IREP Final Presentation

Ashwin Pillai & Joshua Zhu

Who we are and what we do

Joshua Zhu

University of British Columbia, Canada

- Mechanical/Aerospace Engineering Undergraduate
- Part of Aeronautics Robotics Rocketry Club (ARRC) at UBC
- Part of Akaflieg at TU Darmstadt
- Made in Macau/Hong Kong

Ashwin Pillai

University of Colorado Boulder

- Aerospace Engineering Undergraduate at the University of Colorado Boulder
- Part of CU in Space (High Power Rocket Club)
- Love to ski
- Born and raised in Portland, Oregon, USA

Initial Goal

Coming into this research program, we knew our job was to "Develop and Integrate a Flight Deck Simulator Function," but that is a very broad description and it took a long process to figure out what we were actually going to create. We went through a long process of testing and research before creating our actual final functions.

Agenda

1.0 **First Project**
Familiarization exercise

2.0 **Initial Research**
On research assignment

3.0 **Ashwin's Creations**

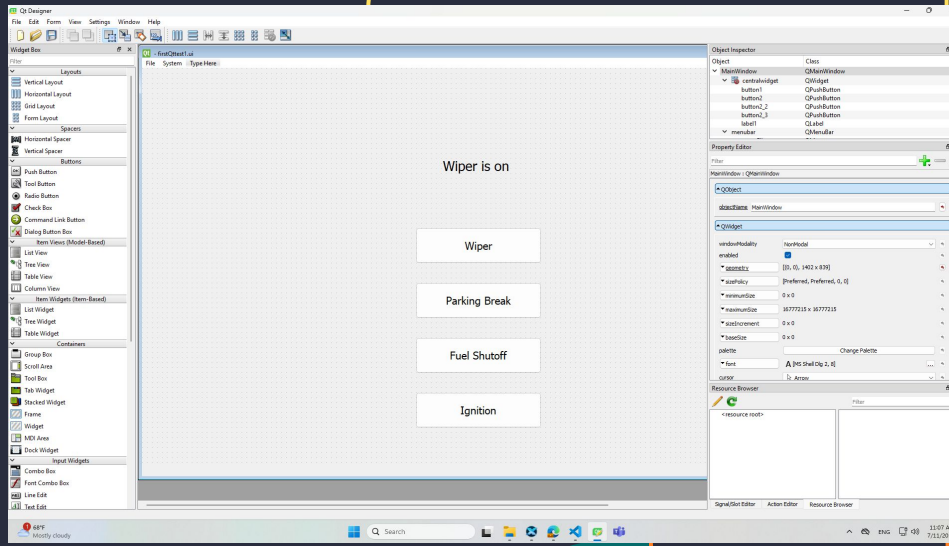
4.0 **Joshua's Creations**

5.0 **Final Product / Conclusion**

1.0 First Project

Introduction to X-Plane

We started out learning XPlane functions and the coding workflow of creating a function for the simulator by designing a user interface. We were given a task to create a user interface that controlled the functions in X Plane. To create the create the UI, we used QT Designer which allowed us to add menus, buttons and other UI tools which were then connected to the X Plane with Python



