Regio Vinco ™
Software Requirements Specification

Author: Richard McKenna
Debugging Enterprises™

Based on IEEE Std 830™-1998 (R2009) document format

1 Introduction

Casual games should be easy to learn but hard to master. Serious (i.e. learning) games should impart a deeper understanding of real world information through gameplay. Both game styles should entertain. In this project, we aim to build a serious, casual game titled ***Regio Vinco***.

The mechanics of gameplay for casual games should simple, only requiring the player to perform simple actions like mouse clicks, letting the user easily pick up the game and put it down. This is particularly true of casual games played on mobile platforms. In this project we'll be making a desktop game that would also be appropriate to port to a mobile or Web platform. It will be serious, it will be casual, and it will be fun.

The ***Regio Vinco*** game is a game for learning about the world and its geography. Through playing the game, the player can practice their knowledge of continent, nation, state, and province borders, as well as capitals, flags, and leaders. The game will let the user select the map of their choice to play and will keep track of player accomplishments for all provided played maps.

1.1 **Purpose**

The purpose of this document is to specify how our ***Regio Vinco*** game should look and play. The intended audience for this document is all the members of the development team, from the game designers to the artists, to the software engineers and even salespeople who will ultimately interface with the customers and advertisers. This document serves as an agreement among all parties and a reference for how the game should ultimately be constructed. Upon completing the reading of this document, one should clearly visualize how the game application will look and operate as well as understand the game mechanics and rules.

1.2 Scope

For Debugging Enterprises, the goal is to make this the first in a series of casual games. This starts with ***Regio Vinco***, but by no means ends there. Therefore it is important to make good design decisions in this project that will result in reusable components. As such, a framework (or set of frameworks), should be designed and constructed to be used to make ***Regio Vinco*** as well as the other similar mouse-push button games.

1.3 Definitions, acronyms, and abbreviations

Casual Game – A game that can be picked up and played quickly, where 15 minutes of entertainment may suffice, but typically where richer experiences are also possible. Easy to pick up, easy to put down.

IEEE – Institute of Electrical and Electronics Engineers, the "world's largest professional association for the advancement of technology".

Framework – In an object-oriented language, a collection of classes and interfaces that collectively provide a service for building applications or additional frameworks all with a common need.

GUI – Graphical User Interface, visual controls like buttons inside a window in a software application that collectively allow the user to operate the program.

Serious Game – A game that proves an experience for the player through which one learns out of game knowledge of some value.

UML – Unified Modeling Language, a standard set of document formats for designing software graphically.

Use Case Diagram – A UML document format that specifies how a user will interact with a system. Note that these diagrams do not include technical details. Instead, they are fed as input into the design stage (stage after this one) where the appropriate software designs are constructed based in part on the Use Cases specified in the SRS.

1.4 References

IEEE Std 830™-1998 (R2009) – IEEE Recommended Practice for Software Requirements Specification

1.5 Overview

This SRS will clearly define how the ***Regio Vinco*** game should look and operate as well as how to actually play the game. Note that this is not a software design description (SDD), which would design how to construct the software using UML. This document does not specify how to build the appropriate technologies, it is simply an agreement concerning what to build. Section 2 of this document will provide the context for the project and specify all the conceptual design, including the game rules and parameters. Section 3 will present how the game interface should be laid out and all program functionality. Section 4 provides a Table of Contents, an Index, and References.

2 Overall description

Here you are, planning a trip to South America, but before you jump on a plane you want to know the lay of the land. Maybe you learned a bit about the continent in school, maybe you've forgotten. In either case you need to make sure you're caught up on South American geography before you touch down. Where can you go to get a refresher course on South American borders, capitals, flags, and leaders? ***Regio Vinco*** of course!

