Michael Weiss, Tahir Peele & Nikolaos Komninos

Introduction

Our group is looking to create a spiritual successor to the game *Peggle* by Popcap / Electronic Arts (EA). *Peggle*, which was initially released in 2007 and received a sequel in *Peggle Nights* in 2008, is a puzzle game released for PC, Xbox 360 and PlayStation 3 inspired by Japanese pachinko machines. The game is simple by nature: the player, who controls a ball launcher, must fulfill the goal of hitting all the orange "pegs" in the level by launching the limited number of balls they are given, with blue "pegs" posing as an obstacle. The easiest way to think of the game is pinball, but certain parts of the board must be hit to win. *Peggle*'s simplicity allows for a lot of flexibility in a potential sequel or spiritual successor, including the addition of a level editor and multiple custom obstacles and abilities - which we plan to follow up on.

Problem Description

Although the video game industry has been said to have boomed multiple times in the last three decades, it has most definitely experienced its greatest growth in the past decade, with the rise of independent developers publishing their own unique ideas and finding great success from it. Despite this, it seemed to us that there's been a lack of distinct physics-based puzzle games that have surfaced during this time. It made us think back to a puzzle game that we love: *Peggle*. As such, the problem we face is as follows: a new entry in the *Peggle* franchise has not been released in over a decade, and not many other major physics-based puzzle games have been released to fill the niche, leaving the market relatively empty. Our solution: to create a new physics-based puzzle game, inspired by and acting as a spiritual successor to *Peggle*, feature the same core gameplay that fans know and love, while also incorporating new features to keep players coming back for more.

This brought about another problem: how could we make our game stand out from the original *Peggle*? The level editor was our primary focus, but feedback we received indicated further differentiation would be required to truly stand out. Whatever the addition may have

been, it was important to consider that the addition in question needed to be unique while also staying true to the formula that made *Peggle* so great. Our primary idea at this time included creating new pegs that alter gameplay upon contact, which can alter how players strategize while playing, forcing them to adapt to the gameplay altering pegs in real time in the event that their launched ball makes contact with said pegs. For example, the original games have white pegs which more or less exist for bouncing and being obstacles, orange pegs which you want to hit to beat the level, purple pegs which give points, and a few others in specific levels; all of these though are ultimately pegs you want to hit and interact with - so we were considering going the opposite route and creating negative pegs you would want to avoid, which may either detract from your score or destroy the ball on the spot. This would keep the gameplay loop similar to what Peggle fans know, but still over a unique twist that changes how you think about each map.

Technical Discussion

The primary topic that needed discussion is the game engine/programming language used to create our game. Our engine needed to be one that was accessible for all three of us to learn within our allotted time frame, as well as one that could effectively and efficiently meet the needs of our game proposal and outline. After passing around a handful of engine ideas, our group decided on using Construct 3 for a handful of reasons. Construct 3 is based in HTML5 and JavaScript, allowing for the game to be fully playable in web browsers. This engine includes a robust Box2D-based physics engine, which handles realistic gravity, collisions, and ball bouncing. This is of course important for a game such as this, which places a strong focus on using the physics of the ball to hit as many of the orange pegs as possible using only that singular ball. The physics system allows for accurate ball movement, making it easier for us to replicate *Peggle*'s trajectory-based gameplay. Additionally, this engine can fulfill our needs of creating a level editor, which will allow for JSON-based drag and drop level layouts, making for easy exporting and importing of levels. As such, Construct 3 seemed to be the best choice for us. Regarding the level editor, we would also like to be able to have users upload their own photos for the stage background as well as sound effects to allow for a truly customizable experience.

Initial Research

Our group was struggling to come up with an idea for a project, so it led to us researching various ideas on games we shared interest in. Once we reached the topic of *Peggle*, our research indicated that the gaming industry has a lack of puzzle/problem solving games that a casual audience can pick up while still learning something, which was something we believed that game series strongly achieved. In order to succeed in our project, it was important we understood what exactly was so compelling to us about the source game in order to work towards creating something that inspires similar feelings. *Peggle* uses a strong feedback loop to reinforce reward based learning as well as strategic thinking - therefore for our successor, we wanted to make sure to capture that feeling of excitement. This feedback loop is accomplished through many factors - a key one of note though is its notable sound design, which gives these uplifting and almost majestic-like sound effects any time something good happens, such as your ball landing into the safe pit and giving you your ball back to fire again with. When a level is completed, the game rewards you with loud celebratory graphical effects and audio, making you feel greatly accomplished for beating even the most basic levels. Therefore, our research has indicated that in order to create a successful feedback loop, and therefore a satisfying game to play, we need to not only meet those needs but go beyond such; we can enhance the positive feedback loop even further via allowing for players to spark their creativity and create Peggle-esque levels of their own to our game, thus providing a potential endless amount of levels and therefore an endless amount of gameplay/replayability, as well other unique features only found in this game. This would not only create a community of players/creators but also inspire other game developers to make more games in the puzzle genre.