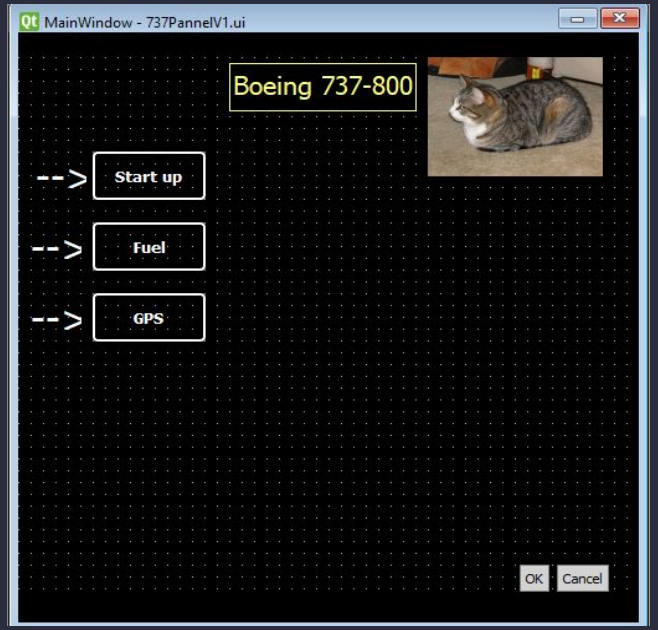
1.0 First Project
Using Datarefs

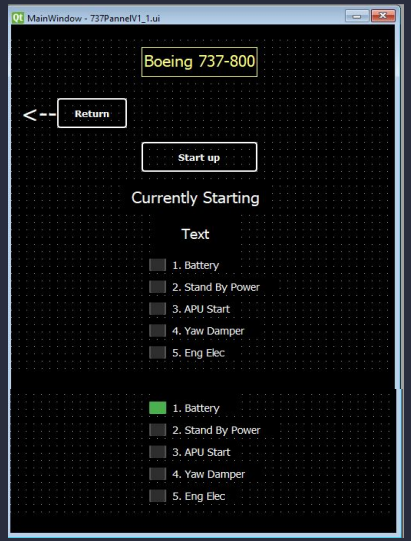
To connect the UI buttons to X Plane, we first needed to find the datarefs correlating to each X Plane function. We use these datarefs to write the logic that connects the buttons to these functions.

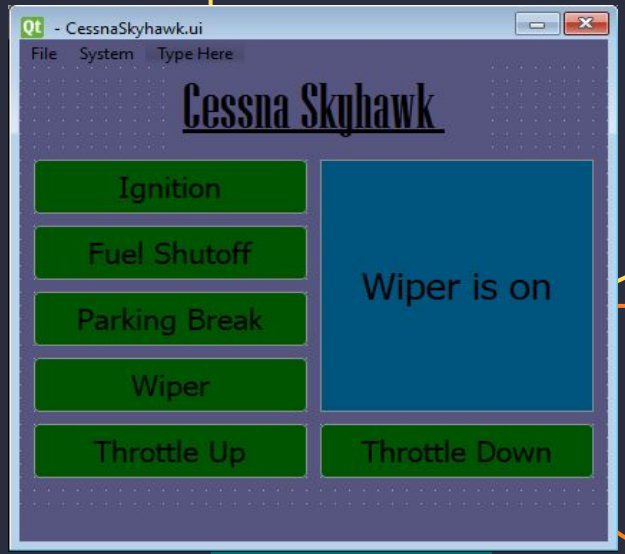
```
# XPlaneLogic.py
from UDP import UDP_backbone
import time
from PyQt5.QtWidgets import QApplication
import threading



class Logic:
    def __init__(self, ui):
        self.ui = ui
        self.start_state = False

        self.xp = UDP_backbone.XPlaneUdp()   # Deine
X-Plane-Verbindung

    def toggle_parking_brake(self):
        if self.parking_brake_on:
            self.ui.set_label_text("Parking Brake Off")

self.xp.WriteDataRef("sim/cockpit2/controls/parking_brake_ratio",
0.0)  # parking brake lösen
        else:
            self.ui.set_label_text("Parking Brake Set")

self.xp.WriteDataRef("sim/cockpit2/controls/parking_brake_ratio",
1.0)  # parking brake setzen

        self.parking_brake_on = not self.parking_brake_on

    def StartingSeq(self):
```

1.0 First Project

First Project Results





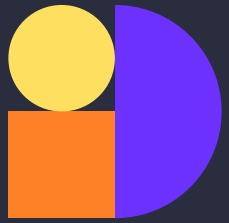


1.0 First Project

What we learned

X Plane Frontend

- How to fly all of the planes
- All of the functions in the planes
- Everything about the simulator



