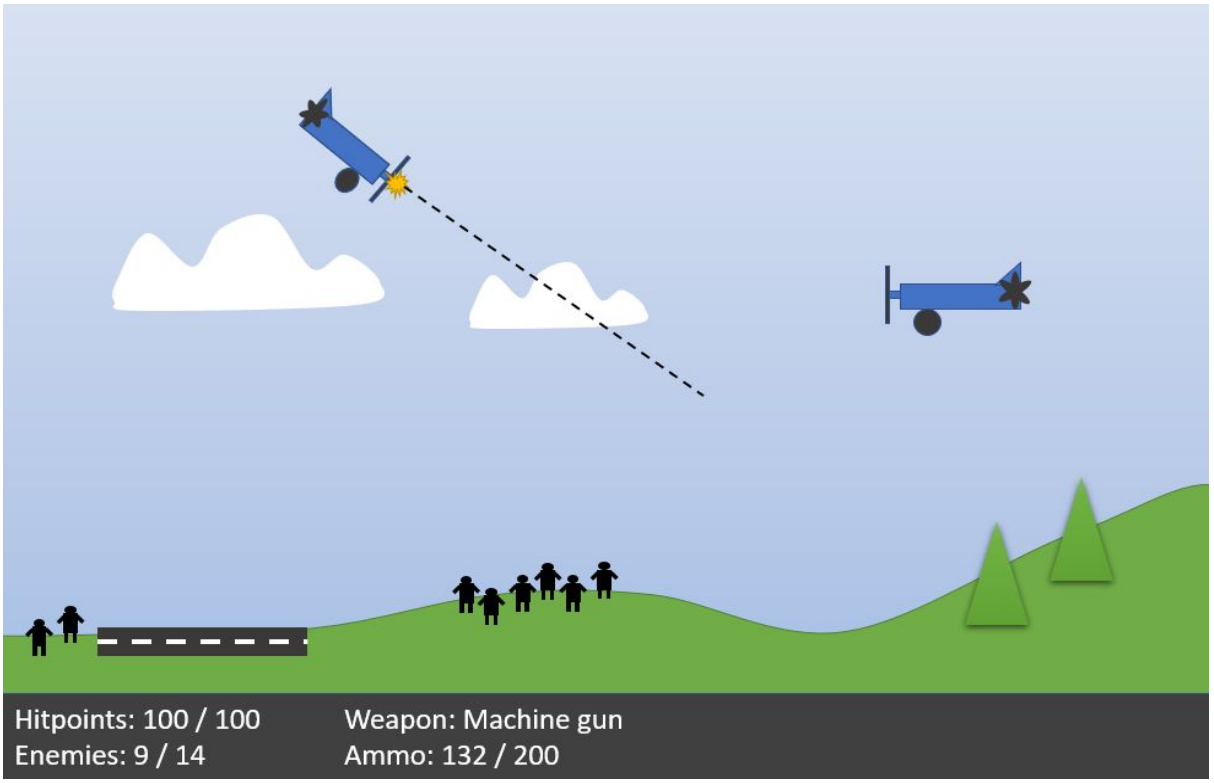
Air combat game - Project plan



Overview

Our goal is to create a 2D air combat game where the player tries to destroy the enemy troops. The game will have different kinds of troops, such as airplanes and infantry, which use weapons in order to shoot the player down from the sky. The player wins the game if they manage to destroy all the enemy planes before losing all their hit points.

We aim to implement the game with features extending beyond the basic requirements. In addition to a set of realistic features that we are sure to get done, we have some others that are planned in case the team has time and everything goes according to plan. As long as time permits it, the team is enthusiastic and aiming for an extensiveness score of 10.

Minimum planned features

- Simplistic 2D graphics
- Movement based on simple but actual physics
- Enemy troops (braindead infantry first, AA guns, AI-controlled airplanes later)
- Different weapons (upgrades etc.)
- Runways for landing (possible win condition by destroying enemy runway)
- Simple menu for starting the game. If there is enough time to implement the multiplayer mode, the menu would also contain options to choose between a single- or multiplayer mode.

Planned if time

- Multiplayer over network

Feature explanation

2D graphics and simple physics

The game will be based on 2D graphics and first we will implement one world or level which the user can play. The user will control a plane which is the main element using physics. The plane will fight against enemy troops.

Troops

The first implemented enemy will be infantry which will walk back and forth the world trying to shoot the player. The AA guns and AI-controlled airplanes will be implemented if we have time. Different kinds of planes have machine guns with different fire rates and some can drop bombs. Infantry will have weapons with significantly lower damage compared to fighter planes.

Runways

Fighter planes will use runways for takeoff and landing. Runways are team bases and possible win condition is to destroy all enemy runways.

Menu

The game will have a simple menu to start the game and replay. The menu can be extended include set up for the multiplayer mode and other additional features found important during the project for example selecting different features.

