Developing a Roguelike Deckbuilding Design Tool with Slay the Spire Modding API

Hoang Luu

Computer Science
Bachelor's Thesis
15 ECTS Credits
Spring 2023
Supervisor: Alberto Alvarez

Acknowledgement

I would like to express my deepest appreciation to my supervisor, Alberto Alvarez, for his support and advice throughout the making of this thesis. I am also very grateful for the feedback which I received from Jose Font, the examiner, as it has helped me improve my thesis. Last but not least, I would also like to give my thanks to my opponents, Mikael Bernau and Lovisa Johansson, for their greatly detailed constructive reviews, which helped me improve my thesis even further. Once again, thank you.

Abstract

Dream Quest(2014) introduced the Roguelike deckbuilder as a game genre. The genre later became popular with the help of Slay the Spire in 2017. As the genre is still fairly new, there are not a lot of tools for designing games of said genre. Taking inspiration from games of the same genre could be a solution to such a problem, one way to do so is via modding. Slay the Spire offers an extensive modding API to users which lets users create new cards, items, events, and even characters. By utilizing these features, it is possible to create a concept for a new game. To use the API, however, would require users to be familiar with programming, which could take a long time to learn for a beginner. Thus, a UI, The Creator was created to let users without experience create mods for Slay the Spire. The UI lets users create cards with different actions and properties without the need to interact with the source code.

Main field: Computer Science
Program: Game Development
Date of final seminar: 29/05/2023

Examiner: José Font

1. Introduction

Roguelike deckbuilder is a game genre that combines deckbuilder with roguelike elements[1]. Deckbuilder is a genre where you gradually gather cards and build a deck of cards[2], similar to the classic Trading Card Game (TCG) *Magic: The Gathering*[3]. While roguelike is a genre where you proceed through a procedurally generated map or dungeon and start anew every time you die[4]. So a roguelike deckbuilder is a game where you would traverse a dungeon while collecting cards to build a deck to battle enemies. The genre first started with Dream Quest (2014) and later on, became a popular genre with the release of Slay The Spire (2017)[1].

As the genre is fairly new, there are not many resources when it comes to designing a game that would fit into the genre. Thus, to design similar games, one would take inspiration from games within the genre that are popular, to get an idea of how a roguelike deckbuilder should be designed. One way to do so is via modding, which is the act of modifying the game's files to create new game content or alter them as well as remove them. This allows you to tweak games and get a what-if view. By doing so, you will be able to see why the game was designed the way it is and why something works while something else doesn't. However, to be able to mod a game, you would need knowledge of programming, which could be a problem for non-programmer designers. Wouldn't it be nice to have some kind of tool that would let users create content for a game without the need to interact with the game's code itself? This question can be answered with the help of Slay the Spire(StS).

StS is a single-player roguelike deckbuilding game developed by the studio Mega Crit[5]. As the player plays the game, they will build their deck along the way while battling enemy AI:s in a turn-based fashion. There is some research that uses the game as its research platform. A recommendation system that recommends the best possible actions while playing the game with the help of trained neural networks, Holm and Takman [6]. A pathfinding system that helps with finding the most optimal path with the highest rate of success using machine learning, Porenius and Hansson [7]. Lastly, an analysis of randomness within the game to determine whether a player is skilled or just lucky while playing the game, Trojanowski and Andersson[8].

The game also allows modding. StS has an extensive API that allows players to modify the game, two of previously mentioned research used the API to carry out their research [6],[7]. With the API, it is possible to create new relics, new cards with new effects, new potions, new keywords, new game events, and even new game characters; these game's related mechanics will be further explained later on. It is also possible to alter the game's existing content, for instance modifying a card's value. By utilizing these features, it would be possible to use this API and StS as a design tool to design/create new roguelike deckbuilding games.

However, to use the API, one must have some knowledge of programming, specifically Java as that is the language in which the API was written. This means that game designers, specifically gameplay designers, without coding experience as well as non-programmers will have a hard time if they were to try and create a mod for the game while in the process of designing their own game. The API does provide good documentation of what it can do and how to do it, however, it would still require the user to be familiar with programming or learning the programming language. Unfortunately, this endeavor is not a simple feat, it is time-consuming and frustrating more often than not. Designers and non-programmers with great ideas could shy away because they cannot afford the time and effort required. An example of what the API looks like can be seen in Figure 1.

The method which is going to be utilized is design science. The issue mentioned above can be solved by creating a UI (user interface) based on the provided API. Meaning if we let designers and non-programmers play around with the features provided through this UI instead of letting them interact with the code, this will allow them to save time from learning to program and put more time into designing. The API also allows the modification of a card's value while in play which means it will be easier for designers to balance cards as they can test the cards with different values in real-time.

The purpose of this research is to create a design tool based on the provided API that interfaces with StS. This design tool will not only allow designers to create game content for StS but it can also help them design their own roguelike deckbuilding game using what they have created with the tool as a base for their game.

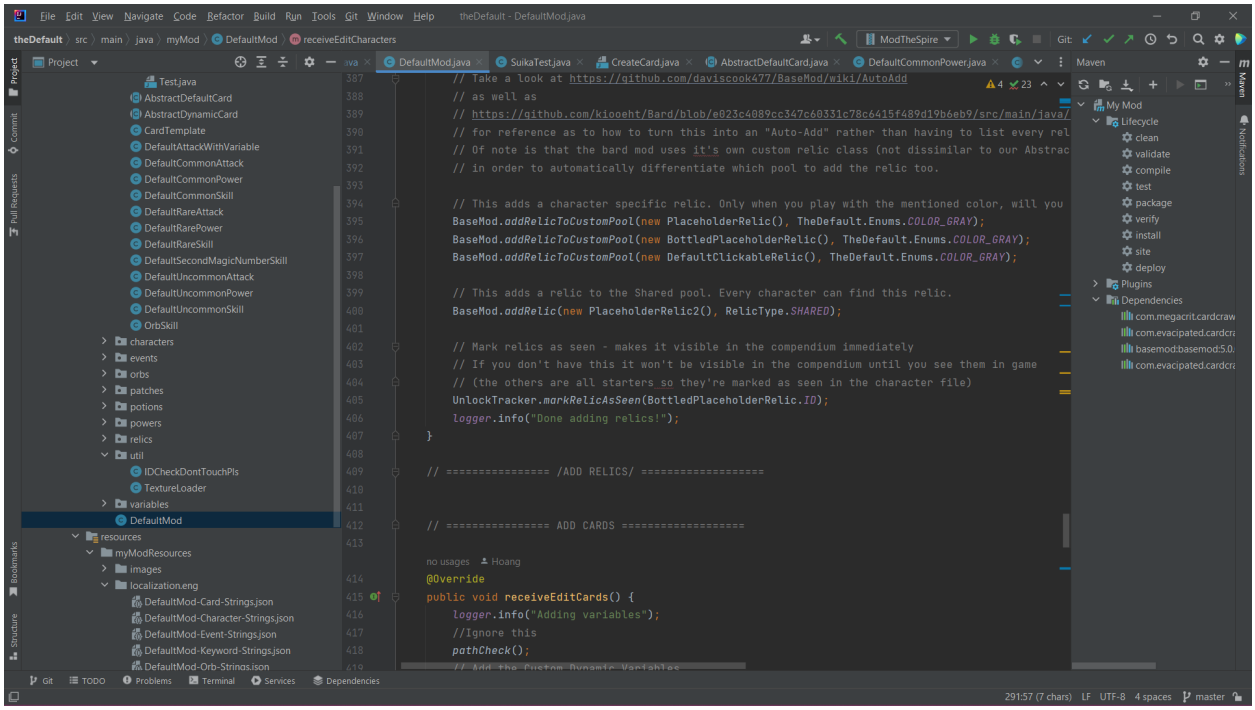


Fig 1 The modding API through an IDE.

1.1 Research Question

RQ.1: What type of features provided by the API can be integrated into the UI?
RQ.2: What are the benefits as well as drawbacks of using the artifact compared to the usage of the API?
RQ.3: How efficient is the UI when used as a design tool, regarding content creation and balancing?

2. Slay The Spire

This section provides knowledge about the game which is required to understand the concepts described in the introduction as well as later sections of this thesis. Information provided in this section is terminologies used within the game as well as game mechanics. All information in this section is compiled from the Slay the Spire wiki [9].

