Architecture Design, TI2806

Blazin and the Goons

Authors:

Shane Koppers,	4359062
Floris List,	4380258
Hidde Lycklama,	4397614
Joris Oudejans,	4393694
Jordy Weijgertse,	4247124

Contents

1 Introduction

In this document the design of the system being built will be discussed. The architecture of the system will be explained as well as split into smaller parts to explain further.

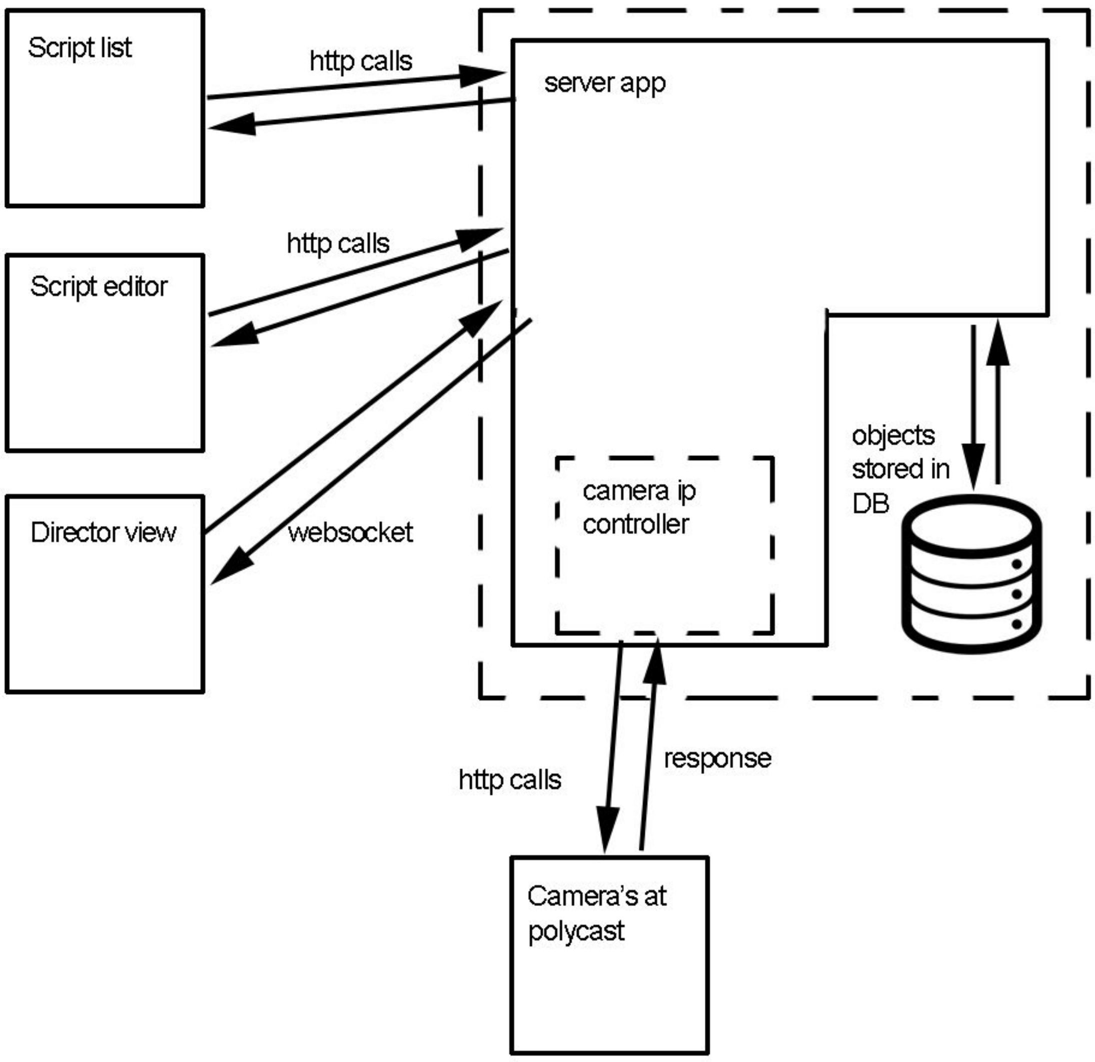
1.1 Design goals

- **Performance**
  Since we need to improve the workflow of the camera crew performance is of utmost importance. The system needs to work fast with as little delay as possible. Too much delay can lead to problems when cameras are not in position when they need to be. The system must therefore be realtime, there is no room for noticeable latency.
- **Reliability**
  The system is being designed to lift pressure and workload off of the camera crew. If the cameras are not in the right preset at the right time or some other malfunction happens, the crew can't trust the system, especially not in a live environment. Especially in a live environment where multiple concurrent users are operating the system, reliability is even more important.
- **Security**
  If the server is not going to be local it should be unable for outsiders to go in the system and control the cameras or change the presets for obvious reasons.

2 Software architecture views

This chapter will explain the architecture of the system. The first paragraph will show the system as a set of subsystems and how they interact with each other. The second paragraph will explain the relation between the hardware and the software of the system. The third paragraph shows how the system manages its data. Lastly, the fourth paragraph explains how the system stays concurrent and prevents unwanted interactions.

2.1 Subsystem decomposition



The whole system is divided into subsystems:

- **Views**
  There are three main views as of now. The script list view shows the available scripts to run. The script editor is the view responsible for creating and editing scripts before

actually running them. The main view is the director view, which contains all aspects that are important during the recording of an event. This view is divided in a couple of subviews, namely the running script, the available presets, the livestream with scheduled next camera's and a dock that contains action buttons and a script timeline.

- **Database**
  The database is used to store the scripts, cameras, actions and presets. Currently it consists of tables to represent the following objects
  - Scripts
  - Actions - actions in a script
  - Presets - presets for cameras
  - ActiveScript - the current active script together with its position and running time.
- **Camera ip controller**
  The component that handles all communications with the cameras. It provides a Java interface for the framework's controllers to interact with. It assumes the cameras are reachable over the ip provided. In the current state of the system, this is when the host is connected to the VPN provided by polycast
- **Server app**
  The server app here is the central part. It delivers the views to the client's computer, fetches scripts presets from the database and converts the JSON-formatted http calls to IP commands for the camera ip controller. The server app also keeps a list of connected client sockets to keep all updated.

2.2 Hardware/software mapping

Most of the hardware/software mapping is explained in the previous paragraph. A user can use a pc to connect to the server, which in turn will return one of the views. The user can then use those views to send calls to the server which will then handle those as explained in the previous paragraph.

2.3 Persistent data management

For local development, an in-memory database is used. This provides good flexibility and makes sure no bugs occur that arise from the (lack of the) state in the database, because every new run the database is refreshed.

There is an online version of the system which interacts with a Postgres database. This instance can be found at http://blazinandthegoons.herokuapp.com

2.4 Concurrency

The application has a lot of possible concurrency problems. This is because all views share the same database and running video process, concurrency can be an issue if not tackled right.

Multiple clients can interact with the server in a very short time which creates an opportunity for race conditions. Elaborate tests have been developed to simulate the interaction of clients with the server. Included with this are tests that verify the right behaviour when multiple users are interacting together.

3 Glossary

Client - A remote browser that runs the app and is connected to the server.
Heroku - Service to deploy application online.
Postgres - Database system.
Server - Mainly the brain of the structure, the server synchronizes the views on every client.
View - A way the program represents itself through a user interface/web page.
VPN - Short for Virtual Private Network which can be used for secure communication between parties.