Software Requirements Specification (SRS)
Project Geotyper

Team: Project group 1 - Geo Typer
Authors: Jack Schnair, Josh Sullivan, Demetri Cassidy, John Swanson, Sam Patriquin
Customer: Massachusetts School Systems
Instructor: James Daly

1 Introduction

Geotyper, an edutainment game, is intended to provide its users with both enjoyment, and the tools to improve their typing, history, and geography skills. This document details the requirements of the creation of this project at various levels, and in addition describes how users will be able to interact with Geotyper's internal systems. The remainder of this section further details the purpose of this document, and in addition section 1.4 provides an in-depth explanation of each section's purpose.

1.1 Purpose

The purpose of this document, the SRS (Software Requirements Specification), document is to give our client(s) insight into the system that is being developed. This system is Geotyper, which is an educational web-based video game. The SRS will first contain an overview of this document and what is to be expected. It will then go into detail of what functions, goals, and constraints are involved in producing Geotyper. It will then detail the specific product requirements surrounding Geotyper. After that, models that describe the use cases, sequences of possible events, internal representation of our system, and the various states our system can enter will be shown. Lastly, various samples from our UI prototype will be shown in effort to depict a basic version of the functions that Geotyper will provide.

1.2 Scope

The product is an educational typing and geography game called Geotyper. Our goal is to create a fun environment for children to learn in. School systems and parents should be able to purchase access to this web based game. It should be accessible from any web browser that supports WebGL. Geotyper is ideal for children in the range of 4th to 6th grade, with different features for different education levels in that range. Students will be able to choose between different games that all allow them to practice typing and additionally review American geography and history. Ideally students who play these games repetitively will gradually increase their proficiency in each skill. Even students above their age groups' learning levels will enjoy the game if played at the highest difficulty, as the games are meant to be fun as well as educational. Geotyper should not be used as a tool to learn the basics of typing, it is meant to increase typing dexterity for those who already know the basics.

1.3 Definitions, acronyms, and abbreviations

- SRS - This document is called software requirements specification, which is abbreviated as SRS
- Geotyper - This is the name we chose to describe our game as a whole
- UI - User interface; this term describes a component of our system that is able to be interacted with by our users

- User/Player - We use these terms in the SRS to describe a theoretical individual person who is playing our game
- Town - The main location Geotyper takes place in, with its setting being inspired by Plymouth Plantation
- NPC - Non-Playable Character; these are characters that the user will be able to interact with in order to access certain parts of Geotyper
- Minigame - One of three user playable game within Geotyper; each game has a distinct set of actions and goals and is started by talking to an NPC
- Scoreboard - An in town intractable object that will allow the user to see their highest ranking in each of the three minigames

1.4 Organization

The following section will contain the details regarding the various functions and their purpose of Geotyper. It will also describe what our expectations of our users are, in terms of our demographic and what knowledge we expect our users to have. Lastly, it will detail various constraints and requirements that we are operating within when constructing our product. The next section, which contains requirements, details an outline of necessary product components for Geotyper. It is formatted as a bulleted list, with the requirements deemed most critical to our product towards the top, and with sub-requirements denoted by an indented bullet under their parent requirement.

The following section contains various models to describe our product. First, the use case diagram itself depicts how each high level case interacts with each other and is accompanied by many use case descriptors that give high level descriptions of what each use case does and when it happens. Additional information is included such as the other use cases it includes or extends, and a reference to the requirement it fulfils. In addition, the class diagrams in section four provide an overview of the different classes that make up Geotyper and how they relate to each other. Each class is given a set of attributes and operations that define what data types they contain and what actions they can perform. In addition, lines between different classes are utilized to define their relationships with each other. Supplemental information is also provided to elaborate upon the purpose of each class, along with their attributes, operations, and relationships. The last part of section four is the sequence diagrams, which portray how an action is performed through a sequence of different class or object functions. The arrows in the diagram can be followed to give an idea of how each class depends on the others to accomplish the given goal. The boxes on each class's timeline represent the time the object is alive and smaller boxes represent the time of each function called. The two sequence diagrams shown in section 4 are elaborated on before each diagram.

The fifth section describes the specifics of the UI prototype and how to access it as a user. System requirements needed to run the prototype are also discussed. Finally, this section demonstrates some usage of the prototype in its current state through images and descriptions. The final section contains a list of all of the references that were used in constructing this document.

2 Overall Description

The following section is intended to contain high level descriptions regarding our product, Geotyper, and its functions. It will detail some of the context around our product, including various constraints, how users will interface with it, and the operations that our product will provide. The core functionality of our system will also be detailed, in terms of the utility that our users can expect to derive from our product. We will also detail some of our expectations about our users, including who our product is targeted towards and what background we expect them to have.

2.1 Product Perspective

Geotyper is intended to be a standalone browser game that provides an entertaining method for students to practice their typing skills, while learning various history and geography facts. Since Geotyper is standalone, it can be used either in conjunction with or separately from an educational system. Geotyper will play within the browser, utilizing the WebGL framework. This method of serving our product should provide an engaging and accessible method for students to exercise their social studies and typing skills.

In terms of the constraints that a web-hosted game will have, Geotyper will require both mouse and keyboard inputs as the method of input. It will also require users to have access to the internet, but that will be the user's main limitation. Besides this, there will be minimal hardware or software constraints, as all that is required of the user is a system capable of running a web-browser, and a connection stable enough to maintain connection to our site. Lastly, As our game is intended for grades four through six, we will restrict our content to that which is relevant to those age groups, based on the Massachusetts History and Social Science Framework.

