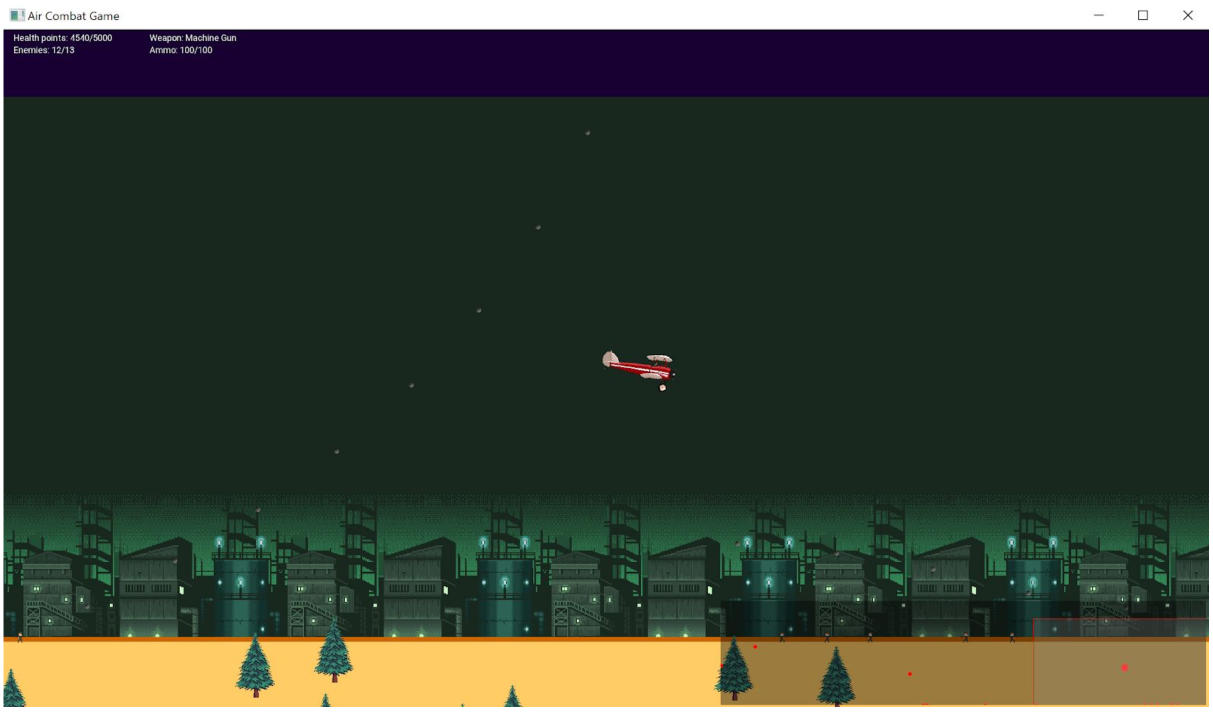
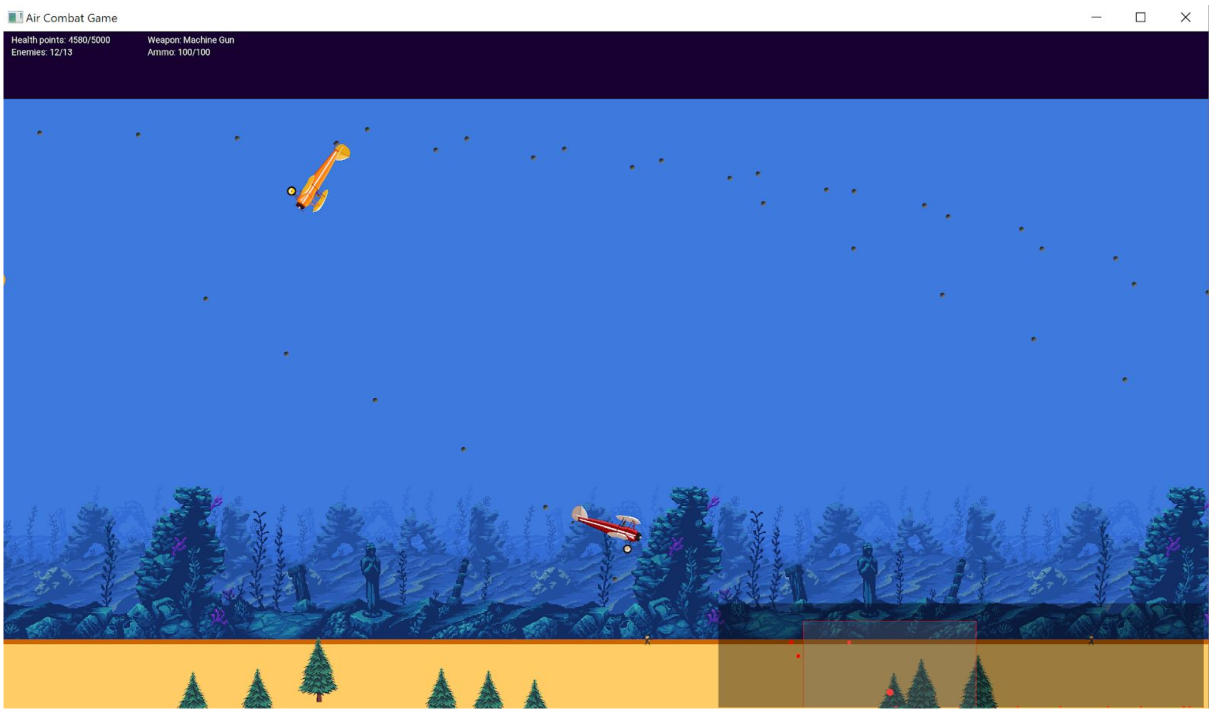
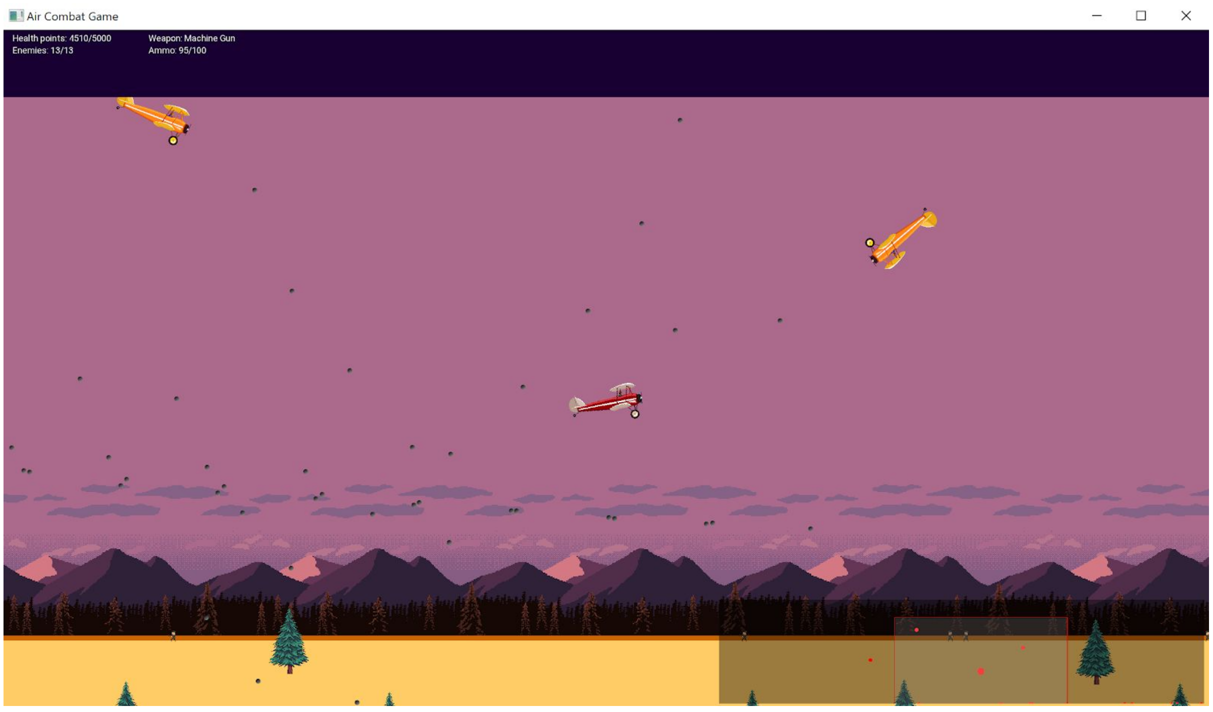
