Architecture Design

10/06/2016

CodeFox

Marco Houtman
Ronald van Driel
Joshua Slik
Matthijs Halvemaan
Lisette Veldkamp

Contents

1. Introduction

This Document provides a sketch of the system that is going to be built during the context project Virtual Humans for Serious gaming. It is used to represent the current state of the design of the system. The systems architecture is described in high-level components.

Design goals

The following design goals will be maintained throughout the project:

1.1. Availability

After each sprint the system should be in a working condition. This is important because it will allow the user to utilise the systems features and provide feedback based on the experience. The final version of the product should always be available and working in the same environment as it has been developed for.

1.2. Manageability

The code will be made publicly available for other programmers who are interested in our progress. All code will be well documented and commented for a clear overview of the system and will enable additional developers to extend or modify the agent.

1.3. Performance

Our product must be at least runnable on mid end personal computer systems. The agent should be able to act and react on real time environmental changes without delays exceeding more than a minute.

1.4. Reliability

The agent can be run with help of SimpleIDE or the GOAL environment in eclipse when the Tygron environment is available to the client and enough hardware resources are available.

1.5. Scalability

Our agent should keep working and interacting when additional agents are added to the environment and when the environment changes. It does not guarantee however that additional features in the environment or in additional agents will be utilized.

2. Software architecture views

In this software architecture section we describe a high level overview of different subsections of the system. In the first section we describe the programming aspect of the system, in the second section we provide a high level overview of the full system and the third section gives a high level overview of the most important subsection of the system.

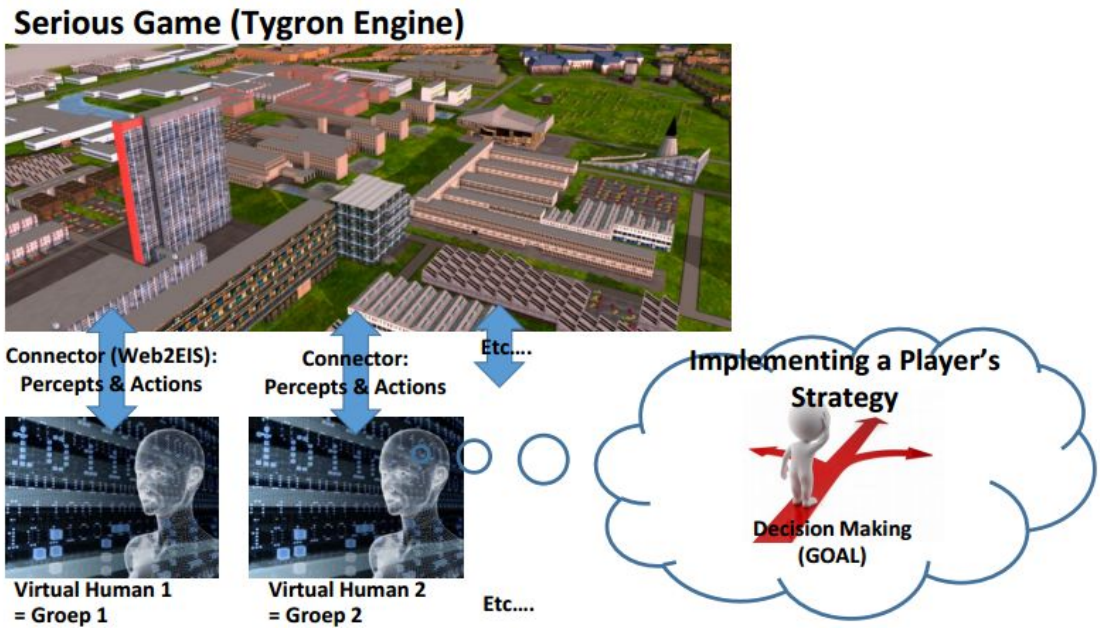
2.1 Subsystem decomposition

For this project both Java and GOAL, a high-level programming language for programming decision logic of cognitive agents that derive their choice of action from their beliefs and goals, will be used. Java is primarily used for the interaction between the agent and the environment. GOAL will be used to program the decision making of the agent using Artificial Intelligence.

