Overlay: an Educational Disc Covering Puzzle Game

Ryan Devaney, Sanya Gupta, Vinh Le
Connor Scully-Allison, Frederick C. Harris, Jr. & Sergiu Dascalu
Department of Computer Science, University of Nevada, Reno
Reno, Nevada, 89509, USA
(rgdevaney13, sanyahgupta)@gmail.com
(vle, cscully-allison)@nevada.unr.edu
(fred.harris, dascalus)@cse.unr.edu

Abstract

In the last decade, video games have quickly become a major contender against present media standards,like movies and music. While video games often serve as a form of entertainment, it has had a long history of being used as an engaging tool for education. However, video games that are more tailored towards education tend to fail in compelling their audience to play as strongly as their non-educational cousins. To address this problem, this paper presents Overlay: an educationally-oriented video game that is designed to teach basic problem-solving skills while maintaining high levels of engaging entertainment. Overlays core gameplay mechanic derives from Richard Kershners Disk Covering problem, where users use basic geometry to solve a positioning problem and progress through the game. As users progress through the game, the levels becomes increasingly more challenging. Progress is tracked through the game and the top-ranking scores are stored online through a web service handling communication to the main data source. These gameplay aspects allow Overlay to seamlessly blend STEM education with viscerally enjoyable entertainment.

keywords: Education, STEM, Unity, Flask, Python, Puzzle Game

1 Introduction

The blending of education with video games frequently yields a result that betrays the engaging aspect that is expected of the genre. With the intent of creating that middle ground where education meets the challenging and engaging aspects of video games, our team developed Overlay, a puzzle game that brings mathematics into a modern gaming setting. In this game, users place circles inside of a larger revolving circle with the goal of covering the larger circles surface area. Once the larger circle is covered, the users score will update based on their performance in both time and accuracy, then advance to the next stage. At all times, Overlays interface states the smallest possible amount of circles that could be used to cover the larger circle. If the user places any number of circles exceeding that of the smallest amount, their scores will be deducted based on poor accuracy. The smallest amount of circles is determined mathematically using the disk covering problem developed by Richard Kersher and will be covered in Section 2. At the end of each game session, the total score is calculated based on each completed stage, the deductions from each stage, and the overarching timer placed on the entire game session.

At the core of the game, Overlay can be broken into two major parts: the main game client and the web service that manages the top ten current rankings. Overlays main game client was designed and implemented with the Unity Game Engine and the C# programming language. The choice of Unity and C# was determined from Unity being a powerful and free game engine, while C# was chosen due to its compatibility to Unity and its robust native HTTP libraries for web communication. The support web service for Overlay was implemented with the Flask Microframework and the Python 3 programming language. The choice of Flask and Python was made due to Flask being a very robust and easy to deploy service framework.

Overlay was implemented with the option of being deployed as both an executable through Unitys application support and as a mobile application for modern smart phones. Additionally, all stages were designed based off a modular template that allows for new gameplay mechanics to be easily added such as different geometric shapes, new stages, and new user challenges. This customizability allows users the option to create a new experience for future play and offer a viable choice for replay value. The rest of this paper is structured as follows: Section 2 covers the background of the disk covering problem and similar work to Overlay in

the field of educational gaming. Section 3 covers the software engineering aspect of Overlay which includes functional requirements, non-functional requirements, and use cases. Section 4 covers the implementation of Overlay. Finally, section 5 covers the conclusion and future work for Overlay.

