Aalborg University Copenhagen
Frederikskaj 12,
DK-2450 Copenhagen SV
Semester Coordinator: Stefania Serafin
Secretary: Christine Pedersen

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Applied Game Balancing Techniques: Development of Scriptable Object Table in Unity

Theme:
Master's Thesis

Project Period:
Spring Semester 2023

Project Group:
Enlit Games 1

Participant(s):
Kristinn Bragi Garðarsson
Simonas Čeponis

Supervisor(s):
Olga Timčenko

Page Numbers: 29

Date of Completion:
May 25, 2023

Abstract:

This study presents an in-depth investigation into game balancing practices among game designers, with a particular focus on understanding the challenges faced and strategies employed in the balancing process. The study evaluates our tool Scriptable Object Table as a potential aid for designers when balancing their game. As a step for developing the tool, we made the game *Web of Lies* to understand the needs for balancing games. Through a series of interviews, the research reveals insightful feedback regarding its scalability, integration, extendibility, user experience, and functionality. The study's findings underline the potential value of implementing features that support designers by facilitating an enhanced iteration process. We found that users put emphasis on control of how they view the data, changing data quickly, tracking the changes, and showing effect of the changes. Despite certain limitations, the study provides a valuable foundation for future research and tool development in the domain of game balancing.

Applied Game Balancing Techniques: Development of Scriptable Object Table in Unity

Kristinn Bragi Garðarsson
Aalborg University Copenhagen
kgarda18@student.aau.dk

Simonas Čeponis
Aalborg University Copenhagen
scepon18@student.aau.dk

***Abstract*—This study presents an in-depth investigation into game balancing practices among game designers, with a particular focus on understanding the challenges faced and strategies employed in the balancing process. The study evaluates our tool Scriptable Object Table as a potential aid for designers when balancing their game. As a step for developing the tool, we made the game *Web of Lies* to understand the needs for balancing games. Through a series of interviews, the research reveals insightful feedback regarding its scalability, integration, extendibility, user experience, and functionality. The study's findings underline the potential value of implementing features that support designers by facilitating an enhanced iteration process. We found that users put emphasis on control of how they view the data, changing data quickly, tracking the changes, and showing effect of the changes. Despite certain limitations, the study provides a valuable foundation for future research and tool development in the domain of game balancing.**

***Index Terms*—Game balance, game balancing, Unity, editor tool, Scriptable Object**

I. Introduction

Mastering the art of balancing can set a game apart from the highly saturated collection of games in the industry. Game balancing is the act of modifying the game to achieve desired design goals. Well-balanced games are fair to the player, leading to less frustrations and more enjoyable experiences (Adams, 2014, pp. 324-358). This implies that balancing is a strong factor in fun in games, which is one of the most important milestones game designers must meet (Fullerton, 2014; Schreiber, 2010). However, balancing a game is a long and resource-intensive process. Game designers use a variety of heuristic and mathematical approaches to theory-based game balancing which will be analysed and discussed Schreiber and Romero (2021).

In collaboration with Get Media Savvy, a US-based collective advocating for media literacy, we worked on the game *Web of Lies* a roguelite deckbuilder card game designed to provide players with an engaging gaming experience imparting essential media literacy skills, which have become increasingly important with rise to the internet and novel information technology. Rapid proliferation of fake news and the spread of misinformation has become a significant global concern (Del Vicario et al., 2016; Kubey, 1997; Mian and Khan, 2020), and many efforts to promote media literacy have been underway (Dave et al., 2022; Wardle et al., 2017). As video games are a popular and pervasive medium, they have emerged as a promising tool for teaching the subject. Our contribution to the project – that aligns with this study – was to make the game fun, ultimately maximizing its impact by ensuring that players are effectively challenged and engaged in the learning process (Schreiber, 2010).

Initially, we set out to investigate balancing in roguelite deckbuilders. However, due to scoping challenges and the limited applicability of focusing on a single finished game, we instead decided to create a generalized tool called Scriptable Object Table (SOT) for balancing games in Unity (Unity Technologies, 2023a), regardless of genre. This approach is more scalable, offers broader applicability to the community, and has greater potential to contribute to the advancement of game balancing techniques within the industry. To guide the design of the tool, it was built based on the balancing concepts and common practices described in this study.

To that end, we put forth the research question:

> *How can we develop a game balancing tool that streamlines the workflow for developers, allowing them to easily iterate and incorporate balancing principles in their projects?*

This paper will examine the principles of game balance in roguelite deckbuilder card games and describe the design and development of a tool we developed that helps with the balancing process of similar games.

II. Game Balance Theory

There are many different definitions of what game balance is (Becker and Görlich, 2020). In this paper, we treat game balancing as a way to modify game values to achieve the desired design goals of the game. To achieve game balance, a variety of different approaches can be applied which have different requirements and development costs (Schreiber and Romero, 2021).

Before diving deep into the concepts, it is important to clarify goals of balancing. Schreiber and Romero (2021) raise a point that *fairness*, *interesting player choices*, and *difficulty* are potential targets for balancing. Sylvester (2013) argues that it is also important that the game balance reflects the narrative and is accessible to the player from a user experience perspective. When possible, balance should be achieved by adjusting values without taking away from other aspects of the game.

Fullerton (2014) dives deeper into balancing the difficulty of the game. Difficulty can be balanced for in multiple ways. Roguelite games open up the possibility of dynamic balancing, because the game naturally becomes easier every playthrough due to increasing player skill and meta-progression.

Tyroller (2019) states that the designer should decide what the balancing goals should be for the game as a whole and the separate game objects.

A. General Concepts of Game Balance

It is important to clarify game balance terms before diving deep into the different methods for balancing a game. Doing so, we facilitate a shared understanding and provide a foundation of the basic concepts for a more effective discussion on the balancing approaches. Furthermore, it enhances our comprehension of the requirements and challenges faced by designers that focus on balancing.

a) Systems and Rules: Schell (2008) writes that the rules of the game *make* the game.

Sylvester (2013) defines a game design as a system of rules that drive the behavior of the game. Therefore the job of a game designer is to create these systems of rules in a way that achieves the design goals of the game.

Schreiber and Romero (2021) define systems as described in the standard dictionary definition: *"a set of interacting or interdependent parts that form a complex whole."*

In this paper, we will use systems and rules to describe how the game works and how the different elements of the game interact with each other.

As games consist of multiple systems working together, a game can be balanced by changing how the various game systems interact with each other.

Fullerton (2014) proposes that splitting systems into smaller subsystems is worthwhile to make the game easier to balance as a whole. It is also important to try to reduce the connections between different systems as a change to a system can change the balance of all other connected systems.

b) Resources: Schreiber and Romero (2021) define resources as values that can be exchanged for different values. The flow of resources, their availability and versatility in comparison to other resources is what defines their relative value. Examples of resources include time, currency, lives, game objects, experience points and so on.

c) Anchors: An important step in balancing a game with many variables is the *anchor*. Schreiber and Romero (2021) define the anchor as an in-game resource that is directly related to every other resource in the game. This direct relationship allows the anchor to be used as a way to assess the value of each resource in the game, whether it is health points, time, score or any other resource that exists in the game. Furthermore, a *comparative anchor* is a unit that can be used as a default unit that other variations of units can be compared against to ensure that all units are balanced with each other.

d) Curves: According to Schreiber and Romero (2021) curves (functions) show the relationship between different game resources. The shape of the curve defines the progression of the game by showing how the increase in one resource affects other resources. For example, a curve could be an attack vs coin curve showing how much attack can be done using a given amount of coins.

Fullerton (2014) recommends that designers keep such data in spreadsheets and develop them together with the technical part of the team. Spreadsheets are stressed as both a good starting point and a way to fine-tune the game balance in the latter stages of development.

It is worth mentioning that while curves are used to keep relationships between resources balanced, it is common for designers to edit values of specific game objects for the sake of keeping the game interesting or for narrative purposes, even though those changes might work against achieving a perfect game balance. Millard (2022) argues that imbalance if used well can make a game better. He argues that single-player games specifically can use this to make the player feel powerful. Imbalance can also be used to encourage player to play in a way that is more fun and discourage boring strategies. Credits (2013) states that imbalance should be carefully considered and designed in order to be effective.

It is also important to consider the extremes of the curves and have extra logic to handle what happens when these extreme values are reached.

e) Granularity: Granularity refers to the proximity of one number to another (Schreiber and Romero, 2021). When numbers are too high, they become meaningless to the player or make it harder to make calculations and reason about their value. If the numbers are too small, they limit the designer's choices when it comes to balancing the game. General game design and balancing decisions can be done to address these issues by adjusting the granularity of the numbers present in the game.

f) Possibility Space: Possibility space defines how many different states of the game can be reached by the player. It can be expanded by adding more complexity to the game. In modern games, the possibility space is so vast that it mainly used as a theoretical term to make conversation about the topic of game balance easier. It is important to understand that a bigger possibility space allows for greater replayability (Schreiber and Romero, 2021). Possibility space can be expanded in different ways, of which not all could be considered desirable for the design of a given game.

g) Determinism: According to Schreiber and Romero (2021), determinism is used to define the different actions the player can take in the game. A deterministic action will always have the same outcome if the state of the game is the same. If an action is non-deterministic, the same action can have different outcomes even if the game state is identical every time. The presence or lack of determinism in the actions defines how the game can be balanced. Game mechanics can be made to be non-deterministic to expand the possibility space at the cost of player control.

h) Randomness: Engelstein (2018) defines the terms *input randomness* and *output randomness*. Input randomness is defined as randomness that modifies game state prior to

player action. For instance, in *Settlers of Catan* (Klaus Teuber, a), the player throws a dice dictating what resources they have to work with before they decide their action that turn. Output randomness happens after player action, such as critical hit systems in turn-based games where there is a certain chance to deal double damage after the player picks their move. Generally, input randomness is said to be more desirable because it supports strategy. However, there are cases where output randomness can be useful to encourage risk assessment. Additionally, output randomness can be used to reflect the random nature of actions happening outside of player control, such as units following player orders being successful or not.

