Final Project Design

Team 15

Team Members - Benjamin Wyss (2898025), Alex King (2207951), Jon Volden (2942501), Tsung Yu Ho (2911531), Jeff Kissick (2414018)

Project Name:

- Dygg

Project Synopsis

- 2-D side-scrolling mobile game featuring procedurally generated worlds, resource management, role-playing elements and exploration, built on the Unity Engine.

Project Description

- This project was chosen because the members of our team would like to gain experience in both game and mobile application development. The end result will be a 2-D side-scrolling game available for download on google play store and app store. The project aims to both entertain users and fill a void within the mobile game industry. We see many games churned out for iOS and Android but very few that are truly free to play and not pay to win. More often than not we see mobile games that appear free on the surface but actually require regular microtransactions, a subscription, or an upfront charge to achieve the full experience. We also wish to make this type of game for mobile devices specifically because it fits best on a mobile device as opposed to a personal computer or console system. Additionally, the genre we have chosen remains largely undeveloped on mobile platforms. We believe that Dygg will be a simple, but fun game to fill this void.

Project Milestones

- Fall Semester 2019
    - Initialize project repository. (9/26/19)
    - Implement base project code, including functionality for character movement and digging within a fixed map. (10/11/19)
    - Document project code (10/25/19)
    - Implement minimal version of game with digging mechanic fleshed out and inventory system working. (11/11/19)
    - Unity Training and Familiarization (12/13/19)
- Spring Semester 2020
    - Add role-playing elements into the game. (3/15/20)
        - Crafting system.
        - Environmental hazards. (hunger, heat, light, etc.)
        - Level-ups/skillpoint system.
        - Additional elements as time permits
    - Implement Boss system. (5/1/20)
    - Deploy application on Google Play Store and App Store (5/8/20)

    - Document deployment features (5/8/20)
    - Implement monetization system (5/15/20)

Project Budget

- Software - Editing software for graphics and audio ($100 - $200) required by the end of the Fall Semester

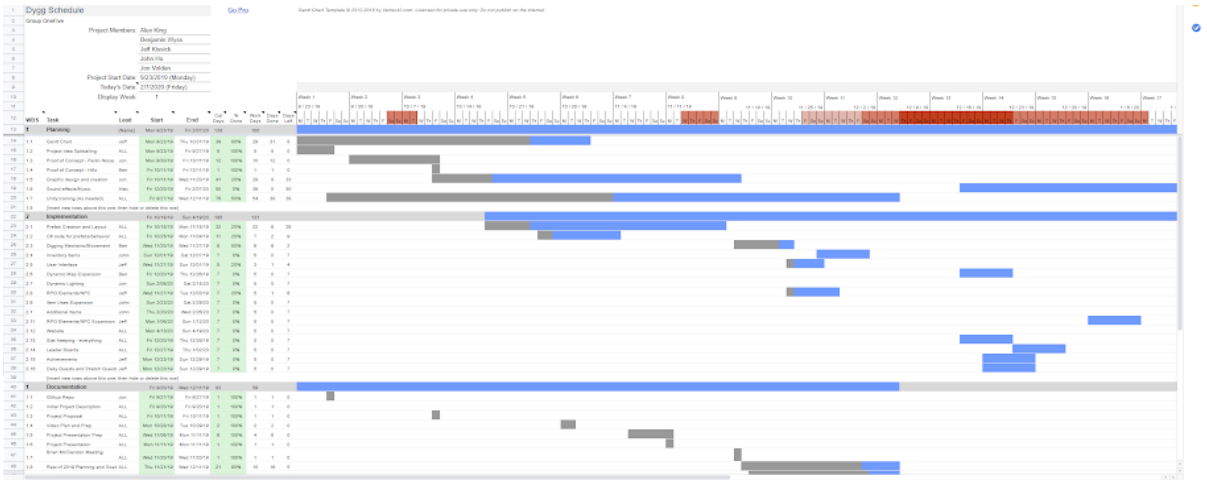
- Vendor - Apple App Store ($99/year), Google Play Store ($25 one-time) required by project deployment time.