2.1 Product perspective

Casual games should be easy to pickup and put down. They should be easy to learn to play and should not take an eternity to complete individual levels. Completing the entire game, however, may be a challenge. Gameplay mechanics should be simple and straightforward, with an emphasis on mouse-clicks/drags rather than keyboard short-cuts. Games such as these are particularly popular on mobile devices, where a player can entertain oneself on a 15 minute subway ride. Note that the ease of use and frequent feedback via rewards are common requirements for casual games like Angry Birds and Diner Dash, but this does not have to mean that the game is overly simple. The best casual games should cultivate player skills and reward game advancement with more difficult tasks. Line balancing, for example, is a common task in these games because it can be easy to present in a gaming context, but difficult to master at advanced levels for the gamer, and thus can provide a rich, rewarding gaming experience. So can requiring the acquisition of knowledge, as our game will do. Our game will only be fun if it presents some challenges to the players, where they'll have to learn a bit about the world in order to clear map levels.

2.1.1 System Interfaces

The ***Regio Vinco*** game will need provide an application home, present maps for selection, provide a means for changing game settings, and for viewing help. The various screen modes for this application are specified below in **Figure 2.1**. Note that ***Regio Vinco*** will ultimately exist as a standalone application, though hopefully one that could be ported to and then played on a number of different platforms.

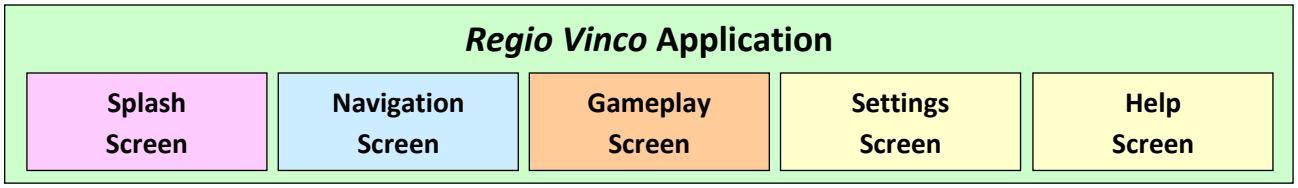

Figure 2.1: The various screen contexts for *Regio Vinco*.

2.1.2 **User Interfaces**

Due to the fact that we'll want to port this to a mobile interface in the future, it is important to keep the interface controls simple. So, we'll need to make sure players will use the application in limited ways. For example, there should only be a few GUI controls inside the game itself, and we'll allow the game to be playable simply by mouse clicks. Figure 2.2 below summarizes the ways with which the user will interact with our ***Regio Vinco*** application, which will be further detailed using UML Use Case diagrams. Note that "Clicks" always refers to a left-mouse-button click. These Use-Case diagrams should be fed as input directly into Section 3.1, external interfaces, which is where the design of the user interface is specified. Here is the full list of UML Use-Case Diagrams:

Use Case	Screen	Use Case
2.1	Splash	Enter Game
2.2	Navigation	Select Sub Region on Map
2.3	Navigation	Go back to Ancestor Region
2.4	Navigation	View Stats/Record for Region
2.5	Navigation	View Stats/Record for Sub-Region
2.6	Navigation & Gameplay	Start Region Name Game
2.7	Navigation & Gameplay	Start Capital Game
2.8	Navigation & Gameplay	Start Flag Game
2.9	Navigation & Gameplay	Start Leader Game
2.10	Gameplay	Try Sub-Region
2.11	Gameplay	Stop Game
2.12	Gameplay	Verify Stop Game
2.13	Gameplay	Cancel Stop Game
2.14	Gameplay	Close Victory Dialog
2.15	Settings & Help Screens	Switch to Navigation Screen
2.16	Navigation & Help Screens	Switch to Settings Screen
2.17	Navigation & Settings Screens	Switch to Help Screen
2.18	Settings Screen	Update Sound/Music Settings
2.19	Help	Scroll Through Help
2.20	All Screens	Exit App

Figure 2.2: Overview of Use-Case Diagrams

Use Case 2.1: Enter Game Use Case

Use-Case:	Enter Game
Primary Actor:	Player
Goal in Context:	From the splash screen, enter the game app where one can start selecting maps to play and then start games
Preconditions:	The game application has been started and the player is viewing the Splash Screen
Trigger:	The player clicks on the "Enter" button
Key Shortcut:	N/A
Scenario:	**1.** Player starts the application, which loads the Splash Screen **2.** Player views the home screen, which is a splash screen with a single button, "Enter" **3.** Player wants to play so presses "Enter" button **4.** Application loads Navigation Screen which displays clickable map of world
Exceptions:	This button should always be enabled from the Splash Screen.
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Used every time the player starts the application
Open Issues:	Size, location, and style of button should be finalized by UI designer

