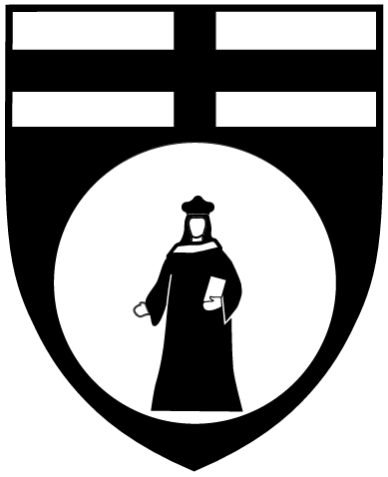


Dibris Dipartimento di Informatica, Bioingegneria,
Robotica e Ingegneria dei Sistemi

VR FOR ROBOTICS

Project Report

Dynamic Hazard Simulation and Autonomous Response in Industrial Environments

---

Professor: Gianni Viardo Vercelli
Advisor: Subhransu Sourav Priyadarshan

UNIVERSITY OF GENOA, APRIL 2025

Member list & Workload

No.	Full name	Student ID	Percentage of work
1	Josue Tinoco	5835667	25%
2	Nima Abaeian	5967579	25%
3	Dikshant Thakur	5943225	25%
4	Girum Molla Desalegn	6020433	25%

Contents

1 Concept Overview

- **Brief Description:** "Dynamic Hazard Simulation and Autonomous Response in Industrial Environments" is a project focused on designing a realistic simulation of an industrial refinery environment. It includes dynamic hazard detection and immediate response systems using both UAVs (Unmanned Aerial Vehicles) and UGVs (Unmanned Ground Vehicles), that can work in perfect synchronization, with custom-developed real-time communication. The project is designed to monitor floating roof storage tanks and respond autonomously to any detected hazards or anomalies, ensuring safety and efficiency in the industrial area.
- **Objectives:**

1. Create a realistic simulation environment of an industrial refinery area using Unreal Engine.
2. Code and simulate the movement of the floating roof storage tanks during operation, for both inflow and outflow of oil.
3. Develop a smart UAV system capable of detecting hazards in the storage tanks, such as tilting roofs or presence of oil spillage.
4. Design a UGV capable of handling real-time communication and coordination with the UAV for more efficient strategies to deal with important inspections.

- **Key Feature:** The integration of real-time hazard detection and autonomous response, managed by collaboration of UAV and UGV. This is further enhanced by the dynamic simulation of floating roof storage tanks, which adds an additional layer of realism to the system.

2 Target Audience

- **Primary Users:** Industry engineers and safety managers, particularly those involved in refineries and other similar hazardous environments. The project focuses on those who rely on traditional labor-intensive hazard detection methods, such as manual inspections or manually controlled UAVs. By automating hazard detection and response, we hope to demonstrate the efficiency and safety benefits of modern autonomous technologies tailored to their industry.
- **Secondary Users:** Students, researchers, and professional developers, interested in areas such as industrial automation, robotics, or environmental safety.

3 Example of Use Case

Actors

- **Primary Actor:** Refinery Operator
- **Secondary Actors:** UAV, UGV

Basic Flow

- **Operator Initiates Monitoring:**
  - Operator starts the monitoring system, which triggers the UAV to begin its patrol.
- **UAV Patrols Refinery:**
  - UAV autonomously flies predefined routes, capturing real-time video and sensor data.
  - UAV uses computer vision algorithms to detect anomalies, such as:
    - Unusual roof movement of floating storage tanks
    - Spills or leaks
    - Unauthorized personnel or vehicles
- **Anomaly Detection:**
  - Upon anomaly detection, the UAV will generate an alarm and signal the UGV, which will then further patrol the specified area and report its findings back to the operator.
- **Air/Ground Drone Response:**
  - UAV adjusts its flight path to focus on the anomaly area, capturing detailed images and videos.
  - UGV is dispatched to the ground-level location indicated by the UAV, using the real-time data to navigate.
- **Operator Assessment and Response:**
  - The operator reviews the data from the UAV and UGV to assess the severity of the situation.
  - Based on the assessment, the operator may:
    - Dispatch emergency response teams
    - Activate safety protocols
    - Implement corrective actions
- **Continuous Monitoring:**
  - The UAV and UGV continue to monitor the situation, providing real-time updates to the operator.
  - The system adapts to changing conditions, adjusting flight paths and ground-level inspections as needed.

