P1: The Legend of Zelda

EECS 494, University of Michigan, Ann Arbor

Winter 2016

1 Introduction

The Legend of Zelda was created for the Nintendo Entertainment System in 1986 by Shigeru Miyamoto's team at Nintendo. Inspired by its creator's childhood wanderings, the game features top-down, multi-room exploration, sprinkled with combat and puzzle solving.

The purpose of this document is to highlight the most critical aspects of an authentic recreation. This list is not all-encompassing– there are plenty of smaller fish you must fry– but these are the essentials for a solid P1 score.

2 Critical Features

2.1 Grid-based movement

There's a unique, easily missed aspect of the player's movement in The Legend of Zelda. Look closely and you'll see that Link moves on a grid, regardless of what direction the player inputs on the directional pad.

As you recreate this grid-based movement, ask yourself why such a feature was included, and what affect (however small) it has on both exploration and combat. How is this grid different from that of Pokemon, and how come they feel so different when it comes to movement.

2.2 Camera movement and control

The Legend of Zelda features a room-based camera that shifts as the player enters into different rooms. Ask yourself why a free-moving, continuously-following camera wasn't used, and its affects on exploration, combat, and general suspense.

2.3 Collectables

Items and collectables are one way that The Legend of Zelda rewards exploration. You will need to implement a small subset of them...

1. Hearts
2. Rupees
3. Bombs

The drop-rates of these items are interesting to study, but won't be heavily graded.

2.4 Weapons

The Legend of Zelda has a number of weapons ripe for recreation. We require you to recreate five of them:.

1. The Wooden Sword
2. The Boomerang
3. The Bow
4. Bombs
5. The Shield

The items must be authentically recreated. For example, the sword should produce a magical shot when used at full health. The boomerang should stun particular types of enemies, kill others, and retrieve items, etc.

2.5 Enemies

The Legend of Zelda features a number of enemies in its dungeons. You are expected to implement every enemy that appears in the first dungeon...

1. Blade Trap
2. Gel
3. Goriya
4. Keese
5. Stalfos
6. Wallmaster
7. Aquamentus

Quality authenticity in the movement and attack patterns of enemies is expected here as well. Keese should eventually stop to rest, Wallmasters should warp you to the dungeon entrance, etc.

2.6 Environment

Exploration is made interesting in part due to obstacles like locked doors and gimmicks such as pushable blocks. The mentioned two are particularly important, but there are others in your burndown chart.

3 General Tips

- The Zeldapedia, located at zelda.wikia.com, is a great resource for details on the first Legend of Zelda game.
- Rooms in The Legend of Zelda are comprised of 16x11 cells. For many purposes, such as routing enemies away from walls, it will be useful to determine what ”cell” a particular object is in. You'll then want to be able to reason about neighboring cells.
- Work in proximity with your teammates whenever possible. Ask each other questions. Project 1 is primarily one large Unity investigation, so parallelize your research / planning.
- Use office hours for a massive efficiency boost.
- Attend Wolverine Soft to ease your transition into Unity.

4 Unity Tips

- Examine the open-source 494_quest project available on Piazza. It includes state-machine technology that will aid in the construction of enemy behavior.
- Global data, such as manager objects, sound files, player objects and pretty much anything that will be a singleton, can be made static and globally accessible from anywhere in your codebase. ”public static (type) (member_name)”.
- GameObjects and components are destroyed when the game switches scenes. Static data survives, however. If you need to persist data across scenes (such as a character selected, file selected, num lives, high score, etc) then static variables, player preferences, or the DontDestroyOnLoad function may be used.
- Literally anything that provides behavior in Unity is a COMPONENT. Renderers, box colliders, rigidbodies– all are components. Every script you write is just a new component (that you wrote). Need player control? Write a component. Need the HUD to update? Write a component.
- Components that work ”globally”, such as HUD-controllers and other managers may often be placed on the Main Camera. It's a good place for them, as one is always guaranteed to exist.
- Does a component you're writing need a reference to another GameObject? An AudioClip? create a public member in your component's script, and drag-and-drop the reference into the component via the inspector. Do not use GameObject.Find. If you rename anything, the drag-and-drop solution will still work, while the GameObject.Find solution will not.