Software Requirements Specification (SRS)
Testographer

Team: Group 2
Authors: Matt Micale, Kelly Ly, Jonathan Tucker, Seneca Anderson, Jacob Austin
Customer: Teachers and Parents of 5th and 6th grade students
Instructor: Professor James Daly

1 Introduction

The Software Requirements Specification for Testographer describes, in detail, the requirements of the system, as well as its design and implementation details. This document also includes instructions for how to run the prototype. Section 1: Introduction provides further details on the SRS, its contents, and its organization.

1.1 Purpose

The purpose of this software requirements specification is to clearly state the requirements of the system from the developers' perspective; it constitutes an agreement between those developers and the end-users. This document describes the goals for the Testographer software, as well as how those goals will be achieved and how the necessary features will be implemented. The intended audience of the Testographer SRS is the developers who will build it, though stakeholders will find it useful should they wish to learn about how the game was built.

1.2 Scope

Testographer is a game intended to help students in fifth and sixth grade learn geography. The game will allow students to quiz themselves on their knowledge of geography. Its goal is to improve their ability to identify regions on a map. By repeatedly playing the game, the student will begin to learn the locations of the regions and become faster at identifying them.

Testographer will allow the students to select a map and a level. Once they do, the game will display a sidebar on the left and the selected map on the right. The sidebar will display a list of region labels, the current score, and the time elapsed. If the student chooses easy mode, the first letter of each region will display on the map as a hint. If the student chooses hard mode, the map will be blank.

The student will click and drag region labels to regions on the map. Each time the student correctly labels a region, they will gain points. When they incorrectly label a region, they will lose points. In the end, they will gain bonuses based on what level they picked and how quickly they filled in the map. This will encourage the student to keep playing in order to beat their high score. After the game ends, the student will have the option to play again, and if necessary, the high score will be updated.

1.3 Definitions, acronyms, and abbreviations

Driver: Class that provides the functions and data necessary to control the game.

Easy Mode: When the user selects this mode, the first letter of each region's name will already be on the map, providing a hint to the player.

Hard Mode: The regions of the map will be unlabelled, with no hints to the player.

Level: The difficulty of the game.

Prototype: An early version of an application that is intended to provide proof of concept, but which does not yet have full functionality.

Region: A distinct geopolitical area within the map. For the United States map, the regions are the states; for maps of continents, the regions are countries.

Sequence Diagram: A diagram depicting the interaction between objects over time.

Use Case: A possible situation in which the game could be used.

1.4 Organization

The rest of this document consists of seven sections. The first, the Introduction, describes the purpose of this document and the scope of the software described herein. It defines terms, acronyms, and abbreviations, and it describes the organization of this document.

Section 2: Overall Description provides an extensive description of Testographer, how it works, the characteristics of its users, its constraints, its assumptions, and its dependencies. It also describes requirements that are beyond the current scope but which may be included in the future.

Section 3: Specific Requirements gives the requirements for Testographer. These requirements describe the functioning of the app and how the users will interact with it. Section 4: Modeling Requirements includes the use case, class, sequence, and state diagrams and descriptions. Section 5: Prototype describes the prototype of Testographer and gives instructions for how to use it.

Section 6: References cites the sources and assets used while researching for and building the prototype. Lastly, Section 7: Point of Contact provides information on who to contact to learn more about this project.

2 Overall Description

The game created, Testographer, is a stand-alone application that has preloaded data about geographical regions and their relative locations on a map. Scores are calculated based on game performance, and high scores for each map are saved. It is assumed that the user has basic computer knowledge. If that is the case, the user should have relative ease navigating and playing through Testographer. The game

is designed to help 5th and 6th grade students master geographical locations on a given map of a larger world region.

2.1 Product Perspective

Our product is new and self-contained. It is loaded with information on the placement of particular regions within a given map. For example, the game stores the whereabouts of all 50 states of the United States on a given map of the country. The game also stores the high scores of the user on a given map.

Along with the maps of the United States and other regions, each gameplay provides a word bank of the available choices of region names. Players will drag a given region name onto a location on the map, and the game will provide the accuracy of the guess. Based on the guesses, the game will also calculate a score. There will also be a timer for each gameplay, with more points awarded for quicker completion.

Our product also contains a help page that gives instructions on how to play the game and a credits page that accredits the developers behind its creation.

According to the Massachusetts Curriculum Framework for History and Social Science, students in 5th and 6th grades begin to deepen their knowledge and understanding of geography. Students in 5th grade learn about United States History, particularly about the American Revolution and the colonies. Students in 6th grade learn about world geography and study a variety of regions and countries around the world [1].

