Boundless Blade

TECHNICAL PLAN

- BY JOEY CLEMENTE -

TABLE OF CONTENTS

IMPLEMENTATION DIFFICULTY SCALE

DOCUMENTATION SCALING

- This scale will dictate how easy to hard programmer implementation was for specific individual systems/mechanics, etc.

- Temporarily, the scale will just be a scale system from 1 to 5.

DIFFICULTY AMOUNT	WHAT DOES THIS RATING ENTAIL?
1/5	**Very Easy -** not hard to implement at all.
2/5	**Easy -** Not hard to implement, but might take a bit more time than usual.
3/5	**Medium -** Took a bit more time and planning to make happen, however I could still execute.
4/5	**Difficult -** Takes a lot of time, planning, and outside research to achieve.
5/5	**Extremely Difficult -** Would consult numerous other people, would take loads of time, planning, and research, and even then it might not get done.

DELIVERY PLATFORM

MAIN PLATFORM: COMPUTER - 2/5

The biggest and most important thing to work with about this delivery platform is thinking about implementation with how it would work with alternate input devices such as drawing tablets (primarily) as well as touch screen. Making sure that the controls work properly and cohesively with the drawing tablet and touchscreen or mouse on any device is a vital portion of our control scheme that needs to be right. This platform allows us to get easy-to-use input up and running super quickly. Allowing access from any computer that has access to a keyboard, mouse, touch, or drawing tablet is the goal for our delivery platform.

DEVELOPMENT ENVIRONMENT

UNITY - 2/5

The main portion of the development is done in the Unity Game Engine. It is a powerful free engine that allows developers to make very well-made and designed games with very easy-to-use user interfaces and an easy-to-use scripting language in C#. It is a powerful combination of systems that allows the developers to save a lot of time with all the nitty gritty and focus on what matters most, actually creating systems and art for the game itself.