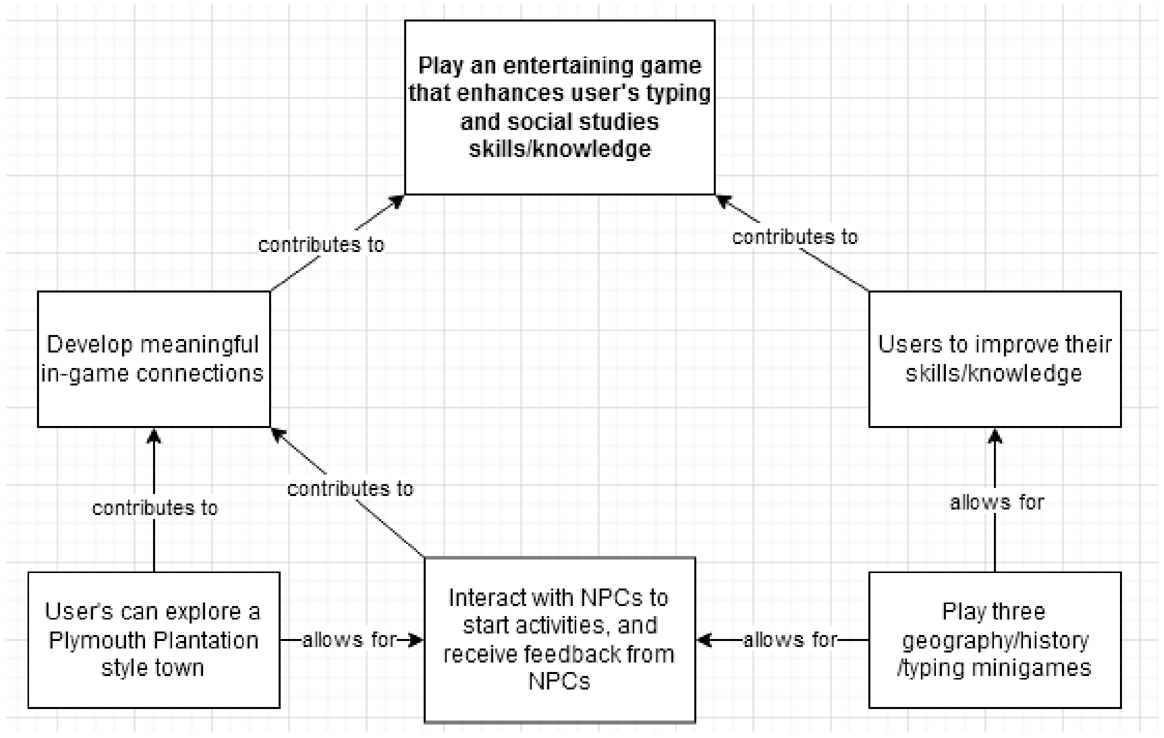
Our main mode of user feedback and communication will be through NPCs in the game. Users will be able to launch minigames by talking to these NPCs, and they will receive feedback from them after exiting the game (feedback that states their game results). This method of communication will be more engaging for the user than a traditional game, where results are just displayed through information boxes, as users will be both connected to the game and its characters. In addition, users will be able to track their progress through a scoreboard, which will be visible and accessible to the users through the town, similar to if the users were to access a mini game.

2.2 Product Functions

The main functions of Geotyper include: interacting with characters to start minigames, playing minigames, and monitoring their progress through a scoreboard. In addition to these main functions, users will also be able to explore the town that Geotyper takes place in and select the difficulty of their minigames. This will allow for users to test their knowledge of topics and for the game to be accessible to a wider range of students.

The goal of these functions is to provide users with an accessible way of playing games that improve their typing and social studies skills that requires minimal overhead. The minigames will not build off of progress from past games, but each instance of a minigame will have different content from its last instance. This will allow users to access our site and play any number of minigames. In addition, an intractable in-town scoreboard allows for the users to keep track of their progress in such a way that fits well with the theme of Geotyper.

Diagram of functions and their purpose/end result


Play an entertaining game that enhances user's typing and social studies skills/knowledge
contributes to
contributes to
Develop meaningful in-game connections
Users to improve their skills/knowledge
contributes to
contributes to
allows for
User's can explore a Plymouth Plantation style town
allows for
Interact with NPCs to start activities, and receive feedback from NPCs
allows for
Play three geography/history /typing minigames

2.3 User Characteristics

Geotyper’s demographic is any student that falls within the 4th to 6th grade range in the US education system. This grade range can also be used as an estimated age range of nine to twelve years old, for students outside of the US education system. Geotyper’s three different difficulties are designed to accommodate for the different grades, with the first difficulty designed for 4th grade, the second for 5th, and the third for 6th grade. Each difficulty will expect users to have the knowledge that an average student would have, who is entering/currently in that grade. Users are not expected to have any expertise in a specific category, but they should have a general knowledge of each category, as stated previously.

2.4 Constraints

Geotyper will necessitate an internet browser and a stable internet connection which will be its greatest limitation. The browser must be able to run HTML 5 and WebGL in order for Geotyper to function. As a typing game, Geotyper requires a keyboard in order to function and there are no alternative control schemes. There are no disability options for those with motor, visual, or cognitive disabilities that could make the standard game excessively difficult. As the game has no audio queues, auditory disabilities will not be a problem.

Geotyper is designed to teach the geography and history of the United States of America which will limit its audience in other countries. Geotyper's difficulty is based on the expected knowledge and capabilities of 4th, 5th, and 6th graders as prescribed by the Massachusetts History and Social Science Framework. As each state in the United States of America controls its own education level, the Massachusetts History and Social Science Framework may not reliably reflect the education level of all 4th, 5th, and 6th grade students in every state.

The game's capabilities are constrained by the design capabilities of the Unity game engine. All security requirements will be handled by Github where the game will be hosted using WebGL. Constraints for the game's successful launch are teachers' interest in a typing game that teaches United States geography and Social Sciences and Geotypers capability to engage and teach students.

2.5 Assumptions and Dependencies

As it is a web browser game, Geotyper will be reliant on a stable internet connection and assumes one is available in order to function. The game will be built using the Unity game engine. The game is run through the WebGL framework and relies on its ability to adapt the code designed in Unity to play on the user's internet browser of choice. The game is reliant on Github to host its source code and handle any updates. It is assumed that any future updates to web browsers, WebGL, or Github will not negatively affect Geotyper and it will continue to function as expected.

That the students have developed at least a 4th grade English reading level, as decided by the Massachusetts English Language Arts and Literacy Curriculum Framework, is assumed in order to read the text in the game. It is assumed that the users have a basic understanding of computers and a rudimentary ability to type.

