Future Business Leaders of America

COMPUTER GAME AND SIMULATION PROGRAMMING

Sophie Tian
Hanlin Zhang

TABLE OF CONTENTS

GAME SUMMARY

Jonathan's Adventure is a 2-D platformer adventure game. The player follows the main character, Jonathan, on his exploration inside a mysterious ruined temple to find treasure.

This game is created to cater towards an audience of all ages, but especially towards the age range of 10-18 years old. The purpose of the game is to strengthen the player's problem-solving and fine motor skills by solving the room puzzles and defeating various enemies throughout the gameplay.

For the project scope, we aim for the game to be released to the public in the near future (fall of 2022) on the Google Play Store. Our budget is also aimed to be as low as possible; to no expense if possible. The game is set to take around a year total to complete and publish to the public.

Link to the user manual: https://tsophie2015.wixsite.com/mysite/home

GAMEPLAY

The game has two gamemodes: singleplayer and multiplayer, both with different objectives.

The objective for the singleplayer game is to pass through all the rooms in the temple by finding all the hidden artifacts. The number of artifacts needed for each room will depend on the number of frames next to the door where the artifacts will go. Snakes, when the player jumps on them, will also drop artifacts if present in a room.

Once the player completes the rooms, there is a final boss level with a boss snake. The player must kill all the snakes that the boss snake spawns in each round without falling in the lava or getting hurt from the snakes. Then, once the player survives all three rounds, the boss snake will fall off and a doorway will appear.

The player will then beat the game and earn a spot on the singleplayer leaderboard with a score, factoring how long the user took to beat the game (the faster, the higher the score).

For multiplayer, two people share the keyboard to play two characters. This part of the game is only the boss level, but endless. They must work together to beat each round of the boss level. The longer the duo survives, the better the score on the leaderboard.

Every player has 5 lives for both gamemodes.

PLANNING

Software Selection

For software selection, we required that the software be simple, relatively easy to use, and versatile. Starting with fundamental game development, our team chose between Windows apps (Java Jframe, C# Windows), online game development sites (Scratch, Code.org), and game engines (Unity, Unreal Engine).

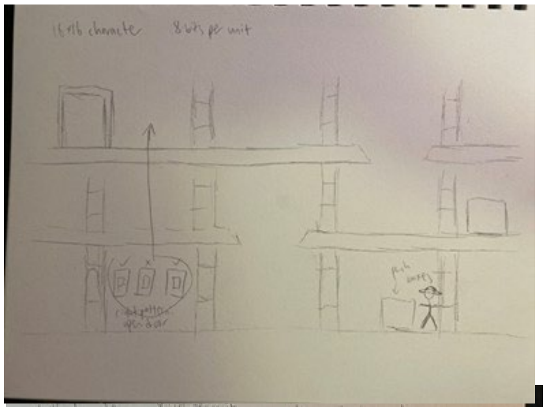
We decided to choose Unity Engine, due to its efficient graphic interface, high versatility, its compatibility for C# scripting, and its user-friendly interface (compared to other options). Unity's ease to create 2-D games and the various resources on how to use the program made the engine the deciding factor for us to utilize it.

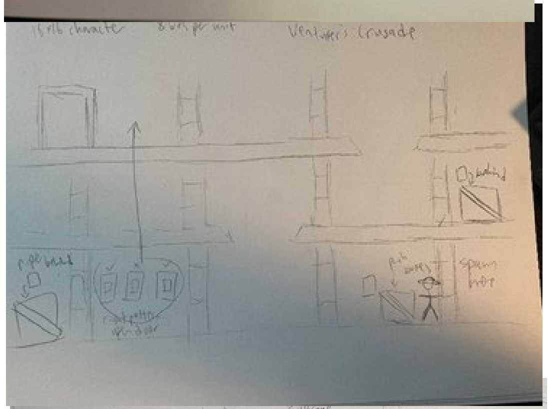
Other software used to facilitate game production were selected for convenience and versatility purposes, due to us already having the proper resources on hand. This includes Procreate (an illustration app on the iPad to create sprites) and Photoshop (for asset editing).