Core Requirements

In order for our program to be successful, it must follow a set of functional and non-functional requirements:

Functional Requirements

Gameplay

- The game must have a main menu to choose between playing and editing
- The game must allow the player to launch balls using a launcher
- The game must have orange pegs that need to be hit to complete a level
- The game must have blue pegs that acts as obstacles
- The game must have red (negative) pegs, which destroy the ball on contact
- The game must have a moving basket at the bottom of the level that will provide an extra free ball if shot into
- The game must implement realistic physics-based movement for the balls, including gravity, bouncing and collision detection

Level Editor

- The user must be able to create custom levels using a drag-and-drop interface
- The level editor must allow players to place and remove pegs, obstacles, and other interactive level objects
- The user must be able to upload custom images for level backgrounds
- The user must be able to add custom sound effects to their levels
- The level editor must support exporting and importing levels in JSON format

User Experience

- The game must provide positive reinforcement with sound and visual effects after beating a level (of the celebratory variety)
- The game must reward skill with extra balls or bonus points

Non-Functional Requirements

Performance

- The game must run smoothly in web browsers without significant lag
- The physics engine must remain consistent with the frame rate
- The game must load levels quickly / in a timely manner

Usability / Accessibility

- The game must support all modern web browsers
- The game must be compatible with various screen sizes and resolutions
- The game must have a simple / easy to use interface for both gameplay and level editing
- The game must have simple controls using the mouse

Scalability

- The game should be designed in a way that will allow for updates adding new content to be developed with ease
- The game should be designed so that updates do not break levels created in previous versions of the game

User Stories

In order to visualize how our game might be used, it was important to create user stories:

As a gamer, I want to be able to play a physics based game that is reminiscent of *Peggle* anywhere, anytime with endless replayability.

As a casual gamer, I want a game that provides quick sessions so that I can play in short bursts when I have free time.

As a competitive gamer, I want a fun, replayable challenge that will allow me to go back to try and beat my high scores

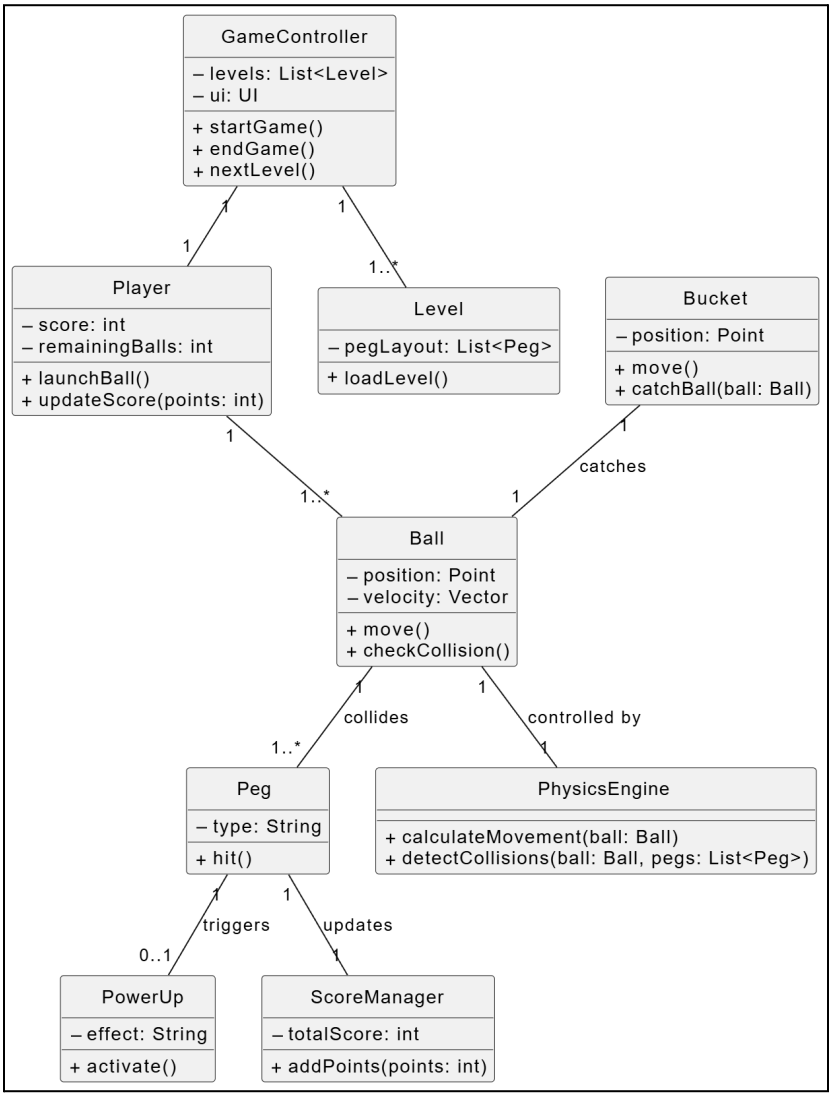
As a content creator, I want to be part of a community where I can design and share levels for others to play, and receive feedback for them.