2.6 Apportioning of Requirements

Geotyper is a simple game and its in-game scope is limited to one interactable town and three minigames. Geotyper has numerous avenues for future expansion in different areas. Further expansion on the number of minigames could increase the games ability to retain user interest. It would also increase the games ability to teach different areas of study in History and Social Sciences through different minigames. A greater variety of questions in the individual minigames would increase Geotyper's replayability which will further help Geotypers ability to teach students. The game's scope is currently limited to the expected capabilities of 4th, 5th, and 6th graders. Further expansion could allow the game to cater to broader grade levels kindergarten through 12th grade.

Studies on the games ability to teach 4th-6th graders could be used to improve the games difficulty in order to challenge and interest students of each grade level at a rate more conducive to their development. Geotyper could also be used in future studies on the capability of video games to teach and engage students.

Geotyper currently has no disability options and could be difficult to use for students with motor, visual or cognitive disabilities. Adding options to help disabled users such as settings for colorblindness, alternate control schemes, text to speech, and key sensitivity controls could expand the game's reach and capability to teach all students of different ability levels. The game will currently only be available in English but could be expanded to more languages.

As it is a browser game, Geotyper is currently limited in its availability due to necessitating an internet connection. Building a Geotyper app would expand the game's availability and potentially draw in parents and teachers that may prefer an app over a game on an internet browser.

3 Specific Requirements

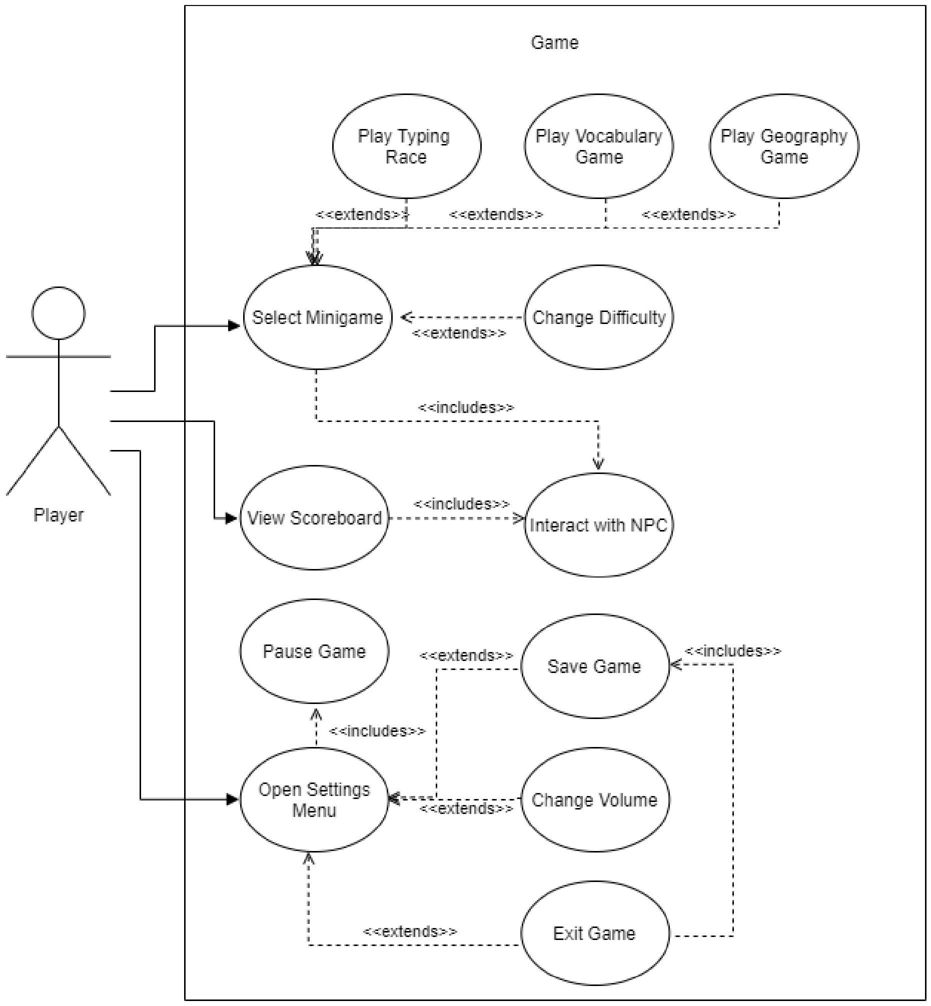
1. The game will be a collection of replayable typing minigames
    a. The game will take place in an old colony town or in Plymouth Plantation
    b. The user will interact with NPCs to play typing minigames
    c. All minigames will be single player
2. The game will follow Massachusetts History and Social Science Framework
3. The user can pause the active game
4. The user should be able to open a settings menu
    a. Doing this pauses the game
    b. The user can adjust the volume
        i. Displayed as horizontal bar that can be decreased or increased by pressing left or right arrow keys
    c. The user may save the gamestate
        i. Includes game statistics and high scores
    d. The user has the option to exit the game from here, automatically saving the game beforehand.
    e. Hover over different options by pressing up or down arrow keys
    f. Select option by pressing "Enter"
5. The user should be able to control their character with a keyboard
    a. Different keys correspond to different actions
        i. W or up arrow to move up
        ii. A or left arrow to move left
        iii. S or down arrow to move down
        iv. D or right arrow to move right
        v. ESC to open settings menu
        vi. E to interact with NPCs or scoreboard
6. The user should be able to select different difficulty levels
    a. Difficulty levels are independent to each minigame
    b. They can change the difficulty before each minigame
    c. There are 3 levels: easy, medium, and hard
    d. Each difficulty has different vocabulary
7. The user can pick from three different minigames
    a. Racing Game
        i. The user will engage in a typing race against an NPC
            1. Players must enter text to match displayed text with exact spelling, punctuation, and capitalization
            2. Easy Difficulty
                a. Simple (4th grade) vocabulary, slow NPC speed
            3. Medium Difficulty
                a. Average (5th grade) vocabulary, medium NPC speed
            4. Hard Difficulty

            a. Complex (6th Grade) vocabulary and fast NPC speed
    ii. If player fails, they may try again
    iii. Player receives score based on completion speed and difficulty
  b. Vocabulary Game
    i. Easy Difficulty
      1. The user will be shown a definition and must pick the word from a list
    ii. Medium Difficulty
      1. The user will be displayed a word and definition, then shown a definition and be required to type the word
    iii. Hard Difficulty
      1. The user will be shown a word and definition, then shown a word and be required to type its definition
  c. Quiz Game
    i. The game will finish after time expires
      1. Easy Difficulty
        a. Player must guess the state from its shape
      2. Medium Difficulty
        a. Matching states to their capitals by typing capital or state corresponding to displayed text
      3. Hard Difficulty
        a. When shown the shape of a state the player must type the state name and its capital
    ii. Time limit of two minutes, score is based on how many correct answers are given
  d. Choices displayed through NPC dialogue that the player can select from
8. The user should be able to view a score board containing various statistics
  a. The user's high scores in different games
  b. Their number of games completed
  c. Average and top speed of racing game for each difficulty