- Special - Possibly hiring a game artist ($100 - $200) required by Spring Semester if we do decide to outsource art

Work Plan

- Alex King
    - Designing graphical and audio elements for game
    - Handling Unity scripting for gameplay elements
    - Implementing Mobile Controls
    - Testing application on Android platform
    - Designing and Implementing crafting system
- Benjamin Wyss
    - Implementing back-end code to handle game logic
    - Designing and implementing map generation system
    - Utilizing Unity Engine to connect back-end code to front-end design
    - Testing application on Android platform
- Jon Volden
    - Implementing game logic and design
    - Handling Unity Scripting
    - Testing application on Android platform
    - Designing graphic elements for game
- Jeff Kissick
    - Communicating deadlines
    - Organizing group meetings
    - Testing development, specifically on iOS
    - Scripting in Unity
    - Importing assets into Unity
    - Adding role-playing elements and npc/dialogue elements
- Tsung Yu Ho
    - Handling Unity Scripting
    - Testing application on iOS platform
    - Implementing user interface elements
    - Designing and implementing item and inventory system

Gantt Charts - [Google Sheets Link]



Preliminary Project Design

- Since Dygg is being designed as a two dimensional game that will be deployed on iOS, Android, and potentially Windows, Mac, and Linux platforms, we decided to build Dygg on the framework of the Unity Engine for ease of implementation and portability. The core functionality of the game will have the user controlling a character as they dig through and explore a procedurally generated two dimensional map of tiles. To procedurally generate this map, we will utilize the perlin noise function, which assigns a pseudorandomized value to each tile within our map, and then generate different tile textures and tile functions based on the perlin noise value of the tile's location. Character movement will function by changing the user controlled character's position by one tile at a time depending on the direction that the user presses. If the user tries to move into a space already occupied by a tile, they will have to wait as their character digs through the space first before moving into the tile. Once these core features are implemented, we can refine their design and add details, as well as implement auxiliary features such as resource management, lighting, and role-playing elements. We can refine our map generation process by adding features such as treasure tiles which are generated for rare perlin noise values, procedurally generated locations for predefined towns and structures that make the game world more dynamic, and streams of water which cut through the map. Streams of water can be generated using the A* algorithm, a graph traversal algorithm which finds the shortest path between two points on a graph. The first auxiliary feature that we will implement is resource management. To implement this, we are going to keep track of equipment items that the user's character has currently equipped, as well as non-equipped items that the user is currently storing in their inventory. The user will be able to obtain rocks, clay, ores, gems, and treasure as they dig through the world, and they can exchange the items that they find for an in-game currency which is used to purchase new equipment and upgrade old equipment. Equipped items will directly affect how the user's character performs, impacting things such as how fast they can dig through tiles and how far their lantern will illuminate tiles for them. Lighting will be accomplished through the Lightweight RP lighting system, a two dimensional lighting library for Unity. We are going to implement a lighting system that darkens what the user can see as they dig further

