

# **DOCUMENTATION FOR THE GAME “BOOM! STOP THE NYSSE”**

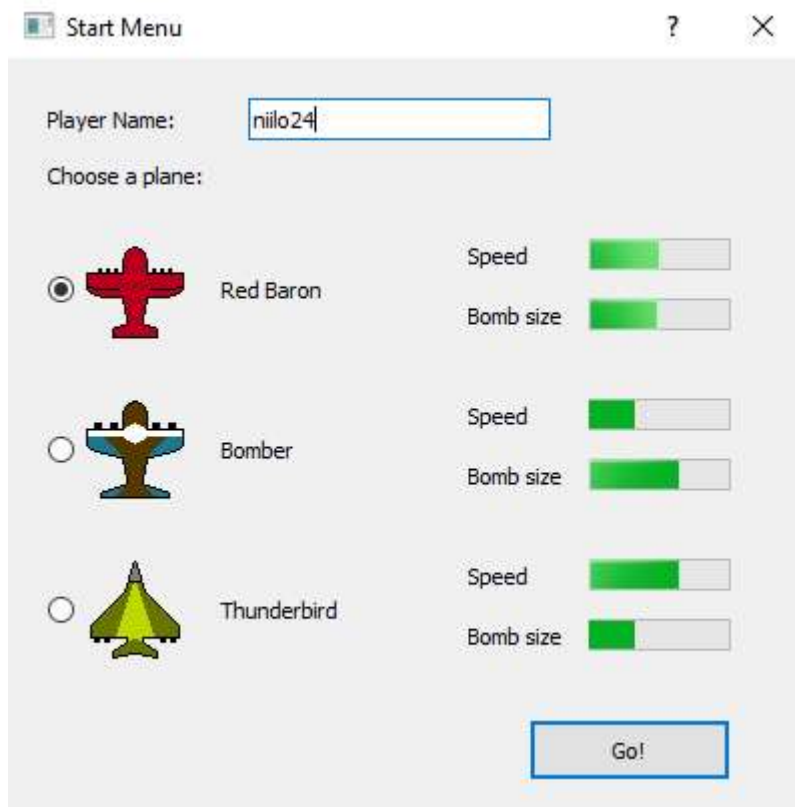
This document contains instructions for playing the game and description of the student side program logic. Also, the workload of both team members is listed alongside with extra features and bugs.

# 1. GAME INSTRUCTIONS

The goal of the player is to destroy as many nyssees as they can. The player controls a plane capable of dropping bombs. In addition to the nyssees, there are also clouds moving across the game area that the player must avoid. Colliding with a cloud damages the plane and decreases score by 5, and after 4 hits the plane is destroyed. After collision, there is a short cooldown phase (different for each plane) during which the plane will not take damage from collisions. The game goes on until the game time runs out or until the plane is destroyed.

## 1.1 Starting the game

At the start of the game, a start menu opens up where player must enter their name and choose a plane. There are three different planes available, each of which have different capabilities.



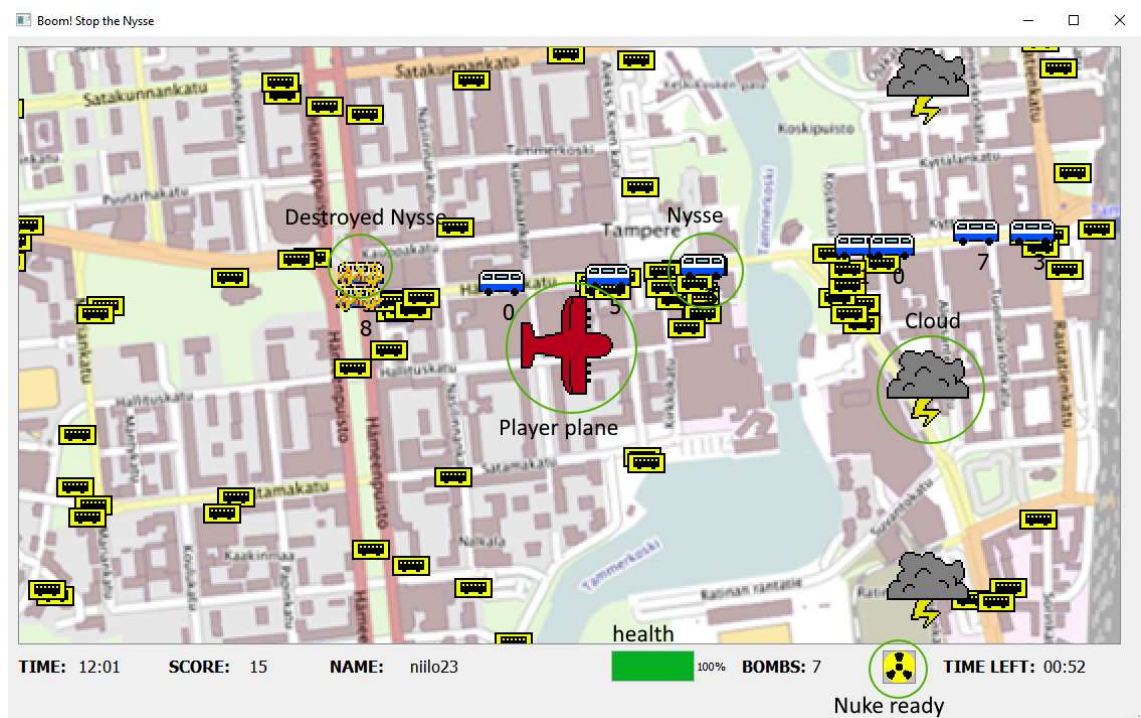
The screenshot shows a 'Start Menu' window with a title bar containing a question mark and a close button. Inside the window, there is a text input field for 'Player Name' containing the text 'niilo24'. Below this is a section titled 'Choose a plane:' with three options, each consisting of a radio button, a plane icon, and a name. The 'Red Baron' option is selected. To the right of each plane name are two sliders for 'Speed' and 'Bomb size'. At the bottom right is a 'Go!' button.