4 Modeling Requirements

Use Case Diagram:

The use case diagram below depicts how the user (player) will interact with the product. The three main functionalities the player can access are selecting minigames, viewing the scoreboard, and opening the settings menu. These functionalities are called use cases and can be related to other use cases through arrows that indicate inclusion and extensions. Use cases that are included by another use case are a part of that functionality. A use case that extends a functionality is a conditionally used case. These two arrows are indicated by the labels “<<include>>” and “<<extends>>”. In this particular use case diagram there is only one actor, the player, so every use case is used by them.



Use Case Name:	Open Settings Menu
Actors:	Player
Description:	Menu appears on the screen that automatically pauses the game. Through the menu the player can change the volume and exit the game.
Type:	Primary
Includes:	Pause Game
Extends:	N/A
Cross-refs:	Requirements 4 and 5av
Uses cases:	Happens when the player presses the “esc” key.

Use Case Name:	Pause Game
Actors:	Player
Description:	Freezes the game so that nothing happens until it is resumed.
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 4a
Uses cases:	The game always pauses when the settings menu is opened.

Use Case Name:	Change Volume
Actors:	Player
Description:	The player is able to adjust the volume of the game by incrementing or decrementing a slider by one at a time. This is done by opening the settings menu, selecting adjust volume and using the up and down arrow keys to move the slider.
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 4b
Uses cases:	Used when the player presses the up or down arrow key when “adjust volume” is selected in the settings menu.

Use Case Name:	Save Game
Actors:	Player
Description:	Player data will be saved in JSON format. Player data includes their game statistics and high scores.
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 4c and 4d
Uses cases:	Used when the player selects “save game” in the settings menu.

Use Case Name:	Exit Game
Actors:	Player
Description:	The game will
Type:	Secondary
Includes:	Save Game
Extends:	N/A
Cross-refs:	Requirement 4d
Uses cases:	Used when the player selects “exit game” in the settings menu.

Use Case Name:	View Scoreboard
Actors:	Player
Description:	Brings up a scene that includes the statistics of each game including each one’s average and high scores.
Type:	Primary
Includes:	Interact with NPC
Extends:	N/A
Cross-refs:	Requirement 5avi and 8
Uses cases:	When the player interacts with the scoreboard NPC the scoreboard will be viewed.

Use Case Name:	Interact with NPC
Actors:	Player
Description:	Will either initiate a dialogue with an NPC to determine what difficulty and game to play or view the scoreboard depending on which NPC is interacted with.
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 5avi, 6, 7a
Uses cases:	Used when the player presses “E” next to an NPC.

Use Case Name:	Select Minigame
Actors:	Player
Description:	Depending on which NPC the player interacts with the player will be asked if they would like to play one of three games. If the player answers yes then the minigame will then be played, if no then the player will exit the dialogue.
Type:	Primary
Includes:	Interact with NPC
Extends:	N/A
Cross-refs:	Requirement 7
Uses cases:	Used when interacting with a townsperson NPC and the play game dialogue path is chosen.

Use Case Name:	Play Typing Race
Actors:	Player
Description:	Starts the “typing race” game which is a game played against a NPC where the player races the NPC to accurately type a block of text before the other.
Type:	Secondary
Includes:	N/A
Extends:	Select Minigame
Cross-refs:	Requirement 7a
Uses cases:	Used when the user selects “typing race” as their game choice.

Use Case Name:	Play Vocabulary Game
Actors:	Player
Description:	Plays the “vocabulary game” which is a game where the player is shown definitions and has to type the word defined by it. At the hardest difficulty the player will also have to type the definition.
Type:	Secondary
Includes:	N/A
Extends:	Select Minigame
Cross-refs:	Requirement 7b
Uses cases:	Used when the user selects “vocabulary game” as their game choice.

Use Case Name:	Play Geography Game
Actors:	Player
Description:	Plays the “geography game” which is a game where the player is shown either the shape of a state, the name, or both. Depending on the difficulty they will have to enter the state name, the capital, or the state and capital respectively.
Type:	Secondary
Includes:	N/A
Extends:	Select Minigame
Cross-refs:	Requirement 7c
Uses cases:	Used when the user selects “geography game” as their game choice.

Use Case Name:	Change Difficulty
Actors:	Player
Description:	Changes the difficulty of the game that will be played depending on which option the player chooses from NPC dialogue.
Type:	Secondary
Includes:	N/A
Extends:	Select Minigame
Cross-refs:	Requirement 6
Uses cases:	Used when interacting with a townsperson NPC and the player wants to play a game.

Class Diagram:

The class diagram contains all of the classes that make up Geotyper. These include: “Unity Entity”, “World”, “Minigame”, “Player”, “NPC”, “Scoreboard”, and “Settings”. Each class is represented by a box in the diagram that has three sections: The name of the class, the attributes, and the functions. The different relationships between the classes can be seen by the type of arrow connecting them. Black triangle heads inherit from the class they are pointing to. Arrows with open heads point to classes that they use. The white diamonds represent aggregation, meaning that the class that the white diamond is next to contains multiple of the class the connection is coming from.