Randomness is particularly important for roguelite games, as they are defined by random content generation during runtime of the game (Brown, 2020). Random generation adds variety in the gameplay and thus extends the replayability of the game. Randomness also forces the player to learn how the game mechanics work instead of relying on memorization of optimal plays. Randomness can also be employed to turn the game into a more casual experience that is less reliant on player skill. Randomness also adds to the impact of rewards given to the player. Therefore, randomness was an essential consideration for balancing our game *Web of Lies*.

i) Probabilities: Probabilities naturally follow when randomness is designed in games. For games with outcome randomness, developers can tweak the chances to make the game more balanced (Schreiber, 2010). There are dependent and independent probabilities. Dependent probabilities change based on the game state, while independent probabilities are completely random. Human psychology dictates that we are naturally really bad at statistics (Schreiber, 2010), as we often develop superstitions with probabilities, such as when we are on a hot streak, we believe that some external factors are influencing the streak. As game designers, we can use this by adapting the probabilities to make it more likely to get a good roll, making the player feel more engaged. This can therefore help with balancing the game by reducing moments of disengagement.

Exposing probabilities can also give players more autonomy over their decisions, making their risk assessments fairer. On the contrary, developers can implement hidden dependent probabilities, such as in *Slay the Spire* (Mega Crit Games), where the probability of rare cards appearing increases every time the player has not seen one when collecting rewards. This can be useful for avoiding runs (play-throughs) where they never get any good cards, but it should be advised to not abuse this, as the player might feel the game is unfair if they start noticing this. Making a game *"feel"* more random can also cater to a more casual player base (Schreiber, 2010). *Catan: Junior* (Klaus Teuber, b) removes one die from the die roll, normalizing the probability curve, making it so that each number has an equal chance of being thrown. This is more suitable for kids, since it reduces the cognitive demand on the player as they do not have to recognize some numbers appearing more frequently such as in the original game with two dice.

j) Dominant Strategies: When the player is presented with choices where a single move is always obviously the most optimal choice, the game's balance is tainted by a *dominant strategy*. Generally, this should be avoided at all costs, as it reduces fun in the game (Adams, 2014). The player should be able to make meaningful choices that fit their playing style. Furthermore, having multiple viable strategies improves the game's replayability, engagement, and strategic depth.

k) Transitivity: Game mechanics can be described as transitive based on the relationships between game elements. A transitive relationship involves the direct comparison between game elements, such that if A is stronger than B, and B is stronger than C, A must be stronger than C. This can lead to a dominant strategy, as players will always prefer the strongest option. Therefore, applying a higher cost appropriately to the stronger option is essential to balance transitive mechanics and avoid dominant strategies (Adams, 2014). Another alternative is to establish intransitive relationships. In this case, strength of a given game element is contextual, meaning that a given object is stronger than another object in a given situation, while being weaker than it in a different situation. The most common example of an intransitive game is *Rock, Paper, Scissors*. Changing the type of relationship between different elements does not change the possibility space, but its does have an effect on how interesting the choices between them are. Intransitive relationships make the choices more interesting by challenging the player to think more (Schreiber and Romero, 2021). They can also be used to avoid dominant strategies, since the player must adapt to the different situations (Adams, 2014). Understanding of the difference between transitive and intransitive relationships is crucial in the decision-making when it comes to balancing game mechanics.

l) Information: Information on the state of the game can be given to or hidden from the player to change the possibility space of the game. Furthermore, the player can be given the choice to uncover more information on the state of the game for a cost. This allows for more interesting choices for the player in which they have to balance knowledge with resources (Schreiber and Romero, 2021).

According to Brown (2020), the amount of information directly affects the player's ability to plan ahead. He warns that too much information can lead the player to paralysis of analysis. Leaving some information hidden is a good way to force the player to adapt and make interesting choices throughout the gameplay time.

m) Solvability: According to Schreiber and Romero (2021), a game is solvable if it is possible to find the optimal action for the player to take at any given moment in the game. Furthermore, solvability is split into:

- Trivial solvability - It is easy for the human mind to find the optimal action.
- Theoretical solvability - the player has all the information required to find the optimal solution, but its impossible due to time constraints or the size of the possibility space.
- Computational solvability - It is only possible to find the optimal action to take by simulating the game over and

over until the optimal strategy is found.

When games are only theoretically solvable, the designer cannot achieve perfect game balance. In these cases, the designer can resort to educated guesses combined with playtesting and metrics to find the right balance.

n) Feedback Loops: According to Schreiber and Romero (2021) feedback loops are game mechanics in which an action is modified by it being used. Positive/amplifying feedback loops add to the power of the action every time it is used. Negative/dampening feedback loops reduce the power of the action every time it is used. Feedback loops can be used to control the progression of the game. While in most cases positive feedback loops are avoided to keep the game in balance, sometimes they are employed to quickly finish the game after it is clear that victory is inevitable for a player. Negative feedback loops are used to stabilize by ensuring that actions become less powerful when being used repeatedly. This keeps the game interesting as the player has to reconsider their choices every time they take an action. Fullerton (2014) calls such mechanics reinforcing relationships. The task in balancing feedback loops is to keep the game balanced without making it stagnate.

B. The Four Approaches to Game Balancing

In the following sections, we will dive deeper into game balance using intuition, mathematical modelling, playtesting, and data.

1) Intuition Approach: The initial game balance is set by the designer based on a rough estimate for what will be a good value. While these values are very likely to be changed later, they can sometimes be enough to get a general feel of the game down before moving on to other methods of balancing. Designer experience dictates how balanced the game will become using this method. Fullerton (2014) writes that balancing a game is as much about gut instinct as it is about mathematics. It is a process that is practiced and improved on over time.

There are cases where manipulating values or relationships between resources is not the optimal solution. This is where the intuition based approach shines. A designer can add or completely remove features and rules present in the game to balance the game as a whole.

a) Estimation Techniques: Felder (2015) proposes several techniques for estimating different game values and costs. The Fermi Solution (Von Baeyer, 2001) suggests that multiple estimations can arrive at a more accurate average estimation than a single experts guess. Felder (2015) suggests that this technique can also be applied to estimation of important game design values. Furthermore, estimation of multiple interconnected values should arrive at a roughly balanced final resulting value.

2) Mathematical Approach: Since every game has numbers, using mathematics to balance a game is a natural fit. There are many tools for analysing and then balancing games using mathematics. Sometimes, the game can be balanced if all elements of the game can be attributed with a value through cost curves and probability analysis. Additionally, having found the curves that define the game balance allows the designers to add content to the game in the future without disrupting the game balance too much or requiring extensive playtesting. Most of the time, mathematical balancing cannot be used without additional supporting techniques as occasionally there are game elements that cannot realistically be put into formulas. Tyroller (2019) suggests not to put too much development effort into mathematical balancing especially in cases where complex game mechanics are prevalent. Furthermore, mathematical analysis cannot take into account the psychological part of what makes games feel balanced. Nonetheless, the mathematical approach can be used to get the game ready for playtesting (Schreiber and Romero, 2021).

3) Playtesting Approach: Intuition and mathematics can only get the game balance so far. As games are experienced subjectively, playtesting is essential to capture how balanced the game feels for its target audience. Schell (2008) defines playtesting as the act of getting players to play the game and see if the design of the game facilitates the expected experience for the player. Brown (2019) defines playtesting and player data analysis as essential steps in verifying the balance of the game. Furthermore, complex games that offer an abundance of actions or abilities run a higher risk of potential exploits. For example, when fighting Father Gascoigne in *Bloodborne* (FromSoftware) the player can exploit positioning of the boss by moving below a staircase where the boss is unable to hit them, making a game that has a reputation for being extremely challenging a breeze. These exploits are mostly found through rigorous playtesting (Adams, 2014), however, AI automation is a promising cheaper alternative (Zarembo, 2019). It is worth noting, that playtesting, while essential, can be a rather expensive process. Sylvester (2013) writes *"Real understanding of a game's balance can never come from watching one or two tests, much playing the game yourself. It comes from absorbing many different players' experiences and combining them into an integrated mental model of how the game is working."* It is therefore important to maximise the benefit from testing on users by balancing the game as much as possible using intuition and mathematical modelling prior to extensive playtesting.

According to Schell (2008), preparing questions ahead of time is important to get the most value out of the playtest. In order to improve the balance of the game, questions should be asked addressing balance concerns based on what the balancing effort is focusing on.