4 Technical Information

- **Technologies & Software:**
  - Unreal Engine 5.2 as the simulation environment
  - Python for programming core algorithms
- **Platforms:**
  - Desktop-based simulation for Windows 10/11
- **Programming Languages:**
  - Python, C++, and Unreal Engine's Blueprint visual scripting system
- **Frameworks:**
  - AirSim SDK for Unreal Engine integration with UAV logic.

5 Development Plan

- **Team and Roles:**
  - **Josue Tinoco: Refinery Environment and Floating Roof Tank Dynamics**
    - Builds the Unreal Engine refinery environment.
    - Develops and codes the simulation model for the floating roof storage tanks.
    - Focuses on the movement of the tank roof during oil inflow and outflow.
    - Ensures high accuracy and realism in the tank geometry and behavioral simulation.
  - **Girum Molla Desalegn & Dikshant Thakur: UAV Model Development**
    - Designs and implements the UAV model, including movement dynamics and sensor integration.
    - Develops flight patterns for spill detection and tank monitoring.
    - Ensures the UAV operates effectively within the existing VR refinery environment.
    - Focuses on real-time anomaly detection and accurate monitoring capabilities.
  - **Josue Tinoco & Nima Abaeian: UGV Development & Real-Time Communication**
    - Real-time communication environment setup for UGV.
    - Developed code for UGV communication.
    - Coordinates and generate real time UGV responses based on real-time data shared by UAVs.
    - Implemented safe inspection path for UGV.
- **Timeline:**
  - **Concept**: Delivered on December 20th.

- **Prototype**: The first simulation prototypes of each section will be completed before the end of 2024.
- **Beta**: First half of January 2025. This will be a test heavy phase where any bugs or unintended actions can be corrected.
- **Launch**: First week of April. The project will be ready for assessment.

6 State of Art

The combination of virtual reality (VR) simulations and autonomous systems is revolutionizing industrial safety by enabling more efficient hazard detection and response. These technologies are particularly crucial in high-risk environments like oil refineries, where traditional methods often fail to address the complexity and urgency of potential dangers.

Context and Importance

Oil refineries face persistent risks, such as oil spills and structural failures, which require rapid and effective responses. Conventional safety systems are highly dependent on human intervention, leading to delays and inaccuracies under challenging conditions such as low visibility or night-time operations. The introduction of UAVs (Unmanned Aerial Vehicles), UGVs (Unmanned Ground Vehicles), and VR simulations offers innovative real-time solutions while reducing human exposure to dangerous tasks.

Advances in Research

Recent developments highlight the transformative role of autonomous technologies in industrial safety. Thomas De Kerf and his team created a UAV system that uses infrared and RGB cameras with machine learning to detect oil spills, achieving 89 percent accuracy across diverse environments. This shows its potential for fast and reliable hazard identification.

Furthermore, Fabio Augusto de Alcantara Andrade and colleagues leveraged virtual reality environments to develop a high-fidelity simulation platform for testing autonomous systems in solar plant inspections. Built with Unreal Engine, this approach reduced the risks and costs associated with physical testing while providing a robust framework for refining autonomous systems. Applying such methods to refinery settings could optimize UAV-UGV coordination for hazard management.

Challenges and Opportunities

Despite their immense potential, autonomous systems are not without challenges. Issues like sensor errors and interference from environmental factors can compromise their reliability. To address these limitations, integrating human oversight into autonomous operations remains essential. As the technology continues to evolve, its ability to function effectively in diverse and unpredictable conditions will ultimately determine its success in industrial applications.

7 Tools

1. **Unreal Engine 5.2**

   - **Description:** Unreal Engine 5.2 was the primary tool used for creating the environment, modeling the landscape, and integrating various systems like the floating roof tanks, UAV, and UGV. This version of Unreal Engine was selected because it was the most up-to-date version compatible with AirSim. Versions 5.3 and later gave issues, making 5.2 the most stable choice for this project. Unreal Engine enabled the creation of realistic industrial environments, dynamic systems, and the integration of AirSim for UAV control.

