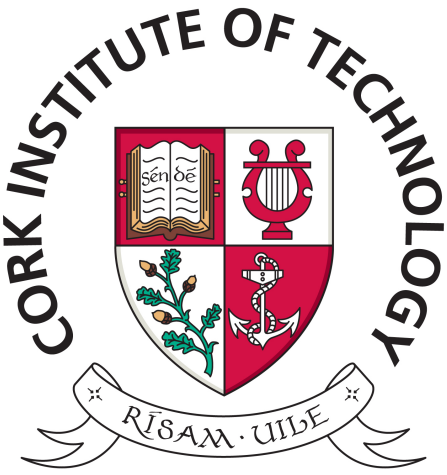


Web Based Multiplayer Game

by

Dominik Bahrynowski

This thesis has been submitted in partial fulfillment for the degree of Bachelor of Science in Web Development

in the
Faculty of Engineering and Science
Department of Computer Science

May 2019

Declaration of Authorship

I, Dominik Bahrynowski , declare that this thesis titled, 'Web Based Multiplayer Game' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an undergraduate degree at Cork Institute of Technology.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Cork Institute of Technology or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

CORK INSTITUTE OF TECHNOLOGY

Abstract

Faculty of Engineering and Science
Department of Computer Science

Bachelor of Science

by Dominik Bahrynowski

The aim I have for this project is to make a web based game. Browser games have been highly popular pretty much since the beginning of the internet, and since then, the power of computers has increased drastically. With the increase of power of PCs, programming technology has also improved significantly and we now have far more tools to work with for pretty much anything. I would like to use the numerous different technologies, to create some sort of a web based strategy game, such as JS, HTML5 and CSS as well as others like Node, probably like two dozen different APIs, libraries and packages and also things MySQL/SQLite and possibly C++. I would like to include features like multiplayer, database support and some proper level of complexity to the game, it would be nice to make a program that I could be proud of and that others could enjoy.

Acknowledgements

I would like to thank my family, friends and dogs for their moral support during both this semester, and my overall 3 and a half years of being at CIT.

I would also like to thank my project supervisor John Creagh for guiding me this semester in the creation of this document.

Contents

Abbreviations

JS	**J**ava**S**cript
HTML	**H**yper **T**ext **M**arkup **L**anguage
CSS	**C**ascading **S**tyle **S**heets
NPM	**N**ode **P**ackage **M**anager
SQL	**S**tructured **Q**uery **L**anguage
D3	**D**ata **D**riven **D**ocuments
SVG	**S**calar **V**ector **G**raphics
UI	**U**ser **I**nterface
IO	**I**nput/**O**utput
DB	**D**ata**B**ase
IDE	**I**ntegrated **D**evelopment **E**nvironment
API	**A**pplication **P**rogramming **I**nterface

I dedicate this to my Mother and Grandfather

Chapter 1

Introduction

I plan to make a web based browser game. The purpose of this document is to investigate and research what different technologies can be used for this, as well as the already implemented solutions out there such as open source libraries, etc. I will go trough the various technologies and solutions to see how viable they all are for the purpose of making the project.

1.1 Motivation

Browser based games have always been popular. As a child I remember playing them on ancient computers with dial-up internet, sometimes with friends which was preferable as all things are generally more fun to do with others. As I began learning about web development during my time spent in CIT in the web development course I have learned many things, and this opened my mind to many possibilities of what web based programming could be used for. While trying to come up with an idea for a final year project, I wanted to pick something to further develop my existing skills and knowledge, and I also wanted to design something that I would find interesting and enjoyable to make.

1.2 Contribution

The project I am making is highly relevant to the course that I am doing, as well as the included modules. some of these include

Server Side Web Frameworks: The knowledge learned from this module I will use to manage the backend of the application

Scalable Microservices: Similar to SSWF, this module will help me design a functional backend

Interactive Data Visualization: This will be important for visualising data in the form of a player map

Game Development: In this module I will learn some things about game development, some of which should apply to my project

Front End Frameworks: From this module I will learn how to make a good looking and functional front end

Database Management: these modules will help me create a functional database for the project

Knowledge I have gotten from my internship about software development and web programming will also help me with this project, as I have worked with many relevant technologies, including JavaScript and Node.js

1.3 Structure of This Document

- Introduction
- Background
- Problem
- Implementation Approach
- Conclusion

Chapter 2

Background

Projects such as mine, web based browser games have been relatively popular for a long time, and they have their own niche following on the internet. In this chapter I will examine the various technologies available that could be used for my project, and also existing applications that are similar to what im making.

2.1 Thematic Area within Computer Science

1. The core topic of this project is using Web based Programming to develop a game. I want to use my programming skills to develop a web based multiplayer game.
2. The core area that this falls into is Web Applications. Web applications are a sub section of programming where applications are written to utilize web technology in order to do tasks over a network, usually the Internet.
3. The main area of Computer Science that this fits into is Programming.

2.2 A Review of the Web Applications thematic area

2.2.1 Current state of the art

2.2.1.1 V8 engine

V8 is Googles open source high-performance JavaScript engine, written in C++. It is used in Google Chrome, the open source browser from Google, and in Node.js, among others. It implements ECMAScript as specified in ECMA-262, and runs on Windows 7

or later, macOS 10.12+, and Linux systems that use IA-32, ARM, or MIPS processors. V8 can run standalone, or can be embedded into any C++ application. [1] The V8 is an open source JavaScript engine created by google to improve JS execution inside browsers. It is written in C, C++ and most importantly Javascript and designed for both server-side and client side JS applications. It is currently used in google browsers as well as in the JavaScript runtime environment Node.js The biggest selling point of the V8 JavaScript engine is its performance and efficiency. In order to run more efficiently, the V8 engine converts JavaScript code into more efficient machine code instead of using an interpreter. [2]

V8 uses a method of compiling code that involves compilation to machine code at the moment of execution the main difference between V8 and other engines is that V8 doesnt produce any extra code and instead goes straight to machine code. [3]

The V8 engine can run on its own or it can be embedded into C++ code among other things

2.2.1.2 Node.js

Node.js is a JavaScript runtime built onChrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem,npm, is the largest ecosystem of open source libraries in the world. [4]
The V8 engine is a solid base for Node but just the interpretation of the language we want to use isnt enough to create a proper environment which is supposed to run server side. Besides the V8 engine inside the internal works of Node a big part of the job is played by components created in C and C++ like for example libuv, crypto and http-parser. Node.js has an implemented way of calling those libraries created in other languages trough binding and addon mechanics.

The usual approach to input/output operations like for example reading and saving many languages have a synchronous approach that forces this sort of queue where the program moves forward through the instructions line by line. Because of that there is no way of skipping the moment of waiting.

The creator of node.js criticized this approach in his first presentation about node, he said that for this behavior two things are responsible, the culture of demanding results instantly, which was for a long time considered to be the only good approach to input/output operations but because of that demand for instant result a lot of time was wasted with the wait for the requested result. The other problem being infrastructure

because for a long time creators of programming languages had a wrong approach to asynchronousity, with problems like no alternative to synchronous access to databases because of the lack of things like anonymous functions od callbacks or the amount of complicated possibilities put off programmers.

JavaScript on the other hand as a language used for user interfaces which by definition have to be non-blocking was great for solving problems with input/output.

Working with anonymous functions which also work as callbacks in JS is normal but in other languages there is no possibility to have a function to pass a function as a parameter which would help with asynchronousity. In node it was decided to use all the good parts of the JavaScript language to complete input output tasks in a non-blocking way.

