CS30 Simulation and Tools:

Ryan Cryar
Enrico Sundjojo
Arin Reinsch
Agenda

- Team overview
- Simulation team structure
- Devops
- Simulation
- Rendering
- Testing
Team Mission

Provide a full simulation and test environment for the autonomous race car to provide necessary testing of the car for competitions
### Team Wins

<table>
<thead>
<tr>
<th>Category</th>
<th>Wins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>16x</td>
</tr>
<tr>
<td>Endurance</td>
<td>16x</td>
</tr>
<tr>
<td>Efficiency</td>
<td>3x</td>
</tr>
<tr>
<td>Autocross</td>
<td>17x</td>
</tr>
<tr>
<td>Skid Pad</td>
<td>8x</td>
</tr>
<tr>
<td>Acceleration</td>
<td>1x</td>
</tr>
<tr>
<td>Design</td>
<td>10x</td>
</tr>
<tr>
<td>Cost</td>
<td>1x</td>
</tr>
<tr>
<td>Business Plan</td>
<td>8x</td>
</tr>
</tbody>
</table>
Team timeline

- Competition is release
- Everything is based around design, testing testing, and more design
- More software still needs writing even after code freeze

<table>
<thead>
<tr>
<th>Autonomous System</th>
<th>Oct-19</th>
<th>Nov-22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background Research Phase</td>
<td>Nov 23</td>
<td>Jan 4</td>
</tr>
<tr>
<td>Preliminary Research Phase</td>
<td>Jan 4</td>
<td>Feb 28</td>
</tr>
<tr>
<td>Preliminary Design Phase</td>
<td>Feb 14</td>
<td>Feb 28</td>
</tr>
<tr>
<td>Preliminary Design Review</td>
<td>Feb 28</td>
<td>Apr 17</td>
</tr>
<tr>
<td>Critical Design Phase</td>
<td>Apr 10</td>
<td>Apr 17</td>
</tr>
<tr>
<td>Critical Design Review</td>
<td>Apr 18</td>
<td>Jun 20</td>
</tr>
<tr>
<td>Drive Readiness Review</td>
<td>Jun 20</td>
<td>Aug 3</td>
</tr>
<tr>
<td>(DV) Competition Readiness Review (FSE)</td>
<td>Aug 31</td>
<td>Sep 5</td>
</tr>
<tr>
<td>(DV) Post Competition Assessment Review (FSS)</td>
<td>Sep 6</td>
<td>Oct 11</td>
</tr>
<tr>
<td>21D Pre-Development</td>
<td>Sep 6</td>
<td>Oct 11</td>
</tr>
</tbody>
</table>
Sim team structure

Overview

- Arin - Devops and Rendering
- Ryan - Sim Core
- Enrico - Testing
Overall team structure

GFR

Driver → Driverless

Mechanical

Electrical

Aerodynamics
Epowertrain
Suspension
Chassis

Autonomous System

SLAM
Sim
Trajectory
State Estimation
Machine learning

Us :)

Global Formula Racing
DHBW Ravensburg / Oregon State University
Devops

Goals

- Maintain stable master branch
- Validate new code
- Manage test data
- Track progress
- Streamline deployment
Devops

Tools

- Docker - virtualize build environments
- Jenkins - manage builds
- Bitbucket - version control
- Jira - manage agile projects
Devops

Dockerfiles - builds from user trigger

```bash
FROM ubuntu:latest
LABEL MAINTAINER "Armin Reinsch" <Armin.Reinsch@global-formula-racing.dev>
ENV ROS_DISTRO melodic

# Update container
RUN apt-get update -y && apt-get install --no-install-recommends -y apt-utils gnupg wget ca-certificates lib-release
RUN echo "debconf debconf/frontend select NonInteractive" | debconf-set-selections

# Install supporting programs
RUN apt-get -y install gnupg;
RUN apt-get install -y git
RUN apt-get install build-essential clang-3.9 git make ninja-build python3-requests python-dev tzdata sed curl wget unzip autoconf libtool

# Configure ROS environment
RUN sh -c "echo "http://packages.ros.org/ros/ubuntu $(lib_release -c) main > /etc/apt/sources.list.d/ros-latest.list"
RUN apt-key adv --keyserver "hkp://keyserver.ubuntu.com:80" --recv-key C1CF6E31E0BA553789D34FC35626F03A84E1B700
RUN apt-get update
RUN apt-get install -y ros-melodic-desktop

# Install Home
RUN update-alternatives --install /usr/bin/clang++ clang++ /usr/lib/llvm-3.9/bin/clang++ 100;
RUN update-alternatives --install /usr/bin/cmake clang /usr/lib/llvm-3.9/bin/cmake 100
RUN git clone --depth=1 -b 6.18 https://github.com/epimain/UnrealEngine_glt -t UnrealEngine_6.18; 
RUN cd ~/UnrealEngine_glt && Setup.sh && ./GenerateProjectFiles.sh.sh & & make; 
RUN cd -
RUN mkdir Carla
RUN git clone https://github.com/carla-simulator/carla; 
RUN ./Setup.sh

# Preparation for Jenkins user
RUN echo "$Jenkins Ali-NOSSNAG: A:*" >> /etc/sudoers 
RUN mkdir /home/jenkins 
RUN useradd -u 1042 -d /home/jenkins jenkins 
RUN chown -R jenkins:jenkins /home/jenkins 
RUN chmod 755 /home/jenkins 
RUN usermod -a -G jenkins 
RUN echo "*/bin/bash" 
```
Devops

