to control the python client using a Controller to control the car and game hud
- SPRS-25 - User should be able to send API calls to toggle on and off the front cameras
- SPRS-26 - User should be able to see the frame rate of the simulation in the hud, to monitor its performance

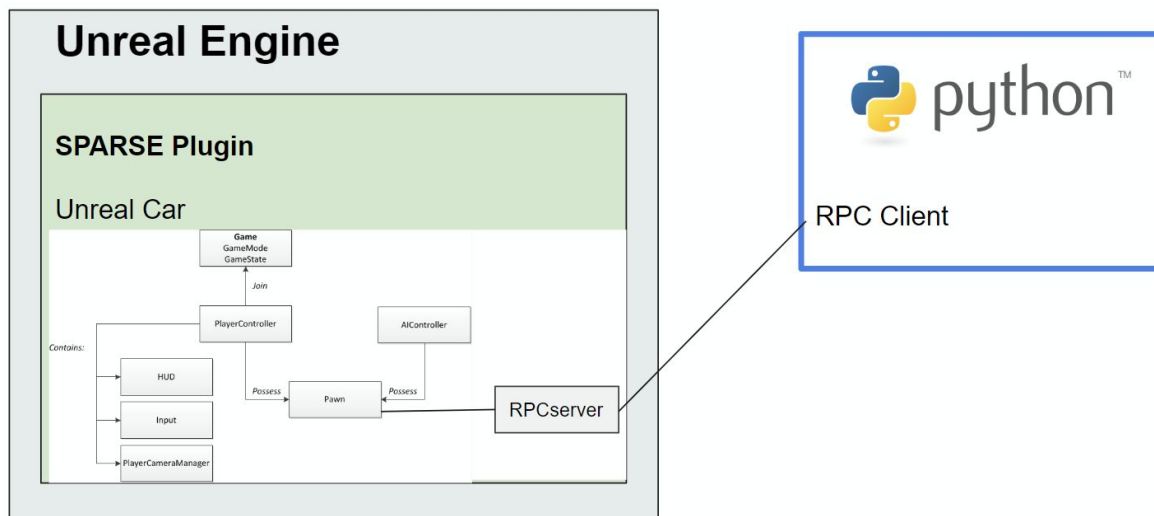
## **SYSTEM DESIGN**

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