4) Metrics Approach: Once the game is live, player actions can be tracked using analytics solutions to assess player behavior and game balance. Data can show which items are not being used, which levels are too difficult, which classes are underpowered and so on. Other types of design issues are still found best through playtesting, but when it comes to adjusting numbers, statistical data provides the most accurate answer.

Of course, physical games are not well suited for this approach as they would require the players to self-report their game experiences, which is rare outside of competitive

games like chess. Furthermore, the metrics approach has costs associated with setup and maintenance of the analytics system used for gathering and storing data (Schreiber and Romero, 2021).

III. Web of Lies

In collaboration with Get Media Savvy and Perednytė et al. (2023), we co-developed *Web of Lies*, which we used as a foundational project for further developing of Scriptable Object Table (SOT). Having a real-world project gave valuable context and exposed potential requirements that we may have overlooked had we developed the tool devoid of the real-world setting.

Web of Lies is a roguelite deckbuilder game set in a medieval fantasy setting. The player takes the role of a spy that was sent by the king to discredit the rebellion. The player does so by spreading disinformation among the population. Disinformation is spread through cards that can be played on a grid of pawns. An outline of the planned design of the full game can be found in Appendix C.

Fig. 1. Web of Lies.

We define roguelite games as games that are structured in the form of multiple runs and have a meta-progression element that persists between those runs. The genre borrows elements from the roguelike genre, but tries to achieve a less punishing player experience. A single run is a player journey from start to either completing the game or failing to do so. Each run makes the following run easier through the development of player skill and meta-progression. The player is expected to play many runs until they are capable of winning the game.

Our experience developing *Web of Lies* together with the analysis of game balance theory showed us that there is a need for tools to make balancing games faster and easier to do.

IV. State of the Art

Researching game balance and developing *Web of Lies* gave us a better understanding on how this process of balancing is done. It is clear now, that regardless of the approach taken, a common occurrence when balancing games is the input and modification of a large amount of values that relate to different objects in the game. In Unity (Unity Technologies, 2023a), such values are often stored in Scriptable Objects (Unity Technologies, 2018).

Traditionally, game designers rely on sheets to input relationships and tweak numbers to balance their games. An issue with this approach is the lack of integration with the development environment, leading to increased iteration times. In this section, we will investigate various solutions that make help the user to work with Scriptable Objects in Unity.

A. Sheet Codes

Sheet Codes (Final, 2022) is a tool that compiles values in a single table view. This allows for organization and swift editing of the different values in a way that creates a good overview of a big amount of game objects.

It does have the limitation of requiring the user to create Scriptable Objects from scratch using the tool. This might not be suitable for projects that have started development before the tool is employed.

B. Scriptable Object Editor

Scriptable Object Editor (Agent40, 2022) adds a new editor window which focuses on Scriptable Objects. It makes it easier to find the different Scriptable Objects and edit them in a similar fashion to the inspector window. A nice feature it adds is to create new Scriptable Object instances from the same window.

However, this tool does not provide an overview of the various Scriptable Object instances in a single view, which makes it hard to compare them. Comparing various objects is a core part of balancing a game.

C. Bona Data Editor

Bona Data Editor (Fyrvall, 2022) is a tool that makes editing different Scriptable Object types at the same time possible. It needs a programmer to tag variables in the code with a special attribute, which clutters the code and requires extra work from the programmer. It also lacks the feature of showing the different values in a single table for better overview and comparison of different objects.

D. CSV Serialize

CSV Serialize (VisualWorks, 2019) is a tool which moves data stored in the CSV format to Scriptable Objects. This is useful when the designer prefers adjusting values in tools like Microsoft Excel (Microsoft Corporation, 2023) or Google Sheets (Google LLC, 2023). It does have the limitation of not exporting values back to the CSV format after they were edited in the Unity editor.

V. Scriptable Object Table

We propose Scriptable Object Table (SOT), a plugin for Unity designed and created by us to ease the balancing effort for game developers. It adds an editor window to Unity which allows the designers to edit all instances of a given Scriptable Object in a single window. This allows for fast and easy comparison as well as editing of different Scriptable Object instances that already exist in the project.

A. Design

The Scriptable Object Table allows the developer to select the Scriptable Object they want to edit with a built-in selection window (Figure 2).

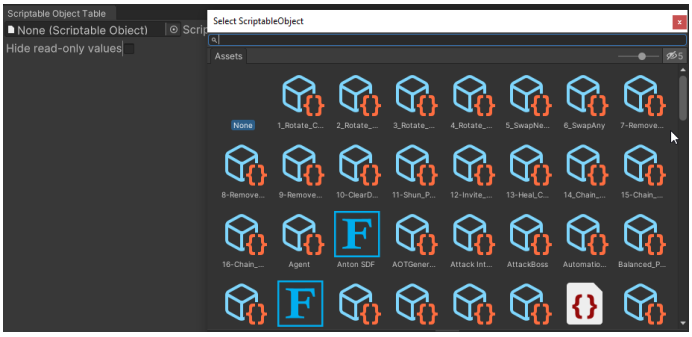


Fig. 2. Selection of the Scriptable Object.

Once the Scriptable Object is selected, all instances of the same type are shown in the table (Figure 3).

File Path	cardName	manaCost	cardMark
Assets/Data/Cards/1_Rotate_Clockwise.asset	Turn Right	1	mark-rotate
Assets/Data/Cards/2_Rotate_CounterClockwise.asset	Turn Left	1	mark-rotate
Assets/Data/Cards/3_Rotate_180.asset	Turn Around	2	mark-rotate
Assets/Data/Cards/4_Rotate_Neighbors.asset	Redirect Trust	3	mark-rotate
Assets/Data/Cards/5_SwapNeighbour.asset	Rumor Swap	2	mark-swap
Assets/Data/Cards/6_SwapAny.asset	Reorganize	3	mark-swap
Assets/Data/Cards/7-Remove_Block.asset	False Context	1	mark-removeSuspicion
Assets/Data/Cards/8-Remove_Thorns.asset	Satire	2	mark-removeLiteracy2
Assets/Data/Cards/9-Remove_Both.asset	lower Guard	1	mark-removeBuff
Assets/Data/Cards/10-ClearDownedStatus.asset	Calm Down	2	mark-heal
Assets/Data/Cards/11-Shun_Pawn.asset	Ostracize	2	mark-shun
Assets/Data/Cards/12-Invite_Pawn.asset	Recruit	3	mark-invite
Assets/Data/Cards/13-Heal_Chain.asset	The Truth	3	mark-heal
Assets/Data/Cards/14_Chain_Attack.asset	Propaganda	3	mark-attack
Assets/Data/Cards/15-Chain_Thorns_Attack.asset	Tall Tale	4	mark-strongAttack
Assets/Data/Cards/16-Chain_Attack_targeted.asset	Take Down	3	mark-targetedAttack

Fig. 3. Scriptable Object instances listed in a table.

Where possible, values are made editable using different input fields depending on the type of variable. There are also cases where the values are not editable. Then the value is rendered as a label instead of an editable field. Sometimes it is by design of the underlying Scriptable Object, where the developer coding the Scriptable Object wanted to limit editing in the editor. There are also cases where displaying the value in a single cell would compromise the table layout. Therefore, nested classes, arrays and lists were not made editable in the SOT.

In some cases, these read-only values comprise a big part of the Scriptable Object (Figure 4). To remove the clutter, we added a toggle which enables hiding such values. We also implemented a feature, where the user can click on the read-only value to be taken to the Scriptable Object in the inspector view, which in turn might allow them to edit the value.

There might also be values that cannot be displayed at all due to unforeseen reasons. To address such cases, we have added a warning that appears whenever such errors are encountered to provide the user with feedback.

Usually, such plugins are installed as packages in the Unity Editor (Unity Technologies, 2023a) using Git (Torvalds, 2023)

cardActionDataList
System.Collections.Generic.List`1[NueGames.NueDeck.Scripts.Data.Collection.CardActionData]
System.Collections.Generic.List`1[NueGames.NueDeck.Scripts.Data.Collection.CardActionData]
System.Collections.Generic.List`1[NueGames.NueDeck.Scripts.Data.Collection.CardActionData]
System.Collections.Generic.List`1[NueGames.NueDeck.Scripts.Data.Collection.CardActionData]
System.Collections.Generic.List`1[NueGames.NueDeck.Scripts.Data.Collection.CardActionData]
System.Collections.Generic.List`1[NueGames.NueDeck.Scripts.Data.Collection.CardActionData]

Fig. 4. Not editable instances rendered as labels using the ToString() function.

repository links. This requires the user to utilize Git, which can complicate the plugin installation process for inexperienced developers. To avoid this, we packaged the plugin as a Unity asset and uploaded it to the Unity Asset Store. This allows the user to install the package by just pressing a couple buttons and does not require them to have Git installed on their development machine. Additionally, publishing on the Unity Asset store makes the plugin easier to find as the user can search for the tool on the official Unity Asset Store website.

B. Iterative Approach

We followed an iterative approach when developing the tool. Each time we made changes, the tool was shown to several developers, who gave feedback on the tool.

Initially, every second column was colored a different shade of grey. Testers found that coloring every second row was more useful.

