Architecture Design

VH-Danshal

1 Introduction

This document provides set of overviews relating the to the architecture of the system. This particular section will cover the design goals. Each view will provide a different high level description of system and its different subsystems.

1.1 Design goals

In order to maintain a high-quality system we have set the following design goals for our code.

1.1.1 Design goal: Performance

Our code should be able to run on most modern consumer pc's. This is possible because of the architecture of our product. The connector should handle all low level functions and the GOAL agent should provide high level artificial intelligence, simulating the decision making of a real human player.

1.1.2 Design goal: Availability

Our team uses Travis for continuous integration in order to make sure our main branch is always a working product. This way a prototype can be shown to our client each week. The client can then review the changes and help us improve on our product and implement additional features.

1.1.3 Design goal: Code quality

Our code should be of high quality so it is easy to modify and maintain by both us and future users. To reach this goal we make use of checkstyle, junit, FindBugs, PMD, Cobertura and Powermockito.

1.1.4 Design goal: Functionality

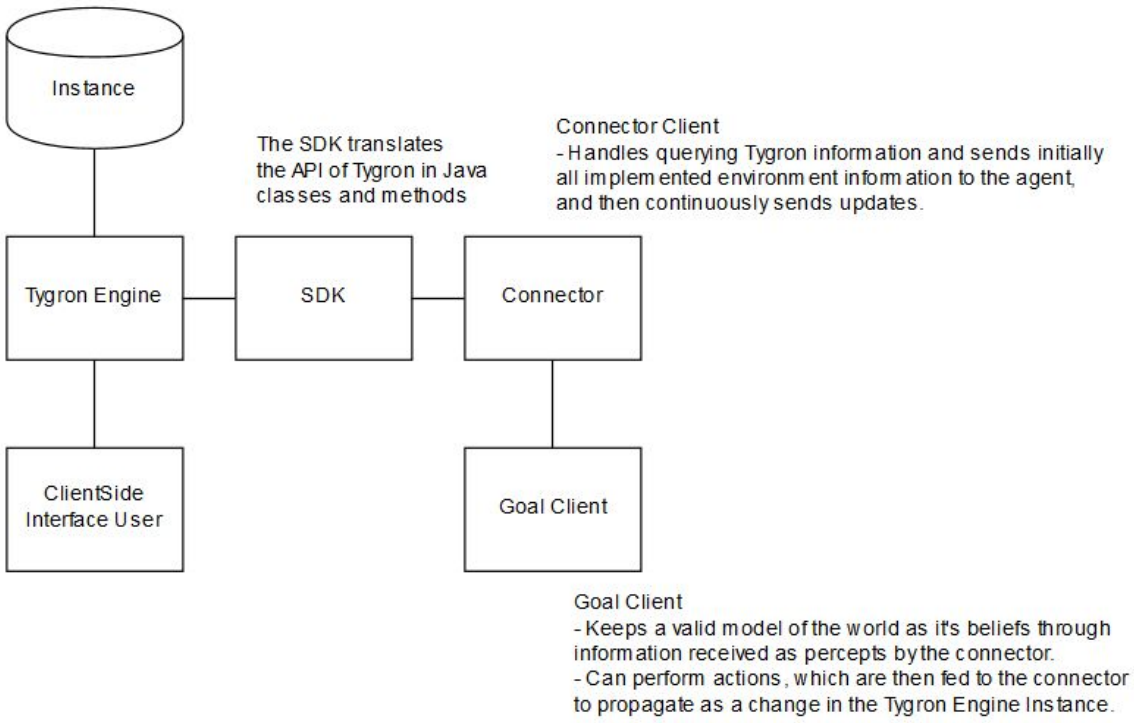
We implement an agent that can fulfill the role of municipality in the TU Delft environment within the Tygron engine. The bot will make decisions through knowledge of the environment and communication with other players and tries to change the environment to improve it in favor of the indicators of the municipality.

2 Software architecture views

2.1 Subsystem decomposition

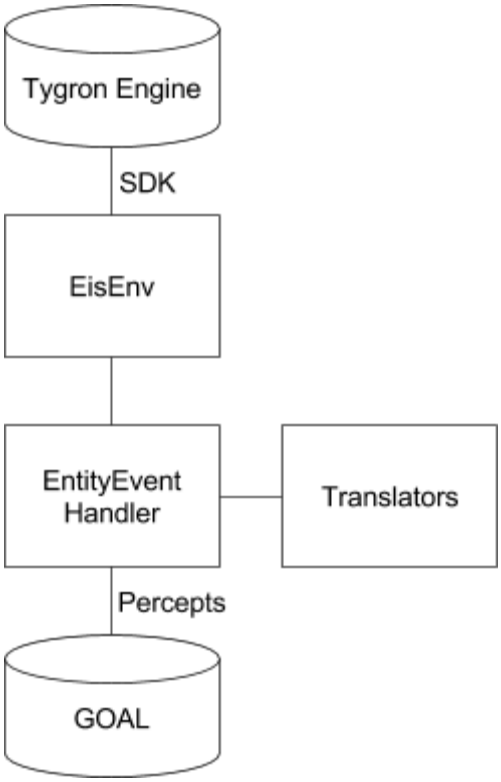
The following diagram reflects the architecture of the Tygron game & engine, as well the way in which users and AI have to connect to the engine. It shows what behaviours should be handled by what systems.

The given assignment states that the Goal Client, the AI, has to be build and connect to the SDK via the connector. The focus therefore lies on improving the current Connector and constructing the AI.