2 Background and Related Work

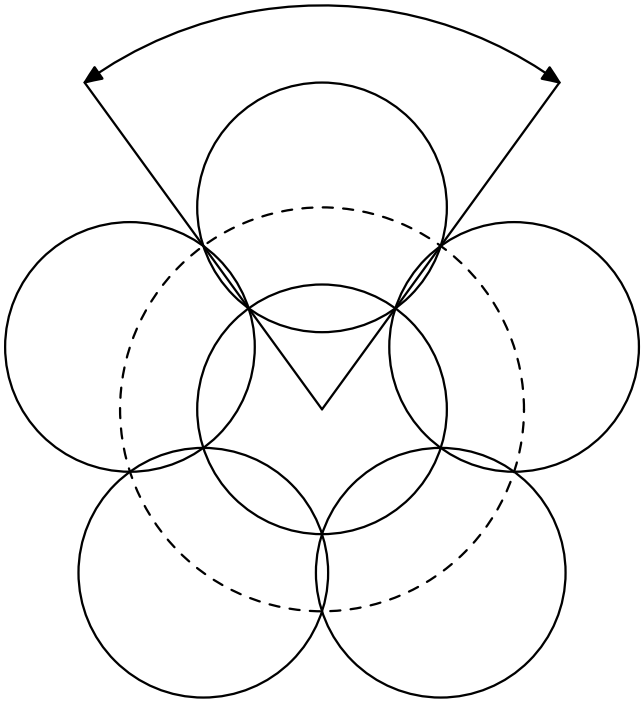

Figure 1: This is a visual representation of the Disc Covering problem. With the dashed line indicating the larger circle being covered, we see the optimal number of smaller disks required to cover it up.

The fundamental mechanic of Overlay is based on a fundamental mathematical problem introduced by Richard Kershner [5]. The problem explores the minimal amount of circles required to "cover a set." That is, how many smaller circles of a given radius are required to cover one large circle of a given radius. A visual example of this problem can be seen in Figure 1. The specifics of this paper are highly esoteric and hard for laymen to grasp however, because of it's mathematical basis it provided two benefits to the design of this game. First, it provided a clear, and easily implemented optimal solution for each puzzle. Second, it provided a unique opportunity to educate players about otherwise arcane mathematical concepts and hopefully drive a general interest in mathematics.

Overlay is certainly not the first game built using mathematical concepts as a core gameplay element. Many games in the puzzle genre often employ basic geometry and associated mathematical theorems to determine optimal play. Tetris, a classic game puzzle game, has complex combinatorics underlying it's gameplay [1, 8]. Similarly, other games like Valve's critically acclaimed *Portal* and the indie hit *Antichamber* frequently utilize geometric mathematics to create complex looking 3D puzzles with clearly-defined, optimal solutions [2, 9]. Although well saturated, Overlay contributes uniquely to this field by providing a fast paced, easy to pick up experience with almost no learning curve. Additionally, Overlay offers a unique entry point into the specific Disc Covering problem employed as a gameplay mechanic.

The field of educational games has been well researched with significant academic work researched for Overlay. Specific to the field of puzzle games, the educational merit of mathematical and geometric games has been explored by Sarah Newcombe [7]. In this paper, Newcombe argues that puzzle games like Tetris, can have a positive impact on a student's spatial thinking required to succeed in STEM fields. In addition to the stated goal of driving interest in Math topics, Overlay meets many criteria expressed by Newcombe required to bolster a students spatial thinking. This work reinforced many game design decisions made to optimize the educational aspects of this software.

In addition to the work by Newcombe, other authors have also shown that puzzle games like Overlay can have a positive educational impact on players. Bobby Law showed a positive correlation between puzzle games an computational thinking, indicating another possible avenue of educational value for Overlay [6]. Fokides, in a 2016 case study, also indicated the promise of Math-based educational games by comparing a game-using primary school group against a non-game-using control group from the same school[3]. The results were very positive in favor of the the game group and further indicates the promise of games like Overlay.

Outside of broad indications that games can be a valid educational tool, significant recent work has been done to understand how best to design video games for educational purposes. In thier educational paper, Horn et. al., explored how to evaluate the strategies used by players as a metric of educational impact [4]. In another paper by Scozzi et. al., a novel approach was proposed by the authors to best iterate on the design of educational games [10]. Both of these works were considered in Overlay's game design so that any

educational aspects of the game could be best evaluated and expanded upon.

3 Software Specification

This section is broken into three main parts: functional requirements, non-function requirements, and the use cases associated with this software. The functional requirements provided detail the technical functionality present within Overlay and are organized in levels to differentiate core features from secondary ones. The non-functional requirements serve as constraints, both in hardware and software, on the Overlay game. Finally, the use cases show the interaction between users and the Overlay game.

