Technische Hochschule Würzburg-Schweinfurt
Fakultät Informatik und Wirtschaftsinformatik

Bachelorarbeit

Dynamic Dice Simulator: Design and Implementation of a Customizable Dice App with Server-Side Image Integration

vorgelegt an der Technischen Hochschule Würzburg-Schweinfurt in der Fakultät Informatik und Wirtschaftsinformatik zum Abschluss eines Studiums im Studiengang Informatik

Julian Sehne

Eingereicht am: September 2, 2024

Erstprüfer: Prof. Dr. Heinzl Steffen
Zweitprüfer: Prof. Dr. Isabel John

Summary

This paper explores the development of a versatile tool designed to address the limitations found in existing dice simulation apps for tabletop game development. The study identifies critical gaps in current tools, particularly in the areas of customization and creative support. The newly developed tool introduces advanced features such as weighted dice probabilities, custom images for dice faces and flexible group configurations. These enhancements provide game developers with a powerful platform for creating innovative and engaging dice games, significantly surpassing the functionality of existing options.

The findings suggest that this tool effectively fills a gap in the market, offering capabilities that enable developers to push the boundaries of traditional dice games. However, the study also highlights certain limitations, such as the exclusion of iPhone apps from the research, which may have affected the comprehensiveness of the feature analysis. The paper concludes by recommending future research to evaluate the tool's effectiveness as a creativity support system and suggests expanding the scope to include a broader range of platforms and related tools to ensure its continued relevance and evolution.

Abstract

Dice games have long been an integral part of board gaming, yet board game developers often lack adequate tools to effectively create and innovate within this genre. This paper addresses this. First core functionalities and limitations of existing dice game tools in android are identified. The play store and F-Droid as well as from looking at the history of dice games are searched, creating a list of features that serves as a base for creating a dice simulation app. Second, current research in creativity support tools is examined to establish design principles for software developers in creating tools that support creativity. From these design principles together with the market research a list of features is gathered for an android app. The result of this investigation is the development and implementation of a new app, Dynamic Dice, designed to empower developers with advanced customization options and innovative features for dice-based game creation. Finally, an evaluation of the work and future research is outlined.

Contents

1 Introduction

Dice rolling has long been a fundamental element of tabletop gaming[1], providing both randomness and strategy to a wide range of games. As the popularity of tabletop games continues to grow, so does the demand for versatile and customizable tools that can support the development and play of dice-based games.

However, existing dice rolling apps often fall short in meeting the diverse needs of game developers and enthusiasts, particularly when it comes to customization and flexibility. Current market offerings are typically limited in scope, catering to specific games like Yahtzee or offering only basic customization options that do not fully satisfy the creative demands of game developers. For instance, many apps only feature standard, numbered dice, with limited or no support for custom images on dice faces. Even those that do offer custom images, such as "Custom Image Dice," often lack other essential features, like the ability to calculate the sum of a roll, thereby limiting their usefulness in more complex game designs. These constraints prevent developers from fully exploring and realizing innovative game mechanics, highlighting the need for a more versatile and comprehensive tool.

In response to these limitations, this thesis presents the development of a highly configurable dice rolling app designed to empower both tabletop game developers and players. Key features include extensive customization options, such as the ability to define custom die faces with images, apply weighted probabilities and assign states to dice in a dice group.

In developing this tool, the principles of Creativity Support Tools (CSTs) were central to the design process. As defined by Wang (2017): "A Creativity Support Tool runs on one or more digital systems, encompasses one or more creativity-focused features and is employed to positively influence users of varying expertise in one or more distinct phases of the creative process". These tools are instrumental in enhancing creative processes by providing users with the resources they need to explore, experiment and innovate. These principles guided the selection of features for the app, ensuring that the app not only serves as a functional tool for dice simulation but also actively supports and stimulates the creative process.

[1] Defined as "analog games that are played on a flat surface and include both board and card games"[1]

By integrating these principles, the tool offers a platform that empowers game developers to explore new possibilities in game design, fostering creativity in a traditionally rigid domain.

Chapter 2 will define the scope of the app. First it'll explore the state of dice, including their history, mechanics and the current state of dice simulation apps. From this exploration, a feature list will be defined to meet the current needs of users. Following this, the science of creativity support tools will be examined to identify how it can further enhance the app's capabilities with defining features from design principles for CSTs. Chapter 3 shows the app's architecture, business logic and gathered features in detail. Also addressing limitations of the app. Chapter 4 will evaluate the methods and finally chapter 5 will summarize the findings and outline further research.

2 Background and Related Work

This chapter lays the foundation for the Dynamic Dice app by examining the current state of dice games, their mechanics and the existing apps that simulate dice. It will also delve into Creativity Support Tools (CSTs) to identify design principles relevant to an app tailored for tabletop game developers. Based on this exploration, a comprehensive list of features will be compiled to serve as the foundation for the app's development.

2.1 State of Dice games

This first part of the chapter covers dice games. The history of dice, popular dice games and dice simulation apps gives insights on how people use and have used dice to fully understand the needs and requirements for a dice simulation app that aims to help tabletop game developers. The result of this chapter will be a list of features the app will be incorporated in Dynamic Dice to cover all the current usage for such an app.

2.1.1 History

Dice games have been enjoyed by people across countless cultures for centuries. The earliest known dice were excavated in archaeological sites in Mesopotamia, dating back to around 3000 BCE. These dice, typically made of materials like bone, wood or stone, were used in various games and religious practices[1].

In addition to these six-sided dice, knucklebones, dating to around 5000 BCE[2], were also used in ancient games (figure 2.1). Knucklebones were sourced from the ankle bone of hooved animals such as cows and goats and were likely initially used to predict fortunes[2].

[1] `https://www.victorytailgate.com/blogs/news/the-rich-history-of-dice-games`
[2] `http://www.ancientresource.com/lots/roman/roman_dice.html`

Figure 2.1: Dice made from knuckle-bones 5000 BCE

Dice are fundamental to many games because they introduce a random element that makes the outcomes unpredictable and the games more engaging. This randomness ensures fairness, as each player has an equal opportunity to win. Dice provide complexity to a game's design, making it more challenging and enjoyable [3].

The mechanics of dice rolling are deeply rooted in probability. For example, in a typical six-sided die, each face has an equal probability of appearing, but some outcomes, such as rolling a seven with two dice, are more likely due to the number of combinations that can produce that result. This probabilistic distribution adds depth to the gameplay, allowing players to strategize based on the likelihood of certain outcomes [3].

Game mechanics, including dice rolling, are methods invoked by players to interact with the game world. These mechanics are defined as actions (or verbs) that players can perform, such as rolling, climbing, shooting or running. In the context of dice games, rolling the dice is a mechanic that connects players' actions with the game's objectives and challenges [4].

As one of the oldest and simplest mechanics, dice rolling is used in a variety of popular games such as Craps, Yahtzee and Dungeons and Dragons. The consistent random chance provided by dice rolling allows players to plan and anticipate potential outcomes. A key aspect of dice rolling mechanics is the presence of both positive and negative consequences, which add tension and excitement to each roll as players hope for favorable results [5].

Overall, the need for consistent randomness, the distribution of probabilities and the balance of positive and negative outcomes are essential components that make dice rolling an enduring and enjoyable mechanic in games throughout history.

2.1.2 Popular games

Dice games have captivated players for centuries, blending chance and strategy in ways that continue to appeal. Several popular dice games have achieved significant milestones

in sales and player engagement, each utilizing dice mechanics uniquely to enhance gameplay.

Yahtzee is the highest-selling and most popular dice game, with around 50 million units sold annually and 100 million regular players worldwide[3]. The game uses dice to provide a consistent random chance, allowing players to strategize around the possible outcomes each roll can produce. They can reroll a selection to accomplish different face sums.

Catan, released in 1995, according to their website (appendix 5) has sold over 40 million copies globally. It heavily relies on the distribution of probabilities. Players must strategize based on the likelihood of rolling certain numbers, which influences resource generation and trading. This probabilistic element adds depth and complexity to the game.

Dungeons & Dragons (D&D), a quintessential tabletop RPG with over 13.7 million players worldwide[4], leverages dice to determine outcomes in its fantasy narratives. The use of polyhedral dice in D&D adds excitement and unpredictability, keeping players engaged as they face the positive and negative consequences of their rolls[5].

Qwixx, nominated for the "Spiel des Jahres in 2013"[6], features a simple yet strategic dice mechanic. Players must make quick decisions based on their dice rolls, balancing risk and reward to maximize their scores. This quick gameplay keeps the tension high and the decisions impactful. This game also differentiates between different dice through colors.

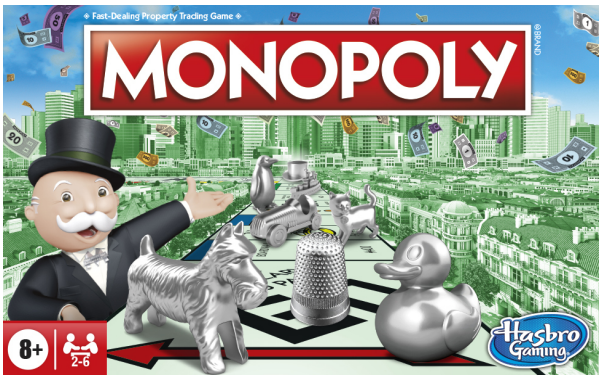

Figure 2.2: Monopoly

Monopoly(figure 2.2, while not exclusively a dice game, heavily incorporates dice rolling to determine player movement around the board. This random chance element ensures

[3] https://playtoday.co/blog/gaming-basics/most-popular-dice-games/
[4] https://www.dramadice.com/blog/the-most-played-tabletop-rpgs-in-2021/
[5] https://agreatbecoming.com/2020/06/20/im-still-amazed-by-the-rise-in-popularity-of-pen-paper-rpgs
[6] https://spiele.tips/alle-spiel-des-jahres-2013-brettspiele-im-vergleich

that every game is different, adding unpredictability and excitement as players navigate buying properties and paying rents [7].

These games highlight the versatility and enduring popularity of dice mechanics. From the simple random chance in Yahtzee and Monopoly to the strategic depth in Catan and the excitement in Dungeons & Dragons, dice mechanics provide essential elements that make these games engaging and enjoyable. They also show different ways dice are used, e.g. summing up the faces or putting them into different categories.

2.1.3 Dice Apps

Dice games, with their rich history and diverse mechanics, have continually evolved to captivate players through the ages. Today, they stand as a testament to the timeless appeal of randomness and strategy in gaming. However, as technology advances, so do the opportunities to enhance and streamline these beloved games. The integration of digital tools, particularly smartphone apps for quickly developing and testing different ideas for tabletop games can bring big benefits for the board game community. This section will uncover what tools players and tabletop game developers currently have.

To find apps a search in the google play store, F-Droid and Copilot has been done. Most apps were found on the play store by searching the keyword "dice" and examining all items that matched the keyword. The same was done on F-Droid. The AI (Copilot) was given a prompt to find Dice rolling apps with features "locking, weight dice, custom images, multiple dice". While browsing through all the results, the images for all the apps from the result were quickly examined, filtering any app that was a rubix cube app simulator, a dice merge game, a dice fighting game, very simplicstic looking dice or a specific simulation of one particular game (in most cases yahtzee).

The remaining 25 apps were all downloaded and examined. The features of the apps were all captured by adding any new feature to a list giving the feature a name and description. Table 2.1 shows the final list. This table provides an overlook for current tools tabletop developers and players can use to simulate dice in games. It shows which features have value to users. A full list of the features for each app can be found in the appendix (5).

[7] `https://www.fun.com/best-selling-board-games-all-time.html`

Feature	Description
Multiple dice	You can roll more than one die at the same time
Selection of dice	The app featured a variety of dice options to choose from (for example dice with 4, 6, 12, 20 faces)
Custom number range	The app featured more than just a selction, you can define the exact number of faces for a die
Images	You can set images as the face of a die
Sum	After rolling multiple dice you could enable to see a sum of the dice
different colors	Dice can have different colors
Re-rolling	The selection of dice can be rolled again
Rolling history	You can see the faces and dice of previous rolls
Adding dice	Dice can be added to the current selection of dice
Removing dice	You can remove a dice from the current selection of dice
Locking	From the selection of dice you can choose which dice should be rolled again or not
Hide dice	You can toggle dice on and off from the screen
Statistic	You can view a statistic for the different dice faces that were rolled
Roll single	You can roll a single die of the selection
Presets	You can create a group of dice that can be selected and rolled
Drawable sides	You can draw a picture for the face of a die

Table 2.1: Feature list from dice apps

Most apps were very simple and did not offer a lot of special features, some apps were focused on RPG games and offered a selction or custom number range for dice, presets, history and sum feature. Only two apps were able to provide custom images for dice in the free version of the app. And both of these apps lacked critical features which makes them cover a fewer variety of games. "Custom Image Dice" is not able to give a sum for dice and thus also not able to calculate a sum which limits the possibility to create Role playing games. "Würfel 2D" has great features, allowing for a custom number range, but also standard dice for quick configuration. You can also create a dice with images, but it would always be a 6 sided dice which heavily limits possibilities again.

