Politico - Literature Review

Joshua Jackson – P16179167

Introduction

This document is a review of the relevant literature for my final year project; It will cover what games have inspired the project and what elements have been taken from said games. It will also discuss games on the web as a means of being cross platform and the technology choices made for such games and the project as a whole. Finally, the document will cover the potential use of politics and fuzzy logic within the game with references to my prior research into the subject.

Similar Games

The idea behind this project is very simple and so where gameplay ideas have been taken from other games it is usually very small scale and requires discussion as it is more nuanced than a like for like copy.

The biggest inspiration for the project is the mobile game Reigns **[1]** that first implemented the idea of having a Yes/No decision be the only mechanic that a player can interface with. The biggest difference is that, in Reigns, the effects of a decision lie entirely within that decision as opposed to being a combination of different factors which is limiting in terms of being able to look at different areas within the game to inform any decision.

Alongside Reigns, another inspiration for the project is the board game Diplomacy **[2]** that utilises politics and the idea of making political decisions that affect different areas within the nation the player controls where a core tenet of the game is that beyond the initial start-up, there is no real randomness to the game, something I hope to partly take through my use of fuzzy logic (see section on Politics and Fuzzy Logic)

Some other notable mentions would be the Victoria **[3]** and Civilization **[4]** series where the idea of multiple parts of a nation making up an overall outcome comes from and how the player can see and interact with the individual parts.

Games on the Web and Cross Platform Games

Early games on the web used technologies such as Flash and Java Applets which required the user to have external plugins/extensions installed on their machines in order to run them. Some of the most popular games for the web such as Fancy Pants Adventure **[5]** or Line Rider **[6]** were made using the technologies. These games weren't able to run in every browser as the proprietary technology wasn't a browser standard, this started to change slowly through the introduction of HTML5 which included a Canvas that could be painted to, overtime this has phased out the use of external technology and now most web browsers fail to support the outdated technology such as Flash player.

Although games were thriving early on after the introduction of HTML5, they were only really cross platform in the sense that they could be played on any device with a browser but it wouldn't be a comparable experience, many games just serving the desktop site with no support for touch controls. Some web games that decided to focus on being cross platform ditched the browser altogether and used a web wrapper such as Adobe AIR **[7].**

The focus of this project is to be completely cross platform whilst being in the browser, accessed in exactly the same way on all platforms, designed to be responsive and scale to any screen size with intuitive controls that work on all platforms without the need to query the platform. This will be driven through the use of mobile first design **[8]** that scales up from a mobile screen to any sized

screen and will be aided through the use of modern technology that is designed to scale and last over time (see section on Technology).

Politics and Fuzzy Logic

The main theme of the game revolves around making political decisions that have various effects on the nation's attributes. As a minimum viable product, these effects could be randomly decided, but a more robust gameplay experience should make use of advanced techniques and, from my previous research, fuzzy logic within the domain of politics, especially in non-crucial scenarios such as a video game is perfectly suited and can produce impressive results.

To give a bit of background, the idea behind the project is that you make a decision that sits somewhere on the political spectrum (Left – Centre – Right etc) and then, depending upon how the different areas within the nation sit on the same spectrum, the player's yes/no decision will have an effect on the attributes of the nation. Without the use of fuzzy logic, the nation's political leaning could be decided at random, but with the use of fuzzy logic we can split up a nation into regions and then, for each region, give it various factors such as number of universities, average salary etc and use a fuzzy inference system to determine, accurately, where that region should lean on the political spectrum.

This is a topic which has a lot of research behind it, and from my own research I have found that there are many areas in which Fuzzy Logic can be applied to politics outside of just determining which way an area will land on the political spectrum. Some of these include using a fuzzy inference system that uses details about a politician to determine the likelihood they would win an election **[13]**, determining the process a political leader could take when making a decision **[14]** or improving candidate selection within a political party by creating a fuzzy inference system that mimics intuitive decisions about candidate selection **[15]**. The conclusion reached through research and testing of a fuzzy inference system within this domain is that "the system can be used within less serious situations such as within Video Games that feature the simulation of political leaning where accuracy isn't important, but the use of a fuzzy system instead of randomly assigning political leaning would add more depth to the game or simulation."**[16]**

Outside of Fuzzy Logic, examples of other games that introduce politics as a main theme are the Tropico Series **[17]** and then an example of a game that deal with modern politics like this project is For a Better Country **[18]**.

Technology

Technology has come a long way from Flash and Java Applets to modern HTML5, JavaScript and SVG/Canvas and so it is important to look at the available options for a project of this kind and why using a certain technology is beneficial.

This project uses React and TypeScript; React **[9]** is a framework for JavaScript that allows you to write declarative components that output as HTML on a page and TypeScript **[10]** is a super-set of JavaScript that adds static typing to the language. React is designed for user interfaces and so games that heavily rely on other elements aren't suited to this platform and the use of the HTML5 Canvas is probably more suitable whereas games such as strategy games or games that have a high focus on their user interface can take advantage of React and its ecosystem. An example of a game made using React is The Danger Crew **[11]** and an example of TypeScript's use within game development can be seen with the Excalibur game engine. **[12]**

The use of React also brings with it a Component driven approach by default **[19]** where each UI element is broken down into its components that take properties to determine how it should look and behave.