As a community member, I want to relate to others through the levels we all play and create.

UML Diagrams

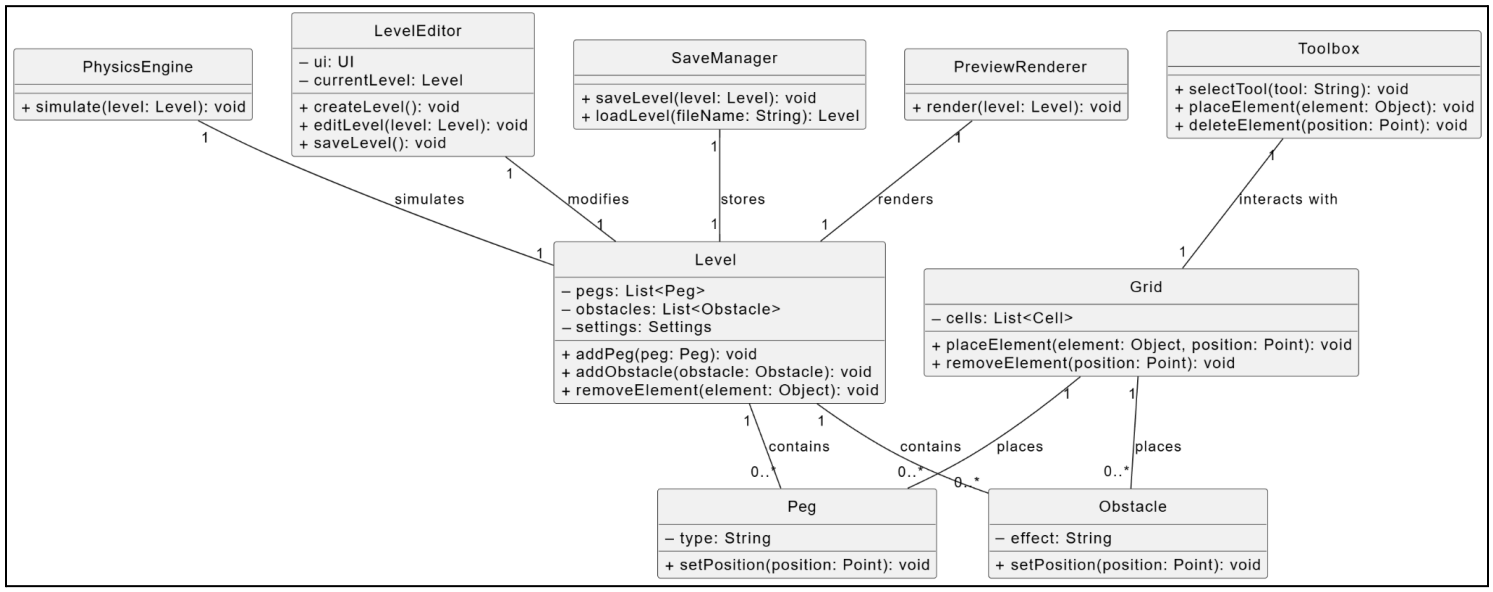
BASE GAMEPLAY:

The GameController manages multiple Levels, controlling the game's progression and UI. The Player launches Balls, which interact with Pegs and Buckets, while the PhysicsEngine handles movement and collisions. The ScoreManager updates the score when pegs are hit, and PowerUps may be activated upon collision. The Bucket can catch balls, and the game continues until all balls are used or the level is completed.