2. **AirSim**

   - In this project, AirSim was used because it provides realistic physics simulation and sensor modeling in a virtual environment. This allowed us to test and develop autonomous navigation, perception, and control algorithms in a virtual environment which was created with the help of Unreal Engine 5.2, while also offering the capability to obtain real-time data and monitoring, all within a single plugin.

3. **Blender 4.3**

   - **Description:** Blender 4.3 was used to model custom assets, such as the containment barriers, which were missing from the available asset packs. These assets were then integrated into the Unreal Engine environment to complete the refinery setup. It was also used to obtain the raw version of the Unmanned Ground Vehicle (UGV) model and export it to UE's specifications.

For more details on the project, including code, please refer to our GitHub repository.

8 Description

8.1 Landscape and Industrial Area Design

- **Set up the landscape:** A heightmap from Earth Explorer (U.S. Geological Survey) was used to recreate a section of Busalla in Unreal Engine, focusing on the Oil Refinery area. The 1-ARC resolution (30m per data point) proved insufficient for accurately depicting elevation changes near the refinery, the highway, and the river. To address this, the terrain was manually sculpted, smoothed, and adjusted using UE's Landscape tool.

  For materials, various options were explored, including a custom landscape material. The MW Landscape Auto Material from the Epic Games Store was ultimately selected for its ease of use and visual quality. Materials were carefully assigned based on elevation. Initially, 3D grass and trees were added but later removed to optimize performance and reduce project size. The river was recreated using Epic Games' water plugins, and environmental elements like lighting, clouds, and fog were adjusted for a cinematic effect.

- **Create the refinery area:** Most 3D assets were sourced from the Fab.com store, particularly a chemical plant asset pack. Each asset was carefully placed to fit the refinery layout, which, while not a 1:1 recreation of IPLOM, retained key features such as the five tanks on the left, the piping network, the central road, and industrial structures on the right. Some artistic liberties were taken for the design, but essential elements were preserved. Missing assets, such as containment barriers, were modeled in Blender, including UV mapping for material compatibility.

  For the storage tanks, a non-floating roof model was modified using UE's modeling tool. This involved removing unnecessary components and adjusting dimensions to match the IPLOM refinery. The roof was then separated to make it dynamic (explained in the next section).

8.2 Dynamic System for the Floating Roof Storage Tanks

- **Dynamic Roof Blueprint:** An Actor blueprint was created with the Tank Roof as a component. The event graph logic smoothly updated the roof's position, adjusting it within predefined maximum (filled) and minimum (emptied) values. Each roof was assigned an ID to link it to a specific tank. The speed and trigger for movement were controlled through a widget. Most of the code involved conditional blocks and mathematical calculations for variable values, including time, speed, linear interpolation, and updating the actor's Z-location each tick.

- **User Menu Widgets:** To allow dynamic changes to tank states, actions, and speeds, a user interface resembling a pause menu was created using a widget blueprint. The Canvas contained a widget switcher to toggle between the "Control Menu" and "Storage Tanks" menus. Logic was built around "On Clicked" events, with a large portion handling button color updates based on selections. The "Event Construct" retrieved an array of Dynamic Roof Actor instances, which were looped through in the "Send Data" section

to send triggers to each tank. If the ID didn't match the tank, the trigger was discarded in the roof actor's blueprint.

- **Global Widget Control:** Initially, widget creation and visibility toggling using the TAB key were handled in the PlayerController blueprint. Due to AirSim restrictions, this was later moved to the Level blueprint.

8.3 Unmanned Aerial Vehicle

- **Installation of AirSim:** To set up AirSim for UAV control, we began by installing the required dependencies. We installed Visual Studio Community 2022 along with necessary components, including "Desktop Development with C++" and the Windows 11 SDK. After installing the dependencies, we cloned the AirSim repository from GitHub. It's recommended to avoid installing AirSim on the C drive to prevent potential script issues. After cloning, we built AirSim by navigating to the AirSim folder via Developer Command Prompt for VS 2022 and running the command "build.cmd". This started the installation process, which may take some time.

- **Setup of AirSim in Unreal Engine:** After installing AirSim, we created a new project in Unreal Engine 5.2 with the "Game Mode" template and selected C++ for development. We then copied the AirSim plugin into the project directory, opened the project in Visual Studio, and configured the project to use the AirSim plugin by modifying the necessary project files. The setup was completed by selecting the AirSim game mode within Unreal Engine.