Bibliography

1. Reigns Game - https://reignsgame.com/reigns/ (Last accessed 19th December 2019)
2. Diplomacy Board Game Rulebook - http://www.diplomacy-archive.com/resources/rulebooks/2000AH4th.pdf (Last accessed 19th December 2019)
3. Victoria Game Series by Paradox Interactive - https://www.paradoxplaza.com/victoria-ii/VAVA02GSKvic2001.html (Last accessed 19th December 2019)
4. Civilization Game Series - https://civilization.com/en-GB/ (Last accessed 19th December 2019)
5. Fancy Pants Adventure World 1 by Brad Borne - https://www.bornegames.com/games/fpa-world-1/ (Last accessed 19th December 2019)
6. Line Rider - https://www.linerider.com/ (Last accessed 19th December 2019)
7. Adobe AIR - https://www.adobe.com/uk/products/air.html (Last accessed 19th December 2019)
8. Mobile First Design - https://darwindigital.com/mobile-first-versus-responsive-web-design/ (Last accessed 19th December)
9. React Documentation - https://reactjs.org/ (Last accessed 19th December 2019)
10. TypeScript Documentation - https://www.typescriptlang.org/ (Last accessed 19th December 2019)
11. The Danger Crew - https://thedangercrew.com/ (Last accessed 19th December 2019)
12. Excalibur TypeScript Engine - https://excaliburjs.com/ (Last accessed 19th December 2019)
13. Election Results Prediction System based on Fuzzy Logic by Harmanjit Singh, Gurdev Singh and Nitin Bhatia - https://www.ijcaonline.org/archives/volume53/number9/8450-2245 [Last Accessed: 02/12/19]
14. Fuzzy Decision Making in Politics: A Linguistic Fuzzy-Set Approach (LFSA) by Badredine Arfi- https://www.researchgate.net/publication/31402550_Fuzzy_Decision_Making_in_Politics_A_Linguistic_Fuzzy-Set_Approach_LFSA [Last Accessed: 02/12/19]
15. A Proposed Model for Candidate Selection Process in Political Parties Based on Fuzzy Logic Methodology by Yılmaz Gökşen, Onur Doğan and Mete Eminağaoğlu - https://www.mdpi.com/2297-8747/17/2/152 [Last Accessed: 02/12/19]
16. Using a Fuzzy Inference System to Determine the Political Leaning of an Area (Page 10) by Joshua Jackson – [Referencing my own work submitted to TurnItIn]
17. Tropico Series - https://www.kalypsomedia.com/uk/games/tropico-6/ (Last accessed 20th December 2019)
18. For a Better Country on Steam - https://store.steampowered.com/app/1131420/For_a_Better_Country/ (Last Accessed 20th December 2019)
19. React Components and Props - https://reactjs.org/docs/components-and-props.html (Last accessed 20th December 2019)

Politico - Functional Requirements

Joshua Jackson – P16179167

Introduction

This document will look at the functional requirements of my final year project from the point of view of potential users. I will identify system users by their potential playstyle which will be used to outline what functionality needs to be in place to satisfy the requirements of any given playstyle.

As the project is a game and not a piece of software, modelling a certain kind of user is more difficult and as the only change to approach is through playstyle so, instead of modelling users and their use cases, I will be modelling specific playstyles.

Overall Functional Requirements

As this project is a game, by default there are some overall functional requirements that are shared among all users no matter their specific playstyle. These are outlined below:

- The user must be able to start a new game.
- The user must be able to save an ongoing game (either auto-save or manual).
- The user must be able to load/continue a game from an existing game save.
- The user must be able to progress to the next decision by clicking a button within the game (next turn button).
- The user must be able to make a Yes/No decision that affects their overall attributes.
- The user must be able to view information about any region in their nation.
- The user must be able to view how many decisions they have currently made.
- The user must be able to view the status of all their attributes.
- Upon losing the game, the user must be able to exit back to the start screen.
- Upon winning the game, the user must be able to exit back to the start screen.
- The user should be able to switch to a mobile device or a desktop device and have the same gameplay experience.

User Types

User who prefers a slow playstyle

A slow playstyle will usually be taken on by a user who has played strategy games before and likes to know *how* to play a game instead of randomly clicking around and figuring things out by chance.

Functional Requirements:

- A how-to guide upon starting the game

User who follows politics

A user who follows politics will most likely view the game by how accurate it is when it comes to political decisions.

Functional Requirements:

- List of decisions to be informed by real-world data

User who prefers a fast playstyle

A fast playstyle is most likely going to be adopted by most users which entails clicking around the game until they figure out how it works.

