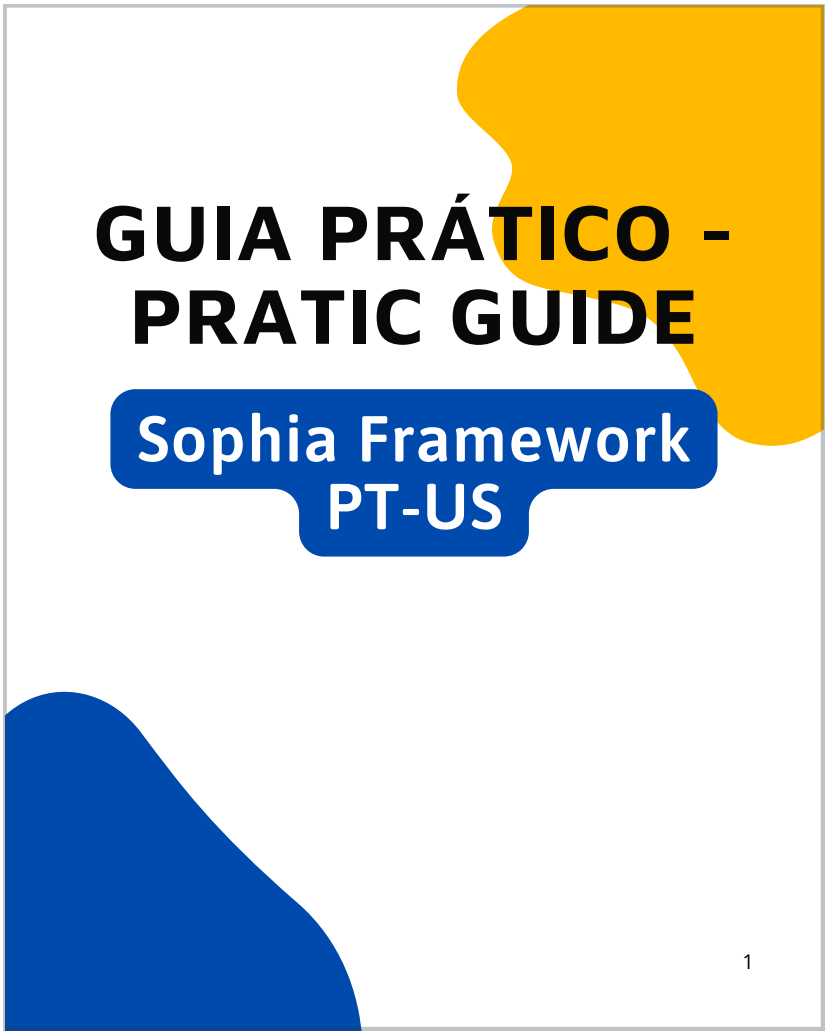

GUIA PRÁTICO -
PRATIC GUIDE
Sophia Framework
PT-US
1

O QUE É O FRAMEWORK SOPHIA?

O FRAMEWORK SOPHIA É UM SISTEMA INTELIGENTE PARA CRIAÇÃO DE NPCS INTERATIVOS EM JOGOS NARRATIVOS DESENVOLVIDOS COM UNITY 2D (TOP DOWN).

ELE PERMITE QUE PERSONAGENS NÃO JOGÁVEIS TENHAM:

- RESPOSTAS EMOCIONAIS REALISTAS
- MEMÓRIA ATIVA DE INTERAÇÕES
- APRENDIZADO POR REFORÇO COM REDE NEURAL
- VOZ SINTETIZADA EM TEMPO REAL

TUDO ISSO CRIA EXPERIÊNCIAS MAIS IMERSIVAS, SENSÍVEIS E DINÂMICAS DENTRO DO JOGO. SOPHIA PODE SER USADA EM PROJETOS EDUCACIONAIS, RPGS, AVENTURAS E JOGOS SOCIAIS.

🇺🇸 ENGLISH:

WHAT IS THE SOPHIA FRAMEWORK?