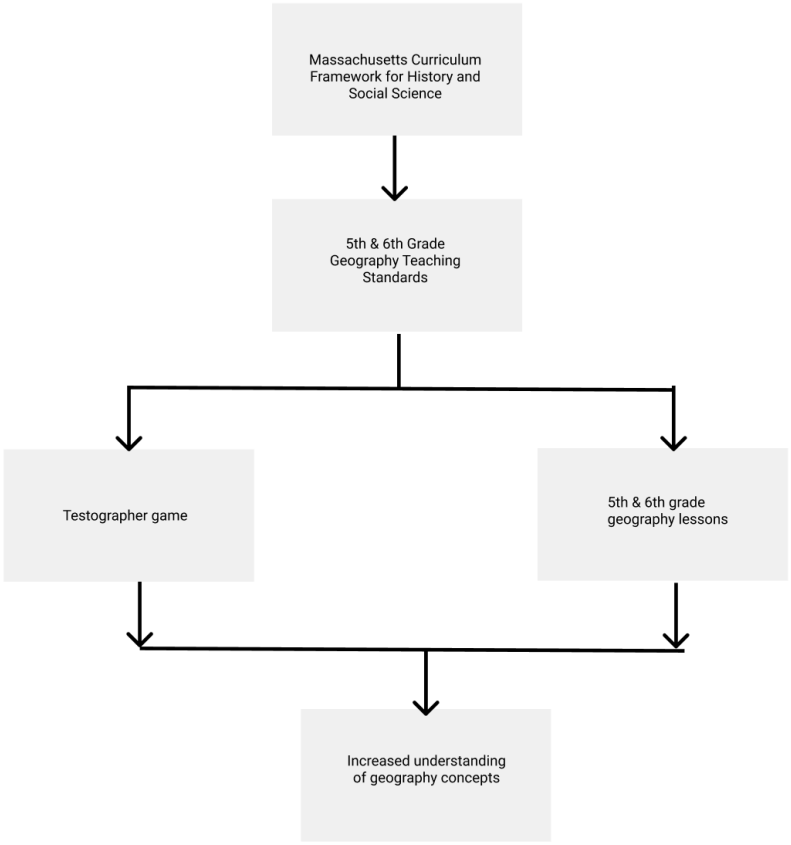
Our product is meant to be used as an aid to learning geography for students in the 5th and 6th grades. It is not designed to be used on its own but instead as an adjunct to classroom teaching of the placement of smaller geographical regions with respect to larger regions.

Our game is designed to be run on desktops and laptops and not on smartphones, tablets, or other devices.

Our product displays its information on a laptop or desktop screen and uses the mouse to track user choices and to track drag-and-drop guesses of particular regions. A keyboard is not needed to play the game.

Our game is unlikely to take up large portions of computer memory and can easily and quickly be downloaded.

A pictorial representation of how Testographer fits within the context of the Massachusetts History and Social Science education system is shown below:



2.2 Product Functions

Our software consists of a game driver that controls user choices on difficulty and map selection and controls gameplay itself. The game driver also provides access to the game instructions and credits pages. Each instance of a playthrough is represented by a CurrentMap object that stores the user's current score and is updated as the user makes guesses for regions. The CurrentMap object initially loads a Map object of the given larger region. The Map object is not manipulated throughout the playthrough.

Each map is composed of regions, each of which stores its own name and whereabouts on the map.

2.3 User Characteristics

In order to use the software, the user must be able to know how to use basic computer functions such as mouse right-clicks vs. left-clicks, holding down the left click in order to drag names into the correct boxes, and other basic technical knowledge that most 5th and 6th grade students are likely to have.

It is also assumed that the user has a general idea of geography before playing the game. While the game is designed to help teach geography, it works best if the player already knows the general layout of the map they are playing with. For example, if the user has no idea where any of the states of the United States are located in relation to each other, they will have a very difficult time using the software.

2.4 Constraints

The product must run on either Windows or macOS. The game must also be able to communicate with the running computer's mouse. The computer must have an Internet connection to download the game, but not to play it.

2.5 Assumptions and Dependencies

In order to run the game, it is assumed that Unity is properly working on the user's computer. It is also assumed that the user has a solid understanding of the English language. The user must also know basic computer skills such as dragging and dropping with a mouse. Given that some of the text in the game is small, gameplay depends on the user having adequate vision.

2.6 Apportioning of Requirements

Functionality that is likely to come with future installments of the software includes the choice of guessing region capitals instead of region names. Future functionality will provide more maps to choose from—for example, Asia, Africa, South America, and Oceania.

Another feature that is currently out of the scope of the project but will come later is the ability to provide hints to the user if they are stuck on a certain region. For example, if the user selects a hint for the state Rhode Island, the game will provide a hint like, "East Coast of the United States."

One more feature currently not available is having a bank of region shape objects instead of region name objects. The user would then drag the region as it would appear on a map onto its correct location where it would "snap" into place.

