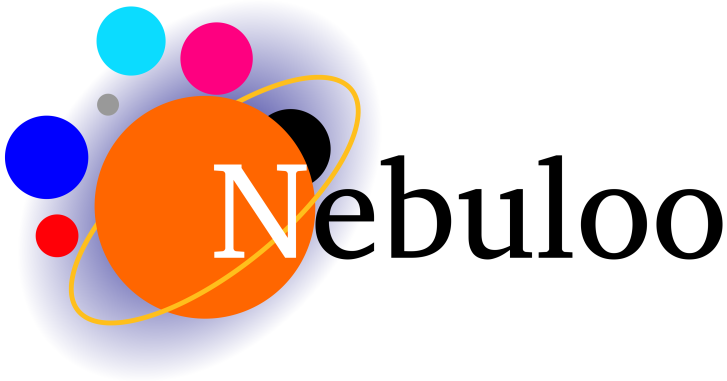
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
THE UNIVERSITY OF TEXAS AT ARLINGTON

PROJECT CHARTER
CSE 4317: SENIOR DESIGN II
FALL 2019



NEBULOO
FALCON

ALEJANDRO WAUMANN
JUAN BARRAGAN
LUIS GONZALEZ
SUNIL NIROULA

Revision History

Revision	Date	Author(s)	Description
0.1	07.05.2019	AW	document creation
0.2	11.14.2019	LG	Made small changes. Added logo
1.0	12.06.2019	LG	official release

CONTENTS

List of Figures

1 Vision

Should every ground-based anti-aircraft weapon be operated by a human? We certainly don't think so. Why have a human (assisted by technology) shoot down an enemy aircraft when we can automate that process? An autonomous process can make decisions faster, shoot with higher accuracy, and improve at superhuman speeds. With enough information, it can also shoot more accurately, differentiate between enemies and allies to ensure that only the enemy is hit, and make more informed decisions as it receives data from multiple sources.

Should these systems be solely trained in physical environments? Simply put, no. It is expensive and many scenarios will not be accounted for due to time constraints and possibly financial constraints. Hyper-realistic simulated training environments are a proven solution that can make the process of training AI-driven operators significantly cheaper and lead to more robust systems due to the larger test-scenario domain and the increased speed at which each scenario can be simulated.

2 Mission

While a sophisticated system would be ideal, we will be building a less sophisticated version of an autonomous anti-aircraft system. It will be able to identify when a plane is flying near it, track it, and shoot it down. While tracking it, our system will be deciding whether or not it should shoot down the plane and when it should fire the missile. The seeker missile itself will use this information to know which target to track and destroy.

Since this system will be AI-driven, it will need to be trained. To do this cheaply and quickly, we will design a high performance vulkan-based rendering engine. It will be designed to specifically support the features we need to make the simulated training environment as realistic as possible. This will allow the learning in simulation to transfer well into the physical world and since more scenarios can be tested with a larger scenario domain, the system will also be more robust.

3 Success Criteria

Upon completion of the alpha release, the software will provide the following services:

- Display a simulated environment
- Display the anti-aircraft weapon camera view
- Allow the user to move the camera around

Upon completion of the beta release, the software will provide the following services:

- Allow the user to switch to autonomous mode
- Autonomous mode will be able to consistently identify and shoot down aircrafts
- The missiles will simulate a seeker missile

Upon completion of the V1 release, the software will provide the following services:

- Manual mode will allow the user to select an object to shoot and the autonomous system will take over to shoot it down
- The missile camera view will be shown while it is airborne in a corner of the user interface
- No support will be provided after the V1 completion.

4 BACKGROUND

There are currently two parts to the problem we are trying to solve. First is the use of human operators for ground-based anti-aircraft weapons. They generally use a remote and a camera view to operate the weapon. This is expensive since there is generally one operator to each machine and extremely inefficient since human brains have low-capacity working memory. This makes decision making in critical situations slower and more unreliable than it should be. Aside from decision making, aiming and tracking to accurately shoot down targets is a difficult task for a human and generally requires a lot of constant practice to remain proficient at performing this task. Second is the issue the training, whether it be humans or autonomous systems, operators to operate these machines taking a lot of time and money in the physical world. It can even require schedules sessions, permits and permissions, and other obstacles that make it more time consuming and expensive to train operators. This time constraints limits the number of test scenarios that operators can be trained on.