X Plane Backend

- How to access, read, and debug datarefs

User Interface Design

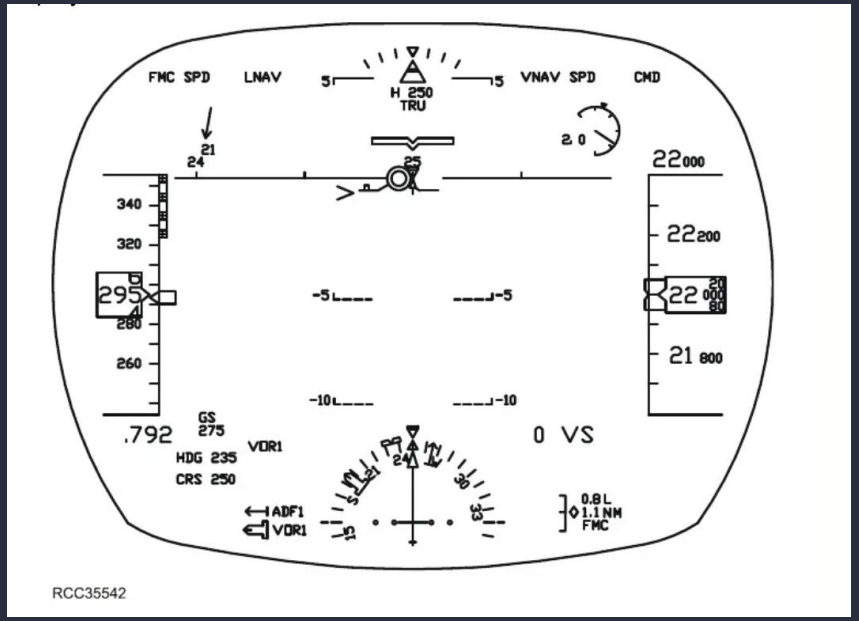
- QT Designer: creating menus, adding functions, connecting them to xplane.

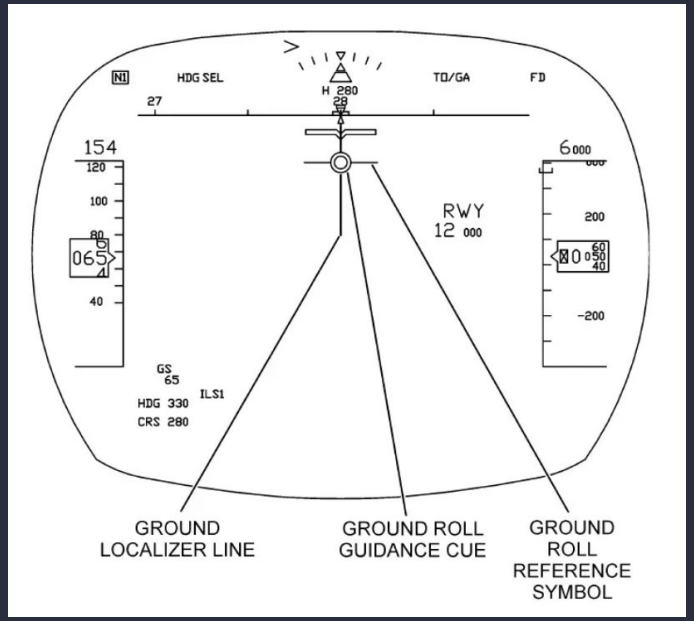
2.0 Initial Research

- After first project, given current assignment
- Tasked with creating a heads-up display for the single pilot simulator
- We began by looking at current HUDs in today's generation of planes.

2.0 Initial Research

Rockwell Collins HGS-6000





- Take off mode
- Landing Mode
- Low visibility Mode

2.0 Initial Research

Bombardier global 7500 HUD

- Taken style for inspiration
- Green highlights
- Gradient of colors

2.0 Initial Research

Ideas!

Essential Flight Data

- Adaptive Airspeed Indicator
- Enhanced Artificial Horizon with Dynamic Orientation Markers
- Intelligent Angle of Attack Display
- Real-time Wind Vector Display
- Predictive Altitude Trend Vector

Navigational & Situational Awareness

- AI-Predicted Flight Path with Hazard Avoidance
- **Proximity Traffic Visualization**
- **Dynamic Restricted Airspace Overlay**
- **Augmented Reality Waypoint Markers**
- Landing Assistance Markers

Pilot Support

- Real-Time Performance Metrics (G-Limits, Structural Load)
- Fatigue and Cognitive Load Monitor
- Autonomous System Override Prompts
- AI Copilot Suggestions
- Voice Command Confirmation Visualizer
- Pilot Biometrics & Health Status Display

Environmental Awareness & Communication

- **Dynamic Air Traffic Control Data Feed**
- Emergency Landing Zone Highlight with AI Selection Logic
- Urban Area Density Overlay