- **Modification of Python Client and JSON File:** For UAV control, we created a Python client by modifying a script `hello_drone.py` to suit the project's needs. Additionally, we edited the AirSim JSON configuration to define the required sensors, such as distance sensors for inclination detection and an RGB camera for oil spill detection.

- **Python Environment Setup:** With the Python client and JSON file configured, we modified the Python script to retrieve and display data from AirSim, including real-time updates of the UAV's position and the tank's inclination. We used heuristic functions to calculate the tank's inclination based on sensor data and classified the inclination into secure, monitoring, and danger zones. To control the UAV, we implemented functions to move it to specific coordinates. OpenCV was used to detect oil spills from the camera data.

- **Multi-processing for Real-Time Data Processing:** To handle multiple tasks simultaneously, such as UAV control and data processing, we used multi-threading. This allowed functions to run concurrently and ensured smooth updates. A queue system was used to manage shared data between threads, and an alarm system was implemented to notify of hazardous situations.

8.4 Unmanned Ground Vehicle

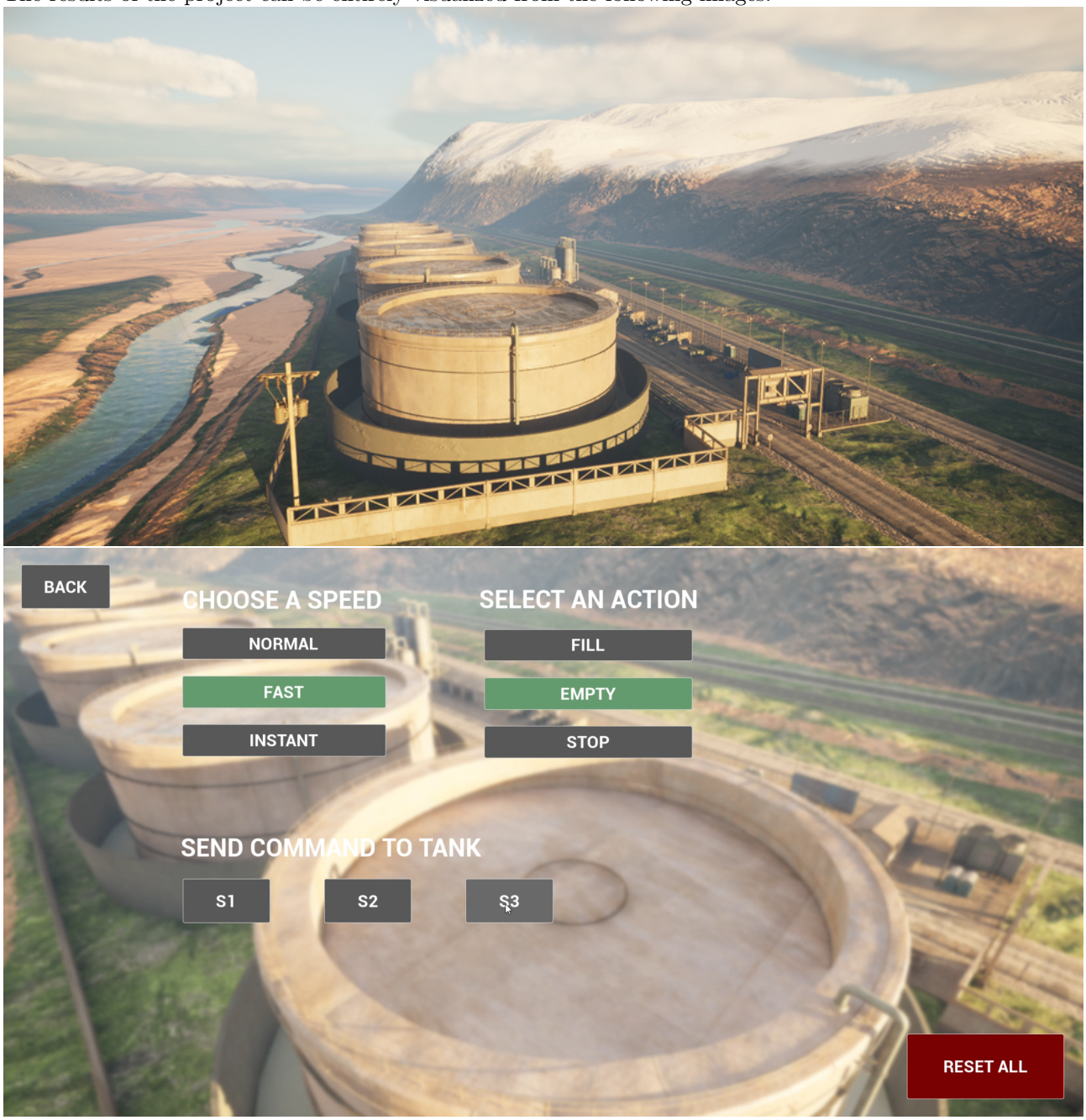
- **Real-time Communication:** The OSC plugin allowed us to create a UDP socket server with a specific name, IP address, and port number. Initially, the code was placed in the GameModeBase blueprint to

