Nebuloo – Project Falcon

Alex Waumann, Sunil Niroula, Juan Barragan, Luis Gonzalez
CSE Senior Design

University of Texas Arlington | College of Engineering

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.

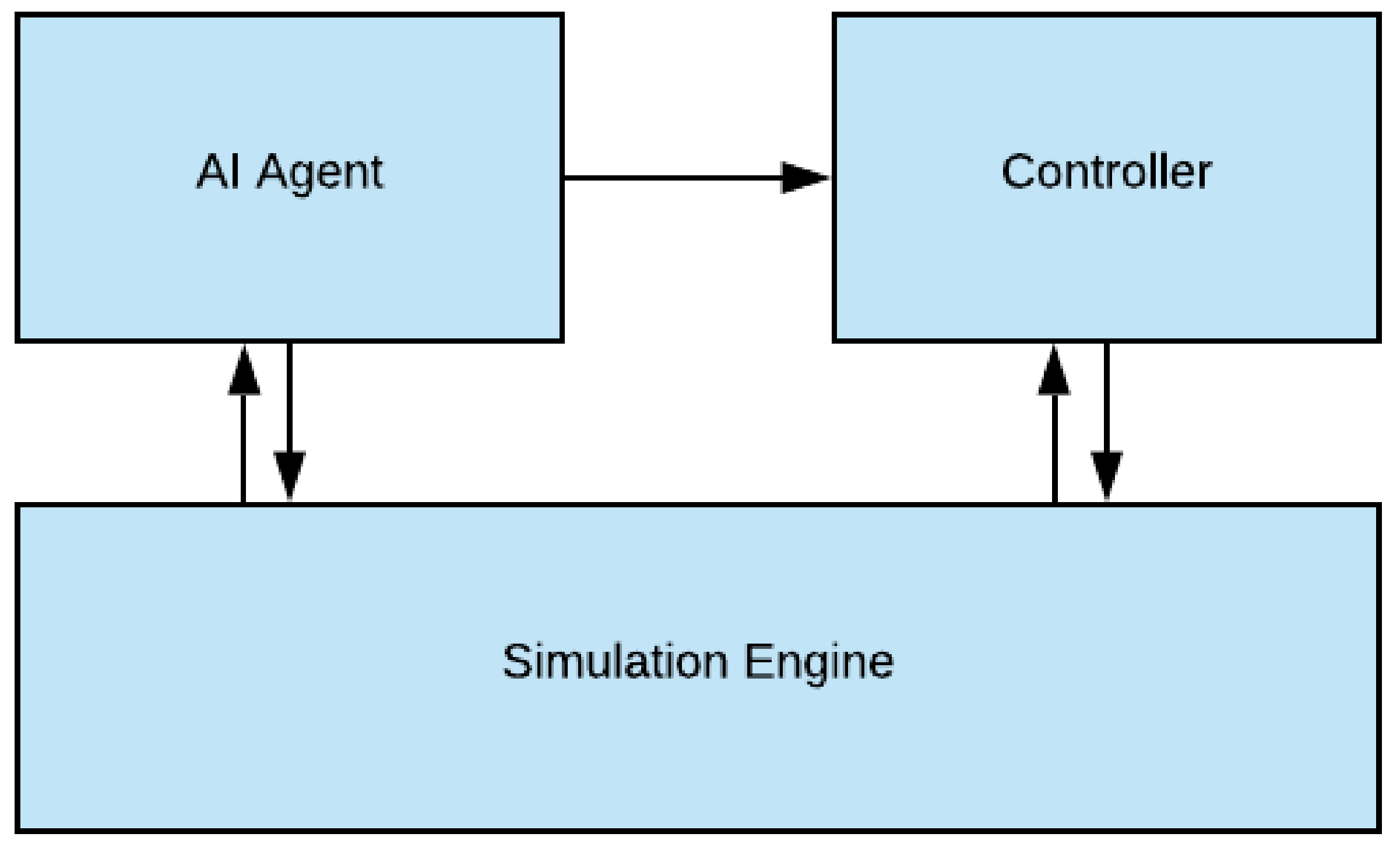
Core Features

The overall conceptual design offers the end user with a desktop application that can be used to generate simulated 3D environments that can be used to train AI models in.

#	Feature	#	Feature
1	Simulation engine will be able to import 3D models in glTF format.	6	Controller layer should handle user input and map to certain actions.
2	Be able to create simulated environments using existing 3D models.	7	Controller layer will allow the user move around the simulated environment.
3	AI Agent layer should be able to detect specific objects in the environment.	8	Simulation engine should be able to render scenes at a constant 60fps.
4	AI Agent layer should be able to track detected objects in the environment.	9	Simulation engine + AI agent should run together at 24 fps at the very least.
5	Simulated environment can be used to train AI models.	10	AI Agent Layer + Controller layer should work together to track moving objects in a scene.