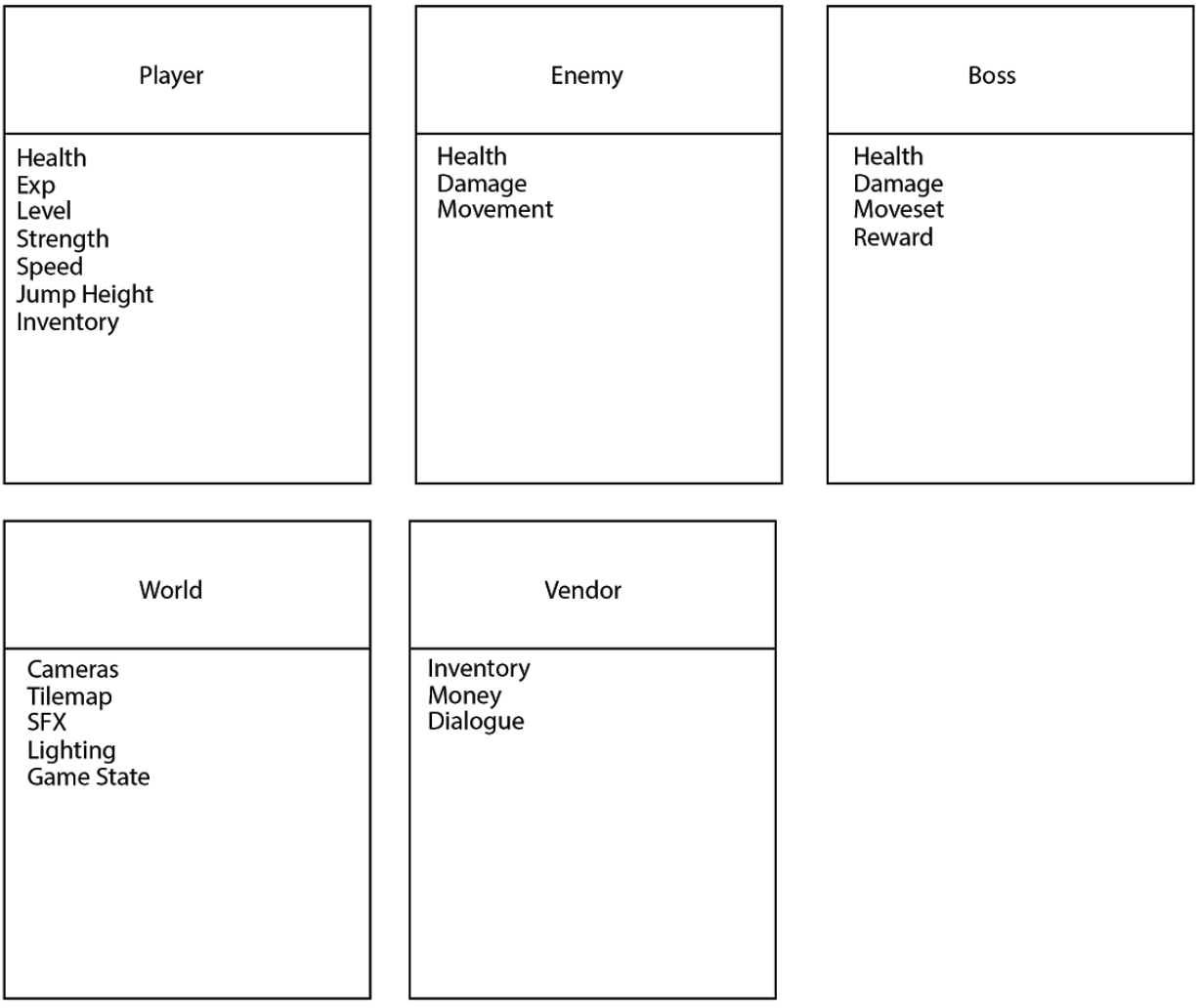
down into the ground, but their character's lantern will light up nearby spaces for them to see. Role-playing elements will be implemented after other auxiliary features are added, and will include a variety of components. One such component is character statistics that can be upgraded as the user's character levels up. This will be implemented by awarding the user with experience points as their character accomplishes specific goals, and then leveling up their character as their total experience points surpass certain thresholds.

