SS

NORTHERN ARIZONA UNIVERSITY

School of Informatics, Computing & Cyber Systems

Serpent Studios

Team 4 Sej Online

Spring 2023

Sponsor:

Dr. Patrick Kelley

Mentor:

Italo Santos

Team Members:

David Hermann

Johnathan Ray

Tyler Morales

Nick Shugrue

Nickolas Maxwell

Table of Contents

1 Introduction

Sej Online is an online version of an earlier project created by our sponsor Patrick Kelley. The original digitized Sej was a project made for fun that lacks key features which we hope to implement in Sej Online. The key features include two-player networked gameplay, persistent scorekeeping, an elegant user interface, and improved graphics. The project is built in the Unity game engine which includes a testing framework.

In general, software testing is used to judge whether a software product is free of bugs and other defects while it is still in a developmental stage. It does not include concerns stemming from design issues but it may be able to diagnose the symptoms. Software testing is an overarching category that includes numerous different types of testing such as unit testing, integration testing, usability testing, and near the end of development, acceptance testing. Unit testing is concerned with testing individual code functions, methods, or modules. Integration testing judges whether individual functions, methods, or modules work together properly. Usability testing is a black box approach that looks at the user experience in general from the perspective of a user. Acceptance testing judges whether a software project has met the goals outlined in previous design documents and determines whether the project is ready for delivery to a client.

For Sej Online, we plan on conducting unit, integration, and usability testing. We do not include any plans for acceptance testing yet since we have several more weeks of development left. For unit testing, we will use the Unity Test Framework which allows for testing in both "Edit Mode" where the game elements are manipulated through editing tools and "Play Mode" where the game is played as normal with special input parameters. For integration testing, we will use the results of our work with unit testing to combine game elements together in a testing environment. We will have some limited testing for usability but it will not be our focus since the entire UI has been extensively prototyped and tested throughout development. The upcoming sections in this document lay out our testing plans in more detail, beginning with unit testing.

2 Unit Testing

Unit testing determines whether key functions, methods, and/or modules work correctly by themselves with known input/output values. The goal is to determine whether a wide range of values produce expected behavior Typically, in unit testing, portions of the code or system are broken up into measurable smaller “units” These tests are usually automated, and used by devs to determine if the “unit” that has been defined is operating as desired.. For Sej Online, we will use the Unity Test Framework as it is included with the Unity game engine development environment. The Unity Testing Framework includes options to conduct tests in Edit Mode and Play Mode. For Edit Mode tests, we will create test assemblies, which are groups of tests. Tests in test assemblies can be prompted to run as long as no compilation errors are found in the tests or scripts they test. Play Mode tests require separate assemblies from Edit Mode tests. Play Mode tests will allow us to conduct tests while the game is running as if it were being played through the Unity editor. Conveniently, the Unity Testing Framework automatically generates code coverage reports as tests and test assemblies are made. We will use these automatically generated reports to ensure that most important scripts and assets work correctly. A list of scripts, classes, and functions we will test is detailed in subsequent subsections. It should be noted that many of these unit tests do not contain boundary values because the functions/scripts are either called without any values or they generate self-contained values. Additionally, most of these scripts are modular which means erroneous values are impossible.

2.1 StartMenu

- Boundary values: None
- Expected input 1: Game executable started
- Expected output 1: StartMenu initialized, menu aesthetics initialized
- Expected input 2: Any key press
- Expected output 2: StartMenu fades and scene passes control to MainMenu
- Erroneous input: None
- Fallback output: None

2.2 MainMenu

- Boundary values: None
- Expected input 1: Control given from StartMenu
- Expected output 1: MainMenu fades in, set as active
- Expected input 2: Options button clicked
- Expected output 2: Button click sound played. OptionsMenu given control
- Erroneous input: None
- Fallback output: None

2.3 OptionsMenu

- Boundary values: PlayerPrefs values if present
- Expected input 1: Control given from MainMenu
- Expected output 1: Options menu set as active
- Expected input 2: User selects a new resolution
- Expected output 2: The game resolution refreshes itself
- Expected input 3: User selects a new volume with the slider
- Expected output 3: The game updates the volume value
- Erroneous input: None
- Fallback output: None

2.4 ProfileMenu

- Boundary values: None
- Expected input 1: Control given from MainMenu
- Expected output 1: Options menu set as active
- Expected input 2: User inputs a player name
- Expected output 2: The statistics of the player are fetched from the database and displayed
- Erroneous input: The user inputs an invalid username or is not logged in
- Fallback output: The game displays "0" for all values and "Not logged in!" on screen