Plane	Speed	Bomb size
Red Baron	High	High
Bomber	Medium	Medium
Thunderbird	Low	Low

*Start menu*

## 1.2 Playing the game

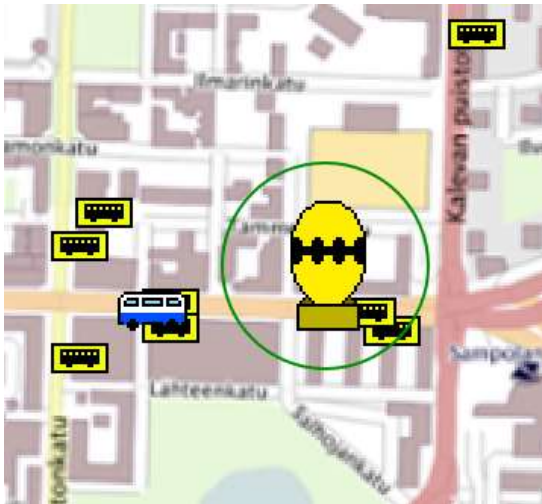
The game round begins, when player clicks the "Go!"-button. The direction of the plane is controlled with WASD keys and bombs are dropped using SPACEBAR. Player gains one (1) point for every nysse destroyed and one (1) point for every passenger inside the destroyed nysse.



*Game view*

## 1.3 Special weapon

During the game, the player may find a nuke that spawns within the game area in a random location. The player may collect the nuke, after which they can drop the nuke using R key. The nuke is much more powerful than a regular bomb and thus has area of effect. The player may only have one nuke collected at a time. New nuke spawns after the previous one has been collected and dropped. If the player has a nuke ready to be dropped, it is indicated with a symbol in the lower right corner of the MainWindow.



*Nuke ready to be collected*

## 1.4 End of the game

The game ends, when the game time runs out or when the plane is destroyed after hitting clouds too many times. After the game ends, game round statistics are displayed in the end dialog, along with all time TOP5 scores of the game.

## 2. GAME LOGIC & WORKLOAD DISTRIBUTION

The course side template of the project contains the logic for moving the nysses and passengers, as well as some interfaces. During the coding of the game, the most important interface was the `CourseSide::ICity`, from which the `City` class is inherited. The `City` class moves the actors in the city by receiving function calls from the `CourseSide::Logic` object. The `City` also calls the actor functions in the `MainWindow`, which controls the actor graphics. Other than the `City` class, the student side classes were inherited from Qt Classes.

The `Engine` class is used for initializing the game logic and the city. The `Engine` class also creates the Start menu. After initialization, the course side logic sends function calls to the city object, which contains the `MainWindow`.

The actors used by the student side of the program are inherited from `QGraphicsItem`, which contains methods for setting/changing coordinates and drawing images on the graphics scene of the `MainWindow`, as well as checking collisions. The actors are created stored in the `MainWindow`, with the exception being the `Player` object, which is created in the `City` object.

### 2.1 Statistics unit testing

Unit testing class `TestStatistics` is run from the main function. It creates a new `Statistics` object and runs unit tests for its variable setting functions. `TestStatistics` achieves this by using `QCOMPARE()` defined in `Qtest`.

