Major Project Design

Izak Baldacchino a1830164
Hamoun Mohammadi a1716074
Tharane Thamodarar a1787411

https://github.com/HamounM1/Battle_Jets.git

Project Specification - Major Practical

Prop Fighter

Introduction

The project is a turned based fighting game where you, playing as a fighter jet, have to battle an enemy jet. The plan is to utilise abstract classes which are then inherited into child classes such as the enemy squadron and player squadron. The game is interactive, it's features include the player choosing the type of aircraft they would like to use in battle, access to the store to upgrade and the difficulty level of battle (the enemy pilot skill). The game is a single player turn based where each round has a few phases and the player initially chooses to commence battle. Each phase the player chooses an action in response to the outcome from the previous play. After each action a result will be displayed of their health and the outcome of the previous action. The player will be presented with a squadron of aircraft and their specifications. The winning conditions are the enemy retreats, severe damage to aircraft or death of pilots.

Design Description

Memory allocation from stack and the heap:

- Arrays: arrays will be used for upgrades, armour, REPAIRS(Maybe).
- Strings: names, boolean, status reports, attacks calls, upgrads.
- Objects: aircraft (enemy and player).

User Input and Output

- User input is used during different phases of the round.
- Choosing upgrades
- Choosing type of aircraft

Code Style

All code will be accessed via github. Either done on our personal laptop CS50IDE or university computers. The naming convention (constant variable will be written in all uppercase with an underscore) in this project will be in Apps Hungarian.
The abstract class is used to inherit information to the child classes where the aircraft specifications, damage control and squadron will be kept. Code comments will be used to briefly describe what each section of code does to reduce confusion and ambiguity among the group. A few additions might occur if time allows such as a store, credit score, visual and sound effects.

Class names:
Function names:
Variable names:

Schedule Plan

Goal by the end of week 8 is to come up with an idea, general outline of features and delegation of jobs in the group.

Week 8

- Topic idea
- Plan
- Features in the game
- Code sharing
- Google docs
- Design document

  Delegated Components
  - Class file : Izak
  - Game specifics: Tharane
  - Enemy behaviours: Hamoun

Week 9

- Basic skeleton of code
- Makefile
- Testing strategies
- Add more to design document if there are changes
- Add more into the code of the game
- Testing
- Get feedback
- Sort out SVN

  Delegated Components
  - Class file : Izak
  - Game specifics: Tharane
  - Enemy behaviours: Hamoun

Week 10

- Have a functioning code
- More testing
- Makefile edits
- Add more features
- Final touches
- Upload to SVN
- Get some feedback

  Delegated Components
  - Class file : Izak
  - Game specifics: Tharane
  - Enemy behaviours: Hamoun

Week 11:

- Have a final copy
- Present the project
- Final makefile
- Have everything uploaded before the practical

Testing

- Array list use dynamic memory
- bool function to add to array
- difficulty & assigns 'n' which is number of upgrades you can set (n on array)
- you can ~~reset~~ reset stats & have only/ credits & the shop be reset so you can upgrade difficulty.
- 'ready' function to start next battle
- shop

11/5/21

Aircraft struct.

Problems:

1 - Need to represent different types of aircraft

2 - Aircraft need to share common attributes.

3 - There needs to be a system in which two aircraft can interact with each other.

4; How to represent the health of the aircraft?

5 - what functions and behaviors will the aircraft have?

are the aircrgt going to simply just attack and maneuver?

Behaviours:

- attack : (use weapons)
- evade : (avoid an attack)

attack:
when the aircrafts are in position, one can choose to attack or evade based on what the enemy has done.

At the begging of the sortie both planes begin to climb. The plane with the higher climb rate will get higher and be able to attack first.

11/5/21

Concept 1:

The player choses an aircraft to fly, they begin a sortie with an enemy plane.

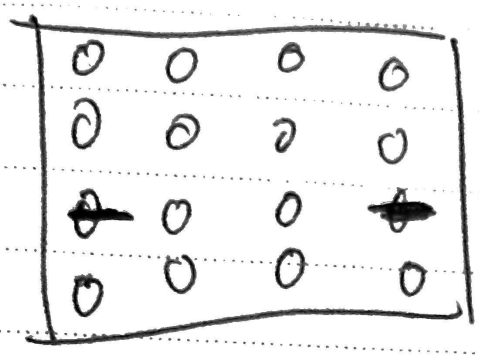
The battle has an initial maneuvering stage. This stage involves getting to a position relative to an enemy aircraft.

(Maybe represented by a simple grid)

The actions each plane can take will be either close distance or climb, ~~this can be limited~~ the player at any time can choose to attack, but the position relative to the enemy will affect the attack.

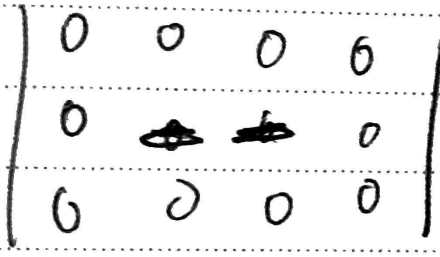
(range and alignment).

for example



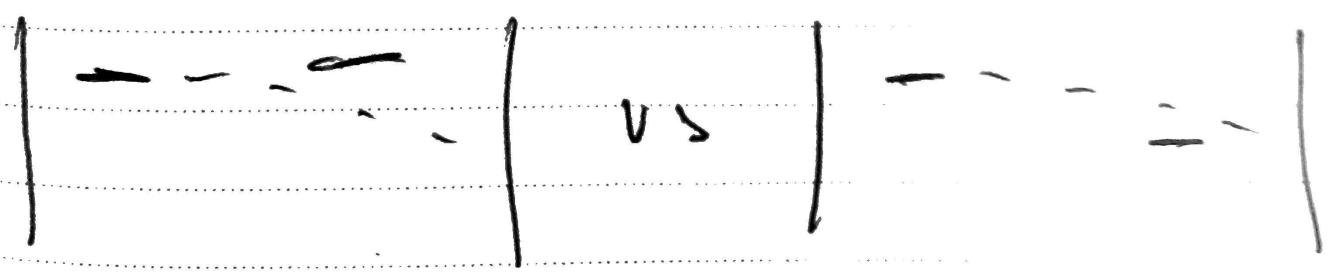
planes are alligned but few and ~~the shot will be less effective~~ but

the attack might succeed but the rest will expect.



The attack will almost certainly succeed.

~~The~~ planes on the same level can't shoot as few (bullet drop)
planes one above can shoot further).



13/8/21

|main.cpp| ← will contain the simplest amount of code.

|aircraft.h|
|aircraft.cpp| ← defines the abstract class for an aircraft.