2.5 DiceScript

- Boundary values: None
- Expected input 1: Script called when dice rolled
- Expected output 1: Randomized direction/force vectors applied to dice for rolling simulation
- Erroneous input: None
- Fallback output: None

2.6 WandScript

- Boundary values: None
- Expected input 1: Script called when wands rolled
- Expected output 1: Randomized direction/force vectors applied to wands for rolling simulation
- Erroneous input: None
- Fallback output: None

2.7 DiceCheckZoneScript

- Boundary values: Dice collision value
- Expected input 1: Dice collision value
- Expected output 1: Script determines which side the dice is on and returns that value to all players connected to the game
- Erroneous input: None
- Fallback output: None

2.8 WandCheckZoneScript

- Boundary values: Wand collision value
- Expected input 1: Wand collision value
- Expected output 1: Script determines which side the wand is on and returns that value to all players connected to the game
- Erroneous input: None
- Fallback output: None

2.9 DBManager

- Boundary values: username, score
- Expected input 1: Script called
- Expected output 1: Script returns whether the user is logged in
- Erroneous input: None
- Fallback output: None

2.10 CloudSave

- Boundary values: username, games played, games won, games lost
- Expected input 1: Script called with a new user
- Expected output 1: The script initializes username, games played, games won, games lost and sends them to the cloud database
- Expected input 2: RetrieveKeys() function called
- Expected output 2: Debug log prints all keys currently stored
- Erroneous input: None
- Fallback output: None

3 Integration Testing

Integration testing is the process of ensuring that the transfer of information between modules is functioning properly. A lot of Sej Online involves getting several pieces of information from one part of the game to another. If a line of code does not execute properly, then it could result in a vital piece of info not being displayed to the user, resulting in, at best, confusion and, at worst, a non-functioning game. By using integration testing, we hope to avoid these pitfalls as often as possible. We plan on observing whether or not data is passed between the different modules properly, specifically to the multiplayer server from the main menu and the actions taken by the users, as well as from the multiplayer server to the database.

When a user navigates from the main menu to hosting an online game of Sej, they are given a six character code that they can give to other users so that they may play with each other. If this code is not displayed to the host user, then other users will be unable to join the game and the host will have no one to play with. We believe that implementing an integration test here would be appropriate: when a user decides to press the button on the main menu that allows them to host a game, they should be brought to a level that has the game itself and a code that they can share with others. Additionally, a user who wishes to join will also have to press a button on the main menu, as well as a text prompt where they can enter the code. Once this is complete, the user joins the game being hosted by another user and the two can play together. If this does not work, then one user is stuck in the game, and the other is stuck on the main menu. Here, another test can be implemented: if the user enters the correct code, then they will join a game that is already being hosted by another user.

Once these two actions occur, an actual game of Sej can commence. During the game, a user can interact with the buttons, dice, and wands found in the scene. These interactions need to be visible not only to the user who initiates these actions, but to the user who is waiting for their turn. If one of the users does not see the interaction occur, not only will it cause confusion on their end, but it will give an unfair advantage to the user who can see it occur. Because of this, an integration test is needed here: if a user interacts with any of the objects in a scene, the server needs to show it to both users. In addition, due to the fact that Sej is a turn-based game, there need to be moments where a user is unable to interact with certain objects. For example, if it is currently the host's turn to throw the wands, the other client should not be able to throw the

wands. If this were to happen, it could result in abuse from users who want to ruin other people's fun. It would be appropriate to have another test here: if it is not the user's turn, then the server should make it so that they cannot interact with game objects in the scene.

When a game is complete, each user's number of wins and losses is updated in a database. The server looks at the scores of the current game and, based on their values, communicates with the database which user won and which user lost. This transfer of data must be precise; if the data does not go through or if the wrong data is sent through, then the database will contain inaccurate information. This will confuse users who want to keep track of how many games they have won and lost. An integration test is needed here to prevent this from happening: if the server detects that a user has won, it will inform the database of this and the database will update the user's won games value. Additionally, if the server detects that a user has lost, it will inform the database of this and the database will update the user's lost games value.

4 Usability Testing

Usability testing refers to examining how users will react to a product. This is vital to the development of a product because it allows for the creators to get direct feedback from potential users. In developing Sej online there must be a large emphasis on user intractability. The core of a game is its playability. Determining playability, however, can prove to be quite difficult. The goal of this would be to measure how usable your product is for the average user. Our project, Sej Online, is fairly linear, making usability testing straight forward. As we will be able to design and test for specific features and usability as the game is designed and production progresses.

