Software Engineering Design Specification

Team 4

May 3, 2016

Contents

1 Introduction

This document is divided into two sections. The first describes the high level design of the system including architectural decisions based on the system requirements. It describes the structure of the system as a whole. The second section describes the lower level implementation details, including the interaction of the domains of the system and the class layout and how they should be implemented.

2 High Level Design

This section describes how the requirements should be implemented. The system has been subdivided into sections that meet the requirements laid out by the user.

2.1 Architectural Design

The systems shall be divided in several subsystems. The organization of these subsystems is described in Figure 1.

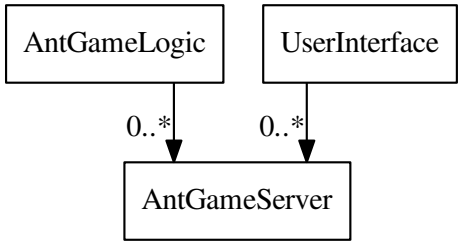
AntGameLogic

This domain shall handle the game logic of the system. It shall accept game information such as Ant Game Worlds and Ant Brains. It will allow for the playing of games as described by the requirements document Section 3.1. A new instance of the AntGameLogic will be created for every client that connects to the AntGameServer.

AntGameServer

The User Interface Client will connect to the Ant Game Server and request a new game. The server will then

Figure 1: Domains of the Ant Game System



generate a new Ant Game Logic instance with the information that the User has uploaded. A new instance of the Ant Game Logic is spawned for every user connection.

UserInterface

This domain of the system will allow for user interaction as well as vizualizing the Ant Game. The data for vizualization is provided by the AntGameLogic instance that corresponds to this interface client instance.

2.2 Common Tactical Policies

This section describes how work will be carried out in the team. Understanding of how all tasks are to be carried out is essential.

Testing

Unit level testing will be carried out for all the game logic code. this includes tasks such as parsing the ant brain and world files and all normal gameplay. Testing will be carried out with `py.test`.

Documentation

All methods shall be documented with Pythons `docstring` syntax. Normal methods for annotating method parameters and return values should follow the `Sphinx` autodoc syntax.

Implementation Language

The implementation shall be written in Python 2.7.10.

User Interface

The user interface shall written using HTML5 canvas.

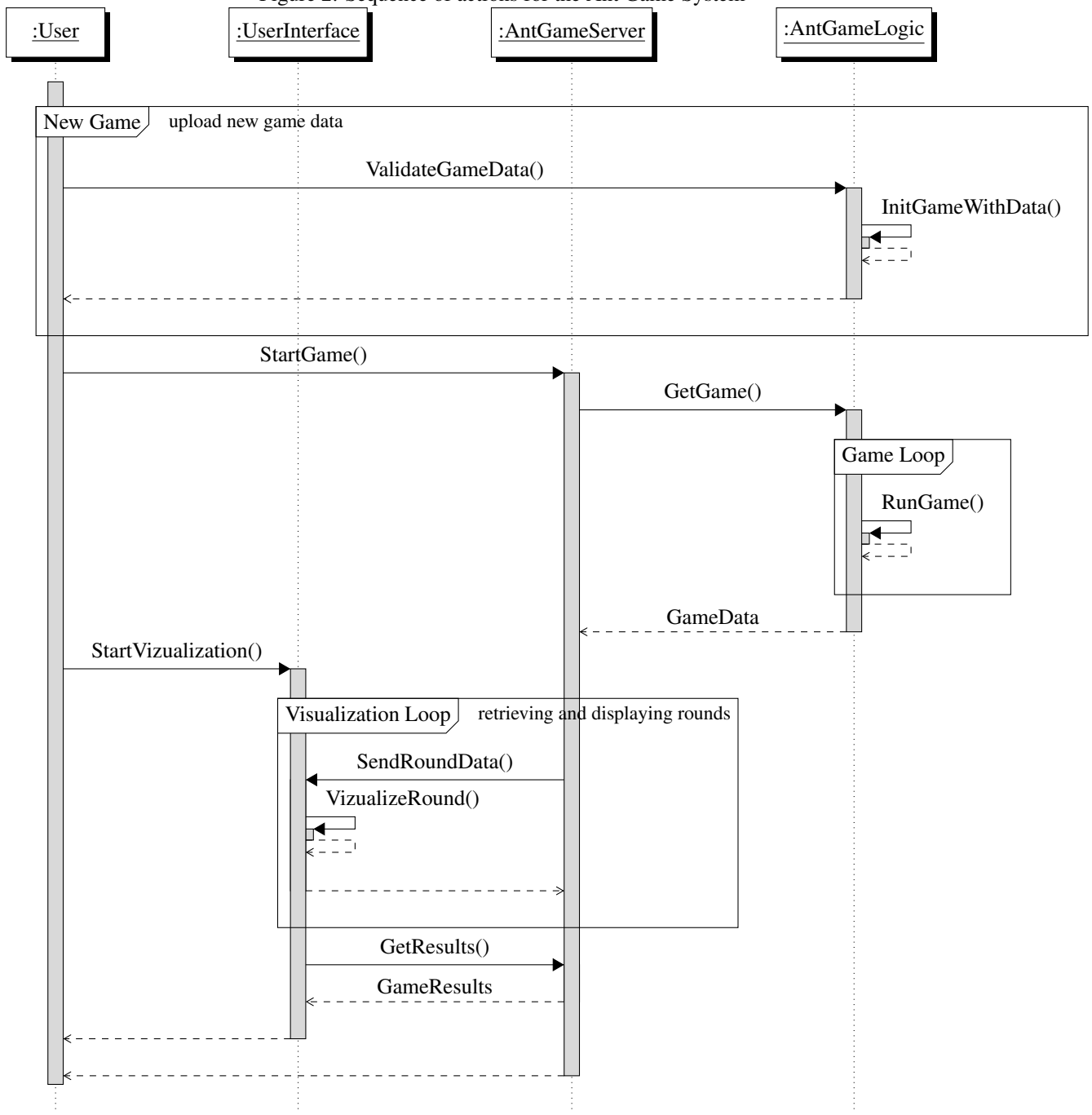
3 Low Level Design

This section describes how the system should be implemented. The description of how objects should be implemented is given. Some opportunities for inheritance have been identified.

3.1 Sequence Diagram

The order of interactions that the system will undertake are described in Figure 2.

Figure 2: Sequence of actions for the Ant Game System



3.2 Class Diagrams

In Figure 3 an abstract description of the Ant Game logic classes is presented. Some opportunities for abstraction have been identified, in particular:

- Instruction classes may inherit from a common interface, as all the instructions have a different action to be performed. The type of instruction and the action to be carried out is determined at runtime, i.e. the instruction attribute

of an ant is dynamic.
- Rocky Cells are a type of cell, but they cannot have any other attributes. It therefore makes sense to inherit from a Cell supertype.

Figure 3: Game Logic Class Structure