Functional Requirements:

- Exclude long-winded explanations as to how the game works unless the user has specifically triggered it.

Politico - Test Plan

Joshua Jackson – P16179167

Introduction

This document discusses the general approach taken to testing through the use of Test Driven Development and outlines the detailed test plan taken to ensure the application works as expected.

Test Driven Development

This project is being developed with testing as the most important factor as when an application/game gets large, it becomes increasingly more difficult to solve issues and to have confidence in the codebase when developing new features or changing an existing feature.

Test Driven Development (TDD) is the act of writing code in such a way so that it can be easily tested which, in turn, leads to more maintainable code that is also easier to reason about as it has to be verbose in order for tests to be written for it. This means that if functionality can be written more concisely so that it is harder to reason about, the preferred way of writing code will eventually occur as tests will need to be written which proves difficult when code is vague and hard to understand.

Unit Tests

The most common tests are unit tests whereby the smallest unit of functionality can have a test written for it that runs the functionality and checks that the result is as expected if the functionality is working; These tests are all then ran before new code can be committed, ensuring that old code isn't being broken by the new changes.

End To End Tests

The other kind of tests are end to end tests (E2E) which test functionality more literally by carrying out actions that a user would. These can be carried out within the code like unit tests using libraries such as Selenium, but can also be carried out by an actual person following a given test plan (outlined on the next page[s])

User Testing

Alongside TDD, a big part of the TDD methodology being followed will involve external user testing by users following a given test plan and giving feedback on their results, ensuring that they match the expected results and setting tasks for future sprints to resolve any issues the users' faced. Ideally this user testing would take place at the end of each sprint where new functionality was introduced, but would most likely take place at the end of an entire epic as it may be hard for people to find the time. As the developer, I will also carry out this testing myself.

Test Plan

The following section of the document provides a template table of test cases that a user would fill out whilst testing the project. It also contains some tests that have been completed by myself to demonstrate how the table will be used.

Test Case Table Template

The following table is a template to be followed by anybody who tests the game. […] is used as a replacement for user input.

Tester: […]	**Date of Test: […]/[…]/[…]**	**Device Used: […]**
Functionality being tested	**Expected outcome**	**Actual outcome and Comments**
Starting a new game	When the user launches the game they should be able to click a Start New Game button which will launch the game view.	[…]
Loading an existing game	Once the user has started a new game, upon refreshing the browser window, they should be presented with the option to continue that game.	[…]
Rendering the Game View	When the user has started a new game or continued an existing game, they should be presented with a game view that shows the following: • A collection of attributes. • A map of the UK. • A "next turn" button. • A turn counter showing the number of turns that have passed.	[…]
Responsive Design	When the user resizes their browser window the user interface within the game should all scale and be usable.	[…]
Making a Decision	When the user clicks the Next Turn button, a modal should popup with a decision on it, allowing them to respond Yes or No. When the user responds to the decision, their attributes at the top of the screen should be adjusted accordingly.	[…]
Triggering the "End Screen" by losing	When the user's has made enough decisions for any of their attributes to fall below 0, an "End Screen" should appear on the screen giving the user the ability to exit back to the start screen and showing them the following: • The number of decisions they made during their playtime • The last known state of each of their nation's attributes.	[…]
Triggering the "End Screen" by winning	When the user has made their maximum number of decisions and	[…]

	still has each attribute above 0, an "End Screen" should appear on the screen giving the user the ability to exit back to the start screen and showing them the following: • The last known state of each of their nation's attributes	
Exiting back to the start screen	When the user has either won or lost the game, they should be able to click and "Exit" button which should take them to the start screen that only has the option to start a new game.	[…]
Hovering over a region on the map	When the user is using a mouse and hovers over a region on the map, that region should be highlighted to show the user that they can interact with it.	[…]
Viewing information about a region on the map	When the user clicks on a region on the map, a modal should pop up showing them the following: • The population of the region • The statistics of the region that would influence how it leans on the political spectrum (These are subject to change and so haven't been listed in the test plan) It should also give the user the option of closing the modal which should return them to the normal game view.	[…]

Test Results

The following table is an example of a user test performed by myself. Every bit of functionality that currently exists in the game is tested. This should also be replicated by end to end tests as mentioned on page 1 and any new functionality should be added to the table to ensure it is always up to date.