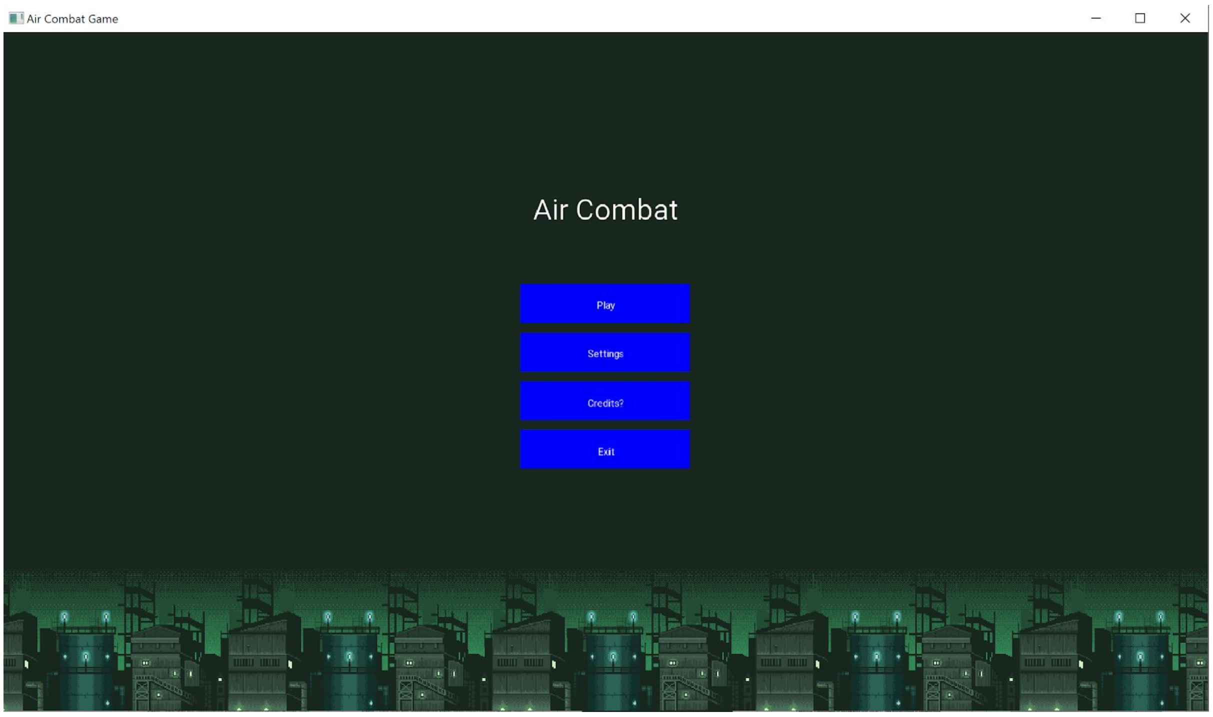
Air combat game - Project documentation






