

TED UNIVERSITY/CMPE 492

Senior Project

High-Level Design Report

***Group Members***

*Arda ŞENBAKLAVACI*

*Gökhan İRTEM*

*Murat BEDİZ*

*Zeynep ALAPALA*

***Supervisor***

*Ahlaque AHMAD*

***Juries***

*Aslı GENÇTAV*

*Fırat AKBA*

Table Of Contents

1. Introduction**3**
   1. Object design trade-offs3
      1. Minimize Overall System Cost 3
      2. Increase Overall System Reliability 3
      3. Reduce Software Complexity 3
   2. Interface Documentation Guidelines 4
   3. Engineering Standards 4
   4. Definitions, Acronyms And Abbreviations 4
2. Packages**5**
3. Class Interfaces**5**
4. Glossary**6**
5. Referances7
6. INTRODUCTION

Each passing day, the sector of Gaming Industry grows way faster than it’s other entertainment competetors.Our Project Game’s name is War Trails. The project's objective is to develop a 3D Airplane Fighting game using the Unity engine and C# that is based on semi-realistic flight models of warplanes from the Second World War. Product will also be designed to be played against AI opponents.

**Purpose of The System:**

War Trails’s main purpose is to give the player a chance to play with their favorite planes which has realistic controls and physics against AI controlled opponents. Also the game will run at WebGL (secondary task).

**Design goals:**

War Trails will have basic environmental also weather effects will be present and basic landforms like cities, forests, lakes, mountains, deserts, and oceans will be included in the map design. The controls will be based on a realistic simulation of each individual plane from a third-person perspective, and some aspects will vary between warplanes due to their different models, performances, and capabilities. The War Trails will have a basic campaign that will lead the player through various battles throughout the war.The campaign's missions will be played against AI opponents who are capable of basic dogfighting and level flight.

For the enemy AI aircraft opponents that controlled by CPU, according to the location of the user’s aircraft, mimicking a short path to reach the User AI’s location (follow) like a realistic dogfight.

As the secondary task, the game will be represented as running on WebGL instead of direct output executable file.

* 1. OBJECT DESIGN TRADE-OFFS
     1. Minimize Overall System Cost

We used Unity as a free licensed program. Also we use free open source packages for our Project’s base implementation (Aircraft Physics).Also for AI implementation, we used Unity ML agent which is also free.

* + 1. Increase Overall System Reliability

Unity ML agents is a reliable system that used on many simple game projects that gives enough reliability after the correct implementation.

* + 1. Reduce Software Complexity

Using the ready 3rd party open source packages instead of our buildings,may reduce the complexity for avoiding over implementations and extra resource usage,imports etc…Our buildings can increase complexity…Also using Unity ML as AI Agent makes the system to have less extra 3rd party implementations may avoid complexity.(Unity ML is not directly included at Unity but it is installed using Packets).

* 1. INTERFACE DOCUMENTATION GUIDELINES

Our Project is a collecting the researched facts,sources (ready to use open source packages etc…) and implementing them as most usefull and gain less complexity as possible by using the Unity integrated AI (Unity ML) which is ready to use with Unity.

* 1. ENGINEERING STANDARDS

We used the accepted engineering standards.We designed the project with the UML Diagrams ,classes,sub components etc…Also we have showed the Hardware – Software relationships using theese diagrams at our Project.And finally started implementing the project according to theese diagrams that we designed.

* 1. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

Player: The user side interections.

Enemy: AI controlled opponents.

Map: The visiual background and the playground of the game.

Physics: Controlles based on internal and external effects such as weight ,center of mass, power of the engine, friction while landings,take offs etc.

Attack: Friendly AI/User engaging its enemy , using Vectoral Object Distances.

Damage: Affects that happens just after enemy/player hit its opponent with a shot or collusion , like penetration of armor of the plane and destroying the airplane’s components like wing etc.

Models: Plane patent/designs retrieved from other sources or made with Blender from designer itself.