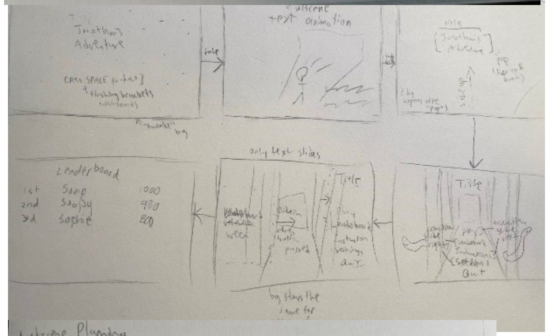
Hardware Selection

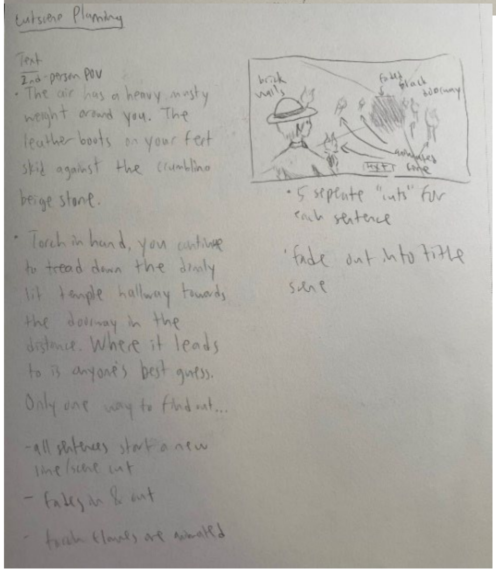
Like software selection, hardware selection was based off existing resources we had to our disposal, as well as how familiar we are to utilizing it. Our team mainly used a ROG Zephyrus G14 laptop (for scripting), iPad Pro 9.7 Inch, and Apple Pencil 1st Generation (both for graphics development).

Planning Process/Documents








Cutscene Planning

Text
2nd-person POV
- The air has a heavy musty weight around you. The leather boots on your feet skid against the crumbling beige stone.
- Torch in hand, you continue to tread down the dimly lit temple hallway towards the doorway in the distance. Where it leads to is anyone's best guess. Only one way to find out...

-all sentences start a new line/scene cut
- Fades in & out
- torch flames are animated

- 5 seperate "cuts" for each sentence
- fade out into title scene

PLAN OF WORK LOG

Date	*Task*	*People Involved*	*Comments*
12/25/21	Initial Start of Game	Sophie Tian Hanlin Zhang	Created the Unity project with a basic player
12/29/21	Movement of Character Sprite, Ability to Close Game	Hanlin Zhang	Added horizontal movement and jumping to a temporary player sprite Scripted as to make the game close out
12/30/21	Created basic UI	Sophie Tian Hanlin Zhang	Added a basic main menu UI and assets
12/31/21	Began on Level One	Hanlin Zhang	Started scripting the start of level one (ground check, adding platforms, etc.)
1/1/22	Player Animation	Sophie Tian Hanlin Zhang	Added player animation frames (sprites)
1/3/22	Code Mechanics for Level One	Sophie Tian Hanlin Zhang	Uploaded brick sprite and created code mechanic for Level One
1/4/22	Adjusted Main Menu GUI	Sophie Tian	Added more sprites and edited the main menu
1/7/22	Started Artifact Mechanic, Finished Level One	Sophie Tian Hanlin Zhang	Began the artifact mechanic for the levels and added the sprites for it Level One is complete and “winnable”
1/8/22	Began on Level Two	Hanlin Zhang	Started and completed work on Level Two
1/9/22	Edited UI Graphics, Miscellaneous Changes	Sophie Tian Hanlin Zhang	Edited the main menu UI and edited the button graphics