Air Combat Game
Health points: 4510/5000
Enemies: 13/13
Weapon: Machine Gun
Ammo: 95/100


Air Combat Game
Air Combat
Play
Settings
Credits?
Exit

Overview

Our project is a 2D air combat game with the goal of destroying all enemy troops. The game has two kinds of enemies: airplanes and infantry, which use their 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 without losing all their hit points or crashing to the ground.

The following basic features are implemented:
- 2D graphics
- Simple physics
- Enemy troops (infantry, AI enemy planes (advanced))
- Player plane

Additional features:
- Menu
- HUD
- Possibility to land on the ground
- Edges
- The enemy planes have better AI movements
- Advanced graphics
- Minimap
- Switchable parallax backgrounds
- Animated explosions
- Framerate-independent game logic
- Resizable window
- Game view zoom
- Multi-platform build commands

Feature explanation

2D graphics and simple physics
The game is based on 2D graphics and we implemented one level that the user can play. However, the user can customize the level by choosing a background from three different alternatives. The user controls the plane which is the main element using physics. The plane will fight against enemy troops.

Troops
The game has infantry troops which walk back and forth the world trying to shoot the player. There are also AI enemy planes which are heading to a random point or trying to estimate the player’s movement. Due to the AI movement, the enemies will move differently during each instance of the game.

Menu
The game has a menu with hoverable buttons. The buttons are used to start a new game, check controls or exit the game.

HUD
HUD at the bottom of the game window displays the player's hit points, a number of projectiles, enemies left and the weapon type.

Edges
The battlefield has edges on all sides not allowing the units to go through. The camera stops right before an edge so that the player will stop at the edge of the game window.

Landing
The planes can land on the ground if the landing angle is small enough. The plane is destroyed and the game ends if the angle is too big or the plane is upside down.

Minimap
The minimap displays enemies, projectiles, explosions and the player relative to their position in the battlefield.