Tester: Joshua Jackson	**Date of Test:** 22/12/2019	**Device Used:** Windows 10 Laptop using the Chrome web browser
Functionality being tested	**Expected outcome**	**Actual outcome and Comments**
Starting a new game	When the user launches the game they should be able to click a Start New Game button which will launch the game view.	Clicking the New Game button launched the game view.
Loading an existing game	Once the user has started a new game, upon refreshing the browser window, they should be presented with the option to continue that game.	After starting a new game, refreshing the browser window prompted me with the option of continuing my previous game.
Rendering the Game View	When the user has started a new game or continued an existing game, they should be presented with a game view that shows the following: • A collection of attributes. • A map of the UK. • A "next turn" button. • A turn counter showing the number of turns that have passed.	The game view was rendered as expected however the map of the UK seemed a bit small which left a lot of space.
Responsive Design	When the user resizes their browser window the user interface within the game should all scale and be usable.	When resizing the browser window, it resized successfully all the way down to a 320px device.
Making a Decision	When the user clicks the Next Turn button, a modal should popup with a decision on it, allowing them to respond Yes or No. When the user responds to the decision, their attributes at the top of the screen should be adjusted accordingly.	When clicking the next turn button I was presented with a modal that allowed me to click Yes or No. When I clicked Yes or No (tried with both) the modal closed, my turn count increased and my attributes were modified slightly.
Triggering the "End Screen" by losing	When the user's has made enough decisions for any of their attributes to fall below 0, an "End Screen" should appear on the screen giving the user the ability to exit back to the start screen and showing them the following: • The number of decisions they made during their playtime • The last known state of each of their nation's attributes.	Upon making enough decisions to get a 0 on my financial attribute I was presented with an end screen that showed that I had lost and my relevant attributes/statistics.
Triggering the "End Screen" by winning	When the user has made their maximum number of decisions and still has each attribute above 0, an "End Screen" should appear on the screen giving the user the ability to	Upon making enough decisions to win the game I was presented with an end screen that showed that I had won the game.

	exit back to the start screen and showing them the following: • The last known state of each of their nation's attributes	
Exiting back to the start screen	When the user has either won or lost the game, they should be able to click and "Exit" button which should take them to the start screen that only has the option to start a new game.	Exiting back to the start screen in both of the above scenarios worked as expected.
Hovering over a region on the map	When the user is using a mouse and hovers over a region on the map, that region should be highlighted to show the user that they can interact with it.	Hovering over any region on the map highlighted said region a colour.
Viewing information about a region on the map	When the user clicks on a region on the map, a modal should pop up showing them the following: • The population of the region • The statistics of the region that would influence how it leans on the political spectrum (These are subject to change and so haven't been listed in the test plan) It should also give the user the option of closing the modal which should return them to the normal game view.	When clicking on most of the regions, it worked as expected, I got a popup that showed the expected details about each region I clicked on and then when I clicked the back button the game view returned to normal. However when clicking on Ireland the next turn button disappeared (as with the other regions) but no region popup appeared.

Politico – System Design Documentation

Joshua Jackson – P16179167

Introduction

This document will cover the system design for my final year project. I will be documenting the architectural approach I have taken as well as how that looks in the real-world with a discussion on class/component design alongside UML diagrams where applicable. Finally, I will show mockups of the user interface as well as a brief discussion around how certain parts of the designs can be made as React components.

Architectural Approach

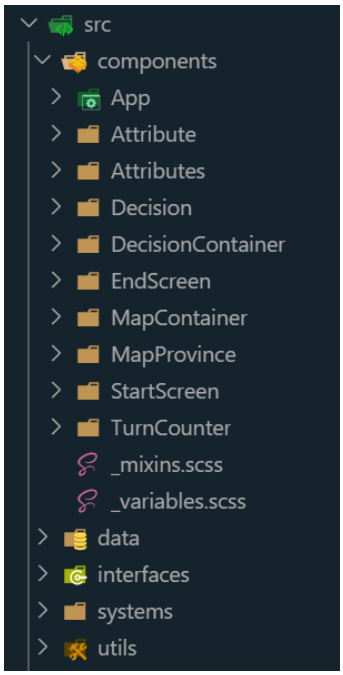
The architectural approach taken is heavily based around having a component driven application where each piece of the UI is split into composable components that are initialized from within a root App component, allowing each piece of the UI to be manipulated individually without requiring a knowledge of the entire application; To aid with this, “Pure” components are used as much as possible which means that the component has no internal functionality, it only displays HTML onto the page and the dynamic part of the component is achieved by passing properties into it, this results in easily testable and maintainable components that don’t act as black boxes. Components are kept within their own folder that contains a .TSX file (React) and a .SCSS file (Styling) and every component follows the same structure, where a component has a test, a .test file will also exist keeping a component truly independent from other components.

Where a component isn’t pure and is linked to another part of the application, TypeScript classes are used to make sure that connection isn’t just obvious to the developer but it also obvious to the compiler.

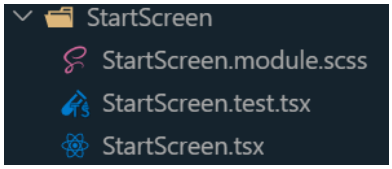
Functionality that utilizes an API are abstracted away via the use of API-agnostic TypeScript interfaces that make it easy to switch to another API if required without ever needing to update the code that uses the API. This is done by having a Systems class that initialises the API-agnostic systems that, if required, can just be swapped out in only that location. An example of this is the DataStorage class, it is an interface that only has get/set functionality for key/value pairs, the game uses the DataStorage interface but the Systems class intialises this as a browser LocalStorage implementation of that interface. If the game was to use cloud storage, all that would need to happen would be to implement a class with get/set functionality that calls out to the cloud storage provider under the hood without requiring any changes to existing code that relies on DataStorage.