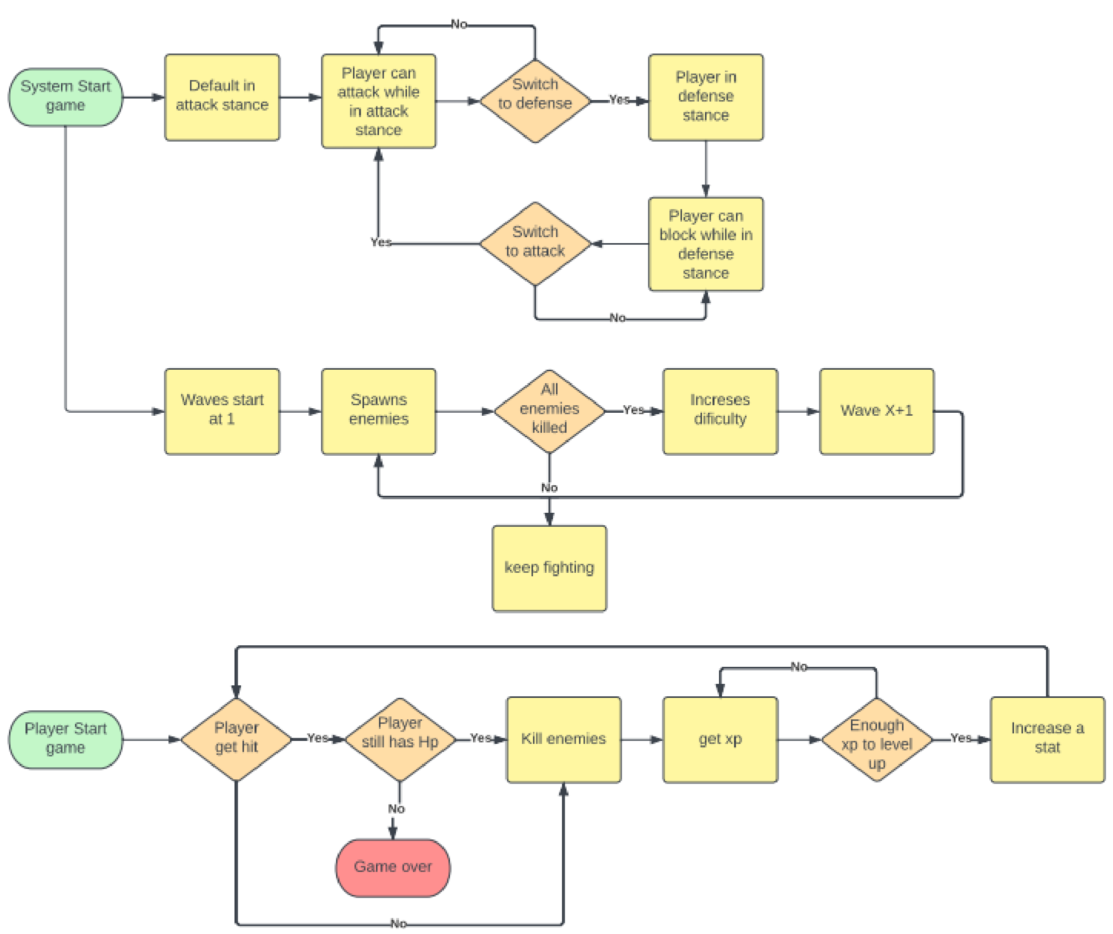
PHOTOSHOP - 2/5

All of the art is being developed in the Adobe software, Adobe Photoshop. Photoshop is one of the most used programs in the world for graphics design and art. In this instance, we will be using it to make game assets and animations. It is an extremely powerful tool that really allows the artist to unlock their full potential and create some wonderful-looking art assets, just like Tu and Olivia did (shoutout Tu and Olivia for the art).

VERSION CONTROL - 2/5

A majority of version control is either handled in gitbash or gitkraken. Both programs revolve around the popular git software that makes version control, especially in game development relatively simple and straightforward. We have the occasional merge conflict here and there, but the simplicity and easily accessible gitkraken allow us to stay on track and not have to worry that much about git and it’s limitations and functionality.

LOGICAL FLOW DIAGRAM



Game Mechanics and Systems

Systems

Wave Management - 2/5

- Enemies will attack the player, coming at them from all directions in a wave based pattern, where each successive wave contains larger quantities of powerful enemies that the player has to clear out of the main level. Spawners will be scattered throughout the main level that will instantiate enemies at a consistent pace.
- A majority of the wave management will take place in a Wave Manager which will allow for designers and programmers to fine tune any values that would need to be changed during testing periods, it will have the ability to change spawn speeds, amounts, enemy types, etc.

Enemies and Typing - 2/5

- Enemies themselves will lock in on the player's location and follow them at a speed either equivalent to or less than the player's base movement speed so that players won't get easily overwhelmed.
- Two different enemy types will be involved so that there is some variety to the game play and so that the player has some reason to block using their weapon. One type of enemy will just walk towards their location, and the other can have a ranged attack, forcing the player to be more vigilant about their surroundings, and so that the block move can be useful.
- Enemies will gradually walk towards the player from all distances, however, for the ranged unit, there is an attacking range before they stop to attack the player.
- The enemies will have changeable radii and movement speeds that will be very accessible to everyone in the engine which can be used for fine tuning.

PLAYER LEVELING - 3/5

- Once an enemy is defeated, you will automatically gain exp which will be added to an exp bar. Once filled, the game will briefly pause and allow for the player to select an ability upgrade from a pop up window. After that selection has been made, the player will reach the next level, which will cause the next wave to start
- The leveling system will allow you to increase your stats as you progress through the game, how that will be facilitated will be through a Progression Manager that will be publicly available to the designers and programmers to make sure the game is balanced properly and can be enjoyed on both ends of the player/developer experience.

FORGES - 3/5

- Three possible upgrade cards can be chosen by the system to display on player level up. The system goes through one by one to select each of three cards by using the rules of 80/20 as stats vs ability upgrades. Stat upgrades are used to increase the player's health, damage, movement speed, etc. Ability upgrades grant the player newfound powers that modify the player's mechanics. They are either passive upgrades or have cooldowns. They affect how strategies can be made in ways such as damaging enemies or healing.