From these findings a list of features was created that outlines current needs of users for dice simulation apps and thus requirements for an app focused on developing such games. Though, for the implementation of the app 3 features were crossed. Hiding dice was only found in one app and it is only different from locking dice in a UI perspective, but since the app is not focused on the UI this feature will not be part of the app. Similarly drawing sides is eliminated, it was only found in one app and can be achieved through uploading a drawn image to the app making use of the images feature. Lastly,

statistics were one of the least common features and as it is not directly needed to simulate dice for developing dice games, it's complexity outweighs the gains and is thus crossed out.

Even without these features having all of the other features will outperform other apps of this kind in functionality, effectively filling the gap for dice game creators by providing a robust tool that meets the diverse needs of developers and enthusiasts alike.

2.2 Creativity Support Tools

The second part of this chapter is about Creativity Support Tools (CST). First we need to understand what they are and what the current research on this topic is. Then a list of design principles that serves as guide for software developers seeking to create a tool that supports creativity is created. And finally from these design principles ideas for features are brought about that are collected in yet another list.

2.2.1 Field Overview

This section defines CSTs, their state in Human Computer Interacion, history and current research.

Definition and history of CST

Cherry (2014) offers a clear explanation for CSTs: "A creativity support tool is any tool that can be used by people in the open-ended creation of new artifacts. One could consider a set of paints, brushes and a palette a creativity support tool for painting. One could consider a piano, score sheets and a pencil the creativity support tool necessary for certain types of musical composition. [...] A designer may use a camera to photograph something and then edit, compose and layer that photograph with other graphical imagery in Adobe Photoshop to create a final advertisement that is a physical artifact displayed on a billboard" [6]. This description provides a tangible sense of what such tools might include. However, it underscores the challenge of defining precisely what constitutes a creativity support tool.

A recent literature review offers a broader definition: "A Creativity Support Tool runs on one or more digital systems, encompasses one or more creativity-focused features and

is employed to positively influence users of varying expertise in one or more distinct phases of the creative process" [7].

The origins of creativity support tools are often traced back to Shneiderman (2002), who described them as tools designed to enhance creativity: "The goal of designing creativity support tools is to make more people more creative more often, enabling them to successfully cope with a wider variety of challenges and even straddle domains. Some tasks may be routine, such as doing computations or searching databases, while others require innovative leaps to identify associations, discover correlations or recognize opportunities" [8]. A more succinct definition by Resnick (2005) states that creativity support tools "enable people to express themselves creatively and to develop as creative thinkers" [9].

CSTs fall under the broader domain of Human-Computer Interaction (HCI) research and are part of a larger class of systems known as creativity support environments (CSEs) [7, 6]. CSEs can range from specialized hardware and instrumented spaces to purely collaborative digital environments. Additionally, Creativity Support Systems (CSS) are specialized information systems designed to enhance creative processes across various domains, including product design, idea generation and research and development. These systems aim to stimulate and document creative processes, facilitating the generation of novel and useful ideas or products. Their significance lies not only in practical applications but also in embodying creativity theories, making abstract concepts like divergent and convergent thinking tangible [10].

The genesis of modern creativity research is frequently traced back to Guilford's work, which is considered a pivotal moment in the field's contemporary studies [7]. This event sparked the second wave of creativity research, dominating the late 1980s and early 1990s. The emerging third wave is marked by an emphasis on collaborative work and digitization, with a notable increase in the use of Creativity Support Tools (CSTs) in creative processes. This wave also features a shift towards predominantly empirical research methodologies [7].

CSTs are applied in various fields such as visualization and simulation, concept mapping and brainstorming, architecture and design, mathematics, software development environments, video editing, drawing and painting, animation, music, photography, wikis, blogs, online presence, writing and presentation [11]. These tools support a wide array of creative tasks, from writing and drawing to more technically oriented activities like software design, development and architecture, extending even to the act of research itself [12].

It is also known that the impact of creativity support tools on both individuals and society is substantial, enhancing endeavors in scientific, engineering, humanist and artistic fields [13, 11, 6].

In summary, Creativity Support Tools (CSTs) are defined as tools that aid in various ways in the creation of new artifacts, ranging from tangible items like paints and software to broader digital systems with creativity-focused features. Originating from Shneiderman's (2002) emphasis on enhancing creativity across various tasks and Resnick's (2005) focus on creative expression, CSTs are a part of Human-Computer Interaction (HCI) research. They fall under Creativity Support Environments (CSEs), which include both hardware and digital spaces and Creativity Support Systems (CSS), which enhance creative processes in fields like product design and research. Modern creativity research, sparked by Guilford's seminal address, has evolved through waves of focus on collaboration and digitization.

Current Research

This section explores related work in the field of creativity support tools (CSTs) and Human-Computer Interaction (HCI), examining various studies and frameworks that have shaped the understanding of these tools.

The most recent overview in the field is given by Stefanidi et al. (2023). They provide a comprehensive analysis of HCI literature reviews, offering a conceptual basis for authors, reviewers and readers. Their analysis spans multiple domains, including user experience and design, HCI research, interaction design and children, AI and machine learning, games and play, work and creativity, accessibility, well-being and health, human-robot interaction, automotive user interfaces (AutoUI), specific application areas and specific sensory inputs [14]. This extensive categorization underscores the multidisciplinary nature of CST research and its broad applicability across various fields.

Looking more into the field of CSTs, Frich et al. (2019) observe a significant trend in the complexity of CSTs over time. Their study shows a decrease in high-complexity CSTs, which were prominently represented in literature in 1999 (100%), 2002 (50%), 2008 (20%), 2009 (17%) and 2014 (7%). In contrast, low-complexity CSTs have seen increased representation since 2007, while medium-complexity tools have maintained consistent representation. This trend highlights the evolving nature of CSTs and their adaptation to user needs and technological advancements [7]. Their review of 143 papers from the ACM Digital Library (1999-2018) emphasizes the foundational work of Fischer and Shneiderman, who recognized the potential of computers to enhance human creativity. Frich et al. (2019) call for consistent evaluation methods, better integration of creativity research, a balanced focus on both novice and advanced tools and a clear, consensus definition of CSTs. The Creativity Support Index (CSI) is noted as a significant step towards standardized evaluation [7].

This evaluation method, the CSI, evaluates the ability of CSTs to assist users in creative work through dimensions such as exploration, expressiveness, immersion, enjoyment,

results worth effort and collaboration. It is a questionaire, inspired by the NASA Task Load Index, which results in a score on a scale of 100. The result can be interpreted like a school grade, meaning above 85 is still very good while everything under 50 is perceived as an F grad, insufficient. The CSI represents a significant advancement in the standardized evaluation of CSTs [6]. Cherry (2014) also notes that CSTs are often perceived as productivity tools, valued for their functionality, integration into existing workflows, performance, user interface, learning support, costs and emotional connection.

Shneiderman (2002) established a basis of design "Tasks" that developer can use to improve their software. It outlines a comprehensive framework of activities for creative work: collect, relate, create and donate. This framework emphasizes the non-linear path of creative activities and the importance of each stage in fostering creativity [8].

Gerber and Martin (2012) contribute to the field by offering design principles for creativity support in web-based product customization. Their work highlights the importance of tailoring CSTs to specific application contexts to maximize their effectiveness [15]. Wang (2017) identifies three latent variables for supporting creativity: playfulness, comprehension and specialization. Playfulness encourages trialability and iterative creation, comprehension facilitates quick understanding of ideation artifacts and specialization provides task-specific support and re-use through selection and arrangement [10]. Despite domain-specific idiosyncrasies, researchers believe that some principles and components are common across different creativity support systems.

Wang (2017) also notes that idea finding is the most commonly supported stage in CSTs, while problem finding is the least supported. This observation aligns with the broader understanding that creativity theories often suggest multiple stages in the creative process, yet single-user CSTs typically support only one or two stages [10].

Abrams (2002) introduces a novel system with several unique features designed to support musical composition. These features include a free-form "idea space," a configurable main workspace, an idea capturing facility, a workflow tracking mechanism and the ability to create various relationships among musical elements. Such systems highlight the potential for CSTs to provide flexible and adaptive environments for creative work [12]. Key theories and models in the field include the componential theory, dual pathway to creativity and associative memory theory. These theoretical frameworks underpin empirical studies and findings that further enhance our understanding of CSTs. "QSketcher" which is the name of the system also provides an "Infinite Vault". It keeps a history of all the changes and therefore allows user to tinker more and go back to follow up on previous ideas.

Resnick (2005) focuses on "composition tools" for generating, modifying, interacting and playing with creative content. These tools enable users to explore different creative possibilities and iterate on their ideas, thereby enhancing the creative process [9]. Kosof

(2021) proposes a model for tabletop dungeon design to aid dungeon masters in developing engaging roleplaying games. This model draws parallels to educational frameworks, illustrating how CSTs can support structured and goal-oriented creative activities [16].

Palani et al. (2022) investigate the values held by creative practitioners when adopting CSTs. Their study combines empirical observations from YouTube videos, interviews and survey responses to develop a conceptual framework of these values. This research provides valuable insights into the factors that influence the adoption and use of CSTs in creative practice [17]. Shneiderman (2007) outlines several design principles for CSTs: support exploratory search, enable collaboration, provide rich history-keeping and design with low thresholds, high ceilings and wide walls. These principles aim to make CSTs accessible to novices while offering advanced functionality for experts, thus supporting a wide range of creative tasks [11].

In conclusion, the field of creativity support tools (CSTs) and Human-Computer Interaction (HCI) is marked by its multidisciplinary nature and evolving complexity. The comprehensive analyses and frameworks provided by researchers like Stefanidi et al. (2023) and Frich et al. (2019) highlight the broad applicability and adaptive nature of CSTs. The development of standardized evaluation methods, such as the Creativity Support Index (CSI) and the emphasis on tailored design principles underscore the importance of different approaches in enhancing creative processes. As CSTs continue to evolve, they hold significant potential to support diverse creative activities across various domains, fostering innovation and productivity

2.2.2 Design Principles

Looking at the works of CSTs, there have been many design principles and ideas to improve a software or system. This section will got into the details of those principles and give a list for developers to design their work in a way to spark creativity and serve as a background for a feature list for the dice simulation app.

In his seminal work, Shneiderman (2002) outlines eight key tasks—**Searching, Visualization, Relating, Thinking, Exploring, Composing, Reviewing and Disseminating**—that serve as foundational design principles for creativity support tools. These tasks, while not an exhaustive or perfect set, provide a valuable checklist for analyzing existing software and guiding the design of new tools aimed at enhancing creative processes. While going through each of these base principles similar ideas will be added from related work.

Searching is the starting point of the creative process, enabling users to gather the necessary information, resources and expertise. Effective search capabilities are crucial for accessing relevant data, previous work and potential collaborators, thereby laying

the groundwork for innovation. Hewett et al. (2005) further highlight that specialized search tools, tailored to different media types or specific user needs, can significantly enhance the creative process by providing users with targeted and efficient access to the data they require. These tools act as a gateway to the vast amount of information available, enabling users to make informed decisions and generate novel ideas based on a comprehensive understanding of the existing landscape. By incorporating advanced search functionalities into creativity support tools, developers can ensure that users are equipped with the resources they need to push the boundaries of their creative endeavors [18].

Furthermore, effective exploration tools should inspire users by presenting relevant and complete solutions that can serve as a springboard for further creative work. This approach not only supports divergent thinking—where users generate a broad range of ideas—but also helps in convergent thinking, allowing users to refine and select the best options [10].

Visualization plays a vital role in organizing knowledge, identifying relationships and spotting gaps. By transforming abstract data into visual formats, users can better understand complex information, draw connections and structure their thoughts more effectively [8]. Abrams (2002) also highlights the importance of providing users with meaningful ways to manipulate content, such as through structured or improvisational approaches. By supporting both structured workflows and spontaneous creativity, visualization tools can cater to different work styles and help users develop and refine their ideas. Moreover, the ability to capture, organize and manipulate information visually not only aids in the creative process but also ensures that no valuable content is lost, as all captured data can be stored and accessed later for further development [12].

Relating refers to the facilitation of communication and collaboration among users. Whether through email, chat or more specialized tools, the ability to easily share ideas and clarify requests is essential for fostering creative exchange and protecting intellectual property [8]. According to Cherry and Latulipe (2014), the Creativity Support Index (CSI) emphasizes the importance of such tools in enabling collaboration, noting that a system should allow others to work with the user easily and share ideas effortlessly within the tool. When users feel that their creative efforts are respected and that the tools they use facilitate meaningful interaction, the result is often a more productive and innovative collaborative process [6].

Thinking, particularly through brainstorming and lateral thinking, is central to creativity. This task encourages exploring multiple possibilities and breaking free from conventional mindsets, allowing users to generate a diverse array of ideas [8]. However, supporting creative thinking in software design goes beyond just enabling idea generation; it requires tools that cater to different thinking styles and creative processes. As Abrams (2002) suggests, creativity tools should support both "structured noodling" and formal construction, recognizing that creators have diverse work styles, each requiring

different levels of structure and improvisation. Tools that facilitate concurrent activities, allowing for both planned and unplanned exploration, can help users develop ideas in a more fluid and dynamic environment.