The font size was initially made bigger to allow for better readability. It was reverted to default font size after testers mentioned that they would prefer to see more values in the table at once.

C. Implementation

a) Scriptable Object Selection: When working with Scriptable Objects, developers create custom Scriptable Objects for their needs. To enable working with custom Scriptable Objects, we added an Object Field to allow the user to select the type of Scriptable Object they want to edit. The Object Field is a built-in UI element that comes with UI Toolkit, a package for working with the User Interface in the Unity engine (Unity Technologies, 2023a). It was important to limit the type of object to be selected to Scriptable Objects. We save the selected value the editor is closed in case the user closes the SOT window and wants to come back to the same Scriptable Object they were editing before.

b) Grid Layout: The UI Toolkit does not come with support for a grid layout by default. We therefore created our own implementation for positioning UI elements in a grid.

First, we calculate the width of every column, based on how much space the cells will require to be shown. In most cases, we use the string length to determine the width of the cell. There are special cases such as the color field. Cells which allow editing of a color value are sized solely according to the name of the color variable, which is used for the label in the header row of the column.

Once all of the column widths are found, they are applied to every cell in the column to ensure that all values are aligned.

In addition to this, styling of the input fields has to be adjusted so that the values are aligned properly.

```
margin: 1px;
padding: 0;
border-width: 1px;
```

To separate the cells more clearly, the right border is colored grey.

```
border-right-color: grey;
```

c) Supporting Different Types: In the Unity Editor (Unity Technologies, 2023a), different types of variables are edited using different input fields. We check the type of each variable found inside the given Scriptable Object and dynamically provide a fitting input field for it.

```
...
if(value.GetType() == typeof(UnityEngine.Color
    ) || value.GetType() == typeof(UnityEngine
    .Color32))
{
    visualElement = new ColorField();
    ((ColorField)visualElement).
        SetValueWithoutNotify(value);
}
if(value.GetType() == typeof(UnityEngine.
    Vector2))
{
    visualElement = new Vector2Field();
    ((Vector2Field)visualElement).
        SetValueWithoutNotify(value);
}
...
```

Variables that refer to Transforms, GameObjects and similar types are made to not accept references to scene objects as Scriptable Objects should not have references to instances of objects found in the scene.

```
((ObjectField)visualElement).allowSceneObjects
     = false;
```

d) Unsupported Types: As discussed in the Design section, certain types cannot be given an input field because of size required to display them. In these cases, a label is used to display the variable.

To make it easier to edit the value, we added functionality which opens the Scriptable Object in the inspector view when the label is clicked.

```
element.RegisterCallback<MouseUpEvent>((evt)
    => { Selection.activeObject =
    scriptableObjectData.
    scriptableObjectInstance; });
```

To support hiding of the read-only values, we added a toggle which hides the respective columns.

```
...
foreach(var field in fields)
{
...
    if(hideReadOnlyFields)
    {
        if(MakeVisualElementForValue(field.
            GetValue(scriptableObjectInstance)
            ) is Label)
        {
            invalidFieldsRemoved.Remove(field)
                ;
        }
    }
}
return invalidFieldsRemoved;
```

Removing the columns reduces the width of the Scroll View content. This should force the Scroll View to automatically resize, however there is currently a bug in the Unity Editor (Unity Technologies, 2023a), which prevents it from happening. To force the size of the Scroll View to update, we resize the window slightly. Not enough for the user to notice, but enough to force an update of the Scroll View.

```
var new_len = new StyleLength(UnityEngine.
    UIElements.Length.Percent(scale_swap ?
    99.9f : 100f));
ScrollView scroll_view = rootVisualElement.Q<
    ScrollView>();
scroll_view.style.width = new_len;
scroll_view.style.height = new_len;
scale_swap = !scale_swap;
```

e) Documentation: Documentation describing how the tool should be set up used was also included together with the tool. The documentation can be found in Appendix B

VI. Methods

We conducted semi-structured interviews on experts to gain an understanding of how game balancing is achieved with our tool Scriptable Object Table (SOT) in the development process, as this method allows collection of rich and detailed data on game balancing with a good amount of flexibility (Bjørner, 2015).

A. Participants

We used purposive sampling to gather participants, as it allows researchers to use their own expertise to select the right participants (Bjørner, 2015). In this case, we were specifically focusing on finding people with a strong background in game balancing and experience with Scriptable Objects in Unity (Unity Technologies, 2023a). To ensure they fit the basic requirements we asked them of their experience during the interviews. This gave us verification that we had the right participants for this study.

7 participants were interviewed. 6 of the participants were male and one was female. All participants resided in Denmark. Experience in the games industry ranged from 2 to 15 years. 4 participants had experience with balancing games. 3 participants had experience with developing Scriptable Objects and developing tools in game engines for game designers.

B. Procedure

The interview was conducted remotely via video conferencing, with one person interviewing and another taking notes. Note-taking was done to facilitate the interview and ensure the answers were understood correctly. Furthermore, the note-taker could pitch in with follow-up questions if needed.

To ensure a structured and effective interview, we made a guide document (Appendix A) consisting of key themes we needed to address. The guide was open-ended, allowing us to diverge and dig deeper into certain areas.

The participants were asked to use SOT before the interview until they got a good sense of its utility. To assist them, we provided documentation alongside the tool they could refer to when learning the tool. In some cases, the participants did not have time to use the tool, so we presented the tool during the interview for them instead.

The interviews were recorded and transcribed for referencing and analysing the data.

1) Pilot Test: Before conducting the full-scale interviews, we did a pilot tests on our co-developers. This ensured the process was effective and efficient, and gave the interviewee and note-taker practice, improving the overall quality of the interviews.

C. Data Analysis

We did thematic analysis to find recurring themes within the data. This method offered flexibility and the ability to analyse the complexity of the data.

We used Miro (Inc., 2023) to transform interview recordings into flexible items we could move around and cluster together. Because of the extra paraphrasing step of noting down in Miro, we risked a loss of context. To avoid this, we made sure that both researchers coded the interviews for intercoder reliability, and agreed that each one would note down and describe all sentences the participant said, removing the risk of selection bias.

Both researchers analysed all the interview data without consulting with each other. Only when both researchers were completely done, then we had a meeting to discuss common findings and any discrepancies within our analysis. Finally, we made a graph and established relationships between the themes, creating a framework for how SOT can be used for balancing games.

VII. Findings

A. Participant Understanding of Balancing

Interviewees gave a variety of definitions for what balancing is. One common trend that arose was the idea that balancing is very dependent on the game itself. Game feel, difficulty and interesting choices were also mentioned as goals for balancing. Social deduction and asymmetrical games were mentioned as example of games that are difficult to balance.

All approaches mentioned in section II-B were mentioned during interviews without being brought up by the researchers. 4 interviewees mentioned they employed the intuition approach when balancing games. 3 interviewees mentioned playtesting as an integral part of balancing a game. The mathematical approach was mentioned by 3 participants, however only one of them had used it in the past in some way. The metrics approach was mentioned by 2 participants, either of which had not employed it in their own games. The importance of iteration was stressed by 4 interviewees.

1) Common Challenges When Balancing: Based on the interviewees' prior experiences with other tools, we found that they shared some common challenges when balancing games.

a) Navigation in the Editor: Navigation was mentioned as an issue by all 7 participants in different ways. 4 participants mentioned the frustration of having to click a lot to navigate between different windows inside the game editor to adjust and compare values. 5 interviewees mentioned finding the values that need adjustment in the project as a common issue. Sometimes the value that should be adjusted is not clear from the start according to some participants.

b) Getting an Overview: 4 participants said that it is hard to get an overview of the data, when it is scattered across the project. It often led to the development of special tools for getting an overview or the change of project structure to facilitate better overview.

c) Playstyles: 1 participant mentioned that different player playstyles increase the amount of playtesting that needs to be done to ensure a change in the balance is good for the game.

B. Enhancing the Iteration Process

From our analysis of the responses, we observed four themes around concepts closely linked to the iterative process of game balancing. Scriptable Object Table (SOT) and advanced tools for the domain of game balancing appear to have high potential to empower game developers with a more integrated, user-friendly, and robust iteration experience by providing the ability to (1) control the view of in-game data; (2) tweak data quickly within the engine; (3) track their changes; and (4) visualize the impact of their changes. In the following sections we will dive deeper into each of theme, providing further details and how they relate to SOT.