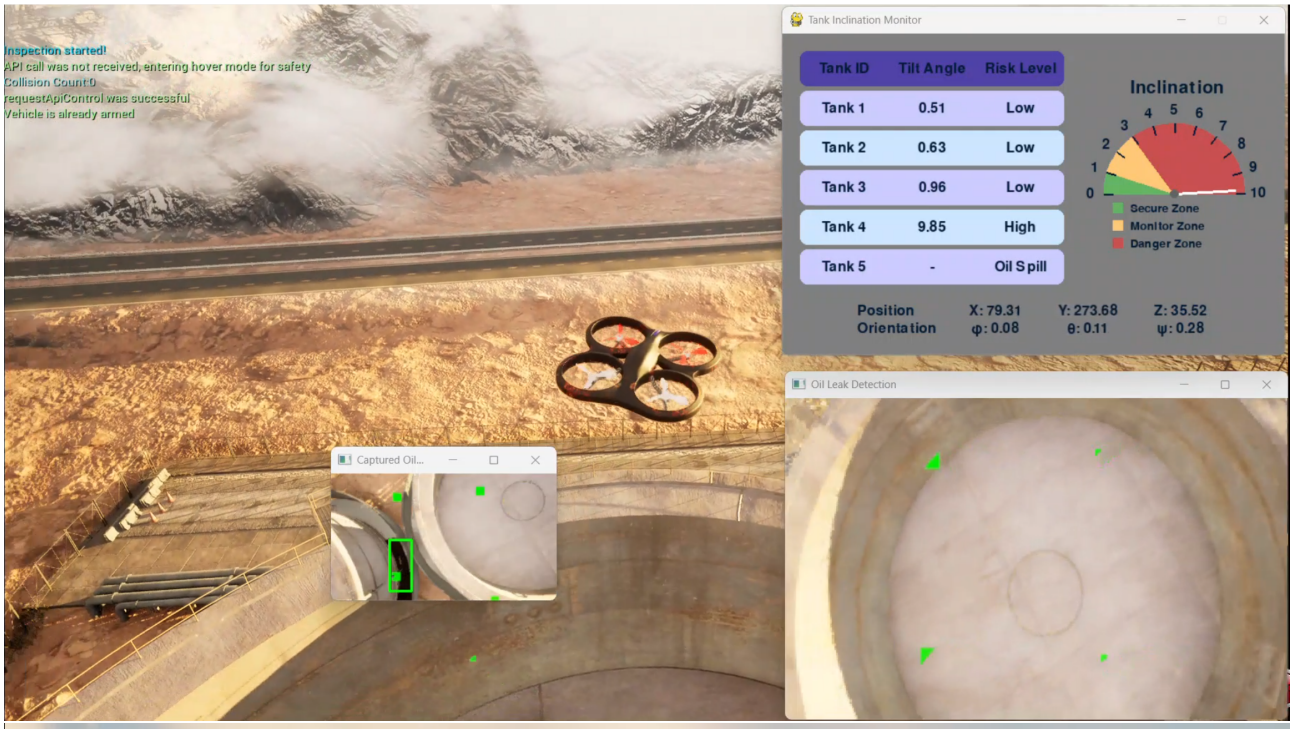
receive incoming messages from the AirSim Python file whenever an alarm was triggered by a storage tank. This message was then sent to the UGV blueprint for processing. Due to AirSim restrictions, the communication logic was later migrated to the Level blueprint.