Dockerfiles - automated builds from bitbucket

```bash
FROM ubuntu:latest
LABEL MAINTAINER "Arin Reinsch" <Arin.Reinsch@global-formula-racing.de>
ENV ROS_DISTRO melodic

# Update container
RUN apt-get update -qq && apt-get -qq install --no-install-recommends -y apt-utils gnupg wget ca-certificates lsb-release
RUN echo 'debconf debconf/frontend select Noninteractive' | debconf-set-selections

# Install supporting programs
RUN apt-get -y install gnupg1: \ 
apt-get install -y git

# Configure ROS environment
RUN sh -c "echo \"deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main\" > /etc/apt/sources.list.d/ros-latest.list\"
RUN apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-key C1CF6E31E6BA3BBF289D5270CF36A30763A1BB9\nRUN apt update
RUN apt install -y ros-melodic-desktop

# Permission for Jenkins user.
RUN echo \""jenkins ALL=NOPASSWD: A:+\" >> /etc/sudoers \n    mkdir /home/jenkins \n    useradd -u 1001 -d /home/jenkins jenkins \n    chown -R jenkins:jenkins /home/jenkins \n    chmod 755 /home/jenkins \n    usermod -a -G jenkins

# Configure shell
CMD ["/bin/bash"]
```
Devops

Jenkins configuration

- Github hook with SCM polling
- SSH agent
- Docker agent
- Bitbucket plugin
- Build token trigger plugin
Devops

- Pull branch into a docker container
- Run inline bash script to build workspace
- Publish results to bitbucket
Devops

Jenkins builds - remote trigger script

```bash
#!/bin/bash
GREP_RETURN_CODE=0
BUILD_URL="http://gfr_admin:1142a49026d811b57cf8c4b68244f0e6f@localhost:8080/"
#API token needs to be generated from jenkins (can only be seen once)
APITOKEN="1142a49026d811b57cf8c4b68244f0e6f"
#use wget to get servers crumb
CRUMB=$(wget -q --auth-no-challenge --user= -O /dev/null -o /dev/null -t 10 "http://localhost:8080/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,"\",//crumb")")
#trigger jenkins job with post format http://user:APITOKEN@serveraddress:port/job/jobname
curl -X POST $BUILD_URL/job/gfr_main_build/build?token=gfr2001 -H "CRUMB"
#Status Message
echo -e "$BUILD_URL/jobs/gfr_main_build/buildID"
while [ $GREP_RETURN_CODE -eq 0 ]; do
  echo -e "Building job with jenkins"
  sleep 10
  curl --silent $BUILD_URL/lastBuild/api/json | grep result\:"null" > /dev/null
  GREP_RETURN_CODE=$?
done
#Get the ID of the last build
BUILD_ID=$(wget -q -o $BUILD_URL/job/gfr_main_build/lastBuild/buildNumber)
#Create output file name with the build ID
FILESTRING=$(echo "./build_results_$BUILD_ID")
#Get the console output of the most recent build
OUTPUT=$(wget -qO- $BUILD_URL/job/gfr_main_build/lastBuild/consoleText)
#Write output to file
echo $OUTPUT >> $FILESTRING
#Status Message
echo -e "Build Complete\nSee output at $FILESTRING"
```
DevOps

Roadblocks

- SSH key management
- Docker out of Docker
- Publishing build results to bitbucket
Next Steps

- Parameter passing with user triggered builds
- Integrate testing framework
Simulator

Overview

- Provide detailed simulation
- Accurate data
- Easy to test
- Easy to modify
Simulator

Goals

- Automate testing
  - Test new code before it gets merged to master
- Visualize different parameters
- Make the whole process much easier
Simulator

- Automate testing
  - Test new code before it hits the simulation
- Visualize different parameters
- Make the whole process much easier

