# Emerging Architecture Design
# version 1.0

May 7, 2015

# 1 Introduction

This document provides an overview of the architecture that is going to be build for this project. The components are covered at a high level view, to give a representation of how the different components work and the dependencies between them.

## 1.1 Design goals

The following design goals will be maintained throughout the project:

- Modular
  The system will be build in a modular way. This is to keep all the different modules easily testable and leaves the possibility to extend the system later on.
- Performance
  The system will try to optimize for running as fast as possible and as efficient as possible, by using an event driven architecture.

# 2 Software architecture views

This chapter is about architecture of the system. First the different parts of the system are explained and the relations between them. In the second paragraph the link between the software and the hardware are discussed. The third paragraph illustrates the data management of the system.

## 2.1 Subsystem decomposition (sub-systems and dependencies between them)

The system is broken down into 3 subsystems:

- Backend
  The backend runs the game-engine and holds the model of the game. All the logic is contained in the sub system. The backend uses domain events to notify the connecting client if there is any update
- Renderer
  The renderer connects to the backend. Its role is to display the model changes on the screen
- Client
  The client is a html / js website hosted that the users of the system use to play the game. It send all the commands to the backend

### 2.2 Hardware/software mapping (mapping of sub-systems to processes and computers, communication between computers)

The subsystems of the system are designed so it can be run on only one machine or as the main intend to use it on 3 different machines.
The backend and the html/js website is hosted on a server/computer. This computer is connected to a dedicated access point.
Each of the renderers runs on a separate machine that is connected to a beamer.
The renderers and the client devices that are used to access the website connect to the backend machine through a socket connection.

### 2.3 Persistent data management (file/ database, database design)

The system doesn't use persistent storage, it stores the whole game model in memory.

### 2.4 Concurrency (processes, shared resources, communication between processes, deadlocks prevention)

This system uses non blocking web sockets using a protocol made for this game to communicate between the backend and the clients. The backend and the renderer share the same core code base e.g. the model and network classes.

## 3 Glossary