Technical Design Document

THE LABYRINTH OF HAWARA

Iryna Volodymyrova

19/02/2023

Introduction

In "The Labyrinth of Hawara," the player takes on the role of Dr. Arina Bogati, an archaeologist searching for a rare artifact in the mysterious Egyptian labyrinth of Hawara. The game features top-down maze exploration and puzzle solving, as well as enemies and obstacles to overcome.

Rationale

The goal of the game is to provide a fun and engaging experience for players who enjoy maze exploration and puzzle solving. By incorporating tools, pickups, and interactables, the game offers a variety of challenges and opportunities for strategic thinking.

Background

"The Labyrinth of Hawara" is set in the ancient Egyptian labyrinth of Hawara, which is said to be one of the largest and most complex labyrinths ever constructed. The labyrinth was built by the Pharaoh Amenemhet III in the 19th century BCE and was used for religious and ritual purposes.

Terminology

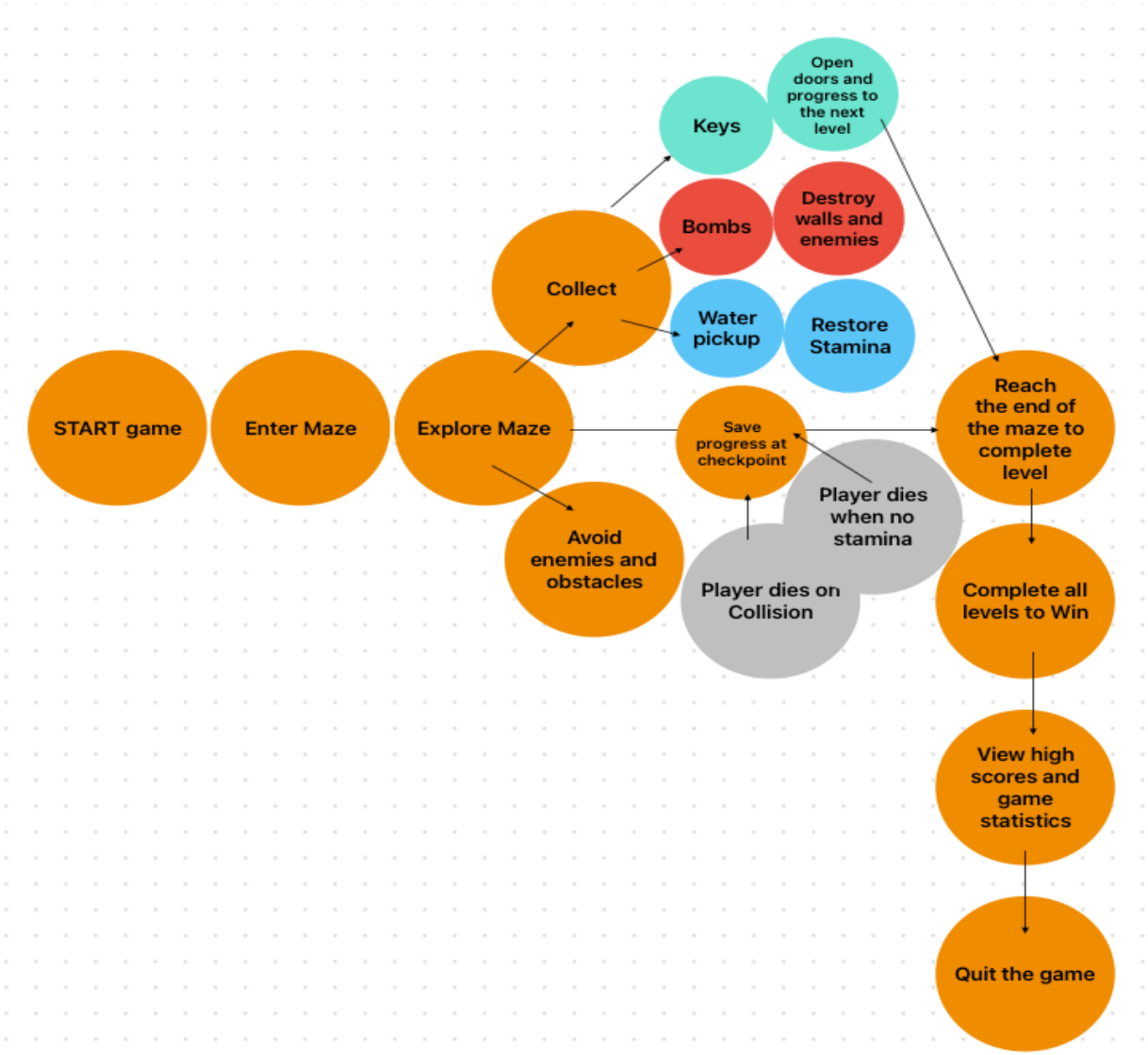
- Dr. Arina Bogati: The protagonist of the game, an archaeologist searching for a rare artifact in the labyrinth of Hawara.
- Keys: Tools that the player can use to open doors and progress to the next level.
- Bombs: Tools that the player can use to destroy walls and enemies.
- Water: Pickups that the player can collect to restore stamina.
- Stamina: A timer that represents the player's health and time remaining in the game.