Finally, the flexibility to manipulate and organize ideas meaningfully is key to supporting creative thinking. Creativity tools should allow users to rapidly explore different ideas, experiment with various approaches and structure their thoughts into a coherent whole. This means providing users with the ability to capture, organize and relate information across different tasks, whether they are conducting research, creating a presentation or developing a program. By supporting a wide range of projects and thinking styles, these tools can effectively foster the cognitive processes essential for creativity [12, 9].

Exploring in the context of creativity support tools involves conducting thought experiments and simulations to understand the implications of various decisions. This process allows users to test different scenarios, observe outcomes and refine their ideas safely and effectively [8]. The essence of exploration lies in the ability to notice and utilize non-obvious features of problem elements, which often leads to innovative solutions [10]. For a creativity support tool to be truly effective in facilitating exploration, it should be evocative, adaptive and open-ended. This means incorporating features such as visualization and abstraction to help users see beyond the obvious, customization and automation to tailor the experience to individual needs and peer-production and rejuvenation to keep the creative process dynamic and fresh [15]. The design of such tools should also support "structured noodling" a concept where users can engage in both planned and spontaneous creativity, experimenting with ideas in a way that feels natural and unforced [12].

Additionally, exploration tools should be intuitive and trustworthy, making it easy for users to try out different alternatives and backtrack when necessary. Features like a robust Undo function are crucial, as they encourage users to take creative risks without the fear of losing their progress.

Composing in creativity support tools involves creating and refining various media, such as documents, music, graphics or other digital content. Effective tools facilitate both the creation process and the iterative refinement crucial for high-quality output. As Shneiderman (2002) notes, intuitive tools enhance users' creative potential by making it easier to bring ideas to life. However, creators have diverse needs and no single approach suits everyone [8]. Abrams (2002) emphasizes that tools must support different work styles, from unplanned improvisation to formal construction, allowing for spontaneous creativity and systematic development [12].

Reviewing is a crucial part of the creative process, allowing users to revisit, edit and refine their work. This iterative approach fosters continuous improvement and learning, helping creators reflect on their progress and build upon existing ideas. History-keeping tools are invaluable, as they record activities and provide a detailed account of the

creative journey, ensuring no valuable content is lost. A well-designed system should automatically capture all creative content with annotations and timestamps, preserving even the most spontaneous ideas for future use. These tools support both structured and unstructured creative processes, catering to diverse work styles [8, 12].

Moreover, history-keeping tools should facilitate easy backtracking, which is essential in creative workflows where experimentation is key. The ability to try different alternatives, compare outcomes and return to previous versions without losing progress fosters innovation. These tools must be trustworthy and intuitive, encouraging users to experiment freely without fear of irreversible mistakes [15]. Additionally, they enhance collaborative work by enabling the sharing of tasks, annotations and progress among users. This comprehensive framework supports complex projects, ensuring every step of the creative journey is documented and accessible for future reference [12, 9].

Additionally, preserving all captured content is crucial; nothing should ever be lost. By implementing features like an "Infinite Take Vault', where content is automatically annotated and stored, users can ensure that their ideas are preserved in their original context, ready to be revisited or further developed at any time.

Finally, **Disseminating** is the task of sharing the final product with a broader audience. Effective dissemination tools ensure that creative works reach the right people, whether through direct communication or more public channels, thereby maximizing the impact of the creative endeavor [8].

The principle of "Low Threshold, High Ceiling and Wide Walls" is a critical design philosophy in the development of creativity support tools. This principle emphasizes the importance of creating tools that are accessible to novices, yet powerful enough to meet the needs of experts, while also supporting a broad range of creative activities. A low threshold ensures that beginners can easily engage with the tool, minimizing the learning curve and allowing them to start creating quickly. This is crucial in fostering initial engagement and encouraging users to explore the tool's capabilities without feeling overwhelmed [9, 11].

On the other hand, a high ceiling is essential for allowing more experienced users to push the boundaries of their creativity. Tools designed with a high ceiling provide advanced features that enable experts to work on sophisticated projects, offering depth and complexity without imposing limitations. This flexibility is vital for sustaining long-term engagement and ensuring that the tool remains relevant as users' skills and creative ambitions grow [15].

The concept of wide walls complements this approach by ensuring that the tool can support a diverse array of projects and creative endeavors. Wide walls mean that the tool is versatile enough to accommodate different styles of creativity—whether users are engaged in exploratory programming, music composition, graphic design or another

creative field. The ability to combine general primitives or elements in novel ways is a key aspect of this versatility, allowing users to apply the tool across various domains and tasks [18].

Collaboration is a fundamental aspect of most creative endeavors, particularly in environments where diverse teams come together to combine their unique strengths. As Resnick (2005) highlights, in both educational settings and the professional world, creative work is often accomplished in teams, where the diversity of talent is not just common but essential. Effective creativity support tools must therefore facilitate collaboration by allowing each team member to contribute according to their expertise, whether it be in art, sound design, scriptwriting, programming or any other discipline [9].

Furthermore, a robust collaboration tool should include features that allow for the seamless sharing and manipulation of content among team members. For example, systems that allow content to be annotated and preserved in its original context ensure that no idea is lost and that the creative process can be revisited and refined as needed. These tools should also support the organization and relationship of ideas, links, documents and other resources across different tasks, fostering a collaborative environment where information can be easily accessed and shared. Cherry and Latulipe 2014) emphasize that a good collaboration tool should make it easy for others to work together, share ideas and ultimately produce results that are satisfying and worth the effort. By incorporating these features, creativity support tools can effectively enhance the collaborative process, enabling teams to create more innovative and cohesive work together [12, 6, 15].

Open Interchange is crucial for creativity support tools, allowing users to easily import and export data from conventional tools, thus integrating seamlessly into existing workflows. This openness is vital in environments where creators use various tools for different project aspects. For instance, transferring data from a spreadsheet to creative software can streamline the creative process. By supporting interoperability, these tools cater to users working across multiple platforms simultaneously [9].

Additionally, Open Interchange involves the extensibility of tools. Customizing and extending tools through plug-in architectures or open data models is essential for meeting users' evolving needs. This flexibility enhances tool capabilities by adding functionalities tailored to creative workflows. For example, in music composition, importing external sound libraries or exporting compositions into various formats enhances utility, enabling users to experiment and structure musical fragments effectively [12].

Simplicity in creativity support tools is a crucial design principle that fosters creativity by reducing complexity and balancing essential features with ease of use. When tools are simple and intuitive, they lower the cognitive load on users, enabling them to focus more on the creative process rather than the mechanics of the tool itself. This is particularly important in environments where users need to explore multiple alternatives, experi-

ment with different ideas and backtrack when necessary. A well-designed tool should be trustworthy and self-revealing, allowing users to try things out with confidence, knowing that they can easily undo actions and refine their work. If creativity tools are hard to discover, use or access when needed, they provide little benefit. Therefore, it is essential that these tools are intuitive and support the user's creative flow without imposing unnecessary complexity. Simplicity, therefore, not only enhances usability but also encourages creative exploration by making it easy to sketch out different alternatives and iterate on ideas [9].

Moreover, reducing unnecessary features can paradoxically lead to greater creative potential. By focusing on the essential elements that truly support the creative process, tools can avoid overwhelming users with options and instead provide a streamlined experience that fosters both structured and unstructured creativity. As Resnick (2005) notes, simplifying tools can improve the user experience by removing constraints that might otherwise hinder creativity. This approach aligns with the need to support diverse work styles, from structured noodling to formal construction, ensuring that the tool is accessible and effective for all users, regardless of their experience level. In this way, simplicity becomes a powerful enabler of creativity, allowing users to engage deeply with their work without being distracted by unnecessary complexity [18, 12].

Another helpful principle is the selection of the "primitive elements" that users will manipulate. These foundational components or "black boxes" must be chosen thoughtfully, as they form the building blocks of the creative process. Even when tools offer the same level of capability, the way these elements are presented can significantly impact usability and creativity. There are simpler and more complex ways to implement these functions and the goal should always be to enhance the user's experience without overwhelming them. Balancing user suggestions with careful observation and participatory design processes is essential in this regard. While user feedback is invaluable, it is important to recognize that not all requests may be practical or feasible. Therefore, an iterative design approach—one that involves continuous testing, refinement and user engagement—is crucial. This ensures that the tools evolve in a way that truly supports creativity, offering intuitive and powerful elements that users can easily understand and manipulate [9].

Minimizing cognitive load is a fundamental principle in the design of CSTs, aimed at reducing the mental effort required for users to engage with the tool effectively [15]. By designing interfaces that are intuitive and user-friendly, developers can ensure that users can focus on the creative process rather than struggling with the complexities of the tool itself. Essential features should be clearly presented and easy to access, avoiding unnecessary complexity that could overwhelm or frustrate users. For instance, tools like Adobe Photoshop offer robust functionalities but also provide intuitive interfaces that allow users to experiment with different effects and revert to previous states as needed. This balance between functionality and simplicity is crucial for enabling users to explore creative possibilities without being hindered by a steep learning curve [18].

Moreover, seamless integration with other tools and workflows is key to minimizing cognitive load. Creativity support tools should not operate in isolation but should integrate smoothly with the other software and devices that users rely on in their creative process. This integration reduces the need for users to constantly switch contexts or learn new systems, allowing them to maintain their focus on the task at hand. Additionally, providing ample learning resources, such as tutorials and documentation, can further reduce cognitive load by helping users quickly become proficient with the tool. When interfaces are designed with simplicity and ease of use in mind, users are more likely to have a positive experience, which not only enhances their creative output but also fosters a sense of satisfaction and emotional connection with the tool [17].

Informed participation is a key principle in the design of creativity support tools, emphasizing the importance of providing users with the resources and support they need to understand and effectively utilize these tools. To maximize the potential of a creativity support tool, it is crucial that users are not only provided with essential features but also with the educational resources necessary to master them. This includes comprehensive tutorials, detailed documentation and accessible learning aids that cater to users at different skill levels. For example, software companies like Adobe offer extensive online tutorials and documentation to help users navigate complex tools such as Photoshop, enabling them to fully leverage the tool's capabilities without feeling overwhelmed [18].

Additionally, informed participation ensures that users can seamlessly integrate new tools into their existing workflows, thereby enhancing their creative process. When users are well-supported through educational resources, they can more easily adapt to new tools, minimizing the friction that often accompanies the learning of new software. This support is especially important in environments where tools need to integrate smoothly with other software and devices that users rely on. By providing resources that help users understand not just how to use a tool, but how to integrate it into their broader creative practices, developers can foster a more effective and satisfying user experience. In this way, informed participation becomes a critical factor in the successful adoption and use of creativity support tools, ensuring that users feel confident and empowered in their creative endeavors [17].

These principles can serve as a guide for developers creating creativity support tools. They will also serve as a guide for the development of the dice simulation app and to better work with them, a list containing all the principles with a short name and explanation was created (table 2.2)

Design Principle	Explanation
Searching	Facilitate efficient information gathering to access relevant resources and experts.
Visualization	Utilize visualization tools to organize knowledge, identify relationships and spot gaps.
Relating	Support clear and protected communication for collaborative creativity.
Thinking	Encourage brainstorming and lateral thinking to explore multiple possibilities.
Exploring	Enable thought experiments and simulations to explore implications of decisions safely.
Composing	Support creation of various media forms, facilitating easy composition and refinement.
Reviewing	Include history-keeping tools for reviewing, editing and storing activities.
Disseminating	Provide robust dissemination options to share creative products with broader audiences.
Low Threshold, High Ceiling, Wide Walls	Design tools to be easy for novices to start (low threshold), allow experts to work on sophisticated projects (high ceiling) and support a wide range of projects (wide walls).
Support Collaboration	Facilitate concurrent user activities and collaborative work styles.
Open Interchange	Ensure easy import and export of data from conventional tools and support extensibility.
Simplicity	Reduce complexity to foster creativity, balancing essential features with ease of use.
Choose Black Boxes Carefully	Select primitive elements that users manipulate thoughtfully, balancing user suggestions with participatory processes.
Minimize Cognitive Load	Design interfaces that are intuitive and reduce the mental effort required to use the tool.
Informed Participation	Provide resources and support to help users understand and effectively utilize the tools.

Table 2.2: Design Principles for Creativity Support Tools

2.2.3 Notable Features for a Dice Simulation App

Building upon the design principles outlined in the preceding section, potential features for a mobile dice simulation app aimed at stimulating creativity can be explored. This

section will discuss relevant features derived from these design principles.

One of the primary features that can significantly enhance the app's utility is Resource Library Integration. According to the principle of searching, integrating a comprehensive resource library within the app will allow users to access a wealth of images, sample dice designs and other resources quickly. This library can be continually updated with new content, ensuring users have a rich repository to draw inspiration from. Additionally, integrating expert resources, such as tutorials and design tips from experienced tabletop game developers, can further empower users by providing them with valuable insights and knowledge.