2.0 Initial Research

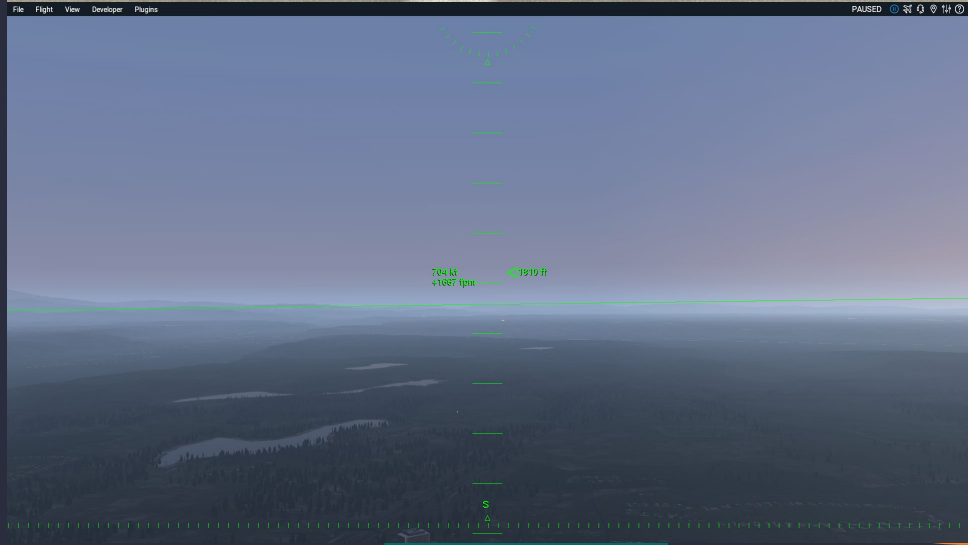
Software and trade-offs

Option	Tools / Languages	HUD Capability	Runway Alignment	Difficulty
1. C++ Plugin (using X-Plane SDK)	C++, OpenGL/Vulkan	✅ Full custom HUD	✅ Real runway alignment	🔥 Hardest (but most powerful)
2. FlyWithLua	Lua scripting	✅ Basic HUD overlays	❌ No true alignment (no 3D projection)	🟢 Easiest
3. XPPython3 Plugin	Python 3 + SDK	✅ Similar to Lua	❌ Same limits as Lua	🟡 Easy
4. External App with Qt Designer	Qt + Python/C++ + UDP or DataRefTool	❌ Not in-sim (external window)	❌ Cannot overlay on runway	🔶 Medium (only useful for control panels)

2.0 Initial Research

FlyWithLua Solution

- 2D Heads Up Display
- Features:
  - Pitch Indicator + Roll Indicator + Compass
  - Vertical Speed and Airspeed
  - Synthetic Horizon
- Limited by 2D, creative capabilities



2.0 Initial Research

C++ X Plane SDK

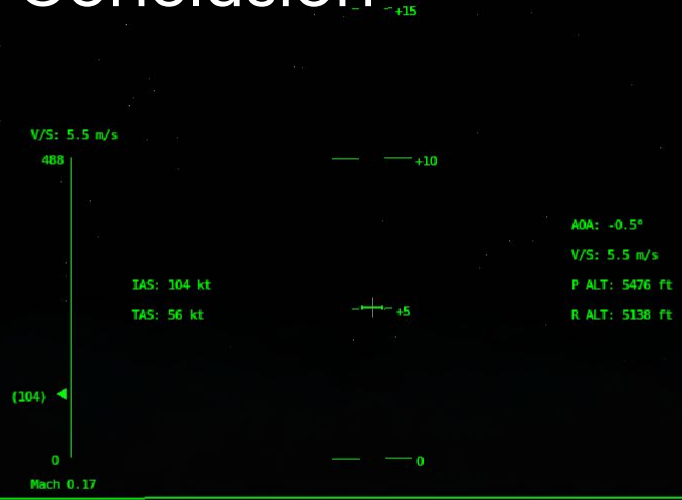
The C++ X Plane SDK was very complicated to set up, but once running, it had endless capabilities for creating the HUD.