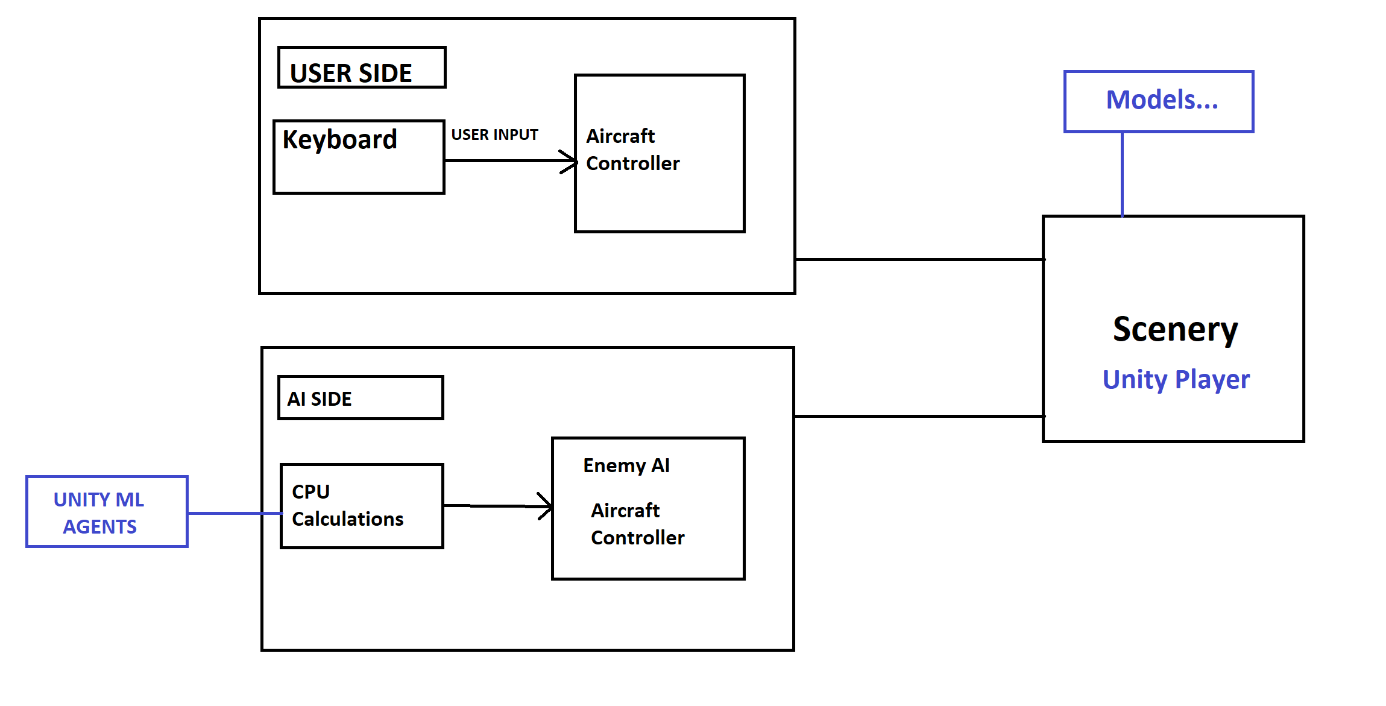
Campaign: Maps, storymode or background design of the events.

Path: The shortest way (route) to be followed, created by AI that enemy opponent will follow to reach the user aircraft’s location.

1. PACKAGES

Our base system (without the AI implementation) consists of 2 main packages , Airplane Tutorial which is a Physics Based Airplane Controller done at Unity that uses Aircraft-Physics package as the physics simulations.All of theese packages are implemented using C sharp ,compiled at Unity and has Open Source licence.

Now we are imported the Untiy ML package and started to implement the AI to the Enemy Aircraft’s controller as we showed at the Class Interfaces.