Use Case 2.2: Select Sub Region on Map Use Case

Use-Case:	Select Sub Region on Map
Primary Actor:	Player
Goal in Context:	While on the Navigation Screen, the user may wish to navigate to a different map to play that map
Preconditions:	The game application has been started and the player is currently viewing the navigation screen, which is displaying a game map, on which some Sub-Regions are playable (appear as their greyscale color)
Trigger:	The player clicks on a playable (i.e. greyscale) Sub-Region
Key Shortcut:	N/A
Scenario:	1. Player is viewing the navigation screen, which has a map loaded. Some Sub-Regions may be playable (greyscale), some may be not playable (semi-transparent pink). 2. Player clicks on one of the playable Sub-Regions 3. Game should unload the current map and instead load the map and data for the selected Sub-Region, rendering it in the UI
Exceptions:	Only Sub-Regions with playable maps should be displayed as greyscale colors during navigation as those are the only ones that are playable
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Used likely every time user plays
Open Issues:	Size, location, and style of map should be finalized by UI designer

Use Case 2.3: Go back to Ancestor Region Use Case

Use-Case:	Go back to Ancestor Region
Primary Actor:	Player
Goal in Context:	The user is currently viewing a playable region, but now wants to navigate up to one of its parent regions.
Preconditions:	The player is currently viewing the Navigation Screen with a child map (i.e. not the world) being displayed.
Trigger:	The player clicks on one of the textual ancestor links
Key Shortcut:	N/A
Scenario:	1. Player is viewing the navigation screen, which has a map loaded and the ancestor regions are listed in order as textual links 2. Player clicks on one of ancestor regions 3. Game should unload the current map and instead load the map and data for the selected ancestor region, rendering it in the UI
Exceptions:	The root node (i.e. the world) will have no ancestors, and so will have no ancestor links
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Used likely every time user plays
Open Issues:	Size, location, and style of links should be finalized by UI designer

Use Case 2.4: View Stats/Record for Region Use Case

Use-Case:	View Stats/Record for Region
Primary Actor:	Player
Goal in Context:	During navigation, the user may view their stats for the region they are currently viewing
Preconditions:	The player is on the navigation screen, viewing a playable map
Trigger:	The player has either first entered the navigation screen or has selected a region to view
Key Shortcut:	N/A
Scenario:	1. Player is on the navigation screen viewing a map 2. Player views their stats and accomplishments history for playing that map
Exceptions:	Player stats may not be available for a map if the player has not yet played the current region
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Used likely every time user plays
Open Issues:	Size, location, and style of stats display should be finalized by UI designer

Use Case 2.5: View Stats/Record for Sub-Region Use Case

Use-Case:	View Stats/Record for Sub-Region
Primary Actor:	Player
Goal in Context:	The user may wish to see a summary of their game stats for a number of sub-regions in the currently loaded map
Preconditions:	The player is on the navigation screen and there is at least one playable map as a sub-region.
Trigger:	The player mouses over a playable region
Key Shortcut:	N/A
Scenario:	1. Player is on the navigation screen viewing a playable map. 2. Player mouses over a sub-region 3. If the sub-region is playable, it will display the stats for the sub-region as well as the best player scores for that region. 4. If the sub-region is not playable, it will simply say so
Exceptions:	Note that sub-regions that don't have playable maps should produce no stats, and so should provide feedback to say as such
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Used likely every time user plays
Open Issues:	Size, location, and style of stats display should be finalized by UI designer

Use Case 2.6: Start Region Name Game Use Case

Use-Case:	Start Region Name Game
Primary Actor:	Player
Goal in Context:	Lets the player start playing a game where one has to guess a sub-region's name on the map
Preconditions:	The player is on the navigation screen, which means they must be viewing a playable map
Trigger:	The player clicks on the "Start Name Game" button
Key Shortcut:	N/A
Scenario:	1. Player is on the navigation screen viewing a playable map 2. Player clicks on Start Name Game button
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of button should be finalized by UI designer