2.2.1.3 Node Package Manager

Node Package Manager or NPM is a package manager for applications created in JavaScript. Currently it is one of the largest code registries and open source works in the world. In NPMs resources you can find known frameworks such as angular and react as well as things like basic single functions that act as utilities for programmers. [5] Early on NPM acted as a manager for packages only designed for server-side works but as time went on it became a tool for front end development. The base part of work with NPM is a package.json file that contains the specification for all the packages and dependencies in a project. It contains a lot of information with things like name of the project, author, licenses, description or version. The most important part though is the list of all the dependencies with objects that maps package names, versions and types. [6]

2.2.1.4 SQLite

SQLite is an SQL database engine. Rather than being a server side or a client side database engine, it is embedded into the application. [7]

Due to its relative ease of use, light weight and functionality it is currently the most used database engine in the world, and its used in anything from web applications to operating systems. [8]

2.2.1.5 D3

D3 or Data Driven Documents is a large JavaScript library used for the purpose of interactive data visualization. It makes use of the widely implemented SVG, HTML5,

and CSS standards, this allows it great control over visual elements. D3 is embedded in JavaScript and it has functions to create, edit and manipulate SVG objects, as well as to harness data and connect it to documents. [9]

2.2.1.6 SVG

SVG or Scalar Vector Graphics is vector based image format. SVG images and objects are defined by XML text files, the main strength is its use of vectors, this allows for loss less image size manipulation. [10]

2.2.1.7 React

React is a front end framework for creating UIs for JavaScript applications, it is created and maintained by Facebook. 'React can be used as a base in the development of single-page or mobile applications. Complex React applications usually require the use of additional libraries for state management, routing, and interaction with an API'. React is a very powerful framework and well suited to creating large applications, though one of its downsides is that its rather heavy.

2.2.1.8 Vue.js

Vue is another front end framework, similar to React its purpose is to create a functional front end for JavaScript applications

2.2.1.9 agar.io

Agar.io is a multiplayer browser game. It is currently one of the most popular on the internet. It was developed by a Brazilian student, and its written in JavaScript and C++. It is notable because it allows for up to a hundred players to join a single session and it works without any issues. Its popularity created a whole series of spin off games inspired by it hosted on the .io domain.

2.2.1.10 History of the British Isles

One of my primary interests is history, so I wanted to inject that into this project. I believe it would make the game more interesting. There are a few time periods and regions that I have considered such as the Alexiad, Crusades, Ancient Rome but I have

decided for the time being to go with Viking age British Isles which is in my opinion a pretty interesting time period. [11]

2.2.2 Academic resources

2.2.2.1 Book resources

- Node.js in Action: This book talks about Node.js, the Runtime enviroment used to execute Javascript code on command line level. This is usually done for the purpose of creating a back end for a web application, such as a server. The books goes into detail about a lot of the functionality that node provides, how its used as well as the different concepts. The book also goes into details about how node works and how it should be used. The knowledge here is important for anyone intending to learn and use node.js for their projects. The relevance to this project is that I intend to use this, and as such it will be the technology responsible for maintaining the entire backend and server side of the application. [12]

- Learning Vue.js: This book talks about the vue.js framework. Vue is a high end frontend framework used in web applications. The purpose of vue.js is to expand on and replace Jquery, an older, very important Javascript framework. This book goes into detail about the different functions of vue, such as how to build responsive applications, and how to use its reactive and reusable components for the purpose of writing great frontends for applications. There are many different JS frameworks, too many to list here, but I want to look at this as it is one of the better and most popular ones. Any serious web application requires the use of a front end framework, and so I intend to use one for my project. [13]

- Practical D3.js: This book is about the D3 JavaScript library, As mentioned earlier, the purpose of D3 is to allow interactive data visualization. The book teaches the reader on how to use D3 to its fullest and create functional visualizations. The game will use D3 for a large chunk of the graphical side of things, one of the more advanced functionality of D3 is the ability to use D3 to create large, scalable and interactive maps, and this is something that I intend to use for this project, it will allow me to have a map that can be interacted with by the user trough user input. [14]

2.2.2.2 Other resources

- LearnCode.academy: LearnCode.academy is a youtube channel about programming and web development. It has over a hundred videos on web based programming, on things ranging from basic HTML and CSS all the way up to things like Node.js and various frameworks such as the previously mentioned vue among other things. It is a great youtube channel, with a lot of knowledge relevant to this area.

- r/programming: r/programming is a large programming community on reddit. With over 1.3 million subscribers its one of the biggest programming communities on the internet with many posts being made every day. As web development is a large subsection of programming, it is a great place to talk about the different web development technologies with many other like minded people with the same skills. It also contanins a ton of information on anything programming related, from book recommendations, trough related blog and website links to dozens of guides on any subject programming related.

- Implying we can discuss programming: another programming related community, this one is present in many locations, such as discord and facebook, its an internet meeting place for programmers where people come together to talk and discuss about anything programming relating, and as such it also has a lot of web developers talking about web development related things.

Chapter 3

Problem - developing a multi player game

3.1 Problem Definition

What I want to achieve with this project is to use my web development skills to write a web based multi player game.

3.2 Objectives

There are three parts to this application:

1. The front end, I must write a front end to the application that will serve as the UI for the player. It must be dynamic and be able to relay the information about the game.
2. The back end, I must create a server application that will handle all the inputs from the player as well as sending and receiving information to and from the client. It will also have to handle multiple players and instances of the game.
3. The game, I have to write an application that will act as the game that will be handled by both the front end and back end

3.3 Functional Requirements

The functional requirements for this projects are:

1. A clear and transparent UI that the user will be able to look at and read information about what is going on. This should read data from the server and client so that the user can see what is happening in the game.
2. The ability to handle user input with a mouse and keyboard. This is because the user must be able to control the game with their inputs.
3. The server must be able to handle multiple players at once. This is intended to be a multiplayer game so this is necessary.
4. The server must also be able to efficiently feed the information to the clients. This is important because there will be a lot of information being sent back and forth.
5. A database is required for information storage.
6. The front end must be able to visualize data in an interactive way for the user.
7. The game hast to actually be something viable as a game that someone could enjoy.

3.4 Non-Functional Requirements

1. The game must be able to be played by at least two people.
2. The server must be able to efficiently send and receive information with low latency. The client must also be able to receive and send information to and from the server efficiently and with low latency. The latency should be no more than 1000ms.
3. The UI must be be dynamic and be able to translate information into something visible for the user.
4. The server must include a database. Preferably embedded within the application itself.

Chapter 4

Implementation Approach

4.1 Architecture

The technologies that will be used in this project include:

- JavaScript
- HTML and CSS
- vue.js
- node.js
- SQLite
- D3

The programming environments that I will use for this project are Webstorm IDE and Notepad++ for writing the actual code, and SQLite studio for the purpose of manipulating and working with the embedded database.

The architecture of the project involves two main parts, the front end and the back end.

4.1.1 Back end

The back end server will be written in Node.js, utilizing technologies such as NPM and SQLite for its own use.

- Express: Express is a node framework that supports web application

- Express router: The router will be used to manage and feed to the user the different HTML, CSS and JS files and the ability to navigate between them.
- RESTful API: this is needed so that the server can handle GET and POST requests from the clients
- SQLite database: the server will include an embedded database, for the purpose of storing configuration and other data

4.1.2 Front end

The back end will be written using HTML, CSS and JS, together with the vue.js framework

- HTML: Markup for the application
- UI: The front end will have a UI that will allow the user to interact with the game
- CSS: Styling for the HTML
- Interface: an interface to interact with the server

4.2 Risk Assessment

To understand the risks that come with doing this project i tried to take a look at the different parts of the application and see what could go wrong. To this end, ive also decided to do some prototyping, which I talk about later in this chapter.

