

VIRTUAL HUMANS FOR SERIOUS GAMING ARCHITECTURE DESIGN

Jannelie de Vries
Harmen Kroon
Jasper van Tiburg
Tom Brunner
Nando Kartoredjo



EEMC
Delft University of Technology
Delft June 2, 2016

Contents

1	Introduction	2
1.1	Design goals	2
1.1.1	Availability	2
1.1.2	Manageability	2
1.1.3	Performance	2
1.1.4	Reliability	2
1.1.5	Scalability	2
2	Software architecture views	3
2.1	Subsystem decomposition	3
2.2	Hardware/software mapping	3
2.3	Persistent data management	3
2.4	Concurrency	4

1 Introduction

This document provides a sketch of the system that is going to be built during the context project Virtual Humans. The architecture of the system is explained in the form of high level components of the system. These components are split in to sub components and sub-systems. This document will be continuously updated during the project development.

1.1 Design goals

During the development process we define certain goals the project should reach, next to solving the original problem. The following design goals will be maintained throughout the project:

1.1.1 Availability

The system will be built by the principle of Scrum, meaning that each week a working version is available for testing. By doing this we ensure that we are building what the clients wants; if (s)he sees features (s)he does not like we remove them, if they have other requirements we add them.

1.1.2 Manageability

The bot is able to be turned off and on. Its property can be managed (including its money). The goals can be altered to a higher demand.

1.1.3 Performance

It should not be easily noticed that the bot is not a human due to slow functioning of the bot.

1.1.4 Reliability

The bot should work in any given scenario, although it might decide it is better to do nothing.

1.1.5 Scalability

Our bot will be able to work together with other bots and in theory an infinite amount could be added. In a practical sense if an infinite amount would be added, an error would occur more likely with the tygron engine than with our bot.

2 Software architecture views

This chapter discusses the architecture of the system. It is first split into smaller parts and their dependencies on each other. The second paragraph describes the relation between hardware and software is explained. The third paragraph explains the data management. And the fourth paragraph describes the concurrency of the system.

2.1 Subsystem decomposition

The component diagram at **figure 1** shows the relations between the Tygron API, the connector and the GOAL agent.

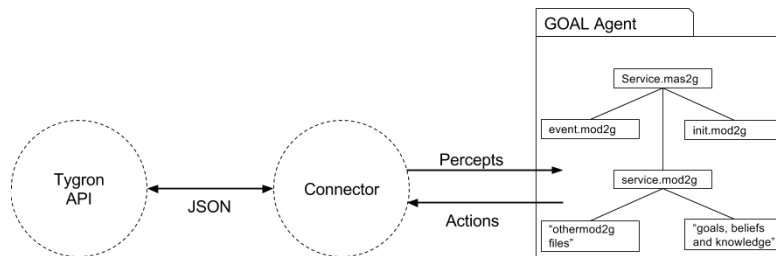


Figure 1: The component diagram

This figure shows the different components the project uses and how they connect to each other. You can see how our bot connects to the connector which in turn will connect to the Tygron API. The Tygron API exists of the module SDK (software development kit) which consists of all the graphics and calculations, basically the tygron game itself. The Connector uses the environment module, which is everything needed to connect a bot to the game and get and send information from the bot to the game. Finally, the Goal Agent consists of all the modules needed to run the bot.

This starts with the service.mas2g file, which is more or less the main class from which everything happens. This makes use of the init.mod2g file to get all data it needs at the start of the game, and events to get data which is changed, added or removed later in the game. Finally it uses service.mod2g, which handles the main actions the bot does. It uses goals, beliefs, and knowledge as data it can use and the other mod.2g as actions it does (for example build.mod2g for building).

2.2 Hardware/software mapping

Our virtual human uses a different connection approach to the server than a real human user. When a session is created via the server, a user connects through its own client to that session. Our virtual human will connect to the session using a connector and the Tygron SDK. Visualisation of the actions performed by the virtual human are visible through a separate instance of the Tygron Engine.

2.3 Persistent data management

Our GOAL agent is not responsible for storing persistent data, so it does not have external files or databases. This is so because just like a regular user would save his progress on the tygron server, our bot will ensure everything is stored on the database of the Tygron server as well.

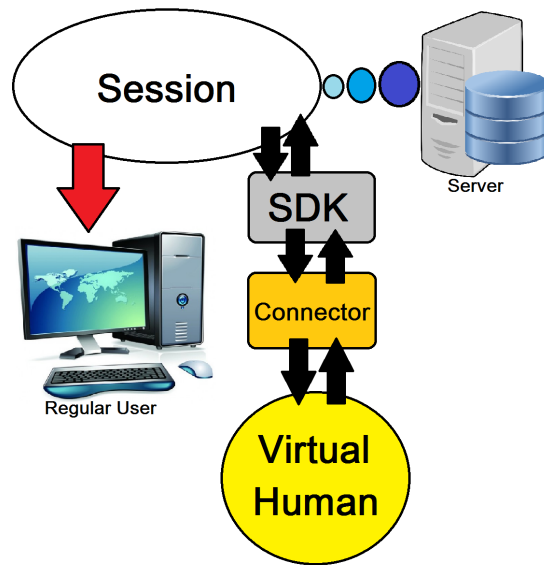


Figure 2: Hardware software mapping

2.4 Concurrency

Our GOAL Agent is just one process. It shares resources with other users of the Tygron environment. If a deadlock occurs, this would be on the Tygron environment and not very differently than when a deadlock would occur with human users. That is why our GOAL system won't have issues with deadlocks.