System Design Document For “Hills”

Version 1.0

Date: 2017-05-16

Author: Gustav Engsmyre, Anton Annlöv

Revised by:

This Version Overrides all previous versions.

1. Introduction

The goal of this project is to create a loosely coupled, modular and easily expanded application. The program should follow the Passive Model-View-Controller design pattern

relatively closely. The Model should not have any couplings to the framework LWJGL. The design should also be easily testable.

1.1. Definitions, acronyms and abbreviations

LWJGL - Light Weight Java Game Library. Framework that allows access to OpenGL API.

GPU - Graphics processing unit, The Graphics card.

Java - Programming Language, platform independent.

JRE - Java Runtime Environment. Software that runs the java applications.

GLFW - Library used for window handling.

OpenGL - Graphics API.

FPS - Frames per second, how often the program and thereby the view updates.

NPC - non-player-character, an actor which moves but is not controlled by the player.

API - A set of functions which can access features created previously.

Sprites -

2. System Architecture:

2.1. Overview:

The application is based on a passive MVC model. The application runs on a single computer.

2.2. Software decomposition:

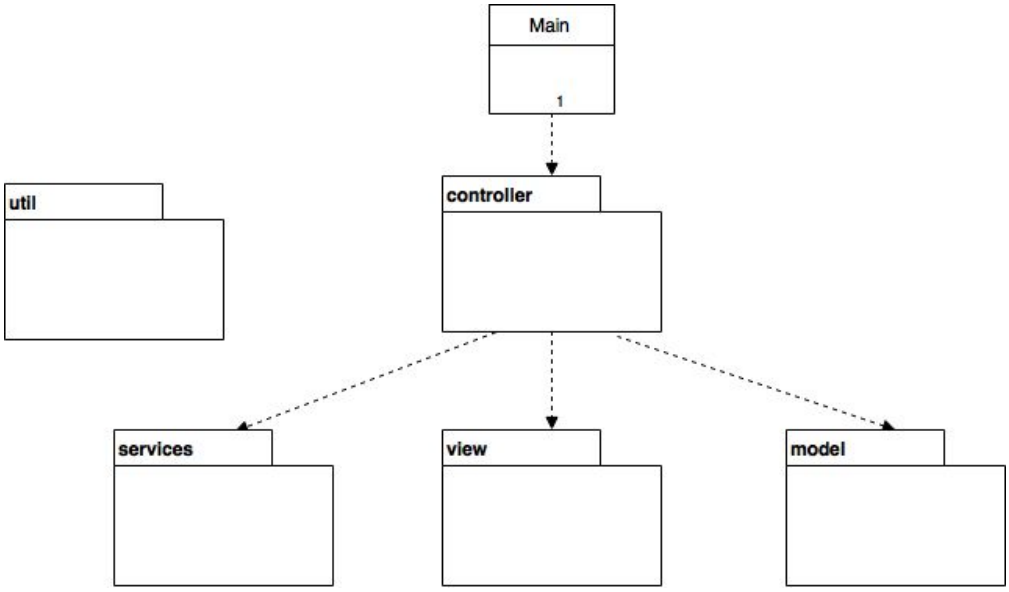
The application on the highest level consists of five packages; controller, model, services, util and view, as well as a main class used to start the program. The model, view and controller packages are what makes up the MVC pattern. The service package perform operations that are not necessary part of the main program and util contains code that can be reused by any other program.

The model contains all data for the game, and also handles all the game logic. The model should be completely isolated and should not have any dependencies on any other package in the application, the exception being util. Model is separated from all other parts of the program so it could be easily modified and reused by any other developer. All calls to the model are made from the controller package.

The controller package handles all updates to the model and handles calls to services and view, all communication in the application goes via the controller package. The package contains a controller for each feature in the model that requires an update. Each of the controllers handles sending update calls with data to interfaces which represent each of the features. Controllers also handles updating the view with needed information from the model each frame. Because all interaction between the user and the application happens through the keyboard and mouse, inputs are handled in separate controllers which other controllers can subscribe to in order to receive inputs. The implementation is handled this way because multiple different parts of the program needs inputs, and the technical inputs should be separated from the game itself.