Goals
Simulator

- Automate testing
  - Test new code before it hits the simulation
- Visualize different parameters
- Make the whole process much easier
Simulator

Camera Perception → Lidar Perception → Master Control → Finite State Controller → Torque Vectoring and Traction Control → Controls Optimizer → Actuator Controls → Steering Motor

Camera Perception → Mapping and Localization → Boundary Estimation → Trajectory Planner

Sensors → Sensor Processing → Motor Inverters → Brake Pneumatics
Simulator

Tools

- CARLA - core simulation environment
- Rviz - allows to visualize data after simulation
- ROS - development environment

Put pics of rviz
### Lidar attributes

<table>
<thead>
<tr>
<th>Blueprint attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>channels</td>
<td>int</td>
<td>32</td>
<td>Number of lasers</td>
</tr>
<tr>
<td>range</td>
<td>float</td>
<td>10.0</td>
<td>Maximum measurement distance in meters (&lt;=0.9:6 is in centimeters)</td>
</tr>
<tr>
<td>points_per_second</td>
<td>int</td>
<td>56000</td>
<td>Points generated by all lasers per second</td>
</tr>
<tr>
<td>rotation_frequency</td>
<td>float</td>
<td>10.0</td>
<td>Lidar rotation frequency</td>
</tr>
<tr>
<td>upper_fov</td>
<td>float</td>
<td>10.0</td>
<td>Angle in degrees of the upper most laser</td>
</tr>
<tr>
<td>lower_fov</td>
<td>float</td>
<td>-30.0</td>
<td>Angle in degrees of the lower most laser</td>
</tr>
<tr>
<td>sensor_tick</td>
<td>float</td>
<td>0.0</td>
<td>Seconds between sensor captures (ticks)</td>
</tr>
</tbody>
</table>

### simulator

```python
# Find the blueprint of the sensor.
blueprint = world.get_blueprint_library().find('sensor.lidar.ray_cast')
# Modify the attributes of the blueprint to set image resolution and field of view.
blueprint.set_attribute('image_size_x', 1920)
blueprint.set_attribute('image_size_y', 1080)
blueprint.set_attribute('fov', 110)
# Set the time in seconds between sensor captures
blueprint.set_attribute('sensor_tick', 0.0)
# Provide the position of the sensor relative to the vehicle.
transform = carla.Transform(carla.Location(x=0.0, z=-1.7))
# Tell the world to spawn the sensor, don't forget to attach it to your vehicle actor.
sensor = world.spawn_actor(blueprint, transform, attach_to=my_vehicle)
# Subscribe to the sensor stream by providing a callback function, this function is
called each time a new image is generated by the sensor.
sensor.listen(lambda data: doSomething(data))
```
Simulator

Current standing

- Maps made and processed
- Validating simulation
- Integration in progress
- Packaging everything together
- Build script
● Unreal doesn't play nicely on Linux
● Our server doesn't properly support Unreal
● Trying to get tests inputted
● Unreal keeps crashing upon loading of project?
An Unreal process has crashed: UE4-

We are very sorry that this crash occurred. Our goal is to prevent crashes like this from occurring in the future. Please help us track down and fix this crash by providing detailed information about what you were doing so that we may reproduce the crash and fix it quickly. You can also log a Bug Report with us using the Bug Submission Form and work directly with support staff to report this issue.

Thanks for your help in improving the Unreal Engine.

Please provide detailed information about what you were doing when the crash occurred.

Crash reports comprise diagnostics files (click here to view directory) and the following summary information:

Assertion failed: InSize = (-NULL) || InSize + Offset = Size [File:/home/ryan/UnrealEngine_4_22/Engine/Source/Runtime/VulkanRHI/Private/]
libUE4Editor-Core.so:GenericPlatformMisc::RaiseException(unsigned int) [/home/ryan/UnrealEngine_4_22/Engine/Source/Runtime/Core/Plugins/]
libUE4Editor-Core.so:UnrealErrorOutputDevice: Serialize(char t const, ELogVerbosity::Type, FName const&) [/home/ryan/UnrealEngine_4_22/Engine/Source/Runtime/Core/Private/]
libUE4Editor-Core.so:OutputDevice::LogImpl(char16_t const,...) [/home/ryan/UnrealEngine_4_22/Engine/Source/Runtime/Core/Private/]
libUE4Editor-Core.so:AssertFailedImpl(char const*, char const*, int char16_t const,...) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-Core.so::CheckVersionFailedImpl(char const*, char const*, int char16_t const,...) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::Map(unsigned long, unsigned long) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateResource(VulkanRHI::FDeviceResourceHeap::EType, unsigned int, unsigned int) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/]
libUE4Editor-VulkanRHI.so::VulkanRHI::FDeviceMemoryAllocation::AllocateBufferMemory(VulkanRHI::FDeviceMemoryRequirements const, unsigned int, char const*) [/home/ryan/UnrealEngine_4_22/Engine/