Finally, wherever non component-specific functionality can be abstracted out into a reusable utility, the opportunity is taken as it aids with unit testing and keeping everything maintable.

Folder Structure:

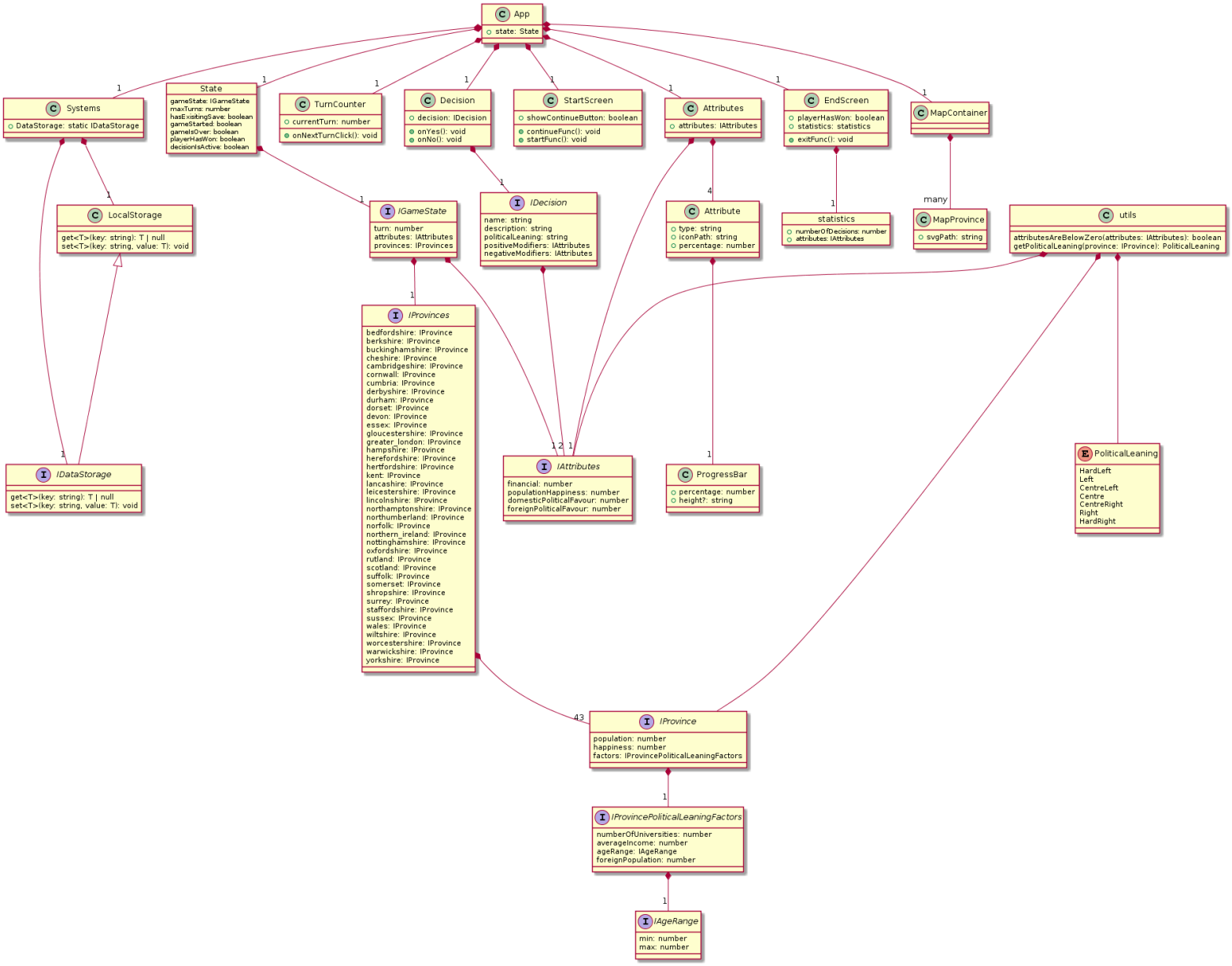


Component Folder Structure:



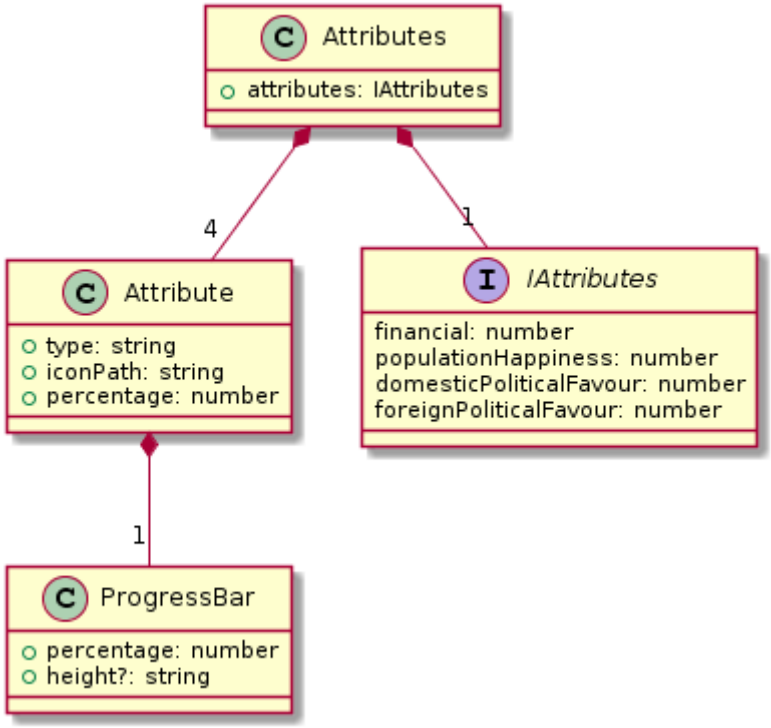
Class/Component Design

App

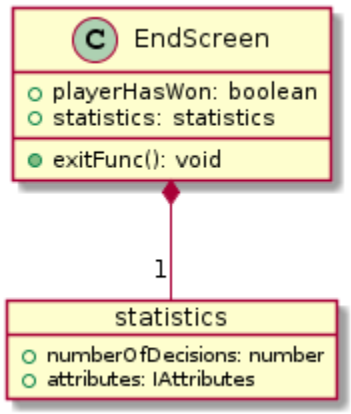


View the above PlantUML diagram by clicking here or view the diagrams of the individual components (where applicable) below.