The Tygron environment allows multiple agents to connect and interact. Each agent requires a connector which acts as an interface between Tygron engine and the agent (as seen in the image below). This interface is required for all agent interactions with the environment.

All agents have separate logic but are able to interact with each other through the Tygron Environment.

Our product will consist of the implementation of a single virtual human. For our product a basic connector has been provided but can be extended when desired. Most implementational logic will be required for the agent's decision making in GOAL.



Koen Hindriks, "Kickoff Virtual Humans 2016" (Powerpoint Presentatie, Den Haag, April 21, 2016).

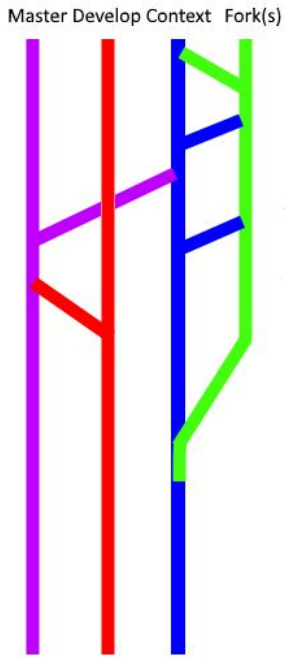
2.2. Web2EIS-Connector

This chapter will focus on the connector used to communicate between the Tygron Engine and the GOAL Agent. First a total overview of our workings with the connector is displayed in

the Connector Overview, next is an explanation on how to work with the connector in Working with the Connector.

2.2.1. Connector Overview

The Web2EIS-connector is our base connector currently being expanded by several people involved with the (EIS)hub and by the groups of the Virtual Humans for Serious Gaming project. The groups all have a fork of the Tygron-EIS and are expected to develop their own Percepts and Actions for the virtual human they are developing.



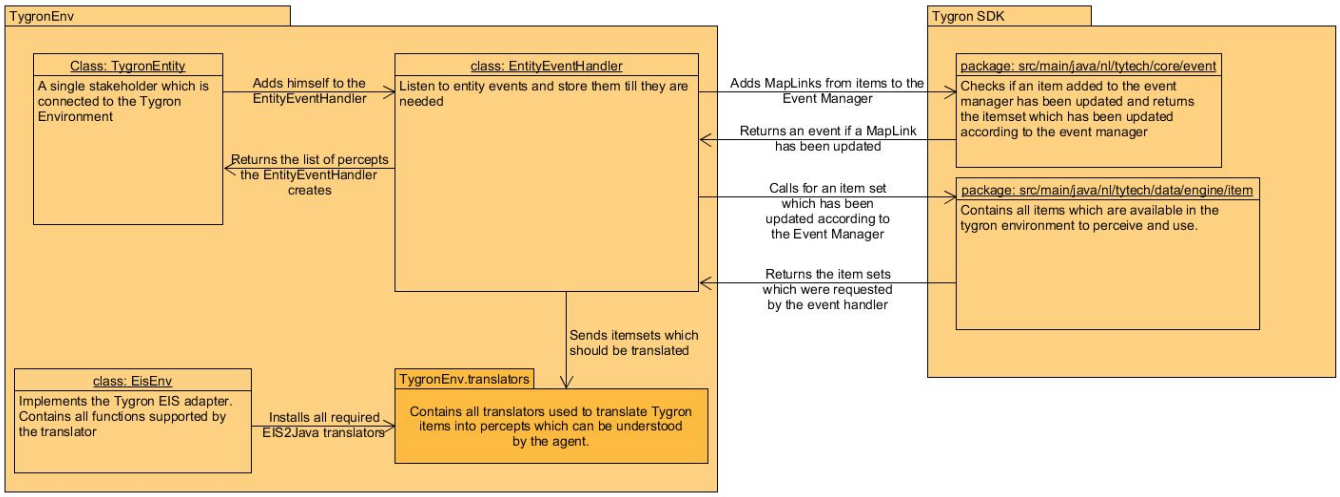
The dev branch is being developed by the (EIS)hub programmers and when they are ready to release their branch is merged into master. After release and a reviewed pull request master will be merged into the context branch. The groups will fork the context branch so they can work on their percepts and actions. When one of the context groups has a working feature for the context branch it will make a pull request and when the request is positively reviewed will merge their changes into the context branch so that it can be used by the other project groups.