3 Specific Requirements

1. There will be a start screen that will provide the user with four options
    a. There will be two map options: the United States of America and Europe
        i. Each map option should also have a “High Score” label that corresponds to the high score for that map
            1. The initial high score will be 0 points
        ii. For the United States map, the regions will be states, and the user will be matching state name labels to states on the map
        iii. For the Europe map, the regions will be countries, and the user will be matching country name labels to countries on the map
        iv. Each map option should redirect to the level selection screen
    b. There will be a credits option that will display a list of all the team members
    c. There will be an instructions option that will display information on how to play the game

2. There will be a level selection screen that will provide the user with options to select difficulty
    a. There will be an “Easy” option in which the regions on the map will display the 1st letter of the region name as a hint
    b. There will be a “Hard” option in which there are no abbreviations
    c. Each of the options should redirect to the gameplay screen
    d. There should be an option to return to the start screen

3. The gameplay screen will display an image of a map
    a. The map will be displayed in the center of the screen
    b. Labels for regions will be displayed on the left side of the screen
    c. The score for the current game will be placed on the right side of the screen
        i. The initial score should be 0 points
    d. The time elapsed will be placed on the right side of the screen below the score for the current game
        i. The initial elapsed time should be 0 seconds
        ii. The time elapsed should update every second

4. The user can drag and drop each label to a region on the map
    a. Each region can only correspond to one label
    b. The label will stick to the region if the user had a correct match
    c. The game score will increase by 50 points for each correct match
    d. The label will move back to its original position on the left side of the screen if the user had an incorrect match
    e. The game score will decrease by 25 points for each incorrect match
    f. Up to 10 region labels will appear on the left side of the screen at a time
        i. A new label, if there are any remaining labels that have not yet been displayed and matched to a region, will take the place of a label that was correctly matched

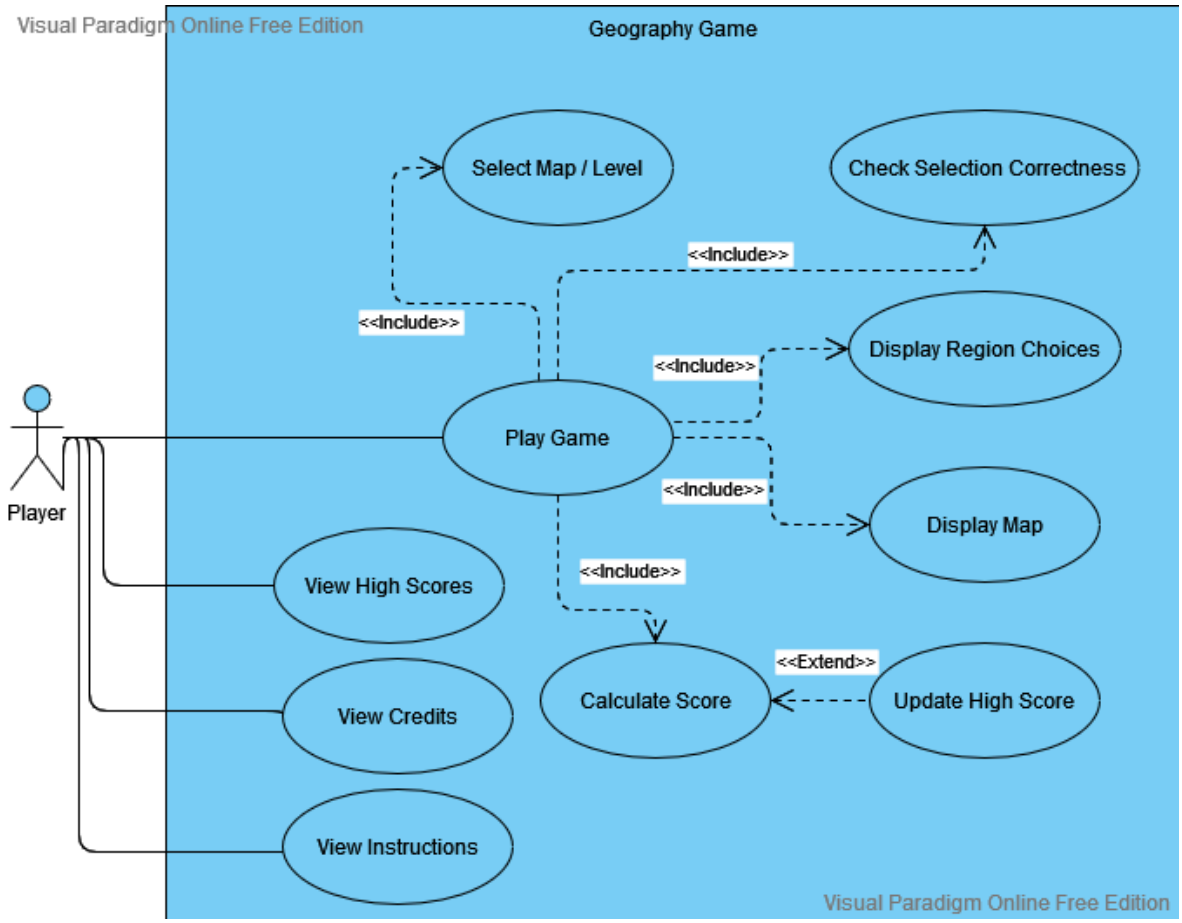
5. The final game score will be calculated when all regions are correctly labelled
    a. The score will be updated by an added time bonus
        i. The time bonus will be calculated such that 1500 - elapsed time (in seconds)
        ii. If the calculated time bonus is below 0 points, the time bonus should be set to 0 points
    b. The score will be multiplied by the difficulty level after the time bonus has been added
        i. In easy mode, the score will be multiplied by 1
        ii. In hard mode, the score will be multiplied by 2
    c. If the final score is below 0 points, the final score should be set to 0 points
    d. The high score for the map will be updated if the final game score is greater than the current high score