- Dygg will feature different dynamic towns for the user's character to interact with and explore. Each town will have one or more shops and different non-playable characters for the user's character to interact with. These non-playable characters will serve to deliver exposition to the user and provide them with quests to complete. Visiting the shop will allow the user's character to upgrade their equipment and sell items, and this is an integral part of the progression of the user's character throughout the game. The items that the shopkeeper sells will be used to unlock different ways to play the game and interact with the world. This will also serve as a way to control what parts of the game the user's character can access, allowing us to gate off portions of the game until the user's character has the necessary equipment and experience to tackle the challenges of different areas of Dygg. As the user progresses through the game, their character will gain experience points along the way. Once the user's character has acquired enough experience points to level up, they can choose to allot skill points towards different character statistics. These character statistics control how fast the user's character can move, how high their character can jump, how much health their character has, and how fast their character can dig through tiles. This leveling system will further serve as a means to incentivise the user to continue playing Dygg. The character leveling system, as well as the shops in different towns, serve as the primary means of progression in Dygg; once the user's character has acquired all items in the game, every area of Dygg should be accessible to their character.
- When considering design constraints, we need to be aware of both technical and business constraints that affect our project. Since this project is mostly an academic project, we will mainly focus on technical design constraints. Given our previous experience in Unity, aspect ratio is one of the main design constraints we will face when porting a game onto Android and iOS systems. Most of these systems support a 16:9 aspect ratio, but there are exceptions that must be accounted for. We need to support an adjustable aspect ratio within our game to ensure that Dygg fits properly on every device's screen. Moreover, in working with mobile phones, we also have to be conscious of how a phone's computing power is often less than that of a personal computer. Knowing this, we will have to design fast and efficient algorithms which build our world of infinite tiles in smaller chunks as the user's character gets closer to unexplored locations. This will ensure that Dygg has smooth runtime speeds on mobile systems. An additional design constraint that we face is the programming language that we will write scripts in. Since we are utilizing the Unity game development framework, we will be required to use C# to create scripts for Dygg. This will require our team to take some time and become familiar with C# to effectively write scripts for Dygg. Concerning the limitations of Unity's framework,

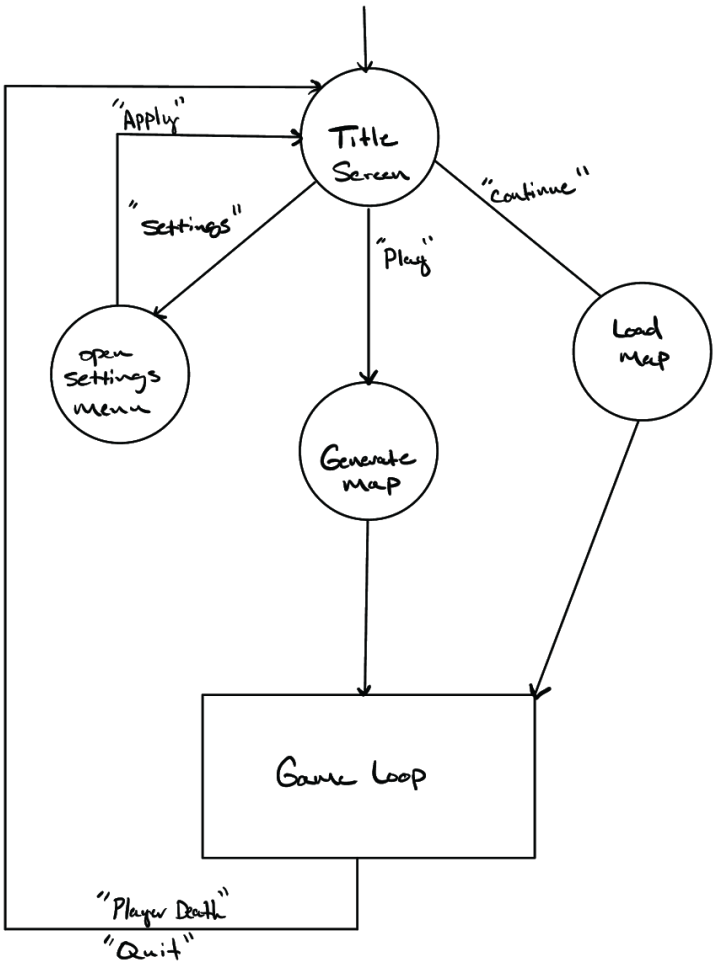
Unity supports libraries and extensions, so we will be able to utilize these libraries to aid us in either adapting to or overcoming framework limitations that we face.