2.1 Gameplay

Slay the Spire is a game where you climb The Spire, there are a total of three acts(levels) plus one special act where you need to meet a certain condition to get to it. Each act has floors that players need to ascend to beat the game. Once they beat the game, they will unlock Ascension, which essentially is just a difficulty level that increases each time the game is beaten with that difficulty level, there is a total of 20 levels of ascension. As the players play the game, they will encounter enemies to fight, events that either grant bonuses or inflict negative effects on the player. After each fight, they will get to choose from one of three random cards to put in their deck. Players can also collect cards by encountering certain events or buy them from the shop. At the end of each floor, there will be a boss that needs to be beaten to continue upward. As players play the game, they will gather cards to build their deck, and gain items in the form of relics that give permanent effects and also potions that have various effects. When they die, they will lose all of their relics and their cards that they have collected throughout the game. There are currently four characters that can be chosen from before starting a playthrough (a run).

Fig 2. Engaging in a battle against a boss in order to ascend to the next floor.

2.2 Terminology

- **Relics:** Relics are items that provide permanent passive bonuses for the player throughout a run. Most relics are universal while some are unique to each character.

- **Character:** Before starting a playthrough of the game, the player can choose one of four available characters, Ironclad, Silent, Defect, Watcher. Each of these characters has its own unique deck of cards and a color theme.

- **Events:** Events are special scenarios that players can stumble across during a run. They will either provide something positive or negative to the player, this is random.

- **Potion:** These are one-time-use items that provide different bonuses. They can only be used during combat with some exceptions that can be used outside of combat.

- **Deck:** A place that gathers all the cards that you possess.

- **Draw Pile:** All cards in your deck will be in this pile as combat starts. When you draw cards, you draw from the draw pile.

- **Discard Pile:** This is the pile your cards go to after you have played them, at the end of the turn, all unplayed cards will also be sent to this pile. There are however exceptions to this mechanic, for instance, if a card contains the Keyword Retain, it will not be discarded at the end of the turn. If your draw pile is empty when you draw a card, the entire discard pile is shuffled into the draw pile.

2.2 Cards

2.2.1 Card's Type

- **Character-specific cards:** There are many different types of cards with different rarities and colors. The color of a card dictates which character it belongs to, these are called colored cards. For instance, A red card would belong to the Ironclad while a green card belongs to the Silent. When obtaining cards, which is after a battle is finished, or when an event is encountered, or buying from the shop, only colorless cards and cards relevant to the chosen character will appear. There exists a relic that allows the appearance of other colored cards.

Fig 3. Ironclad's strike card.

Fig 4. Silent's strike card.

- **Card type:** Each card is associated with one type. There are a total of five types in the game: Attack, Skill, Power, Status, and Curse.

    1. **Attack Card**: Usually deals direct damage to one or more enemies, with some adding a secondary effect.

    2. **Skill Card**: All sorts of temporary and strategic effects are in this card type. Commonly grant "Block" or other temporary buffs to the player or debuffs to one or more enemies.

    3. **Power Card**: A permanent upgrade for the entire combat encounter. Some Powers give flat stats like "Strength" or "Dexterity", others require certain conditions to be met or add beneficial effects whenever a certain event occurs in battle. Once played, the Power card is removed for that combat.

4. **Status Card**: Cards added to the deck during combat encounters that are designed to fill the deck, preventing the player from drawing beneficial cards. Some Status cards also have additional negative effects. Status cards are removed from the deck at the end of combat.

5. **Curse Card**: Unplayable cards added to the deck during in-game events. Similar to status cards, they are designed to fill the deck and prevent the player from drawing beneficial cards, with some of them having additional negative effects during combat. Unlike Status cards, Curse cards persist in the players' deck until removed by other means.

- **Universal cards:** Besides colored cards, there exist three other types of cards that are universal to every character, which means that any character can obtain these cards. They are colorless cards, status cards, and curse cards. Colorless cards are playable cards that do not belong to any character. Status cards and curse cards are unplayable cards that affect the player negatively as mentioned.

- **Card's rarity:** The rarity of a card can be seen in the color of a card's banner behind its name, this applies to all cards. There are three types of rarity, Common cards have a gray banner, Uncommon cards have a blue banner while Rare cards have a golden banner. Additionally, there are two other rarities that the game recognizes even though their banner color is gray like common cards. These are Basic cards and Special cards. Basic cards are default cards from the starting deck of a character and special cards are only obtainable through events or other special means.

Fig 5. Different card rarities.

2.2.2 Card's Traits

- **Card's Keywords:** Keywords are shorthand terms used to describe specific actions or statuses. For instance, a curse is a type of keyword, which is associated with curse cards. In Figure 5, the highlighted golden texts are Keywords.

- **Card's States:** Card states are keywords that explicitly affect the cards. There are five states and these are:

    1. **Innate**: Causes the card to always start in your opening hand
    2. **Retain**: The card stays in hand after the end of the turn instead of being discarded.
    3. **Exhaust**: Once played, the card will be removed from combat and cannot be used again for the rest of the combat.
    4. **Ethereal**: Automatically exhausts at the end of turn when in hand.
    5. **Unplayable**: cannot be played normally and does not have a cost.

- **Card's Actions:** The effect of the card. For instance, the "Defend" card in Figure 5, describes one action, and the action is to gain five blocks that block damage from enemies' attacks. A card can have more than one action, for instance, the Anger card below has two actions, the first one is to deal 6 damage, and the second action is to add a copy of the card into the discard pile.

Fig 6. Anger card

- **Card's cost:** Each card will have a certain cost, energy. Players will need to have enough energy to play a card corresponding to its cost. Unplayable cards however don't have a cost as they are unplayable.

3. Background & Related Research

3.1 Roguelike deckbuilder

The modern iteration of the genre came from Peter Whalen's *Dream Quest* in 2014, the player ventures into a dungeon in the form of grids and battles monsters with their deck of hand-drawn-stick figure cards[1]. Despite it being the first of its genre with its uniqueness, it didn't invoke a lot of attention. However, in 2017, Mega Crit, founded by Anthony Giovannetti and Casey Yano, took the fundamentals of *Dream Quest* and created *Slay the Spire*[1]. This game was first released for Windows in November 2017 as early access, which later became a fully released game in January 2019. During its early access stage, the interest in the game was low but this quickly changed as it gained more and more attention thanks to online streaming and videos of the game. It had over one million units sold by June 2018. By March 2019, only two months after its full release, sales of the game reached 1.5 million[10], [11].

3.2 Modding

Modding is the act of modifying an existing game to either create or tweak content of that game[12]. Game modding can also be recognized as a leading method for developing/customizing game software[13]. According to Scacchi, there are at least four types of game mods: user interface (UI) customization, game conversions, machinima, and hacking closed game systems[13]. Game conversions is the focus of this thesis. Game conversions can be subdivided into partial conversions, and total conversions.

Partial conversion is to add or modify (a) the appearance and/or capabilities of in-game characters, including both user-controlled and non-player characters (NPC); (b) game objects such as weapons, potions, spells, and other resources; (c) levels, zones, maps, terrains or landscapes; (d) game rules or (e) game mechanics[13].

Total conversion is to create an entirely new game from an existing game. For instance, *Counter-Strike* (CS), released in 2000 by Valve, was initially released as a mod for *Half-Life* in 1999 by Minh "Gooseman" Le and Jess Cliffe [14]. CS has now become a game with the highest player rate on steam. Another popular example is *Defense of the Ancients*(DotA), which was a mod for *Warcraft III: Reign of Chaos*, created in 2002[15]. In 2013, DotA 2, the sequel to DotA, was released by Valve as a standalone game[16]. Inspired by DotA, Riot Games released *League of Legends* (LoL) in 2009[17], which became one of the most popular titles to date[18]. Another recent popular title, *Auto Chess*, released in 2019[19], is a mod created from DotA 2, the title was so popular that it spawned a whole new genre, 'Auto battler'.

Modding as a design practice has several benefits. Nowadays, it takes a team of developers and programmers as well as money to release a game, because of that, game companies are very careful with what they produce, which constricts designers[20]. With modding, however, these constraints are not present, because sales and profit are not a part of it, designers have the freedom to be creative and play around, which is something bigger companies cannot afford to do[20]. Modding, fundamentally, is also less stressful than the process of designing and developing a game from scratch[21], because most of the work is already done. The base game is already there, the only thing left to do is to modify the game as you like. Testing times are also greatly reduced because the game must have undergone a testing phase before being released as a commercial product. Modding can also be used to create credible games, as gamers today can be highly critical[20], they are more likely to use the mod of an already accepted game than a game without origin.