To support identifying relationships and organizing knowledge, a Group Visualization feature could be considered. This feature allows users to group multiple dice together and visualize how they interact when rolled simultaneously. Users can create different groups, assign colors or states to each die and observe the combined outcomes. This visualization aids in understanding the interactions between different dice and identifying any patterns or gaps in the design. For example, in a board game where players must roll specific combinations to unlock abilities or progress through levels, developers can use this feature to test different dice groupings. By simulating various rolls, they can see how likely players are to achieve the desired combinations, ensuring the game is both challenging and fair. By providing a clear visual representation of grouped dice, this feature supports users in making informed design decisions.

Collaboration Features are essential for fostering clear and protected communication, as highlighted in the design principles. In a creative field like tabletop game development, collaboration tools within the app could allow users to share their designs with team members, receive feedback and co-create in real time. More specific solutions could be a general chat feature, dice and dice groups could be made sharable between designers, so that others can add comments and co-design. Users could discuss design ideas, make collective decisions and refine their projects together.

Idea Generation Tools are fundamental for encouraging brainstorming and lateral thinking. These tools can include mind maps, random idea generators and prompts to stimulate creative thinking. By incorporating such features, the app can help users explore multiple possibilities and break out of conventional thinking patterns. For example, a random idea generator can provide unexpected combinations of images, inspiring users to consider new design directions they might not have thought of otherwise. A note taking functionality could be set to quickly write down thoughts for a dice or group of dice.

To enable thought experiments and simulations to explore the implications of decisions a Simulation Mode could be thought of. This mode would simply be the rolling environment, but instead of just rolling dice one can make changes to these dice. For example edit the dice or add and remove other dice.

An Activity Log is a crucial feature for history-keeping and reviewing. This feature records all activities, edits and versions of dice designs, providing users with a comprehensive history of their work. Users can review past actions, compare different versions and restore previous states if needed. This feature supports reflective practice, allowing users to learn from their past decisions and improve their designs iteratively. The Activity Log acts much like the back button on a web browser, giving users the confidence to explore various options without the fear of losing their progress. An undo Button would be making use of this activity log.

Sharing Options are essential for robust dissemination of creative products. Derived from the principle of providing robust dissemination options, this feature allows users to share their dice designs and simulation results with a broader audience. Users could be able to click a share button on a dice or dice group design that would send it directly to social media platforms, send via email or upload to the app's community. By facilitating easy sharing, this feature encourages collaboration and feedback, enabling users to showcase their work, receive constructive criticism and gain inspiration from others. Sharing Options also help users disseminate their creative products to potential players or collaborators, enhancing the reach and impact of their designs. In the app a share option could be set for dice, dice groups and images, too.

Conversely, the export function should enable users to save and share their designs in various formats, making it easy to integrate with other tools and platforms. For instance, users can export their dice designs as image files or video sequences to present in other software or share on social media. Other common forms of export could be JSON, CSV or XML. This feature ensures that users can leverage a wide range of media types in their creative process, fostering innovation and creativity.

Easy import and export of data closely relates to the dissemination functionality. One could export dice or dice groups and send it to somebody who could import the data. At best this export format is common to be used with other tools as well. Something like JSON format.

To set a low threshold an easy layout for novice users can be imagined. As well as tutorials or guides through the app. These tutorials guide new users through the app's features and functionalities, helping them get started quickly and effectively. Beginner Tutorials can include step-by-step instructions, video guides and interactive walkthroughs that demonstrate how to create and roll dice, import images and use various other tools. This way the app ensures that even users with no prior experience in tabletop game development can begin creating dice designs with confidence. This feature reduces the learning curve and makes the app accessible to a broader audience.

To also ensure a high ceiling advanced Tools specifically made for tabletop development could be added to give value for experienced users. These tools might include custom scripting capabilities to make changes to a configuration after a dice roll or other events.

Advanced Tools enable users to push the boundaries of their creativity and develop highly detailed and unique dice designs. By offering advanced functionalities, the app can support a wide range of creative needs, from simple designs to complex, professional-level projects.

Versatile Tools support the principle of wide walls, ensuring that the app can accommodate a broad spectrum of creative projects. These tools should be flexible and adaptable, allowing users to work on various types of dice designs and gaming scenarios. For that it is important not to limit a user for how many images a dice can have or color or number. Versatile Tools ensure that users can tailor the app to their specific needs, whether they are designing dice for a fantasy role-playing game, a board game or an educational tool.

A Streamlined Interface, aligned with the principle of reducing complexity to foster creativity, focuses on creating a clean, intuitive and user-friendly interface. The goal is to balance essential features with ease of use, ensuring that users can access all necessary functionalities without being overwhelmed by a cluttered interface. This might involve simplifying menus, providing clear icons and labels and organizing features logically. By reducing the cognitive load required to navigate the app, a streamlined interface helps users focus more on their creative tasks and less on figuring out how to use the tool. This feature is especially important for maintaining user engagement and satisfaction. In more specific words, a User-friendly design in the app would be a central navigation and an onboarding screen that showcases different use cases of the app

User Experience Design, which focuses on creating interfaces that are intuitive and reduce mental effort, emphasizes the overall experience of using the app. This involves designing every aspect of the app—from the layout and navigation to the interactions and feedback mechanisms—with the user's comfort and efficiency in mind. For example reoccurring functionalities like holding actions that work the same to manage dice and dice groups could be seen as intuitive, as well as adding a die or a dice group uses the same flow.

With these ideas a list of features can be created for a dice simulation app that supports creativity 2.3. Because of limited resources and time the next section will explain which features will be chosen for the implementation of the dice simulation app.

Feature	**Description**	**Design Principles**
Resource Library	you can search for images, dice designs and design tutorials within the app	Searching, Relating
Group Visualization	you can test different designs simultaneously by rolling multiple dice	Visualization
Co-design	you can work on dice configurations with others	Support Collaboration
Add Comments	you can add comments to others work	Support Collaboration
Chat	you can chat with others	Support Collaboration
Brainstorm	you can add notes to your designs	Thinking
Random Config	you can generate random die configurations	Relating, Exploring
Simulation mode	you can roll a configuration and make changes to it	Exploring
Activity Log	changes made to a configuration will be saved and the user can come back to them	Exploring, Reviewing
Undo Button	you can undo changes in an active configuration	Exploring
Share Designs	you can share configurations over social platforms and email	Support Collaboration, Disseminating
Import / export	you can export data as JSON and import data from JSON file	Open Interchange
Beginner tutorial	you can find tutorials and guides through the app	Low Threshold, Informed Participation
Scripting	you can set rules for certain events in a configuration that will change the active dice	Composing, High Ceiling
Versatile Tools	you can create a wide range of die configurations	Wide Walls, Composing
User-friendly design	there is a central navigation and an onboarding screen	Simplicity, Choose Black Boxes Carefully, Low Threshold
Intuitive design	managing dice and dice groups uses the same functionality for adding, editing, deleting and duplicating	Minimize Cognitive Load Simplicity

Table 2.3: Feature list for dice app to support creativity

2.3 Feature selection

Given the limited resources available, it is essential to prioritize features that will have the most significant impact on user creativity support. After careful consideration, the focus is on the following features: **Group Visualization, Random Config, Versatile Tools, Simulation Mode, User-Friendly Design and Intuitive Design**. This section explains why these features were chosen.

Group Visualization has already been part of the design to roll more dice in parallel with dice groups. It is also easily combinable with the simulation mode which allows users not only to roll different configurations, but also make changes to them. So a user can add or remove dice in the rolling screen, quickly testing different ideas. A random dice configuration is a very easy implementation to get the user to relate and get inspired by other designs and functionalities of the app. It will help with exploring new ideas. Versatile tools are the root of the app, it should not limit the user in his creative ideas and ensure a variety of possibilities for all kind of dice games. By focusing on an intuitive and userfriendly design there is less of a need for tutorials and ressources covering the low threshhold by mere design while also making the overall experience for experienced users better. While collaboration features such as Co-design, Chat and Add Comments offer valuable functionality, they are most effective when integrated into a comprehensive collaboration suite. However, given our focus on delivering a streamlined and efficient tool within the current resource constraints, these features are less critical. Additionally, the tabletop game development often is done in a solo capacity, which would make the features useless for a big part of users.

Moreover, while features like Import/Export, Share Designs and Activity Log provide additional convenience, their value is lessened in the absence of external data sources or collaborative needs. Instead, we provide a simpler solution through the ability to save and duplicate dice and groups, offering users a way to preserve their work without the complexity of a full activity log or undo button. Similarly, while not a full ressource library, the already present "selection of dice" feature gives a small selection of work to relate with.

In conclusion, the selected features—Group Visualization, Random Config, Versatile Tools, Simulation Mode, User-Friendly Design and Intuitive Design—represent a strategic balance between functionality, user experience and resource efficiency. These features will allow us to deliver a powerful, user-centered app that meets the core needs of tabletop game developers while staying within the resource limitations. With this the feature list for the implementation is complete (table 2.4) and an app can be created. The next chapter will present my implementation of these features as well as architecural decisions and limits of the app

Feature	**Description**
Multiple dice	You can roll more than one die at the same time
Selection of dice	The app features a variety of dice options to choose from (for example dice with 4, 6, 12, 20 faces)
Custom Number range	The app featured more than just a selction, you can define the exact number of faces for a dice
Images	You can set images as the face of a die
Sum	After rolling multiple dice you see the sum of the dice
Different colors	Dice can have different colors
Re-rolling	The selection of dice can be rolled again
Rolling history	You can see what dice and their faces were rolled
Adding dice	Dice can be added to the current selection of dice
Removing dice	You can remove a die from the current selection of dice
Locking	From the selection of dice you can choose which dice to roll
Roll single	You can roll a single die of the selection
Presets	You can create a group of dice to quickly select a custom configuration
Group Visualization	You can test different designs simultaneously by rolling multiple dice
Random Config	You can generate random die configurations
Versatile Tools	You can create a wide range of die configurations
User-friendly design	there is a central navigation and an onboarding screen
Intuitive design	Managing dice and dice groups uses the same functionality for adding, editing, deleting and duplicating
Simulation mode	You can roll a configuration and make changes to it

Table 2.4: Complete Feature list for dice app

3 Solution

This chapter introduces the solution, focusing on the design and implementation of a configurable dice rolling app intended to support tabletop game developers. The chapter covers the architectural decisions, implementation details, key features and potential limitations of the app.

3.1 Architecture

This section provides a high-level overview of the architecture of the app and also explains the decisions made against the Server sided Image integration.

Firstly, the app was developed in android. This was due to its popularity in the market. Therefore there were a lot of helpful documentation for development and tools and libraries. It has been chosen over iOS due to ecosystem familiarity and because learning different platforms would be unnecessarily time intensive. This is also why this app was not developed using hybrid frameworks (like React-Native, Flutter Iconic), despite it being beneficial to the app functionality. Because using a hybrid framework would allow the app to work on IOS, Android and perhaps also as a web-app when using Ionic. This would allow more users to use the app and since the main native functionalities are uploading images, there is no direct need for any specific device or platform. Regardless, a significant amount of users can benefit from this tool.

For programming android apps, really only Java and Kotlin come into consideration. Kotlin was selected as the primary programming language for this project due to its growing prominence in the Android development community. Officially endorsed by Google as the preferred language for Android development, Kotlin offers a modern, expressive syntax that enhances developer productivity. Its seamless interoperability with Java allows developers with java experience to leverage their skills while benefitting from the concise syntax that reduces boilerplate code, making the codebase more readable and maintainable, which is particularly beneficial in large or complex projects.

Another significant advantage of Kotlin is its robust type system, which includes null safety as a core feature. This reduces the risk of null pointer exceptions, a common

source of runtime errors in Java applications. By enforcing null safety at the language level, Kotlin allows developers to write safer and more reliable code. Additionally, Kotlin integrates well with Jetpack Compose, Android's modern toolkit for building native UIs and Proto Datastore, a data storage solution. This makes Kotlin an excellent choice for Android projects that prioritize both performance and developer experience.

Kotlin's endorsement by Google for Android development further reinforced the decision, as it signifies strong and ongoing support from the Android community and ensures that Kotlin will continue to receive updates and improvements tailored specifically for Android development. The decision was also influenced by the need to work effectively with Jetpack Compose and Proto Datastore. Kotlin's design makes it particularly well-suited for use with these technologies, enabling more streamlined and efficient development. The language's concise syntax improves the developer experience by simplifying complex code structures and reducing the amount of code needed to accomplish tasks

Jetpack Compose is a modern UI toolkit introduced by Google for building native Android user interfaces. Its declarative approach marks a significant shift from the traditional imperative UI design methods, enabling developers to describe the UI's state and letting the framework manage the rendering process. This approach simplifies the creation of dynamic and responsive UIs, as developers can focus on defining the "what" of the UI rather than the "how." Jetpack Compose's tight integration with Kotlin further enhances this process, allowing for more expressive and concise code that directly represents the UI's state.