- Enemies: Obstacles that the player must overcome to progress through the maze.
- Interactables: Objects in the maze that the player can interact with using their tools.

Non-Goals

- Multiplayer functionality
- Integration with external systems or services

Game Play Diagram



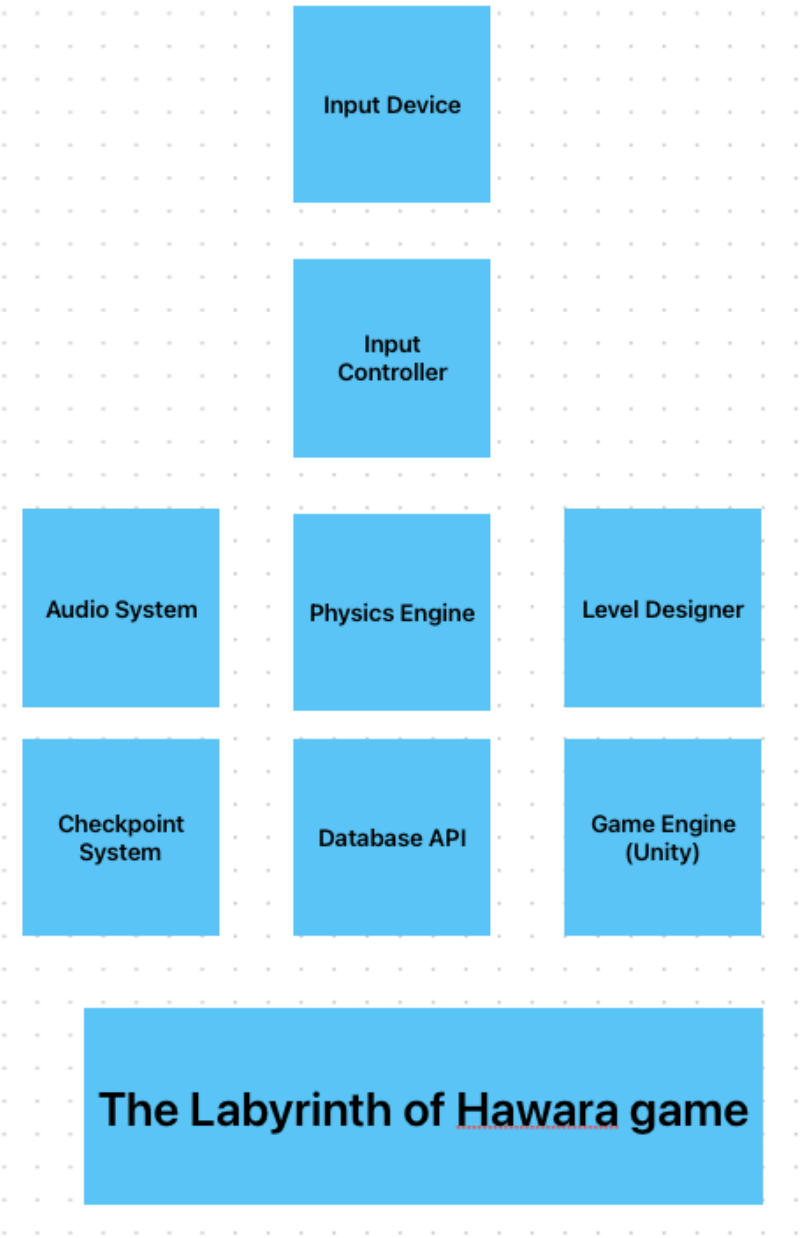
Proposed Design

1. Start Screen: Displays the game title and allows the player to start a new game.
2. Input Controller: Receives input from the player and updates the player's position and movement based on the input.
3. Audio System: Plays music and sound effects during gameplay.
4. Physics Engine: Simulates the movement and interactions of objects in the game environment.
5. Level Designer: Designs the layout of the mazes for each level and specifies the location of walls, pickups, interactables, enemies, and checkpoints.
6. Checkpoint System: Allows the player to save their progress and restart from a certain point in the maze.
7. Database: Stores data on the player's progress, including their current level, score, and the number of times they have attempted the maze.
8. End Screen: Displays a summary of the player's performance at the end of each level and allows the player to restart the game or quit.

System Architecture

- Game Engine: Coordinates the functionality of the subsystems listed above and ensures smooth gameplay.
- Player Controller: Receives input from the player and updates the player's position and movement based on the input.
- Maze Generator: Creates the layout of the mazes and places objects such as walls, pickups, interactables, enemies, and checkpoints.
- Combat System: Determines how the player's tools and pickups affect enemies and how enemies affect the player's stamina.
- User Interface: Displays the game screen and allows the player to interact with the game.

High Level Diagram



Data Model

"The Labyrinth of Hawara" is designed to be lightweight and should have minimal impact on system performance. However, the game may require a certain level of hardware specifications to run smoothly.

- Player Data: Stores information on the player's progress, including their current level, collectables, and the number of times they have attempted the maze.
- Maze Data: Stores information on the layout of each maze, including the location of walls, pickups, interactables, enemies, and checkpoints.

System Design

The game will be built using a modular approach, with small, reusable components. This will allow for easier experimentation and modification of individual game systems.

- Monolithic classes will be avoided where possible, to reduce the complexity of the code and make it easier to maintain.
- However, certain classes such as the player may need to be implemented as monolithic classes due to their unique properties.
- The game will use a "Toolbox" approach to game managers, where a single singleton object will hold all of the global managers needed for the game, such as audio, input, and graphics managers.