Graphics
The game has beautiful graphics, including animated explosion effects and moving parallax background. Enemy planes explode on death, creating an animated explosion. Background can be changed in game to mountain, industrial or underwater theme. Additionally, most sprites can be or are flipped during the game, as an example, infantry are facing towards the direction they are moving.

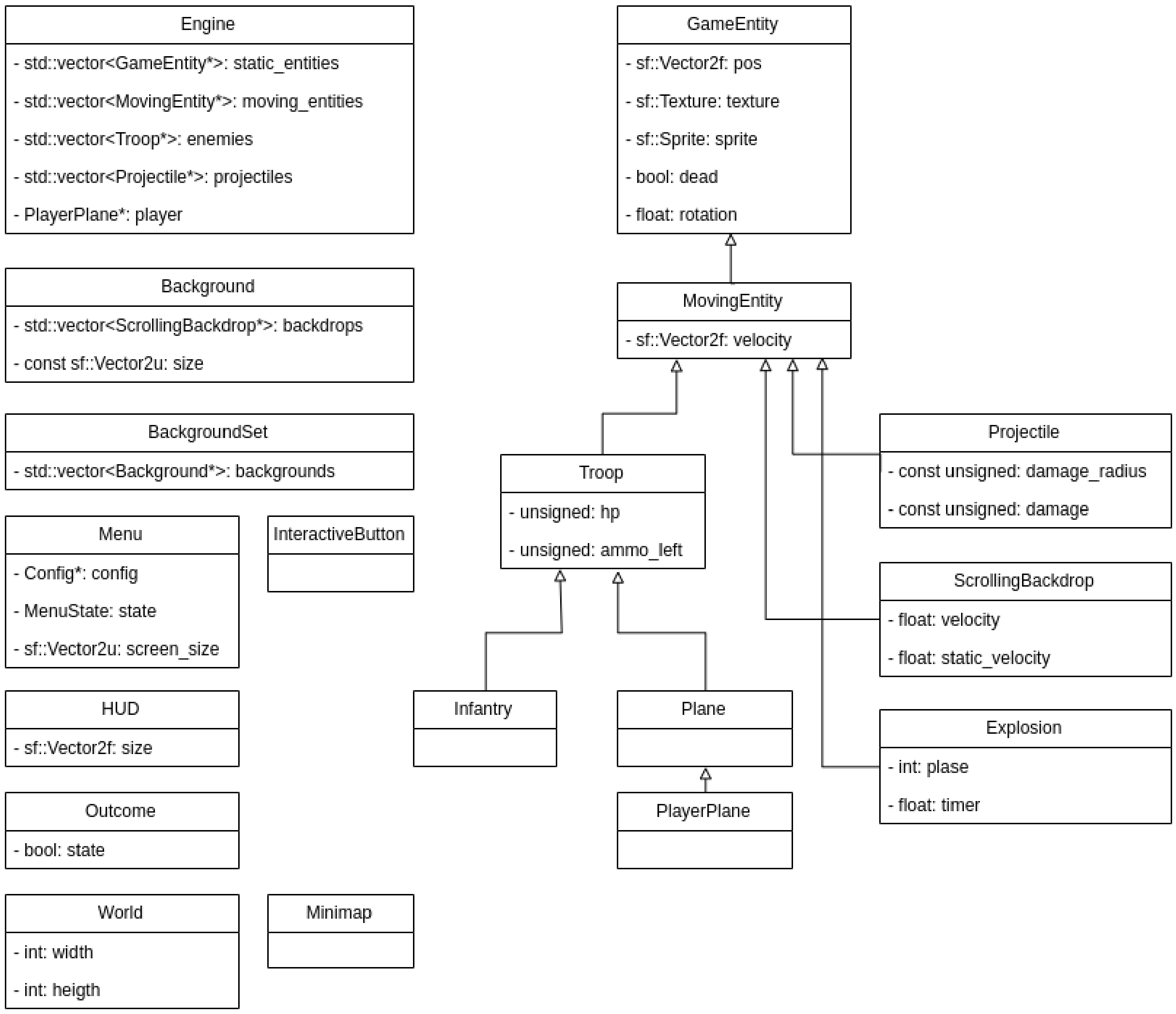
Framerate-independent game logic
Engine and entity logics are relative to time instead of the framerate. This ensures similar experience for every device.

Resizable window
The window can be resized which updates the minimap as well relative to the window size and shape.

Zoom
The view can be zoomed in and out updating the minimap relative to the view.