Data Dictionary:

Element Name	**Description**
Unity Entity	An entity in the Unity engine is an individual “thing” that populates a game or program’s world. They are used to identify pieces of data that belong together.
Attributes	
sprite: Sprite	A 2D graphics object that can be used to represent objects in the game.
controller: AnimationController	A tool that manages animations and animation transitions for objects based on in-game conditions.
position: Transform	A component that represents the position, rotation, and scale of an object.
collisionBox: Collider	A component that defines the shape of an object and handles collisions with other entities in the game’s world.
physicsBox: RigidBody	A component that utilizes physics simulation to handle the motion of an object.
Relationships	
A collection of components that make up and represent objects like the Player class and the NPC class. Entities exist within the game’s world.	
UML Extensions	
N/A	

Element Name	Description
Player	An in-game object that the user controls. This represents the game’s protagonist.
Attributes	
name: String	A string that represents the player’s name.
Operations	
move(direction: keyCode)	Triggers the user’s character to move in a direction that corresponds to their key press.
interact()	This allows the user to interact with an NPC in the world that is nearby.
talkToNpc()	This triggers a nearby NPC object to initiate dialog with the user.
viewScoreboard()	This triggers the game’s scoreboard to appear on-screen.
Relationships	
The objects from the player class are able to interact with NPC objects and can trigger them to initiate dialog with the user. This class is also able to modify attributes of the Settings class.	
UML Extensions	
N/A	

Element Name	Description
NPC	An object in the game’s world that responds to interactions from the Player class and initiates minigames.
Attributes	
dialog: list<string>	A list of strings that an NPC uses when talking with the Player class.
scoreboard: Scoreboard	An object of the Scoreboard class that is used to display high scores for minigames.
Operations	
displayDialog()	This operation displays a dialog string through the UI.
selectGame(): int	This occurs when selecting which minigame to play. Returns an integer value representing the chosen game.
selectDifficulty(): int	This occurs when choosing the difficulty of a minigame. Returns an integer value representing the chosen difficulty.
Relationships	
As an entity, members of the NPC class occupy the game’s word and respond to interactions from the Player class. Class members are capable of interacting with the Scoreboard class and causing members to display scores on-screen.	
UML Extensions	
N/A	

Element Name	Description
World	This class contains different elements of the game, such as entities and minigames, and facilitates the scenes in which these elements are located.
Attributes	
settings: Settings	An object containing the current game's settings selected by the user.
scoreboard: Scoreboard	An object containing high scores for their video games.
scene: Scene	A Unity object containing the assets relevant to the user's current visuals.
Operations	
changeScene(scene: Scene)	Changes the world's current scene.
startMinigame(game: int)	This operation is used to launch a specific minigame that the user can play.
Relationships	
A world is composed of different entities, such as NPCs and the Player. They are capable of displaying information from the scoreboard class and can utilize a Settings object to handle user preferences.	
UML Extensions	
N/A	

Element Name	Description
Minigame	A class for handling the minigames that users can play. Minigames are initiated as a result of the Player class interacting with members of the NPC class.
Attributes	
currentScore: int	The user's current score in the ongoing minigame.
highScore: int	The highest score achieved in a specific minigame by the user.
difficulty: Difficulty	The current difficulty of the minigame being played.
scene: Scene	The Unity object that contains the assets relevant to the user's current minigame.
Operations	
updateScoreboard()	This operation is called to update the scoreboard with the latest high scores, if the user achieves a new high score.
selectDifficulty(): Difficulty	Prompts the user to select a difficulty for a minigame and returns their decision. This is used to update the difficulty attribute.
Relationships	
Minigames are contained within and are initiated by the game's world. Once the user achieves a new high score, the Minigame class interacts with the Scoreboard class to update its information.	
UML Extensions	
N/A	

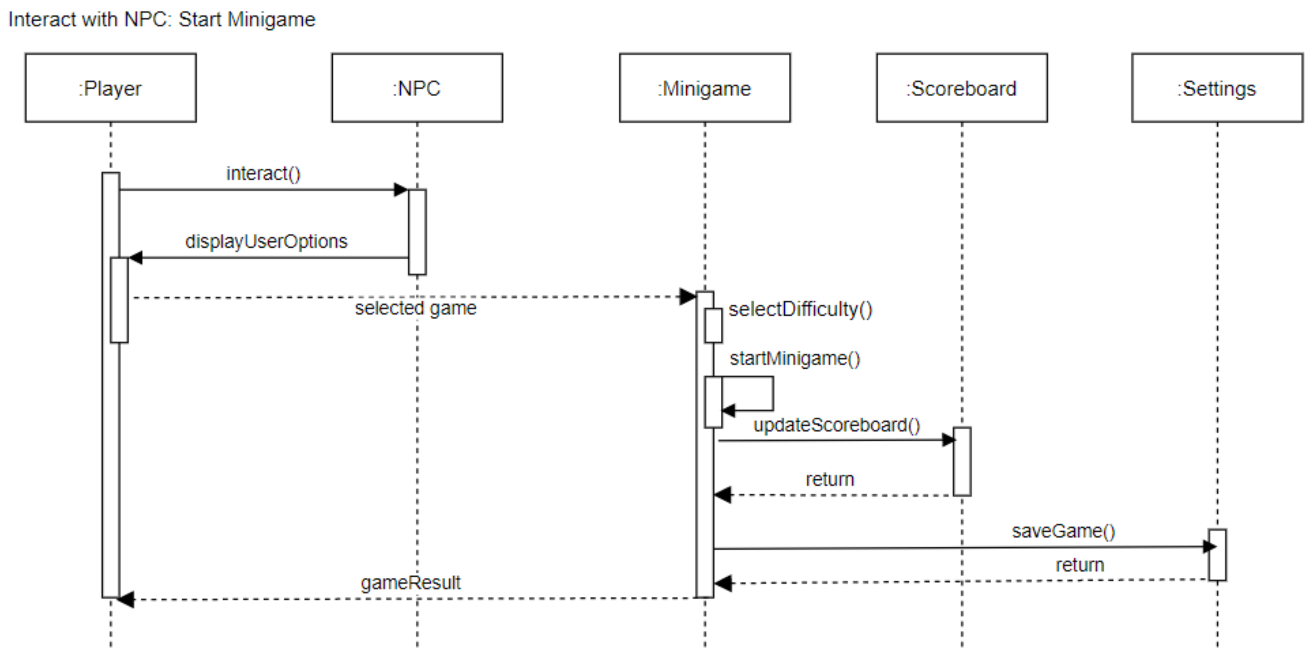
Element Name	Description
Scoreboard	A class for displaying the high scores for the game's three minigames.
Attributes	
speedGameHighscore: int	The current high score for the typing race minigame.
vocabGameHighscore: int	The current high score for the vocabulary minigame.
geoGameHighscore: int	The current high score for the geography minigame.
ui: UIComponent	The user interface component for displaying the scoreboard's information on-screen.
Operations	
openScoreboard()	Displays the scoreboard's user interface on-screen.
closeScoreboard()	Removes the scoreboard's user interface from the screen.
Relationships	
Objects of the Minigame class update a Scoreboard's information whenever a new high score is achieved.	
UML Extensions	
N/A	