3.3 GUI integration

A designer's job is to create visual presentations of concepts and ideas[22], therefore, programming skills are often not a part of their skill sets. Their responsibilities lie not in the functionality of an application, but rather in its appearance. How are designers then able to give the application its look? The answer to this is Graphical User Interface(GUI). GUI allows designers to point-and-click, drag-and-drop predefined components to create the look of an application. With game design tools, instead of just looks, they are able to also design game content without the need to look at any code. From a broader point of view, GUI enables end users to interact with the underlying code without the need to understand the said code, making it easier for them to understand and use the application.

Research laboratories and academic institutions can develop advanced software for scientific applications but often lack proper documentation. This makes the application very difficult to use by anyone other than the author of the code, meaning this would also be difficult for non-programmers to use[23]. The complexity of the software and the various options available can also lead to input errors and compatibility issues. To address this, a new GUI development kit called Rappture[24] was introduced, which generates an interface for a specific application and specifies required variables, reducing input errors. Zori, a quantum Monte Carlo (QMC) program developed by Lester group at the University of California, Berkeley[25], is a program package developed over 4 years, to optimize wave functions and develop QMC methods. The package includes algorithms for handling large molecular systems. This package also suffers from the aforementioned problem. With the help of the Rappture toolkit, Olivares-Amaya Et. al were able to generate a GUI, Zopi(Zori Processing Interface)[23], for the Zori computer code which made it easier for users to access and use the software.

3.4 Frameworks Developed from Modding

Many frameworks have been developed from game modding. For instance, the Dota 2 AI framework, created by Tobias Mahlmann, is a framework used in the Dota 2 Bot competition where participants submit AI controllers for the MOBA game *Defense of the Ancients 2* (Dota 2) and let these AI:s engage in 1v1 battles[26]. The framework makes use of the modding capabilities that the game has which enables participants to connect external AI controllers to Dota 2 matches..

3.5 Balancing

One important aspect of roguelike deckbuilding games is balancing. The balancing of cards, usable items, passive items, playable characters as well as NPC, et cetera. As all these tasks intertwine, they create a complex system and it will not be a simple matter to balance that system, which poses a great deal of problems for designers. The evaluation of the system depends on the playtests with human players, which can be costly as well as time-consuming. it is therefore better to automate such processes with the help of AI[27]. "*Demonstrating the Feasibility of Automatic Game Balancing*", is a research project conducted by Volz et. al. using the game *Top Trumps*[27]. For the research, they use simulation- and deck-based objectives on the game. They describe a process for creating a deck of cards that is optimized to create a fair and exciting game of Top Trumps, based on the win rate and average number of tricks of the cards. Using a multi-objective evolutionary algorithm, they generate a deck of cards. This algorithm tries to find a set of cards that achieve specific objectives, or goals, related to the fairness and excitement of the game. The results show that it is possible to apply the presented approach to more complex games, for instance, real-time strategy games.

Another research that was carried out regarding card game balancing is "*Evolving the Hearthstone Meta*", by Silva Et. al. Hearthstone has over 2000 cards, and to be able to tune all of them without disrupting the game system would not be a simple task. The approach that was taken by the authors for this problem was to analyze the win rate of match-ups of different decks, played with different strategies, then compare their performance from before and after tuning different cards. Then, search for combinations of changes to card attributes to make them approach 50% win rates using an evolutionary algorithm[28]. Then, also utilizing multi-objectives, they expanded their algorithm to search for the result, without making too many changes to the existing cards. For their conclusion, they proposed heuristics from the result that can be used to determine the target of balance changes, which are the cards in this case.

4. Research Methodology

The chosen methodology for the proposed research questions was design science research[29]. It is a model of guidelines for how research should be carried out, it consists of six activities, and these activities were described by Peffers as:

1. **Problem identification and motivation**: “Define the specific research problem and justify the value of a solution”.

2. **Define the objectives for a solution**: “Infer the objectives of a solution from the problem definition and knowledge of what is possible and feasible”.

3. **Design and development**: “Create the artifact. Such artifacts are potentially constructs, models, methods, or instantiations”.

4. **Demonstration**: “Demonstrate the use of the artifact to solve one or more instances of the problem”.

5. **Evaluation: “**Observe and measure how well the artifact supports a solution to the problem”.

6. **Communication**: “Communicate the problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness to researchers and other relevant audiences, such as practicing professionals, when appropriate”.

Following these activities, it helps researchers do research with processes that are understood and accepted. It will be easier to research as there are guidelines to follow as well as making the research reproducible if somebody wants to do research of their own on the same subject.

4.1 Problem Identification and Motivation

As previously mentioned, roguelike deckbuilder is a fairly new genre, which means there are not yet many resources when it comes to designing such a game. Specifically, a design tool to help designers create and design cards. Slay the Spire modding API can be used as a base for the said tool, as it offers many different elements to create a fully functional mod for the game. However, to be able to use the API, the user must need some knowledge of programming. Even though the API is filled with comments to help users understand what to do, it will still require the user to have some basic knowledge in programming to be able to use the API efficiently. Learning a programming language is a time-consuming process, especially if the user has never coded before. This can repel users with potentially great ideas for a new game inspired by Slay the Spire, wanting to

use the API as a base for their game, because they do not want to invest their time into learning the language.

4.2 Define the objectives for a solution

The goal is to create a UI integrated with the API to let the users interact with the API without a need to learn Java. The UI would allow users to create things that the API can, for instance, cards, relics, events, and even potions. The developed artifact should offer easy usability, which is defined as how satisfied users are when using it. This is important because the artifact is treated as a design and creation tool to help users design and create custom content for the game without prior knowledge of programming.

4.3 Design and development

The artifact is developed using Java, java swing in Intellij[30] with the help of Slay the Spire example mod TheDefault[31], and three other modding libraries, baseMod[32], StSLib[33], and ModTheSpire[34]. The UI is created with the help of java swing as it offers UI-related classes which have different components such as buttons, text fields for user inputs, checkboxes, et cetera. It offers “click and drag” a component(s) from a list of components to a canvas without having to code the properties for the component itself. The core purpose of the application is to help the users create content for the game, it is removing the coding part of the development from the user, what the users need to do is play around with different options and write in their desired inputs in dedicated fields.

4.4 Demonstration

The artifact as of now can only be used to create cards. When creating a card, a list of actions will be present for the user to choose from, what actions they want the card to have. The artifact can be used to create complex cards with different parameters and different actions. It is possible to add all available actions from the created action list to a card and that card would still be functional, which means there are many different combinations of different actions and they are only limited to the user’s imagination. There are however some actions that are far too complex to add to the action list. There are also simply too many actions to completely add them all to the UI within the given time. From all actions that are available in the UI, it proves that theoretically, it is not impossible to add every available action from the API.

4.5 Evaluation

The evaluation process of the artifact mainly consists of functionality testing(qualitative evaluation), which is to test every available feature provided by the artifact and make sure they work as intended, meaning a card created from the artifact shall behave the same way as a card created from the API. The test is in the form of an experiment, creating cards with each action as well as creating a card with every available action to see how it will behave, comparing the card-creating process between the API and the UI. The artifact can be seen as an extension to the API as it takes features of the API, which requires programming knowledge to use, and puts them into an application that lets users interact with the features without needing to learn JAVA. The capabilities of the UI will also be compared with the capabilities of the API, in other words, what can the UI do that the API can't and vice versa.

There will also be a user study, focusing on the UI's usability and the card creation iteration. Testers will be given the freedom to play around with the artifact and create cards on their own with minimum guidance, they are also given a guide if they want a more concise explanation[35]. Testers are also allowed to create one card however they like, then use it against a boss from act one. This eliminates random factors that would appear when traversing the map in a normal playthrough. While fighting the boss, users are encouraged to go back and forth to the UI to tweak the card's stats, they do not need to try and balance the card, as the purpose of the test was to test how smooth the iteration is. After they are done testing the artifact, they are given a questionnaire. The questionnaire is divided into two parts. The first part focuses on usability, they are to answer these after they have created their first card. The second part focuses on the iteration, they are to answer these after they have played the game while making changes to their card. The starter deck of the testers was all the same except for their unique card.