3.1 Functional Requirements

Regarding base level feature, the Overlay game consists of six total functional requirements. The first requirement of Overlay indicates the necessity of a main menu screen to start the game, change user settings, view the current leaderboard, and exit the application. The second requirement is that a user may interact with game client during a game session, through a mouse click or touch input. The third requirement ensures that once progress is made inside the Overlay game, the ensuing levels will increase in difficulty by implementing a new game mechanic. The fourth requirement is that a score will be kept up to date and visibly shown during the extent of the session. The fifth requirement involves the losing state of the game, where a dynamic number of losing states will be added to the game as the user progresses deeper. Finally, the sixth requirement is that the Overlay game will communicate with a web service to transmit a score for evaluation and update the global leaderboard appropriately. Additional features were also added that are considered as level two requirements. The first requirement was the ability for users to customize the theme and music of their game instance through the settings page. The second requirement allowed users the option to customize their gaming experience by importing their own game objects. Finally, the third requirement involved specialized visual and audio notifications that were triggered when a user progresses deeper within the game. These notifications indicated that a user either increased the difficulty, unlocked new losing states, or are severely approaching a losing state.

3.2 Non-Functional Requirements

During the development of Overlay, a handful of constraints were placed on the system in the form

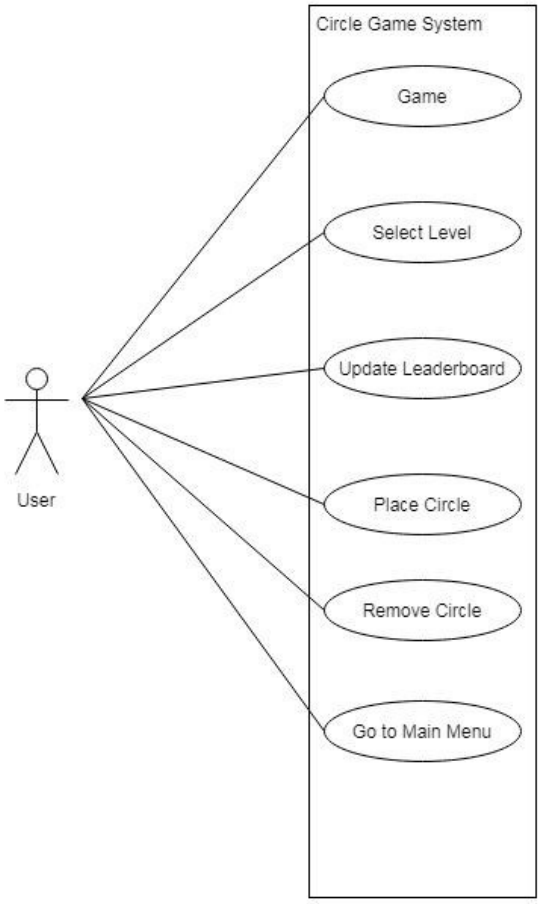

Figure 2: Use case diagram for Overlay showing user interactions with the system.

of non-functional requirements. The first constraint required that the Overlay game client be available as an executable on a computer or as an application on a mobile device. The second constraint required the game to be built with the Unity Game engine. The third constraint required that the game client be developed with C# as the main scripting language. The fourth constraint required the Overlay web service to be developed with the Flask Microframework and the Python programming language. Finally, the fifth constraint is that the Overlay game requires internet access in order to save scores of a game session.

In addition to these multiple software constraints, further constraints were placed upon the hardware Overlay was expected to run on. First, Overlay was only required to run on android mobile devices and Windows PCs. Additionally, to best facilitate the running on these devices, Overlay was required to compile into an executable taking up no more than 1GB of persistent memory on the device. It was also

Figure 3: When the application is run, the main menu is loaded for the user to navigate.

required to be extremely responsive to touch inputs, with absolute minimal latency between touching a mobile screen and the spawning of a circle.

3.3 Detailed Use Cases

This section presents the detailed use cases that serve as a representation of the interaction between the role of user and the Overlay game system. Further details are shown below about each individual use case and a use case diagram is provided in Figure 2.