6. There will be a game complete screen at the end of the game
    a. The screen will display the final game score, time elapsed, and the high score for the map
    b. There will be an option to return to the start screen

4 Modeling Requirements

4.1 Use Case Diagram

This diagram contains the possible use cases for Testographer. Each use case is described in an oval. The lines represent how the use cases interact with each other and outside actors.



4.2 Use Case Descriptions

Use Case Name:	Select Map / Level
Actors:	None
Description:	When the game starts, the player must first choose a map and a level.

Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	Play Game

Use Case Name:	Check Selection Correctness
Actors:	None
Description:	Check to see if the player has matched the correct name to the correct region.
Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	Play Game

Use Case Name:	Play Game
Actors:	Player
Description:	Does everything needed to set up, play, and finish the game. This includes: selecting a map and a level, showing the player the available choices, checking if their selections are correct, and giving them a score based on their performance.
Type:	Primary
Includes:	Select Map / Level, Check Selection Correctness, Display Region Choices, Display Map, Calculate Score
Extends:	None
Cross-refs:	None
Uses cases:	None

Use Case Name:	Display Region Choices
Actors:	None
Description:	Shows the player what choices are available to fill in the regions with.

Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	Play Game

Use Case Name:	Display Map
Actors:	None
Description:	Displays the selected map to the screen for the player to fill in.
Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	Play Game

Use Case Name:	View High Scores
Actors:	Player
Description:	Allows the player to view their previous high scores.
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	None

Use Case Name:	View Credits
Actors:	Player
Description:	Allows the player to view the authors' names.
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	None

Uses cases:	None

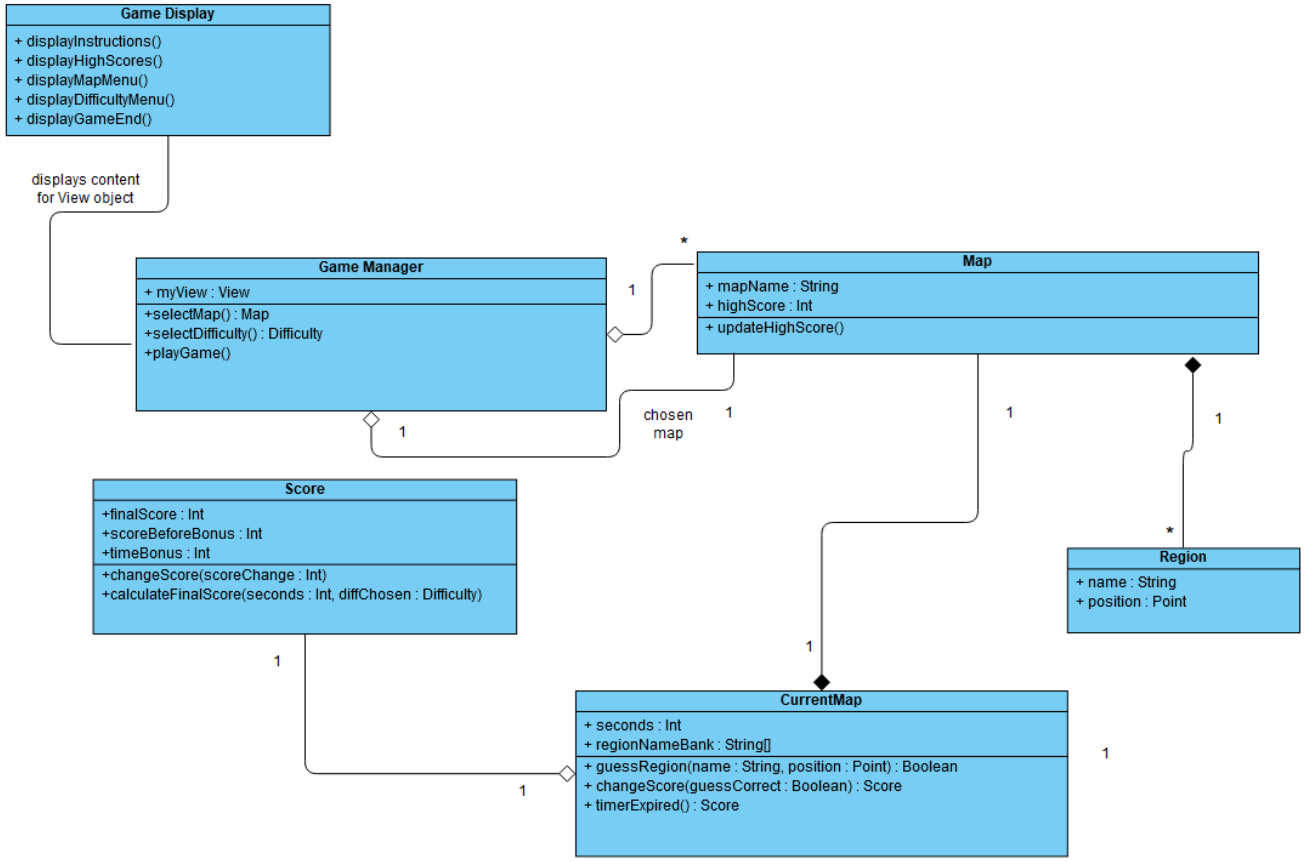
Use Case Name:	Calculate Scores
Actors:	None
Description:	Calculates the player's score based on their performance. "Performance" as defined for our game is a combination of correctness and speed.
Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	Play Game