The solution to the first problem is an autonomous system. In general, this system should be able to identify, track, and shoot down planes, although this can be generalized to be any flying object. This system should be able to use various data sources to decide when and if it should shoot an aircraft down. However, automating the operator doesnât solve training problems. It will still be expensive and time consuming to train these systems, albeit less expensive than human operators. This is because the autonomous system will still need to be trained on various scenarios and tested in those same scenarios to verify that the system performs as expected and properly makes the correct decision more often than its human counterpart.

The solution to the second problem is a hyper-realistic simulated training environment. It needs to be close enough to reality that training in the simulated environment transfers well into the physical environment. This has already been proven to work well with autonomous robot hands and self-driving cars. It is also used to train human operators for various unmanned vehicles. It would make training cheaper, faster, and more robust. Scenarios can be procedurally generated to cover a large scenario-space than can be done when doing physical training/testing. The simulations can also be ran at very fast speeds to cover years worth of training in days or weeks.

5 RELATED WORK

It has been shown that training in simulation, when done right, can transfer very well to performance in the physical world. This has especially proven useful where autonomous agents controlling machines are involved. While a training environment can be made using Unreal Engine 4 or Unity, they will not be easily scalable since those APIs accommodate many system configurations and donât provide the flexibility needed to take full advantage of the dedicated graphics hardware. For a hyper-realistic simulated training environment, high performance will be crucial. Custom designing a rendering engine using the Vulkan API in C++ will allow us to meet this high performance specification and it will allow us to only support features that we will need for the training environment, which could reduce the amount of overhead performance loss.

Ue4, unity Not high performance Accommodates many systems

Custom vulkan rendering engine High performance Accommodates a single system

Human with remote control Operator has target outlines Still has to track target using remote Operators may forget crucial info when making decision Operators tend to work long shifts which could cloud judgement when tired and make memory worse than it already is Expensive to pay operator Operator is less efficient than machine at controlling these systems

Autonomous system can more informed decisions faster It can aim and shoot targets down faster and more accurately

State of the art recognition systems Computer vision systems in general (in python usually)

6 System Overview

With the current design, there will be a core rendering engine using the Vulkan C++ API. This rendering engine will be used to make the simulation module. The simulation model will output frame data and send it into the object detection/recognition module. This module will alter the frame to add bounding boxes around recognized objects and add labels to them. The altered frame will then be forwarded to the object tracking module along with data on which object it should track. This module will generate a movement suggestion that will be sent the the controller module. The controller module will receive user input data (if in manual mode) and simulated external data that will be available to the software in a physical system. This could include data such as distance and elevation estimates of the target. This will product actions that are sent to the simulation module. The module will provide handles to pan around and to shoot.

The user will be presented with a camera view of the anti-aircraft weapon. It will show the bounding boxes and labels around recognized objects. There will be a software switch for the user to switch between autonomous mode and manual mode. In autonomous mode, the software will look for an aircraft, track it, and shoot it down (if it decided to shoot it down while tracking it). If the software is unsure whether it should shoot it down (decided by some threshold value) then it will show the user a view showing why it is unsure and ask the user whether it should shoot down the target. In manual mode, the user will be able to pan around and select any object with a bounding box. Once selected, the software will takeover until it shoots down the target. The user will be able to cancel this action at any point leading up to the shot. When a missile is launched, its camera view will be overlayed in a corner of the main weapon camera view. It will show the missile seeking out the selected target.

7 Roles & Responsibilities

The product owner and the team members are the only stakeholders for this product since there is no customer. This project will allow the product owner to develop a vulkan rendering engine for a practical purpose that has a business case. It will also allow team members to improve their programming skills, gain experience developing in a team using version management and project management software, and learn to apply computer vision, machine learning, and simulation concepts.

There is one product owner and four team members. The product owner will be the scrum master for the entire duration of the project and will be responsible for guiding the development of the project, organizing team meeting, generating the project documentation, maintaining the project webpage, and developing the vulkan rendering engine. Each team member will be responsible for the development and testing of 1-2 major modules. They will each be responsible for following the coding style and guidelines to ensure that documentation can be properly generated.

Each team member will be responsible for taking their own notes during team meetings and creating status reports at the end of each milestone (after each integration stage). All integration stages will be team efforts since it will require the integration of all major modules into a single version release. Build errors are expected at this stage and we should all work together to resolve these errors as efficiently as possible.