MECHANICS

PLAYER MOVEMENT - 1/5

- Movement is based around the actuation of the "WASD" keys on the keyboard, these keys are set in stone currently and can be used in multiple different variations to allow the player to use all 8 directions including diagonals.
- This type of movement allows the player to explore the level, defeating enemies and developing strategies along the way. For example, the player may utilize their free movement to group the enemies up to quickly take them all out at once.

DRAWING TO ATTACK OR BLOCK - 2/5

- User input for the implementation of this mechanic is supposed to be through a drawing tablet of some kind. The player will use the actuation of the drawing tablet's pen or touch screen to interact with the drawing itself. The input can also be used in a traditional mouse setting but is not the primary or proper way to play.
- This mechanic allows the player to freely attack enemies without being restricted by a single mouse click. It also limits the use of multiple complicated controls so that the mechanic may be used easily. Many different types of strategies and play styles may adapt through this type of attacking/blocking, as it is basic enough but at the same time as customizable as the player wants.

CYCLING - 1/5

- Player input using the key "F" on their keyboard allows for the player to cycle through both of these different modes.
- This allows the player to switch between offensive and defensive playstyles at a moments notice in an effort to ease the difficulty of the control scheme. Furthermore, it also allows the player to utilize the drawing mechanic for both offense and defense as well.

ART PIPELINE

All assets will be made in Adobe Photoshop.
Finished assets will be exported as PNGs.

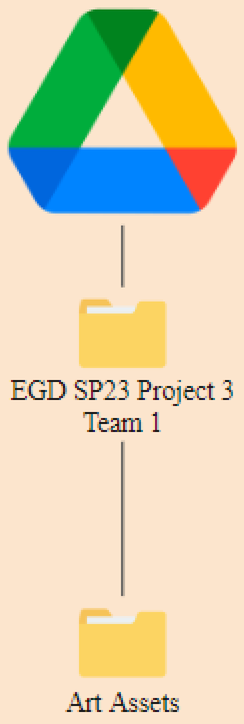
FILE NAMING CONVENTION

Assets will be named in Camelcase with asset type followed by name.

For Example:
"typeName.png"
e.g. characterPlayer.png
propPillar.png

Asset Types include:
- Environment
- Prop
- Character
- UI

FILE DIRECTORY



Google Drive:
Art assets are uploaded to Google Drive.

"EGD SP23 Project 3 Team 2" folder → "Art Assets" folder

This file directory acts as **backup**, as well as a space for other team members to quickly access the assets for documentation if needed.

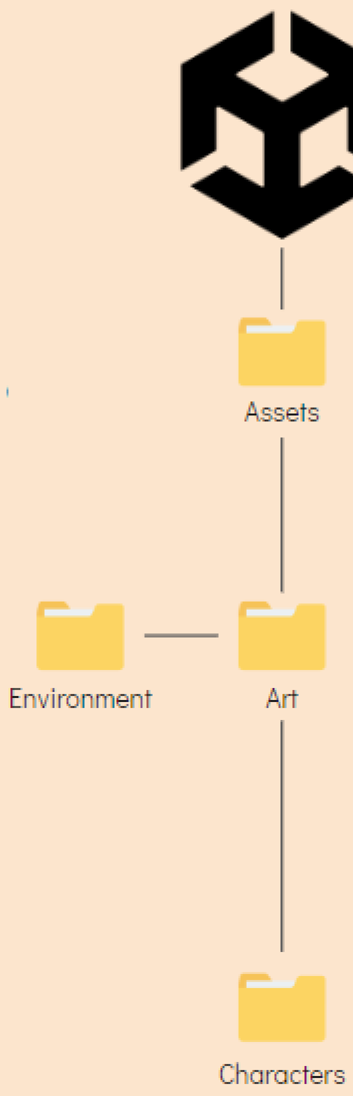
ART PIPELINE IMPLEMENTATION

UNITY

Artists should have project repository cloned on their device. When assets are finished and ready to be implemented into the game, the artist will pull from the repository through **GitKraken** for the most recent version of the project.