One of the standout features of Jetpack Compose is its ability to handle UI recomposition efficiently. Recomposition refers to the framework's capability to update only the parts of the UI that have changed, leading to a more responsive and fluid user experience. This is particularly beneficial in applications where the UI needs to reflect frequent state changes. Moreover, Jetpack Compose is equipped with robust debugging tools like the Layout Inspector and UI debugging with Preview, which allow developers to visualize and troubleshoot UI components in real-time, significantly improving the development workflow.

Jetpack Compose was chosen for this project due to its ability to provide an excellent user experience through efficient UI recomposition. This is necessary for the features as the user should be able to render and roll many dice and make changes the explore and tinker with ideas

Proto Datastore is a modern and lightweight data storage solution provided by Google, designed specifically for handling structured data in Android applications. Unlike traditional key-value storage systems, Proto Datastore allows for strongly typed data using Protocol Buffers, which are both space-efficient and performance-optimized. This ensures that the data stored is type-safe, reducing the likelihood of runtime errors and making the codebase more robust and reliable. Proto Datastore's integration with Jet-

pack Compose is particularly noteworthy, as it enables developers to directly interact with stored data using Compose's collectAsState function. This integration allows the UI to automatically react to data changes in the Datastore, creating a seamless connection between the app's state and its visual representation.

One of the main advantages of Proto Datastore is its performance. It is designed to be quick and efficient, allowing developers to work directly with the data stored in the Datastore without the need for intermediate steps or complex data transformations. This direct interaction streamlines the development process, enabling faster data retrieval and updates. Additionally, Proto Datastore is well-suited for applications that require persistent data storage with minimal overhead, making it an ideal choice for lightweight and responsive applications.

The advantages of Proto Datastore were also a reason why it has been decided not to do Server sided images. This idea came up, because loading images to the server might have been an easier solution, but practice showed that uploading images from the device and saving it there is as easy. Furthermore, this makes the app not rely on an internet connection, which makes it more reliable for the user and also easier to work with as a developer. Another advantage of using a Cloud based architecture would be the ease to share data which is part of the design principles to improve the creativity of the app. But as it has been decided to focus on other ideas, this advantage does not hold against the benefits from using internal storage.

3.2 Implementation Details

Entities

After presenting the high-level architecture in the previous section, this section delves into the detailed data storage and flow within the app, as illustrated in the class diagram (see Figure 3.1).

The app manages three primary data entities: Dice Group, Die and Image. Each of these entities is stored in its own Proto Datastore. The data flow begins with the Image entity, which is the first item a user interacts with. Users can upload multiple images, rename them and tag them as needed. These images are stored in the Image datastore as Base64-encoded strings.

Following the creation of images, users can proceed to create Die entities. During this process, each image is assigned a weight and value, forming a Face for the die. The Face entity contains a reference to the original Image entity, indicated in the class diagram

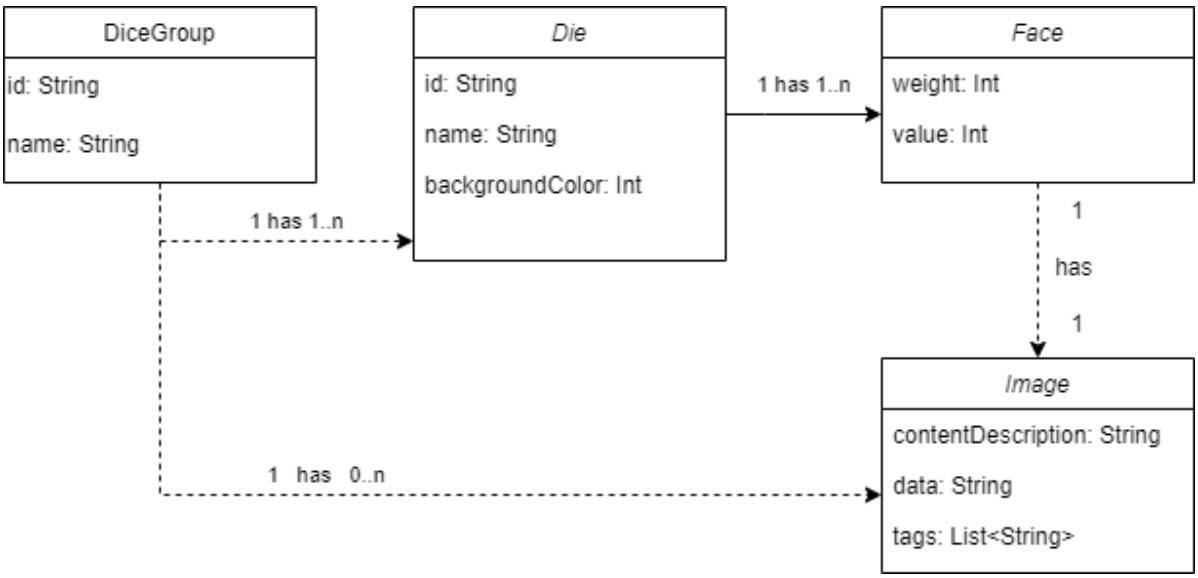

Figure 3.1: Proto datastore entries

by a dotted line, representing an indirect association through the image's identifier. The Die entity itself stores the Face entities directly, which is depicted with a continuous line in the diagram. Each die can have one to many faces, allowing for a diverse range of possibilities in dice configurations.

Finally, users can group multiple dice into Dice Groups, which are stored in the third datastore. A Dice Group can contain one to many dice and it may also include an additional list of images that represent different states a die can have during gameplay and a name.

Model-View-ViewModel

Building upon the detailed data storage and flow described in the previous section, the app's architecture is further strengthened by the implementation of the Model-View-ViewModel (MVVM) pattern. This pattern is integral to maintaining a clean separation of concerns within the app, ensuring that each component—Model, View and ViewModel—has a distinct and focused role. By dividing responsibilities in this way, the app becomes more modular, making it easier to develop, test and maintain.

The Model in this architecture represents the data layer, which includes the three Proto Datastores: Dice Group, Die and Image. These datastores are responsible for persisting the app's data, ensuring that Dice Groups, dice and images are stored reliably and are accessible across different sessions. The Model handles the business logic related to data storage and retrieval, abstracting away the complexities of data management from the

rest of the application.

The View is the user interface of the app, built using Jetpack Compose. It is responsible for displaying data to the user and handling user interactions. The View observes the data provided by the ViewModel and updates dynamically based on changes in the underlying data. Thanks to Jetpack Compose's declarative nature, the View can easily render complex and responsive UIs, ensuring that users have a seamless and intuitive experience as they interact with the app.

At the heart of this architecture lies the ViewModel, which acts as the bridge between the Model and the View. The DiceViewModel instance, specifically, serves as the app's single source of truth, managing the data flow and ensuring that the UI always reflects the current state of the underlying data. The ViewModel holds references to all three datastores and provides the necessary functions for modifying, adding and removing entities such as dice, dice groups and images. When a user performs an action, such as creating a new die group or rolling a die, the ViewModel handles the logic, updates the relevant datastore and emits the updated data back to the View.

This setup ensures a smooth and efficient data flow within the app. For instance, when the user clicks a roll button, the ViewModel will handle the change and the View observes this change through jetpack compose's state and automatically updates the UI to display the result. This reactive data-binding approach, facilitated by the MVVM pattern, not only simplifies the development process and maintainance, but also enhances the user experience by ensuring that the UI is always in sync with the latest data.

In summary, the MVVM pattern plays a crucial role in organizing the app's architecture. By clearly delineating the responsibilities of the Model, View and ViewModel, it promotes better code maintainability and scalability. The DiceViewModel serves as the linchpin of this architecture, coordinating the interactions between the UI and the underlying data, thus ensuring a responsive and cohesive application.

Jetpack Compose

To support the responsiveness, Jetpack Compose was used to render the UI. It is a modern UI toolkit that simplifies and accelerates UI development in Android applications through its declarative programming model. One of the key features of Jetpack Compose is its recomposition mechanism, which plays a crucial role in creating dynamic and responsive user interfaces. Recomposition is the process by which Jetpack Compose automatically updates only the parts of the UI where the underlying data has changed, ensuring that the UI always reflects the current state of the application. Importantly, components whose state has not changed remain untouched and are not re-rendered, which optimizes performance and maintains a smooth, responsive user experience.

In a typical UI framework, developers must manually manage the state of UI components, which can lead to complex and error-prone code. Jetpack Compose, however, eliminates much of this complexity by allowing developers to declare what the UI should look like based on the current state. When the state changes, Jetpack Compose triggers a recomposition, where it efficiently recalculates only the parts of the UI that need to be updated, rather than redrawing the entire screen. This leads to better performance and a more fluid user experience.

For instance, consider the function rollDice() (listing: 3.1) in the app, which is responsible for simulating the rolling of dice. The currentDice list is updated by mapping over each die and applying the roll() function to dice that are not locked. The roll() function modifies the die's state, which results in a new die object being created through the copy function. This updated list of dice is then reassigned to currentDice. Because currentDice is a stateful property observed by Jetpack Compose, updating it triggers recomposition. Compose detects that currentDice has changed and automatically recomposes the UI elements that depend on this list, such as the dice displayed on the screen. The recomposition ensures that only the dice that have been rolled are visually updated, while the rest of the UI remains unchanged. This efficient update mechanism keeps the UI responsive, providing immediate feedback to the user as they interact with the app.

Listing 3.1: Roll dice function

```
fun rollDice() {
  countRolls++
  currentSum = 0
  currentDice =
      currentDice.map { die ->
        if (die.dieLockState != DieLockState.LOCKED) {
          die.roll().also { currentSum += it.current?.value ?: 0 }
        } else {
          die.also { currentSum += it.current?.value ?: 0 }
        }
      }
  saveToHistory(countRolls, currentSum, currentDice)
}
```

In summary, Jetpack Compose's recomposition mechanism is a powerful tool for building dynamic and responsive UIs. It allows developers to focus on defining what the UI should look like based on the current state, while Compose handles the complexities of updating the UI when the state changes

3.3 UI and business logic

As the app is very UI orientated, This section explains the business logic of the app while working through a standard user flow of creating a die.

The first three Screens (figure 3.2) depict how the user can get started to creating a die. The Burger Menu shows the overall structure of the app. It has 5 main screens. "Roll" is the landing page, the screen where the user always starts when using the app. Each Screen receives data from the ViewModel which is directly linked to the datastore as well as callbacks to this ViewModel for changing the state and data of the app. The "Roll" screen gets a selection of the dice and other rolling states as well as functions to roll, lock and modify those states.

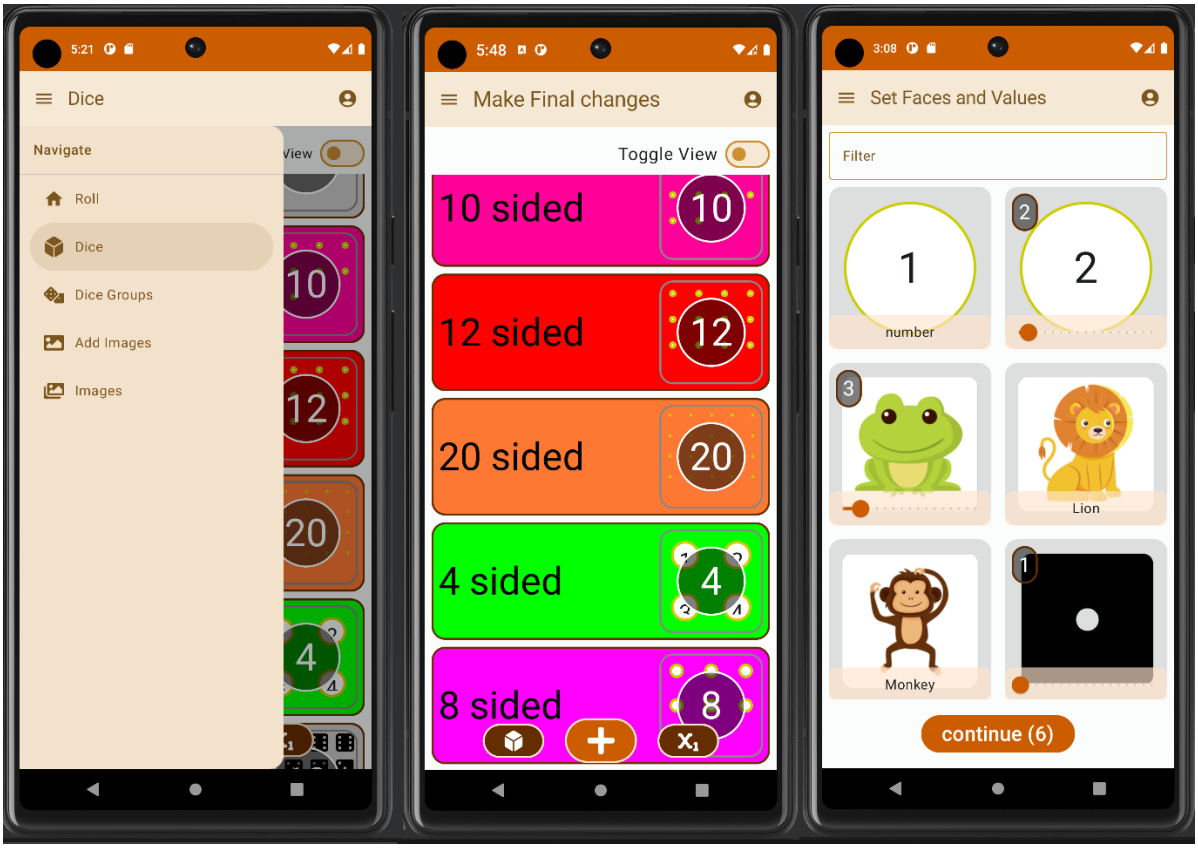

Figure 3.2: Create Die Flow

The next screen "Dice" is the place where all the dice from the datastore can be managed. This screen can be seen in the center of the figure. Some notable actions are the creation of a die, removing, editing or duplicating a die. "Dice Groups" is similar to "Dice". It uses the same components to display the Elements and to create new Elements. It is

a Screen to manage Dice Groups. "Add Images" is a Screen allowing the user to save images to the app that can be used to create dice and dice states. It only receives a callback to add images while "Images" is the place to delete images.