Element Name	Description
Settings	Records and manages the volume levels of the game
Attributes	
audio: audioController	This attribute manages and plays audio clips.
Operations	
setVolume(volume: int)	Adjusts the volume levels for certain parts of the game's audio.
exitGame()	Allows the user to exit out of the game.
saveGame()	Saves the current state of the game.
closeSettings()	Removes the setting options from the screen.
openSettings()	Displays the game's settings options on-screen.
Relationships	
The World class utilizes the Settings class to manage the game's volume. It is also capable of opening and closing the settings options on-screen.	
UML Extensions	
N/A	

Sequence Diagram 1.1:

The following two diagrams depict two possibilities of what could happen when the player selects a minigame. The process of doing this requires the player to first talk to a townsperson NPC. The NPC will display dialogue that displays the name of one of the following games: “Typing race”, “Vocabulary game”, “Geography game”. The player can then select “yes” or “no” for if they want to play the game or not. The first diagram here shows what will happen when one of the games is selected from the NPC. A minigame object will be created and the difficulty will then be selected by the user from the options of “Easy”, “Medium”, and “Hard”. The minigame object will then start the minigame with the correct game type and difficulty and continue looping the game until it is over. Once the game is over, the scoreboard will update through the scoreboard object if a high score was achieved. Since there is new data to be saved, the game is then saved from the settings object. This sequence ends by returning the game result to the player so they can see it.

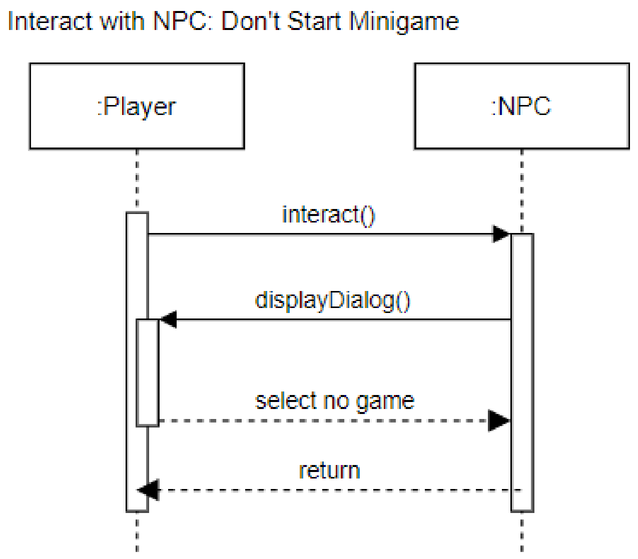
Diagram 1.1



Sequence Diagram 1.2:

This diagram shows the sequence of actions that occur when in the previous scenario the player chooses “no” from the NPC’s dialogue. When this happens the NPC will simply recognize that no game was selected and the dialogue will end. The player will then be free to walk around the town once again and interact with something else.

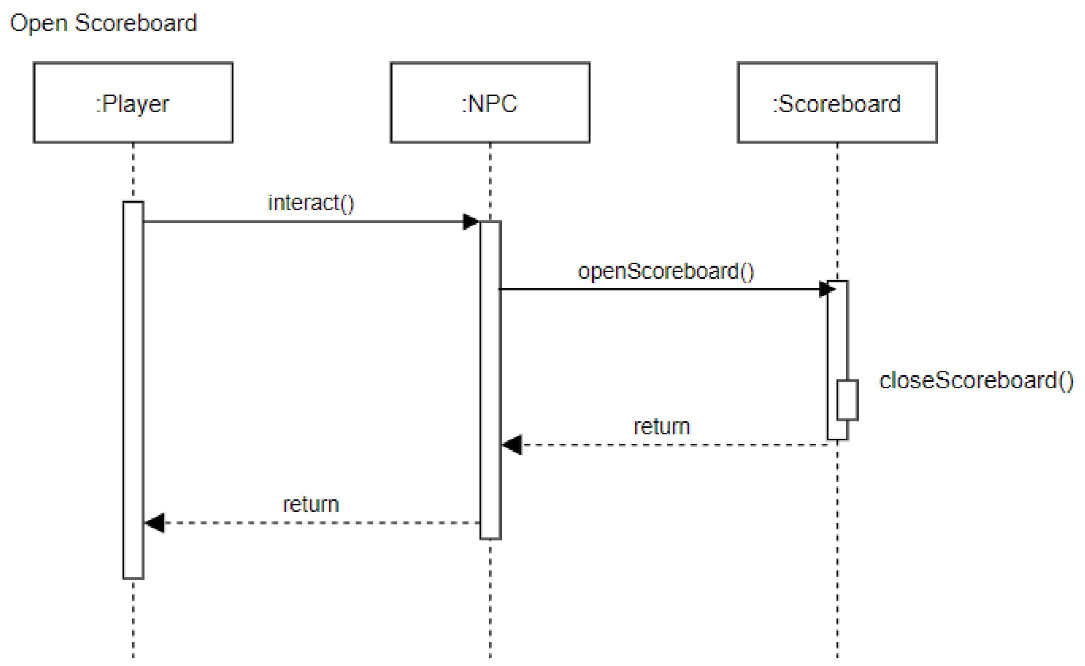
Diagram 1.2



Sequence Diagram 2:

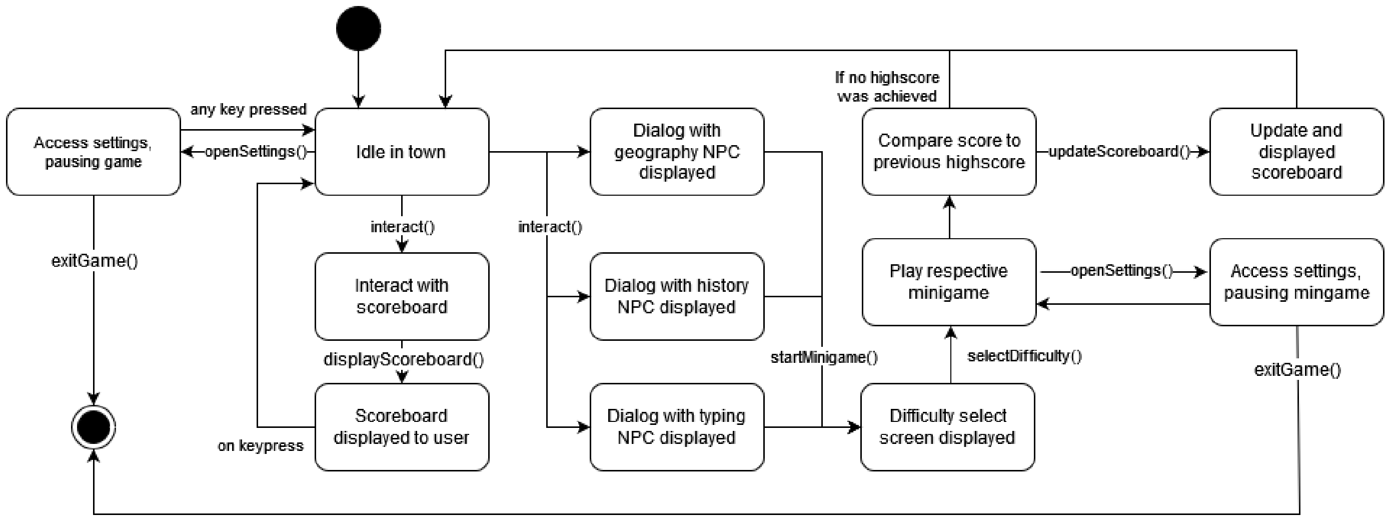
The following diagram shows the sequence of events that occur when the player wants to view the scoreboard. The player will start by finding the scoreboard NPC in the town and interacting with it. This works exactly like it does with a regular townsperson NPC, but with the scoreboard it will call the scoreboard object and the scoreboard scene will open. The player can view the scene as long as they like until a key is pressed and the scoreboard closes. Once this happens the regular town scene will be viewed again and the player can walk around normally.

Diagram 2



State Diagram:

This diagram shows the different possible states that Geotyper can enter while in use. The state diagram begins at the solid black circle, which is when the user enters Geotyper. The user is able to initiate transfer to different states via various in game actions, which correspond to function calls. These transitions are depicted by solid arrows, with the corresponding function shown on the label. At any point, after accessing the settings menu, the user can exit the game. This is represented by the black circle enclosed by another circle on our diagram.


If no highscore was achieved
Access settings, pausing game
any key pressed
openSettings()
Idle in town
Dialog with geography NPC displayed
Compare score to previous highscore
updateScoreboard()
Update and displayed scoreboard
interact()
interact()
exitGame()
Interact with scoreboard
Dialog with history NPC displayed
Play respective minigame
openSettings()
Access settings, pausing mingame
displayScoreboard()
startMinigame()
selectDifficulty()
exitGame()
Scoreboard displayed to user
on keypress
Dialog with typing NPC displayed
Difficulty select screen displayed

5 Prototype

The prototype will demonstrate the town setting, player control, NPC interaction, a settings menu, and scoreboard interaction. The player (currently a white circle) will be able to move around a placeholder town and collide with three NPCs representing each minigame. The player will also be able to collide with the scoreboard. Interacting with the NPCs by hitting 'e' on the keyboard will bring up a UI component that will ask the player if they want to play a specific minigame or exit the conversation. If the player chooses to play a minigame, nothing will happen as the game functionality will not be implemented yet. Next, if the player chooses to interact with the scoreboard, a UI component will appear with placeholder graphics demonstrating the ability to view scores. The player can exit this menu. Finally, if the player hits the 'esc' key, a UI component will appear demonstrating the settings menu. This menu will have no functionality yet other than the player being able to exit this menu.

5.1 How to Run Prototype

To run the prototype, any web browser is required along with a stable internet connection. The user's browser should support HTML5, but this should be default to any web browser. The game is hosted on itch.io, a website built for hosting indie games. As long as the user can access this site, the prototype will run. There are no operating system restraints, the game will run on all major operating systems