- Regarding business constraints, the monetization system which Dygg utilizes will be constrained by the application stores which we distribute our game on. The forms of monetization that is supported by both the Google Play Store and the Apple App Store are application purchases, in-game purchases, and advertisements. Since we want to make a game that is freely available to download, Dygg's monetization system will focus on in-game purchases and advertisements.

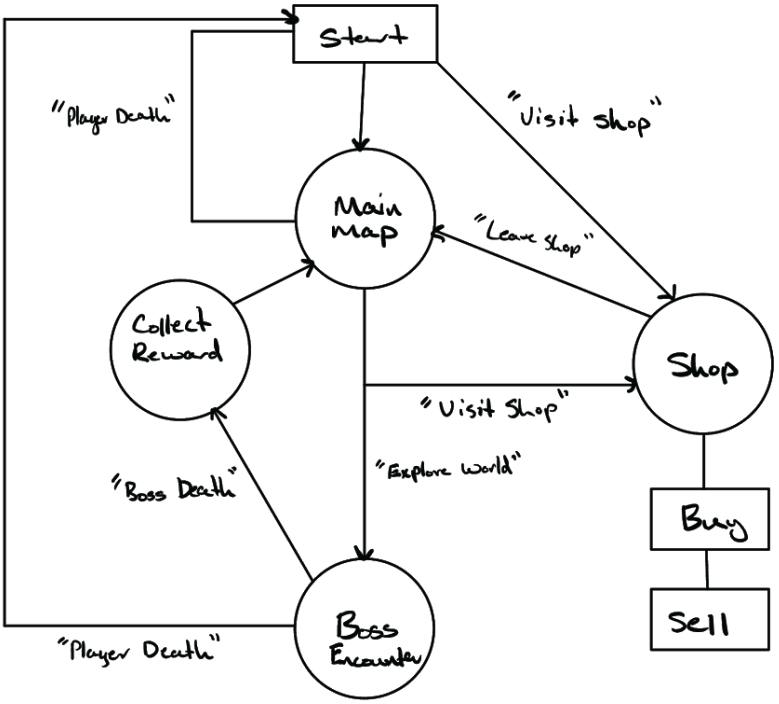
Class Overview



Game Overview Diagram



Game Loop



Ethical Issues

- Even though Dygg will be a mobile game that is freely available to download, we as developers have the potential to access, store, and analyze user data. Dygg will not store any personal information about its users; the only data which it will store is local information related to game saves, character progress, and application settings. Since this information will be stored locally on users' devices rather than remotely on a server, users will not have to worry about any of their game information being compromised. Furthermore, as a mobile application, Dygg could potentially be monetized to make a profit off of selling in-game items and running advertisements. With this potential ability, Dygg will not utilize predatory monetization or mandatory purchases, but Dygg will instead focus on optional purchases and non-intrusive advertising.

Intellectual Property Issues

- While other games exist that utilize mechanics similar to Dygg, our inclusion of role-playing elements and our development for mobile platforms separates Dygg from other games within the same genre. Dygg will offer a unique experience that differs from other mobile games since it will largely focus on exploring an infinite world rather than completing a finite set of static levels. Additionally, Dygg will exclusively utilize artwork and assets either created by our team or purchased and licensed for our team. This will allow us to deploy Dygg on the Google Play Store and Apple App Store without worrying about using copyrighted content.

Change Log

- Gannt chart updated to more accurately match our work timeline and also to display who works on what aspect of the project.
- Project milestones dates updated for accuracy.
- Updated work plan to more accurately reflect what each team member has worked on and will later work on.