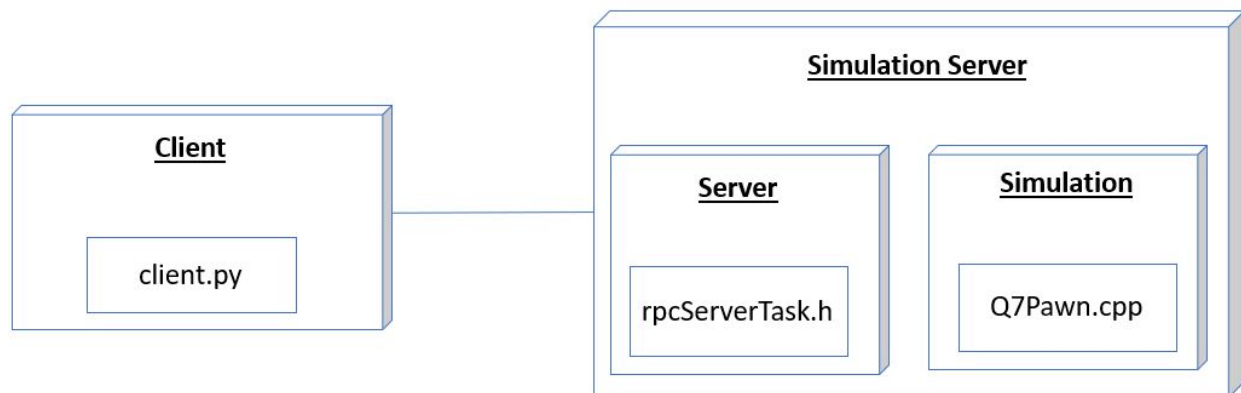
## Architectural Patterns



## System and Subsystem Decomposition



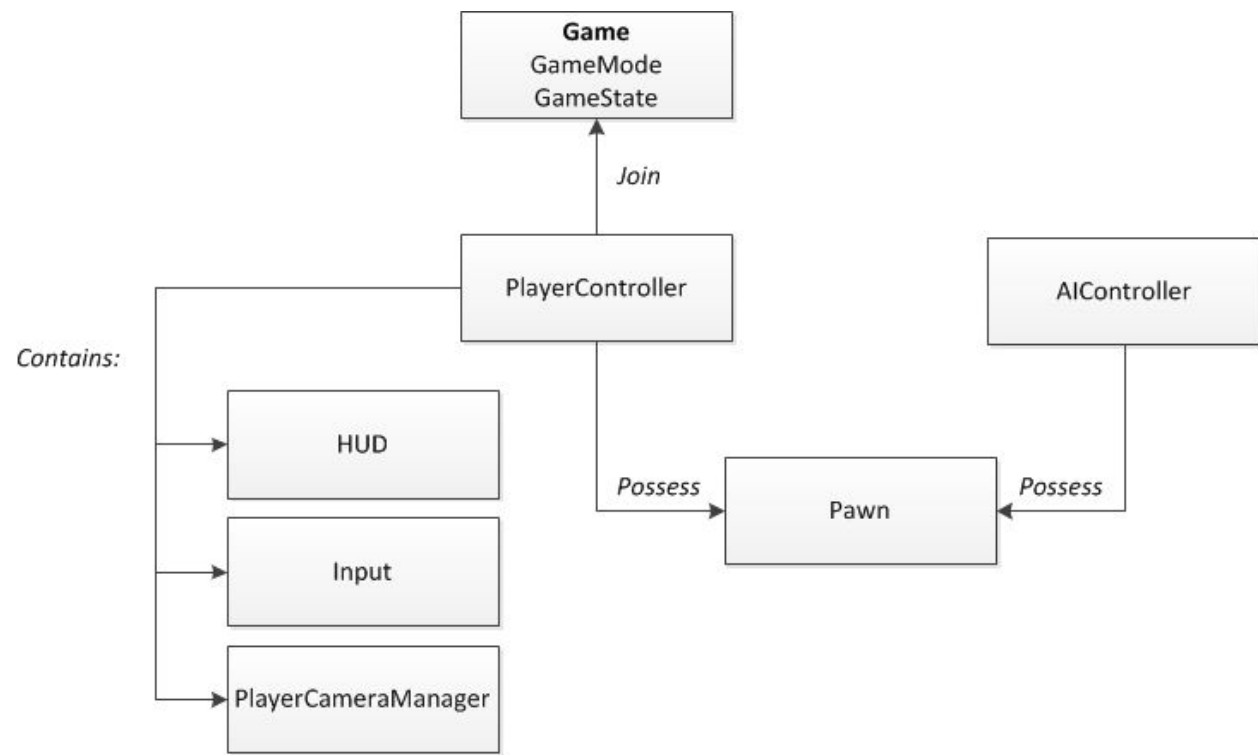
## Deployment Diagram

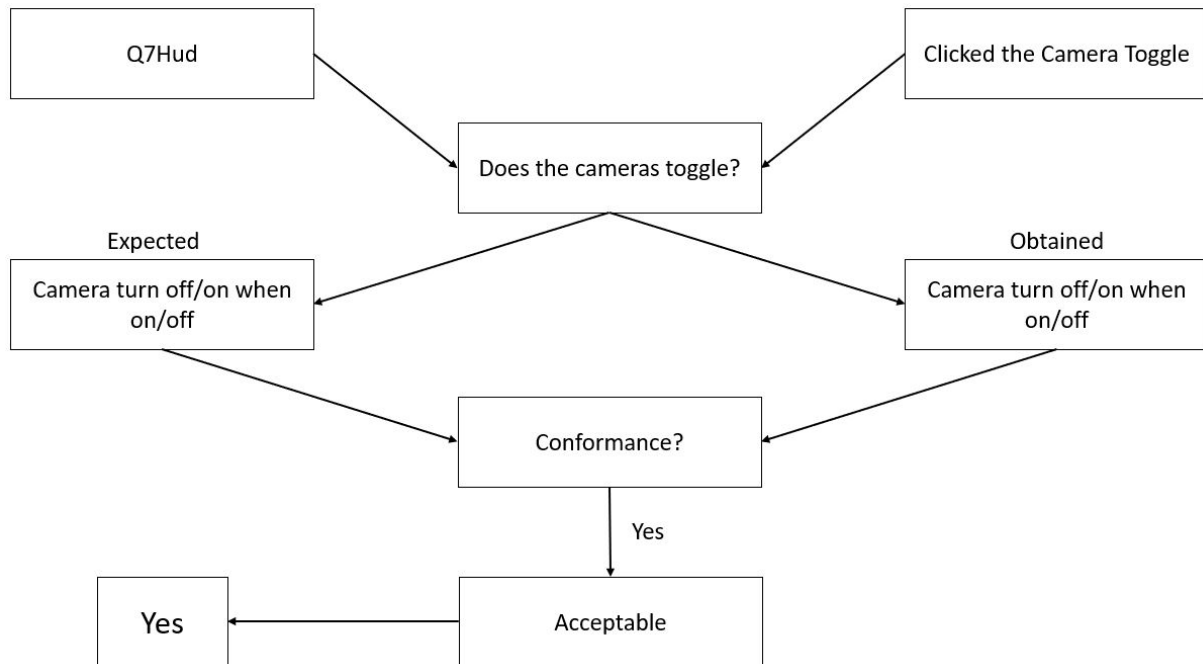


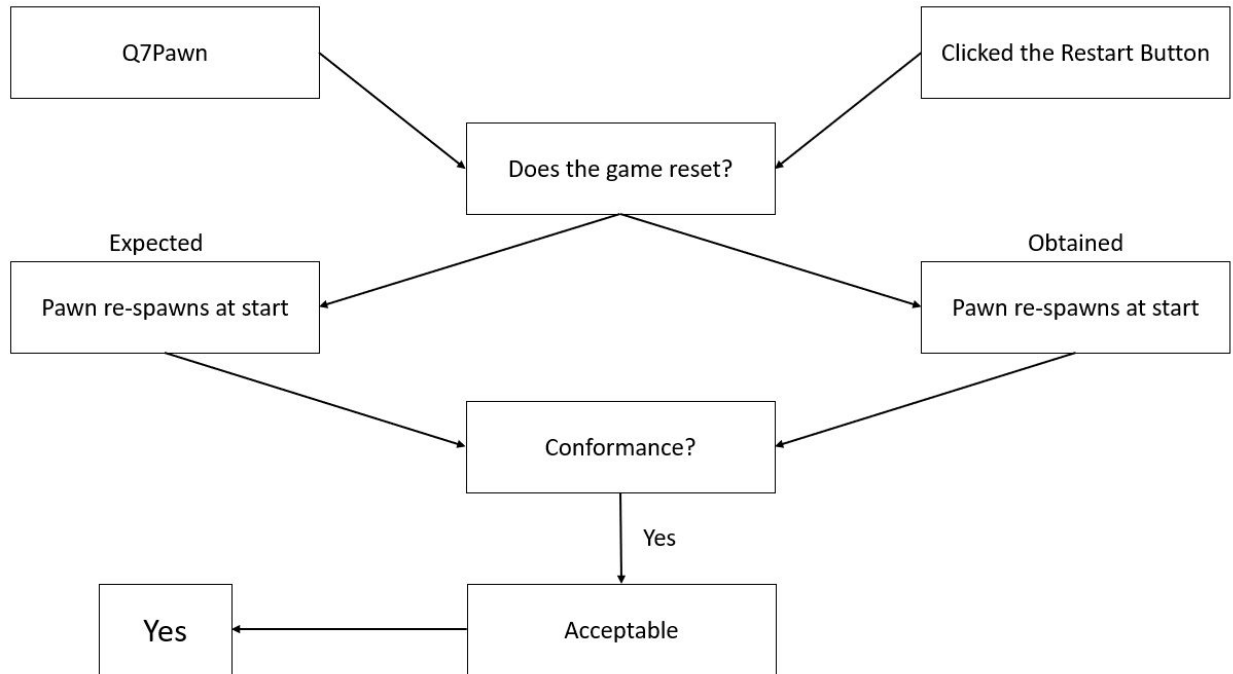
## Design Patterns

- While we did not create any design patterns ourselves we did use many of the existing Design Patterns in Unreal and extended them to fit our needs