Link to page: **https://infinite-cut.itch.io/geotyper**

5.2 Sample Scenarios

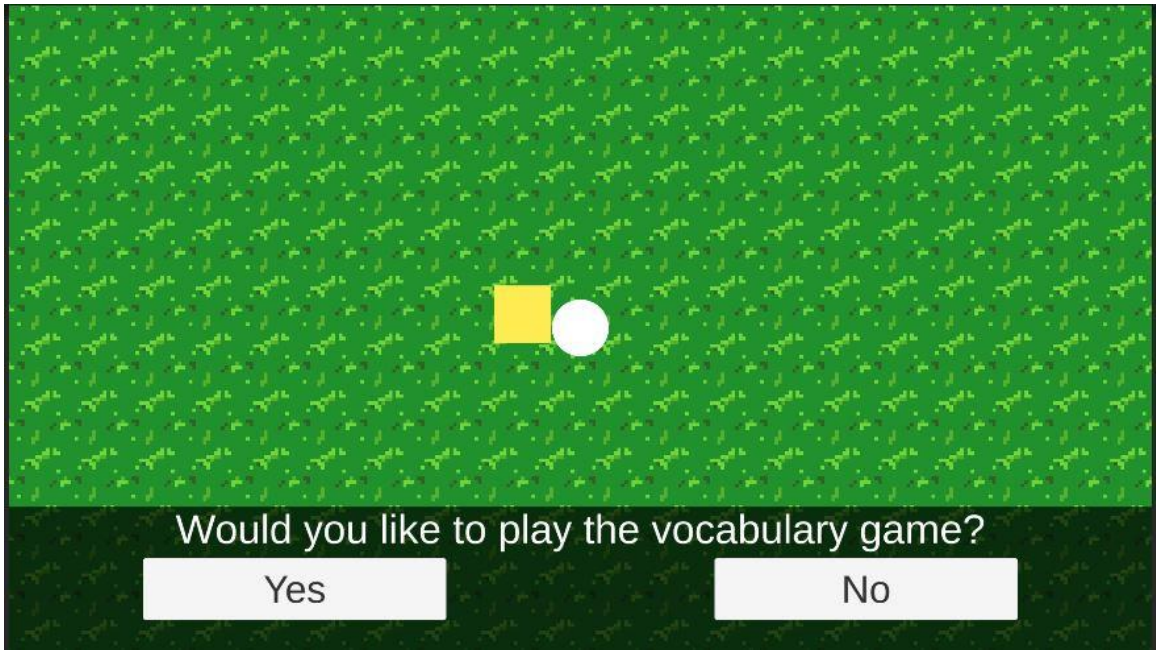


Fig.1) Represents the user pressing the key ‘e’ when interacting with the NPC standing for the vocabulary minigame. In the future, this interaction will launch the minigame.

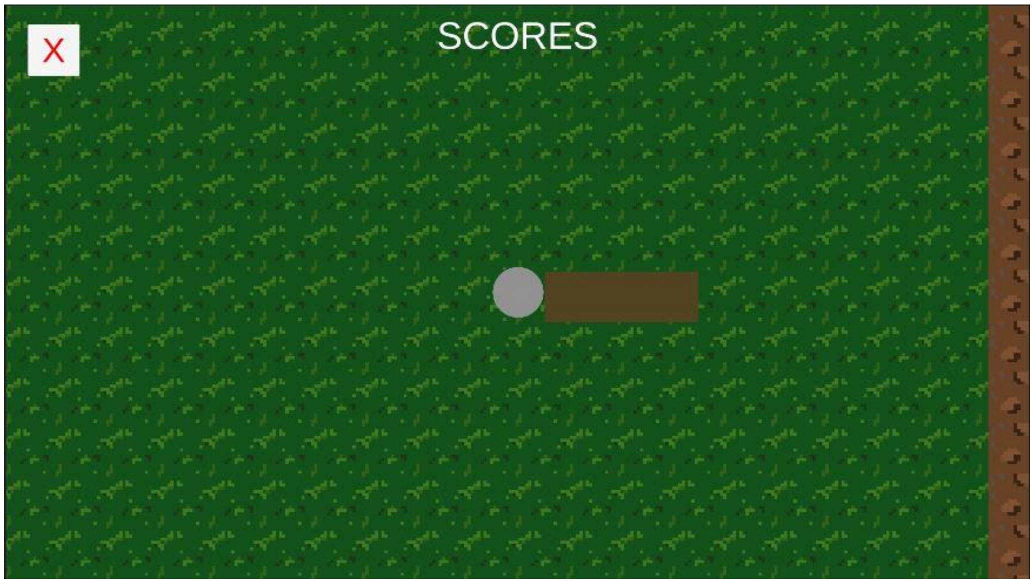


Fig 2.) Represents the user interacting with the scoreboard by pressing the key 'e'. In the future, the user's highscores will be displayed here.

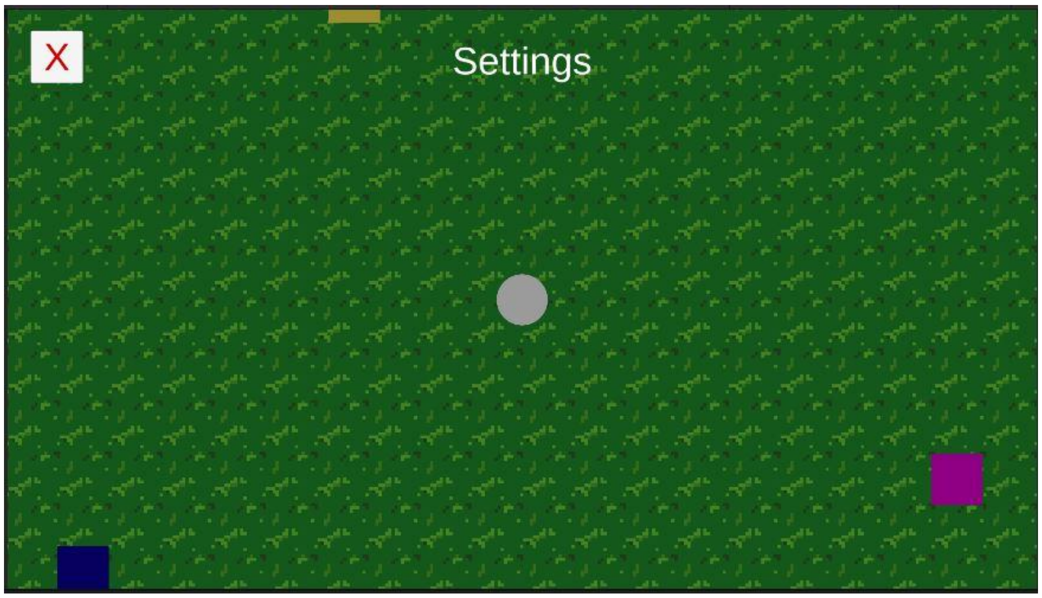


Fig 3.) Represents the user opening the settings menu by pressing the 'esc' key. In the future, the user will be able to change various settings here.

6 References

[1] Sullivan, J. (n.d.). Software Engineering Product. Retrieved from https://joshua119.github.io/SoftwareEngineeringProject/.

[2] D. Thakore and S. Biswas, "Routing with Persistent Link Modeling in Intermittently Connected Wireless Networks," Proceedings of IEEE Military Communication, Atlantic City, October 2005.

[3] History and Social Science Framework. Massachusetts Department of Elementary and Secondary Education, 2018, https://www.doe.mass.edu/frameworks/hss/2018-12.pdf

[4] English Language Arts and Literacy. Massachusetts Department of Elementary and Secondary Education, 2017, https://www.doe.mass.edu/frameworks/ela/2017-06.pdf

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.