Multi-platform build
The software has build commands for multiple platforms. This allowed developing on WSL. Additional information for the builds in README.md

Software structure



The software uses 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:

Class Engine maintains the game and its game entities. The engine is used to start the game and handle the refresh, draw and input loops.

The base class GameEntity represents any entity drawn on the screen. GameEntity includes both static and moving objects. All GameEntities exploits the SFML library in order to draw elements.

Moving entities have a function act that enables movement and other advanced features. Each moving entity can have specific internal logic and state. MovingEntity includes a class Troop that is a base class for planes and infantry.

The game contains different planes which can be AI-controlled enemy planes or user-controlled player-plane. The enemy planes are instances of the Plane class. The player's plane is initialized from the PlayerPlane class which extends the Plane class. The PlayerPlane class overrides the act method so that the user can control the plane with the keyboard. The controls' states are saved to a Keys struct. We aimed to get the movement for the planes to be as physically-based as feasible while retaining sensible freedom of movement and feel for the controls in a 2D setting. Any further digging into specific mechanical calculations or algorithms didn't make sense so far.

The game and menu states are saved to enums which control what is drawn on the window. The class menu contains a selection to play the game, look at the controls or quit. The credits page was not implemented.

Instructions

Instructions for building and using the software

- Refer to README.md included in the source repo

How to use the software: a basic user guide

User plane controls:

arrow-left	rotate counterclockwise
arrow-right	rotate clockwise
arrow-up	flip the plane
arrow-down	turn on/off the engine
d	shoot

Other commands:

b	change the background
ESC	close the window
Scrollwheel	zoom in/out

The window can be resized

Testing

No testing libraries were used, as the limited project scope did not warrant them. Testing was done alongside the development, mostly as user testing.

Debugging was performed mostly using simple standard output calls and in tricky situations with valgrind. Memory leaks were tested using valgrind/memcheck.

Work log

‘Planned’ column contains the respective fields from the original Project plan, while ‘Completed’ shows what was accomplished.

Week	Planned	Completed	Estimated work hours
44	Project plan	Project plan - Everyone A skeleton for the program (create build setup, get libs installed and working, implement simple drawing loop) - Atte	Atte: 8 Johannes: 3 Matti: 3 Yvonne: 3
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).	Getting to know with the SFML library - Everyone Implementing a skeleton for the program - Already complete Engine class started - Matti MovingEntity class - Atte Troop class - Atte Infantry class - Atte Plane class - Atte Numerous lib/build fixes - Atte Key controls - Johannes, Yvonne Fullscreen mode - Matti Initial event handling - Matti	Atte: 10 Johannes: 5 Matti: 5 Yvonne: 5
46	Implement some user-controllable plane Troop Firing mechanics	Implement some user-controllable plane - Johannes Troop Firing mechanics - Johannes, Yvonne Global variables for resources - Atte Scalable viewport - Atte Proper entity transforms - Atte Entity drawing - Atte Parallax background - Atte Projectile class - Matti	Atte: 14 Johannes: 12 Matti: 6 Yvonne: 5
47	Runway/landing Hitpoints Win condition HUD	Runway/landing - Not done Hitpoints - Not done Win condition - Not done Firing mechanics - Johannes Rework projectiles - Johannes, Matti Runway class - Yvonne Rework plane movement - Atte Switchable multibackgrounds - Atte Rework infantry movement - Yvonne	Atte: 6 Johannes: 15 Matti: 10 Yvonne: 10

		HUD - Yvonne	
48	Menu Sound effects Multiplayer over network	Menu - Atte Sound effects - Left out Multiplayer over network - Left out Global font resource loading - Atte Enemy planes - Matti Infantry projectiles shooting - Yvonne Projectiles rework - Matti	Atte: 8 Johannes: 0 Matti: 12 Yvonne: 5
49	Bug fixes Spare time for getting any unfinished work in order	Dead enemy removal - Yvonne Infantry adjustments - Yvonne Game outcomes - Yvonne World class - Johannes Plane crashing - Johannes	Atte: 0 Johannes: 5 Matti: 4 Yvonne: 10
50	9-13.12. Final demonstration meeting 13.12. Deadline for final git commit, completion of the project.	Settings menu page - Yvonne World borders - Johannes Teams - Johannes Infantry refinement - Johannes Enemy plane AI - Matti Minimap - Atte Ground trees - Atte Camera world lock - Atte Camera zoom - Atte Landing - Yvonne Animated explosions - Johannes Replay - Yvonne	Atte: 8 Johannes:16 Matti: 8 Yvonne: 16