August - 2025

UNITY PROGRAMMER

INTERVIEW DOCUMENT

Pedro Vilas Bôas

Thought Process

The first thing I did was to lay out a small **Game Design Document**. I thought about the kind of game I wanted to make and the core mechanics I wanted to build. Alongside this, I created and maintained each task, feature, chore, bug, and improvement throughout the entire development process using Notion.

I wrote down the **key requirements** for the project: it had to have an inventory, world interaction with items and/or NPCs, and a save/load system. Even though world-building was optional, I made it a personal requisite because I believe it's what gives a project its soul.

Thinking about those foundations led me to the core mechanic of the game: **an Item-Based Skill System**.

From there, I started crafting the specific mechanics and gathering the assets I'd need. I decided on three core item-based skills to build the experience around: a **double jump**, a **flying dash**, and an **instanced skill** that would damage NPCs.

This central idea shaped the entire architecture of the project.

GitHub Project

Game Design Document

Notion Project

System Overview

The project's architecture is built around a central **Item-Based Skill System**, where player progression is driven by the items they discover and equip.

This core loop is powered by a **Inventory and Equipment System**, which communicates with a decoupled **Stat System** to **dynamically modify character attributes**. It also interfaces with the **Player Action System** to unlock new abilities, such as the double jump or powerful special attacks.

Player interaction with the world, including **looting chests** and **speaking with characters**, is managed by a flexible Interaction System that triggers a **data-driven Dialogue System**. All of these gameplay features are supported by foundational, persistent managers for Audio, Saving and Loading, and overall Game State.

These systems are designed to be highly modular, communicating through a centralized C# event bus to ensure a clean and scalable codebase that is easy for designers and other developers to work with.

Personal Performance Assessment

Throughout this project, I think showed significant growth as a developer and game developer. My debugging skills were consistently tested, allowing me to identify and resolve subtle but critical bugs, from state management flaws in the Inventory and Dialogue UI to race conditions in the event-driven Audio and State Managers.

More importantly, I believe I had strong architectural intuition. I made a decision to challenge initial thoughts that could have led to "*God Objects*", consistently pushing for more **modular solutions**.

On decoupling the Inventory and Equipment managers from their UI and building the core Item-Based Skill System on a data-driven **ScriptableObject** foundation shows my vision for the codebase's integrity and longevity.
I'm proud of myself for integrating these complex systems in a tight timeframe. As I said before: If I don't know something, I'll research, I'll study, and I'll keep trying until I get it right. This persistence is what allows me to grow as a game developer.