Use Case 2.7: Start Region Capital Game Use Case

Use-Case:	Start Region Capital Game
Primary Actor:	Player
Goal in Context:	Lets the player start playing a game where one has to guess a sub-region's capital on the map
Preconditions:	The player is on the navigation screen, which means they must be viewing a playable map
Trigger:	The player clicks on the "Start Capital Game" button
Key Shortcut:	N/A
Scenario:	1. Player is on the navigation screen viewing a playable map 2. Player clicks on Start Capital Game button
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of button should be finalized by UI designer

Use Case 2.8: Start Region Flag Game Use Case

Use-Case:	Start Region Flag Game
Primary Actor:	Player
Goal in Context:	Lets the player start playing a game where one has to guess a sub-region's flag on the map
Preconditions:	The player is on the navigation screen, which means they must be viewing a playable map
Trigger:	The player clicks on the "Start Flag Game" button
Key Shortcut:	N/A
Scenario:	1. Player is on the navigation screen viewing a playable map 2. Player clicks on Start Flag Game button
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of button should be finalized by UI designer

Use Case 2.9: Start Region Leader Game Use Case

Use-Case:	Start Region Leader Game
Primary Actor:	Player
Goal in Context:	Lets the player start playing a game where one has to guess a sub-region's leader on the map
Preconditions:	The player is on the navigation screen, which means they must be viewing a playable map
Trigger:	The player clicks on the "Start Leader Game" button
Key Shortcut:	N/A
Scenario:	1. Player is on the navigation screen viewing a playable map 2. Player clicks on Start Leader Game button
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of button should be finalized by UI designer

Use Case 2.10: Try Sub Region Case

Use-Case:	Try Sub Region
Primary Actor:	Player
Goal in Context:	During gameplay, the user selects a region as a guess
Preconditions:	The player is currently playing a game that has not yet concluded
Trigger:	The player clicks on a sub-region on the loaded map
Key Shortcut:	N/A
Scenario:	**1.** Player is playing the game **2.** Player clicks on a sub-region on the loaded map **3.** If the clicked sub-region matches the sub-region required by the clue at the bottom of the stack (different for different game modes), then it's a successful guess and the game advances with a correct guess **4.** Else if the clicked sub-region has already been found, ignore the guess **5.** Else, the clicked sub-region must be a bad guess, so process it as such
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session for every game
Open Issues:	Size, location, and style of map should be finalized by UI designer

Use Case 2.11: Stop Game Use Case

Use-Case:	Stop Game
Primary Actor:	Player
Goal in Context:	Stop a game currently in progress
Preconditions:	A game is currently in progress
Trigger:	The player clicks on the "Stop Game" button.
Key Shortcut:	N/A
Scenario:	1. Player is playing a game on the current map 2. Player clicks the "Stop Game" button 3. The app checks with the player to make sure they wish to stop the game
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per game session.
Open Issues:	Size, location, and style of button should be finalized by UI designer

Use Case 2.12: Verify Stop Game Destination Use Case

Use-Case:	Verify Stop Game
Primary Actor:	Player
Goal in Context:	The app must verify that the user really wishes to stop the game in progress
Preconditions:	The player has already asked to stop the current game
Trigger:	The player clicked on the stop game button
Key Shortcut:	N/A
Scenario:	1. Player is playing a game 2. Player clicked "Stop Game" button (see Use Case 2.13) 3. Dialog box opens verifying that the user really wishes to stop the current game, providing two options: Ok and Cancel 4. Player clicks Ok button 5. Current game is stopped and the user is returned to map navigation, with the current map displayed
Exceptions:	Note that this should be a modal dialog so no other buttons or controls should be selectable until either Ok or Cancel are selected
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session for every game
Open Issues:	Size, location, and style of dialog and button should be finalized by UI designer