8 Cost Proposal

8.1 Cost Proposal

Major expenses for the project include the integration development/testing system hardware, project management software, and the digital assets to be used for the simulated training environment. The largest expense, by a large margin, will be the system hardware. This will be the custom-built linux system used for the integration and optimization stages of development and testing of the product.

This will be performed for the alpha release, the beta release, and the V1 release. The vulkan rendering engine will be optimized for this machine and all installation/setup scripts will be specifically designed to work on this machine. This will ensure that everything just works and that the software is able to run at the required specifications.

8.2 Preliminary Budget

Item Description	Cost
System Hardware + estimated taxes	$530.00
Desktop Case: Corsair 100R	$59.99
Motherboard: Asus B450 Prime	$103.99
CPU: Ryzen 5 2600	$139.99
GPU: Nvidia GTX 1050 2GB	$99.00
Memory: Corsair Vengeance 8GB (2x4GB) 2400 MHz	$42.99
PSU: EVGA 450W BT	$44.46
Hard drive	$0.00
Monitor	$0.00
Keyboard and mouse	$0.00
Jira Core software	$60.00
Private Server	$60.00
3D Plane Models	$60.00
Terrain Maps	$50.00
Vectorized Logo	$40.00
Total	$800.00

8.3 Current & Pending Support

UTA CSE Department: $800.00

9 Facilities & Equipment

The team will need access to a linux system with a dedicated GPU to complete the project. This is because a large component of the system is a hyper-realistic simulated training environment created using a custom-built vulkan-based rendering engine. It will be optimized to create the training environment and support only the features that are needed to generate a realistic training environment. While access to GPUs may be available at the UTA senior design labs, they will not satisfy the requirement that the system and rendering engine will be optimized for a single system-configuration.

Development will largely occur in the UTA labs and the personal laptops of team members; however, integration and optimization development/testing will occur on a linux machine running arch linux and customized to meet the desired specification of the project software. This machine will be purchased

using a part of the budget supplied by UTA. Since there will only be one machine, integration development/testing will be done in a team environment and each team member will be able to ssh into the machine for integration development. Final testing and optimization will be done locally on the machine itself.

10 Assumptions

The following list contains critical assumptions related to the implementation and testing of the project.

- UTA will provide financial support in a timely manner
- A basic rendering engine will be ready by the 5th sprint cycle
- Team will have access to GPU for development and testing
- Dataset used to train recognition model will be large enough
- The development system will also be the delivered system
- System parts will arrive in time for the first integration stage

11 Constraints

The following list contains key constraints related to the implementation and testing of the project.

- Final prototype demonstration must be completed by December 1st, 2019
- The dataset used to train the object recognizer module will be limited in size
- Total development costs must not exceed $800
- Team does not have prior experience with computer vision and machine learning
- C++ must be used since the rendering engine must be high performance

12 Risks

The following high-level risk census contains identified project risks with the highest exposure. Mitigation strategies will be discussed in future planning sessions.

Risk description	**Probability**	**Loss (days)**	**Exposure (days)**
Rendering engine not being ready in time for first integration stage	0.10	21	2.1
Availability in 3D models due to financial support delay	0.30	14	4.2
Failure to finish major modules in time for integration stages	0.20	7	1.4
Failure to resolve build errors during integration stages	0.50	7	3.5

Table 1: Overview of highest exposure project risks

13 Documentation & Reporting

13.1 Major Documentation Deliverables

13.1.1 Project Charter

The project charter will be maintained using Google Docs. Each section will be its own document on a shared Google Drive folder. Each document will contain a history of any and all changes made. Changes made to each of the documents will be persisted to the tex files on the project repository. Those tex files will be used to generate the project charter PDF. This document will be updated on a need basis. The initial version will be delivered on July 7, 2019 and the final version will be delivered on December 6, 2019.

13.1.2 System Requirements Specification

The system requirements specification will be maintained in the project repo. We will be using Jira Core to track issues and changes to any system requirement as development proceeds. This document will change on a need basis. The initial version will be delivered on July 22, 2019 and the final version will be delivered TB.

13.1.3 Architectural Design Specification

The architectural design specification will be maintained in the project repo. This document will change on a need basis. The initial version will be delivered on July 22, 2019 and the final version will be delivered on December 6, 2019.

13.1.4 Detailed Design Specification

The detailed design specification will be maintained in the project repository. This document will change on a need basis. The initial version will be delivered on July 22, 2019 and the final version will be delivered on December 6, 2019.