Use Case Name:	Update High Score
Actors:	None
Description:	If the player receives a score higher than their previous best, update the high scores to reflect the new high score.
Type:	Tertiary
Includes:	None
Extends:	Calculate Score
Cross-refs:	None
Uses cases:	None

Use Case Name:	View Instructions
Actors:	Player
Description:	Shows the player instructions on how to play the game.
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	None

4.3 Class Diagram

The diagram below contains descriptions of each class required for our project. The lines represent the interactions between classes. Open diamonds on the ends of these lines show that one class is a part of another class, while closed diamonds show the class is created out of one or more of the subclass.



4.4 Class Descriptions

Element Name	Description
Game Manager	This class controls the operation of the game. It handles user interaction as well as loading necessary components for the game to function.

myView: View	A variable containing a view class. The class will tell the Game Manager what screen should currently be displayed.
selectMap(): Map	Function takes in the user's choice for which map they want to play and changes the current view to the appropriate map.
selectDifficulty(): Difficulty	Function takes in the user's choice for which difficulty/level they wish to play and adjusts the game accordingly.

Element Name	**Description**
Map	This class will contain the name of the associated map as well as the high score received on that map.
mapName: String	The name of the map.
highScore: int	The highest score achieved on this specific map.
updateHighScore()	This function will be called to update the high score if the current score is greater than the high score.

Element Name	**Description**
CurrentMap	CurrentMap will contain the information about the map currently being played. It will handle the region selection options, the time elapsed, and the current score.
seconds: int	The number of seconds elapsed since the game began.
regionNameBank: String[]	An array of strings of all of the regions still available to be guessed.

guessRegion(name: String, position: Point): Boolean	Takes in the name associated with the guessed region and the guess's position on the screen. Returns true if the correct region is placed in the correct position.
changeScore(guessCorrect: Boolean): Score	Changes the score based on the correctness of the guess.
timerExpired(): Score	Once a maximum time has been reached, the game will end and the final score will be returned.

Element Name	**Description**
Region	Each guessable area on the map will have a Region object associated with it. It will have the name of the region as well as the position it is meant to be placed.
name: String	The name of the region.
position: Point	The correct position for this region on the map.