The biggest and most important risk in the project as far as I see it comes with the difficulty that is in implementing the interactive map, it takes a ton of work, and while its certainly doable, it will no doubt take many hours of work.

A somewhat lower risk is the difficulty in programming the communication between the server and clients, it must be done efficiently and without errors, to mitigate the risk I must make sure to implement this in such a way that no data is lost and that the clients and servers validate the success of responses and requests.

Some of the more negligible risks include:

- Security and firewall problems for communication between the clients and server.
- Server crashes or errors.

- Scalability issues.
- Project or implementation delays that can happen at one of the stages.

4.3 Methodology

To understand the depths of developing this project I researched web development and its different technologies. The objective was to create a multi player game that is good and that people can enjoy. During the research phase I looked at other such games that exist out there and looked at their advantages and disadvantages, and came to my own conclusions.

The management approach that I think would work best here is the agile approach. Agile is all about change and can be used during the building stage to ensure all areas are correctly developed. This is good for me, as at some point trough the project I might realize that something large needs to be changed.

I also plan on following a loose month long sprint schedule detailed in the next sub section, however I believe it is necessary for me to be flexible, as certain parts might take significantly more time than expected, and I believe it is important to be prepared for such an event.

4.4 Implementation Plan Schedule

The project has multiple components that must all work together. The order for implementation will be something as follows:

- Implementation of a basic back end system. A 'skeleton' of a node server must be built first, that will include basic routing, a Rest API for handling requests as well as implementation that will allow to embed a database.
- A front end application will be implemented that will include a user interface and the ability to make requests to the server and receive data from the server.
- Creation of a database and filling it with necessary data.
- Development of a full implementation back end of the application, that will have the finalized database embedded within it, as well as (other stuff)
- Front end implementation of the visual side of the game that will work with the user interface

- Back end implementation of the game, the actual programming of the game.

I have roughly about 4 months to complete this project. I would like to finish the first 4 items from this list in the first two months, and then move on to the rest of the project, which will be the implementation of the game itself which will take at least a month. I would also like to leave at least 2 weeks to polish the project, and to have spare time in the case of any delays. I want to be flexible with the work schedule to avoid any unnecessary stress.

The schedule should go something like this:

- Month 1: Implement a back end
- Month 2: Implement a front end
- Month 3: Implement game, combine the front and back end with each other
- Month 4: Finalize everything, bug fixes and testing.

I will have to do at least 3 other modules in the final semester of my course, so I will also have to balance time around that as I have no doubts that the modules will throw plenty of work at me.

4.5 Evaluation

I intend to evaluate the success of the project in a few ways:

- Have some/all the functional and non functional requirements been met?
- Have I managed to implement everything the way I wanted to and in a way that works?
- Is the application bug free and optimized so that it does not throw errors or crash?
- Is the game itself playable?
- Is it enjoyable? I will ask some people to play it and ask them to fill out a short survey on what their opinion is.

The answers to these questions will be important for the purpose of coming to a conclusion at the end of the project, as well as on what would be possible future improvements.

4.6 Prototype

As part of the research phase of the final year project, i have developed a prototype for the purpose of testing the various technologies and their suitability. The primary purpose of the prototype was to develop a proof of concept as well as testing my idea of using an Interactive D3 map for a game. I have included code snippets at the end of this document in their appropriate section.

To this end, I have taken a large JSON data set that contains the vectors for European countries, and then used that data set to try and generate an interactive map. This was a rather difficult task, as its pretty hard to work with such massive data sets with hundreds of vector points for each object. At the very least this proves the concept is viable, and the experience from this will help me to develop this project.

I have also created a simple node server with a RESTful API that can handle GET and POST requests, an Express router and some other things, though this was relatively simple compared to the previous part. The purpose of this was to find the best way of feeding large data sets to the client.

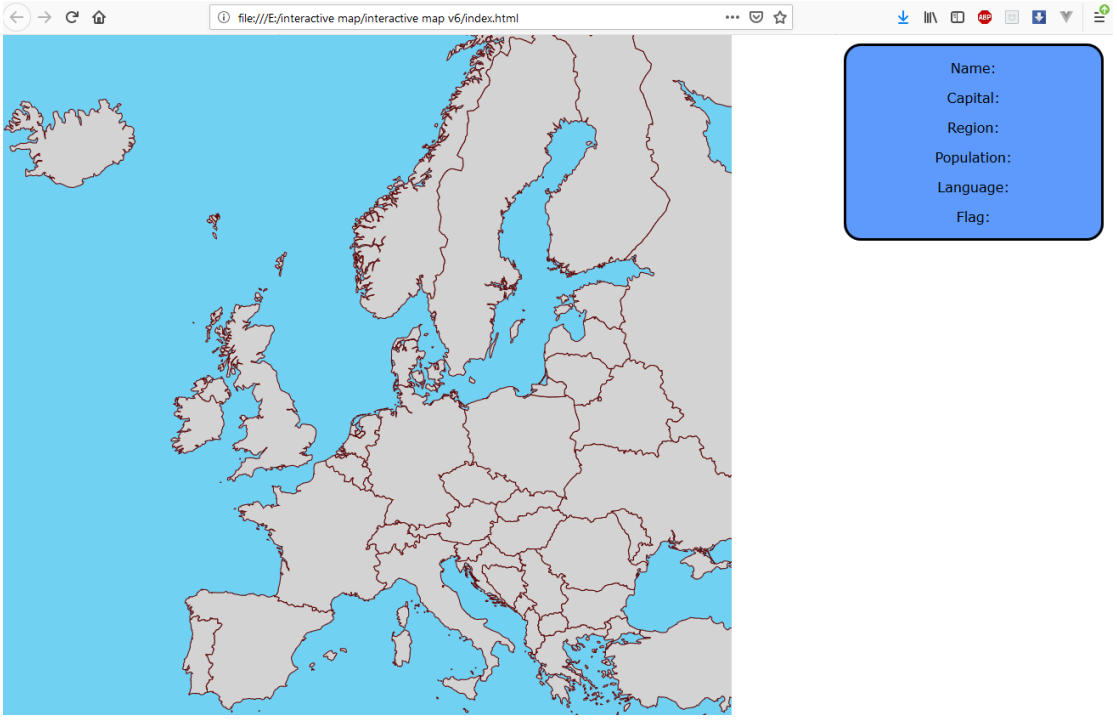


FIGURE 4.1: Loaded map

What this is supposed to demonstrate is the concept that I want to use, the ability to make an interactive map in D3 with clickable SVG objects, and the act of interacting with an object (in this case a country) by clicking on it makes other things happen, in this case fetching a bunch of information about the country from a server.