Zone Load: No zone load attempted
Custom Waypoints: 0
Waypoint Load: No load attempted

2.0 Initial Research

Initial Research - Conclusion

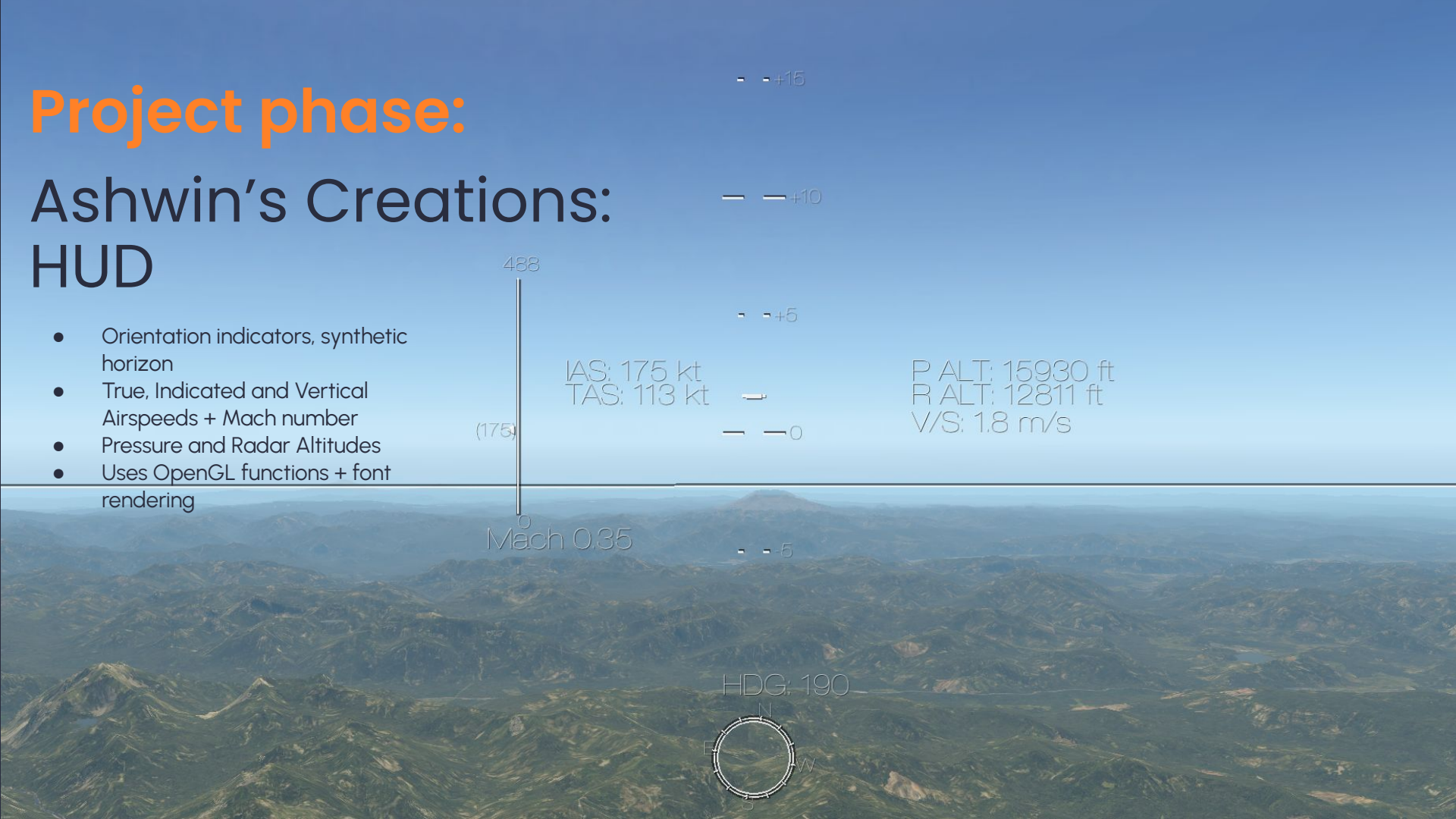
After conducting the initial research, we decided to go with the C++ one