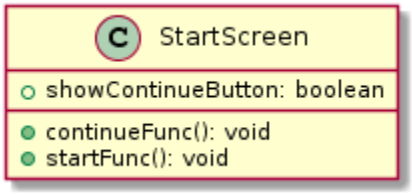
Attributes, Attribute and ProgressBar



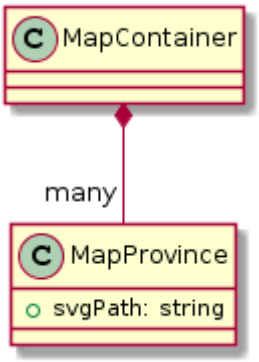
EndScreen



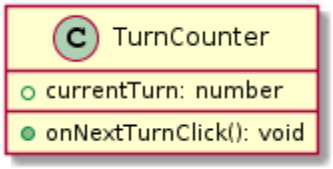
StartScreen



MapContainer and MapProvince



TurnCounter

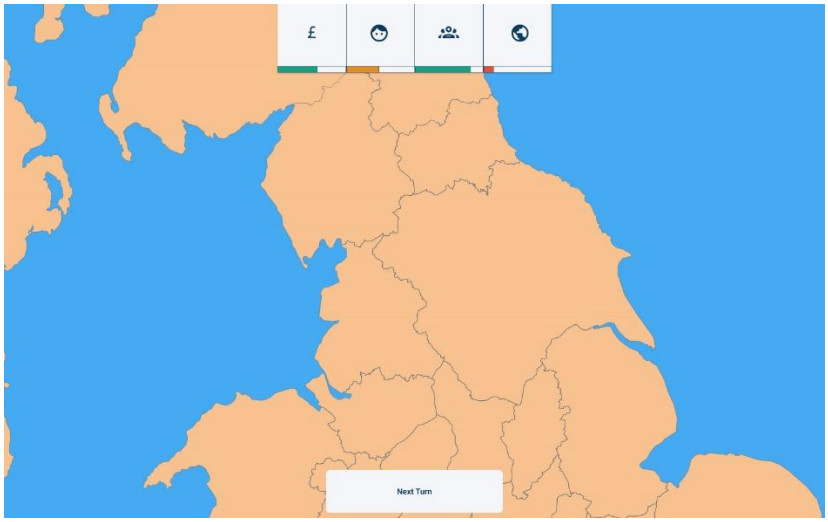


UI Designs

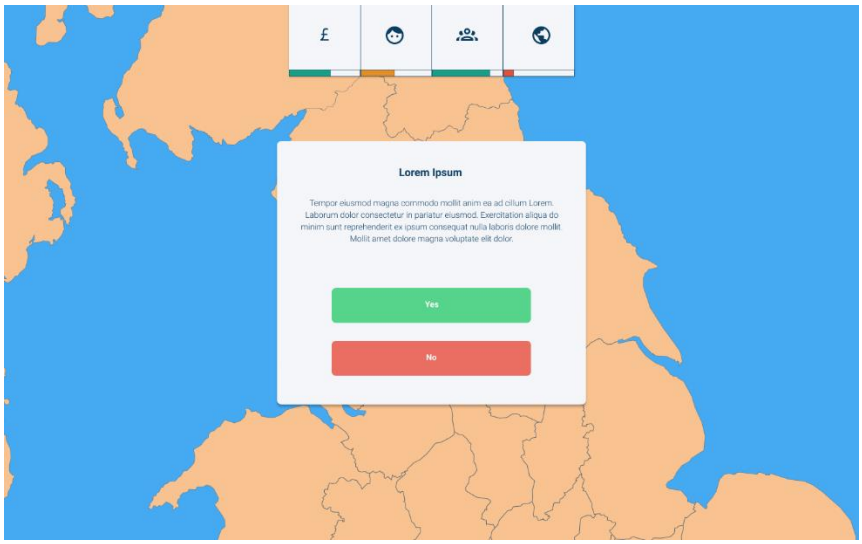
The UI has been designed within the web application Figma with a mobile-first focus that includes both a mobile and a desktop mockup for each piece of functionality.