- **Initial Setup for the UGV (Rover):** The UGV was used to demonstrate real-time communication and coordination with the UAV. Rather than implementing complex algorithms like AI Navigation or autonomous navigation, the UGV's task was simplified to moving to a storage tank in alarm, simulating an inspection, returning to its base, and simulating sending data to a control room. A free-to-use 3D asset of a Ground Rover was selected and modified to allow for a separation between the body and wheels, enabling animation during movement.
- **Creation of Spline Blueprints:** The spline blueprint was created as an Actor Blueprint with a Spline component. Each spline added to the level was assigned an ID variable for identification. The spline path was created with careful attention to avoid sharp turns, sudden jumps, or angle discrepancies.
- **Creation of Rover Blueprint:** The logic behind the Rover blueprint can be divided into the following sections:
  - **Obtain Spline Actors:** An array was created containing all the spline actors placed in the level. These were classified accordingly and stored as variables for later use.
  - **Alarm Queue:** All incoming alarms were placed in a queue (First in, First out). These were obtained from the blueprint that creates the OSC server.
  - **State Machine:** The rover dynamically switched between four states during active operation: move, inspect, return, and send.
  - **Move/Return Action:** A custom timeline was used to change the rover's location and rotation along the spline path, based on a carefully created time curve.
  - **Wheel Control:** The wheel speed was set to ensure the rover moved at the appropriate speed while in the move or return states.

9 Results

The results of the project can be entirely visualized from the following images.




Inspection started!
API call was not received, entering hover mode for safety
Collision Count:0
requestApiControl was successful
Vehicle is already armed
Captured Oil...
Tank Inclination Monitor
Tank ID
Tilt Angle
Risk Level
Tank 1
0.51
Low
Tank 2
0.63
Low
Tank 3
0.96
Low
Tank 4
9.85
High
Tank 5
-
Oil Spill
Inclination
0
1
2
3
4
5
6
7
8
9
10
Secure Zone
Monitor Zone
Danger Zone
Position
X: 79.31
Y: 273.68
Z: 35.52
Orientation
φ: 0.08
θ: 0.11
ψ: 0.28
Oil Leak Detection


The tank S5is in alarm status!
Alarm received from server: /VR_server
UGV is moving towards tank 4!
The tank S4is in alarm status!
API call was not received, entering hover mode for safety
requestApiControl was successful
Collision Count:0
Vehicle is already armed

10 Conclusions

This project focuses on the creation of a highly detailed and functional industrial environment, with an emphasis on the operation of unmanned aerial and ground vehicles for monitoring and inspecting an oil refinery. The landscape was refined using UE's Landscape tool to enhance accuracy. The refinery area was meticulously designed with 3D assets, many sourced from the Fab.com store, while custom assets were modeled to fill missing elements. The dynamic system for floating roof storage tanks was integrated into the environment, allowing for real-time control and updates via a custom user interface.

The integration of the AirSim plugin facilitated UAV control, with Python scripts and sensors aiding in real-time data processing for tank inclination and oil spill detection. Multi-processing ensured smooth operation for simultaneous tasks, with an alarm system notifying hazardous situations. The Unmanned Ground Vehicle (UGV) was designed to communicate in real-time with the UAV, responding to alarm triggers and performing inspections.

For future works, there are several enhancements that could be made to further improve the project's functionality and realism. For example, the UAV could be equipped with a system to monitor the speed of inflow and outflow of oil, triggering additional alarms when necessary. The UGV could benefit from a more complex AI navigation algorithm, allowing it to follow predefined inspection patterns and execute more diverse actions. The User Interface could also be embedded into a control room within the environment to enhance immersion, displaying live data from both UAV and UGV systems. Furthermore, addressing the landscape issues caused by AirSim's source code, which affects the elevation and materials, would improve the overall visual quality and realism of the environment.

References

[1] Fabio Augusto de Alcantara Andrade et al. Virtual reality simulation of autonomous solar plants inspections with unmanned aerial systems. In *IEEE*, 2021.

[2] Thomas De Kerf et al. Oil spill detection using machine learning and infrared images. *Remote Sensing*, 2020.