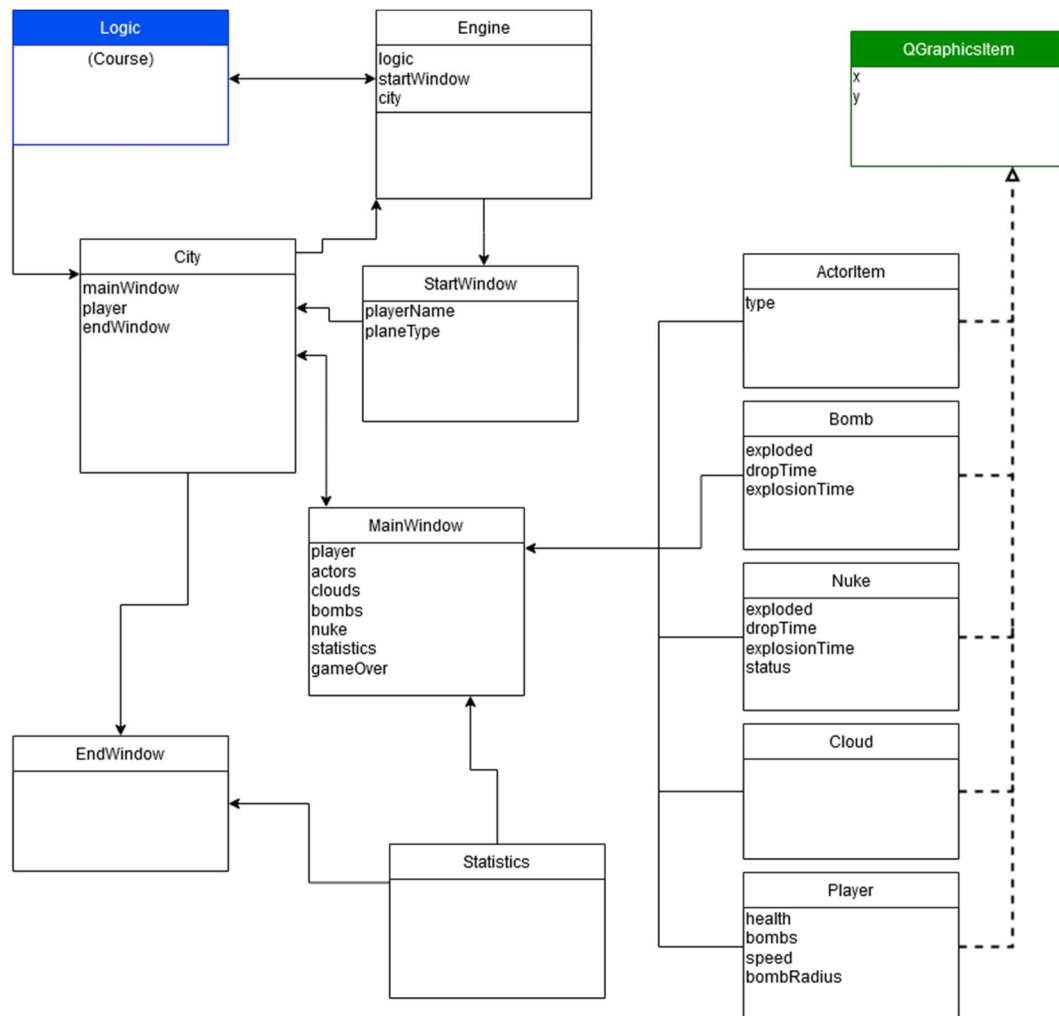
### 2.2 Known bugs

Sometimes a cloud in a vertical set of clouds is placed a few pixels too far to the left.

The nuke explosion graphics are sometimes cut off by a little at the edges.

### 2.3 Class diagram

The following diagram contains the most important student side classes of the program and the essential data stored in the classes. The interaction between classes is described with arrows. Blue color indicates course side, no color indicates student side and green color indicates Qt.



During the game round, the **MainWindow** controls most of the game actions. **MainWindow** stores all the actors in the game and controls their movement and drawing graphics.

## 2.4 Additional features

Below is a list of additional features added to the game, as described in project info ([https://plus.tuni.fi/tie-0240x/fall-2020/modules\\_00/00\\_tyoohej/](https://plus.tuni.fi/tie-0240x/fall-2020/modules_00/00_tyoohej/))

- Even screen updates (The items on the graphics scene move automatically every tick)
- Scrollable map (The game view moves along and is centered on the plane)
- Even movement of the player figure (Plane moves automatically and smoothly to the current direction, the player only controls the direction and bombing)
- Passenger amounts (Passenger count of each nysse is shown as a number next to them)

- Following the game state (Player can see their current score, time left, bombs left, nuke and health)
- Top10-list (After the game round ends, all-time best results are shown next to the game round statistics)
- Updates to the playable figure (Player can collect a powerup (Nuke))
- Own feature: health.
- Own feature: sounds (Won't work on remote desktop)

## 2.5 Workload distribution

Below, the workload of both group members is roughly listed. However, it should be noted that many classes contain changes made by both members.

Väinö Kahala

- Graphics
- Sound
- Movement of actors
- Player plane & controls
- Bombs
- Nuke
- Clouds
- MainWindow

Elias Halkola

- Start dialog
- End dialog
- Statistics
- Statistics unit tests
- Saving the data (hiscores)