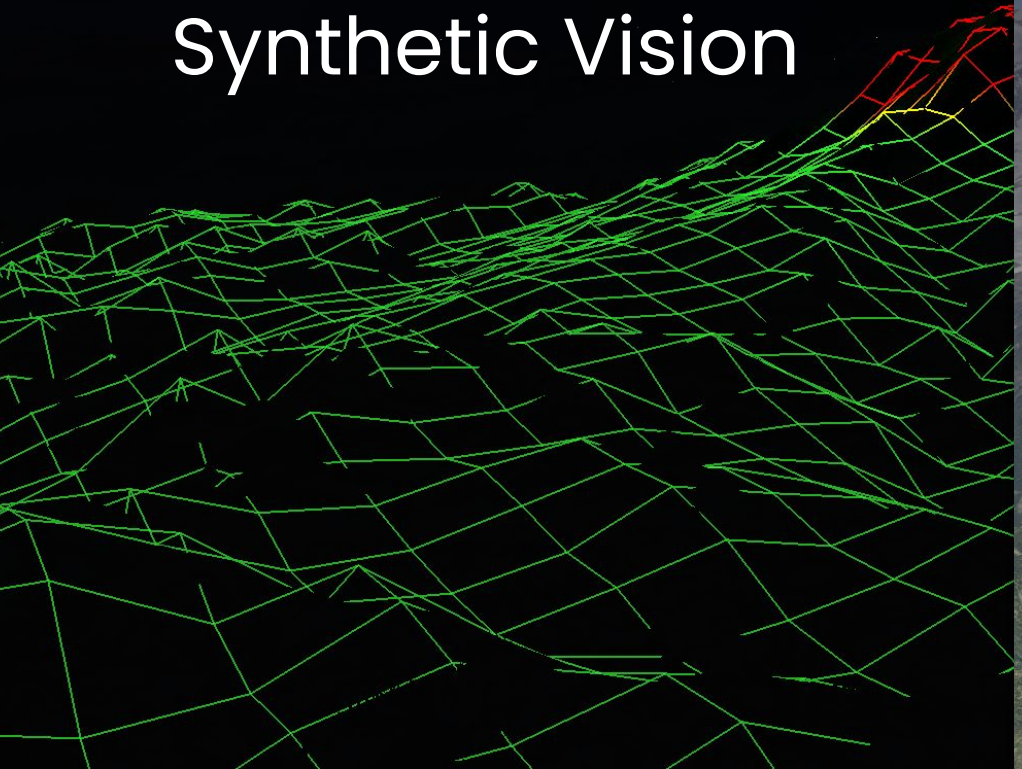
Project phase:

Ashwin's Creations: HUD

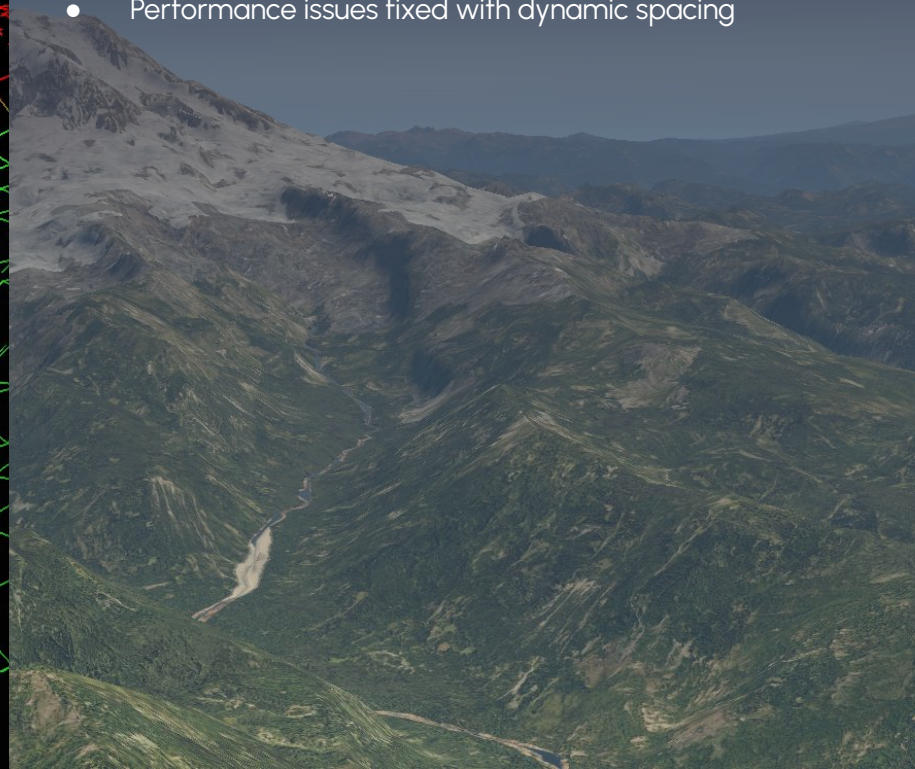
- Orientation indicators, synthetic horizon
- True, Indicated and Vertical Airspeeds + Mach number
- Pressure and Radar Altitudes
- Uses OpenGL functions + font rendering