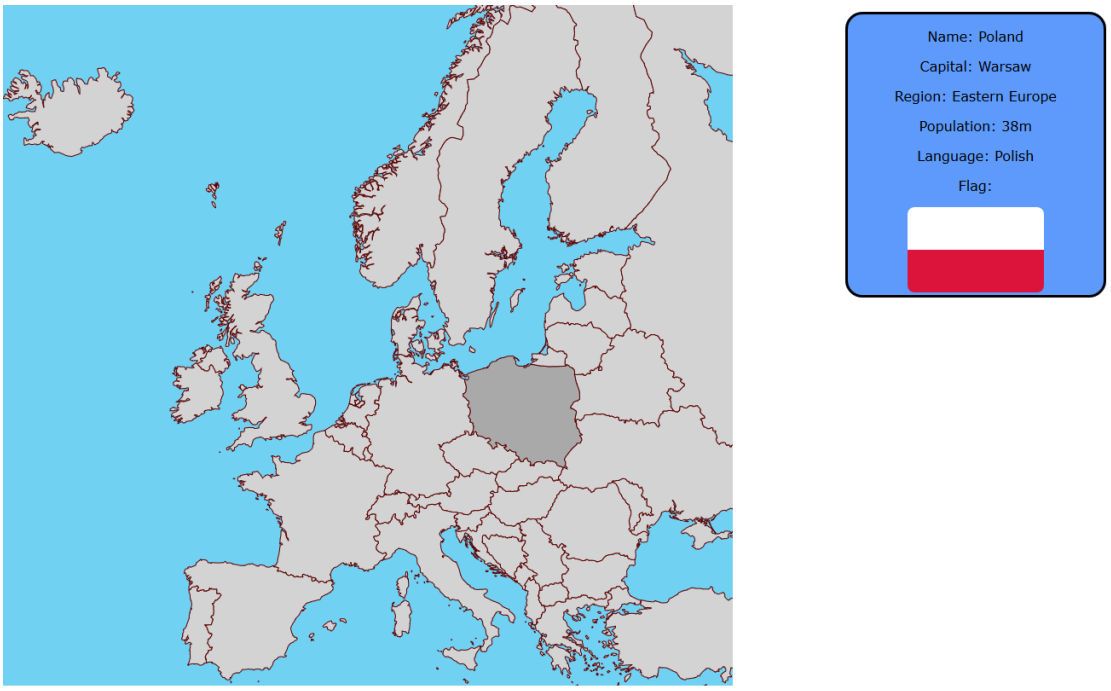


FIGURE 4.2: Selected country

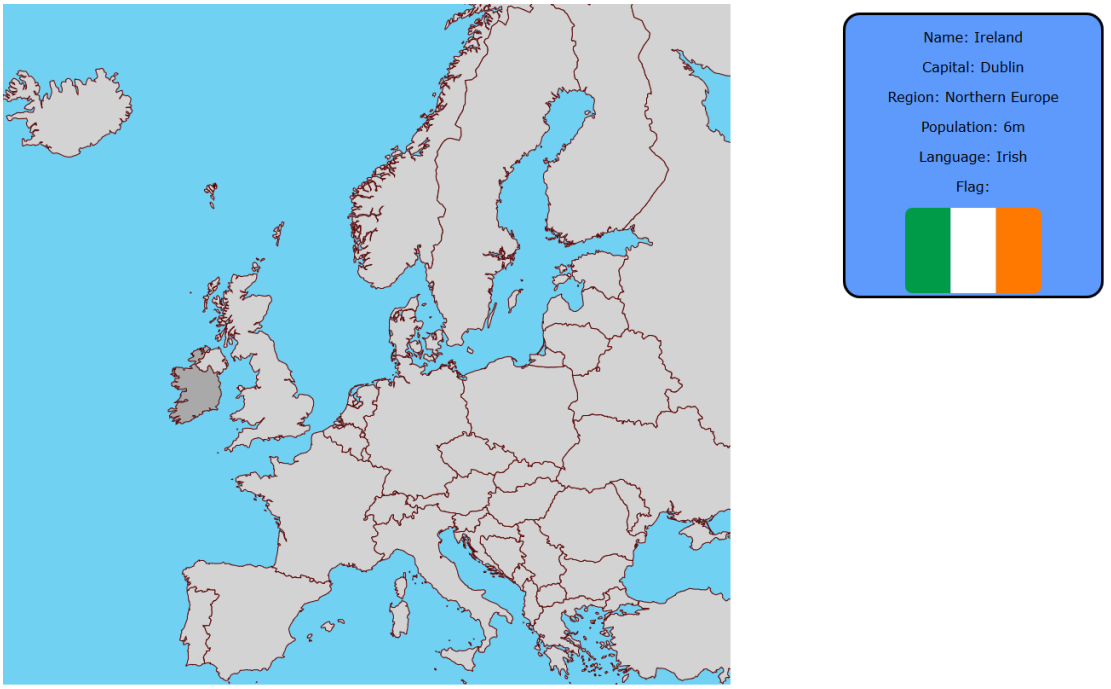


FIGURE 4.3: Selected another country

Chapter 5

Research phase Discussion and Conclusions

5.1 Discussion

The research phase of this project involved a good bit of research and a lot of prototyping.

I first looked at the current 'state of the art' for various web development technologies.

There are a lot of choices for technologies in every part of it, both front end and back end offer many solutions so picking one usually comes down to making a decision as to what is the most suitable one for what the requirements of the project are.

One dilemma i spent a bit of time thinking about is the choice of the technologies for making the front end user interface. Frameworks such as Angular, Vue or React all have different uses and purposes. There is also the choice of just not using one, and instead just using the HTML+CSS+JS combo that is so popular in web development.

A significant problem that ive encountered was during prototyping, the process of creating an interactive map in D3 involves using a large set of arrays, each with SVG vector coordinates, and this is difficult to implement correctly, it took a very long time to get it right in the prototype, so this is something I have to watch out for in the implementation phase.

5.2 Conclusion

The first part of this was to look at the technologies used for server side scripting, the choices were many, from PHP and node to things like Java and C++, but in this case ive

decided to go with node, the primary reasons for this are that it allows for asynchronous execution, and because its the one I am the most skilled with.

The second part was to find an adequate database management system, such as MYSQL, NoSQL or SQLite, here ive decided to go with SQLite as its lightweight nature fits well into this project, and because its embedded into the server application.

The third part was to find a good front end framework to use to design the UI, the choice here was from React, Angular and Vue, the three most popular frameworks, for this ive decided that the best choice is vue, as it is more lightweight and therefore better for smaller projects than the other two, as well as providing some useful functionality such as the vue CLI which makes the debugging process easier and the vue router, and as such, ill probably end up using this.

The fourth part was designing the game itself, i found D3 to be the most useful thing here, one of the more advanced features of this is the ability to create interactive maps with the help of SVG.

5.3 Future Work

An important thing that needs to be added is to host the application on some sort of a server, that would efficiently handle more traffic and handle multiple sessions.

I also believe a big thing that would be good to add to this project would be some sort of an algorithm based AI to control the different factions on the map that arent controlled by any player.

Chapter 6

Implementation

6.1 Solution Approach

One of the early goals on was to assign data to the various provinces on the map and how best to implement it, to solve this I designed multiple object factory patterns that would iterate trough a list of the provinces and create objects based on the data, to tie the object to the province, I assigned both a matching ID, and as a bonus sorted the array so that ID matched the position in the array.

The factory pattern algorythms I have desgined have been made in such a way as to include no hardcoding, and as such, they can be fed any sort of data, this allows for endless modifications of the game, be it adding new nations or changing the map. The game engine I have designed is very flexible and this has been one of my primary goals over the course of the project.

From all the factory pattern created objects I generated a combined 'state' object which would store the current state of the game, using the websockets to send it to the clients and then get it back the same way. I also implemented a turn logic so that the game would run on a turn system.

The turn based system itself isnt a traditional one which you might see in chess or another board game, instead it is a system where all the players take their turn at the time, and when all of them are done, their decisions are sent to the server, which in turn processes the entire turn with things such as comnbat, income or province annexation, and when its done it ends the turn and sends the state object back with the updated information to the clients, whose UI and screen is automatically updated at this point as the next turn starts.

6.2 Difficulties encountered

Here is a list of some of the difficulties I have encountered, sorted by the inconvienience they provided.

6.2.1 Game state

I had to figure out how to store the data about the state of the game. For this I created a series of object factory functions on the server which generate a series of objects using various data that are then put together into a singe 'state' file that is moved around trough the websocket between the server and clients.

6.2.2 Turn system

Another difficulty that I had was implementing the turn system, which as described before allows players to take turns at the same time. The difficulty here was in making sure each players turn gets processed. I solved this by implementing an algorythm that would store each players decisions until the turn is over.