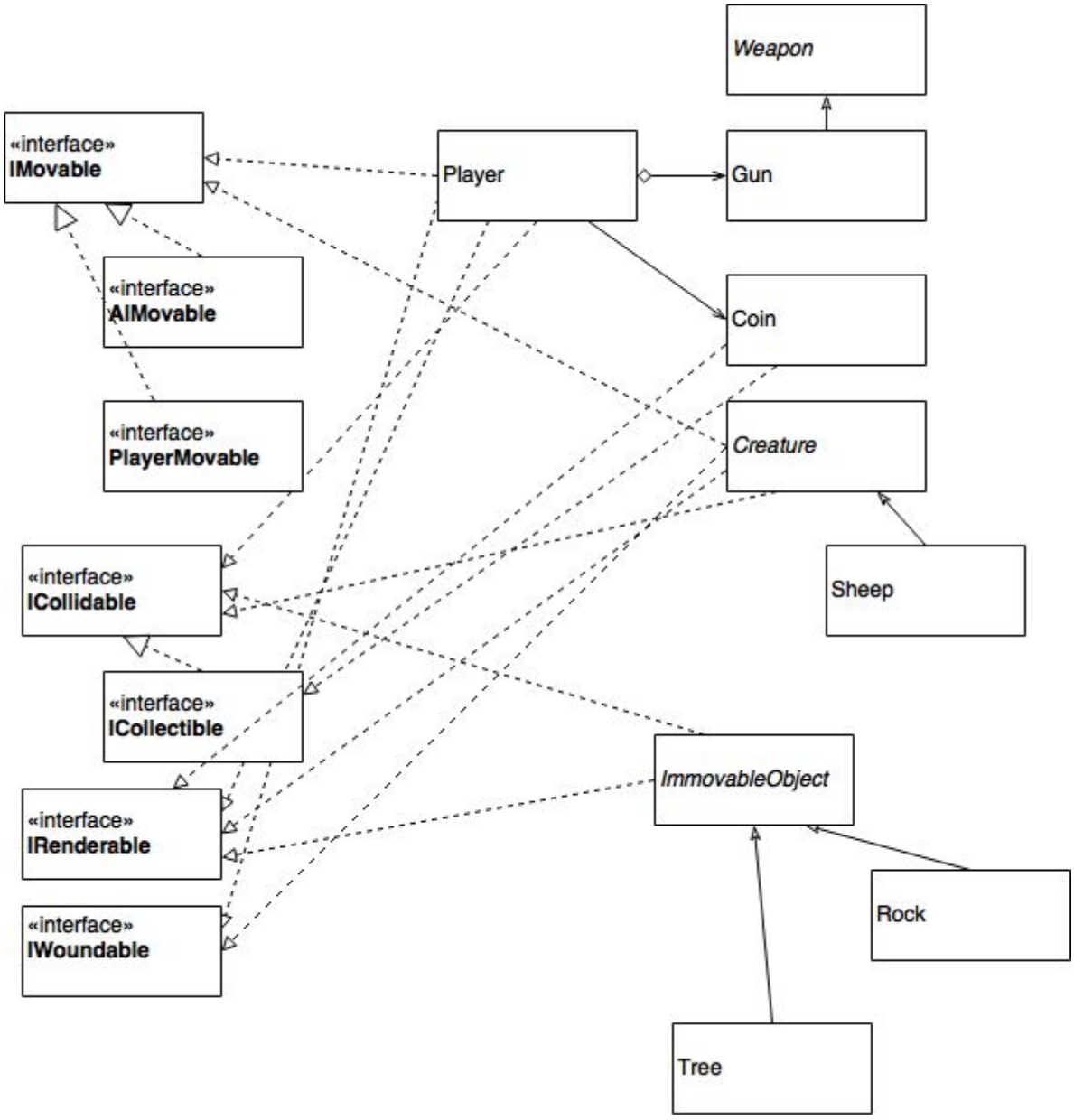
The util package contains all the code that can be reused by other applications and includes pure Math classes such as vectors, matrices and quaternions, but also shader classes and time classes etc.