The second Image in figure 3.2 shows the "Dice" Screen and when clicking on the plus Icon the user will be navigated to the third Screen which is the "Face Selection" Screen. As mentioned this particular Component of selecting items and giving them some kind of value is used in several locations in the app. The function takes a list of selectables and some callbacks. One to display each selected item, allowing a wide range of UI layouts and another to handle the "Save Selection". There is also callbacks for handling an Item click or applying a filter from the Inputfield on top. Also importantly, this Component has a local state for the selection of items which will only be saved through actually saving the selection. Meaning, any navigation that happens before saving will discard the selection.

In Figure 3.3 the first screen shows what happens when clicking on save selection. The user receives an overview of the die that is being created and some adjustments can be made, like changing the name, color or weights. The weights, again, use the same component to select items. This time, the value is used to add a weight to each face which will influence the probability of this face being rolled. A face with the weight of 3 being rolled will be three times more probable than a face with the weight of one.

By clicking on a die in the "Dice" screen, a user can roll the die, which updates the state of the current dice and navigates to the "Roll" screen. This screen renders the current state as provided by the ViewModel. Users can interact with this state in several ways, the simplest being to roll the dice. Rolling the dice updates not only the dice selection but also the history and the dice sum. Jetpack Compose efficiently picks up on these changes and updates the UI accordingly. The third screen in the figure (3.3) displays a list of additional actions the user can take, further enhancing the interactivity of this screen.

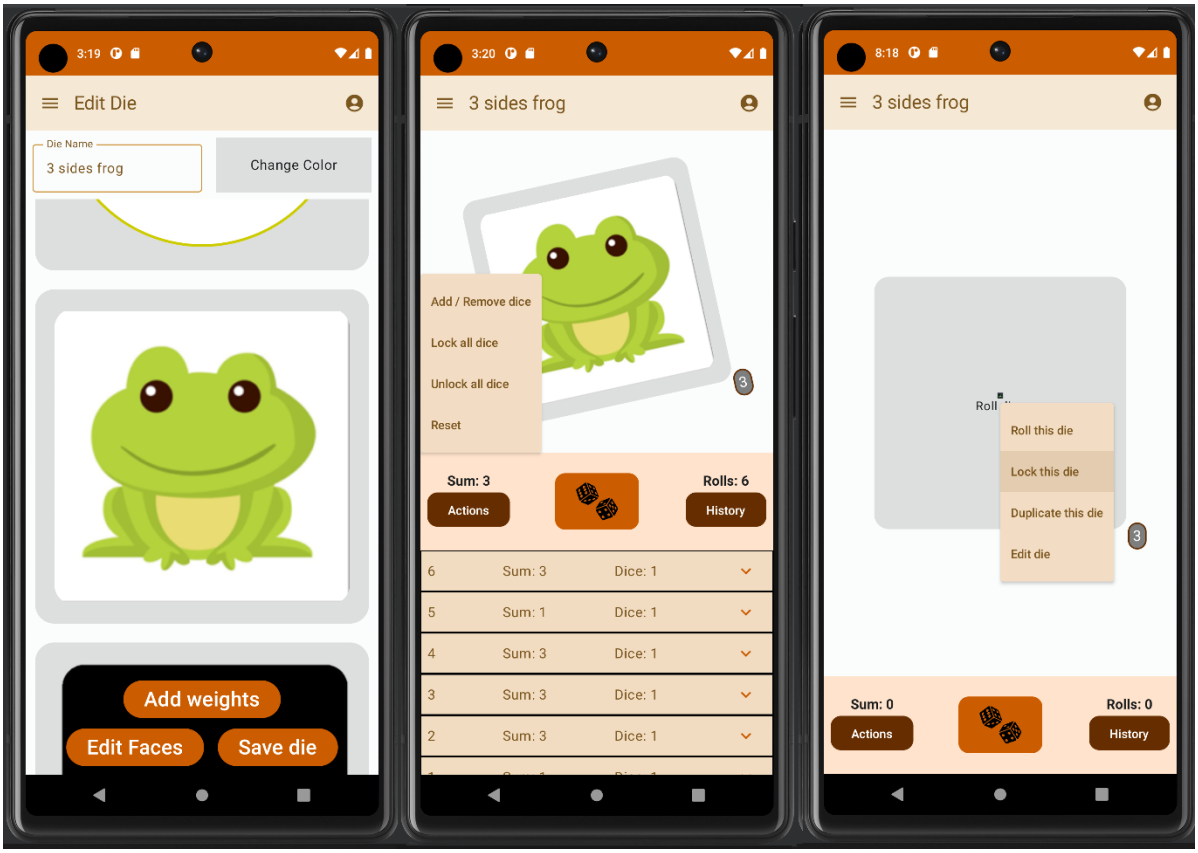

Figure 3.3: Roll Dice Flow

In this particular screen, the state is not directly bound to the datastore. While this approach offers speed and responsiveness, it might not always provide the best user experience for ongoing interactions. Instead, the ViewModel manages its own local state within this screen. To ensure that this state persists, any lifecycle events such as stopping, pausing or navigating away from this screen trigger a save operation. The selection of dice is saved as a temporary Dice Group, which can be managed and explored further in the Dice Groups section, without the risk of losing progress. This allows users to experiment with their dice configurations safely, knowing they can always revert or save their changes permanently if desired.

The OneScreenGrid component plays a crucial role in maintaining a dynamic and adaptable user interface (listing 3.2). This Jetpack Compose component has been refactored to be used in various parts of the application, demonstrating its versatility and efficiency. OneScreenGrid is designed to handle the display of items within a constrained space. It takes a list of items, a callback function for rendering each item and a specified minimum size for each item. The component then dynamically arranges these items on the screen, ensuring that as many as possible fit within the available space without shrinking below the defined minimum size.

Listing 3.2: Function OneScreenGrid

```
@Composable
fun <T> OneScreenGrid(
    items: List<T>,
    minSize: Float,
    modifier: Modifier = Modifier,
    onRender: @Composable (item: T, maxWidth: Dp) -> Unit,
) {
  BoxWithConstraints(modifier = modifier) {
    val density = LocalDensity.current
    val maxWidthPixels =
        getMaxGridWidth(
            items.size,
            containerWidth = constraints.maxWidth,
            containerHeight = constraints.maxHeight)
    val maxSize = floor(minSize.coerceAtLeast(maxWidthPixels))
    val maxWidthDp = pxToDp(density, maxSize)
    LazyVerticalGrid(columns = GridCells.Adaptive(minSize =
   maxWidthDp)) {
      items(items) { item -> onRender(item, maxWidthDp) }
    }
  }
}
```

The flexibility of OneScreenGrid makes it ideal for different use cases throughout the application. For instance, during the die creation process, it efficiently displays the images or die faces, adjusting the layout according to the screen's dimensions. Similarly, when managing dice groups or displaying actions on the "Roll" screen, OneScreenGrid ensures that the user interface remains responsive and visually organized. This component calculates the maximum possible width for each item based on the screen's size and ensures that the grid is rendered efficiently using Jetpack Compose's LazyVerticalGrid. By updating only the necessary parts of the user interface, OneScreenGrid contributes to the application's overall performance, providing a consistent and adaptive interface regardless of how the screen space is utilized.

The OneScreenGrid component exemplifies the power of Jetpack Compose's recomposition mechanism, which only re-renders user interface components when their underlying data changes. As a result, OneScreenGrid ensures that only the items that have been updated are redrawn, while those that remain unchanged are left as is. This optimization enhances the application's responsiveness and reduces unnecessary processing, making the user interface both dynamic and efficient. Through components like OneScreenGrid, the application achieves a seamless and fluid user experience, where users can interact with complex layouts and data-driven interfaces without sacrificing performance or

visual appeal.

3.4 App Functionalities

This section outlines all the features listed in 2.4 and highlights any unique aspects of their implementation. Similar features are grouped together for clarity, with the simpler ones described first. Notable features beyond the core list will also be discussed to provide a comprehensive overview of the app's functionality.

Features from thr feature list

One of the app's core features is the ability to roll **multiple dice** simultaneously. This functionality is fundamental to many of the app's advanced tools, allowing users to explore various combinations and configurations with ease.

To bring multiple dice onto the rolling screen, users have two options. As illustrated in Figure 3.3, dice can be **added or removed** directly to the rolling screen via a button that opens a dialog similar to the die creation process. Instead of configuring individual faces, users can select multiple dice and adjust their quantities (Figure 3.4). The second option is to create a Dice Group from the Dice Groups screen and select it by clicking on the desired group.

Whichever method is used, the third image in Figure 3.4 shows an example setup with multiple dice. The button at the bottom of the screen rolls all the dice and since it can be pressed multiple times, the **re-rolling** functionality is inherently available. Additionally, the action buttons include an option to **roll a single die**, allowing for more focused testing. The ability to modify the active selection of dice by adding or removing them is part of what was referred to as **simulation Mode**. Another way to modify the selection is making use of the button to edit a die. It will navigate the user to the editing screen and when the user clicks save the app will navigate back to the rolling screen applying the changes to the selection and the die entity in the datastore. Moreover, the presence of multiple dice on the screen supports **group Visualization**, as users can view and analyze different dice simultaneously. The action screen also includes a **lock/unlock** feature, which visually indicates locked dice. When the roll button is pressed, locked dice remain unchanged and the UI does not re-render their state.

Figure 3.4: Rolling multiple dice

The action screen also displays a "**history**" text at the bottom, next to the roll button (with two dice). Clicking this opens a log view, as shown in the previous screen in the same figure. This log records the roll count, sum and number of dice rolled. Users can click a down arrow to expand the log and view detailed information about each past roll, including the lock state, name, face and value of each die.

The **sum** feature calculates the total value of all dice faces shown on the screen after rolling. Locked dice are still included in this sum. In hindsight, a "hide" option would have been functionally distinct from the lock functionality, as it could be used to exclude dice from the sum. However, this was not anticipated during the initial design phase.

Moving on to another screen, the "Add images" screen provides the user with a button to use the native document picker to open one or multiple image files. The filename is automatically used as the content description of the image. This, along with adding tags to the **images**, can be modified on the same screen. Tags can later be filtered during the die if the user, for instance, uploads images for a specific die. The images are stored as squared image Bitmaps, which are saved in the datastore as base64 strings.

The ability to use images for the faces of the die is already part of having **versatile tools**. These images allow for endless creativity. In addition to special icons, users can utilize **colors**, drawings and even text to represent anything useful for a tabletop game. Furthermore, the number of images is only limited by storage capacity, making this feature applicable not just for dice but also for other board game mechanics like random card selection.

These same images can be applied to the state of a die. Users can add a selection of images as states to their Dice Group, allowing them to cycle through these states by clicking on a die in the rolling screen. The values, weights and numbers for a face also have very loose boundaries, which is crucial for supporting a **custom number range**. This feature allows users to generate dice with a set of numbers as faces, typically ranging from 1 to x. To facilitate this, there is a secondary button on the dice screen (Figure 3.2) that opens a dialog where users can set the start value and end value, quickly applying this range of faces to the editing screen (figure 3.3).

Another approach is to choose from the **selection of dice** and duplicate an existing die. For example, 4, 6, 8, 10, 12 and 20-sided dice can already be seen on the dice screen in Figure 3.3. These serve as a starting configuration, helping users quickly add more dice while showcasing the various options available. This selection not only aids in quickly adding dice but also inspires users with different ideas, such as using numbers, colors and even images like the animal die or a standard six-sided die with a bias towards six.

The custom number range is not the only secondary button next to the button for creating a die from scratch. There is also a die-shaped button that, when pressed, creates a random die with random faces, values, weights and a random color. While choosing this feature might seem straightforward—simply selecting random elements—making it effective involves numerous decisions.

The code ensures that the die is random, but generating a die with 100 faces, while potentially inspiring, reduces the likelihood of typical use cases that require smaller numbers. Therefore, a function was developed that outputs a distribution similar to a boxplot, where all values are possible, but most results cluster around a median.