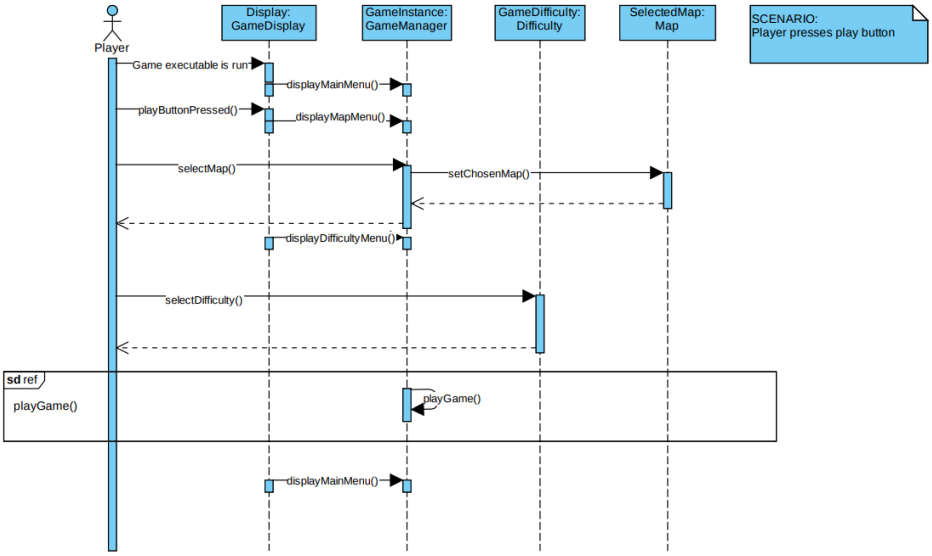
Element Name	**Description**
Score	Contains all aspects of the score and how it will be calculated. This object will have the ability to calculate a time bonus based on how long the player took to complete the puzzle.
finalScore: int	The final score after it is calculated.
scoreBeforeBonus: int	The score before the time bonus is added.
timeBonus: int	The additional score to be added based on how long the player took to complete the game.

changeScore(scoreChange: int)	Change the score by the amount specified by scoreChange.
calculateFinalScore(seconds: int, diffChosen: Difficulty)	Calculates the final score based on the amount of time elapsed and the difficulty of the game.

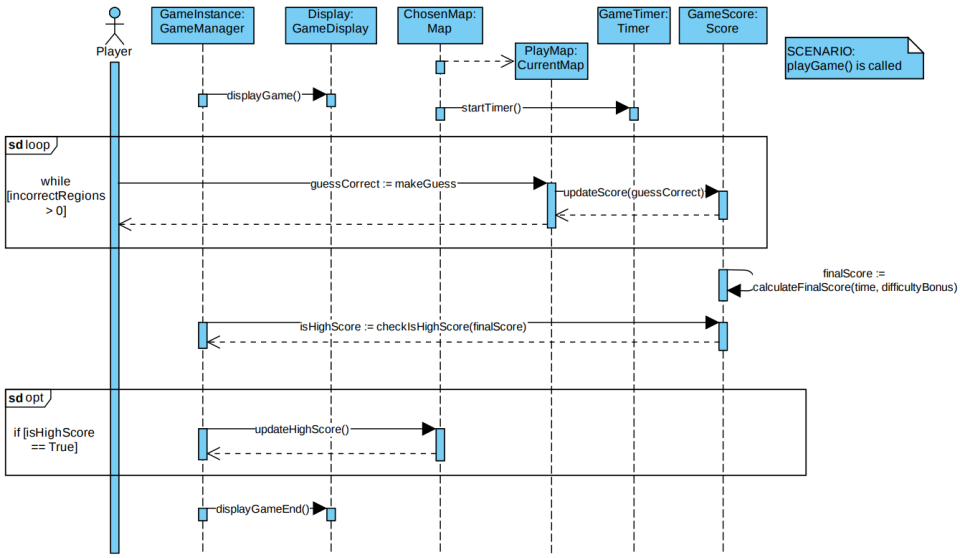
Element Name	**Description**
Game Display	Game Display contains a variety of functions that all share the same purpose: change the player's view to the specified scene.
displayInstructions()	Switch to the Instructions scene.
displayHighScores()	Switch to the High Scores scene.
displayMapMenu()	Switch to the Map Menu scene.
displayDifficultyMenu()	Switch to the Difficulty Menu scene.
displayGameEnd()	Switch to the Game End scene.

4.5 Sequence Diagrams

The game starts up with the main menu being displayed. The player can press the button "Play Game" which will then prompt the display of the map select screen. From here the player may select the map they wish to play on, which will then be stored in the SelectedMap object, as the program then displays the difficulty menu. Once the player selects their desired difficulty level it is saved to the GameDifficulty object and the main gameplay loop will begin - which is described in more detail in the second sequence diagram. Upon exiting the main gameplay loop, the program will display the main menu to the player.