The Service package contains classes that aren’t directly connected to the application but needed nonetheless. For further explanations of each subservice se subsystems.



2.3. Design model

The Model is structured based on Interfaces, every set of actions in the in game has a specific Interface, such as movable and collidable, classes in the model then implements the interfaces of every action the that the specific object should be able to perform. Every interface has a controller built specifically in order to modify and observe the behavior of the actions each of the interfaces represents.



2.4. Libraries and Frameworks

Very few external libraries were used in the making of the prototype. Most of the application is designed from scratch.

2.4.1. LWJGL

LWJGL is a java library used to access the OpenGL, OpenAL and OpenCL. The application is only design to utilize OpenGL. LWJGL accesses OpenGL directly which means that access is

high-performance and fast. LWJGL is only a wrapper for low-level access of OpenGL. This means that all physics, math and Graphics etc. are not present in the library and the functionality of this has been implemented directly into the application instead.

2.4.2. GLFW

GLFW is used in order to create the window that the program runs in, as well as handling keyboard and mouse inputs, GLFW is also used as an updating mechanism for the rest of the application.

2.5. Subsystems:

The main subsystems of the application is as follows:

Camera, which generates a cone shape which is used to view the world through. This lens are currently modified according to the direction the player is facing and the position of the player. The calculated cone is also used as a template for which parts of the world should be rendered, and the level of detail each object in the world should be rendered with.

Debug provides tools that makes developing easier by simplifying the render to make it easier to inspect and analyse the graphical result. Examples of debug features are a FPS counter and Wireframe mode which removes textures from the world.

Display, which handles setting callbacks to the Window, keyboard and mouse etc. This class is a way to simplify and isolate backend calls to the library GLFW.

files, which handles all writing and reading to and from external files.

Generation, which handles generating the terrain of the world, by using Open Simplex Noise (see "http://webstaff.itn.liu.se/~stegu/simplexnoise/simplexnoise.pdf" for details) as a tool to create pseudo random noise, then combining several different maps of noise and lastly overlaying it to a generated island shape in order to create lifelike terrain. Generation also handles generating a map of normal values corresponding to the heightmap. As well as generating a random path for the NPC:s movement.

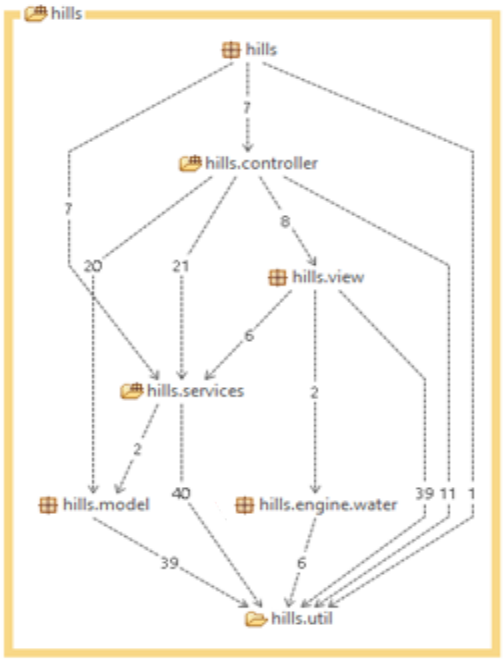
ModelDataService is a service for creating graphical models to use in the world, as of the

finished prototype, only cubes are created for lack of time.

the Terrain Service handles calculating the height of a given location's coordinates. Terrain also decides which terrain nodes should be rendered by the view by calculating which points are closer to the player.

2.6. Dependency analysis:

Dependencies are according to image shown below. Dependencies are closely similar to top level dependencies shown in software decomposition.



Dependency Analasys created by STAN

2.7. Persistent data management

The application does not support in-game saving of any kind. The game does however save the generated maps used to create the world as a PNG file in the resource folder.

2.8. Concurrency Issues:

The application is single threaded and no other concurrency issue has been detected while testing.