Automat a Football Game — Software Project Management Plan

Stanley C. Kemp

April 24, 2021

1 Introduction

1.1 Project Overview and History

This project is to create a small browser-based game that is similar in distribution style to `https://agar.io`. The program should be easy and simple to run on many machines, the ones that support the underlying technologies of websockets and the 2D canvas should do it. The game should be made with accessibility and fun in mind, and there should be multiplayer interactivity.

When coming up with the idea for this website, I thought: what do people usually lack when trying to play online games? Skill, to be sure, but other than that the main problem is lag. This can be very challenging to play past when participating in games like *Fortnite* or online fighting games. Most of the world lives with low-quality internet, or no internet at all.

Turn-based games do not necessarily have these problems, or not in large quantities. But often, they have complex mechanics that are inaccessible to some people, especially children. So, the idea I had was to make a simple turn-based physics-based strategic fighting game, where a player would control the muscles of their own player in order to deplete the health-bar of the enemy, or defend themselves from some enemy.

But unfortunately, there was a large problem with the scope of this project. None of the javascript physics engines had the things that I wanted to have out of the box, and learning them was a hassle. So, I decided that if I was going to build this, then I had to create my own. I had thought that building a small physics engine would be easier than building a large one, but I did not know what I had to build. Turns out it was extremely difficult. I spent a week and a half on the physics before realizing that I wasn't going to have it done in time for the deadline.

I needed a new goal, and quick!

So I thought: what would be simple to *implement*, in addition to the previous qualities that I had thought about before? Text-based games would be pretty simple to implement, but that would require a story or some form of language recognition — not easy to implement — and text games are often impenetrable, relative to games with graphics. Furthermore, I would need to do something on the server-side of things, as it was one of the requirements laid out to me by the professor.

So, the idea that I came up with was to make a game based on Conway's Game of Life. The user game complexity would have to go up somewhat, because of the nature of the game, but not too much, to make it too difficult to understand. The engine behind it could be implemented within a matter of hours, and this made my job much easier than it was with a physics-based game.

1.2 Game Premise/Design

The turn-based aspect of the original design remains the same, allowing the backend code reuse. The goal for this game is simple — have the most cells make it to your opponent's endzone. There would be a predetermined small number of turns, to keep games short and snappy. For each turn there are two parts, one where players build their mechanisms, and one where players can't do anything, but the computer would simulate a large number of turns for the player. Because of the nature of Conway's game of life, players could score own goals.

1.3 Project deliverables — Work Breakdown Structure

1.3.1 TODO General Project Organization by April 27

- ☒ Update SPMP
  - ☒ Timeline
  - ☒ Explain what happened in the evolution of the project
  - ☒ Update other portions of the SPMP according to the instructions
- ☒ Update Use Cases
  - ☒ Add Authenticated/Unauthenticated user
  - ☒ Add Extra Admin Criterion
- ☐ GANNT Chart

- ☐ System Test Procedure
- ☐ Milestone Review

1.3.2 TODO Working Website by May 3

- ☒ Get underlying algorithm for Conway's game of Life up and running
- ☐ Make front-end
  - ☐ Name entry/Introduction
  - ☐ Game queue
  - ☐ Game
  - ☐ Practice Mode
- ⊟ Make backend
  - ☒ Name System
  - ☒ Matchmaking
  - ☐ Moves
  - ☐ Victory/Defeat/Disconnection
- ☐ Connect the Frontend to the Backend
  - ☐ Develop a shared JSON code system, codes for each thing happening
  - ☐ Get connection working through websockets
- ☐ Buy a website name
- ☐ Deploy to Heroku

1.4 Reference Materials

2 Project Organization

This is a one-man-project, so not much time was spent on either Managerial Process or Project Organization. I usually take this apprioach to projects — if it slows me down, I lose it, and if it speeds me up, I use it. However, with a project on the magnitude such as this one, some elaboration is needed.

2.1 Process Model

The software process model I have chosen is the chaos model. The chaos model manages complexity by breaking problems into smaller sub-problems. There are two main rules to the chaos model of development. The first rule is, to solve big problems, solve the smaller problems that they rely on first, then solve the intermediate problems that lead to the big problem being solved. The second rule is always work on the biggest issue first.

2.2 Organizational Structure

The model of the type of this organization is self-employment.

2.3 Organizational Interfaces

2.3.1 Relationship to Hosting Services

The hosting services are going to be alright, as long as I pay them for their work. If enough people join, I might have to upgrade the hosting to a bigger Dyno for Heroku, and I might have to upgrade hosting for the frontend as well.

2.3.2 Relationship to Players of The Game

The relationship to the community of this game is very important, as it is an online game community, and if they aren't happy, they will go somewhere else.

2.4 Project Responsibilities

- Deliver a smooth gaming experience
  - Deliver a quick gaming experience
- Deliver an accessible gaming experience
- Be responsive to user feedback — What needs fixing?

2.5 Low-Level Responsibilities

- Programmer
  - Aware of time

  - Aware of the general state of the project
  - Aware of the code and the bugs inside
- Admin
  - Report to the programmer possible bugs/design flaws

3 Managerial Process

3.1 Change Management

To manage change in the development process, all serious technological changes will be made before programming. Minor changes can be made during the development process, of course, but once I am set on a method, tool, or technique, I cannot change it.

Addendum: Apr 21, 2021. Because of unforseen issues with developing the game, I have decided to make a different game that is easier to implement. More changes to the project check list will come soon.

Sub-Addendum: Apr 24, 2021. Changes are now complete.

4 Technical Process

4.1 Methods, Tools, and Techniques

I plan to use these tools on this project:

- Javascript
  - `Node.js` — server backend framework for Javascript
    - `npm` — Node Package manager
      - `ws` — websocket package
  - Browser-based
    - The built-in websocket API
    - The built-in 2D canvas API (or the alternative — SVG API)
- WebSockets — To allow full-duplex communication between the client and a server
- HTML/CSS
- SVG or Canvas

- Git
- Heroku
- Development tools
  - Firefox
  - EMACS (for writing the software/documenting the project process)
    * Javascript Mode
    * Org Mode