6.2.3 Communication between clients and server

For a little bit I had a mild problem of trying to figure out what is the best way to send data to and from the server, mainly the state of the game. The solution I decided to go with to solve this was to use websockets to send data around.

6.2.4 Centroids

I encountered a pretty severe problem when I tried to implement the centroid for each province. This was supposed to add a point of reference to each province that would be placed in the middle of it. The idea is that I would use that for selecting armies and moving them to other places, but I was not able to implement this. What I did to get around this problem is implement a system where the mouse clicks on provinces are tracked, and that selecting a province also selects its local army if there is one. The effects of this are mostly visual in practice, though its mildly disappointing that I wasnt able to implement this.

6.3 Implementation timeline

6.3.1 Week 1-2

In the first two or so weeks I have worked on prototyping the front end, to this end I have looked at some existing open source games on the internet that are similar to my project:

The first one, called 'Hedgewars' [15] seemed to fit a few of the same categories and there was some information but it was in a different language to what I was using.

Later I found this game called LootRPG [16] which also had an article written about it by its creator that had some interesting and useful information about how he went about things for creating a game in JS. [17]

I also found this piece of technology called PubSubJS [18] which is used for synchronizing data, and I looked at it for suitability in my project.

I have also made a repository for my code for version control. [19]

6.3.2 Week 3-4

In this period I was currently working on the basic server implementation. I made a new branch on github for it called server-test where the current state of the implementation is located. [20]

I branched it from the main project to separate it from the rest of the work as the server was still work in progress, I made a branch for this instead of a new repository because I realized that the server is supposed to be serving the client files to the client anyway, seeing as this is a web application.

Functionality that I worked on implementing:

1. Express with a router
2. RESTful API
3. Embedded SQLite database
4. Web socket

6.3.3 Week 5

In this week I was working on expanding the stuff from the last 2 weeks, notably trying to find the best way to feed files to the client, such as the JSON file with the data for SVG objects. The WebSocket proved itself to be the best way to do this as it supports many clients connecting to the server.

I also started work on trying to get the application to load in a different map from the JSON map files.

6.3.4 Week 6

In this week I worked on the front end side of things, ive made some progress, and ive also asked a lecturer for advice on how to progress.

Notable functionality:

1. Event handlers for the map provinces
2. Loading in different maps
3. Making the code to include as little hardcoding as possible and work with different maps

6.3.5 Week 7-8

At this time I started working a bit on the 'game engine'.

I worked on making serialized object generation on the backend using configuration data. The purpose of this is to create an object for each province on the map so that they can be interacted with, and that they can store various information about it.

These objects will then be sent to and from the client, and the state of each object will be used to implement the multi player system.

6.3.6 Week 9

I didnt do too much this week as I was pretty occupied with the endless and unrelenting stream of assignments at this point in time, so the work done this week was mostly limited to small UI changes and error fixing.

6.3.7 Week 10

This week I worked on finishing the UI layout and some of its functionality, I also implemented code to allow for assigning an owner to each of the provinces, which in turn allows me to display dynamically on the map what provinces are owned by who using colours.

I also worked on implementing a way to get the current state of the game and send it between client and server using JSON files or objects and web sockets. This later enabled me to begin implementing the logic for the turn system.

6.3.8 Week 11

I implemented the turn based system, for this I used mostly the web socket, router, and various node functionality.

I created a system where the clients are capable of sending and receiving the state of the game, which is transferred as a JSON object trough the websockets, The system works by the clients sending their state of the game at the end of their turn, and when all clients have sent in their state object, the server will then process and combine this and send back its own state object to the clients which is used by them to update the local one and visualise the actual state of the game for the players.

6.3.9 Week 12

This week I worked on implementing a movement system for the game, a deceptively hard task to do. I have already implemented event listeners for left and right clicking with the mouse, and this is used to select and tell an army where to go. I also tried to use something called a centroid to calculate a sort of mid point of each province which will be used as a reference for the movement. However this failed, and I created an alternate system based on coordinates that is used to determine what province is being clicked on, if any.

I also created the poster that was required for this project and that will later used for the project open evening and the final presentation.

6.3.10 Week 13 and afterwards

This was the final part of the implementation process.

I created a pretty complex algorithm for determining combat results, it works in three stages; It first checks the relative size of both armies. It then rolls a 'virtual' six sided die to determine the luck of the attacker and finally based on that result it will generate a random number for the purpose of calculating losses, modified by the 'luck roll' earlier on. Should the two armies be the same size or relatively similar in size (no more than 20 percent size difference) it will instead take turns and use the random number generator to assign losses to both armies, and whichever army gets down to 0 first looses the battle.

I also created a function for the purpose of flipping provinces from one owner to another, which usually happens when an army takes a province by force. This is also used when a nation takes another nations capital, which causes an annexation of the entire nation to the attacker. The way this system works is that it modifies the state object and changes owner of the province(s) to whoever takes them.

CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Web Based Multi-player game

Author:
Dominik Bahrynowski
dominik76mm@gmail.com

Abstract

The aim of this work is to make a web based game. Browser games have been highly popular since the beginning of the internet, and since then, the power of computers and the connectivity on the internet has increased dramatically. With this increase, programming technologies and paradigms have also improved significantly and there are now far more tools to accomplish almost any conceivable task. In this work numerous different technologies have been used to create a browser based strategy game, these include languages such as JS, SQL, HTML5 and CSS as well as various libraries and environments such as Node, D3, JQuery and Express. The game uses the board game RISK as a sort of inspiration, but it expands on its concept greatly. The technical functionality of the game includes features such as scalable multiplayer, embedded database, dynamic map generation, turn based system, micro-service based server structure and other things. The developed game captures the complexity of RISK well and includes new user experience features such as an economy based on population, culture and religion mechanics, a more advanced combat system and a reworked turn system. The program I have created is also flexible, and thus allows for easy expansion or change of the playable map, which is generated by a JSON file using D3. It utilises web sockets and a restful API for facilitating of data exchange between the clients and the server.

Technologies used

HTML, JS, node, Vue, SQLite, D3

Research

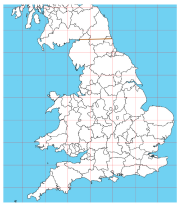
A large part of of the research for this project was done trough various internet sources, such as W3-schools, official documentation or stack overflow.

I also used some book sources like "Node.js in Action", "Learning Vue.js", and "Practical D3.js" which provide valuable insight into the different technologies.

Another very important part of the research process was the use of prototyping, which allows me to gauge what technologies work and how complex the are to implement.

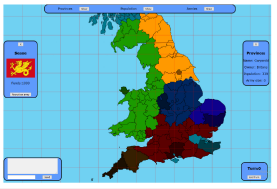
Concept description

The concept of the project was to create a web based multiplayer game. It uses Node as a server-backend for the game, and this communicates with the clients using web sockets and REST. The front end uses D3 to generate an interactive map of Britain from a series of TopoJSON data.



The game is set in the Anglo-Saxon era British isles, and folows a ruleset similar to how the boardgame RISK works but expands on many of the concepts.

The players can interact with the game by left and right clicking on various elements of the map. This can be used to preform actions such as bringing up information about a province, or moving an army to a different area. The game also features a dynamic economy based on elements such as local population



Conclusion

The project has been pretty succesful overall. The functionality and interactivity of D3 is great for the purpose of making a game like this. Web Sockets are also very good for continuous sending of data between multiple clients and the server. Node is also great for the purporse of making servers, as its asynchronous nature makes it easy to write efficient applications, and it is well suited for making any sort of web server.
The use of git version control also made development much easier. I would say this project has been pretty enjoyable to make and I learned a lot in the process.