- Game: The user is ultimately interacting with the entire game throughout.
- Select Level: When the user wants to play they will be selecting the first level.
- Update Leaderboard: After a user submits their score, their score will be reflected onto the leaderboard if it meets the criteria.
- Place Circle: The user will be able to place a circle with mouse clicking or touch input while in the game.
- Remove Circle: The user will have circles removed after they have covered the background circle.
- Go to Main Menu: The user will be able to go to the main menu after completing a game.

4 Implementation

The documentation outlined in Section 3 was used to create a development process that would be both flexible and robust. Our team worked with advisors very closely and often so that guidance could be given very quickly. Tasks were given out based upon the individual's comfort with that task. Initially, we first developed one working level and game specific mechanics were added incrementally to make the level more interesting. Once a singular level was finalized, additional levels and features were added in batches to the game.

4.1 Main Menu

The game includes a main menu so that users are able to select which aspect of the game they want to access. This menu interface is the first thing that comes up at the very start of the games execution, as shown in Figure 3. The menu includes a play button that starts a new game session and leads players into game interface described below. In addition to the play button, there is also an options menu that allows users to change game settings, add music, and upload game assets. Next, there is a rankings button that will query a web service for a global ranking list and show the top ten scores. Finally, there is an exit button that will allow users to quit the game and shut down the application.

Figure 4: An example of the core gameplay loop. Players can place down as many circles as desired; however, circles which exceed the number required for an optimal solution will detract points. Also taking excessive time to solve the puzzle will also result in lost points.

4.2 Game

This interface is produced when invoking the play button from the main menu. The major aspect of the game session is to fill in a larger revolving circle with the fewest amount of smaller circles, as shown in Figure 4. As the game continues, each millisecond decreases the over score, which can be decremented past zero. Completing a stage with the appropriate amount of circles will yield the users a set amount of points, however if that amount is violated, then a portion will be deducted. Each of these statistics: time, circles placed, smallest amount needed, current stage, and the total score are made viewable on the left side of the game interface. In order to keep the game fun and challenging, new mechanics were added every ten levels, shown in Figure 5. These mechanics can include circles

Figure 5: After each subsequent series of 10 stages has been surpassed users will see this message. The difficulty scales with the addition of a new "gimmick," like a super-short timer or an reverse spinning circle.

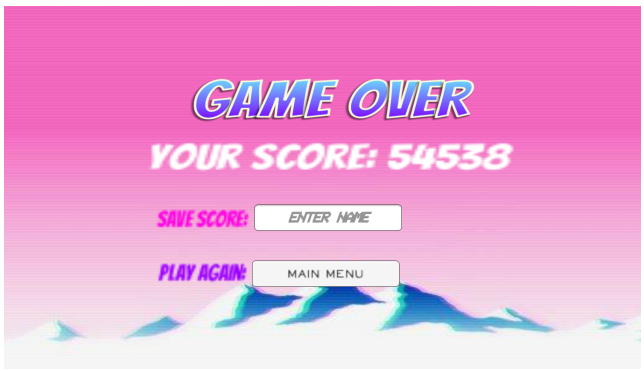

Figure 6: After losing the game, users are presented with this screen. When a name is input, the associated score is uploaded via an HTTP call to a persistent data structure on a remote server.

the user placed disappearing after a set time, the user automatically losing after a certain amount of time, or a culmination of many. Finally, a score is given to the user after they completed the game and if high enough, this score will be sent to the web service to be updated on the global leaderboard.

4.3 Ranking

This interface is generated from two sources, the main menu from the ranking button or after a finished session from the main game. In this interface, the game client immediately queries the global ranking list from the support web service and shows it in a list to the user. It must be noted that by entering the ranking interface via the main menu, it does not allow the user to input a score. However, after the user has ended the game, they are sent to a score submission screen, as shown in Figure 6. Here the user can input their name and email and this information then gets sent to a web service that evaluates the score and potentially saves the top ten scores. It is then that the user is sent to the Ranking interface to view whether their score made it to the global rankings.