1) ***Control of How They View the Data****:* Our findings suggest we should help them in any way we can to view the data in a controlled manner so they only see what they want to see, reducing the mental effort required to parse the data when doing balance adjustments.

a) The Tool Provides a Good Overview: All 7 interviewees found that the tool made it easier for them to get a comprehensive overview of the balancing elements in the game project. They found it helpful to be able to see all the data at once in a familiar spreadsheet-like manner. As one user put it, using our tool *"feels like being a game master in D&D"*. They found it useful to have related information in one place with all instances of the same type together, being able to see how the game was interconnected. To improve this aspect, some users suggested making the tool as extendable as possible to allow for modification by developers. This can

enhance the overview aspect, with the ability to manipulate data for a tailored experience, and special filtering rules.

b) Filtering: 6 interviewees found filtering to be a high priority for the tool. Being able to choose what they see by filtering out specific pieces of data and decluttering the interface allows for more efficient work. This need was observed in multiple points, with interviewees asking for the ability to hide certain rows, use regex for filtering, and apply special filtering rules. They also expressed the desire to filter data by specific attributes or to show objects with certain characteristics.

c) Sorting: 5 interviewees wanted the ability to sort data based on fields. Sorting can improve users' understanding of the balancing elements by enabling them to see minimum and maximum values at a glance, and how the data is laid out. Users suggested sorting by column headers, for instance — in the case of roguelite deckbuilders — card rarity.

d) Categorization: 2 interviewees suggested categorization to provide better visual distinctions through features, which could provide an improved overview. Categorizing the data could be done by splitting objects vertically based on selected fields or coloring cells based on specifications declared by the user.

e) Columns and Rows: 6 interviewees expressed a need for having more control over rows and columns. They appreciated the ability to hide read-only values, but wished for more ways to hide irrelevant columns or rows of their own choosing. Furthermore, they wanted to resize and rearrange rows by dragging them around, customizing the view to their liking and allow them to focus on the most relevant information. This customization would help reduce horizontal scrolling, which they felt strenuous in the current version of the tool.

f) Showing Multiple Types: 4 interviewees found it limiting to only be able to view one type of object at a time. This drawback prevented them from getting a holistic view of multiple interconnected elements in the game. Features such as tabs for showing different objects or allowing the display of different types of objects simultaneously would greatly improve this aspect.

g) Finding Used Scriptable Objects: 1 interviewee contemplated whether it was possible to extract and show only those Scriptable Objects that are in use in a project. This could provide a truer picture of the game, as some objects might be deprecated and viewing those could cause the designer to accidentally balance their game by comparing the wrong things. Identifying which objects are in use is a complex task, but some interviewees suggested methods such as counting references in a build to address this issue.

h) Save the Layout: 1 interviewee first tried using the Unity search in project view before trying our tool. They said they often use that workflow to filter through project assets and found it useful to be able to save their search queries in Unity. This suggests that the ability to save layouts in SOT can prove useful by allowing users to quickly return to a familiar setup.

i) Input Fields: 5 interviewees discussed specific features of the input fields that are interacted with when using SOT. They appreciated some of the features Unity provides such as auto-highlighting and expressions. Some users suggested improvements such as providing a popup editor window for custom fields, seeing images in the table, adjustable markers, and custom property drawers. Popups could be used to display additional information that does not fit in a cell; mouse hover could reveal images; and dropdowns could display additional information. Showing all values could be more beneficial, even if it compromises the aesthetic appeal of the layout. They suggested expanding cells to display more complex data types like arrays. One issue was found where Vector4 values did not fit in the given cell.

j) Graphs: 2 interviewees suggested graphing capabilities to give a better overview. Such functionality could enhance the tool's overview capabilities, allowing for a quick visual interpretation of data patterns, for instance by showing distribution of costs in cards in a roguelite deckbuilder.

*2) **Changing Data Quickly**:* Our findings showed various ways that a balancing tool such as the Scriptable Object Table (SOT) can help game designers make quick changes to the data.

a) Tweaking Data: 4 interviewees touched on how effective tweaking of data is key, given the intertwined nature of game systems. There is a need for a system that helps designers by alleviating pain of modifying data one by one with easy access to the many tweakable parameters. 1 interviewee also appreciated that clicking an input field automatically selects the entire input, although that is a general Unity feature, it further emphasizes the desire of unobtrusive workflow when tweaking data to be more efficient.

b) Finding Data: 3 interviewees were particularly interested in how SOT helped them simplify the process of finding the appropriate data to adjust by experimenting with the data. Experimentation is integral in developers' balancing workflows, as they often tweak multiple values until they locate the one that impacts gameplay as intended, where SOT can offer valuable support.

c) Comparing Values: 4 interviewees found it nice to be able to compare values side by side, since in practice they are usually scattered around. By collecting these values and making them editable in one place with the SOT, the scattered nature of Unity's inspector view can be counteracted. This aids designers in the process of game balancing by providing a quick way of modifying game objects in relation to each other. Furthermore, seeing values next to each other makes it possible to see minimum and maximum values at a glance without searching all over the project.

d) Expressions: 2 interviewees said that the potential introduction of 'expressions' would allow operations on values and the inclusion of calculated values, thus expanding the versatility of the SOT. For example, the user could type "*2" to double the value in the field.

e) Multi-edit: 2 interviewees emphasized a need for multi-editing, enabling developers to select and alter several items concurrently. Coupled with multi-editing, expressions can prove SOT to be an immensely powerful bulk editor.

f) Undo: 1 interviewee was concerned with the way undo works in their version of SOT. They found that a clunky undo system would do harm, as it risks losing changes by mistake. It is thus vital that undo is intuitive for the user in SOT.

g) Showing Unity Types: 2 interviewees recognized that a major benefit of SOT is that it can understand Unity types. As an integrated tool in the engine, it can preserve type recognition, displaying any unique properties such as custom classes and images. For comparison, this context is lost when exporting to external solutions such as spreadsheets.

h) Restrictions on Input: 1 interviewee desired input restrictions within the SOT to maintain integrity of the data and avoid mistakes that could break the game. They suggested tooltips and data validation for unique fields, which can enhance the user experience and prevent errors.

i) Jumping to Object from Scriptable Object Table to Inspector View: 4 interviewees found the ability to jump from the SOT to the inspector view useful, especially because of the limitation of SOT to display complex data. However, the labels that took the user to the Scriptable Objects could better signify that they were interactive elements. For example, they could change color on hover, like buttons do.

j) Runtime Tweaking: 1 interviewee was interested in adjusting parameters during runtime. This would support their ideal workflow of making rapid adjustments during testing sessions, and providing this ability in SOT thus helps facilitating an efficient iterative game development. This functionality is already supported by Scriptable Objects.

k) Live Games: 1 interviewee suggested that the tool might be more useful in live games, since then there is a live playerbase and can make more meaningful changes based on metrics and player feedback. Furthermore, in that context the data would be set up in the same way across iterations, meaning that SOT would show a more familiar setup.

*3) **Tracking the Changes**:* Our findings indicate that balancing tools can be used to help designers track their changes when balancing games, as three interviewees expressed a struggle to do so. In practice, the best designers make small incremental changes and do thorough testing after each tweak. This allows them to better understand their impact and isolate issues more effectively. Interviewees stressed the value of source control for being able to roll back changes. Many designers find source control difficult to work with and our tool can be improved with deeper integration, thus simplifying the process.

*4) **Showing Effect of the Changes**:* Finally, when it comes to enhancing the iteration process in game balance, our findings show that balancing tools can help designers see the effect of their changes. 2 interviewees were excited about the prospect of integrating our tool with an analytics system where changes and outcomes are recorded and linked. This closely aligns with the metrics approach and would bring additional depth to the balancing process, providing quantitative measures of change impacts. Furthermore, they imagined an AI-driven system that could monitor the analytics data and suggest potential correlations between tweaks and shifts in the game's meta. The AI could even anticipate the likely effects of changes before they are implemented, thereby flagging possible significant shifts in the game dynamics and alerting designers of unintended consequences. Such a system could offer a safety net against extreme changes that might throw the game out of balance, such as prematurely making powerful abilities accessible.

C. Specific Feedback on the Scriptable Object Table Plugin

Some of the feedback provided by the interviewees was focused specifically on the tool, rather than on the needs of developers. This section will cover this feedback.

1) Tool Architecture:

a) Scalability: 3 interviewees pointed out that the tool would become more useful as the game project grew in size. 2 interviewees suggested that for smaller games, the usefulness of the tool is limited and that its usefulness might be outweighed by the effort required to install the tool. 1 interviewee said that the scale of the project did not make the tool less useful. Some interviewees mentioned that the tool would need additional features like paging, filtering and search to support bigger projects and avoid overwhelming the users.

b) Paging: Paging is a proposed feature that would split the table into several pages. It was suggested by 1 interviewee to address potential performance issues that would arise when large amounts of data would be shown in bigger projects.

c) Integration Into Project: 6 interviewees pointed out the importance of making the tool easy to integrate with existing game projects. 3 interviewees pointed out the issue of having to change the way the Scriptable Objects are setup to avoid having unsupported types in them. 1 participant stressed the importance of making this tool support other game engines, potentially even making it engine agnostic. 1 interviewee pointed out that the tool is a more modular approach than using data tables in Unreal Engine (Epic Games, 2023), because it does not require the data references to be setup in a specific file. Several interviewees found the ability of referencing objects in the table to be a useful feature. 2 participants found it nice that no extra setup is needed to use the tool (apart from installation).

d) Extensibility: 4 interviewees expressed the importance of making the tool extendable by game developers. They all gave examples of situations where custom data might require different visualization in the table. 1 participant gave an example of custom filtering options being programmable as well.