Use Case 2.13: Cancel Stop Game Use Case

Use-Case:	Cancel Stop Game
Primary Actor:	Player
Goal in Context:	The app must verify that the user really wishes to stop the game in progress, and must provide a Cancel option to prevent accidentally stopping the current game
Preconditions:	The player has already asked to stop the current game
Trigger:	The player clicked on the stop game button
Key Shortcut:	N/A
Scenario:	1. Player is playing a game 2. Player clicked "Stop Game" button (see Use Case 2.13) 3. Dialog box opens verifying that the user really wishes to stop the current game, providing two options: Ok and Cancel 4. Player clicks Cancel button 5. Player is returned to the current game in progress
Exceptions:	Note that this should be a modal dialog so no other buttons or controls should be selectable until either Ok or Cancel are selected
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session for every game
Open Issues:	Size, location, and style of dialog and button should be finalized by UI designer

Use Case 2.14: Close Victory Dialog Use Case

Use-Case:	Close Victory Dialog
Primary Actor:	Player
Goal in Context:	The player has completed a map game and would like to return to map navigation
Preconditions:	The victory dialog is currently displayed
Trigger:	The player completes a map game by properly selecting all sub-regions
Key Shortcut:	N/A
Scenario:	1. Player is playing a map game 2. Player completes map, so no more sub-regions remain in stack 3. Victory dialog opens displaying game stats 4. Player clicks "Close" to close the dialog and return to map navigation
Exceptions:	Note that this should be a modal dialog so no other buttons or controls should be selectable until Close is selected
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session for every game
Open Issues:	Size, location, and style of dialog and button should be finalized by UI designer

Use Case 2.15: Switch to Navigation Screen Use Case

Use-Case:	Switch to Navigation Screen
Primary Actor:	Player
Goal in Context:	While viewing either the Settings or Help screens, the player may wish to switch to the map navigation screen
Preconditions:	The game has started and the player is viewing either the Settings or Help Screen
Trigger:	The player clicks on the "Map Navigation" button
Scenario:	1. Player is viewing either the Settings or Help Screen. 2. Player clicks on the "Map Navigation" button 3. Player is taken to the "Map Navigation" Screen with the most recently ***viewed*** map loaded.
Exceptions:	Note that this means the app must keep track of what is the most recently viewed map.
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Many times per session for every game
Open Issues:	Size, location, and style of button & screens should be finalized by UI designer

Use Case 2.16: Switch to Settings Screen Use Case

Use-Case:	Switch to Settings Screen
Primary Actor:	Player
Goal in Context:	While viewing either the Navigation or Help screens, the player may wish to switch to the settings screen
Preconditions:	The game has started and the player is viewing either the Navigation or Help Screens
Trigger:	The player clicks on the "Settings" button
Scenario:	1. Player is viewing either the Map Navigation or Help Screen. 2. Player clicks on the "Settings" button 3. Player is taken to the "Settings" screen with current audio settings displayed
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Many times per session for every game
Open Issues:	Size, location, and style of button & screens should be finalized by UI designer

Use Case 2.17: Switch to Help Screen Use Case

Use-Case:	Switch to Help Screen
Primary Actor:	Player
Goal in Context:	While viewing either the Map Navigation or Settings Screen, the player wishes to view the help screen
Preconditions:	The game has started and the player is viewing either the Navigation or Settings Screens
Trigger:	The player clicks on the "Help" button
Scenario:	1. Player is viewing either the Navigation or Settings Screens. 2. Player clicks on the "Help" button 3. Player is taken to the "Help" Screen
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Many times per session for every game
Open Issues:	Size, location, and style of button & screens should be finalized by UI designer

Use Case 2.18: Update Sound/Music Settings Use Case

Use-Case:	Update Sound/Music Settings
Primary Actor:	Player
Goal in Context:	The player would like to change game audios settings
Preconditions:	The player is viewing the game settings screen
Trigger:	The player clicks the "Settings" Screen button
Scenario:	1. Player is on the Map Navigation or Help screen 2. Player clicks on the Settings Screen button 3. Player clicks on controls to mute or unmute sound and music
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session for every game
Open Issues:	Appropriate size, location, and style of buttons & Screens to be determined by UI designer

