Casters & Coders

Advisor/Client: Mat Wymore
By: sddec23-13

Branden Butler
Wenqin Wu
Edward Dao

Max Bromet
Theng Wei Lwe
Brennan Seymour

Contact Team Lead at: bseymour@iastate.edu

Problem Statement

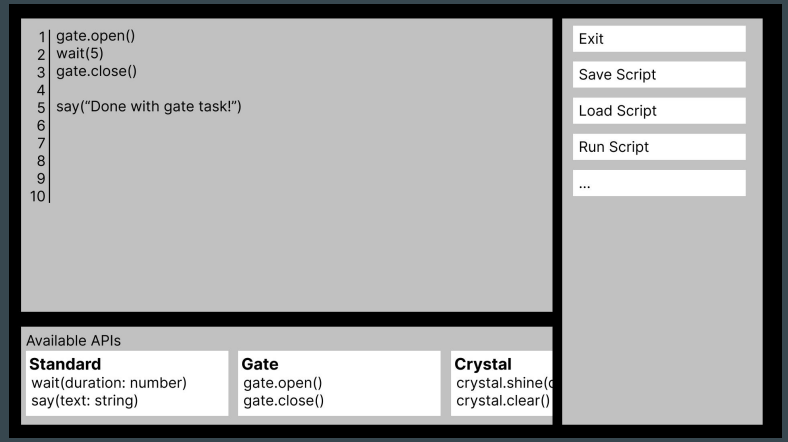
Some people might want to learn basic programming with little to no background knowledge, so we set out to create a high-fantasy puzzle game which:

- Has coding as a main gameplay mechanic
- Teaches basic programming concepts
- Requires little to no background knowledge
- Is fun to play

Overview

- 2D, Top-Down, sprite-based video game
- User control a sprite character by using their keyboards
- There will be rooms with puzzles that the player can solve by writing python scripts
- An editor screen to write scripts
- Scripts will control elements of the environment

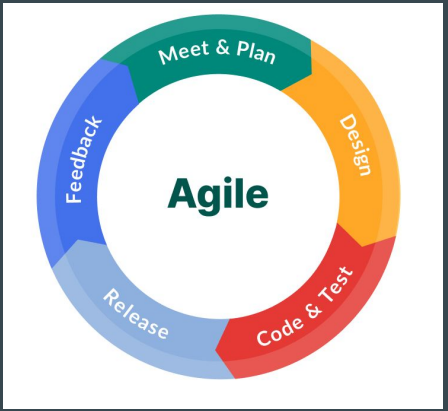
Game Environment Mockup



Text Editor Mockup

Project Management

- We will adhere to the agile project management style
  - 2 week sprints
  - Combined Retro & Planning meeting each sprint
  - Two standups a week
  - Three working teams of 2



Goals for the semester

- **Main Goal** : Minimum working game product by the end of the semester (at least 3 puzzles)
  - Arithmetic puzzles
  - Conditionals
  - Loops
- Learn more about agile team development and collaboration
  - Standup meetings
  - Sprint Planning and Retrospective
  - Stories/Tasks sizing and management

Tools and Technologies

EMBEDDED LANGUAGE

Python

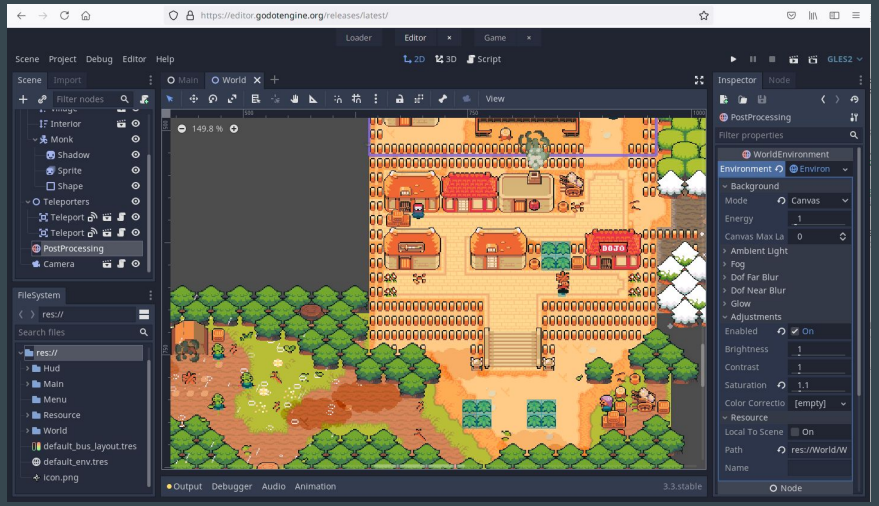
- Easy to use and embed
- Very common language in real world
- Common beginner language
- Dynamic / Duck typing
- Not prototype-based

ENGINE

Godot

- Supports many game logic languages through bindings
- Result - easy to embed scripting languages
- Node system lends itself well to our needs

Task Decomposition



- Engine and Scripting System
- Story, Environment, Design and Asset Creation
- Puzzle Design
- Mechanics of the Game and Level Implementation
- User Interfaces