2) UX Improvements:

a) Scriptable Object Selection: 6 participants pointed out that it would be better to select from Scriptable Object types rather than selecting an instance of a type to show in the table.

b) Missing Functionality from Inspector View: 3 participants missed various features that are implemented in the inspector. This included showing lists, nested values and other supported types. Features provided by attributes (Unity Technologies, 2023b) in Unity were also missed.

c) Visual Appeal: 2 interviewees mentioned that better visual appeal could improve the user experience. As an example, custom icons for fields and thumbnails for images were given.

d) Documentation and Learnability: 5 participants found the tool to be self-explanatory. 1 interviewee gained a better understanding of the tool after using the documentation. 2 participants did not read the documentation, which lead to confusion when using the tool. The documentation was missing explanation for how objects are added to the table. It also missed information on what Scriptable Objects and what types are supported by the tool.

e) Tooltips: 1 participant expressed the need for better tooltips. They said that they should be programmable to show anything the user wants or use built-in unity attributes.

3) Opinions of the Tool: 6 participants had a positive opinion of the tool. 3 participants expressed interest in using the tool in the future. 1 participant did not think they could use the tool. 2 participants said that the name of the tool was fitting. 2 interviewees expressed excitement about further development of the tool.

4) Alternative Solutions: Spreadsheets and custom editor tools were identified as the main alternatives to using Scriptable Object Table. Both of these solutions were found to be more time intensive to implement. Spreadsheet solutions offered familiarity and a bigger feature set for analytics and visualization. However, spreadsheet solutions lacked the integration that built-in Unity solutions offer.

2 participants said that custom editor tools have even better support for specific needs of the project at a cost of even greater implementation time.

5) Other Uses: 3 participants expressed that the tool could be used for more than just balancing. Bulk input of values and getting to know a project were given as examples of other uses. 1 participant thought that the tool is not useful for much outside of balancing.

VIII. Discussion

A. Validity and Reliability

During the interview process, discussions naturally emerged about all four different approaches to game balancing, which aligns with our prior research on the topic. However, it should be noted that the interviewees lacked hands-on experience with the mathematical and metrics approaches, although they understood their importance. Unfortunately, due to time constraints and our limited network, we were unable to interview designers with experience in these approaches. For future studies, it would be beneficial to focus on participants from larger companies that are more likely to utilize these approaches for game balance. In light of this, one of our interviewees had experience working with designers who employed the mathematical and metrics approaches, providing us with valuable insights that the designers themselves may not have had.

It is worth noting that there are numerous ways to balance games, which can vary based on the specific game and the designers involved. Therefore, we acknowledge that our study may have missed certain insights that could have been obtained by interviewing other designers. Although time constraints prevented us from conducting further interviews, we believe that conducting additional interviews after further project development would provide valuable information.

Furthermore, it is worth addressing some potential biases with our study. Firstly, since we personally knew some of the participants more than others they may have been more skewed to react positively in the interviews. However, the nature of results are more objective indications of what worked and what was missing in our tool, with little focus on whether they liked our tool or not.

Secondly, 2 of our interviewees did not have a chance to actually use the tool, and their input was based primarily on their understanding and perceptions of it, rather than practical experience. This could affect the validity of their feedback compared to those who had direct interaction with the tool. Therefore, future work could seek to ensure that all participants have a hands-on experience with the tool to collect more uniformly valid and practical input.

B. Web of Lies

Developing *Web of Lies* allowed us to get the perspective of a designer that balances games. Doing this allowed us to find and experience natural use cases for the tool, which is impossible to gain from the interviews alone. Of course, the ideal scenario would have been to work in a team of more experienced developers, but that was out of our reach. We will be continuing to use the tool as we develop the game further as we have found it very useful for both balancing and as a bulk editor. We hope that we get further insights as we get closer to the final release of the game.

C. Future Work

There is a good potential for future works with our tool for balancing games. We propose an "in-the-wild" approach, in close collaboration with developers as they use the tool to develop their game in a natural environment (Rogers, 2011). By implementing changes that incorporate the key elements from the results of the interviews, we can investigate the usefulness of the tool before and after updating it, and therefore get an even better understanding of developer' workflows with the tool. With this approach we can try to verify our findings, exercising the reliability of our findings.

1) Further Development of Scriptable Object Table:

a) Fixes: Some of the issues identified during the testing are clearly bugs that should be fixed before developing further features. The cell width for showing Vector4 values should be increased so the whole value fits. Scriptable object selection should be fixed so that the user has to choose between types rather than instances of Scriptable Objects. This would make the tool less confusing for first-time users as well as making Scriptable Object types easier to find in bigger game projects. Undo functionality should be improved. Currently, the undo feature treats a single key press as an action, which means that to undo a change of a string value, the undo action must be

done once per character. It should be replaced with undoing the whole change of an input field at a time. As the tool would go on to support multi-editing features, those actions should be undone at once as well. These fixes offer a lot for an estimated relatively short implementation time.

b) New Features: After some discussion, we have identified the next features that we think should be implemented and tested next, based on development costs and usefulness to the users.

It is clear that the users need good control over the data they see to work most effectively. Sorting would be easy to implement, while being very useful in certain cases. Supporting multiple windows at the same time would allow to modify and compare different types of Scriptable Objects, which seems to be a common need when balancing games. Filtering the data could be implemented with relative ease and it would be a powerful feature as it would allow the user to control which objects they see in the table, especially useful for bigger game projects. Choosing whether to hide or show specific columns is another feature that is quick to implement, that would give even more control. Limiting which columns are shown would allow the user to reduce the visual clutter while editing. Together with multiple windows, it would allow for very good overview of many different variables across a game project. Multi-edit is the next feature, which would speed up the bulk editing process. More support for built-in engine features like showing different types and support for attributes would be very beneficial to make the tool more flexible with different projects and reduce the effort needed to modify the table itself.

c) Considerations for Future Development: Multiple design considerations arose during the development and interview processes. It is now clear that the fact that the tool adapts to the project is very important. It reduces the time for starting to use the tool. It also means that the tool is flexible, which would encourage users to use the tool for multiple different projects. This is a very important selling point as a lot of the alternative tools we looked at seemed to require effort to start working with.

Extensibility is a clear need going forward to ensure the teams that work with the tool can adapt it to their needs with ease. In addition to this, the code of the project should be very well documented to ensure programmers can figure out how to modify the tool fast. If adapting the tool is too hard, teams could drop it altogether to develop their own solutions, spending valuable development time on tools. Additionally, referencing the results of this study could help guide the developers in extending the tool.

2) Application of AI for Balancing Games: AI is being applied for balancing for level generation (Shu et al., 2021), in-game AI actors (Pfau et al., 2020) and dynamic difficulty adjustment (Xue et al., 2017). We find that there are opportunities for applying novel machine learning algorithms for assisting in the workflows for game balancing. We see how AI could be used to analyse the changes made by the designer and give them feedback or warnings of some other effects of a given change to balance. The AI could use data collected from live games to make such judgements. Additionally, AI could potentially allow for dynamic generation of overview windows, based on what the designer wants to adjust.

3) Applied Balancing Theory: While our interviews revealed that few designers discussed balancing theory in depth, they did cover all four balancing approaches. We find that a need exists to further emphasize the importance of applying in-depth balancing theory on games. For future work, we propose expanding our tool's features to include more balancing elements by integrating the balancing methods that we researched, such as including cost curves and central values. This theory-based approach could facilitate a more grounded approach to game balancing, potentially improving the overall efficacy and efficiency of the process.

CONCLUSION

This study explored game balancing practices among game designers, identifying the range of strategies used and common challenges faced in the process. We developed a game *Web of Lies* to better understand the needs for balancing; and we made a tool Scriptable Object Table (SOT) to assist in the process. Through a series of interviews, we dived deeper into balancing with our tool SOT. Based on the interviews, we did a critical examination of SOT as a potential tool to streamline the game balancing process, providing a nuanced understanding of its strengths and potential areas for improvement with regards to balancing.

Returning to our primary research question:

> *How can we develop a game balancing tool that streamlines the workflow for developers, allowing them to easily iterate and incorporate balancing principles in their projects?*

Our findings provide clarity for a path forward. The insights indicate the potential value of such a tool for balancing, like improved control of how they view the data, changing data quickly, tracking the changes, and showing effect of the changes. Furthermore, we explored ideas such as the importance of overview and iterations, integration with analytics tools, improved version control, and future potential of AI in balancing. These features, in conjunction with usability improvements, could significantly streamline the game balancing process and assist game developers in creating more balanced and engaging gaming experiences.

In conclusion, this study not only broadens our understanding of game balancing practices and the obstacles designers encounter, but it also provides a roadmap for the development of more refined and effective game balancing tools. By building on these findings, we can revolutionize the game development process, facilitating a more intuitive, dynamic, and effective balancing workflow integrated with the developer environment.

References

Adams, E. *Fundamentals of game design*. Pearson Education, 2014.

Agent40. Scriptable object editor, 2022. URL https://assetstore.unity.com/packages/tools/utilities/scriptable-object-editor-235840.

Becker, A. and Görlich, D. What is game balancing?-an examination of concepts. *ParadigmPlus*, 1(1):22–41, 2020.

Bjørner, T., editor. *Qualitative Methods for Consumer Research*. Hans Reitzels Forlag, Copenhagen, 2015.

Brown, M. How games get balanced, 2019. URL https://www.youtube.com/watch?v=WXQzdXPTb2A.

Brown, M. The two types of random in game design, 2020. URL https://youtu.be/dwI5b-wRLic.

