Inworld AI <-> Roblox Integration

Context

Inworld AI

Inworld AI provides a creative suite that offers a powerful and intuitive way to build interactive AI characters. Create engaging characters whose lifelike, engaging, and expressive personalities are designed to mimic the deeply social nature of human interaction.

Roblox

Roblox is an online platform and storefront where users go to play games. Roblox is not a game, it is a place where people can jump into worlds and experiences created by other developers.

Objective

We are looking for developers to implement a solution that would allow Inworld customers to easily integrate and deploy character brains created from the Inworld Studio into the Roblox platform.

Task Definition

Inworld is a SaaS platform that provides a complicated (streaming) API. Roblox is a well-known game development platform with lots of different restrictions, including one to their http stack for making external requests.

Specifically, we need to develop a way for Roblox game servers to communicate with Inworld platform's capabilities including Dialogue, Emotions, Scenes, Goals and Actions, etc.

This will consist of two subprojects: Developing a **HTTP Proxy** and a **Roblox Connector** (or just distributable Lua scripts to start). We are looking for engineers capable of developing either or both components. **In your application, please indicate which you would like to apply for.**

Inworld HTTP Proxy

Implement a simple and generic proxy-server built on top of our existing Node.js SDK that will allow regular HTTP GET/POST communication between a service and the Inworld.AI platform. This will consist of unwrapping the existing Node.js stream-based sdk into some sort of long-polling get/post solution, similar to https://www.npmjs.com/package/roblox-long-polling. There are lots of Roblox chat implementations based on this approach that can be used as reference (note that more modern approaches would not work here).

Specific Requirements:

1. Service should be implemented in the form of a private Node.js package/server
2. Service should consume Inworld API and SECRET keys as environment variables and pass them to the Inworld Node.js SDK
3. Service should expose a simple HTTP API. As an example for reference (final API can be changed):
    a. **POST /loadScene** (with either a scene name or character name as the payload)
        i. Used to call client.setScene in the Node.js SDK and should return the generated session id (token.getSessionId()) and a list of characters in the loaded scene (client.getCharacters())
    b. **POST /sessions/{SESSION_ID}/characters/{CHARACTER_ID}/text**
        i. Used to internally call connection.sendText(message);
    c. **POST /sessions/{SESSION_ID}/characters/{CHARACTER_ID}/custom**
        i. Used to internally call connection.sendCustom(message);
    d. **GET /sessions/{SESSION_ID}/characters/{CHARACTER_ID}/events**
        i. Used to return a long-polling connection that returns the list of events
4. Authentication and scalability factors are TBD. In the long polling scenario, it should be easy to scale the service as needed and rely on the session state on the Inworld server side.

Roblox Connector

Implement a set of Lua scripts that would allow attaching any Roblox NPC to the Inworld platform, with custom controls for both input and output.

Specific Requirements:

1. Use https://create.roblox.com/docs/tutorials/building/ui/proximity-prompts to start interaction
2. Use https://create.roblox.com/docs/reference/engine/classes/TextBox for text input
3. Use https://create.roblox.com/docs/reference/engine/classes/Dialog for text output
4. Develop internal triggers in Lua to hook up triggers that come from the Inworld platform (onEvent processing)
5. Develop internal API to send custom events towards inworld platform