Use Case 2.19: Scroll Through Help

Use-Case:	Scroll Through Help
Primary Actor:	Player
Goal in Context:	Scroll the view of the help screen
Preconditions:	The app has started and the player is currently viewing the Help Screen.
Trigger:	The player presses the left-mouse-button on the scroll bar
Scenario:	1. Player is viewing the Help Screen. 2. Player presses the mouse button on the scroll bar. 3. Player scrolls up and down as desired to view different portions of help text
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Many times per game session.
Open Issues:	Size, location, and style of Screen should be finalized by UI designer

Use Case 2.20 Exit App Use Case

Use-Case:	Exit App
Primary Actor:	Player
Goal in Context:	Player wishes to exit the application
Preconditions:	Player is anywhere inside application
Trigger:	The player mouse clicks on the window's X button
Scenario:	1. Application is Running 2. Player clicks on Window's X button 3. Application is immediately terminated, killing all threads
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Many times per session for every game
Open Issues:	Precise size, location, and style of window should be finalized by UI designer.

2.1.3 **Hardware Interfaces**

The game should be designed and constructed such that it may be easily ported and published on multiple platforms. Target platforms should include PC and Android, so implementation of the game should be done in Java. Note that mobile platforms are a natural match for this game because of its point and click simplicity. No complicated keyboard commands or combinations are needed. Note that the application for now should be 1200 x 700.

2.1.4 **Software Interfaces**

Regio Vinco will be developed using the Java language. Note that since this may be our first game of many, and so a Game Framework should be developed in tandem with this application, making it easier to develop future titles. That framework should provide all game timing, the update mechanism, GUI controls management, and an overall structure to support seamless custom rendering. The game will also be developed using Java's JavaFX framework for building user interfaces and rendering 2D graphics. We'll call this framework The Point and Click Game Framework, since it can be used to make simple little games like this one.

2.1.5 **Communications Interfaces**

Note that this game will operate solely as a local game. There will be no networking requirements. At some point in the future the app may be integrated with Facebook accounts, but for now it operates as a standalone application.

2.1.6 **Memory Constraints**

Since this game will be ported to mobile devices, memory is an important issue. All images should be provided in the precise sizes they will be rendered. Images should never be scaled up or down. Note that the memory requirements for each platform will be different, but that memory budgets should be defined carefully before each port.

2.1.7 **Operations**

It is the goal of the player to complete all ***Regio Vinco*** maps in all four game modes and so we'll want to keep track of player achievements in this regard. For each map played we'll maintain the player's fastest win, highest score, and fewest incorrect guesses for each game mode, so we must anticipate saving this data and retrieving and displaying as needed.

2.1.8 **Site Adaptation Requirements**

N/A

2.2 **Product functions**

The game does not need to save multiple accounts data or player settings, only the progress of the primary game needs to be saved and loaded. The assumption is that the game is owned by one person and only one person will play it.

2.3 **User characteristics**

Regio Vinco should appeal to a broad audience, including children, who should find the point and click interface easy to pickup and use and the map content easy to learn.

2.4 **Constraints**

Mobile screen resolutions, present a porting challenge. Different models support different resolutions and all mobile platforms fall short of PC/Mac capabilities. Note that the key hardware to plan for are the mouse (or mobile pointing device) and the screen. Screen size is tricky for our game in particular because we want our maps as large as possible such that sub-region selection is easy. Thus, the large 1200x700 scene size. Note that should the app be ported to a mobile platform, this may have to be adjusted.

2.5 **Assumptions and dependencies**

Note that at the moment we are only developing an initial prototype, which will consist of the full game, but only for a limited number of maps. The application should be developed, however, such that additional maps can be added without having to change any source code. Instead, externals settings files (like in XML) should specify playable levels that can be loaded externally.

2.6 Apportioning of the Requirements

It will be determined at a later date precisely how this application will be ported to additional platforms. It is the obligation of the software designers, however, to construct the program such that it may easily be modified. Again, carefully selected file formats for map data will be important for this feature.

3 Specific requirements

Regio Vinco will use a simple interface with few controls. It is not a fast-paced action game or a complex strategy game, but rather a casual educational (i.e. serious). Note that the GUI controls should always be neatly drawn, carefully spaced, and properly aligned.