- A [Pawn](#) is an Actor that can be an "agent" within the world. Pawns can be possessed by a Controller, they are set up to easily accept input, and they can do various and sundry other player-like things. Note that a Pawn is not assumed to be humanoid.
- A [PlayerController](#) is the interface between the Pawn and the human player controlling it. The PlayerController essentially represents the human player's will.
- A [HUD](#) is a "heads-up display", or the 2D on-screen display that is common in many games. Think health, ammo, gun reticle, etc. Each PlayerController typically has one of these.
- The PlayerCameraManager is the "eyeball" for a player and manages how it behaves. Each PlayerController typically has one of these as well.
- The concept of a "game" is split into 2 classes. The [Game Mode and Game State](#) is the definition of the game, including things like the game rules and win conditions. It only exists on the server. It typically should not have much data that changes during play, and it definitely should not have transient data that clients need to know about.
- The [GameState](#) contains the state of the game, which could include things like the list of connected players, the score, where the pieces are in a chess game, or the list of what missions you have completed in an open world game. The GameState exists on the server and all clients and can replicate freely to keep all machines up to date.



**SYSTEM VALIDATION**

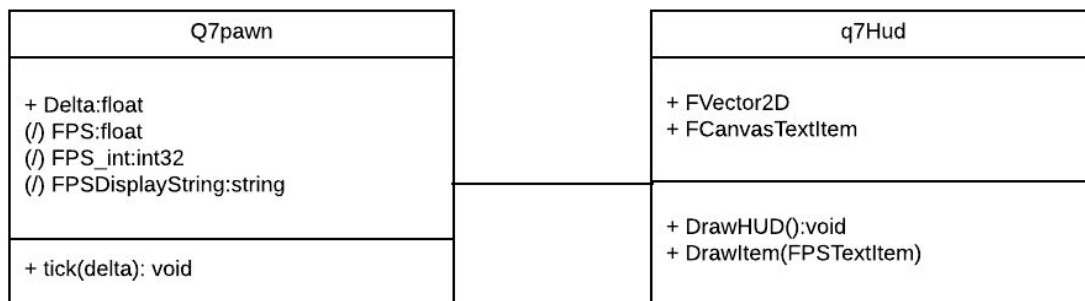
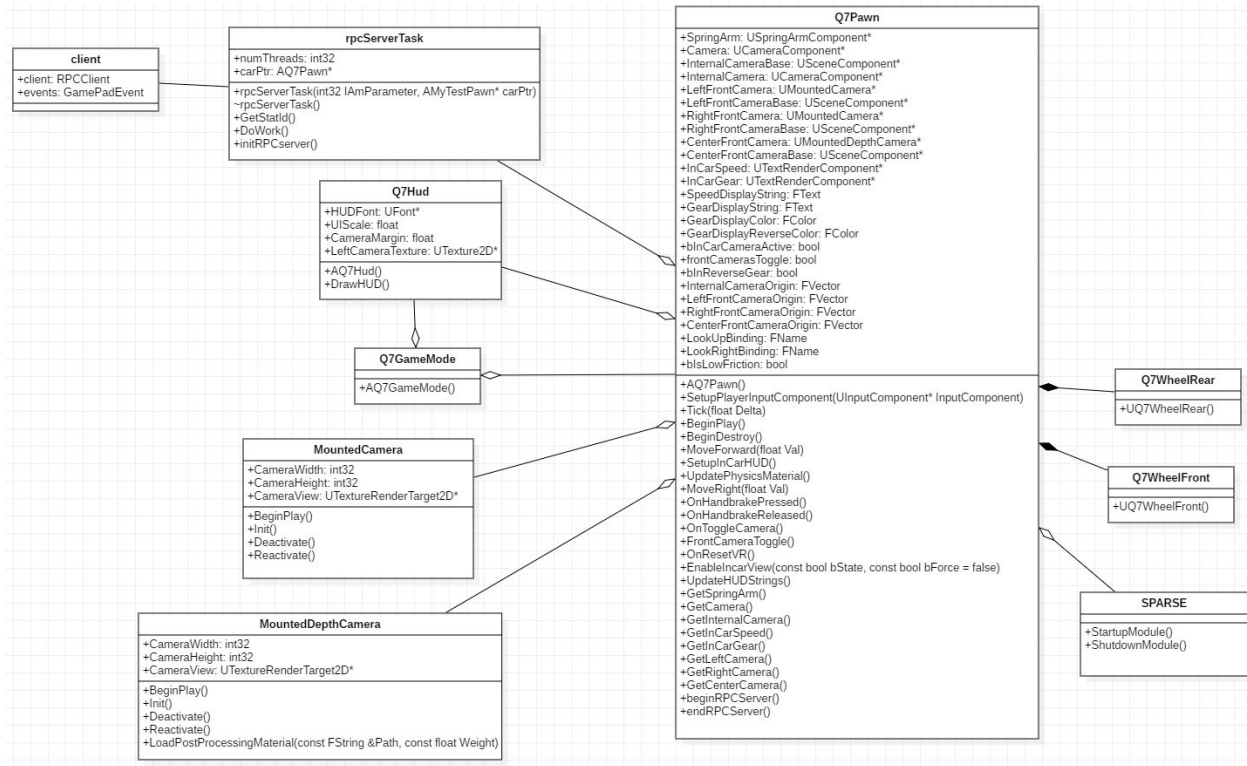