From there, the artist will place necessary art assets into the Unity project.
All asset PNGs should be placed in the folder **Assets** → **Art**, and then into their corresponding subfolder based on asset type (i.e Environment, Characters).

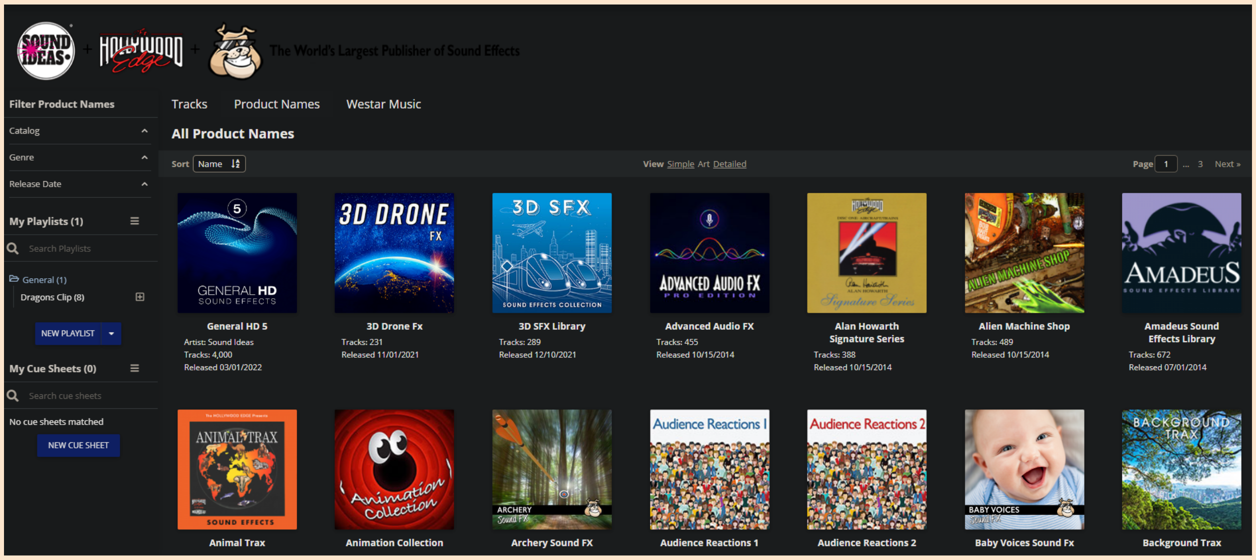
Once in the Unity project, artists will use Git to stage changes, commit, and push to the remote repository. Art assets are then assembled/implemented into game scenes as necessary by **designers and programmers**.



TILESETS

Tile palettes will be created by artists in Unity. Designers and programmers may assemble levels using the tileset as necessary.

Audio Pipeline

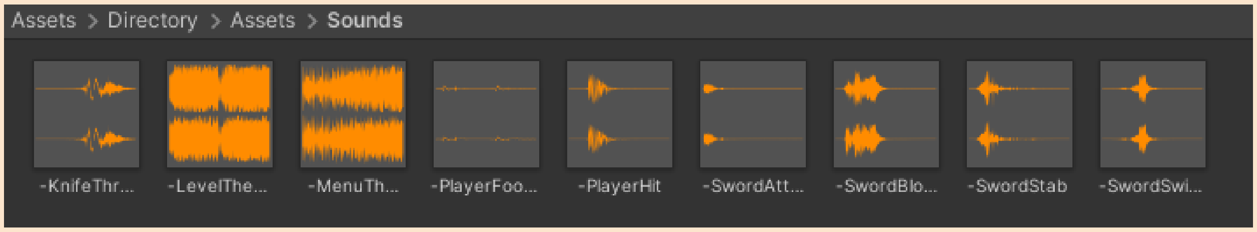


Audio is either found online in our databases or using software such as JFXR or Garageband to create sound effects or create music for our project.

