GENIVI Vehicle Simulator

GENIVI VEHICLE SIMULATOR (GVS) PROJECT

Overview

The project and the initial software code have been developed by Elements Design Group of San Francisco and the Jaguar Land Rover Open Software Technology Center in Portland, Oregon. The motivation was to provide an open source, extensible driving simulator for the development community. While there are multiple potential uses, the primary goal was to create an application to assist in the development and testing of IVI systems.

How was the simulator built?

The project was created with Unity 5.3.4 and runs on Windows 10 64 bit. To obtain a copy of Unity, please go to [www.unity3d.com](http://www.unity3d.com). Note, there are multiple versions of Unity. Please familiarize yourself with their licensing requirements and download the version of unity which best applies to your organization. While Unity and Windows have proprietary licenses, all of the software code in the GVS project itself is open source licensed.

Who are the maintainers of the GVS project?

Daniel Schambach (daniel@elementsgroup.com)  
James Bennet (jbennet9@jaguarlandrover.com)

System Requirements

- CPU – 5th Generation Core i7 GPU
- GPU - Single GTX 980 Ti, GTX Titan, or Quadro K6000
- RAM – 12 GB
- Motherboard - ASUS X99 E WS

Getting Started

Developers

Clone the repository here.

End Users

End Users may download binaries of the application here ([http://bit.ly/GeniviVS](http://bit.ly/GeniviVS)). Several builds have been created for various hardware configurations (running on Windows 10). They are:

- One 1920 x 1080 Monitors
- Two 1920 x 1080 Monitors (total resolution 3840x1080)
- Three 1920 x 1080 Monitors (total resolution 5760x1080)

APPLICATION FEATURES

DRIVING SCENES

There are three different driving scenes in the simulator, (Yosemite, Pacific Coast Highway and San Francisco).

CUSTOM VEHICLE CONTROLLER

A custom vehicle controller has been created to simulate vehicle handling. A detailed GUI interface, exposes many of the vehicle physics parameters, allowing vehicle calibration at runtime. Presets can be saved, loaded, etc. For more information please see Vehicle Controller.

CSV DATA STREAM

A CSV data stream (over IP) is running and drives the instrument cluster application. It outputs in a format similar to that to a Controller Area Network (CAN bus) along with unity specific information (vehicle transform, current speed, infractions, etc.) during the driving session. This allows for comparison/ review of driving and IVI UI data in realtime.

INTEGRATION WITH ERGONEERS D-LAB SOFTWARE

The CSV stream has been integrated into D-Lab allowing the Simulated Vehicle’s CAN bus data to be leveraged along with eye tracking software and IVI prototypes - providing for a wealth of IVI testing and UI prototype review/validation opportunities.

OBSTACLES

Obstacles may be triggered (by the admin) while driving. If the driver hits an obstacle, the event is logged as an infraction which can be reviewed after the driving session. Current obstacle types are pedestrians, animals, rockfall, stalled vehicles, etc.

INFRACTION LOGGING

The following infractions are currently logged by the system (saved as xml, also reviewable in game).

- Running stop signs
- Running red lights
- Vehicle driving over double yellow lines on a single lane highway
- Collisions with terrain, obstacles (triggered by an admin), traffic vehicles, etc.
INFRACTION REVIEW
At the end of a driving session, the admin and driver can review infractions from the most recent session. Screenshots of the infraction along with vehicle data (speed, etc.) are displayed. Session data is saved timestamped and saved out as XML or CSV [to compare with In-Vehicle Infotainment (IVI) UI events.]

AI TRAFFIC SYSTEM
A “non-playing character” (NPC) vehicle traffic system simulates traffic in an urban environment. The road network (a small version of San Francisco) contains complex intersections often found in urban settings (one-way streets intersecting with two way streets, etc.) The NPC vehicles correctly follow the flow of traffic, stop lights, stop signs, etc.

AUTO DRIVE
In the Yosemite Driving Scene, the system can run in “auto drive” mode. To enable, press “A” on the keyboard.

WAYPOINT MAPPING SYSTEM
In the San Francisco Driving Scene, a series of predefined routes (a waypoint system with arrows highlighting a specific route to follow) can be enabled, allowing for reproducible driving paths, etc.

Application flow diagram

Getting started developers
Clone the repository
Load the project in Unity (5.3.4f1)
Go to Assets/Scenes/ and open “Loader”
Press Play

Input Methods
Steering Wheel - tested with Logitech G27, SimSteering and Thrustmaster.
Keyboard - to toggle between steering wheel and keyboard press “z”.
To open the Admin Screen, press F1

Making a Build (In Editor)
A series of editor scripts exist to facilitate building the application. To use, see the Build Menu.
Two applications can be built from build menus.
1 Main Application (Driving Scene)
2 Console (Instrument Cluster)
Please read Scripts/Controllers/NetworkController.cs and adjust code with your individual IP settings. The console can be run locally or hosted.

Making a Build (with PowerShell scripts)
A series of .ps scripts exist which greatly facilitate the build, deploy run process in a development or production environments.

HARDWARE
A/V hardware for the simulator can be configured in many different ways. Here is a sample hardware diagram:
COMPUTERS

The experience can be run on one or two computers depending upon hardware/display configurations. One application displays the driving screen, while another displays the instrument cluster. The application was developed and tested on the following system. Other configurations will likely work, however were not tested.

- CPU – 5th Generation Core i7 GPU
- GPU - Single GTX 980 Ti, GTX Titan, or Quadro K6000
- RAM – 12 GB
- Motherboard - ASUS X99 E WS

AUDIO

The application can be configured to work with many different audio systems. Sound can be stereo, 5.1, 7.1, etc. Various alternate audio configurations have been tested and implemented.

TABLET INTERFACE

The application can be managed/controlled with a tablet interface which is hosted by the main (driving) application. It allows the admin to select vehicles, select scenes, trigger obstacles, change weather and time of day, modify road conditions, etc. To connect to the application via a tablet, open a browser and enter the IP address (with port 8088) of the computer running the main driving scene (e.g. http://192.168.1.113:8088).

MAIN DISPLAYS

The main driving scene is currently output to three Christie Mirage DS+14K-M and edge blended across an 11’ x 28’ parabolic screen (front projection). Current rendered dimensions are 4992 x 1080. The application runs in Stereo 3D (either via Nvidia or Win8 native stereo rendering). We have run it on three 4k displays as well and experienced decent frame rates (although not in 3d). Various alternate screen configurations are possible.

INSTRUMENT CLUSTER

A custom 1920x720 monitor is used to display instrument cluster graphics. The IC displays accurate MPH & RPM for the selected vehicle. Skins have been created for 6 vehicles. All gauges are controlled via a CSV stream over TCP/IP (same stream which is used for connecting to D-LAB software).

INPUT DEVICES (STEERING)

Leo Bodnar SimSteering systems (with FFB)
Logitech G27 (with FFB)
Thrustmaster T300rs
Keyboard input may also be used

Repository notes

The repository uses LFS. On windows it can be helpful to use Windows Git Credential Manager https://github.com/Microsoft/Git-Credential-Manager-for-Windows/releases

Vehicle components

To preview components associated with vehicles, go to Assets/Prefabs/XE_Rigged.

COMPONENTS

VehicleController.cs
VehicleInputController.cs
CarConsoleController.cs
ForceFeedback.cs
VehicleAudioController.cs