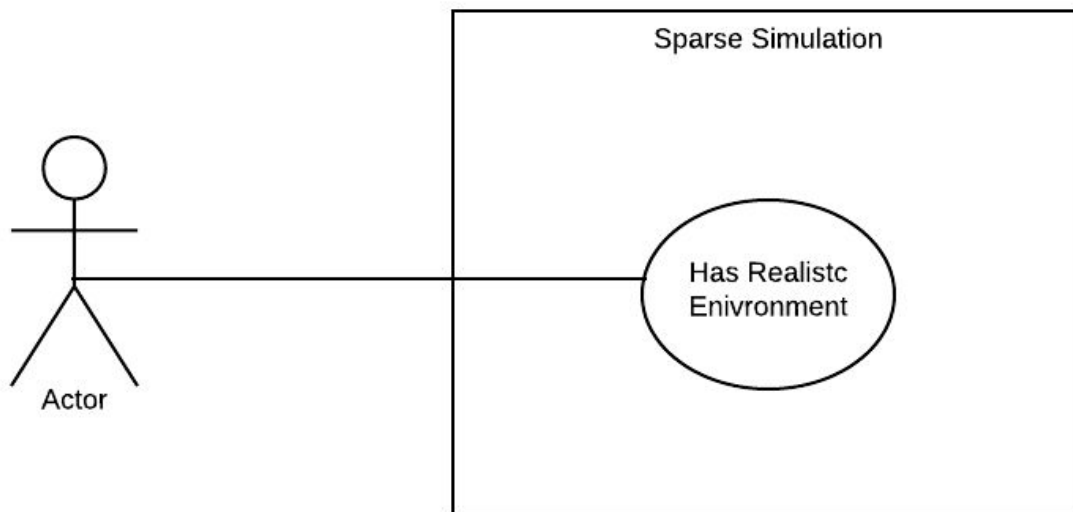
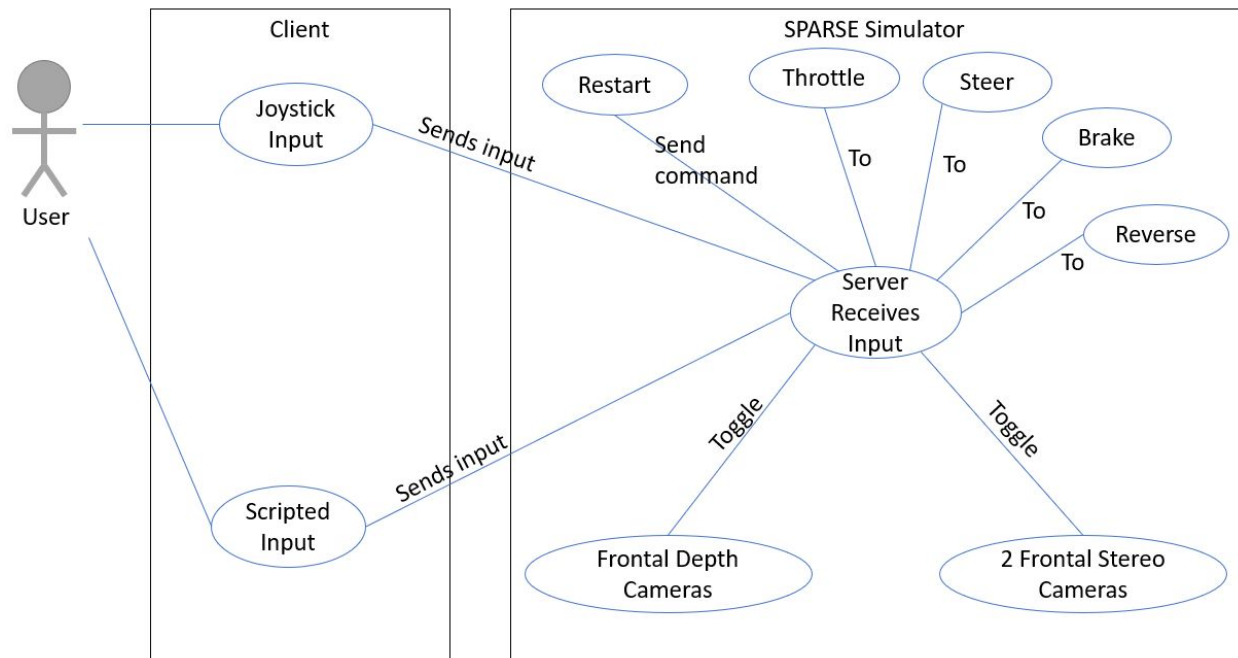