The "getWeightsInRange" function (listing 3.3) generates a list of weights based on a specified range, a base value and a curve parameter. The function first checks if the starting index is greater than the base value, returning an empty list if this condition is met. Then, it creates a list of weights using a mathematical formula involving the base, index and curve parameter. Finally, if the addLeadingZeros flag is set and the starting index is greater than 0, the function adds leading zeros to the list of weights

the result of this function can be used to get a random weighted element with the function "weightedRandom" (listing 3.4. It takes a list of weights as input and returns a randomly selected index based on the relative weights of each element. The function

first calculates the cumulative sum of the weights using runningFold. Then, it generates a random value within the range of the total weight. Finally, it performs a binary search on the cumulative weights to find the index corresponding to the random value, effectively selecting an element based on its weight.

Listing 3.3: Function getWeightsInRange

```
fun getWeightsInRange(start: Int, base: Int, end: Int, curve: Double =
    0.5, addLeadingZeros: Boolean = true): List<Double> {
  if (start > base) return emptyList()
  val weights = (start..end).map { curve.pow(abs(base - it)) }
  return if (addLeadingZeros && start > 0) {
    List(start) { 0.0 } + weights
  } else {
    weights
  }
}
```

Listing 3.4: Function weightedRandom

```
fun weightedRandom(weights: List<Double>): Int {
  val cumulativeWeights = weights.runningFold(0.0) { acc, weight -> acc
    + weight }.drop(1)
  val randomValue = Random.nextDouble(cumulativeWeights.last())
  return cumulativeWeights.binarySearchCeiling(randomValue)
}
```

The same function can be used for creating random Dice Groups, ranging from 1 to 500, with 10 as the baseline. These groups are basically **presets** a user can create, to save configurations, ideas and quickly roll a predetermined combination of dice.

In the development the focus was on giving a **user friendly design**. As already seen, there is also a central navigation which let's the user not get lost in the app. Also when first opening the app the user will go through some slides depicting different use cases. This way the user will not be overwhelmed by all the different functionalities. After the slides the app will show the rolling screen with four dice that are ready to be rolled. This was decided to incentivise the user to explore the app.

Finally, it has been previously explained how managing dice and Dice Groups work equally for adding, editing, deleting and duplicating in regard to **intuitive design**. Additionally, when rolling dice in the rolling screen they would always bounce from left to right. Not only helps this mechanism for triggering re-rendering, but it also gives the user the feeling that something happens when they press a button. This is important, because if the user rolls the same output again, the UI would not change otherwise. One can argue that the different selection Dialogs for images, groups, states and weights are

not intuitive, because while the flow is the same, the slider's values are different each time resulting in different outcomes for similar looking actions.

Additional Features

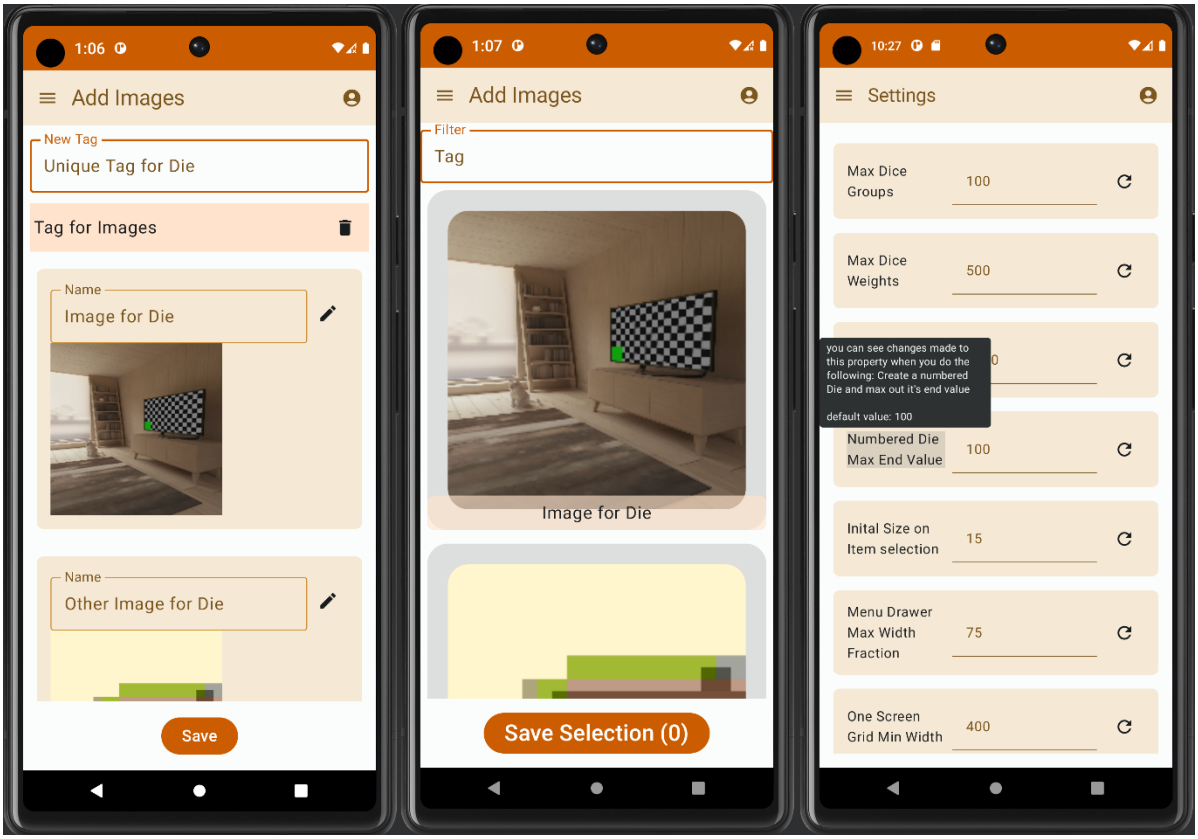

Figure 3.5: Other features

While the UI would not change much without the left and right bouncing of the dice, it would still change the roll counter on the bottom of the screen above the history text seen in figure 3.3. This counter helps keeping a clear history. Also in the rolling screen, the user can reset all the dice, history and rolls. This simply sets all the variables in the ViewModel to their starting values

The absence of a login is also something worth mentioning as this lets the user jump right into action to work on their creative innovations. Another user centered element is being able to tag images. It allows for a quicker way to select images to create a dice. The filter for searching images will just include the Strings from the list of tags and can be seen in figure 3.5.

Lastly, there's a setting page when clicking on the user profile. It allows to change some of the variables, like initial value when selecting a face. The screen also explains where these settings apply and they can always be reset to the default value (figure 3.5). While initially being a developer centered testing tool, it can help for users to personalize their experience and extend some of the boundaries they might still face within the app.

3.5 Limits and Improvements

The previous sections provided a comprehensive overview of all the features and functionalities implemented in the app. This section, however, focuses on the aspects that were not included in the final version, as well as areas that could benefit from further refinement and improvement.

While I, the developer, am familiar with the Android environment, my experience with Kotlin and Android development, particularly in the area of testing, is relatively limited. As a beginner in both Kotlin programming and Android-specific development practices, I encountered several challenges, especially when it came to writing effective tests. Learning to navigate a new programming language and framework simultaneously while developing an app was a demanding process and this was particularly evident in the testing phase.

Testing is a crucial aspect of software development, ensuring that code behaves as expected and reducing the likelihood of bugs or errors in the final product. Despite my inexperience, I made a concerted effort to incorporate unit tests, especially for utility functions like the weightedRandom function (Listing 3.4), using the JUnit testing framework. However, the testing coverage is not as comprehensive as it could be, primarily due to my novice understanding of best practices in this area.

Given the importance of testing, it is clear that a more robust and thorough testing strategy is needed for further development (for example in the creativity support features). Establishing a strong testing foundation is essential, as it not only helps in identifying and fixing issues early but also facilitates easier refactoring of code. As this codebase evolves, having a well-structured test suite will be invaluable in ensuring that any changes or additions to functions and classes do not introduce new bugs or regressions.

Another critical area for improvement is error handling. In the current implementation, error handling is minimal and often relies on defaulting to empty or mock data when issues arise. For instance, in scenarios where errors might occur—such as during user input validation or data loading—the app typically does not provide meaningful feedback to the user. Instead, it quietly falls back to placeholder data, which can lead to confusion

or a lack of awareness of underlying issues.

A more robust approach to error handling could involve implementing toast messages or other forms of user notifications to clearly communicate when something goes wrong. By providing immediate feedback, users can better understand what actions to take or what adjustments are necessary. This improvement would enhance the overall user experience by making the app more responsive and informative.

However, it's also important to recognize that Kotlin's strong null-safety features inherently reduce the need for extensive error handling in some areas. Kotlin's design minimizes the occurrence of null pointer exceptions, which are a common source of errors in other programming languages. Additionally, the app leverages modern frameworks like Jetpack Compose and Proto Datastore, which include built-in mechanisms for handling asynchronous actions and provide fallback values. These frameworks mitigate some of the risks associated with errors by ensuring that the app remains functional even when unexpected issues occur.

Nevertheless, the reliance on these frameworks does not eliminate the need for thoughtful error handling. Enhancing the app's error handling capabilities would contribute to a more polished and reliable product, particularly as the app grows in complexity and functionality.

When these areas are improved the next focus point should be adding some kind of note functionality to dice and Dice Groups. This gives the user the ability to quickly write down ideas so that they will not get lost. It helps in managing Dice Groups and association to the ideas. Developers can write down game mechanics, things that went well or did not during testing and many more. It also is a good foundation to a collaboration tool, where a note could be made into a comment. This additional feature would have been very valuable in the aspect of creativity support tools.

In summary, the app needs a better testing suite that will provide evidence that the features are working as intended even after refactoring or extending the app. Furthermore, an error handling that gives the user feedback is necessary for a better user experience and the next feature should involve notes for dice and Dice Groups

4 Evaluation

The objective of this paper was to develop a comprehensive tool for tabletop game developers that not only fosters creativity but also integrates all the key features available in existing dice simulation apps. The goal was to create an all-in-one solution for dice game development, offering a superior and more versatile experience than the options currently available. This evaluation assesses the methods employed in the research and development process, analyzes the results, identifies limitations and suggests possible improvements for future research.

To achieve the project's objectives, a thorough investigation of the Android market was conducted, utilizing sources such as the Play Store, Google and AI-driven searches. This research led to the compilation of a feature list encompassing the functionalities of existing dice apps. Additionally, an in-depth exploration of design principles within the field of creativity support tools (CSTs) informed the framework for the app, which was then implemented.

While the exploration of dice games and their history provided valuable context and underscored the importance of dice games in the broader scope of tabletop gaming, it contributed little to the development of a dice simulation app. A more in-depth investigation into how dice are used across various games would have better informed the feature requirements for the app. Similarly, while the market research effectively identified a gap for a comprehensive, all-in-one tool, it could have been more thorough. Exploring additional keywords such as "Random," "Chance," "Picker," and "Lucky Draw" might have uncovered more relevant tools and use cases. Furthermore, the exclusion of iOS apps from the research represents a missed opportunity, as insights from that ecosystem might have provided valuable perspectives.

The research into Creativity Support Tools was successful in identifying principles and design ideas that could enhance the app's functionality. However, it should be noted that no directly comparable work, particularly in the realm of smartphone apps, was found. Most of the existing literature is either not recent or focuses more on AI-driven tools. This shift in research focus towards AI could have been better highlighted in the section on current research.

Initially, the plan to store images on a server was reconsidered as the research progressed. It became evident that modern, type-safe local storage solutions like Proto Datastore

were more suitable for this project. The use cases examined did not justify the need for server-side storage, as most of the app's functionalities operate independently of such infrastructure. Implementing server-side storage would have introduced significant data security challenges and unnecessary complexity for a feature that primarily benefits image sharing. Therefore, the decision to utilize local storage proved to be both efficient and secure.

When compared to other tools on the market, Dynamic Dice successfully integrates all the features identified as essential for a comprehensive dice simulation app. There is no other android app currently available that combines this level of functionality, making Dynamic Dice a uniquely powerful tool for dice game development. On top of combining all the functionalities there have been features like "weighted faces", "random dice" and "states for dice" that have not been part in any apps of the research. This gives users new possibilities to develop ideas. However, while the app's technical capabilities are clear, no formal evaluation has been conducted to assess whether it effectively supports creativity. Without such an evaluation, it remains uncertain how well the app achieves its goal of fostering creativity among game developers.

One significant limitation of this study was the lack of research into smartphone apps, particularly those available on iOS. This gap made it challenging to derive comprehensive requirements from related work. Additionally, the absence of a user-centered design approach limited the study's ability to identify and address the most critical issues and use cases for tabletop game developers.

Future research could benefit from a more extensive engagement with tabletop game developers through a user-centered design approach. Such an approach would likely provide deeper insights into the critical issues and important use cases that were not fully captured in this study. Never the less, it is worth noting that regular interaction with board game enthusiasts provided valuable iterative feedback throughout the development process. This aligns with the iterative design methodology endorsed by Wang (2017) [10], which emphasizes the importance of continuous feedback and refinement.