Project Milestones

Engine and Scripting System

- Script API implemented

Story, Environment Design, and Asset Creation

- Completed Simple Storyline
- Completed map design and designs of characters

Puzzle Design

- Create puzzles that teach arithmetic, conditionals, and loops

Mechanics of the Game and Level Implementation

- Fully functional player character
- Interactable objects

User Interfaces

- Fully functional HUD for player
- In-game IDE for user input with syntax highlighting
- Dialogue boxes

Gameplay

PUZZLES

Environment-Based Puzzles

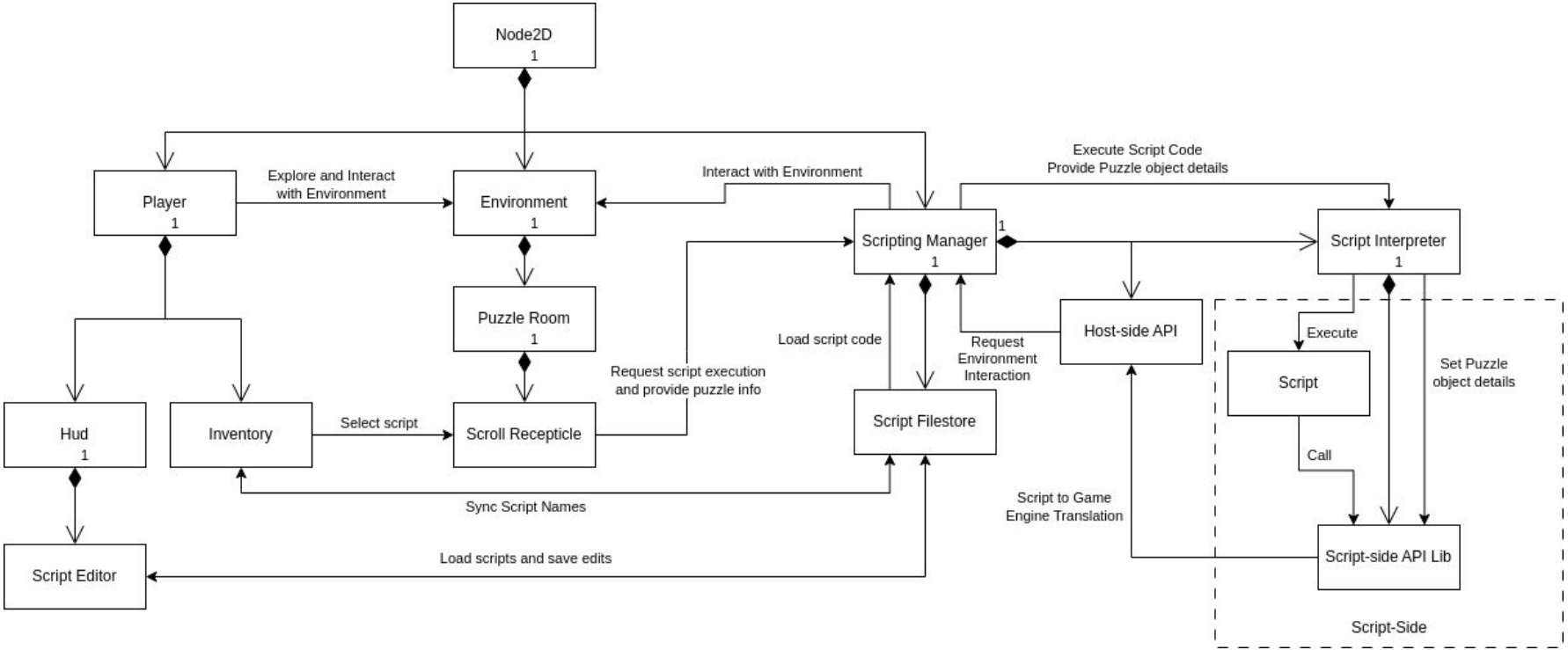
- Scripts directly control objects in the environment
- Each script is provided a simplified API to control its attached game object
- Success is measured in area traversal - can you get to the end of the puzzle chamber

Technical Challenges - Scripting System

- Integration between the user script and the world objects
  - API is easy to use and understand from the user POV
    - Allows a smooth introduction to Python
  - Ensure game object state is synchronized with what the script sees
- Script “bookkeeping”
  - Ensure script states don’t leak out
    - A script should not be able to see objects or variables from previously-ran scripts
  - Track script files and map them to in-game “scrolls”
  - Ensure buggy scripts don’t crash the game
    - Return useful error messages
  - Sync editor with backend script files

Scripting System Design

Godot Node Graph



Technical Challenges - UI

- IDE
  - Offer information about available functions from the scripting API for each puzzle
  - Provide rudimentary syntax highlighting
  - Time permitting, present basic completions
  - Allow saving + loading scripts
- Dialogue Boxes
  - Show dialogue from NPCs
  - Allow basic response selection
  - Define dialogue paths with YAML files

Technical Challenges - Game Logic

- Provide game progress saving & loading
- Add some basic settings (volume, etc.)
- Manage scene loading / unloading as players walk between rooms
- Consume interfaces from Scripting Engine to allow player-made scripts to control game objects
- Create developer tools as necessary

THANK YOU

...