System design document for ”Nollans första dag”

Alexander Brunnegård, Julia Böckert, Malte Carlstedt
Steffanie Kristiansson, Clara Simonsson

7 oktober 2021
version 4

1 Introduction

This document describes the background and the construction of the application. The application is described from a perspective of a developer and includes terms and proceedings to come up to the final result, the game.

The game is called ”Nollans första dag” and it is about completing different tasks to come up to the main goal, create the ”Nollbricka”. The user controls a player and can make it walk through Campus, which is built up by different maps, and complete the tasks.

1.1 Definitions, acronyms, and abbreviations

1.1.1 Slick2D

A 2D Java Game Library, set of tools to make development of Java games easier

1.1.2 Non-Playable Character (NPC)

An idle character

2 System architecture

2.1 Overview

2.2 Software decomposition

The game consists of four packages which are model, view, controller and tasks. The model, view and controller is the three components which are included in the MVC pattern.

The model package handles the game logic and has seperated models for different components. It consist of a playermodel, mapmodel and all the different mapstates. All the logic about collisions with objects are also in the model package.

The view and controller has two classes, one for the player and one for the map.

The task package has a model, view and controller as well and there are seperated MVC:s for each task. This structure makes it easier to expand and add more task because they´re not dependening on each other.

There are six different classes in the application which doesn´t belongs to a package. These are EnterTask, GameMenu, HelpViewMenu, MainGame and StateSetup. The EnterTask does as the name describes. It checks if the criterias for entering a task is achieved and starts it.
The Gamemenu and helpmenu

2.3 Application flow

When the application is started the StateSetup class adds all states which will be used in the game. The state will always be zero at the beginning because it is connected to the game menu where the user can choose if it wants to start the game or go to the help.

When the actual game is started the application will enter another state.

3 System design

3.1 UML diagram

The figures below show two UML diagrams of the packages, and the task package specifically.

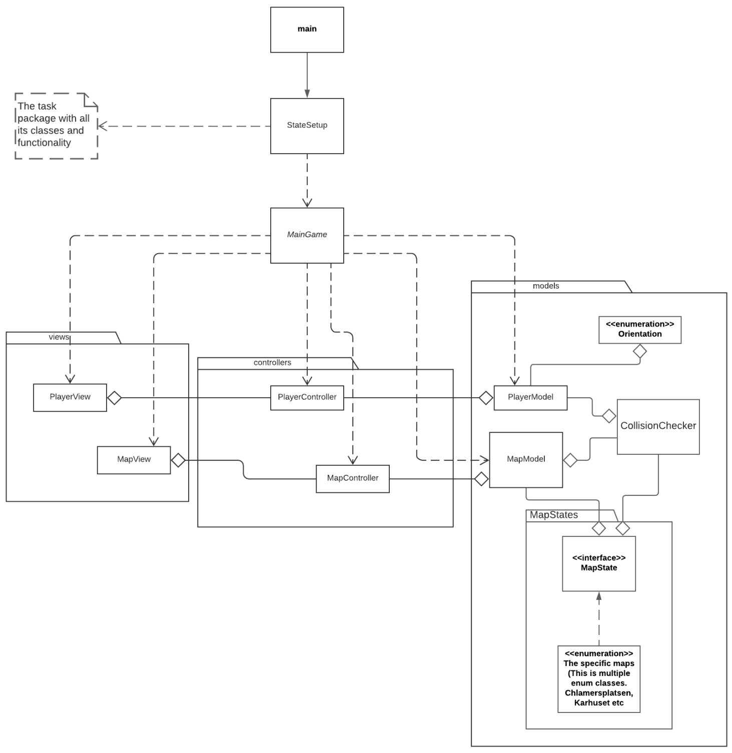
main
StateSetup
The task package with all its classes and functionality
MainGame
views
PlayerView
MapView
controllers
PlayerController
MapController
models
<<enumeration>> Orientation
PlayerModel
CollisionChecker
MapModel
MapStates
<<interface>> MapState
<<enumeration>> The specific maps (This is multiple enum classes. Chlamersplatsen, Karhuset etc

Figure 1: UML diagram representing the whole game (missing: NPC-classes)

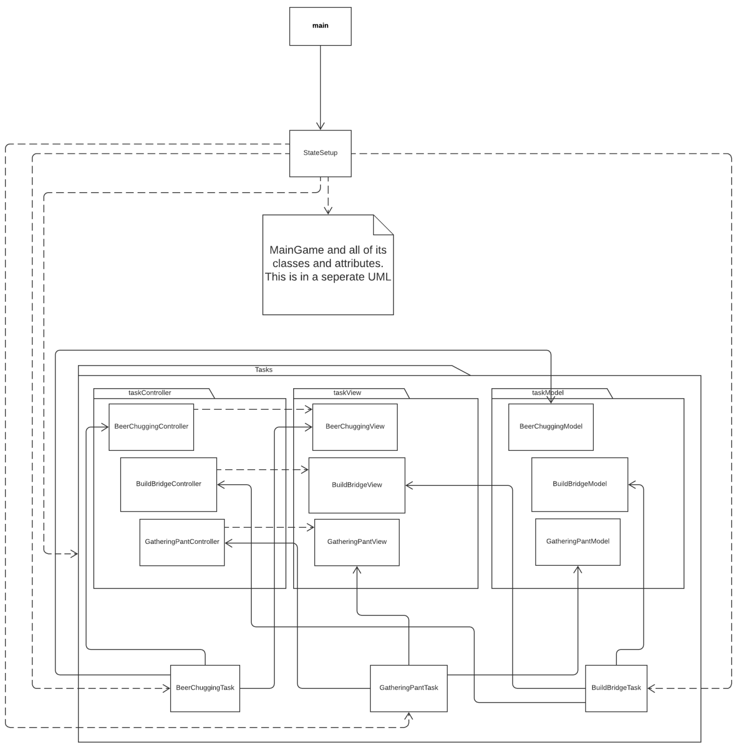

Figure 2: UML diagram representing the tasks

3.2 Design Patterns

3.2.1 Factory Pattern

To initialize every separate NPC class, a factory class was created. This way, the client code can simply call the factory and choose which NPC-object to create, without knowing any of the underlying initialization code.

3.2.2 State Pattern

4 Persistent data management

5 References

List all references to external tools, platforms, libraries, papers, etc. The purpose is that the reader can find additional information quickly and use this to understand how your application works.