☑ Include log files with submission. I understand that logs contain some personal information such as my system and user name
☑ I agree to be contacted by Epic Games via email if additional information about this crash would help fix it

Close Without Sending  Send and Close  Send and Restart
Simulator

Problems

```
[2020.02.24-19.09.50:267][ 0]LogVulkanRHI: Display: 10: Flags 0xe Heap 1 HostVisible HostCoherent HostCached
[2020.02.24-19.09.50:285][ 0]LogVulkanRHI: Warning: Failed to allocate Device Memory, Requested =32768.00Kb MemTypeIndex=10
Assertion failed: InSize == (~0ULL) || InSize + Offset <= Size [File:/home/ryan/UnrealEngine_4.22/Engine/Source/Runtime/VulkanRHI/Private/VulkanMemory.cpp] [Line: 387]


Signal 11 caught.
Malloc Size=65538 LargeMemoryPoolOffset=65554
```
Rendering

Overview

- Provide 3D assets and meshes to the simulator
- Create test tracks
Rendering

Tools

- CLARA map generator - Create base map
- Unreal Engine 4 - Edit and configure 3D Assets
- Sketchup Pro - Convert 3D mesh types
Rendering

Texture Creation
Rendering

Map Building
Testing

Overview

- Messages are changed to have universal data type and name to fit into the new interface
- Each readings from the simulator will have noise to produce real life accuracy
- Process raw data to be handled into specific messages
## Testing

### Goals

- Create a message type that are universal
- Convert raw data into specific message types for clarity and ease of access
- Able to show 100% output data with or without noise (ground truth)
- Setting up tracking ID for all data types to boundary estimation

<table>
<thead>
<tr>
<th>Header</th>
<th>time</th>
<th>time_stamp</th>
<th>seq</th>
<th>frame_id</th>
<th>Note: Native ROS message type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32</td>
<td>seq</td>
<td>frame_id</td>
<td></td>
<td></td>
<td>YES</td>
</tr>
</tbody>
</table>

Note: Native ROS message type
- Static and Dynamic data are separated
- While static provide ground truth elements
- Dynamic data processed with added noise built into it
Testing

Node chart

```
  gfr_sim
    ^
    |  
  car_model  
    ^
    |  
  cone_sensor
    
    
```

```
  car_model noise
    ^
    |  
  node
```

```
  cone_sensor noise
    ^
    |  
  node
```
Testing

Overview for noise nodes

- Each node is subscribed to a message type
- Car model and cone sensor are the only dynamic data that require added noise

```cpp
class Car_Noise{
    public:
    Car_Noise(ros::NodeHandle &nh, nh::nh){
        pub_ground_truth_ = nh.advertise<gfr_sim::State>("/gfr/sim/base_pose_ground_truth", 1);
        pub_car_info_ = nh.advertise<gfr_sim::CarInfo>("/gfr/sim/car_info", 1);
        state_pub_ = nh.advertise<gfr_sim::State>("/gfr/sim/car_model_noise", 1);
    }

    //auto state_pub_ = nh.advertise<gfr_sim::State>("/gfr/sim/car_model_noise", 1);

    sub_f_kin_correction_ = nh.subscribe("/gfr/sim/base_pose_ground_truth", 1, &Car_Noise::noiseCallBack, this);
    //pub_ground_truth_ = nh.advertise<gfr_sim::State>("/gfr/sim/car_model_noise", 1);

    private:
    ros::NodeHandle nh;
    ros::Publisher state_pub;
    ros::Subscriber sub_f_kin_correction_;

    void Car_Noise::noiseCallBack(const gfr_sim::State &msg);
    double Car_Noise::getGaussianNoise(double mean, double var);
};

double Car_Noise::getGaussianNoise(double mean, double var) {
    std::normal_distribution<double> distribution(mean, var);
    // construct a trivial random generator engine from a time-based seed:
    long seed = std::chrono::system_clock::now().time_since_epoch().count();
    std::default_random_engine generator(seed);
    return distribution(generator);
}
```
New data types are integrated into the new interface

Car model noise node are basically done

Cone sensor noise node is in progress

Setting up tracking ID for each cone to run tests
Next term

- More validating data and testing
- Improving usability of the simulator
- Reinforce testing interface into SLAM and Boundary Estimation
- Improving CARLA and Continuous Integration Server