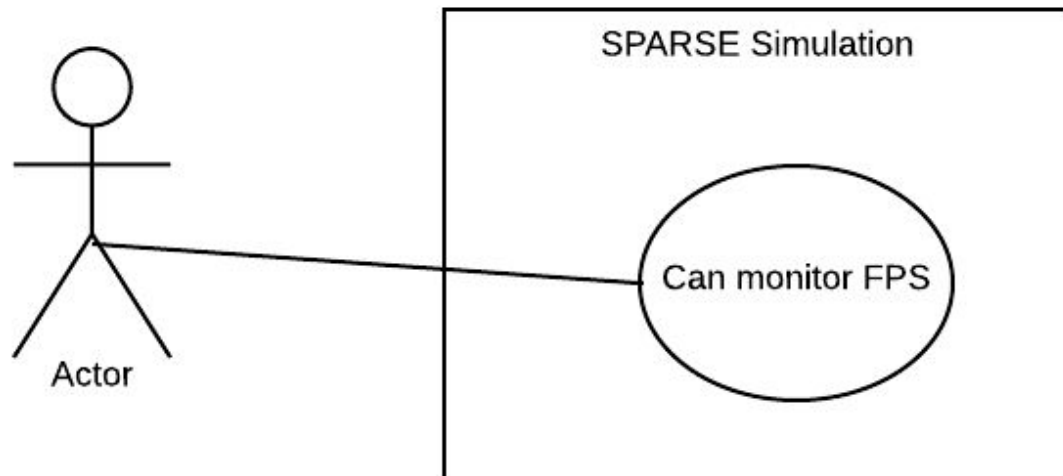
## **GLOSSARY**

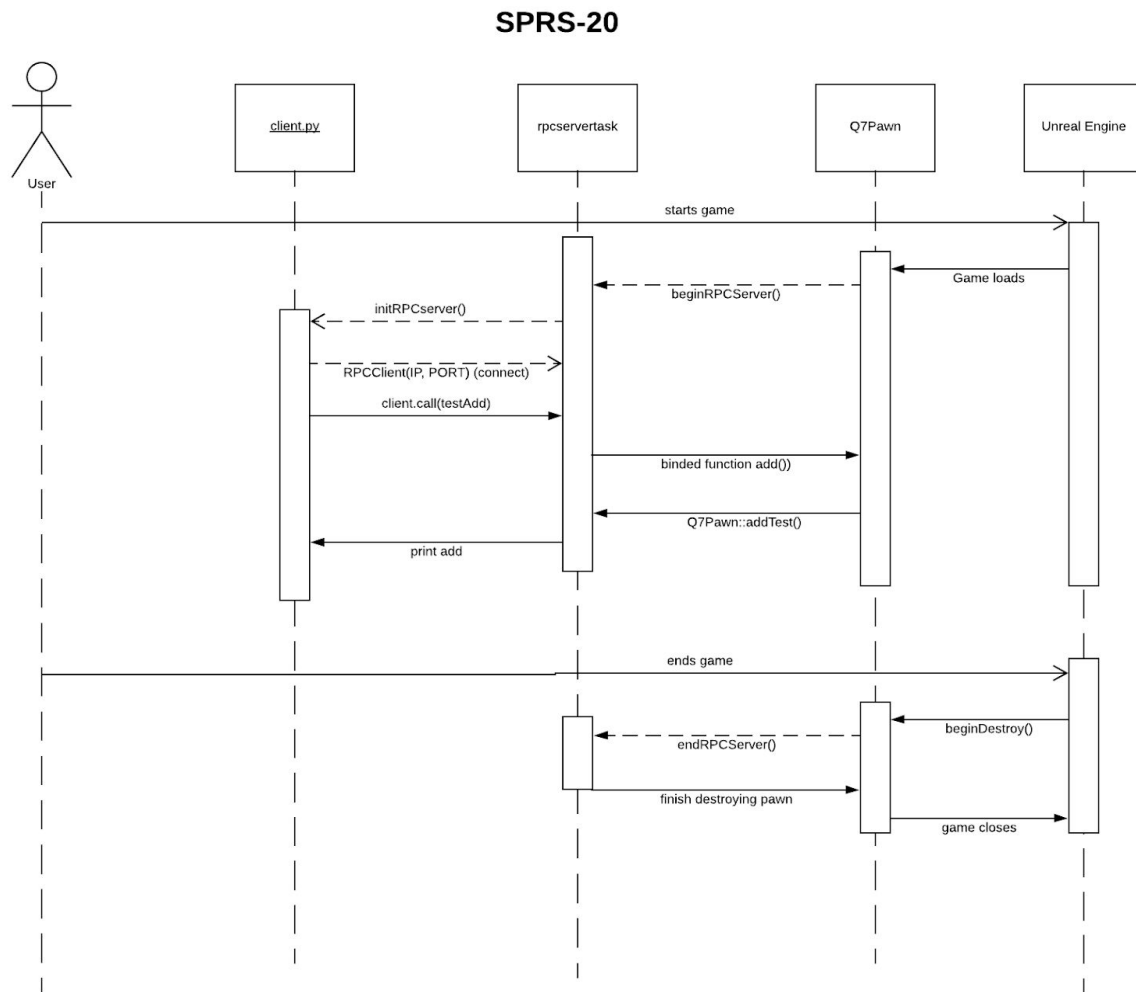
## APPENDIX

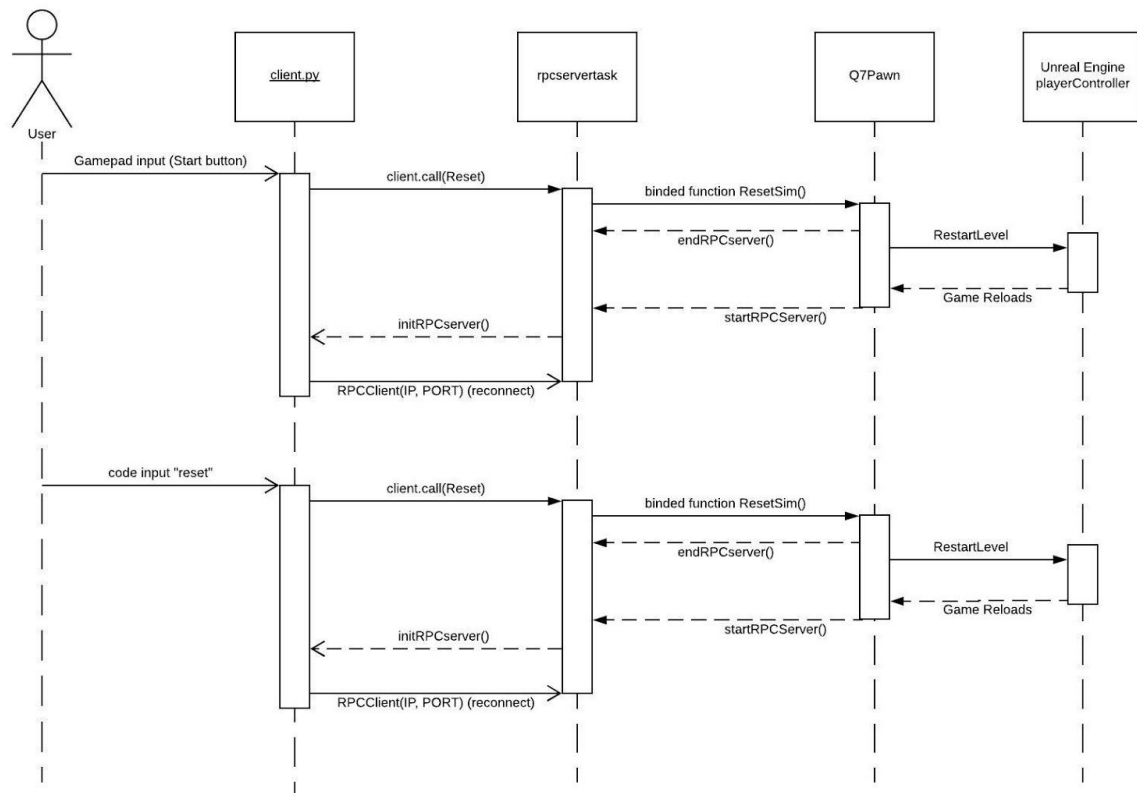
### Appendix A - UML Diagrams

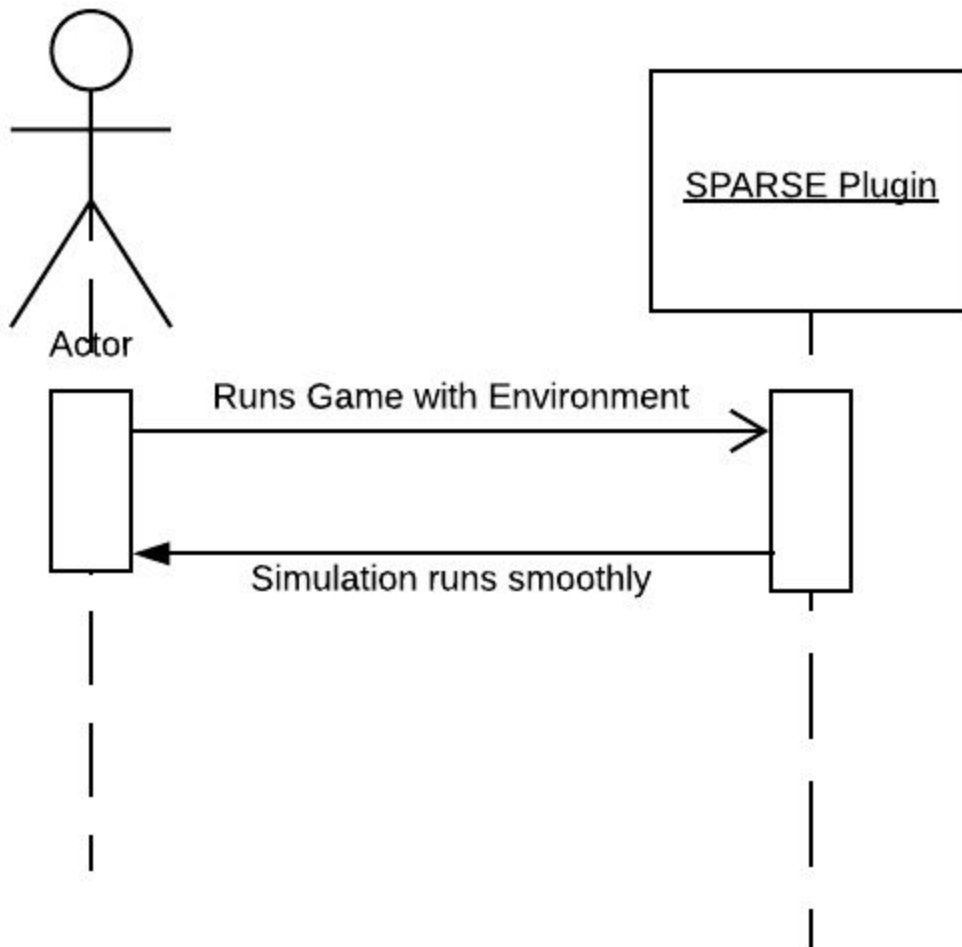


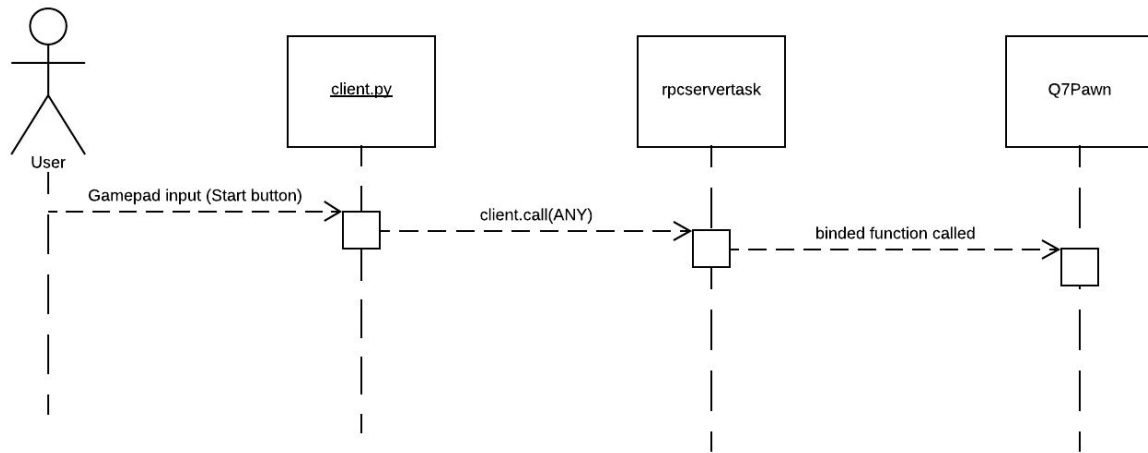






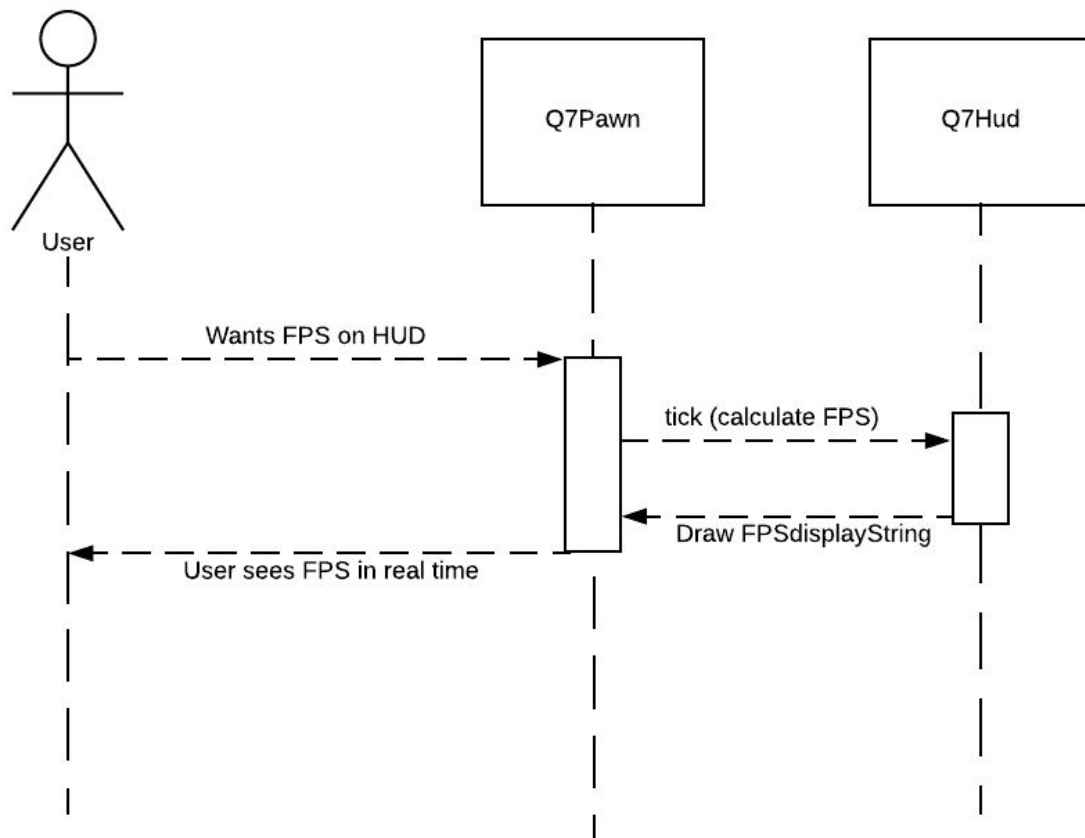
**Use Case: Reset**



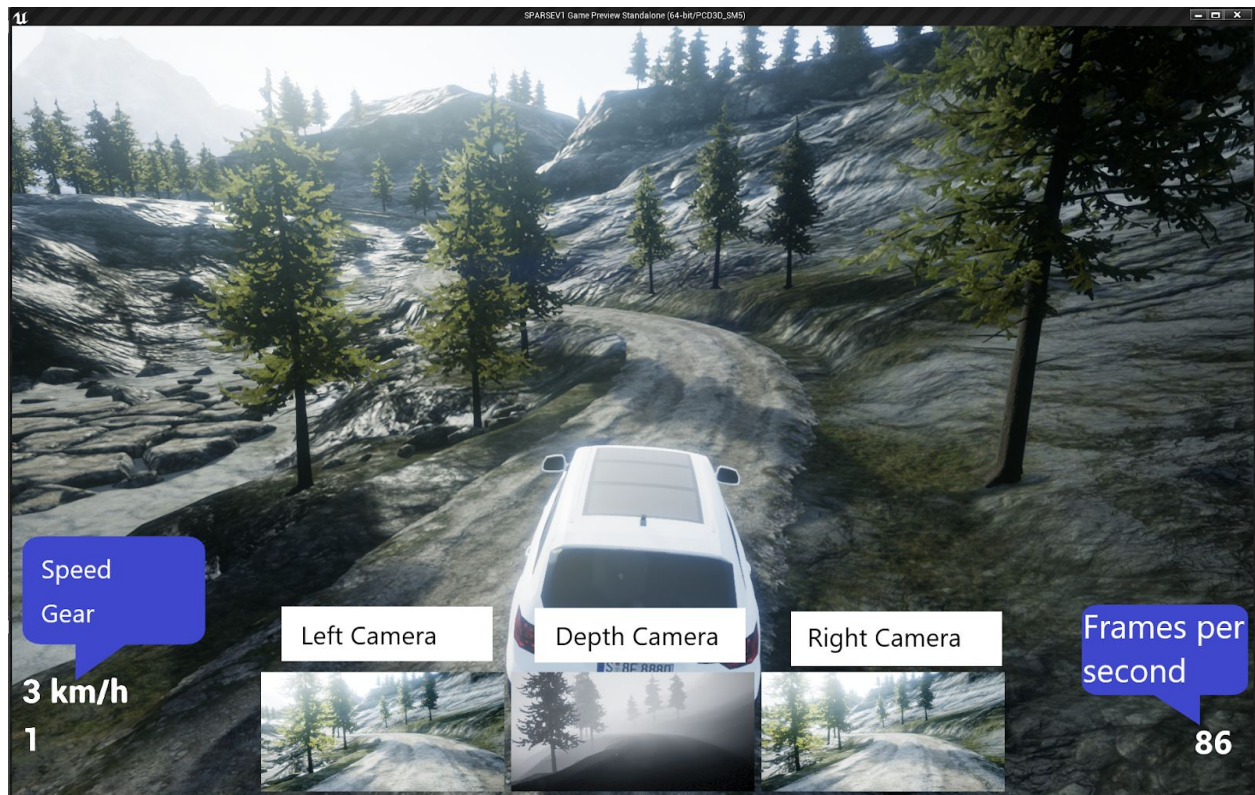


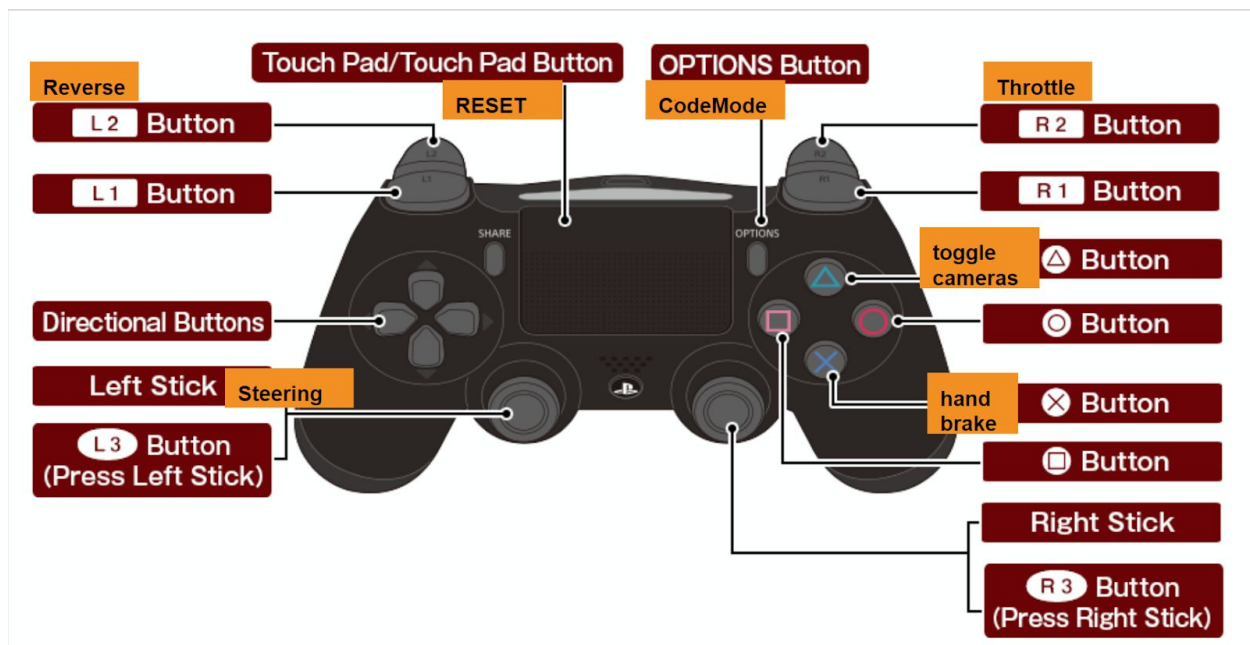


## Use Case: FPS



## Appendix B - User Interface Design





## Appendix C - Sprint Review Reports

### *Sprint 1*

#### Stories Completed:

- SPRS-20 - User should be able to make an API call through RPC so that they can control the simulation environment
- SPRS-2 - User should be able to send API calls to throttle the car
- SPRS-3 - Users should be able to send API calls to steer car left and right
- SPRS-19 - User should be able to send API calls to brake the car

#### Incomplete Stories:

- None

Key Outcomes: We have our Development environment running and project to work off of.  
Problems: None

### *Sprint 2*

Stories Completed:

- None

Incomplete Stories:

- SPRS-20 - User should be able to make an API call through RPC so that they can control the simulation environment

Key Outcomes: Downloaded rpc lib, we noticed we had been too ambitious and we are only going to focus on the platform setup with RPC