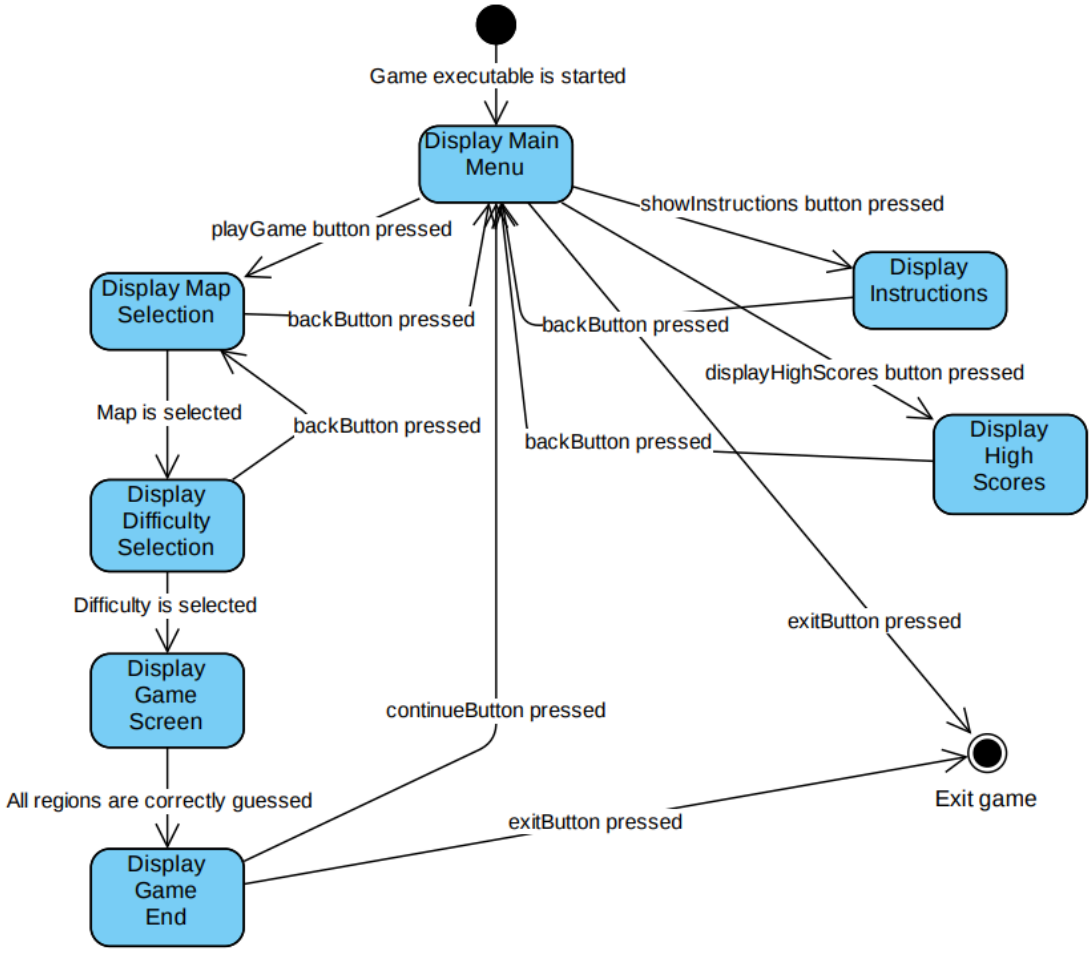


In this diagram, we detail the main gameplay loop. Once the program enters the gameplay phase, a CurrentMap object is created to store map information and aid in tracking the score. The game page is then displayed, and the timer starts. Here the user can input guesses until all regions have been correctly guessed. Each time a guess is made, the score is updated - increasing on correct guesses and decreasing on incorrect guesses. Once all regions are correctly guessed, the game will calculate a final score, taking the raw score value and updating it with a time bonus and a difficulty modifier. This value is then compared to the high score for the map. If the new score is higher than the old high score, we replace the high score for the map with the new final score. After this, the program displays the game end screen.



4.6 State Diagram

The state diagram represents the interactions between the user's button presses and the corresponding displays.



5 Prototype

The prototype for Testographer contains the start page, instructions page, credits page, map menu, difficulty menu, and gameplay screen. In terms of system functionality, the prototype will include functioning buttons that take the user to the desired screen. For example, the credits and instructions screens each contain a button that takes the user back to the main menu. Also, the user is able to select a map and a difficulty which is saved in the game manager. On top of this, it will

contain a layout of the gameplay screen, which will eventually simulate a playthrough.

On the gameplay screen, the user will be able to drag a limited number of region names to a limited number of grey boxes on the map.

5.1 How to Run Prototype

In order to run the prototype, Unity must be installed on the machine (https://store.unity.com/). Unity version 2021.2.1f1 must also be installed.

The downloads for the game can be found on the GitHub repository for the Testographer project (https://github.com/lykelly19/testographer). After downloading the project scenes and scripts, Unity can be used to run the prototype.

In Unity Hub, click on the "Add" button. Click on the folder that contains the downloaded files. Unity 2D should open automatically with the folder and program contents.

The prototype can be run by creating and running a build.