That being said, it may be more intuitive to certain users and less intuitive to others. Users who already play video games will be familiar with navigating with and through a game's user interface, as they have no doubt done so before, which is why we intend on making this game's UI similar to those that are already on the market. This practice is common in the industry as many users will prefer a game that has a similar look and feel, allowing them to easily get into the content and not spend time learning how to use the games interface or various other systems. The games industry has spent large amounts of time in developing what are considered the most intuitive and well designed user experiences, that has been slowly learned and used by smaller developers, and as such there is a large wealth of common knowledge to draw upon in developing Sej with usability in mind. However, regardless of industry practices, users who have played little to no video games may still not be as familiar with these tropes. In keeping with this thought, we will take special care in ensuring that every aspect of the game has an obvious purpose that the user can understand easily and quickly. As well as incorporating numerous usability tests catered to our specific platform.

We could begin by starting the game, and analyzing the opening menu for any aspects that may enhance or hinder its usability. This can be done in a professional way by following in tandem with industry standards and guidelines. Next, we continue to analyze each sub menu. Here we can select each of the options and gauge the user's reactions to said options to get feedback. They can interact with each interactable item to see its function and, upon a later exit interview give possible comments. Next the player usability test could transition into getting into a game, either locally or online. Here we could record the user's interaction while loading the game. Finally, we inspect how easy it is to play the game by playing through a few games of Sej

to get the user familiar with the rules and gameplay. Any aspects that take away from the game play, poor HUD, blocky movements or buggy/confusing gameplay could be addressed here. We took all of these aspects into account in pre-development, however we left room for any necessary improvements. The aspects we took into consideration when designing the usability of our program focused mainly on displaying a presentable, user friendly game. Each team member has an inherent background as end users as we have had comprehensive amounts of experience with the game in development. The novelty of the game was accented with enhanced graphics and UI to allow for smooth gameplay for the player.

When analyzing how a user navigates each of the starting menus, we will analyze how quickly and intuitively they can get from the starting menu to other parts of the game. In general, if the user wants to play a game of Sej online, they should be able to click through each of the menus without getting lost and while having a full understanding of what each submenu can offer them. If they can do so without getting confused, then we can consider this test a success. We suspect that users that are already familiar with video games will have an easier time navigating than those who are not, which is why we should pay close attention to how non-gamers navigate the menus. If they are unable to get to where they need to go, then we may want to reconsider how we can better accommodate the game so that it is more intuitive to operate.

In creating the above mentioned, and with a desire to create a professional more suitable usability test for Sej we look to other companies and studios that have created relevant content in the industry for inspiration, and how they defined their own usability tests. It is common in the industry for studios and designers to look at their users in great detail. In order to develop a successful system that can be used, you must know the capabilities and actions that these users may be capable of taking. Once a good baseline of who the user is, what they are capable of doing as far as interacting with the game can begin the testing. Examples of such testing include measuring the difficulty that our users face when performing certain actions within the system. In utilizing the Unity test framework we are able to access various bits of data that allow us to formulate statistics that allow us to analyze various things like user menu navigation, how long it takes for our users to access to the areas of the game desired. We can view such things as how often users adjust volume, to better judge the proper levels or audio. This allows us to make the adjustments necessary that allow users the experience desired. User exit interviews can be done

for play testers to help determine what they felt was functioning correctly, and what they felt made it more difficult to play the game. Testing is done on Sej across multiple devices to monitor the usability on the vast amounts of hardware available to users. By ensuring that Sej can be played on as many machines as possible we guarantee maximum device usability to users.

As usability is based on the features developed, it will progress in short bursts as Sej development continues, by selecting users of the developers to be our play testers, composed of friends and family the group is roughly half regular game players, and half composed of non game players. Initially tests are made to determine connectivity is working. We then allow the users to test various functionality as it is developed, functions like using codes to join games, interacting with the interface and getting appropriate output such as animations, game information, or audio cues. We follow this course with more testing of later features, graphics/ audio and user appeal, clarity of the in game rules for Sej users and the way we display them in game. These various stages continue to the end of Sej development where users are offered an exit interview to gain final insight on the nearly finished product.

Conclusion

To conclude, this Software Testing Plan for Sej Online outlines three critical sections of testing to ensure the project is as bug-free and stable as possible. Each of our unit tests outlines the most important functions, modules, and scripts we plan on testing and ensures that all the minute details of our game function as intended. The integration tests outline how the large portions of the system, such as our database integrations, game rules, and the multiplayer server, work together and how information is passed between each of them. Finally, the usability tests focus on tests that measure how simple it is for a user to interact with the game and whether the results of those tests meet our standards for an easy to use product. Ultimately, it is our goal to make Sej Online as error-free as possible with these testing strategies. We believe that, by creating tests that focus on displaying the correct information to the user and preserving data between modules, any user who picks up the game can intuitively understand what is going on, will have a fun time playing it, and will want to come back to play it again.