Problems: Having a lot of problems getting rpc lib or gRPC to compile

### *Sprint 3*

Stories Completed:

- None

Incomplete Completed:

- SPRS-20 - User should be able to make an API call through RPC so that they can control the simulation environment

Key Outcomes: Started focusing on using rpc lib

### *Sprint 4*

Stories Completed:

- SPRS-20 - User should be able to make an API call through RPC so that they can control the simulation environment

Incomplete Stories:

- None

Key Outcomes: Finally got the platform setup with an RPC server that runs on its own thread

Problems: None, hopefully it stays that way

### ***Sprint 5***

Stories Completed:

- SPRS-2 - User should be able to send API calls to throttle the car
- SPRS-3 - Users should be able to send API calls to steer car left and right
- SPRS-19 - User should be able to send API calls to brake the car

Incomplete Stories:

- None

Key Outcomes: Car can be throttle, brake and steer using RPC

Problems: None

### ***Sprint 6***

Stories Completed:

- SPRS-8 - Users should have 3 cameras to get data from their car
- SPRS-21 - Users have a realistic model of a car to drive with, so they can simulate realistic scenarios

Incomplete Stories:

- SPRS-10 - Users have access to live stream of the visual data from the car

Key Outcomes: Have a more realistic car model and have 3 front cameras to view. We decided to give up on Live streaming for the last sprints

Problems: Streaming images was more complicated than anticipated

### ***Sprint 7***

Stories Completed:

- SPRS-13 - Users should have a realistic environment to drive the car on
- SPRS-22 - User should be able to send API calls to reverse the car
- SPRS-23 - User should be able to reset the platform to restart the pawn to its default position

- SPRS-24 - User should be able to control the python client using a Controller to control the car and game hud
- SPRS-25 - User should be able to send API calls to toggle on and off the front cameras
- SPRS-26 - User should be able to see the frame rate of the simulation in the hud, to monitor its performance

Incomplete Stories:

None

Key Outcomes: Have a good working demo to demonstrate the project

Problems: None

## **Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents**

### **Installation/Maintenance:**

- First Install Visual Studio 2017 x64 and the Workload: “Game development with C++”
- Now install Unreal Engine 4.20
- Install python 3
- Use “pip3 install” to install mprpc, inputs and keyboard.
- Clone from Bitbucket for current master:  
<https://sp-jira.cis.fiu.edu:7443/scm/sprs/sparse-v1.git>
- Get and Build rpclib for you OS in 64bit. Get the headers and lib file and place it in  