References:

Github repository: https://github.com/EternalDog/fyp
Linkedin profile:
https://www.linkedin.com/in/dominik-bahrynowski/

Acknowledgements:

I would like to acknowledge Sean McSweeney and John Creagh for their supervision and support during the project

FIGURE 6.1: Poster

Chapter 7

Testing and Evaluation

7.1 Functional requirements fulfillment

To reiterate, the functional requirements for this projects were:

1. A clear and transparent UI that the user will be able to look at and read information about what is going on. This should read data from the server and client so that the user can see what is happening in the game.
2. The ability to handle user input with a mouse and keyboard. This is because the user must be able to control the game with their inputs.
3. The server must be able to handle multiple players at once. This is intended to be a multiplayer game so this is necessary.
4. The server must also be able to efficiently feed the information to the clients. This is important because there will be a lot of information being sent back and forth.
5. A database is required for information storage.
6. The front end must be able to visualize data in an interactive way for the user.
7. The game hast to actually be something viable as a game that someone could enjoy.

Functional requirements evaluation:

1. The first requirement has been fulfilled, there is a UI that reads information provided by the server.

2. The second requirement has been mostly filled, the application takes in left and right mouse button clicks, however I found implementation of keyboard inputs to be redundant.

3. The third requirement has been fulfilled, the server handles multiple clients with ease.

4. The fourth requirement has been filled, the server is able to feed data to the clients trough the web socket.

5. The fifth requirement has not been truly fulfilled, there is in fact an embedded SQL database, however I found it to be redundant and it currently doesnt do anything.

6. The sixth requirement is fulfilled, The application uses D3 very well to visualize data and turning code into actual provinces on the map, which all can be interacted with by left or right clicking.

7. The final requirement has also (subjectively speaking) been fulfilled, the game is certainly in a playable state and I think its something people could enjoy.

7.2 Non-functional requirements fulfillment

Recap of non functional requirements:

1. The game must be able to be played by at least two people.

2. The server must be able to efficiently send and receive information with low latency. The client must also be able to receive and send information to and from the server efficiently and with low latency. The latency should be no more than 1000ms.

3. The UI must be be dynamic and be able to translate information into something visible for the user.

4. The server must include a database. Preferably embedded within the application itself.

Evaluation:

1. The first requirement is filled, the game can theoretically support any number of players, there are 8 nations on the map so 8 players is currently the functional maximum, but for future expansion more could be added and the game would still work.

2. The second requirement is filled, websockets are a very efficient method of sending data to and from the server. On a locally hosted servers there have been no major delay, and the data transfer is practically instant, so in a live environment the only delays would be based on the clients.

3. The third requirement is filled, the UI is pretty transparent and its easy for someone to understand what is going on, while at the same time it also shows a good bit of information.

4. The fourth requirement has not really been filled, while there is an embedded database, I have found it to be redundant so it does nothing.

Chapter 8

Discussion and Conclusions

8.1 Discussion and overview

8.1.1 Research phase overview

The research phase of this project involved a good bit of research and a lot of prototyping.

I first looked at the current 'state of the art' for various web development technologies. There are a lot of choices for technologies in every part of it, both front end and back end offer many solutions so picking one usually comes down to making a decision as to what is the most suitable one for what the requirements of the project are.

One dilemma I spent a bit of time thinking about is the choice of the technologies for making the front end user interface. Frameworks such as Angular, Vue or React all have different uses and purposes. There is also the choice of just not using one, and instead just using the HTML+CSS+JS combo that is so popular in web development.

A significant problem that ive encountered was during prototyping, the process of creating an interactive map in D3 involves using a large set of arrays, each with SVG vector coordinates, and this is difficult to implement correctly, it took a very long time to get it right in the prototype, so this is something I have to watch out for in the implementation phase.

8.1.2 Implementation phase overview

The implementation phase of the project was mostly made up of programming. Early on I still worked on some prototyping and testing of various technologies however I quickly moved on to actual implementation.

One of the challenges early on was to assign data to the various provinces on the map and how best to implement it, to solve this I designed multiple factory pattern objects that would iterate trough a list of the provinces and create objects based on the data, to tie the object to the province, I assigned both a matching ID, and as a bonus sorted the array so that ID matched the position in the array. From all the factory pattern created objects I generated a combined 'state' object which would store the current state of the game, using the websockets to send it to the clients and then get it back the same way. I also implemented a turn logic so that the game would run on a turn system.

8.2 Project and Solution Review

I believe my implementation of the project has been pretty successful. I managed to achieve the vast majority of the functional and non functional requirements, as well as achieved the proposed goal at the start of this document, which is to make a web based game with multiplayer.

8.3 Conclusion

8.3.1 Primary conclusions

I believe the project has been pretty successful overall. The biggest thing for me to take out of this project is experience and improved skills, this ranges from learning of new technologies and improving my existing technical skills, as well as putting to practice skills from my various modules. On top of that there is also the experience i gained in things such as agile development and using version control.

Next big conclusion to take from this is that JavaScript is a very powerful languages, and it is very well suited for this sort of task, there are countless libraries, plugins, environments and such, which allow a skilled developer to create their own games, or any other sort of application. This shows that the state of the technology has improved much from what it was a decade or two ago, where the power of the language and what you could do with it was much more limited.

8.3.2 Secondary conclusions

D3 is a great library, it is perfect for pretty much any sort of data visualization, in this case it was creating interactive maps. It takes advantage of the power of SVG and JSON or CSV stored data to create interactive elements on the screen for the user.

Node.js is a great invention, its asynchronous nature and the syntax it inherits from JavaScript makes it great for server side web development, I personally think its much better than PHP which on the other hand I believe has some of the worst syntax of any programming language.

HTML and CSS are pretty great and have come a long way from what they were a decade or two ago, same goes for browser comatibility for code, which generally isnt much of a problem anymore, when before it was a pretty big issue.

8.4 Future Work

Here is a list of features I would like to implement in the future should I continue working on this project:

1. Increase the map size. There is a pretty distinct lack of Ireland on the map. I could also add places such as France, Northern Germany and Scandinavia. The code I have written isnt hardcoded, and this allows for a pretty easy way of adding new places to the map, new countries, etc.
2. Implement some sort of an algorithm based AI to the game, so that it can be played in Single Player or even have it fill out the nations in multiplayer that aren't played by anyone.
3. Host the game somewhere and implement an instancing system where you could have many people playing on their own separate instances of the game.
4. Add some sort of an icon onto the province to display that there is an army sitting there as opposed to having separate map mode buttons at the top.
5. Improve on the UI design to make it more visually pleasing and informative for the user.
6. Change the combat so that there are different types of units
7. Fully implement the culture and religion mechanics that I started working on but didnt get around to finishing.

8. Implement some sort of a basic Command Line Interface that could be interfaced with by the clients to allow them to do certain things, such as restarting the game, to make things easier.

9. Fully implement a system that allows for saving and loading a game, perhaps by storing the state of the game in the database when its saved and then using that database to access the data when its loaded.

Bibliography

[1] *V8 Documentation*, https://github.com/v8/v8/wiki.

[2] T. Laurens, "How the v8 engine works," 2013.

[3] A. Zlatkov, "How javascript works: inside the v8 engine + 5 tips on how to write optimized code," 2017.

[4] *Node.js Documentation*. [Online]. Available: https://nodejs.org/en/

[5] *NPM Documentation*. [Online]. Available: https://www.npmjs.com

[6] *NPM - Package.json documentation*. [Online]. Available: https://docs.npmjs.com/files/package.json

[7] "Sqlite website." [Online]. Available: https://www.sqlite.org/about.html

[8] *SQLite documentation*. [Online]. Available: https://www.sqlite.org/docs.html

[9] *D3 documentation*. [Online]. Available: https://github.com/d3/d3/wiki

[10] "Svg documentation."