3.1 **External interfaces**

The player will only use mouse input to start and run the game, so the GUI will be straightforward. Note that the UI renderings below approximate the look for this application, but the precise sizes and positions of all UI components should be refined at implementation to unsure quality form and function. Figure 3.1 below shows how the Splash Screen should be laid out.

Figure 3.1: *Regio Vinco Splash* Screen

Below is an example rendering of the Map Navigation Screen. Note that in this case all of Afghanistan's provinces are playable in their own right except for two, which is why they are that pale pink. Also note that the player's stats for Afghanistan are displayed below the map while since the player has the mouse over the Nangarhar Province, the stats for the user playing that province are displayed on the right. Note that each stat would be updated if the player were to beat those numbers. Also note that if the player hasn't played a region, stats should simply say "No Games Played". Player stats must be recorded and saved for all played regions. So, should the user close the program and re-open all the stats are remembered.

Now note the buttons for starting various types of games (name, capital, flag, leader), and the stop button to interrupt a game. Finally, note the ancestor links below the game stats for this region as well.

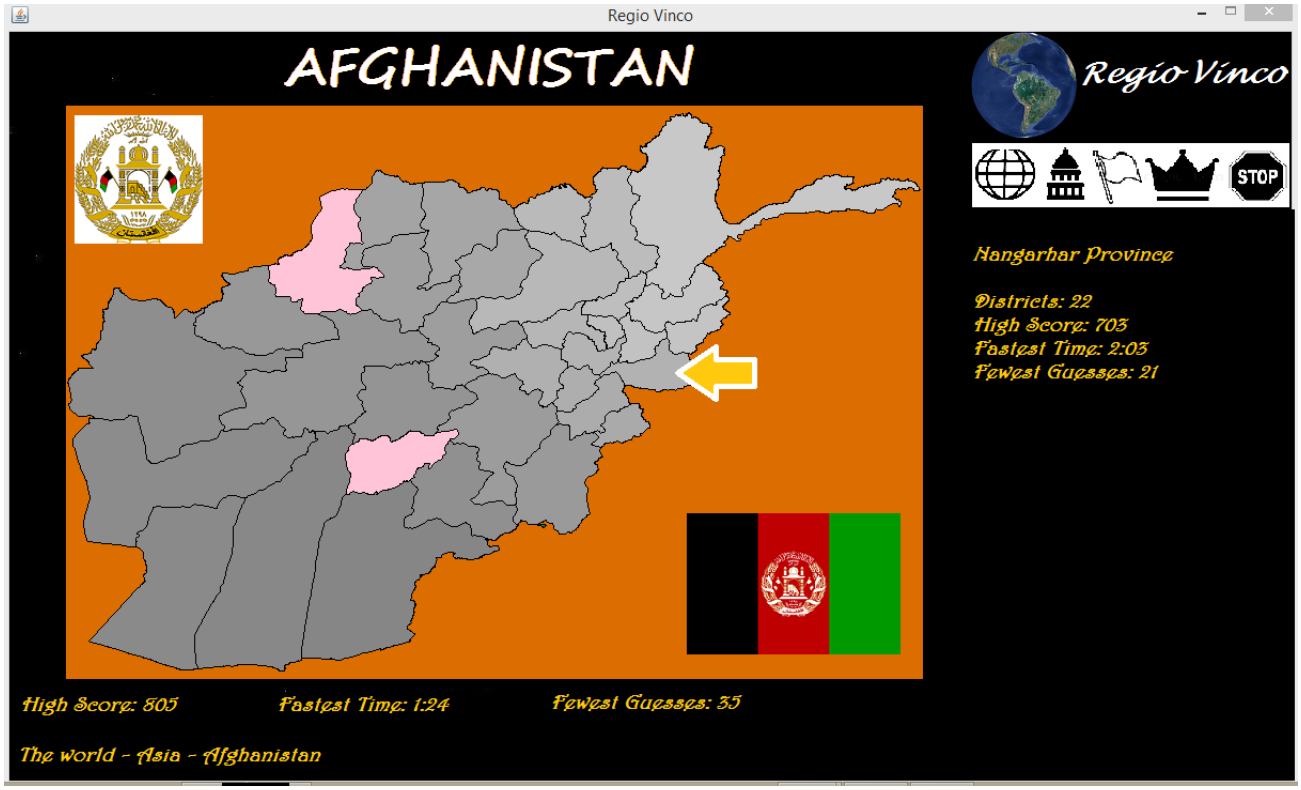

Figure 3.2: *Regio Vinco Map Navigation* Screen.