\\sparse-v1\\Plugins\\SPARSE\\Source\\ThirdParty\\rpclib
- In the root of the project right click the .uproject file and click “Generate Visual Studio project files”
- Now open the SPARSEV1.uproject, if everything is done correctly the project will rebuild and open the Unreal Editor

### **Shortcomings:**

- Unreal can be a little confusing and its documentation is not the best.

- Compiling Unreal and C++ is not as simple as other languages and the library are system specific so users will have to build their own libraries
- There is a known bug in rpclib where `srv.run();` does not block. used `srv.async_run();` instead
- The Q7 asset could use some higher quality materials and optimization

**Wishlist:**

- Extract the Data from our camera's and stream it to an RTSP server
- A C++ library for windows that we can actually get to build.
- More maps, maybe a city map.

**REFERENCES**

<https://docs.unrealengine.com/en-us/>  
<https://docs.unrealengine.com/en-us/Gameplay/Framework>  
<http://api.unrealengine.com/INT/API/index.html>  
<http://carla.org/>  
<https://github.com/carla-simulator/carla>  
<https://github.com/Microsoft/AirSim>  
<https://mprpc.readthedocs.io/en/latest/>  
<https://inputs.readthedocs.io/en/latest/>  
<https://github.com/zeth/inputs/blob/master/docs/index.rst>  
<http://rpclib.net/>  
<https://github.com/rpclib/rpclib>