Github's issue system is being used by the context groups to keep a clear overview of who is working on which part of the connector. An issue can be created by any team member and can be assigned to team members of all context groups.

2.2.2. Working with the Connector

There are two ways in which an agent can interact with the environment, these are the percepts, which is an observation of something happening in your environment, and Actions, which are operations which an agent can do to influence the environment.

The percepts are called on by a tygron entity, he adds himself to the EntityEventHandler which focusses on returning the percepts which can be defined with the help of MapLinks from the Tygron SDK as can be seen in image 3.



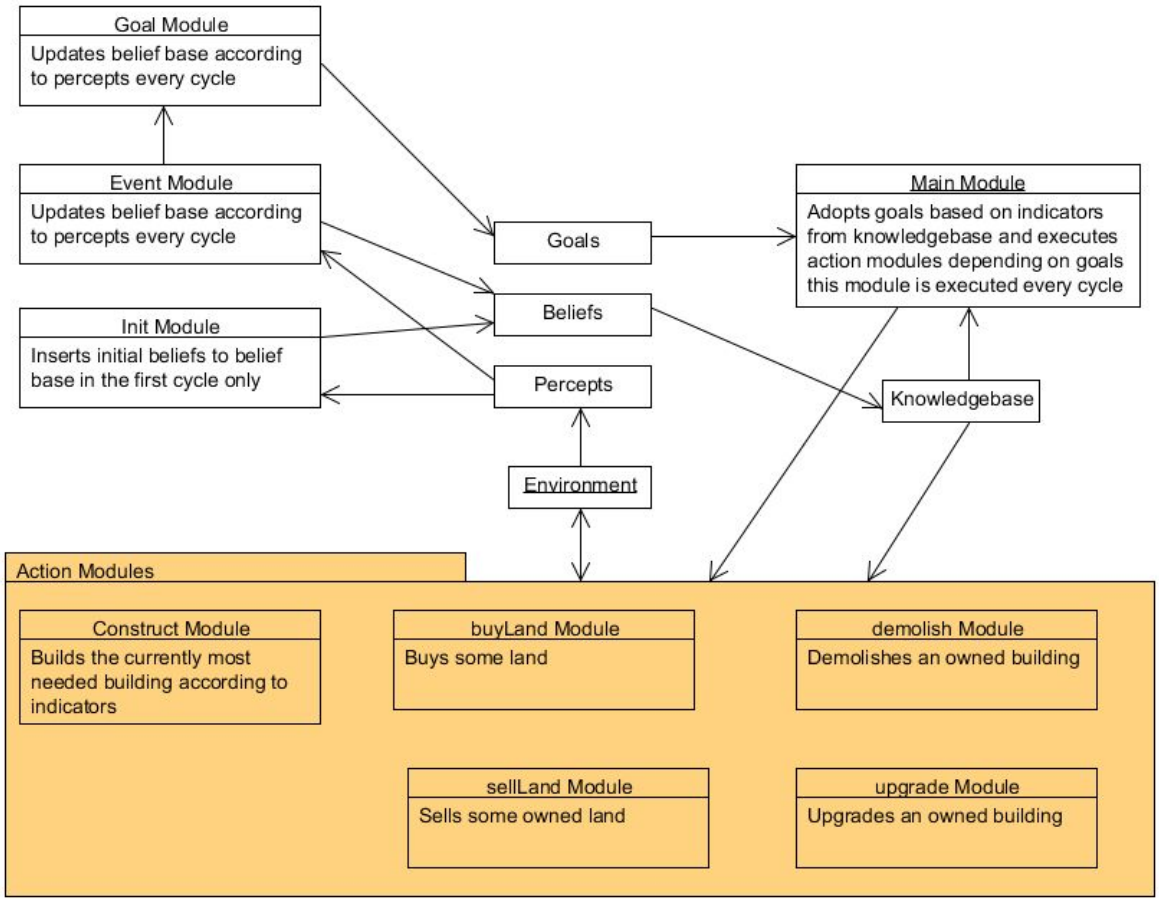
3. An overview of how the eis-connector handles percepts.