It is not necessary to have testers test the API without the UI because in order to use the API, the testers will need to have some form of programming knowledge, which is a problem this thesis tries to solve. Explaining how to use the API to the testers could take a lot of time, which can affect the test result. The API also has a lot of modules. For testers to use the API effectively and to compare it to the UI, they would need to understand how these modules are connected. It took around two weeks for the author, who has zero experience with the API but have programming experience, to be familiar with the API. This shows the API's complexity and it would not be a wise decision to force testers to test something that they could perceive as tedious. This is because it could discourage them which in turn can generate a biased result.

4.6 Communication

The artifact tries to solve a problem regarding creating a mod without programming knowledge. A game designer's job is to conceptualize, implement and maintain gameplay systems that are considered fun by players[36]. Even though the artifact only offers card creation, it is possible to develop it further to include the creation of relics, potions, and events as mentioned using the already implemented card creation system. When all these things are included, the artifact can be a powerful tool that designers can use to develop new content for Slay the Spire which could serve as a base for a new game of the same genre.

5. The Artifact

It consists of two parts, the UI and a file generator is the first part, and the second one is the battle data collection. The process for developing the artifact was done iteratively, as new actions are discovered, they are added to the list of available actions. The artifact has been dubbed The Creator.

5.1 UI & File Generator

The UI is developed using java swing as mentioned. When the application starts, the user will be able to choose a card type from a list as well as input the name of the card(figure 7). After the ‘Create New Card’ button is clicked, the user will be able to choose where to put the card file in the computer, for the card to work, however, it must be within the DefaultMod directory. After choosing a location, the user will be taken to the main part of the UI, the card creation view(figure 8). Creating a card with the name of an already existing card will however load in the content of that card, which means the user can edit an existing card by simply typing in the name of that card.

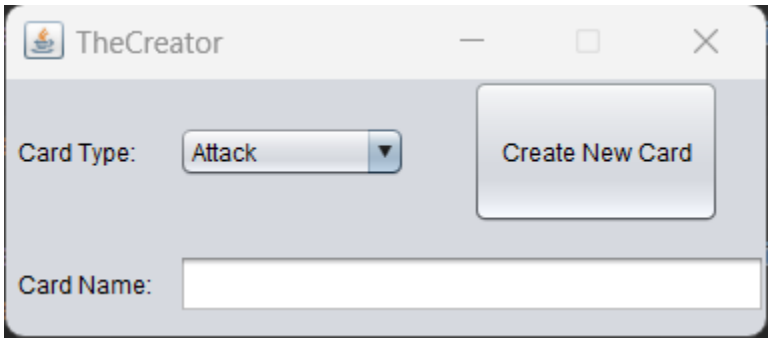


Fig 7. File creation

In the view (figure 8), there will be a list of actions with different categories. From this list, the user can choose to add actions to the card, the chosen actions will be displayed in a table. Within this table, the user can input some values as well as give an action

different properties. From the table, the user can also choose which action to be removed. The user is also provided an option where they can add/remove an action upon an upgrade of the card.

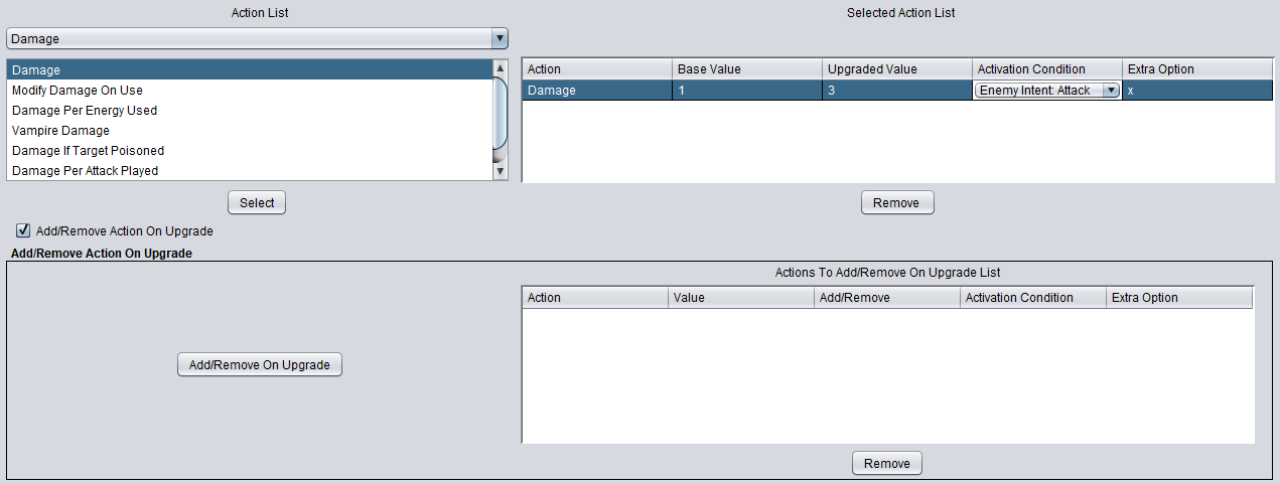


Fig 8. Action adding

Once the user is done with adding actions to their card, they are to input the cost as well as the upgraded cost for the card, choosing a rarity for the card and the target of the card (figure 9). They are also able to write a description that will appear on the card, a different description upon upgrade is possible if they so desire by checking a checkbox. In the description, they can use keycodes, which are used to display an action's value on the card that changes as they play the game, for instance **!D!** would display the damage value of a card. There is also a checkbox to determine whether the user wants the card to be seen in the card library without the need to encounter it, this is checked by default.

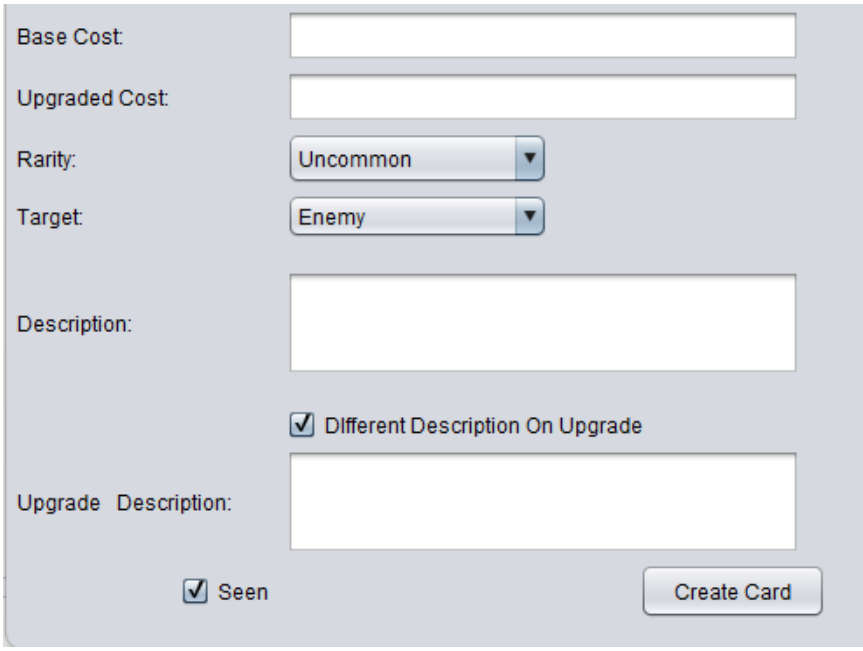


Fig 9. Cost, rarity, target, description

The user can choose to add a state to the card as well as to remove them upon upgrade by checking the boxes (figure 10). Checked means they are active, unchecked means they are not active. Unplayable is not present because the API treats this state differently. Creating a Curse or a Status card will automatically give the card the unplayable state because that is how the game treats these card types.