Below is the Gameplay Screen, which displays the same game buttons, but now has the game stack on the right, and the game stats below the map. Note that the image displays the sub-region name version of the game. Should it be the capitals version, the stack would have the sub-region capital names. Should it be the flag version, it should have the flags for the sub-regions. Should it be the leader version, it should have a stack of the leader names next to a small image of the leader.

Figure 3.2: *Regio Vinco Gameplay* Screen.

Now for the remaining UI features:

- the Settings screen should simply contain Checkboxes for muting/unmuting the sound effects and music
- the Help screen should succinctly describe how to play the game
- the win dialog should display the completed game stats with a nice graphic
- the verify/cancel stopping game dialog need only have a message and those two buttons

3.2 **Functions**

One of the important things to consider in our game application is providing the appropriate feedback to the user while the game is happening. Players need feedback to enjoy their game experience. This is typically done with visual cues or sound.

Game Event Cues

The player will need sound cues for some different scenarios. Note that different sounds should be used for different cues:

- **Correct Map Sub-Region** – this cue should play during gameplay when the user chooses the correct sub-region on the map
- **Incorrect Map Sub-Region** – this cue should play when the user chooses an incorrect sub-region on the map.
- **Map Navigation –** each time the user changes regions during navigation, either going into a sub-region or navigating back to an ancestor region, this sound effect should play
- **Victory** – when a player completes a map, the song for that region should play. This might be a national anthem for a nation, or some other song to represent the region

3.3 **Performance requirements**

The primary performance concerns will be with rendering, since it is a real-time graphical application. The program should run with a target rate of 50 frames per second. Note that gameplay animation should be consistent (and so data updates should be properly scaled) regardless of the frame rate, but the higher the frame rate, the more responsive the program will feel to the player, which is important for games. So, the game should not play faster or slower on different systems.

3.4 **Logical database requirements**

N/A

3.5 **Design constraints**

JavaFX will be used because it effectively leverages each system's available rendering technologies and provides platform independence for personal computers. Therefore the assumption is that the user will have a mouse pointing device. We will not assume any other devices.

3.6 **Software system attributes**

As professionals, all members of this project must take this project seriously. We are dedicated to producing robust software that exceeds the expectations of our customers. In order to achieve this level of quality, we should build a product with the following properties in mind:

3.6.1 Reliability – The program should be carefully planned, constructed and tested such that it behaves flawlessly for the end user. Bugs, including rendering problems, are unacceptable. In order to minimize these problems, all software will be carefully designed using UML diagrams and a Design to Test approach should be used for the Implementation Stage.

3.6.2 Availability –Customers may download and install the game application for free.

3.6.3 Security – All security mechanisms will be addressed by future revisions

3.6.4 Extensibility – It is important that more maps can be added to the game, so file formats for should be carefully considered such that the game can be easily extended.

3.6.5 Portability – To start with, the game will target desktop Java applications. Future ports should be to Android first.

3.6.6 Maintainability – Update mechanisms will be addressed by future revisions.

3.7 **Organizing the specific requirements**

Note that the game genre is simple enough that we need not worry about using an alternative arrangement of the content of this document. The specific requirements for this application already fit neatly into the sections listed in the IEEE's recommended SRS format.

3.8 Additional comments

It is important to keep in mind that the UI designers, map creators, and sound designers should make updates to the game themes and content as need to make something that looks great. It will be to their discretion to design all the interface controls in an effective, interactive style.

4 Supporting Information

Note that this document should serve as a reference for the designers and coders in the future stages of the development process, so we'll provide a table of contents to help quickly find important sections.

4.1 Table of contents

4.2 **Appendixes**

N/A