Defining a percept starts with adding a translator in the Translators package and you add the translator to the EisEnv class to make sure it is being used to translate the events in the correct way. In the EntityEventHandler you add a new MapLink which links to the SDK resources, when it is being updated the Event package returns a validator. This validator is recognized and the EntityEventHandler calls for the item sets which has been updated according to the validator. This itemset is sent to the translators class which gets a message from the EisEnv which translator to use for the class package.

2.3. GOAL agent

Our virtual human will percept events from the Tygron environment. The useful events will be processed in the agent as beliefs or goals for the mental state. Based on the mental state and the implemented decision rules, the agent will make smart decisions. Then the agent will observe what happens after its decision and will learn from this so that next time when there is a similar situation the agent can handle it better.

The connection from the agent with the environment is also shown in the overview below.



The first thing the agent does when launched is executing the Init Module which inserts basic percepts into the belief base. After the init module the continues the first cycle. At the start of every cycle all new percepts from the environment are processed and updated in the belief base by the event module. At the end of the execution of the even module it calls the Goal module. In this Goal Module the agent will use his ‘knowledgebase’ to deduce knowledge from to the belief base. Based on this knowledge goals are adopted or dropped. Thereafter the agent executes its main module. The main module determines which action module will be executed based on the goals of the agent. he first module will be the Buying Module. If the agent wants to construct more buildings but the agent has no available land or prefers to build on other land the agent can request to buy certain land.

The second module will be the Selling Module. If the agent owns ground but it does not need to construct buildings on this location or prefers to build on another location this ground can be sold to increase budget.

The third module will be the Demolishing Module. If the agent owns a building which it cannot use for upgrading and does not help to satisfy indicator targets it can demolish this building.

The fourth module will be the Constructing Module. If our agent owns empty ground and the agent still needs to construct buildings it will build this building.

The fifth and final module will be the Upgrade Module. If our agent owns buildings but it is desirable to have another building to satisfy the indicators better it can upgrade this building.

2.4. Hardware/software mapping

The virtual human will run a session on a computer or a laptop. The Tygron engine and the Tygron server run on other hardware than our virtual human.

It is able to connect to the session via the Tygron-EIS connector and the SDK. The Tygron-EIS connector communicates with the Tygron API. So through this connector the virtual human will be able to communicate with the Tygron server.

2.5. Persistent data management

The Tygron server will preserve the data in the map that our agent uses. All actions that are being executed that have an effect on the Tygron environment will be stored on the Tygron server. All this information is accessible for agents or users.

Our agent will collect this data (or at least the useful parts of the data) from the Tygron environment when it is needed for the strategy of the agent. The agent itself will have no knowledge of previous game sessions.
However, as soon as a game session is being started, the agent will collect data from the environment and will store it in his belief base. He will then use it when necessary. When the session ends all data in the belief base will be lost.

2.6. Concurrency

Each of the Virtual Human Context Project groups will develop one agent. These five agents will be all the stakeholders of our map and they will have to work together to fulfill their goals. To work together they will have to communicate with each other. Every agent does not make use of concurrency. Still, all agents will run in the same environment (parallel to each other) and therefore in that case, there will be concurrency.