Realtime Unity Voice Development Workflow Manual

This manual describes the full workflow, infrastructure, and objectives of the Realtime Unity Voice Development Pipeline. It serves as a handover package for assistants across multiple chats, ensuring continuity and consistency even when sessions reset or become too long.

# 1. Project Goal

The Unity project is designed to enable a hands-free, real-time development workflow where the developer issues voice commands that are processed by Whisper for speech-to-text and bridged into Unity. Unity ingests the commands, generates scripts, and applies changes instantly in the Editor. This enables rapid prototyping, testing, and iteration without breaking immersion or requiring manual edits.

# 2. Workflow Loop

The intended workflow loop operates as follows:

1. 1. Developer starts Play Mode in Unity to test the current build.
2. 2. Upon exiting Play Mode, Unity automatically pushes changes and logs to Git.
3. 3. Developer notifies ChatGPT that Play Mode has ended.
4. 4. ChatGPT fetches Git logs and project changes, then prepares a patch based on instructions.
5. 5. ChatGPT provides a downloadable `.patch` file.
6. 6. The router system detects the downloaded patch and auto-applies it to the Unity project.
7. 7. Developer re-enters Play Mode to test the new changes.
8. 8. The loop repeats.

# 3. Components

The workflow consists of the following main components:

- Whisper STT (Python): Handles real-time voice transcription with silence gating and passes raw C# text to Unity.

- Unity VoiceCommandReceiver.cs: Listens for incoming commands, writes/updates scripts in Assets/GeneratedScripts, and applies changes.

- Patch Router (PowerShell): Watches the Downloads folder for `.patch` files and applies them automatically.

- Git Integration: Unity auto-commits changes after Play Mode and pushes them to the repository for assistant access.

# 4. Workflow Visualization

Below is a high-level diagram (ASCII representation for portability):

[ Voice Command ]   
 ↓  
[ Whisper STT → Python Bridge ]   
 ↓  
[ Unity Receiver.cs → Auto Scripts ]  
 ↓  
[ Play Mode Test ]  
 ↓ Exit  
[ Git Push (logs + code) ]  
 ↓  
[ ChatGPT: Analyze + Generate Patch ]  
 ↓  
[ Download Patch ]  
 ↓  
[ Router Auto-Applies Patch ]  
 ↓  
[ Back to Play Mode ]

# 5. Usage Notes

- Always inform ChatGPT after exiting Play Mode so it can analyze logs and prepare patches.  
- The `.patch` router ensures minimal manual effort—just download the patch, and it applies.  
- Keep Git repository synced frequently to avoid drift.  
- The manual itself should be iteratively updated with project evolution.