Design Details

The simulation engine layer will feed the AI agent layer a frame buffer It will also feed the controller layer information about the state of the simulation and settings that the user may have set that should influence the move evaluation. The AI agent will give the simulation engine a modified frame buffer since it will act as a filter that add bounding boxes and label to objects that it recognizes. It will also send its move suggestion to the controller layer. The controller layer will only send data to the simulation engine layer to alter the state of the simulation.



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.

System Architecture Diagram

* Controller will use renderer to create models and set their transformations.

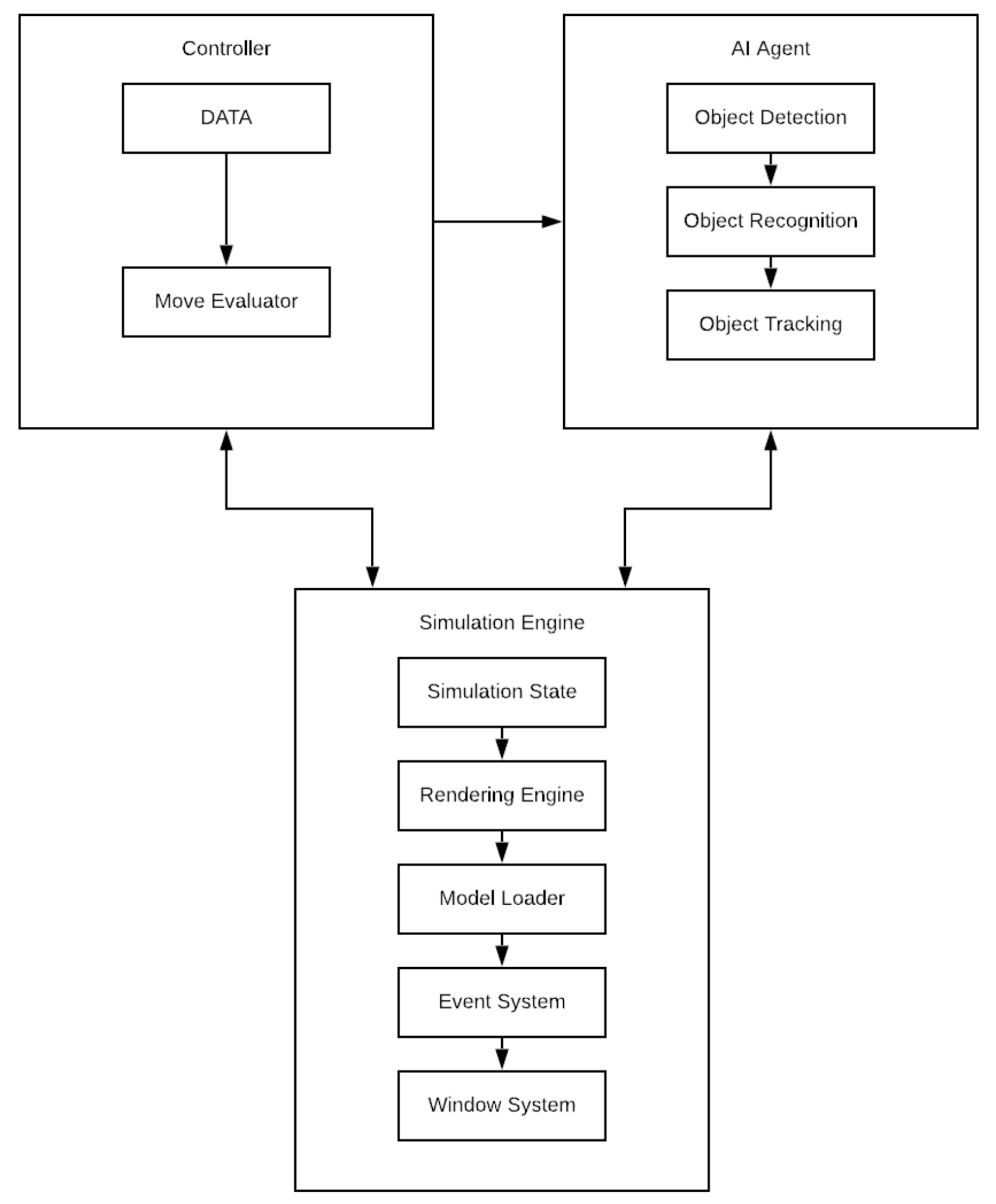
* The controller uses the dedicated input system to process input and take action based on that action.

* Rendering engine will provide an image that the controller can take and hand off to AI system.

* The AI system will specify the bounding box coordinates that the rendering engine should draw.