3.0 Project phase:
Ashwin's Creations: Synthetic Vision



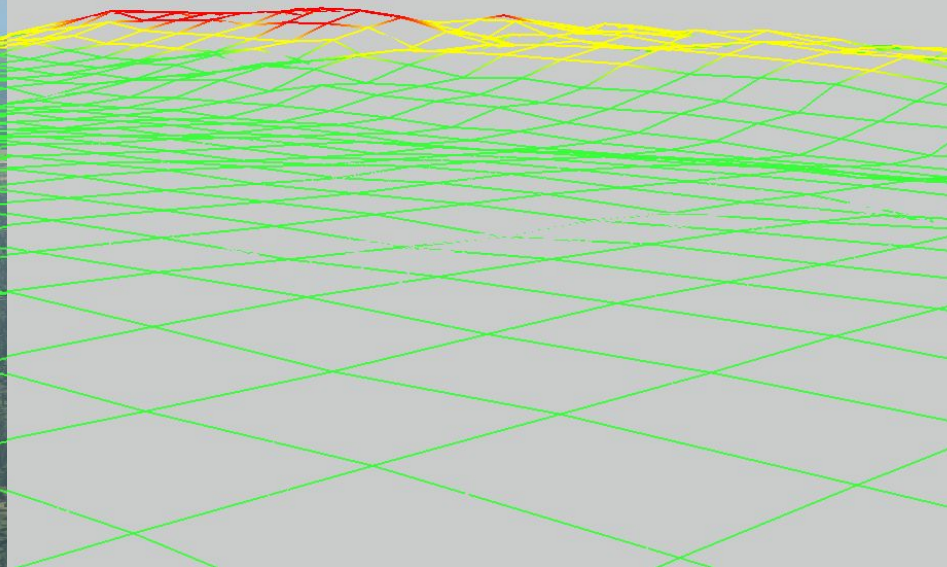
- Gives an accurate visual of terrain regardless of time of day or weather (Night time shown)
- Possible due to XPlane's XPLMProbeTerrainXYZ function
- Open GL drawing - Visible at any time of day or during poor weather
- Matched to aircraft position
- Performance issues fixed with dynamic spacing



Project phase:

Ashwin's Creations: Synthetic Vision

- Dynamic Altitude Based Color Coding
  - Above 1000 feet
  - Below 1000 feet
  - Below 500 feet
- Dynamic Vertical Offset
- Ultimately gives a clear vision of the surrounding terrain for a pilot



4.0 Project phase:

Joshua's Creations:

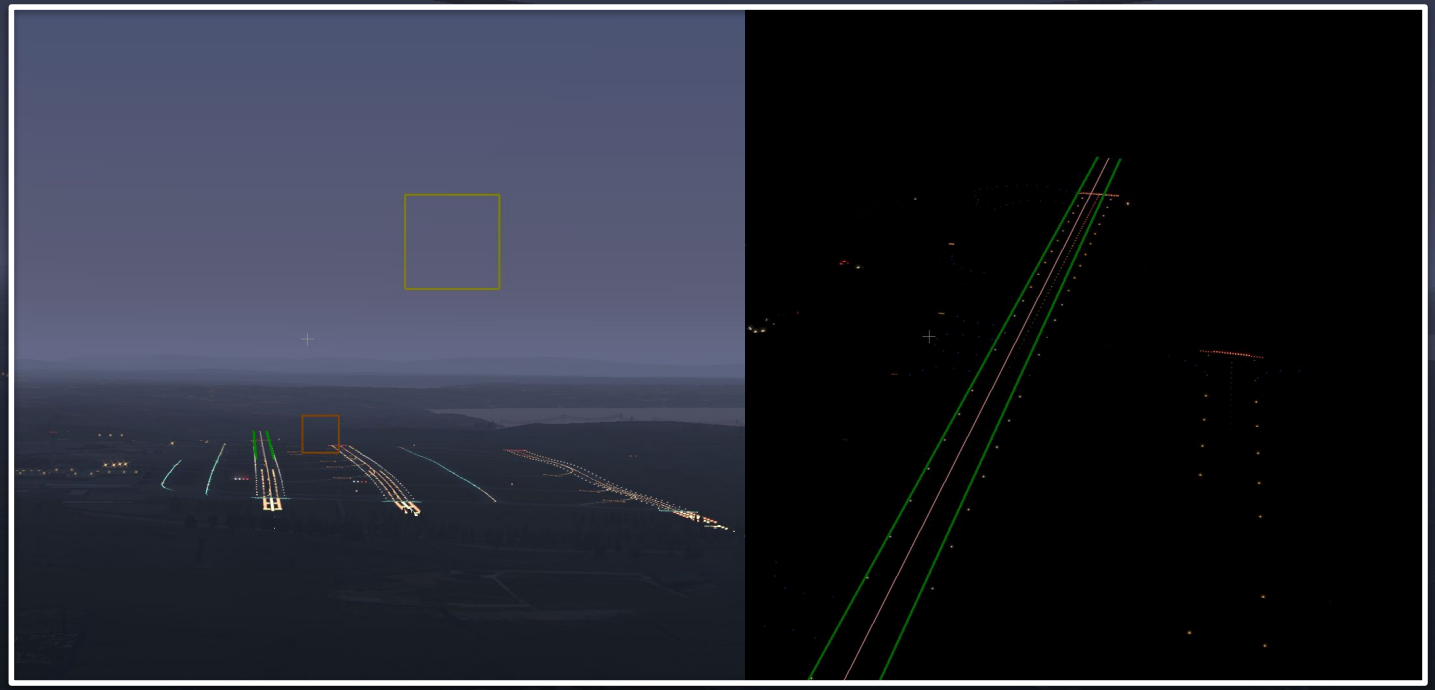
4.0 Project phase:

Joshua’s Creations: Landing Assist

4.0 Project phase:

Joshua's Creations: Landing Assist

- A 3D projection of the landing path
- Draws rectangles at different heights
- Draws outline around runway
    - Green
    - White



4.0 Project phase:

Joshua's Creations: Waypoints

4.0 Project phase:

Joshua's Creations: Waypoints

- Uses XPFlightPlanner
  - Creates a list of coordinates
- Draws rectangles and lines of the flight path
  - Blue: Climbing
  - Green: Cruising
  - Red: Descending
- Draws arrows indicating where the next waypoint is
- Tried adding text, it does not want to draw it
- Fully Customizable

4.0 Project phase:

Joshua’s Creations: Airspace Zone

- Draws a shape given n-coordinates
- Tried adding text
- Fully Customizable

4.0 Project phase:

Joshua's Creations: Airspace Zone

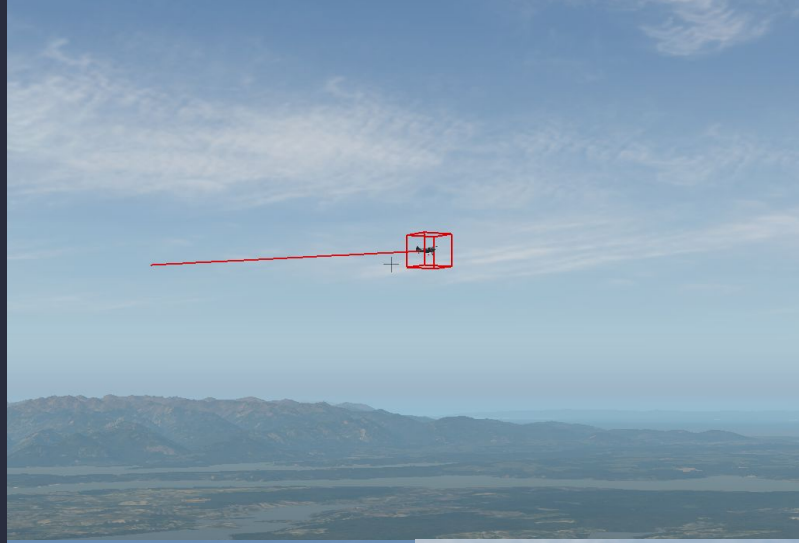
4.0 Project phase:

Joshua’s Creations: Aircraft highlight

4.0 Project phase:

Joshua's Creations: Aircraft Highlights

- Uses 3 different colors to indicate how close the aircraft is to the player's aircraft
  - Red < 1.0 KM
  - Orange 1 - 2 KM
  - Green > 2 KM
- Also draws a line behind the aircraft to show direction in low visibility conditions



Final Product

- Implement functions into single pilot simulator
- Relatively simple (plugin)
- Multiview issues fixed
- FOV Issue fixed

Thank you

Any questions? Ask away!