13.2 Recurring Sprint Items

13.2.1 Product Backlog

Each task will be broken down into subtasks that can be completed independently. Any tasks that depends on another task will be tagged as such. Task items will be prioritized by a measure of value provided for completing the item. The product owner, Alejandro Waumann, will decide how the items should be prioritized and what items should go into each sprint. The product backlog will be tracked and managed using Jira Core software with the scrum method.

13.2.2 Sprint Planning

The product owner will plan each sprint at the end of the current sprint, with the exception of the very first sprint. Using the current sprint progress, along with a general milestone timeline, the sprint for the week will be planned. There will be a total of 16 sprints.

13.2.3 Sprint Goal

The product owner will decide the sprint goal. There is no customer planned for this project so only the product owner will be involved in this process.

13.2.4 Sprint Backlog

The product owner decides with backlog items make their way into the sprint backlog. The backlog will be tracked and maintained using a scrum board with Jira Core software. This software will allow the product owner to easily assign items to each team member and monitor their progress.

13.2.5 Task Breakdown

Each individual task will be assigned to a team member. Once a sprint is drafted, team members will pick which task they will do on a first-come first-serve basis. Any conflicts during this process will be resolved by the product owner and the team members involved in the conflict. Each team member will document any time spent completing the task using the Jira Core software.

13.2.6 Sprint Burn Down Charts

The product owner will be responsible for generating the burn down charts for each sprint. Using the Jira Core software, the product owner will be able to monitor the total amount of effort expended by each individual team member. The format for the burn down chart is TBD.

13.2.7 Sprint Retrospective

After every sprint and before the start of the next sprint, there will be a sprint retrospective meeting. The team will discuss things they struggled with to see what part of the process can be improved. The team will document the general discussion and each individual will take notes on what they plan to improve in the upcoming sprint.

13.2.8 Individual Status Reports

Individual status reports will not be essential since Jira Core will be used to track and manage sprint progress. However, at the end of each milestone (alpha, beta, v1 release), each member will report on major individual accomplishments, improvements, and struggles so that they can create an improvement plan for the next milestone.

13.2.9 Engineering Notebooks

Engineering notebooks will be updated at every team meeting, which will occur at a minimum of once a week. Any design work done to complete a task should be documented in engineering notebooks. There will be a minimum of one page per week-long interval.

13.3 Closeout Materials

13.3.1 System Prototype

The final system prototype will show the proficiency of the autonomous system in its hyper-realistic simulated training environment. This will be demonstrated on the hardware used to develop, integrate, and test the software during the senior design presentations.

13.3.2 Project Poster

The project poster will include a general overview of how the autonomous system functions and one of how the hyper-realistic simulated training environment functions. It will also include difficulties that needed to be overcome and features that are implemented in the V1 release. The final dimensions and the delivery date are TBD.

13.3.3 Web Page

The project web page will include the rendering engine documentation, autonomous system documentation, and a video demoing the technology. It will be accessible to the public until the project server goes offline. It will be updated throughout the project. The team will need access to the rendering engine documentation to develop the autonomous system software.

13.3.4 Demo Video

The demo video will give a general overview of the autonomous systemâs information processing. It will also show the system at work in the hyper-realistic simulated training environment. All features

will be demoed throughout the video. It will be roughly two minutes long.

13.3.5 Source Code

Source code will be maintained in the project git repo. Jira Core will be used to link sprint items to changelists. Since there is no customer, no source code or binaries will be provided. The rendering engine will be closed-source and the autonomous system will be open-sourced using the MIT license terms (it will include the rendering engine library as a dependency). The license terms will be listed in a single âLICENSEâ file. The code will be open-sourced after the V1 release.

13.3.6 Source Code Documentation

Doxygen will be used to generate source code documentation. The final documentation will be provided in a browsable HTML format on the product webpage.

13.3.7 Installation Scripts

There will be installation scripts to setup the linux development/deployment environment. After the installation scripts are run, the user can just run program by typing âfalconâ into the terminal. The installation script will setup defaults configurations for the training environment. A script to update the object recognition model will also be provided.

13.3.8 User Manual

A digital manual will be available on the project webpage in HTML format and on the open-sourced git repo in markdown format. There will be no setup video. The digital manual will provide installation instructions and the open-sourced repo will have modular setup scripts.

References