Credits, E. Perfect imbalance - why unbalanced design creates balanced play - extra credits, 2013. URL https://www.youtube.com/watch?v=e31OSVZF77w.

Dave, N. N., Sparks, M. A., and Farouk, S. S. An introduction and guide to becoming a social media savvy nephrologist, 2022.

Del Vicario, M., Bessi, A., Zollo, F., Petroni, F., Scala, A., Caldarelli, G., Stanley, H. E., and Quattrociocchi, W. The spreading of misinformation online. *Proceedings of the national academy of Sciences*, 113(3):554–559, 2016.

Engelstein, G. Ludology: Gametek classic 183 - input output randomness, 2018. URL https://www.dicetower.com/game-podcast/ludology/gametek-classic-183-input-output-randomness.

Epic Games. Unreal engine 5. [Game Engine], 2023. URL https://www.unrealengine.com/en-US. Accessed May. 19, 2023.

Felder, D. Design 101: Balancing games, 2015. URL https://www.gamedeveloper.com/design/design-101-balancing-games.

Final, F. Sheet codes, 2022. URL https://assetstore.unity.com/packages/tools/visual-scripting/sheet-codes-157729.

FromSoftware. Bloodborne. https://www.playstation.com/en-dk/games/bloodborne/. Accessed: 2023-05-24.

Fullerton, T. *Game design workshop: a playcentric approach to creating innovative games*. CRC press, 2014.

Fyrvall, B. Bona data editor, 2022. URL https://github.com/bonahona/BonaDataEditor.

Get Media Savvy. Get Media Savvy: About. https://getmediasavvy.org/about/. Accessed: 2023-03-06.

Google LLC. Google sheets, 2023. URL https://www.google.com/sheets/about/.

Inc., R. Miro, 2023. URL https://miro.com/.

Klaus Teuber. The Settlers of Catan (English third edition 1997) | Board Game Version | BoardGameGeek. https://boardgamegeek.com/boardgameversion/24859/english-third-edition-1997, a. Accessed: 2023-05-23.

Klaus Teuber. Catan: Junior | BoardGameGeek. https://boardgamegeek.com/boardgame/125921/catan-junior, b. Accessed: 2023-05-23.

Kubey, R. W. Media literacy in the information age: Current perspectives. 1997.

Mega Crit Games. Slay The Spire. https://store.steampowered.com/app/646570/Slay_the_Spire/. Accessed: 2023-05-23.

Mian, A. and Khan, S. Coronavirus: the spread of misinformation. *BMC medicine*, 18:1–2, 2020.

Microsoft Corporation. Microsoft excel. Retrieved from https://office.microsoft.com/excel, 2023.

Millard, A. How "bad" balance can be a good thing, 2022. URL https://www.youtube.com/watch?v=l1WYmHz3hog.

Perednytė, D., Flig, K. E., Munk, S., and Timčenko, O. Information Disorder: Raising Interest Through Games. Unpublished, 2023.

Pfau, J., Liapis, A., Volkmar, G., Yannakakis, G. N., and Malaka, R. Dungeons & replicants: automated game balancing via deep player behavior modeling. In *2020 IEEE Conference on Games (CoG)*, pages 431–438. IEEE, 2020.

Rogers, Y. Interaction design gone wild: striving for wild theory. *interactions*, 18(4):58–62, 2011.

Schell, J. *The Art of Game Design: A book of lenses*. CRC press, 2008.

Schreiber, I. Game balance concepts. https://gamebalanceconcepts.wordpress.com/2010/06/17/hello-world/, 2010. Accessed: 2023-02-20.

Schreiber, I. and Romero, B. *Game balance*. CRC Press, 2021.

Shu, T., Liu, J., and Yannakakis, G. N. Experience-driven pcg via reinforcement learning: A super mario bros study. In *2021 IEEE Conference on Games (CoG)*, pages 1–9. IEEE, 2021.

Sylvester, T. *Designing games: A guide to engineering experiences*. " O'Reilly Media, Inc.", 2013.

Torvalds, L. Git, 2023. URL https://git-scm.com/. Accessed May. 8, 2023.

Tyroller, J. Top 10 tips on how to balance your game, 2019. URL https://www.youtube.com/watch?v=uo9qIDbJvT8.

Unity Technologies. Scriptable objects, 2018. URL https://docs.unity3d.com/Manual/class-ScriptableObject.html.

Unity Technologies. Unity: 2022.2.15f1. [Game Engine], 2023a. URL https://www.unity.com. Accessed May. 8, 2023.

Unity Technologies. Attributes, 2023b. URL https://docs.unity3d.com/Manual/Attributes.html.

VisualWorks. Csv serialize, 2019. URL https://assetstore.unity.com/packages/tools/integration/csv-serialize-135763.

Von Baeyer, H. C. *The Fermi solution*. Dover Publications, Mineola, NY, January 2001.

Wardle, C. et al. Information disorder: Toward an interdisciplinary framework for research and policy making (2017). 2017.

Xue, S., Wu, M., Kolen, J., Aghdaie, N., and Zaman, K. A. Dynamic difficulty adjustment for maximized engagement in digital games. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 465–471, 2017.

Zarembo, I. Analysis of artificial intelligence applications for automated testing of video games. In *ENVIRONMENT.*

TECHNOLOGIES. RESOURCES. Proceedings of the International Scientific and Practical Conference, volume 2, pages 170–174, 2019.

APPENDIX A
INTERVIEW GUIDE DOCUMENT

The following two pages contain the interview guide document we used when conducting the interviews.

Introduction

- Greet the participant and **introduce yourself**
    - **Simonas interview**
    - **Kristinn notes**
- Thank you for trying out our tool
- Briefly describe the project and its goals
    - **Goals**: make it easier for game designers to balance games by facilitating an integrated environment
- Explain purpose of interview and expected duration
    - **Purpose**: gain a better understanding of how our tool can be used for balancing games
    - **Expected duration**: 20-30 minutes
- Confirm that the participant has **consented** to participate and that the interview will be **recorded**

Background and Experience

- Ask the participant to provide a brief overview of their **background in game development**, with a focus on their **experience with game balancing**
- Inquire about their **experience with Unity and Scriptable Objects**

Game Balancing Approaches and Challenges

- Ask the participant to describe **their general approach to game balancing**
    - Could they give some examples?
- What are the common **challenges they face** in the process
    - Could they give some examples?

Using Scriptable Object Table (SOT) in Game Balancing

- What was their process when using the SOT?
- What advantages do they see in the tool?
- Could you compare SOT with alternative solutions?
    - What drawback does SOT have in comparison?
- Discuss any challenges they faced while using SOT
    - how did they overcome them?
    - What issues do they see with the tool?

SOT Features and Improvements

- Ask for suggestions on **how to improve** SOT to better support game balancing tasks
- Discuss any **additional features** they would like to see integrated with SOT that would assist the balancing process

Conclusion

- Summarize the **key points** from the interview
- Ask if they had **3 main takeaways** they would highlight from the interview, what would they be
- Ask if they have **any additional comments** or questions
- **Thank the participant** for their time and valuable insights

APPENDIX B
SCRIPTABLE OBJECT TABLE DOCUMENTATION

The following page contains the documentation included with the Scriptable Object Table plugin.

Scriptable Object Table Documentation

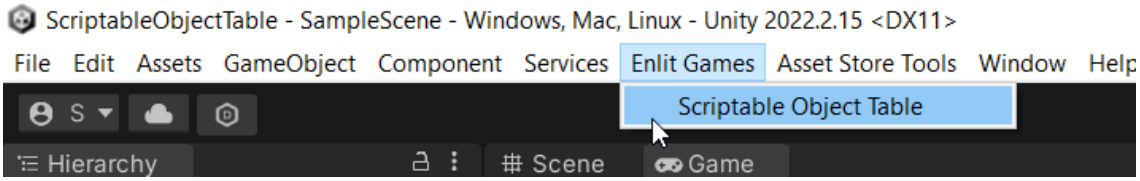
SETUP

If you are not using assembly definitions, you don't need to do anything. Skip to Usage section.

If you are using them in your project, you need to add an assembly definition file inside the the ScriptableObjectTable folder which is located in the plugins folder after installation. Then reference the assembly definitions that contain classes that are used in the scriptable objects in your project.

USAGE

You can find the Scriptable Object Table View under Enlit Games > Scriptable Object Table.



Next, you need to select the Scriptable Object which you want to edit together with other Scriptable Objects of the same type.

You should now see all the Scriptable Objects of the same type in a single table.

You can edit the values here and they will be changed on the actual Scriptable Object. Some more complex values are not supported, like lists or nested Scriptable Objects.

You can also use the **Hide read-only values** toggle to hide values that cannot be edited in the table view, like arrays or lists. You can still click on read-only values to see the scriptable object in the the inspector view.

APPENDIX C
GAME DESIGN DOCUMENT

The following pages contain the full PDF for our game design document.

This document was written for the intended purpose of outlining the design for a full scaled game, so some of the described mechanics or game structures may not be implemented in the prototype used for this project.

Web of Lies

Game Design Document

GMS x Enlit Games

Gossip in the community. Spymaster in the window.