1/10/22	Added Game Sounds and Started Boss Level	Sophie Tian Hanlin Zhang	Added various audio files and started on the boss level
1/30/22	Polished Main Menu Code	Hanlin Zhang	Cleaned up and commented code for the main menu
2/5/22	Finalized Working Singleplayer	Hanlin Zhang	Completed updates on the game managers and revamped the file management system
2/7/22	Added README	Sophie Tian Hanlin Zhang	Created and updated the README file
2/8/22	Refactoring Code	Hanlin Zhang	Bug fixes, miscellaneous additions (death animation, congratulatory messages)
2/10/22	Revising Graphics	Sophie Tian	Imported edited sprites and reworked in-game graphics
4/7/22	Began Cutscene, Revamped File Manager	Hanlin Zhang	Began scripting for the cutscene and revised the game file manager
4/15/22	Imported Cutscene Files	Sophie Tian Hanlin Zhang	Created and imported cutscene graphics and finished work on cutscene
6/1/22	Inserted Multiplayer Assets and Sound Files	Sophie Tian	Added sprites for Player 2 and various audios
6/3/22 - 6/25/22	Program Optimization and Code Refactoring	Hanlin Zhang	Extended work on optimizing game performance
6/18/22-6/24/22	Various Animation Additions and GUI Improvements	Sophie Tian	Extended work on revising graphics, assets, GUI, etc.

USER INTERFACE

The main goals for the user interface center around maintaining a simplistic and user-friendly design, as to prevent visual clutter and make the game beginner-friendly to use.

We mapped out the various panels and branches a user can access, specifically for the main menu, and made sure the panels are easily accessible and simple. For example, our main menu first allows the user to choose between a singleplayer and multiplayer gamemodes, with the various panels (e.g., instructions, leaderboard) separated into those two game types.

The usage of various design principles also played a major role in the aesthetics of the user interface, ensuring that the foreground is distinct from the background, utilizing balance and alignment, etc.

PROGRAMMING

Language

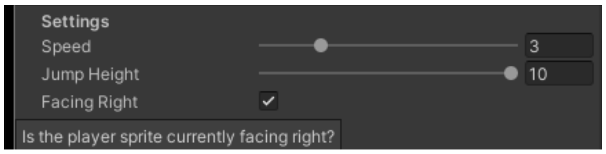
The entire game was scripted with Microsoft's .Net 4.6 framework via the C# 8.0 language. All assemblies were compiled through Roslyn.

Process

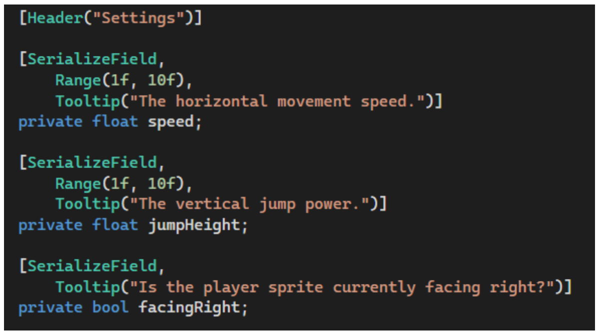
Our programming process came in four stages: planning, scripting, debugging, and optimization. For planning, we preferred to use pseudocode over other methods such as flowcharts. After that, we scripted with Microsoft's Visual Studio Community 2022 IDE. During debugging, we used multiple tools such as the Unity Editor's play mode and VS's debugger. Finally, we used Unity Editor's Profiler tool to monitor CPU usage and RAM allocations to maximize performance. During this phase, we also added comments.

Comments and Documentation

Our comments and programming documentation follow two industry standards: Unity Attributes and Microsoft's XML Documentation Comments. We chose these because they both provided numerous benefits. Unity Attributes allow us to modify settings without having to dive into code. See example below.



Settings
Speed 3
Jump Height 10
Facing Right ✓
Is the player sprite currently facing right?