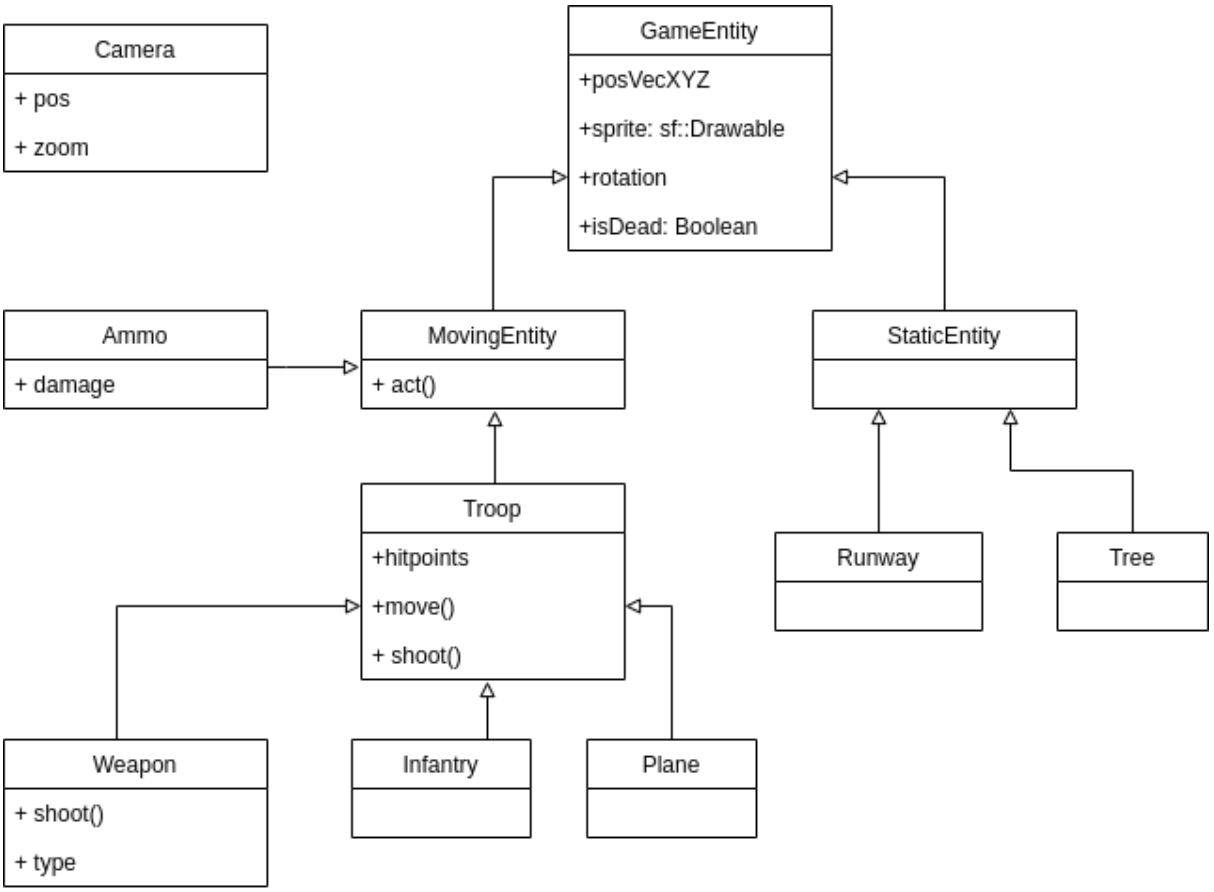
Multiplayer over network

The team is interested to implement multiplayer mode which would work over network but it is also considered to be laborious and therefore out of scope. Multiplayer mode would enable the players fight against each other and the planes would shoot each other.

Sound effects

The gameplay experience would be better with sound effects. However this is not nearly a top priority and thus may be out of scope.

Software structure



The software will make use of the SFML graphics and multimedia library for rendering graphics. We deemed external physics libraries unnecessary for the project as any needed physics calculations would be relatively sparse and simple to implement ourselves, mostly related to just the planes.

The high-level and very simplified class hierarchy for the main parts of the software is as follows:

The base class GameEntity represents any entity drawn on the screen. This includes static props etc. and moving objects, including AI-/player-controlled planes and simple things like projectiles and clouds. All game entities include an element for drawing in the form of an sf::Drawable class from the SFML library. In practice still will most likely be a shape of a texture.

All moving entities implement the void function act(float deltaTime), which advances their movements and state according to their specific internal logic/user controls. This logic may also spawn additional GameEntities (projectiles, particles).

Planes are represented by a single class for now, regardless of if they are AI- or user-controlled. The controls are planned to be defined by internal variables but this might change into a split with separate classes for each. We aim to get the movement for the planes to be as physically-based as feasible while retaining a sensible freedom of movement and feel for the controls in a 2D setting. Any further digging into specific mechanical calculations or algorithms doesn't make sense yet.

The camera class/struct is simply a container for the information on the current game view relative to which absolute world coordinates for GameEntities are transformed prior to drawing.

The game world is not represented with it's own class for now as it would mostly just be a container for the GameEntities and camera. This might get changed in the implementation phase if deemed necessary.

Project schedule

We are aiming for a good head start in the first weeks in order to get a good development setup and rhythm going as early as possible. The schedule will surely change as the project progresses, but we are aiming for the following schedule/order of implementation:

Week	Tasks / Goals
44	1.11. Project plan submitted to git
45	Getting to know with the SFML library. Implementing a skeleton for the program (create build setup, get libs installed and working, implement simple drawing loop) Implement game world and a general moving object (gameEntity, movingEntity).
46	Implement some user-controllable plane Troop class Firing mechanics
47	Mid-term meeting with advisor for questions and feedback Runway/landing Hitpoints Win condition HUD
48	26.11. mid-term peer-reviews delivered Menu Sound effects Multiplayer over network
49	Bug fixes Spare time for getting any unfinished work in order
50	9-13.12. Final demonstration meeting 13.12. Deadline for final git commit, completion of project.