2.1.1 The Connector

The Connector is the link between the AI and the SDK. It is able to communicate information over the state of the game. It allows translation from Java (used by the SDK) to GOAL (used by the AI) and back. The connector itself is also written in java.



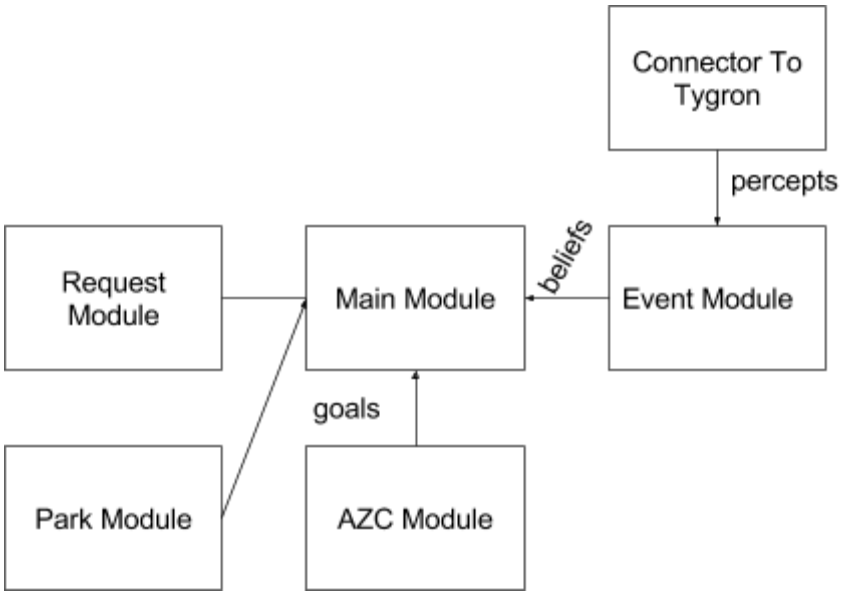
The connector supports our agent and its ability to interact with the environment. It can assist the AI by performing complicated calculations in order to give the AI more concrete information. It give the information to the agent in the form of percepts. It stimulates actions by communicating actions the AI wishes to perform to the SDK.

2.1.2 The AI

The agent represents a stakeholder in the game. It is an artificial intelligence that represents our stakeholder. It needs to make informed decisions in order to advance its interests. The fact that it is a AI lends itself to using GOAL as a programming language.

As the stakeholder in question, the municipality, we will need to review building permit requests. And we also have to reach our own goals by building parks and immigrant housing. GOAL is used to perform high level decision making.

The Ai is structured in modules. The event module will handle all percepts and insert that information as beliefs. The park and azc module will evaluate our current situation and if they deem it possible and necessary to build their respective buildings they will adopt the goal to do so. The main module will then try to reach all goals.


Connector To Tygron
percepts
Request Module
Main Module
beliefs
Event Module
goals
Park Module
AZC Module

2.2 Programming languages

2.2.1 GOAL

GOAL is a high level multi agent programing language. It uses percepts to view the environment and creates believes to built a representation of the environment. The bot can than perform actions to interact with the environment.

Percepts
Currently we have these basic percepts

- functions(<X>)
  Returns a list of all functions our bot can perform.
- settings(<X>)
- buildings(<X>)
  Returns some basic information about all buildings, this is used to percept when a building is successfully built
- stakeholders(<X>)
  Returns all stakeholders
- request(<type>, <Id>, <[Answer(<AnswerId>, <Description>)]>)
  When a request is made to buy land or for approval of the construction of a building, this Is percepted for agents for which it is relevant. Generally the one who wants the action and the one who needs to agree with it. The Id is connected with the Actionlog percept. Only those requests of type “INTERACTION” need to be answered. Generally the possible answers are a Integer between 0 and 3, where 0 is Yes, 1 is No, 2 is take it for free, 3 is close the window, which is not really relevant for our agent.

Actions

- building_plan_construction(Id, Level, MultiPolygon)
  Builds a simple building with [level] floors
- popup_answer(<Id>, <AnswerId>)
  With this a request can be answered

2.3 Hardware/Software Mapping

The only hardware component necessary for the connector and AI to run is a PC or laptop. The connector uses the SDK provided by Tygron, which is also present on the laptop. Using the SDK, the connector is able to produce a jar file on the laptop. The jar file can then be imported to the bot, which can then run from the PC or laptop as well.

While the agent runs, it does connect to the server. The server and Tygron engine to run on different hardware components when compared to an AI. Communication between the AI and the server is via the Internet, using HTTP. Different agents can be run from the same computers or separate ones. An agent has it's own client instance and is loosely connected to their own connector environment instance, it's loose because they are different programs. These two instances have to be on the same machine. Each agent connector has it's own connection with the Tygron server .

2.4 Persistent data management

Persistent data is data that is stored even after the application is done running.
The agent and the connector both do not have to store persistent information. So there is no need for a database or external files.

2.5 Concurrency

There is no concurrency present in our AI, nor have we implemented it into the connector. The AI uses the GOAL programming language, which itself does not allow concurrency. An interesting cause for concurrency is when multiple agents work together in one Tygron instance. The Tygron server has to account for handling the requests in an orderly fashion, but this is far from the scope of our agent program.

3 Glossary

Goal Client & AI
These terms can be used interchangeably and refer to the artificial intelligence we have to build during this project. Our artificial intelligence will play the role of Municipality.

The Game & Session
These terms can be used interchangeably. They refer to the environment in which the AI needs to operate. The environment runs in sessions and represents a planning sessions in which multiple stakeholders participate.

Stakeholder
An entity connecting to the Tygron engine in order to attend a session. They can either be human or an AI.