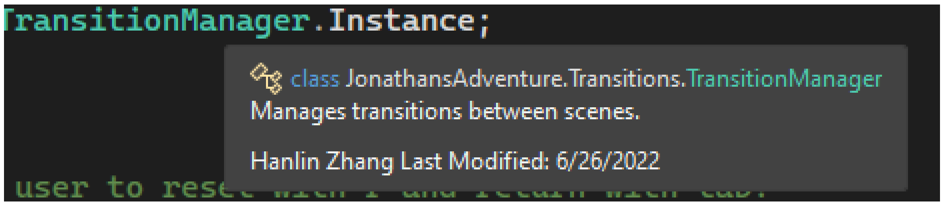
```
[Header("Settings")]

[SerializeField,
    Range(1f, 10f),
    Tooltip("The horizontal movement speed.")]
private float speed;

[SerializeField,
    Range(1f, 10f),
    Tooltip("The vertical jump power.")]
private float jumpHeight;

[SerializeField,
    Tooltip("Is the player sprite currently facing right?")]
private bool facingRight;
```

XML comments help create documentation for the code and help the IDE understand the code.



```
TransitionManager.Instance;
```

class JonathansAdventure.Transitions.TransitionManager
Manages transitions between scenes.

Hanlin Zhang Last Modified: 6/26/2022

Program Modules

During the process of developing the game, we relied on three modules: UnityEngine, TMPro, and System. UnityEngine is a namespace provided by the Unity Editor that contains important runtime scripts. TMPro is another namespace provided by Unity for managing UI text. Finally, System is a namespace by Microsoft with many fundamental components of both the .Net framework and C# language.

Program Structure

Jonathan's adventure is split into multiple assemblies under the root JonathansAdventure namespace. The assemblies are as follows: Core, Cutscene, Data, GameBoss, GameEnvironment, GameFoundation, GameMulti, and MainMenu. Each assembly contains multiple Classes, Enums, Interfaces, and more that make create the functionality behind Jonathan's Adventure.

All programming components of Jonathan's adventure can be found on our programming documentation website at https://periwinklestudios.github.io/.

ASSETS

Fair Use

All sprites/visual assets are original and created by our team. Any other assets, such as fonts and audios, are properly cited below.

"Jump" sound effect made using Jsfxr 8Bit Sound Maker (website: https://sfxr.me/) by Eric Fredricksen under CC0/public domain.

"footstep02" (walk sound effect) created by pepingrillin on Freesound (website: https://freesound.org/) under CC0/public domain.

"Rocks" (door open sound effect) created by adamgryu on Freesound (website: https://freesound.org/) under CC0/public domain.

"8-Bit Bossfight" (multiplayer music) created by Volvion on Freesound (website: https://freesound.org/) under CC0/public domain.

"8 Bit Video Game Fight Music" (main menu music) created by bone666138 on Freesound (website: https://freesound.org/) under CC0/public domain.

"8-bit Text Scroll Sound" (typing sound effect) created by EVRetro on Freesound (website: https://freesound.org/) under CC0/public domain.

"Dramatic tension loop" (cutscene music) created by BloodPixelHero on Freesound (link: https://freesound.org/) under the CC BY 4.0 license (license link: https://creativecommons.org/licenses/by/4.0/).

"Retro arcade music #3" (boss level music) created by BloodPixelHero on Freesound (link: https://freesound.org/people/BloodPixelHero/sounds/596525/) under the CC BY 4.0 license (license link: https://creativecommons.org/licenses/by/4.0/).

Other background music and sound effects are created by Mixkit (website: https://mixkit.co/) under their Sound Effects Free License for personal and commercial use.

"Retro Gaming" (font) by Daymarius on https://www.dafont.com/ is under personal and commercial use.

"Karmatic Arcade" (title font) by Vic Fieger on https://www.1001fonts.com/ is licensed under the 1001Fonts Free For Commercial Use License (FFC).

Sprites, Sounds, and Fonts

Our team aimed for a unique and indie-style visual for the entire game, and the best way to achieve this was to hand-create the majority of the visual assets. By achieving this, we can personally tailor the assets however needed, and reduced the need to use many copyrighted materials. Overall, this choice greatly benefitted our game development by allowing fine detail control to perfect every aspect of the game.

Any other assets that cannot be hand-created (e.g., sounds, fonts) were ensured to be under the public domain or under a CC license.