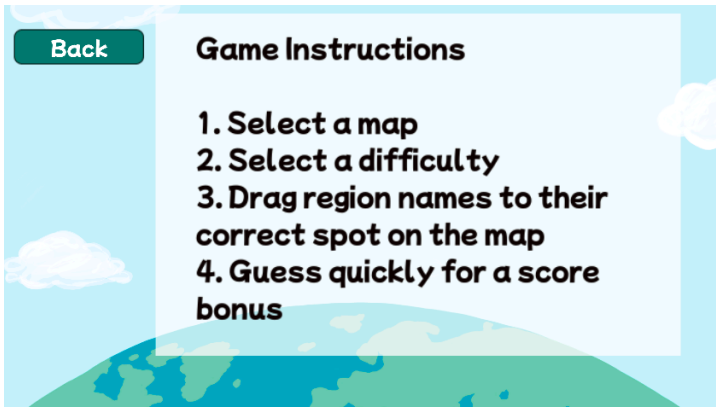
To create and run a build, click on File > Build and Run. Select an empty folder to store the files. Name the file and click on the Save button. Navigate to the folder in your computer's file explorer application. Open the file called "testographer"; the game will load and run.

5.2 Sample Scenarios

First, the user will start at the main menu:

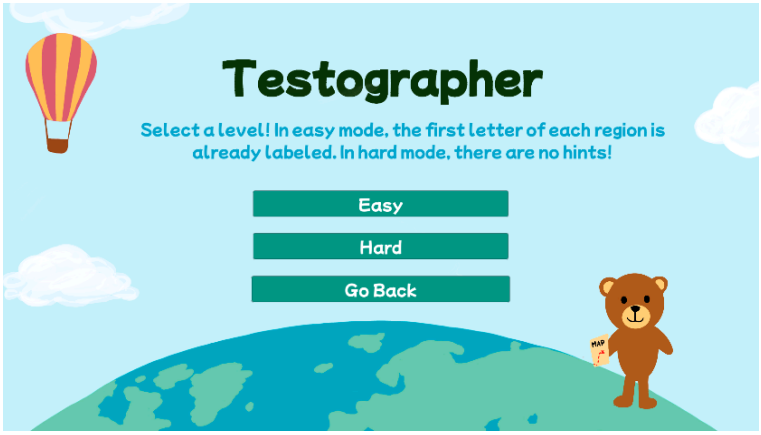
If they select credits, they will be sent to the following page which lists the developers' names:

After clicking on the “Back” button, the user will be sent back to the main menu. If the user wants to know how to play the game, they can click the “How to Play” button which takes them to this screen:

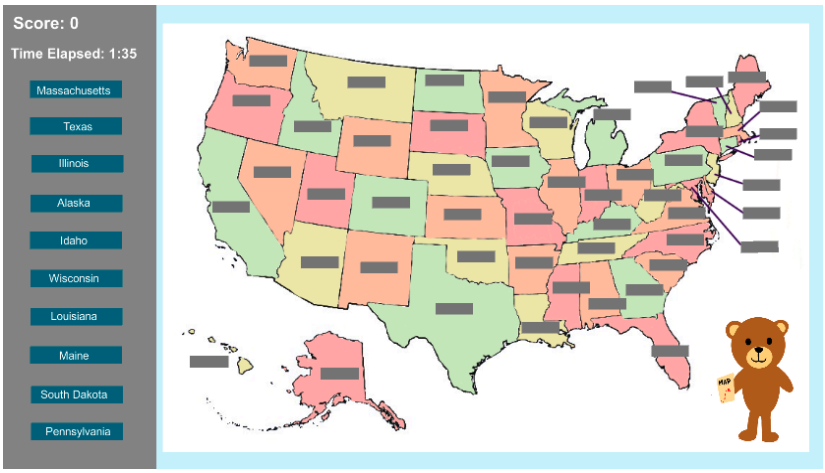


If the user then hits “Back” and selects “Play Game,” they are sent to the following map choice screen:

Depending on which map is chosen, the Game Manager saves the choice so the correct map can be loaded on the Game Page. After a map is selected, the user is directed to the difficulty selection screen:



Similarly, to the chosen map, the chosen difficulty is saved through the Game Manager and loaded correctly once the Game Page is opened. The difficulty variable is saved as an integer. If easy is chosen, the difficulty integer is 0. If hard is chosen, the difficulty integer is 0. The Game Page is automatically opened after selecting a difficulty. As of now, the game page consists of this:



The game page consists of a map object, which in this case is the United States. The region names are circle objects with text in them and the region labels are rectangle objects with no text in them. In the working version, the user will drag a

circle object onto a rectangle object of which they believe to be the correct region name. The score will update automatically based on the accuracy of the guess.

The Game Manager will check the coordinates of the dragged region object and alert the user of the accuracy of their guess.

6 References

[1] "Massachusetts Curriculum Framework for History and Social Science," Massachusetts Department of Elementary and Secondary Education, 2018. [Online]. Available: https://www.doe.mass.edu/frameworks/hss/2018-12.pdf. [Accessed Nov. 12, 2021].

[2] Project website: https://senecaquinn.notion.site/Testographer-281c86beda8f4ebcaa13865b112b4b63

7 Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james_daly at uml.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.