1 Introduction

This group will create an integration of the Gamygdala engine in the programming language GOAL.

1.1 Goals

In this project we have multiple goals. Besides the goals between the different groups which can be found in the product vision document we also have goals that are group specific. We think that it is very important for us that we have good code quality. Also functionality is important.

1.1.1 Source code quality

We think that it is important to have readible code that is wel written with the correct design patterns. Other software engineering principles we have learned so far should also be used appropriately and correctly. For example: Method sizes.

1.1.2 Functionality

By testing in different ways we want our code to be very functional. Since our job is to create an integration, functionality is the most important goal for our team.

2 Software Architecture

When a user is creating a virtual human in the programming language GOAL, they should be able to use the Gamygdala engine in an easy and understandable way.

2.1 Programming languages

We will work in javascript and Java. This is because GOAL is written in Java and the Gamygdala engine is written in Javascript. Both these languages we have learned in the first year of the bachelor Computer Science.Mostly we will be working in the source code of GOAL and find a way to build an integration of a javascript engine in a programming language written in Java. For this we also need to have implement a port from javascript to java for Gamygdala.

2.2 Tests

Testing of the software will be done mostly in the JUnit framework since we are writing Java code. The port between javascript and java also needs to be tested thoroughly. Also we will have continious integration during our project which means that everytime our project gets built and we get a report of tests but also source code quality discussed in the next section.

2.3 Code Quality

During this project our group wil use the pull-request way of developping code. This wil make sure that the quality is maintained since nothing can be merged to the master without permission of atleast to other student. It works like this. You create your own branch of the master and then you start coding. When you are finished you push your code to the branch (which is a mirror of the master except for your new implementations). After this you make a request of your branch being pulled to the master (hence pul requests). Then your colleague students can see this on GitHub and comment,reply and approve or dissaprove. When atleast 2 other student finally agree after you made adjustments they can allow you to merge your branch with the master. It helps our group in 2 ways. First of all it makes sure we do code reviews and think about what other are creating and if it is helpful for our goals or not. Secondly, by using this method less merge conflicts will happen. By using tools like Checkstyle we can maintain readability,good line length and method size violations.

2.4 Packages

This is hard to think about from the beginning for our context project since we are using GOAL as our starting point. We will maintain the structure of GOAL and try to fit in our implementation. All the source code and packages of GOAL can be found on their GitHub project.

3 Glossary