Fig 10. Card States

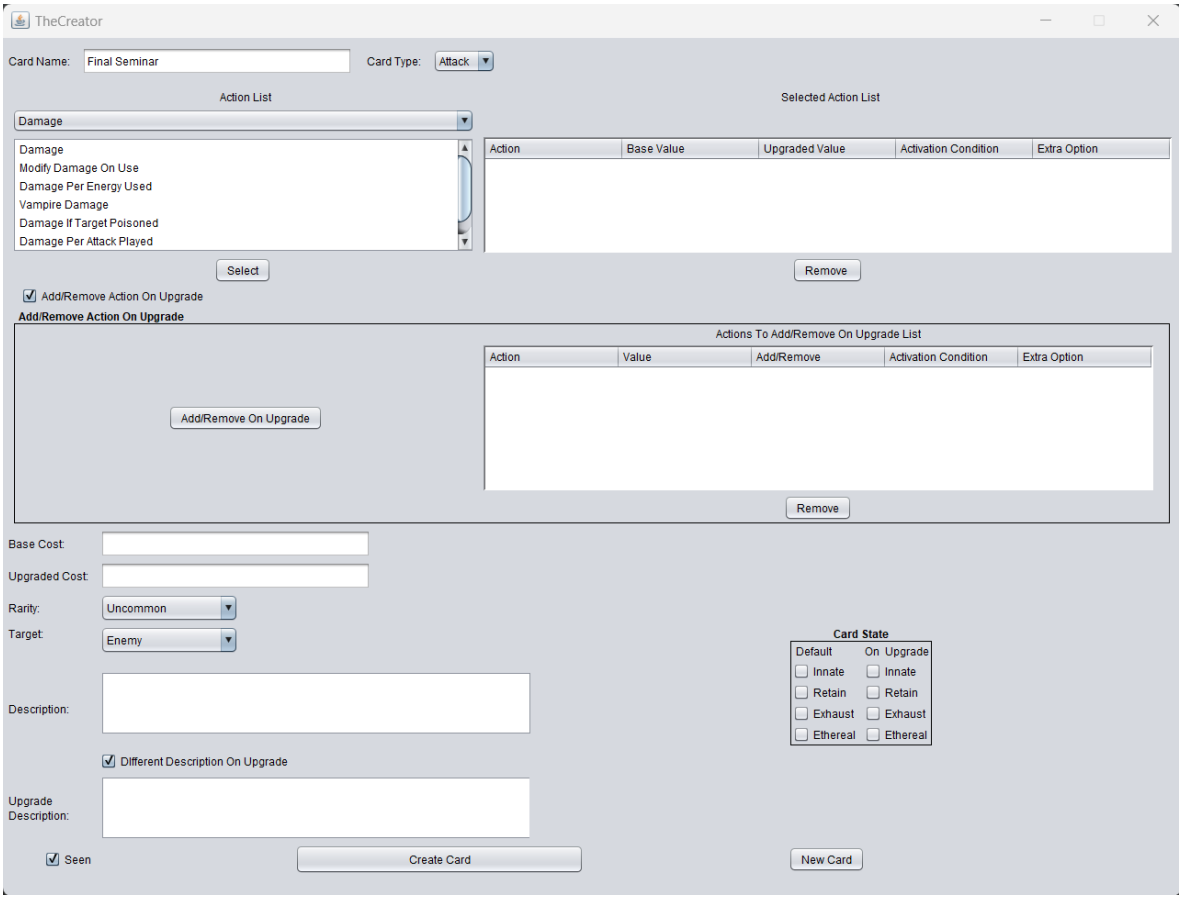


Fig 11. Full view of the UI

The file generator is built by utilizing string builders. It receives input from the UI, parses them, and saves them to variables that are used by different parts of the card. When actions are added to the table and a card is created, the actions will pile up and sort themselves into categories depending on their chosen properties. Once everything is filled and finished in the UI, the generator will receive these data and add them to the card file accordingly, thus creating a card. This data will also be saved to a text file, which will be loaded in the UI when the user wants to edit a card as mentioned before.

5.2 Battle Data Collection

As the name suggests, it collects data from each battle the players encounter while playing the game. It collects data from each turn, writes it to a file, and at the end of the battle, there will be a summary of everything.

The collected data are: What act(level) the player is on; battle order; player's and enemies' health; potions the player currently owns; enemies that the player fights; powers currently applied to the player; damage that the player has done to enemies' health(excluding shield), it also collects if enemies had healed to show consistency with the collected health of the enemies; how much blocks player has gained as well as damage player receives (including shield); cards that were used and their quantity (how many times they were used); cards that were drawn and their quantity; cards that were exhausted and their quantity; what potions were used and their quantity. All things that were mentioned are collected each turn, and at the end of the battle, the total summary of everything will be calculated and also written out, for instance, total damage, the sum of all damage the player had dealt each turn.

Fig 12. Result of Turn 1

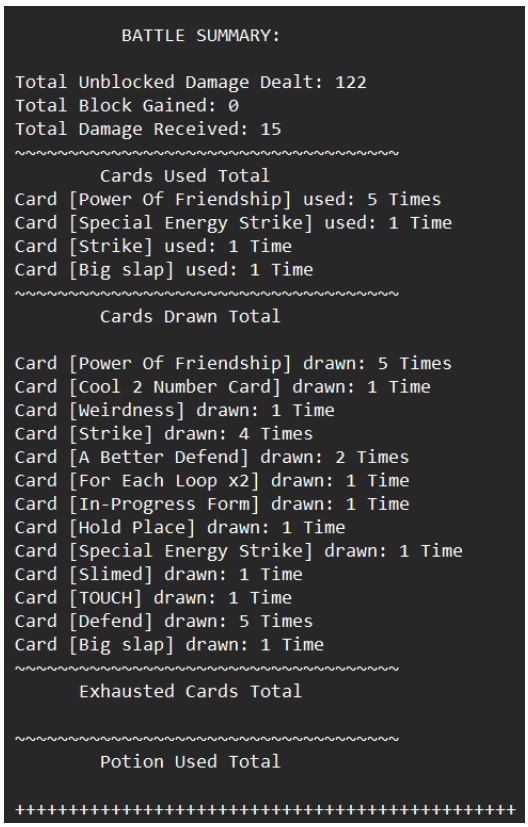


Fig 13. Summary at the end of battle.

5.3 The process

The development of the artifact mainly focuses on the actions that the users can add to a card. There are more than 350 cards currently in the original game and all of them are studied and their actions are extracted and added to the action list, however, there are some actions that are not added to the list because these are too complex while others are too specific to a character from the original game. There are {55} actions in the list, but each action also has extra options once added to the action table. Some actions are not present in the original game but exist in the API, some of these are also studied and added to the list to let the user create something unique that is not a part of the base game. Some custom-made actions are also added to the list because they serve as variations and counterparts to some other actions.

6. Result

This chapter presents the results of the functionality testing and the result of the user study. They will be analyzed and discussed to find answers to the research questions presented in Chapter **1.1.**

6.1 Functionality

The functionality testing was an iterative process from the beginning of the artifact development. As new actions are discovered and extracted from the original cards, they are added to the list. Each time an action is added to the list, it goes through a test to make sure it behaves the same way it would if added via the API. Tests were also done to make sure actions are compatible when they are combined in a single card. This was done by just adding them all to the same card and using it in the game. All available actions in the UI can be added to the same card and work as intended.

There was also a comparative test between the UI and the API. While the UI does not include every action available from the API, the actions that it has, are sorted into categories, showing all actions that are available making it easier for users to see what exists and what does not. While the UI only offers card creations, it is possible to also implement other creation features that the API has, like relics, events, et cetera. This will be further discussed in Chapter **7**. There are things that the API can do that the UI will not be able to do, for instance, creating a completely new action, as its logic must be implemented with codes. These things are core components of cards, with the UI, you can add components to the card, but you cannot create one.

Changes made to a card in the UI will not change the card in the game until the game is restarted. Which means the UI is not capable of real-time creation nor modification. It is however possible to modify a card's actions' value via a console while playing the game, using the keycodes. While the modified values will not replace the original's values, it can still be used to fine-tune the card in real-time and write down these values as notes.

6.2 User study

As mentioned, the user study is divided into two parts, the usability of the artifact and the iteration, with a total of five participants having varying hours in the game and one observer, which is the author of the thesis. The study was conducted in sessions, one participant at a time. The session was online, using Discord and the observer let the participant use the artifact via a remote control program. This is to save time and avoid program conflicts, since the observer already has the correct setup, the participant can jump right into it without the need to install any major programs. The participant was

made aware that they will be facing a boss from Act 1 but not which one, this is to avoid the more experienced players creating a bias card. All participants fought against the same boss, Slime Boss[37], which was chosen at random. They used the modded character 'The Thesis', having the same deck plus the unique card that they created. They did not need to defeat the boss, all they needed to do was to play around with their cards. From the sessions, it showed that even with minimum guidance, which is just some brief explanation of the artifact, it did not take longer than five minutes to create a card. Refer to Appendix D to see all participants' cards.