Tip: use document outline for table of contents

Overview

The player takes the role of a spymaster hired by the king to dismantle a rebellion by distributing different types of mis- & disinformation to people, spreading propaganda among the populace. They must balance between giving them manipulative information, and maintaining credibility within the community.

Genre: Singleplayer roguelite deck builder

Media Literacy Goal

The goal of this project is to give players an intuitive understanding of how mis- & disinformation spreads. When most social platforms are engineered for people to publicly ‘perform’ through likes, comments or shares, it’s easy to understand why emotional content travels so quickly and widely, even as we see an explosion in fact-checking and debunking organizations. With an abundance of information, it can be difficult to spend time fact-checking every source of information, to the point where a lot of users lack skepticism. The game aims to showcase the manipulative power of widely spread mis- & disinformation, when people take in information without questioning the intent behind it or checking the legitimacy.

GMS Keywords

Mis-information	When false information is shared, but no harm is meant
Dis-information	When false information is knowingly shared to cause harm
Mal-information	When genuine information is shared to cause harm, often by moving information designed to stay private into the public sphere.
Information overload	A situation in which you receive too much information at one time and cannot think about it in a clear way
Rumor	Information or a story that is passed from person to person but has not been proven to be true
Propaganda	Information, ideas or rumors deliberately spread widely to help or harm a person, group, movement, institution or nation. It is often biased and misleading, in order to promote an ideology or point of view
Spin	To present news or information in a way that creates a favorable impression
Credibility	The quality or state of being credible; capacity to be believed or believed in
Satire / Parody	False or partially false information shared with the intent of being entertaining or comedic. Might not be intended to cause harm but can risk fooling people
False connection	When the headlines, visuals or captions don’t support the actual content. Though the actual content might have genuine information, the headline or similar might result in false conclusions.
Misleading content	Misleading use of information to frame an issue or individual. This information is usually spread to harm the status of or manipulate the public opinion about a specific target.

False context	When genuine content is shared with false contextual information. The use of genuine information is used to lower the guard of skeptics and make the false information seem more believable.
Imposter content	When genuine sources are impersonated. By utilizing people's trust in an information outlet, like a trusted individual or news company, people frame information as having been shared by them to reach a wider audience or to make the false information seem genuine.
Manipulated content	When genuine information or imagery is manipulated to deceive. Withholding or slightly changing details of the information to change the overall message.
Fabricated content	New content that is 100% false, designed to deceive and do harm.

Informing Elements

Social media platforms:

- Twitter, Instagram, Discord, Reddit, Youtube, 4Chan Tik Tok & Facebook are perfect environments to spread various unfiltered information. On these sites information is presented at such a fast pace, there is no way to make sure the information is correct.

Game Mechanics

Resources

MACRO	Credibility	(HP) - you gain or lose credibility depending on the types of information you spread
	Gold	(Mana) - you have a certain amount available each turn, which is used to play different cards
	Deck of Cards	different messages of mis- & disinformation that you can pull from the deck while playing
micro	Cards in hand	single pieces of information that you spread or single actions that you take to affect the

		community
	Rebellious spirit	A measurement of the community or community leaders' intent to rebel. If their spirit is fully reduced they get dissuaded from rebelling
	Information capacity	A single person in the community can only handle sharing a limited amount of information in a short time. If their capacity is reached they get information overload and will retreat

Active mechanics

- Rotation - rotating a Pawn(in either direction) to change its directional connection so it now spreads information with a different neighboring Pawn.
- Swapping - swaps the grid positions of two Pawns while maintaining their current rotation. Can also be used to move a Pawn to an empty grid position.
- Spread harmful information - starting a chain of harmful information from a single Pawn, which then travels along each Pawn's directional connection.
- Shunning - when a Pawn gets shunned it is removed from the grid leaving its previous space empty.
- Inviting - if there is an empty space in the grid a new random Pawn can be invited in to take up that space.
- Collecting rumors - adding new cards to your deck
- Changing Pawn attributes - add, remove or change which attributes are currently in effect for the individual pawns

Passive mechanics

- Grid - the board is a grid with set dimensions and amount of spaces. Each space in the grid may be either empty or occupied by a single Pawn.
- Deck re-shuffling - as cards are drawn from the deck and either played or discarded, they are added to the discard pile. If there are not enough cards remaining in the deck to draw a full hand of cards you draw the remaining cards in the deck, then the discard is immediately re-shuffled into your deck, and you draw the remaining amount up to a full hand of cards.

NPC attributes

- Directional connection - an indication of what neighboring Pawn(s) a given Pawn will spread information to.
- Information capacity (pawn HP) - once hp drops to 0, the Pawn is overwhelmed and cannot take in or spread any information for a turn.
- Special abilities - some Pawns have abilities that use the Active mechanics to hurt your credibility and reinforce the community.
- Literacy (thorns) - if harmful information is spread through a literate Pawn it will damage your credibility.
- Suspicion (block) - a suspicious Pawn is harder to spread information through and can't be affected directly.
- Community leader - If a community leader Pawn is present, then the community's rebellious spirit can only be lowered by spreading harmful information directly to the community leader Pawn.

Card attributes

- Cost - how much gold do you need to pay to play the card.
- Board effect - which active mechanic action will the card let you perform (ie. rotate a Pawn clockwise, or start a chain of harmful information).
- Resource - a card may increase or decrease a specific resource (credibility etc.).

Gameplay Loop

Structure elements:

- Run - a single playthrough of all the game's stages, or until the player loses the game.
- Stage (week) - the game progression is split into several stages with a community leader encounter at the end of each. Each stage consists of 7 encounters (days in a week based structure).
- Encounter (day) - each day is a new encounter. An encounter is a single field of Pawn with a rebellious spirit that you need to decrease. A given encounter may

have different challenge goals, indicating how you can lower the board's rebellious spirit.

- Information gathering - after each encounter gets to gather information by adding a new card to their deck. The available cards are randomly chosen from the total pool of possible cards.
- Visit - on specific days of the week special characters will visit after your information gathering. Each guest offers a different service:
    - Counter intelligence - remove cards from your deck
    - Buy information - use gold to add more cards to your deck
    - Special challenges - accept special challenges for the next encounter that will either reward or punish you based on whether or not you succeed in completing the challenge.
- Turn - an encounter is played out as a sequence of turns. Each turn has several stages:
    - Draw - the player draws a full hand of cards. The deck is re-shuffled if necessary.
    - Play - the player gets to play cards until they run out of action points or cards to play, or decides to end their turn. Playing a card happens in steps:
        - A card is selected
        - A target (if needed) is selected
        - Action points are paid equal to the card cost
        - The card effect happens
        - The card is discarded
        - Any resource changes take effect
    - Pawn actions - if any Pawns on the board have actions, then those actions take effect.
    - Restoration - any Pawns that were overloaded the previous turn are restored.

Encounter

You spread information to Pawns (NPCs), and then they spread messages between themselves based on the directions of their connections.

Information spreads in chains following the directional connection of each Pawn until it reaches a boss, the edge of the grid, and empty grid space, or would be spread to a Pawn that has already been part of the chain.

Game is played in turns, where the community tries to fight back against the spread of mis-information.

Get new cards as you play. Use your coins to spread different types of mis- & disinformation (cards) to change the positions and directional connections between the pawns. And spread harmful information to damage the community's rebellious spirit.

Run

The game is split into runs. Each run is an attempt to dismantle a rebellion by a new spy sent by the king.

A new deck is built every run as the spy gathers more information/gossip about the community.

The run starts with a basic deck of cards/messages gathered by the spy about the community. After each encounter, the player gets to add 1 card, from a random selection to their deck. The pool of available cards gradually becomes better/stronger.

During an encounter, the player goes through turns where the player chooses cards to play, after which the Pawns on the board get a turn of their own. This continues back and forth until the player either depletes the encounter's rebellious spirit(HP) and gets to progress, or has their credibility depleted, in which case they lose the game.

Each encounter represents a day within the week that makes up each stage in the game. On certain days of the week, different special guests will visit. At the end of each week, a community leader will come by, serving as the boss for the final encounter of that stage, and you have to dismantle their rebellious spirit in order to proceed to the next stage.

Meta Progression

___

__

____

Storyworld

The game follows a medieval fantasy setting.

The king is struggling to keep the people united under him, and a rebellion is on the rise out and about in the towns and cities of the kingdom.

You are a spymaster sent by the king to dismantle this rebellion. To accomplish this you go undercover as a barkeep, listening in on rumors and spreading dis-information and propaganda among the patrons, slowly breaking down the people's rebellious spirit. As the rebellious spirit of the people dwindles, community leaders will show up trying to restore their resolve, and so you will need to break their spirit as well.

Art Direction

- Medieval setting
- Designs inspired by old filigree.
- Sound design is relaxed, but gets tenser if your credibility lowers. And different tones are mixed in depending on the unity of the community you are currently facing, and the cards you are playing.

- High-pitch jovial-like fiddle/violin music whenever pawns are outcasted or long information chains are made.
- The tone should shift between optimistic and downtrodden depending on how well the community is doing.

Mood Board

Tech Spec

PC, Steam.

There is potential for the game to be ported to mobile and consoles.

Because the game is a single-player experience, there is no need for servers or databases.

The game would be developed in the Unity game engine.