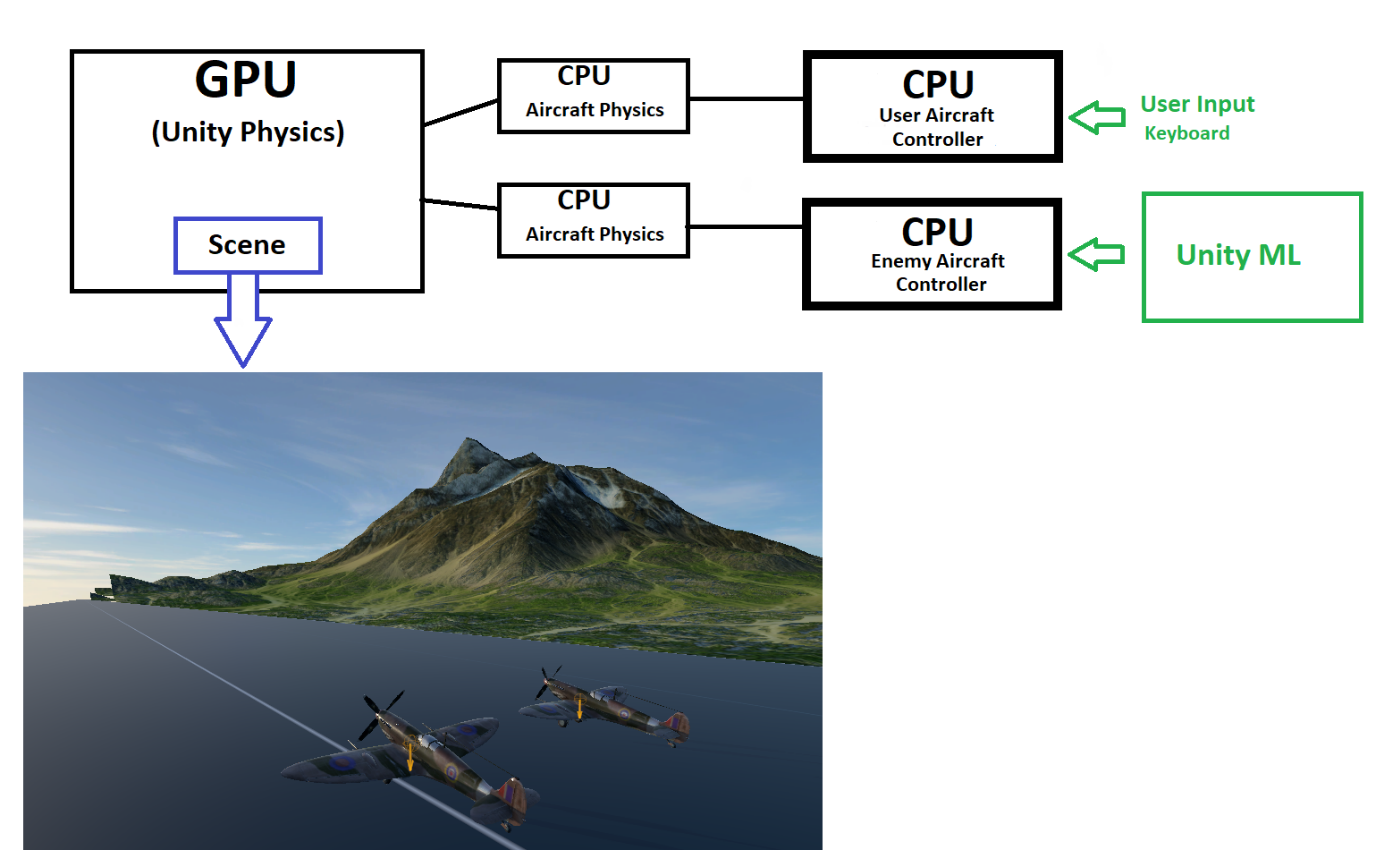
1. CLASS INTERFACES

Our base game package is a simple airplane game simulation that gets input from User’s Keyboard and has stable Physics etc.

User Aircraft Controller gets the input from User Keyboard and moves the plane according to the inputs.

Right now we are working on the implementation of Unity ML onto the Enemy Aircraft’s Controller that sets the movement parameters for the Aircraft Physics of the Enemy Aircraft.

Instead of getting input from user,enemy aircraft’s controller will get the input from the Unity ML as implemented (instead of keys, it will control aircraft’s yaw, pitch, roll etc. for movement controls). Unity ML will take the User Aircraft’s location vector as reference (as target). Also collusion sensors of Unity ML will avoid Enemy AI to collide any obstacle like Mountain etc.



1. GLOSSARY

AI:Artifical Intelligence; Our system that controls plane itself from the calculations that the PC side made taking the User’s location Vector as target.

Location: A place or something’s position (transform).

Vector: The physics part that represents the Model’s location, rotation of the current situation of the Model that lets us to have interactions with.

Controller: The script that controls the models by coding…

Roll: Rotation around the front-to-back axis.

Pitch: Rotation around the side-tp-side axis.

Yaw: Rotation around the vertical axis.

Obstacle: Any thing that blocks one’s way or prevents its movement.

Collusion: The forcefull coming together in direct contact of 2 things (object etc.).

1. REFERENCES

<https://unity.com/> • <https://www.blender.org/>

<https://www.youtube.com/watch?v=p3jDJ9FtTyM>

<https://www.youtube.com/watch?v=p3jDJ9FtTyM&t=134s&ab_channel=JumpTrajectory%22&>

<https://www.youtube.com/watch?v=p3jDJ9FtTyM&t=134s&ab_channel=JumpTrajectory%22t=134s>

<https://www.youtube.com/watch?v=p3jDJ9FtTyM&t=134s&ab_channel=JumpTrajectory%22&>

<https://www.youtube.com/watch?v=p3jDJ9FtTyM&t=134s&ab_channel=JumpTrajectory%22ab_channel=JumpTrajectory>

<https://github.com/Aki78/FlightAI>

<https://www.youtube.com/watch?v=UjkSFoLxesw>

<https://github.com/Aki78/FlightAI>

<https://www.youtube.com/watch?v=e91QA4WfL5Q&ab_channel=IcePops>

https://www.youtube.com/watch?v=p3jDJ9FtTyM&t=134s