All five participants are familiar with game design, this is to ensure that they can provide valuable feedback and identify any potential design flaws or issues. Out of the five, two of them did not have a lot of experience with the game. They were chosen to test if the UI would be easy to use for beginners.

6.2.1 Participant 1

This participant has more than 30 hours in the game. They found the actions in the actions list to be easy to understand except for one action, "Gain Energy If Discard>0", which was a bit confusing to them. This action gives the player energy if at least one card has been discarded that turn. They found it to be very easy to add actions but not as easy when it comes to seeing which actions are added, because some actions' names are longer than the cell's size so they get truncated in the action table. Overall, they thought that it was very easy to create a card with the UI, except for the keycodes which were a bit hard to remember.

For the card iteration, they thought that it was very easy to edit a card, but the iteration was tedious. They did claim that it was acceptable but not being able to edit a card in real-time can waste a lot of time since you have to restart the game each time a card is edited to load in the newer version. This process can be tedious because there will be more than just one card being created and edited using the UI.

6.2.2 Participant 2

The second participant also has more than 30 hours. They found it to be very easy to understand the actions' names, and very easy to add actions as well as create a card. Like the last person, it was easy for them to see the chosen actions except for the ones that have a long name.

They found that editing the card is very easy and the iteration is satisfactory. They think that not being able to edit the whole card while in play is understandable but users should at least be able to edit the card's value, which can create a smoother workflow.

6.2.3 Participant 3

Participant number three has about 5 hours. Like participant one, they also got confused with the ‘Gain Energy If Discard>0’ action, to be more specific, they were unsure about its function: “Do you gain energy per card that has been discarded this turn or in the discard pile or choosing cards in your hand to discard to exchange for energy?”. The confusion surrounding this action shows that its intent is not conveyed properly via the name. But other than that, they thought that the UI was very easy to use.

For the iteration part, they had neither trouble with the editing nor the iteration process. They claim that it was straight forward and it went smoothly.

6.2.4 Participant 4

Participant number four only has about an hour of gameplay but they did manage to finish the first Act. They found that the UI is easy to use, and the actions are simpler than they thought since they did not have much experience, the actions were straightforward and did what the participant thought they would do.

Like other previous participants, participant four also thought that editing a card is very easy to do. The iteration, however, is tedious to them, they understood that tweaking something to make it feels right can take a lot of time but it was still a tedious process for them. They also thought that it would go smoother and less tedious if you were able to edit the card while playing the game.

6.2.5 Participant 5

Participant number five has more than 30 hours in the game. They thought that it was easy to understand the actions in the action list and it was also easy to add them to the action table. It was not easy for them to see all chosen actions and creating a card wasn’t as easy as they thought it would be. They suggested stretching out the cell for the actions’ names to give the names more space while shortening down the cells for the actions’ values since there is no reason for them to be as long as other cells. They also want more keycodes for other values than just damage values because of the ‘upgrade description’. They wanted to avoid having to copy-paste the original description into the upgrade description’s field to just change one number.

As for the iteration, same with previous participants, participant five thought editing a card is very easy to do but want to be able to edit the card in real-time, since resetting the game every time is annoying. They said that it was fun to be able to create their own card and use it in play.

7. Discussion

7.1 Evaluation of the Artifact

Using the result from Chapter 6.1, RQ.1&2 can be answered. The artifact covers the card creation part of the API, albeit incomplete because it does not include all actions. However, as there are already 50+ actions added, this shows that it is possible to add every action given more time. The API also offers relic creation, events, characters, etc. Theoretically, it is possible to integrate these into the UI as well, utilizing the already established card creation system as the creation behaves the same way in the API. While the UI does let the users create unique cards by combining different actions and options, it cannot be used to create more actions. If users want to create a unique action that is not present in the game, the API is a must. The same applies to the aforementioned content creation. In short, with the UI, you can create content using already-defined features, to create new unique features, you have to use the API.

You can create a lot more using the API but to do so, you need the required knowledge to use it, as for the UI, with minimum guidance, anyone can create a card in less than five minutes. While the API offers a lot more variety of actions, finding these actions can be tedious since there does not exist a list that defines all available actions. The UI on the other hand does have a list which makes it easier for users to know what action exists, which in turn, gives users more definitive options. Even though the UI is meant for non-programmers, it can still be a powerful tool for experienced modders. The reason is that they can use the UI to create a card simply by just inputting some values, and then expanding upon that card in the API without the need to create a new class and then importing libraries, creating new variables, and methods. It can also be used as a starting point for those who wish to start learning the API, as the UI follows the same creation structure of a card with the API.

7.2 Evaluation of Usability

Results from chapter 6.2 and the bata data collection part of the artifact can be used to answer the third research question "**How efficient is the UI when used as a design tool, regarding content creation and balancing?**". Starting with the user study, it shows that while it is easy to create and edit cards, some issues need to be fixed. Actions that have longer names will be truncated in the table since the cells for them are too short, this could easily be fixed by just extending the cell for the actions' names, as suggested by one of the participants. Some of the keycodes are confusing and hard to remember, causing one of the participants to have to constantly take a look at the guide. More keycodes are also needed to avoid having to write the same description twice with the

only change being the value of an action. It shows that while the UI does its job, being able to edit a card while playing the game will create a much smoother workflow for users, as valid as this point is, it is not possible to do so because you have to package the game each time you edit something for it to appear in the game. The test result could possibly have been more positive if the fact that the user can modify a card's values using keycodes was discovered sooner. As of the time of writing this thesis, only the original keycodes can be used with the console to alter cards' values.

The UI also collects battle data when players play the game(chapter 5.2). As it records almost every detail of the battle, it enables designers to have a closer look at what occurred during each turn and each battle, which can then help them determine the target of balancing. For instance, they can see if a card is always used or discarded. However, it focuses more on the player's side, since it doesn't take into account monsters' intentions and shielding. Why these were not taken into account, is because each monster has its own set of intents, putting them in would overflow the information inside the data file, thus making it hard for designers to read the file.

7.3 Summary

The actions list of the UI doesn't include every action available from the API, but they are sorted into categories, making it a lot easier for designers to see available actions, comparing this to the API, where it has more actions but they are not sorted in any way, making it difficult to not only find them but also see what is available. While the UI allows users to customize a card with different actions, it cannot be used to create actions. It is possible to add every action from the API to the UI as there are already more than 50 actions and even extra options to these actions. It is also theoretically possible to add other content creations, like relics, events, and characters, using the same system as the card creation.

Using the API, users can create a lot more unique things than the UI can because the UI can only implement predefined features, but the API is a lot harder to use than the UI because it requires programming knowledge. As simple and limited as the UI is, it can still be used to create unique cards. It is also possible to use it as a beginner's choice before moving to the API to create more content with more complicated logic. It can also be used as a complementarity tool to the API, creating a card with the UI and tweaking it further with the API if one wishes to do so. So not only does it benefit non-programmers, but it can also help experienced modders.

As for the artifact's efficiency, while it does its job, having to reset the game each time a designer wants to change something does not scale very well, since there is more than just one card that will need to be fine-tuned once created. It will also need more keycodes

so that users avoid having to use the ‘Upgrade Description’ since its purpose was to show drastic changes in a card once upgraded. Creating more keycodes will also allow actions’ value modification in real-time for more precise tuning. While having a few flaws, the UI is still easy to use and beginner friendly. It also collects details of the battle, how each turn/battle played out, what cards were used, discarded, or exhausted following other information that will be of use to designers when they try to maintain the game’s balance.

8. Conclusion

This chapter concludes the thesis, addressing the research questions and summarizing the answers to them as well as the proposed solution to the stated problem. Following with suggestions for future works.

RQ.1: What kind of features provided by the API can be integrated into the UI?

Content creations that act like a system can be added to the UI, things such as cards, relics, characters, events, keywords, etc. A card’s action, for instance, is also a type of system, but while you can add actions to the UI, you cannot create a new action using the UI. This is because the underlying logic of the action must be implemented using code.

RQ.2: What are the benefits as well as drawbacks of using the artifact compared to the usage of the API?

The artifact can be used without programming knowledge, but it is limited because it cannot be used to create things that are not defined in the API. The API on the other hand, allows users to create contents that are much more complicated but would require programming knowledge to do so. When it comes to card creation, it is also easier to see all available actions in the UI as all of them are presented to the users while being sorted into categories.