Attribute UI

Decision UI Stage 1

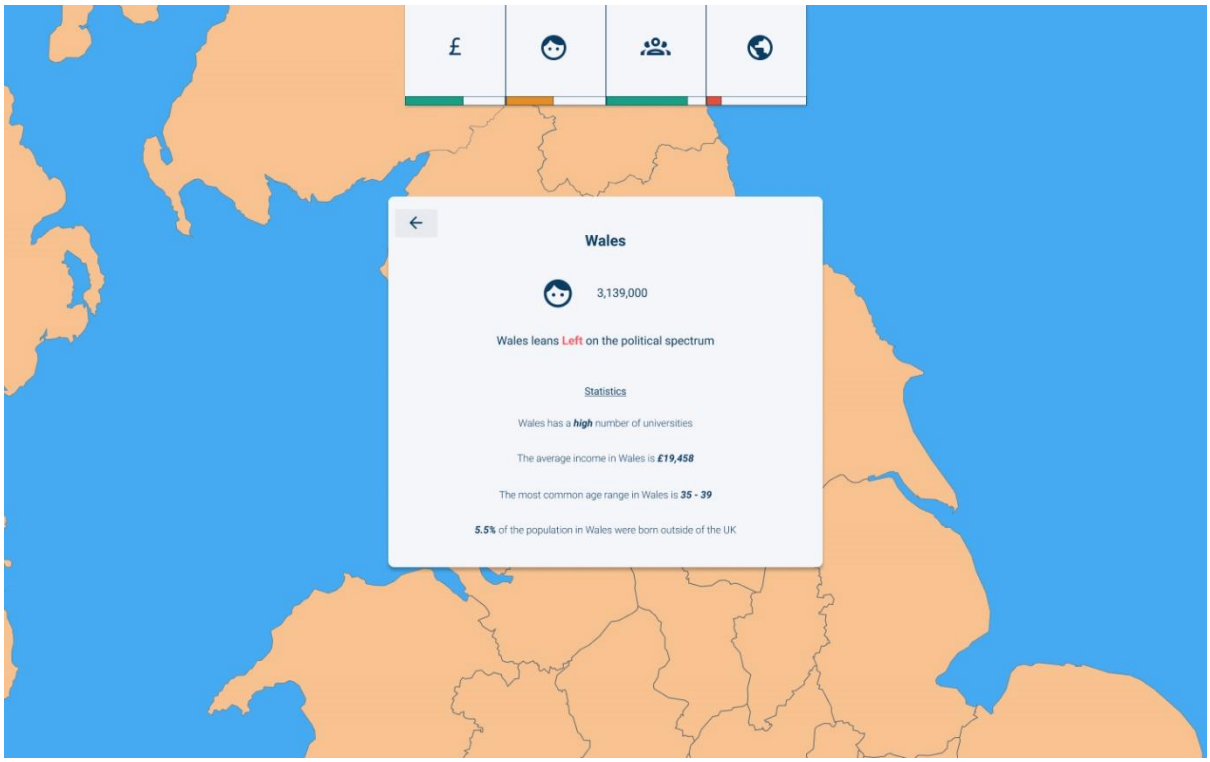


Decision UI Stage 2

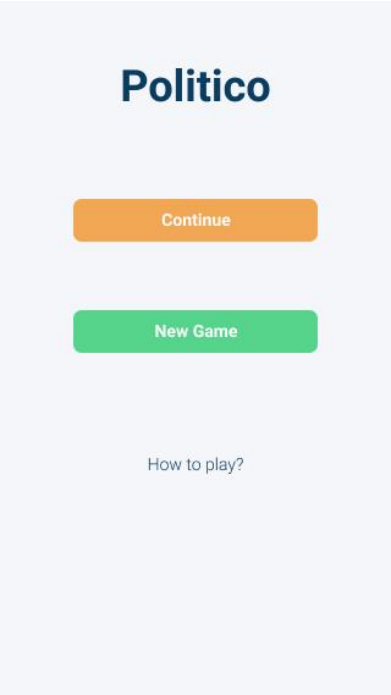


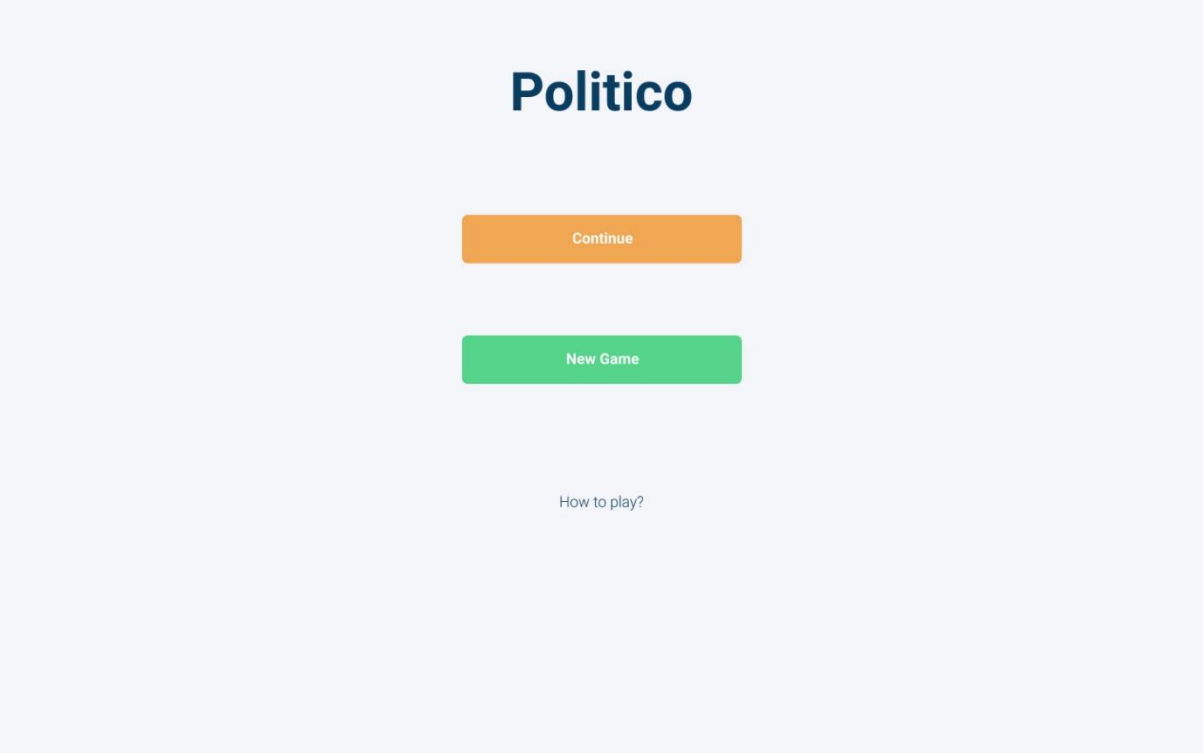
Province UI


Wales
3,139,000
Wales leans Left on the political spectrum
Statistics
Wales has a high number of universities
The average income in Wales is £19,458
The most common age range in Wales is 35 - 39
5.5% of the population in Wales were born outside of the UK


Wales
3,139,000
Wales leans Left on the political spectrum
Statistics
Wales has a high number of universities
The average income in Wales is £19,458
The most common age range in Wales is 35 - 39
5.5% of the population in Wales were born outside of the UK

Start Screen





End Screen


You Win!
You managed to balance all of your attributes and survived 8 years as Prime Minister
You made 48 decisions
Your highest rated attribute was Domestic Political Favour
Your lowest rated attribute was Population Happiness
Exit


You Lose!
You failed to keep your population happiness above 0
You made 29 decisions
Your highest rated attribute was Financial
Exit

Politico - Implementation Report

Joshua Jackson – P16179167

Introduction

This report details the current state of the project implementation as seen in the demo for the first deliverable. The document talks about the frontend as purely user interface and the backend as the functionality that powers the frontend as opposed to the more traditional meaning of frontend in the browser and backend on the server as this entire project will be ran almost exclusively on the client side.

Front-End

The game in its current state is almost fully working in terms of frontend functionality. It features a **start screen** that lets you continue a game or start a new game, it features the **game view** with the nations attributes and the ability to switch to the next turn and to make decisions that alter the attributes. Finally, you will "win" the game if the maximum number of turns is passed, conversely, you will "lose" the game if any of your attributes fall below 0; Either of these events will trigger an **end screen** telling the player they have won/lost. Also, as part of the game view, you can click on a region via a **map of the UK** which will show relevant data about that region. Aside from stylistic changes and some slight polish, the frontend shouldn't need much more adjustment.

Back-End

The backend is very minimal in its current state, the only real functionality that exists is the data storage that saves and loads the last used game state from local browser storage. Everything else powers the frontend using mock data which, once the backend implementation is under way, will be replaced and the frontend will be playable as a game.