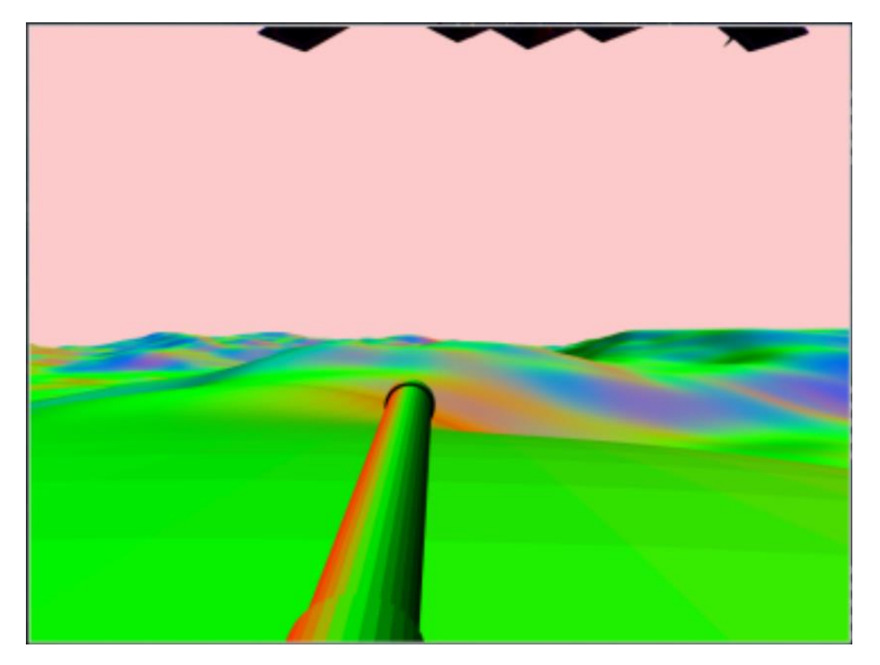
* The controller will hand off the rectangle coordinates to the rendering engine

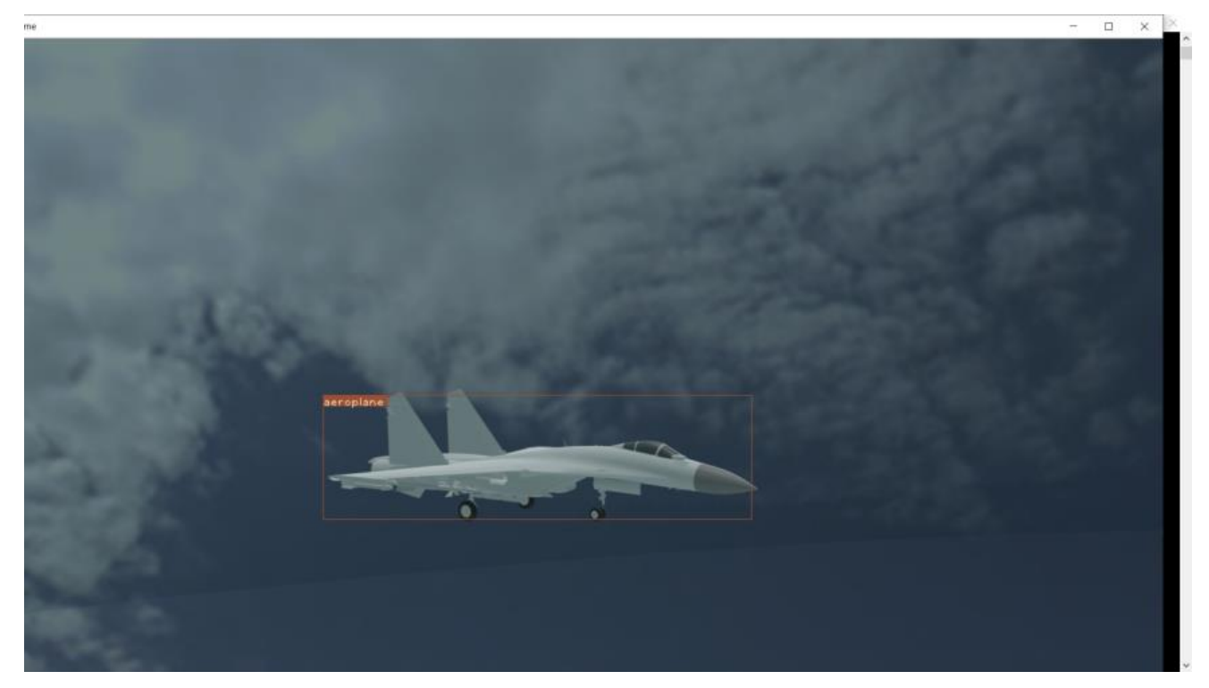
* The rendering engine will draw the rectangle.



Current Status

As of demo day, the simulation engine has met a lot of the planned features, however, the engine is nowhere near completion as the task proved to be more technically challenging than originally anticipated. The controller layer is mostly there, handling the user input to be received by the simulation engine. The AI agent/layer works individually, but we ran out of time to integrate it with the rest of the system. The AI agent/layer is able to detect and track basic objects, however, the pipeline to connect the controller to this layer proved to be quite time consuming and more complex than originally thought.





References

1. **Kronos Vulkan Documentation**, https://www.khronos.org/registry/vulkan/specs/1.1/styleguide.html

2. **Vulkan Tutorial**, https://vulkan-tutorial.com/Introduction

3. **Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3**, by Jonathan Hui, https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

4. **YOLO: Real-Time Object Detection**, https://pjreddie.com/darknet/yolo/