Turn-Based World Tournament — Software Project Management Plan

Stanley C. Kemp

March 31, 2021

1 Introduction

1.1 Project Overview

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.

1.2 Project deliverables — Work Breakdown Structure

1.2.1 TODO General Project Organization [2/4] by March 31

- ☒ Tool Selection
- ☐ Component Graphs
- ☒ Fill out SPMP

- ☒ Timeline

☐ Sign-off

1.2.2 TODO Working Server/Client [0/3] by April 15

⊟ Get game internals working

- ⊟ Connection
  - ☐ HTTPS connection
  - ☒ WebSocket connection
  - ☐
- ⊟ Matchmaking
  - ☒ Get server to make matches without many race conditions
  - ☒ Tell people who their partner is
  - ☐ Requeue people when they are
- ☐ Game [0/2]
  - ☐ Server
    - ☐ Transmission of turns to the other player
    - ☐ Sanitization of data
  - ☐ Client
    - ☐ Application of data
    - ☐ Recognition of current game state

⊟ Get graphics and a user interface working

- ☐ Buttons
- ⊟ Time limit
  - ☒ Decide whether a time limit is really necessary.
  - ☐ Implement time limit
  - ☐ Synchronize from server to client
  - ☐ Implement server-client connection diagnosis through the ping.
- ☐ Canvas Display [0%]
  - ☐ Display Components of models
  - ☐ Display current Health

☐ Implement a physics engine

- ☐ Implement Bricks
  - Weight
  - Friction
  - Velocity
  - Rotation
  - Acceleration
  - Momentum
- ☐ Implement Gravity
- ☐ Implement Joints and Structures

1.2.3 TODO Working Website [0/2] by April 24

- ☐ Register a domain and hosting
  - ☐ Upload `index.html` and `index.js` frontend to the server
  - ☐ Register for Back-end hosting with `Heroku`
  - ☐ Figure out how to connect the front-end to the back-end and then do it.
- ☐ Test it to make sure it works

1.3 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?

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.

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 (minimal usage)
- SVG or Canvas
- Git
- Heroku
- Development tools
  - Firefox
  - EMACS (for writing the software/documenting the project process)
    - Javascript Mode
    - Org Mode