The API has a lot of actions that are not added to the UI, most of them were not added because of the time constraint, some are either too complex to be added or that they are too specific to a character. While the API has more actions, these are not presented to the users at all nor are they sorted. To find them, users must search for them, either by their name or via a card file, or go through hundreds of folders. With that said, while each has its own benefits and drawbacks, the UI can be used as an external tool to the API. For instance, the user can use the UI to create a card, generating most parts of the card, which helps the user save time. The user can then extend the card further by using the API.

RQ.3: How efficient is the UI when used as a design tool, regarding content creation and balancing?

Creating and editing a card with the artifact is simple. As a design tool, it does its job, however, not being able to edit a card while in play can cost a lot of time, since you will need to restart the game each time you want to use the newer version of the card. It is also lacking keycodes, forcing users to use a feature outside of its intended purpose. The artifact also collects battle information and writes it to a text file, letting users see what happened during each turn. By analyzing the text file, users will be able to see if anything needs to be changed to make it more balanced.

8.1 Future Works

In terms of future research, I suggest adding creations of other content, like relics, characters, events, and such before adding more actions to the card creation. This is because it will make the UI feel more complete as it allows users to create more content than just cards. This will make the UI more design-tool-like as you can use it to create a whole new character with their own set of cards, relics, and events that are unique to them. Then based on that, users will be able to have a clearer view when they design their own games. Taking in the feedback from the users, you can also add more keycodes, make it so that users can modify a card's value while in play for a smoother workflow, and make the action table's name column longer to cover the whole action's name so that it will not be truncated.

I also suggest extending the battle collection, collecting data such as enemies' actions each turn, a list of relics owned by the player at the start of battles, and more details on cards' values (as of now, it only collects how many times a card has been used; how many times a card was discarded and how many times a card was exhausted). It can also collect data outside of battle like the options of the paths the player can take, the game already collects the chosen path but not the other available options. An AI that plays the game, a simulation, so you don't have to go through hundreds of battles to find things that need to be fine-tuned.

9. References

[1] L. Gordon, ‘How one of gaming’s most intimidating genres spawned a legion of hits’, *The Verge*, Jun. 30, 2022. https://www.theverge.com/2022/6/30/23181999/roguelike-deckbuilder-genre-slay-the-spire-inscryption (accessed Jan. 26, 2023).

[2] ‘So what exactly is a deck-building game anyway?’, *Destructoid*, Oct. 29, 2014. https://www.destructoid.com/so-what-exactly-is-a-deck-building-game-anyway/ (accessed Apr. 04, 2023).

[3] ‘Magic: The Gathering | Official site for MTG news, sets, and events’, *MAGIC: THE GATHERING*. https://magic.wizards.com/en (accessed Feb. 22, 2023).

[4] B. Stegner, ‘What Are Roguelike and Roguelite Video Games?’, *MUO*, Oct. 24, 2021. https://www.makeuseof.com/what-are-roguelike-and-roguelite-video-games/ (accessed Apr. 04, 2023).

[5] ‘Mega Crit’. https://www.megacrit.com/press/ (accessed Jan. 26, 2023).

[6] ‘Training Neural Networks for a Slay the Spire recommendation system’.

[7] O. Porenius and N. Hansson, ‘Using machine learning to help find paths through the map in Slay the Spire’.

[8] M. Trojanowski and J. Andersson, ‘Are you lucky or skilled in Slay the Spire?’.

[9] ‘Slay the Spire Wiki’. https://slay-the-spire.fandom.com/wiki/Slay_the_Spire_Wiki (accessed Feb. 22, 2023).

[10] ‘Slay the Spire has sold more than 1.5 million copies’, *PCGamesN*, Mar. 19, 2019. https://www.pcgamesn.com/slay-the-spire/slay-the-spire-sales (accessed Feb. 22, 2023).

[11] ‘*Slay the Spire*’, *Wikipedia*. Feb. 18, 2023. Accessed: Feb. 22, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Slay_the_Spire&oldid=1140163491

[12] ‘Games Modding – Why you should do it’. https://intogames.org/news/how-to-mod (accessed Jan. 30, 2023).

[13] W. Scacchi, ‘Modding as a basis for developing game systems’, in *Proceedings of the 1st International Workshop on Games and Software Engineering*, in GAS ’11. New York, NY, USA: Association for Computing Machinery, Maj 2011, pp. 5–8. doi: 10.1145/1984674.1984677.

[14] ‘*Counter-Strike* (video game)’, *Wikipedia*. Feb. 02, 2023. Accessed: Feb. 22, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Counter-Strike_(video_game)&oldid=1137123109

[15] ‘*Defense of the Ancients*’, *Wikipedia*. Sep. 08, 2022. Accessed: Feb. 22, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Defense_of_the_Ancients&oldid=1109109905

[16] ‘*Dota 2*’, *Wikipedia*. Feb. 13, 2023. Accessed: Feb. 22, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Dota_2&oldid=1139057901

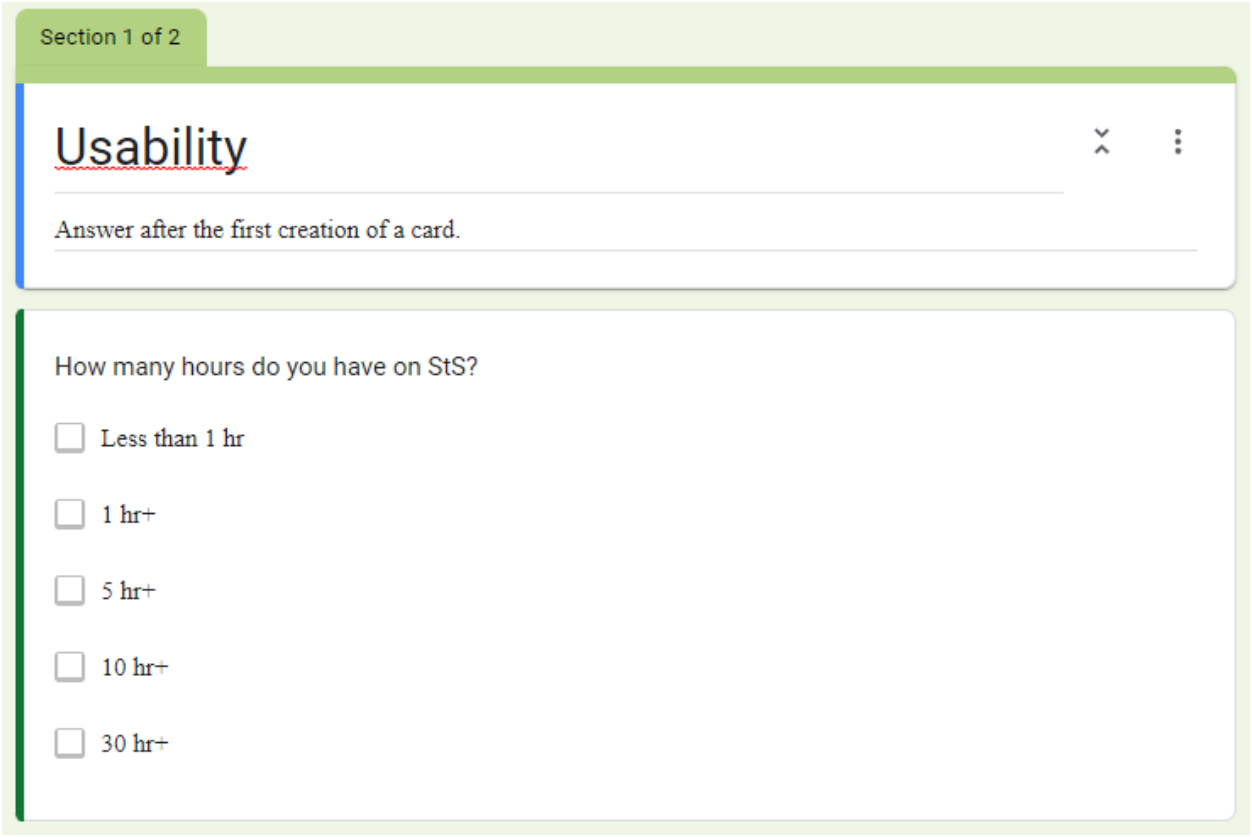
[17] ‘*League of Legends*’, *Wikipedia*. Feb. 16, 2023. Accessed: Feb. 22, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=League_of_Legends&oldid=1139792092

[18] ‘Most Popular PC Games - Global’, *Newzoo*. https://newzoo.com/resources/rankings/top-20-pc-games (accessed May 30, 2023).