[11] "History of the british isles."

[12] B. Meck, A. Young, and M. Cantelon, *Node.js in Action*. Manning Publications, 2017.

[13] O. Filipova, *Learning Vue.js 2*. Packt Publishing, 2016.

[14] R. S. Tarek Amr, *Practical D3.js*. Apress, 2016.

[15] *Hedgewars github repository*. [Online]. Available: https://github.com/hedgewars/hw

[16] *Lootrpg github repository*. [Online]. Available: https://github.com/RobertSkalko/LOOT-RPG-v1.0

[17] R. Skalko, "Lootrpg article," 2016. [Online]. Available: https://medium.freecodecamp.org/learning-javascript-by-making-a-game-4aca51ad9030

[18] *PubSubJS github repository.* [Online]. Available: https://github.com/mroderick/PubSubJS

[19] *GitHub Repository.* [Online]. Available: https://github.com/EternalDog/fyp/

[20] $Server_testrepositorybranch.[Online].Available$:

Appendix A

Code Snippets

```
    response = data;
    $("#name").html("Name: " + data[0].name);
    $("#capital").html("Capital: " + data[0].capital);
    $("#region").html("Region: " + data[0].subregion);
    if (data[0].population > 1000000){$("#population").html("Population: " + Math.round(data[0].population / 1000000) + "m");}
    else {$("#population").html("Population: " + (data[0].population));}
    $("#language").html("Language: " + data[0].languages[0].name);
    $("#flag").attr("src", data[0].flag);
  });
//  changeElement(response);
}

function changeElement(data) {
  $("#name").html(data[0].name);
  $("#capital").html(data[0].capital);
  $("#region").html(data[0].subregion);
  $("#population").html(data[0].population);
  $("#language").html(data[0].languages[0].name);
}

function drawMap() {
  var countries, height, path, projection, scale, svg, width;
  var width = 860;
  var height = 800;
  var center = [6, 68.6];
  var scale = 800;
  projection = (d3.geoMercator().scale(scale).translate([width / 2, 0]).center(center));
  path = (d3.geoPath().projection(projection));
  svg = (d3.select('#map').append('svg').attr('height', height).attr('width', width).style('background', '#71d1f2'));
  countries = svg.append("g");
  d3.json('europe.json', function(data) {
    countries.selectAll('.country')
    .data(topojson.feature(data, data.objects.europe).features)
    .enter()
    .append('path')
    .attr('class', 'country')
    .attr('onclick' , 'nationPressEventHandler(this)')
    .attr('d', path)
    return;
  });
```

```
1  // R00137275
2  // Dominik Bahrynowski
3
4  var express = require("express");
5  var app = express();
6  var path = require("path");
7  let port = 3000;
8  app.use(function (req, res, next) {
9  res.setHeader('Access-Control-Allow-Origin', '*');
10 res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE');
11 res.setHeader('Access-Control-Allow-Headers', 'Content-Type');
12 res.setHeader('Access-Control-Allow-Credentials', true);
13 next();
14 });
15
16
17 app.use('/static', express.static('static'));
18
19 var router = express.Router();
20
21 router.get('/', function(req, res) {
22     res.sendFile(path.join(__dirname+'/index.html'));
23 });
```

FIGURE A.1: Node.js server snippet

```
▶ <path class="country" onclick="nationPressEventHandler(this)"
d="M609.4780093932989,534.870268885136L609.2687801394654,533.34552378555549L608.8

PL

▼ (1) [_]
  ▶ 0: Object { name: "Poland", alpha2Code: "PL", alpha3Code: "POL", … }
    length: 1
  ▶ <prototype>: Array []

pop: 38437239

▶ <path class="country" onclick="nationPressEventHandler(this)"
d="M242.48989816968935,771.7644276967275L242.6991274235226,772.9335303167204L242

PT

▼ (1) [_]
  ▶ 0: Object { name: "Portugal", alpha2Code: "PT", alpha3Code: "PRT", … }
    length: 1
  ▶ <prototype>: Array []

pop: 10374822
```

FIGURE A.2: SVG object in HTML

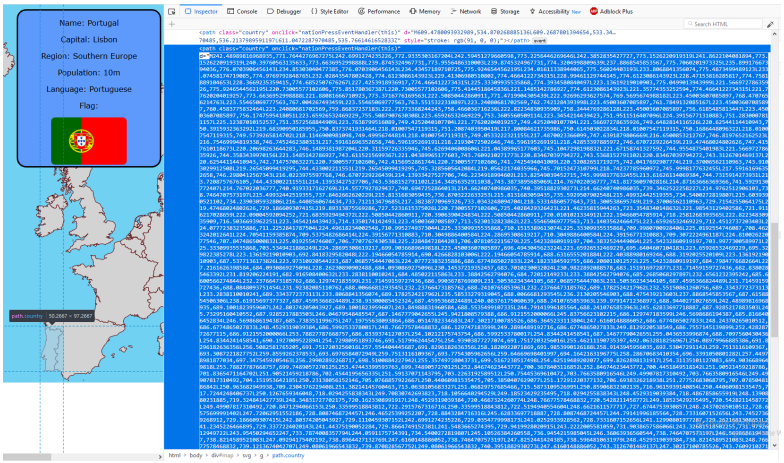


FIGURE A.3: SVG object in HTML with vector points

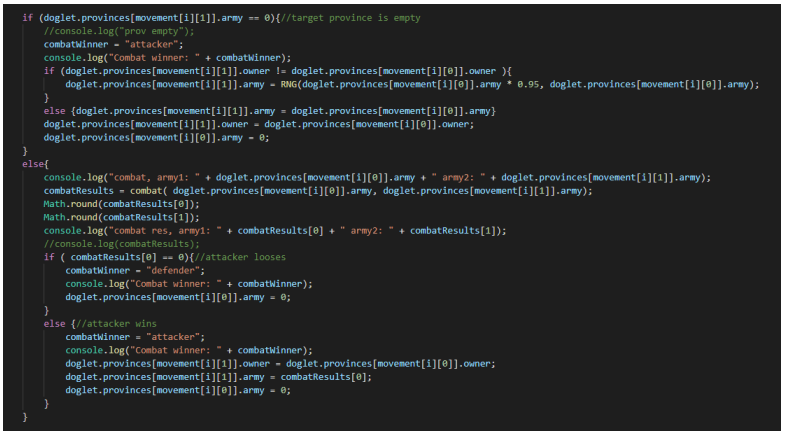
```
function combat(army1, army2){
    if ( army1 >= army2 ){
        console.log("army1 >= army2");
        if ( (army1) > (army2 * 10) ){
            return [army1, 0];
        }
        else if ( (army1) > (army2 * 4) ){
            if (roll_d(6) == 1){return [ RNG( army1 * 0.75, army1 ), 0];}
            else {return [ RNG( army1 * 0.9, army1 ), 0];}
        }
        else if ( (army1) > (army2 * 2) ){
            if (roll_d(6) == 1){return [ RNG( army1 * 0.5, army1 ), 0];}
            else {return [ RNG( army1 * 0.7, army1 ), 0];}
        }
        else if ( (army1) > (army2 * 1.5) ){
            if (roll_d(6) == 1){return [ RNG( army1 * 0.2, army1 ), 0];}
            else {return [ RNG( army1 * 0.6, army1 ), 0];}
        }
        else if ( (army1) > (army2 * 1.2) ){
            if (roll_d(6) == 1){return [ RNG( army1 * 0.05, army1 ), 0];}
            else {return [ RNG( army1 * 0.3, army1 ), 0];}
        }
        else if ( (army1) >= (army2 * 1) ){
            //console.log("even fight");
            while (army1 > 1 && army2 > 1){
                army1 = RNG(0, army1 * 0.5);
                if (army1 < 1){army1 = 0}
                army2 = RNG(0, army2 * 0.5);
                if (army2 < 1){army2 = 0}
            }
            return [army1, army2];
        }
    }
```

