VIRTUAL HUMANS
FOR SERIOUS GAMING
ARCHITECTURE DESIGN

Jannelie de Vries
Harmen Kroon
Tom Brunner
Nando Kartoredjo

EEMC
Delft University of Technology
Delft June 17, 2016

Contents

1 Introduction

This document provides a detailed overview sketch of the system that has been 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 and showcasing. By doing this we ensure that we are building what the clients wants; if the client sees features it does not like we will remove them and if the client has altered its requirements we will add the wanted functionality. All code written is available for review via github repositories.

1.1.2 Manageability

The services stakeholder uses excel based indicators, which are imported into the Tygron Engine. Other stakeholder related settings can be altered through the Tygron Engine alone and are specific to a game scenario. The code for our connector extension and the GOAL agent are all well commented and documented through commits messages, pull requests and documents. The bot is able to be activated and deactived during a session at will.

1.1.3 Performance

The virtual human is designed for cooperating in a Tygron Engine session. Such a session has to be set-up, by creating a correct game and getting every stakeholder connected. The game requires minimal tweaks during development. The connection does depend on the Tygron servers with which every stakeholder connects and wheron the session is run. The virtual human performs faster than a human would in most situations, but slower when a lot of large computations are necessary.

1.1.4 Reliability

As said before, a connection with the Tygron Engine is required. Either due to internet connection problems or server problems this might fail. Another rare reliability problem might occur when the Tygron Engine is updated. There is not an existing version which allows local running of the session without need of the server connection. Game scenarios are stored on the Tygron server as well.

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. First we will go over the subsystem decomposition of the architecture. Then we will discuss the relation between hardware and software in the second paragraph. 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.

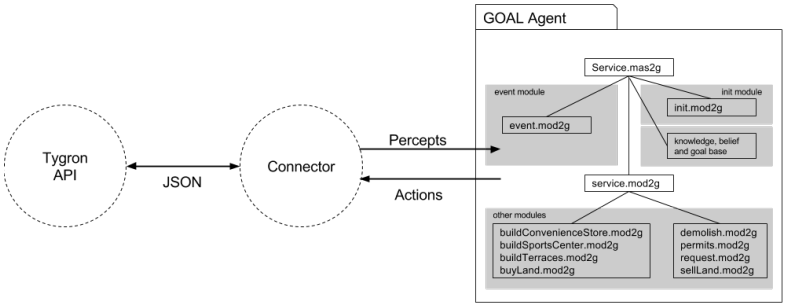

Figure 1: The component diagram

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

2.1.1 Connector

The connector is extended by a contextproject package of which all Virtual Human project teams are contributers. This extended environment provides the virtual human with a lot more possible functionality through more percepts and custom built actions. The package contains both functional code and test code. The importance of having this extension is the ability of performing bigger computations on data, which would be nigh impossible in GOAL.

2.1.2 GOAL agent

Our GOAL Multi Agent System is started from the `service.mas2g` file, which is more or less the main class from which everything happens. This file tells the environment to connect to a session and launches the service agent. The knowledge, belief and goal base are instantiated and subsequently the `init.mod2g` is launched. This init module insert all data it needs at the start of the game into the belief base of the virtual human. After the initialisation, the event module(`event.mod2g`) and the main module(`service.mod2g`) are run alternately. In the event module every incoming percept for that cycle is handled. The main module decides what actions the agent will take by following the programmed rules and using the goals, beliefs and knowledge

as data. The correct module will be called wherein even more rules have to be met. Our virtual human consits of eight extending modules, each providing a specific functionnality of the virtual human. There are three build modules for building the specific service. A module for buying land takes into account in which zone we need land the most. The demolish module sells non beneficial buildings. For communicating with other stakeholders in the game we have two special modules dealing with requests. `permits.mod2g` is desgined to ask the municipality for building permits, whereas `request.mod2g` answers land offers for buying from and selling to other stakeholders.

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. **figure 2** shows that our virtual human will connect to the session using the context specific connector to connect with the Tygron SDK, which in turn handles the connection to the server. Visualisation of the actions performed by the virtual human are visible through a seperate instance of the Tygron Engine.

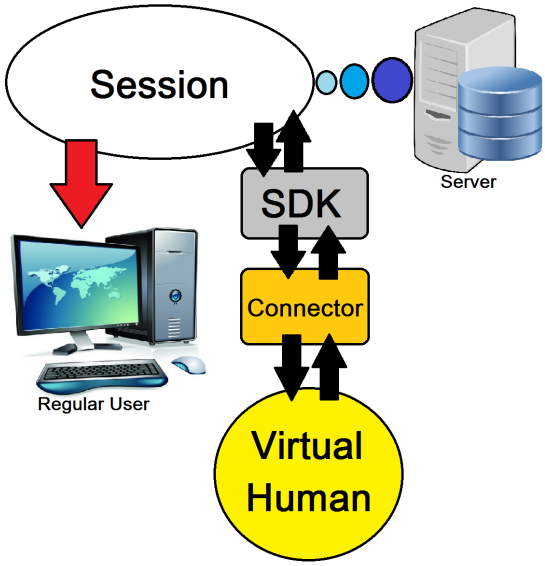

Figure 2: Hardware software mapping

2.3 Persistent data management

When our goal connects to the Tygron engine, the engine loads the game session from its data. Everytime when our bot makes changes in the game, those changes are stored in the engine. Our bot will then read those changes from the engine and choose its next actions, which thereafter will be stored again in the engine. When our bot has reached all its goals, and there are no more updates from the game engine, the data will be kept in the Tygron engine. So our GOAL agent is not responsible for storing persistent data and 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.

2.4 Concurrency

A GOAL agent system is designed in such a way that it always has to go through complete cycles by following the rules. Therefore, it is essential that all modules are run in the correct order. The environment also has no functionality without the input from the virtual human and is thus strongly bound to eachother; GOAL agents are interchangeble however. In practice the extending modules of the virtual human could be run out of order, but would not give the same final outcome. The fact that our GOAL agent is dependent on the Tygron engine is something unavoidable, but this causes various problems. The biggest problem is related to testing the virtual human when we need to use the Tygron engine. A Tygron server crash disrupts all sessions which then causes the Goal Agent tests to fail.

The goal agent shares resources with other users of the Tygron environment. If a deadlock occurs, this might be on the Tygron environment side and not very differently than when a deadlock would occur with human users. Deadlocks in our virtual human are prevented by covering all posible rule outcomes. Furthermore, we keep track of counters when performing certain actions. With those counters we can notice right away if something is failing and the virtual human will act accordingly to that.