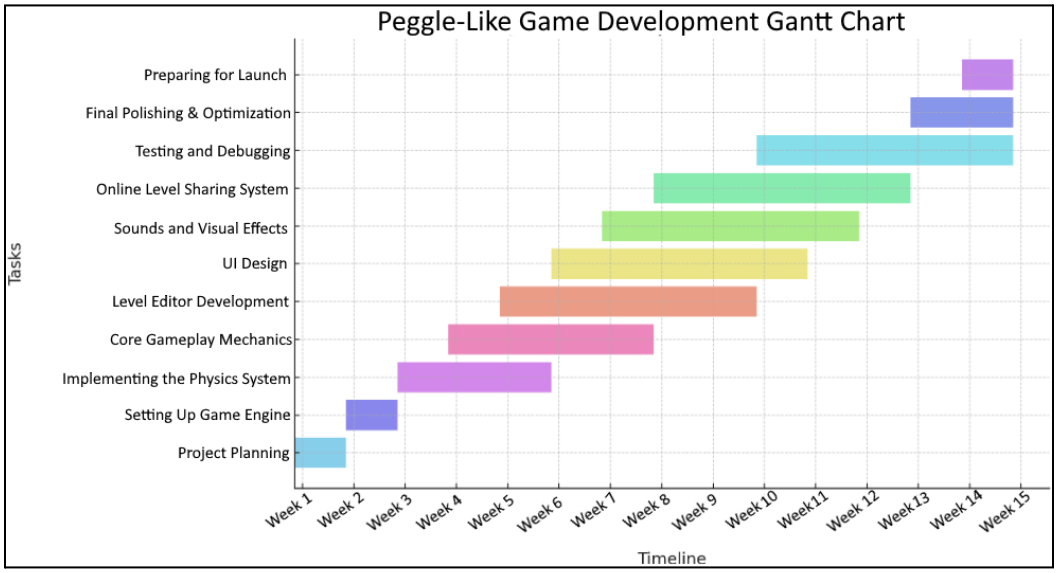
LEVEL EDITOR:

The LevelEditor allows users to create, edit, and save levels, modifying Pegs, Obstacles, and Settings. The Toolbox provides tools for placing or removing elements on a Grid, which organizes level components. The PhysicsEngine can simulate the level's behavior, while the PreviewRenderer visually renders the layout. The SaveManager stores and loads levels, enabling persistence across game sessions.



Schedule

In order to ensure that our project releases on time, a schedule must be followed:



Alongside this, a set of **MoSCoW** requirements must be followed:

Must-Have	Should-Have	Could-Have	Won't-Have
- Accurate physics based gameplay - Pegs to interact with for score - Pegs that are obstacles - Pegs that are needed to win - Pegs that destroy the ball - Moving basket to save ball - Level Editor to create, import and export user-made levels - Add and delete placed assets in Level Editor - Externally exporting and importing user-made levels - Level to play to demonstrate proper gameplay functionality	- Menu to select various levels/layouts for testing - Drag and drop interface for Level Editor - A few levels to demonstrate proper gameplay functionality - Pre-made custom levels made in level editor to show that our game properly loads such	- Placeholder assets (audio/graphics from *Peggle*) - Ability to import/load custom assets (audio, pictures, etc) into Level Editor - Saves score for levels and custom-levels - Unlocking levels via beating the prior level - Bugs that need to be fixed	- Online functionality for importing and exporting user-made levels - Large supply of pre-built levels to play - Fully original soundtrack and graphics, and finished presentation - Peggle*-Masters (characters who alter/change gameplay properties)

Summary

Our group is looking to create a physics based puzzle game in Construct 3 inspired by PopCap's *Peggle* game series. To make our game stand out from the original while still inspiring the same fun gameplay loop, our goal is to introduce unique gameplay mechanics via pegs you want to avoid in a game series where you want to hit every peg possible, as well as a level editor/creator which allows users to share and download fully custom levels providing what theoretically would be endless amounts of gameplay/content. Construct 3 was chosen to accomplish our idea since it is based in HTML5 and has a robust Box2D-based physics engine, which handles realistic the realistic physics we need to make a game like this work - it also runs in web browsers, allowing people a very easy way to play the game. Our research indicated that Peggle's crunchy, loud and exciting audio design and graphical effects upon completing a level aid greatly in forming the intense feeling of satisfaction the game brings, so we aim to use and improve upon this same effect to create a successful positive feedback loop and therefore an enjoyable game anyone can play.

To create our game, we will need to make sure our gameplay (having pegs, balls, and other level objects function as they should), level editor (placing the level objects into a customizable environment and being able to export/import levels remotely), and of course feature proper sound effects, rewards, and so on. Additionally, it is important the game runs smoothly at all times, including within web browsers, as well as maintaining a consistent physics engine regardless of frame rate and load times. It is important to meet the needs of all types of players, whether it be casual gamers, competitive gamers, content creators, level creators, and provide a gameplay experience that caters to all their needs, interests and expectations. To demonstrate the development cycle of this project, we've also created a 15 week mockup Gantt Chart showing what we think our development cycle could look like.

Works Cited:

Arts, Electronic. “About - PopCap Studios - Official EA Site.” *Electronic Arts Inc.*, 20 June 2019, www.ea.com/ea-studios/popcap/about.

“PopCap Games: The Making of Peggle.” *Archive.org*, 2025, web.archive.org/web/20090613055052/www.popcap.com/extras/makingpeggle/big_idea.php. Accessed 30 Mar. 2025.

“Why Is Peggle so Addictive?” *Gamedeveloper.com*, 2023, www.gamedeveloper.com/design/why-is-peggle-so-addictive-. Accessed 30 Mar. 2025.