FIGURE A.4: Part of the combat algorithm

```
function annex(nation1, nation2){
    var n2 = doglet.nations[nation2];
    var prvs = doglet.nations[nation2].provinces;
    console.log(nation1, nation2);
    (doglet.nations[nation1].provinces).concat(prvs)
    for (let index = 0; index < prvs.length; index++) {
        if (doglet.provinces[prvs[index]].owner != nation1){
            doglet.provinces[prvs[index]].army = 0;
        }
        doglet.provinces[prvs[index]].owner = nation1;

    }
```

FIGURE A.5: Annexation function



```
if (doglet.provinces[movement[i][1]].army == 0){//target province is empty
    //console.log("prov empty");
    combatWinner = "attacker";
    console.log("Combat winner: " + combatWinner);
    if (doglet.provinces[movement[i][1]].owner != doglet.provinces[movement[i][0]].owner ){
        doglet.provinces[movement[i][1]].army = RNG(doglet.provinces[movement[i][0]].army * 0.95, doglet.provinces[movement[i][0]].army);
    }
    else {doglet.provinces[movement[i][1]].army = doglet.provinces[movement[i][0]].army}
    doglet.provinces[movement[i][1]].owner = doglet.provinces[movement[i][0]].owner;
    doglet.provinces[movement[i][0]].army = 0;
}
else{
    console.log("combat, army1: " + doglet.provinces[movement[i][0]].army + " army2: " + doglet.provinces[movement[i][1]].army);
    combatResults = combat( doglet.provinces[movement[i][0]].army, doglet.provinces[movement[i][1]].army);
    Math.round(combatResults[0]);
    Math.round(combatResults[1]);
    console.log("combat res, army1: " + combatResults[0] + " army2: " + combatResults[1]);
    //console.log(combatResults);
    if ( combatResults[0] == 0){//attacker looses
        combatWinner = "defender";
        console.log("Combat winner: " + combatWinner);
        doglet.provinces[movement[i][0]].army = 0;
    }
    else {//attacker wins
        combatWinner = "attacker";
        console.log("Combat winner: " + combatWinner);
        doglet.provinces[movement[i][1]].owner = doglet.provinces[movement[i][0]].owner;
        doglet.provinces[movement[i][1]].army = combatResults[0];
        doglet.provinces[movement[i][0]].army = 0;
    }
}
```

FIGURE A.6: Movement function

```
function poulateProvinceArray(){
    var dog;
    var owner;
    var obj = config.nations;

    for (i = 0; i < config.provinces; i++) {
        owner = "";
        dog = provinceData.objects.UK.geometries[i].properties;
        for (i2 = 0; i2 < Object.keys(config.nations).length; i2++){
            if (owner != ""){break}
            for (var prop in config.nations) {
                if ((obj[prop].provinces).indexOf(i) != -1){
                    //console.log("province " + i +" true, owner: " + prop)
                    owner = prop;
                }
                //else {console.log("province " + i +" false")}
            }
        }
        //if (owner == ""){console.log("province " + i +" false")}
        provinces.push(new province(i, dog.NAME_2, dog.NAME_1, owner));
    }
    provinces.push(new province((provinces.length), "London", "England", "Lutae"));
    state.provinces = provinces;
}
```

FIGURE A.7: Object factory

Appendix B

Wireframe Models

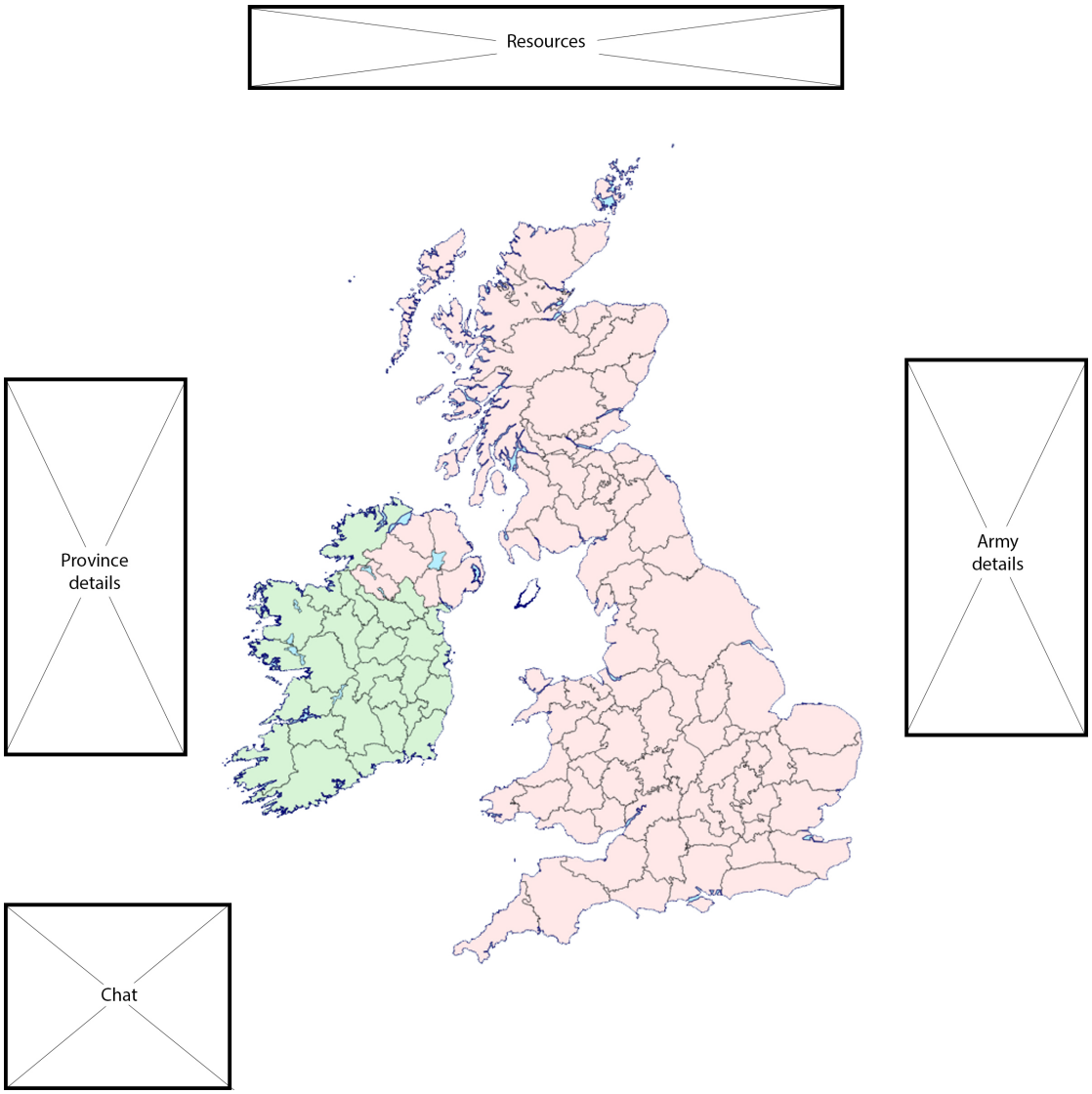


FIGURE B.1: Concept Wireframe