5 Discussion

As a case study, to prove the effectiveness of this software, Overlay was exhibited at a campus wide innovation day. Hundreds of individuals attend this yearly event to view top projects produced by the University of Nevada Reno. Among those participants, many of whom are children in our intended demographic, over 120 users were given the opportunity to play this game and give informally structured feedback on their experiences. From this exposition we draw much of our

conclusions.

Paramount to the primary goal of Overlay, we found that, through playing, many users expressed an interest in understanding the math underlying the disc covering mechanic central to gameplay. This was often casually expressed with questions like "How do [we] balance gameplay," and "How could I get a perfect score?" Questions like these clearly indicated an interest in the mechanics of the game but also in the underlying logic of how the game works. Explanations of the underlying mathematical theory were well received by users who asked questions like these. We suspect that it may be tied to the an investment generated by the fast paced and visceral gameplay.

In addition this this educational reception of our software, it was also observed that Overlay was strongly validated with a second metric: enjoyment. Users overwhelmingly had fun playing this game. Many users, especially those in the 8-12 year old demographic, crowded around the Overlay booth and competed against each other for high scores. Users frequently requested the opportunity for an additional play through after hitting the game over screen. Additionally, many players also commented positively on the visual design of Overlay, noting how "vibrant" and "unique" it is.

6 Conclusion and Future Work

This paper presented a new video game aimed at providing both a sense of engaging entertainment and an education value centered around geometry and mathematics. The main characteristics of the new game have been detailed through the software specification and the implementation. We believe that this game application brings a strong collaboration between educational gaming and modern game design concepts.

Future development of Overlay includes transferring the entire game client over to a web application. This would allow the game to be available on all platforms and operating systems through a web browser and internet connection. This would not eliminate any previous iterations of the game, so the web application would be in addition to the current executable and the mobile application.

Currently, the Overlay game runs off ten iterations of the disk covering problem that are repeated with different and harder mechanics to create additional level. In the future, we would develop finer detailed levels of the disk covering problem. This would create a stronger enforcement of the geometric lesson being taught and create more challenging levels through the greater amount of circles being placed.

Finally, the Overlay web service currently runs on a virtual machine in a physical machine while communicating to a local data source. In the future, we would like to expand to using cloud technologies to host the game on a cloud instance, such as Amazon AWS. In addition, we would like to instead monitor scores and other user data, such as mouse activity and time spent on a specific level, through autonomously storing data on a database, such as SQL Server.

References

[1] Davide Baccherini and Donatella Merlini. Combinatorial analysis of tetris-like games. *Discrete Mathematics*, 308(18):4165–4176, 2008.

[2] Alexander Bruce. Antichamber. `https://store.steampowered.com/app/219890/Antichamber/`, Jan 2013. Last Access Aug 20, 2018.

[3] Emmanuel Fokides. Digital educational games and mathematics. results of a case study in primary school settings. *Education and Information Technologies*, 23(2):851–867, 2018.

[4] Britton Horn, Amy K Hoover, Jackie Barnes, Yetunde Folajimi, Gillian Smith, and Casper Harteveld. Opening the black box of play: Strategy analysis of an educational game. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, pages 142–153. ACM, 2016.

[5] Richard Kershner. The number of circles covering a set. *American Journal of mathematics*, 61(3):665–671, 1939.

[6] Bobby Law. Puzzle games: A metaphor for computational thinking. In *European Conference on Games Based Learning*, page 344. Academic Conferences International Limited, 2016.

[7] Nora S Newcombe. Picture this: Increasing math and science learning by improving spatial thinking. *American Educator*, 34(2):29, 2010.

[8] Alexey Pajitnov. Tetris. `https://tetris.com/`. Last Access Aug 20, 2018.

[9] Valve Software. Portal. `https://store.steampowered.com/app/400/Portal/`, Oct 2007. Last Access Aug 20, 2018.

[10] Monica Visani Scozzi, Ioanna Iacovides, and Conor Linehan. A mixed method approach for evaluating and improving the design of learning in puzzle games. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, pages 217–228. ACM, 2017.