Music is either made or found through a legal source such as the Champlain College library sources like the one we used *Sound Ideas* as our main source.

Unity

All music and/or SFX are in the .wav format and are to be put in the designated audio folder for use. These folders may be expanded outwards depending on the different types of sound effects or music that is needed. All music and SFX has to use the "-FileName" formatting for implementation later on.



Example Asset Path: "Assets/Directory/Art/Sounds" for all of the different sound effects that are used.

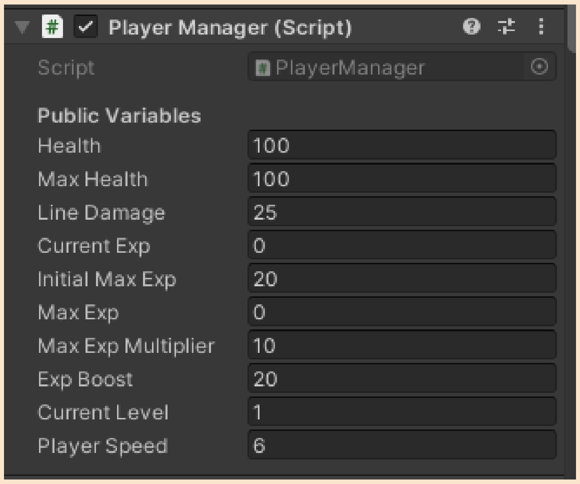
DESIGN PIPELINE

GENERAL COMMUNICATION

Designer's communication really depends on coordination between artist and programmers as well as how the research that has been done cohesively works with all 3 disciplines. If there are any risks or problems, the designer would message the artist or programmer on mattermost to discuss the situation and either wait for a solution or start working toward a solution with them to fix the problem.

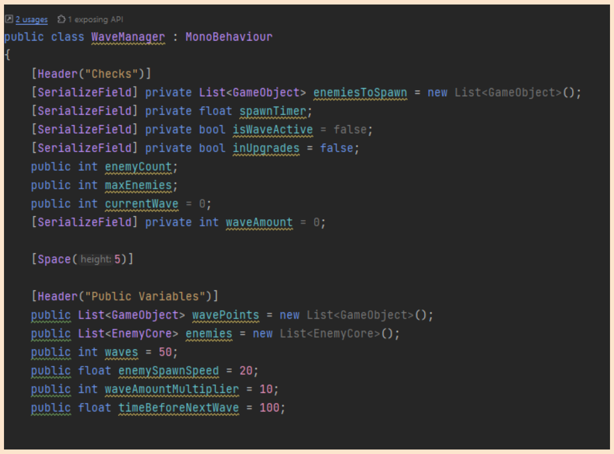
PUBLIC VARIABLES + GAME MANAGER

The Game Manager is a singleton set up for data persistence. There are a lot of already set up tasks that the game manager object initially does. Things such as waves, player management, line management, ui management, upgrades, and everything in between are located in the Game Manager and all have checks and public variables spaces for the designers and artist to use for implementation.



Public Variables

Health	100
Max Health	100
Line Damage	25
Current Exp	0
Initial Max Exp	20
Max Exp	0
Max Exp Multiplier	10
Exp Boost	20
Current Level	1
Player Speed	6



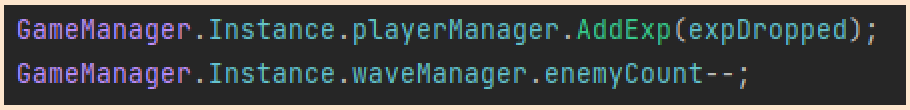
```
public class WaveManager : MonoBehaviour
{
    [Header("Checks")]
    [SerializeField] private List<GameObject> enemiesToSpawn = new List<GameObject>();
    [SerializeField] private float spawnTimer;
    [SerializeField] private bool isWaveActive = false;
    [SerializeField] private bool inUpgrades = false;
    public int enemyCount;
    public int maxEnemies;
    public int currentWave = 0;
    [SerializeField] private int waveAmount = 0;

    [Space( height: 5)]

    [Header("Public Variables")]
    public List<GameObject> wavePoints = new List<GameObject>();
    public List<EnemyCore> enemies = new List<EnemyCore>();
    public int waves = 50;
    public float enemySpawnSpeed = 20;
    public int waveAmountMultiplier = 10;
    public float timeBeforeNextWave = 100;
```