[19] ‘Mod Hall Of Fame’. https://modhof.com/ (accessed Feb. 22, 2023).
[20] C. Camargo, ‘Modding: Changing the Game, Changing the Industry’,.
[21] ‘Exploring the Video Game as a Learning Tool’. https://www.ercim.eu/publication/Ercim_News/enw57/emmerson.html (accessed Feb. 20, 2023).
[22] ‘9 Types of Designer Jobs for Creative People | Indeed.com’, *Indeed Career Guide*. https://www.indeed.com/career-advice/finding-a-job/types-of-designer-jobs (accessed May 03, 2023).
[23] R. Olivares-Amaya, R. Salomón-Ferrer, W. A. Lester, and C. Amador-Bedolla, ‘Creating a GUI for Zori, a Quantum Monte Carlo Program’, *Comput. Sci. Eng.*, vol. 11, no. 1, pp. 41–47, Jan. 2009, doi: 10.1109/MCSE.2009.5.
[24] M. Lundstrom and G. Klimeck, ‘The NCN: Science, Simulation, and Cyber Services’, in *2006 IEEE Conference on Emerging Technologies - Nanoelectronics*, Jan. 2006, pp. 496–500. doi: 10.1109/NANOEL.2006.1609779.
[25] A. Aspuru–Guzik *et al.*, ‘Zori 1.0: A parallel quantum Monte Carlo electronic structure package’, *J. Comput. Chem.*, vol. 26, no. 8, pp. 856–862, 2005, doi: 10.1002/jcc.20215.
[26] J. M. Font and T. Mahlmann, ‘Dota 2 Bot Competition’, *IEEE Trans. Games*, vol. 11, no. 3, pp. 285–289, Sep. 2019, doi: 10.1109/TG.2018.2834566.
[27] V. Volz, G. Rudolph, and B. Naujoks, ‘Demonstrating the Feasibility of Automatic Game Balancing’, in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, Denver Colorado USA: ACM, Jul. 2016, pp. 269–276. doi: 10.1145/2908812.2908913.
[28] F. de Mesentier Silva, R. Canaan, S. Lee, M. C. Fontaine, J. Togelius, and A. K. Hoover, ‘Evolving the Hearthstone Meta’, in *2019 IEEE Conference on Games (CoG)*, Aug. 2019, pp. 1–8. doi: 10.1109/CIG.2019.8847966.
[29] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, ‘A Design Science Research Methodology for Information Systems Research’, *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007, doi: 10.2753/MIS0742-1222240302.
[30] ‘Java Swing Tutorial - javatpoint’. https://www.javatpoint.com/java-swing (accessed Apr. 11, 2023).
[31] ‘GitHub - Gremious/StS-DefaultModBase’. https://github.com/Gremious/StS-DefaultModBase/ (accessed Apr. 11, 2023).
[32] D. Cook, ‘BaseMod’. Apr. 10, 2023. Accessed: Apr. 11, 2023. [Online]. Available: https://github.com/daviscook477/BaseMod
[33] ‘GitHub - kiooeht/StSLib: A collection of keywords and mechanics for other Slay the Spire mods to use.’ https://github.com/kiooeht/StSLib (accessed Apr. 11, 2023).
[34] ‘GitHub - kiooeht/ModTheSpire: External mod loader for Slay The Spire’. https://github.com/kiooeht/ModTheSpire (accessed Apr. 11, 2023).
[35] aswx12, ‘StS-TheCreator’. Apr. 11, 2023. Accessed: May 13, 2023. [Online]. Available: https://github.com/aswx12/MTS
[36] ‘Game Designer Job Description, Salary, Skills & Software’. https://www.cgspectrum.com/career-pathways/game-designer (accessed Mar. 21, 2023).
[37] ‘Slime Boss | Slay the Spire Wiki | Fandom’. https://slay-the-spire.fandom.com/wiki/Slime_Boss (accessed May 10, 2023).

Appendix A

Participant questionnaire after they created their first card



How easy was it to see all chosen actions? *

	1	2	3	4	
Very hard	○	○	○	○	Very easy

Finally, How easy was it to create a card? *

	1	2	3	4	
Very hard	○	○	○	○	Very easy

Feedback, extra comment

Long answer text

How easy was it to understand the actions in the action list? *

	1	2	3	4	
Very hard	○	○	○	○	Very easy

How easy was it to add actions to the action list? *

	1	2	3	4	
Very hard	○	○	○	○	Very easy

Appendix B

Participant questionnaire after card iteration

Card iteration

Answer this after iterating the card creation process

How easy was it to edit a card? *

	1	2	3	4	
Very hard	○	○	○	○	Very easy

How did the iteration feel? *

	1	2	3	4	
Very tedious	○	○	○	○	Very smooth

Feedback, extra comment

Long answer text

Appendix C

Testing procedure

1. Start the artifact
2. A brief explanation of how to use the artifact
3. Freeform test, users can play around, create a quick card then go to the game to see how it works. Give them link to the guide https://github.com/aswx12/MTS (Appendix D)
4. When they are done with creating their card, they are to fill in the first part of the questionnaire.
5. When they are finished, they can start creating their own card and using them in the game. They will be fighting the slime boss
6. After a few turns, they are encouraged to go back and forth to make at least one change to their card.
7. Repeat step 6 for about 15 minutes, or if the boss is beaten or if the tester is satisfied with their card.
8. After step 7, they are to fill in the second part of the questionnaire.

Appendix D

Guide

Guide:

Before using the UI, it is recommended that you should have some kind of program that packages the mod after you have added a card to it. You should also subcribe to **Basemode**, **StSLib** and **ModTheSPire** from steam. In the pom.xml, change the steam path to your steam path. I recommend Intellj with maven.

1. Name the card, choose a location to save the card, for it to work, the card must be within the "Cards" folder. (Should already be at location once the create card button is pressed) Using the name of an existing card will load that card's content(edit mode)
2. Card type: choose the type of the card.
3. Action list, browse categories for actions
4. Once actions are added to the action table, fill out the action value and choose options presented in the table as you see fit.(select actions in table and press remove if want to remove)
5. Want to add/remove action on upgrade? Tick the box otherwise skip to step 7
6. Select from the action list, fill out value as usual, choose to whether add or remove
7. Fill out cost, choose card's rarity and target.
8. Write a description for the card, refer to the keycodes section for dynamic numbers.
9. Want a different description upon upgrade? select the box and repeat step 8
10. Card states: default is when the card is not upgraded. If the box for each item is selected, that state will appear on the version of the card that was selected.
11. Seen is just for the card to be unlocked in the library so you can see it.
12. Create card! wooo
13. Press new card for new card, obviously.

Keycodes

Damage: **!D!**
Vampire damage: **!DVAMP!**
Damage If Target Poisoned: **!DTP!**

(base damage)
Damage Per Energy Used: **!DPE!**
Damage Per Attack Played: **!DPAP!**
Damage Per Skill In Hand: **!DPSH!**

(these keycodes calculate the total damage for you)
Damage Per Energy Used: **!XDPE!**
Damage Per Attack Played: **!XDPAP!**
Damage Per Skill In Hand: **!XDPSH!**

Block: **!B!**

Energy: **!M!**
Energy icon: **[E]**

Want to color code you text?
v text you want to color.
[#0000ff]text[]<-closing
^color code

Appendix E

Created Cards

Participant 1:

Actions chosen:

1. Vampire damage
2. Gain Energy
3. State: Ethereal / changes to Exhaust on Upgrade

Participant 2:

Actions chosen:

1. Damage
2. Apply Poison
3. Repeat

Participant 3:

Actions chosen:

1. Damage (trigger if enemy's intent is attack)
2. Block (trigger if enemy's intent is attack)
3. State: Retain

Participant 4:

Actions chosen:

1. Damage per energy
2. Damage per attack played
3. Damage per skill in hand

4. Sacrifice HP
5. Gain Strength
6. State: Retain

Participant 5:

Actions chosen:

1. Apply Vulnerable
2. Apply Weak
3. Apply Poison
4. State: Exhaust
5. Double Vulnerable (on upgrade)
6. Double Weak (on upgrade)
7. Double Poison (on upgrade)