

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 that 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. By functionality we mean that focussing on

1.1.1 Source code quality

We think that it is important to have readable code that is well 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 use Java. This is because GOAL is written in Java. Gamygdala is written in Javascript, but we will use a port to java so that we don't need to work with Javascript. Mostly we will be working in the source code of GOAL and find a way to build an integration of the engine in GOAL itself.

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, but this won't be done by this group. We aim for a high test coverage. Also we will have continuous integration during our project which means that everytime our project gets built, 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 will use the pull-request method to develop code. This will make sure that the quality is maintained since nothing can be merged to the Master without permission of at least two other students. 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 pull requests). Then your colleague students can see this on GitHub and comment, reply and approve or disapprove. When at least 2 other students 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 others 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 package of GOAL can be found on their GitHub project.

Glossary

Gamygdala An emotional engine created to simulate emotions for virtual identities[2].. 1

GOAL A programming language developped at TU Delft for creating Virtual Humans[1].. 1, 2

JUnit This is a framework to build testsuites in Java.. 1

Master The master branch of a repository is the main of repository. Your main stable version.. 2

package a package it actually a map in which a part of your code is put. This is handy for keeping a good structure.. 2

Virtual Human A virtual human is an agent that uses logical rules to derive the wanted actions and can percept events from the environment. It is AI.. 1

References

[1] GOAL programming language <http://ii.tudelft.nl/trac/goal>

[2] Gamygdala emotion engine <http://ii.tudelft.nl/~joostb/gamygdala/index.html>