In conclusion, while Dynamic Dice represents a significant advancement in dice simulation apps, further research and development are needed to fully understand and enhance its capacity to support creativity in game development.

5 Conclusion

This paper set out to design and implement Dynamic Dice, a comprehensive tool for tabletop game developers that not only integrates the key features found in existing dice simulation apps but also fosters creativity through the application of Creativity Support Tools (CSTs) principles. The research question guiding this project was centered on whether it was possible to create an app that meets the diverse needs of dice game developers while incorporating design principles aimed at enhancing creativity. The goal was largely achieved, although some features and design principles could not be fully implemented within the scope of this project.

A thorough analysis of the Android market led to the compilation of a detailed list of common features in dice simulation apps, providing a strong foundation for the development of Dynamic Dice. This analysis highlighted significant gaps in existing tools, particularly in terms of customization and support for creative processes. In response, Dynamic Dice was developed with extensive customization options and advanced features such as weighted dice probabilities, custom images for dice faces and flexible group states. These enhancements empower developers to push the boundaries of traditional dice games, enabling the creation of new and engaging experiences.

The research also established a robust framework of design principles for creating CSTs, which guided the development of key features within Dynamic Dice. For example, features like random dice generation and versatile tools such as the ability to add images to dice were directly inspired by these principles. However, to fully realize the potential of CSTs in supporting all aspects of creative development, additional features—such as note-taking, collaboration, sharing, exporting and importing data, tutorials and scripting capabilities—could be implemented in future iterations of the app.

Despite the successes, the project faced several challenges. Integrating modern technologies like Jetpack Compose and Proto Datastore proved to be difficult, especially given the limited and sometimes outdated online resources available. Ensuring compatibility with the latest versions of these tools required significant effort and adaptation. Additionally, the lack of related work specifically covering apps or dice simulation tools made it challenging to find direct precedents or benchmarks, further complicating the development process.

The exclusion of iPhone apps from the market research also represents a limitation, as

it is possible that iOS apps exist that already fulfill the identified requirements. Future research should address this gap by including iPhone apps in the analysis to gain a more comprehensive understanding of the market and ensure that future versions of Dynamic Dice are as versatile and widely applicable as possible.

The implications of this work extend beyond the development of Dynamic Dice. The framework and findings from this study can be applied to other areas of smartphone app development, allowing developers to focus more on creative innovation rather than on duplicating existing research efforts. This approach has the potential to accelerate the creation of tools that better meet user needs, contributing to a more efficient and user-centric app ecosystem.

Looking forward, further research should aim to evaluate the effectiveness of Dynamic Dice as a creativity support tool. Using established metrics like the Creativity Support Index (CSI), future studies can provide valuable insights into how well the app meets its goal of empowering developers to innovate within the realm of dice-based games. Additionally, expanding the research to include a broader range of smartphone platforms and related tools will be crucial for ensuring that Dynamic Dice continues to evolve as a leading tool for game developers.

In conclusion, while there is room for further research and refinement, this study has successfully laid a solid foundation for the development of creativity-enhancing tools in dice-based game development. Dynamic Dice represents a significant step forward, providing game developers with the tools they need to innovate and succeed in an increasingly competitive market.

List of Figures

Appendixes

Appendix A: Market search findings.

App Name	Origin	Features	Major limitations
Dice Clubs Kniffel	Playstore	yahtzee simulation	no single player
Würfel	Playstore	multiple dices, custom number range, different colors	no images, no weights
Custom Dice	Playstore	Bemalbare seiten, Farbe, Selection of dice	no images, no number range
Mighty Dice	Playstore	Selection of dice, dice with custom configuration, Rolling history, rerolling, Sum	no images, no different colors for dices
Dicely	Playstore	Selection of dice multiple throws, RPG orientated, Custom Number range, rolling history, Sum	no categorization for dices, no images, no locking
Fantasy Dice Roller	Playstore	Selection of dice , cool animations, reroll, rolling history, removing dices, adding dices, presets, Sum	no categorization for dices, no images, no locking
DiceApp	Playstore	Sum, multiple dices, different colors, rerolling	no images, only standard 6 dice
Cyber Dice	Playstore	Selection of dice, reroll, rolling history, removing dices, adding dices, rolling presets, Sum	no categorization for dices, no images, no locking
Würfel	Playstore	multiple dices, different colors	only standard 6 dice, no images, no locking, limited colors
Dice	Playstore	Sum, multiple dices, different colors, rerolling, Selection of dice, rolling history	no images, no locking, limited colors
Die Würfelbecher	Playstore	multiple dices, custom number range, hide dices, sum, rerolling	no images, no locking, no categorization for dices, limited number range, no individual dice setting
Würfel	Playstore	Sum, different colors, Selection of dice, multiple dices, rerolling	slow, no images, limited number range, no individual dice setting

Würfel	Playstore	Sum, multiple dices, different colors, rerolling	only standard 6 dice, no images, no locking, limited colors, no categorization for dices, no individual dice setting
Würfelwurf SNS	Playstore	multiple dices, rerolling	only standard 6 dice, no images, no locking, limited colors, no categorization for dices, no individual dice setting
Würfel	Playstore	Statistic, Sum, multiple dices, rolling history, rerolling	only standard 6 dice, no images, no locking, limited colors, no categorization for dices, no individual dice setting
Dice	Playstore	Selection of dice, different colors, multiple dices, rerolling, locking	no images, limited number range
RPG Simple Dice	Playstore	Selection of dice,	limited number range, no images, no categorization for dices, no locking, no individual dice setting
My Dice Roller	Playstore	multiple dices, different colors, rerolling, Selection of dice, rolling history	no images, limited number range
Würfel 2D	Playstore	multiple dices, images (but limited), custom number range, adding dices, removing dices	no colors, no locking
Ky6NK	Playstore	multiple dices, Sum, rerolling, rolling history	no images, no categorization for dices, only standard 6 dice
Roll Tracker	Playstore	Statistic, Sum	very limited to 2 6 dices or 1 20 dice, no colors, images..
DieDroid	(F-Droid)	Selection of dice,	imited number range, 1 dice
OneTwo	(F-Droid)	multiple dices, Custom Number range, rerolling, roll single	no images, no categorization for dices, no locking

Custom Image Dice	Copilot / playstore	customizable dice with images, multiple dices, Presets	no standard dices, numbers, colors, no weights, no categorization for same dice (difference), no roll single, no sum
Roll My Dice	playstore	Presets, colors, multiple dices, removing dices, adding dices, Sum, locking	no weights, tedious face count, paywall for images, no number range, no copying dices from other bundles to new bundle

Catan Website, Accessed 25.08.2024, 12:12

ABOUT THE CATAN GMBH

We are part of the entertainment and gaming industry. Today, our primary product is the successful board game CATAN (formerly *The Settlers of CATAN*), which comes with an array of expansions, editions, and spin-offs. Before founding CATAN GmbH in 2002, CATAN creator Klaus Teuber gained valuable experience as an independent game designer and creating game worlds was his passion.

Over more than two decades, Klaus built an impressive portfolio of board games and electronic games, which is now being carried on by his son Benjamin as a game designer. CATAN is available in over 40 languages; since 1995 until now (Q1 2022) over 40 million games were sold worldwide. According to estimates by our licensing partners, approximately 20 million people regularly spend time in the CATAN gaming universe. Overall, ours is an ever-growing family of high-quality entertainment products and programs, all in keeping with a vision devoted to the cultivation of a vast and largely untapped community.

THE CATAN GMBH TEAM

KLAUS TEUBER
Game Designer

Klaus Teuber was born in 1952 at the foot of Breuberg Castle in the village of Rai-Breitenbach, surrounded by forests and gentle mountains. Did this rural, idyllic Odenwald backdrop later motivate him to develop CATAN? Maybe.

Board games did not play a big role in his childhood until he was given the game *Romans vs. Carthaginians* at the age of 11. Soon the ancient armies had left the game board and fought fierce battles on the floor of the children's room. Klaus added new rules to the game and is now convinced that the foundation stone for his later game developments was laid during this time. After graduating from high school and doing military service, he studied chemistry. After completing his intermediate diploma, he joined his father's dental laboratory for family reasons. To compensate for the often very stressful everyday working life, Klaus began developing games in the early 1980s.

In 1988, his debut *Barbarossa und die Rätselmeister* was named game of the year. Further games and game prizes followed until he hit the jackpot in 1995 with *The Settlers of CATAN*. In the following years, Klaus Teuber mainly took care of CATAN and expanded it into a large games portfolio, in cooperation with the editors of the Kosmos publishing house and later with the help of his son Guido, who now lives in the United States, and the American publisher Mayfair.

In 2002, together with his wife Claudia and his two sons Guido and Benjamin, he founded the family business CATAN GmbH, whose task is the development, marketing, and licensing of the games he has developed.

Klaus passed away in April 2023. His two sons continue to run the company and the development of the CATAN world together with the CATAN team in his spirit.

You can find all of Klaus' games in the ludography.

List of Tables

Listings

Bibliography

[1] Frederick J Poole et al. "Tabletop games designed to promote computational thinking". In: *Computer Science Education* 32.4 (2022), pp. 449–475.

[2] Joseph Cataliotti. "Dice History, Invention and Games". In: (2023). URL: https://study.com/academy/lesson/ancient-dice-history-games-facts.html.

[3] ostenbb. "Game Mechanics Research Paper: Dice Rolling". In: *sites.miamioh.edu* (2021). URL: https://sites.miamioh.edu/tabletop/2021/12/game-mechanics-research-paper-dice-rolling/.

[4] Miguel Sicart. "Defining Game Mechanics". In: *the international journal of computer game research* (2008). URL: https://gamestudies.org/0802/articles/sicart.

[5] Trevor Harron. "Game Mechanics: Rolling Dice". In: *Morning Table Talk* (2018). URL: https://boardgamegeek.com/blogpost/77674/game-mechanics-rolling-dice.

[6] Erin Cherry and Celine Latulipe. "Quantifying the creativity support of digital tools through the creativity support index". In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 21.4 (2014), pp. 1–25.

[7] Jonas Frich et al. "Mapping the landscape of creativity support tools in HCI". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–18.

[8] Ben Shneiderman. "Creativity support tools". In: *Communications of the ACM* 45.10 (2002), pp. 116–120.

[9] Mitchel Resnick et al. "Design principles for tools to support creative thinking". In: (2005).

[10] Kai Wang and Jeffrey V Nickerson. "A literature review on individual creativity support systems". In: *Computers in Human Behavior* 74 (2017), pp. 139–151.

[11] Ben Shneiderman. "Creativity support tools: accelerating discovery and innovation". In: *Communications of the ACM* 50.12 (2007), pp. 20–32.

[12] Steven Abrams et al. "QSketcher: an environment for composing music for film". In: *Proceedings of the 4th conference on Creativity & cognition*. 2002, pp. 157–164.

[13] Celine Latulipe. "The value of research in creativity and the arts". In: *Proceedings of the 9th ACM Conference on Creativity & Cognition*. 2013, pp. 1–10.

[14] Evropi Stefanidi et al. “Literature reviews in HCI: A review of reviews”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–24.

[15] Elizabeth M Gerber and Caitlin K Martin. “Supporting creativity within web-based self-services”. In: *International Journal of Design* 6.1 (2012).

[16] Josh Kosof. “Building Creativity Support Tools for Dungeon Design in Tabletop Games: A Participatory Design Study”. MA thesis. Northeastern University, 2021.

[17] Srishti Palani et al. “” I don’t want to feel like I’m working in a 1960s factory”: The Practitioner Perspective on Creativity Support Tool Adoption”. In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 2022, pp. 1–18.

[18] Tom Hewett et al. “Creativity support tool evaluation methods and metrics”. In: *Creativity Support Tools* (2005), pp. 10–24.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorgelegte Bachelorarbeit selbstständig verfasst und noch nicht anderweitig zu Prüfungszwecken vorgelegt habe. Alle benutzten Quellen und Hilfsmittel sind angegeben, wörtliche und sinngemäße Zitate wurden als solche gekennzeichnet.

Julian Sehne, am September 2, 2024

Zustimmung zur Plagiatsüberprüfung

Hiermit willige ich ein, dass zum Zwecke der Überprüfung auf Plagiate meine vorgelegte Arbeit in digitaler Form an PlagScan (www.plagscan.com) übermittelt und diese vorübergehend (max. 5 Jahre) in der von PlagScan geführten Datenbank gespeichert wird sowie persönliche Daten, die Teil dieser Arbeit sind, dort hinterlegt werden.

Die Einwilligung ist freiwillig. Ohne diese Einwilligung kann unter Entfernung aller persönlichen Angaben und Wahrung der urheberrechtlichen Vorgaben die Plagiatsüberprüfung nicht verhindert werden. Die Einwilligung zur Speicherung und Verwendung der persönlichen Daten kann jederzeit durch Erklärung gegenüber der Fakultät widerrufen werden.

Julian Sehne, am September 2, 2024