- The Toolbox approach will help reduce code complexity by providing a centralized location for all game managers, making it easier to manage and update them.
- The game will also use a state machine design pattern to manage different game states, such as the main menu, gameplay, and game over screens.
- The state machine will allow for easier management and transition between different game states, and will help to reduce code complexity.
- The game will use the Unity game engine as the underlying framework, providing a powerful and flexible platform for game development.
- The game will be built using C# programming language, allowing for fast and efficient game development.
- The game will utilize Git or a similar version control system to manage code changes and collaboration between team members.
- Testing and debugging will be done throughout the development process to ensure the game is stable and free from bugs before release.

Interface/API Definitions

- Input Controller Interface: Receives input from the player, such as movement and tool usage, and updates the player's position and movement based on the input.
- Audio System Interface: Plays music and sound effects during gameplay.
- Physics Engine Interface: Simulates the movement and interactions of objects in the game environment.
- Level Designer Interface: Allows the level designer to specify the layout of the maze and the location of objects such as walls, pickups, interactables, enemies, and checkpoints.
- Checkpoint System Interface: Allows the player to save their progress and restart from a certain point in the maze.
- Database API: Stores and retrieves data on the player's progress.

Migration Strategy

"The Labyrinth of Hawara" is a standalone game and does not require any migration from an existing system.

Delivery Platform

The game will be available as a downloadable game for PC and Mac platforms, with the option for players to purchase and download the game from a digital storefront such as Steam or the Epic Games Store.

Hardware Requirements

- Intel or AMD dual-core processor, 2.0 GHz or higher
- 4 GB RAM
- 2 GB free hard drive space
- DirectX 10 or higher compatible graphics card with 1 GB VRAM
- Windows 7, 8, or 10 (64-bit) or MacOS X 10.12 or higher
- Internet connection for activation and updates

Software Requirements

- Unity game engine (version 2019.4.16f1 or later)
- Visual Studio or similar integrated development environment (IDE)
- Git or similar version control system
- Text editing software (e.g. Visual Studio Code, Sublime Text)
- Audio editing software (e.g. Audacity)

These requirements should ensure that the game runs smoothly and without major issues on most modern computers. However, we recommend that players check their system specifications against the requirements listed above before purchasing and downloading the game.