If one would want to access the contents (variables, etc.) of the Game Manager to make something new or change something old, since it is a static Singleton, the process of grabbing information always starts with "GameManager.Instance.*Whatever_data_needed*

Public variables vary throughout the entirety of the editor and scripts, but a majority are located in the Game Manager, there are some public variables that would need to be changed for balancing for something such as the enemies health, movement speed, etc. which are all located on the enemies themselves.



```
GameManager.Instance.playerManager.AddExp(expDropped);
GameManager.Instance.waveManager.enemyCount--;
```

Milestone 1

Deliverables

- Pick a Game Concept
    - Pick a concept we are interested in prototyping.
- One Page Visual Design Document
    - Guide for core design.
- Systems and Mechanics List
    - Shows all initial major mechanics and their functions.
- Art Concept Documents
    - Start developing art direction and aesthetic.
- Basic Prototype
    - Have a basic prototype for presenting our newly made concept.

Goals for Next Milestone

Create a fully-functioning prototype that has the ability to demonstrate the core mechanics, which will set the base for our design and testing in the future.

Milestone 2

Deliverables

- Fully Functional Prototype
  - Prototype with the game's mechanics implemented.
- Art Direction + Initial Concepts
  - Art Direction picked + some initial color palette and art concepts (if applicable).
- System Development
  - Talk about and outline new systems and how to create them for the prototype in place as well as start working on them if time is available.

Goals for Next Milestone

Polish the core mechanics as well as start implementing core art ideas and new systems to expand the game's scope and increase user engagement.

Milestone 3

Deliverables

- Upgrade System
  - Create the game’s most complicated system.
- Work on Art Assets
  - Work on initial assets for players and enemies as well as the environment.
- Balancing
  - Talk about game balance and how everyone will focus on different aspects of balancing so that it isn’t too overwhelming.

Goals for Next Milestone

Focus on game feel and balancing that allows the player to have a better experience playing the game while also implementing systems such as menus to make the game more cohesive.

Milestone 4

Deliverables

- Active Upgrades
    - Focus on designing and implementing the active upgrades so that there is a better variety to the upgrade system.
- Improved Art Cohesion and Animations
    - Work on making sure scaling amongst art assets is true as well as work on animations for the player specifically.
- Menu System
    - Design and implement the menus for the game to give the player the ability to actually engage with the game outside of the gameplay itself.
- Balance
    - Really go all out to focus on balancing the newly made active upgrades as well as the other upgrades so that the game isnt a total mess to play.

Goals for Next Milestone

Focus on game feel, the main gameplay loop is done and relatively polished, keep working on polish and improving the smaller things so that the game continues to feel better.

Milestone 5

Deliverables

- Rework Major Systems
    - Our main focus is on reworking the block and upgrade systems so that they are more cohesive for player use.
- Animations
    - Get a majority of animations in-engine for players and enemies.
- Polish
    - Add polish in all aspects of the game to increase game feel and make the player feel more immersed in the experience.
- Balance
    - Balance the changes with the new blocking system while also tuning and testing upgrades to see cohesiveness with new upgrade reworks and new active upgrades.

Goals for Next Milestone

Polish, polish, polish. Test and balance everything we can while getting in the final things we need to get ready to pitch.

Milestone 6

Deliverables

- Polish
  - Polish everything. Get everything ready to ship.
- Menus
  - Make sure the menus are working and bug free to enhance user experience as well as mesh with the new UI art.
- UI Art
  - Get all of the UI art in the game and make it cohesive with the general feel of the game as a whole.
- Balance
  - Balance the changes with the new blocking system while also tuning and testing upgrades to see cohesiveness with new upgrade reworks and new active upgrades.

Goals for Next Milestone

N/A