THE SOPHIA FRAMEWORK IS AN INTELLIGENT SYSTEM FOR CREATING INTERACTIVE NPCS IN NARRATIVE-DRIVEN GAMES BUILT WITH UNITY 2D (TOP DOWN).

IT ENABLES NON-PLAYABLE CHARACTERS TO EXHIBIT:

- REALISTIC EMOTIONAL RESPONSES
- ACTIVE MEMORY OF PAST INTERACTIONS
- REINFORCEMENT LEARNING VIA NEURAL NETWORKS
- REAL-TIME SYNTHESIZED VOICE

THIS RESULTS IN MORE IMMERSIVE, EMOTIONAL, AND DYNAMIC GAMEPLAY EXPERIENCES. SOPHIA IS IDEAL FOR EDUCATIONAL PROJECTS, RPGS, ADVENTURE GAMES, AND SOCIAL SIMULATIONS.

2

Funções da Sophia - Sophia Functions

O Framework Sophia é uma ferramenta poderosa e modular projetada para criar interações mais realistas e emocionantes entre jogadores e NPCs em jogos. Cada componente desempenha um papel crucial na experiência geral. Por exemplo, o SophiaEmotionResponse analisa a fala do jogador para identificar emoções predominantes como raiva ou alegria, o que alimenta o RelationshipSystem, que ajusta os níveis de afeto com base nessas emoções identificadas. A SophiaRNN utiliza aprendizado por reforço para melhorar continuamente as interações, enquanto o SophiaMemory guarda memórias das falas e emoções para enriquecer o histórico do jogo. O NavMeshModule garante que os NPCs se movam de forma inteligente pelo ambiente, e o SophiaTTS transforma texto em fala usando a tecnologia do Google. O VoiceRecognition captura a voz do jogador para análise emocional, enquanto o WavUtility converte arquivos de áudio para serem usados no Unity. Por fim, o SaveData mantém um registro do estado emocional e do progresso do NPC, garantindo uma experiência de jogo coesa e envolvente.

3

Componentes - Components

Os módulos do Framework Sophia integram-se de maneira coesa para oferecer uma interação rica e emocional entre o jogador e os NPCs (Personagens Não Jogáveis). Quando o jogador fala com o NPC, o módulo VoiceRecognition captura e transcreve a fala. Em seguida, o SophiaEmotionResponse analisa essa fala para identificar a emoção predominante. Essa emoção detectada é fundamental, pois influencia diretamente a memória e o relacionamento entre o jogador e o NPC. O módulo SophiaMemory é responsável por armazenar tanto a fala quanto o valor emocional associado. Paralelamente, o RelationshipSystem ajusta o nível de afeto, modificando a relação com base nessa interação emocional. O aprendizado contínuo do NPC é gerido pelo SophiaRNN, que adapta o comportamento dos personagens através de técnicas de aprendizado por reforço. Em resposta à fala do jogador, o SophiaTTS gera um retorno verbal, que é convertido em áudio pelo WavUtility. Por fim, todo o histórico da interação, incluindo emoções, memórias e o estado do NPC, é preservado pelo módulo SaveData, garantindo que cada interação futura seja mais rica e personalizada.

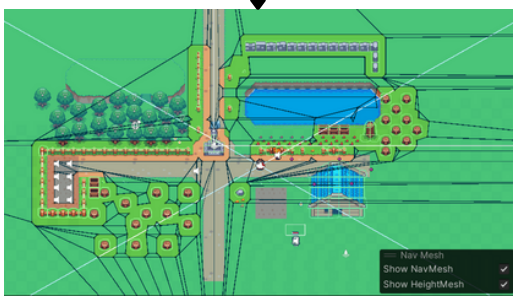
The modules of the Sophia Framework integrate cohesively to provide a rich and emotional interaction between the player and the NPCs (Non-Playable Characters). When the player speaks to the NPC, the VoiceRecognition module captures and transcribes the speech. Then, the SophiaEmotionResponse analyzes this speech to identify the dominant emotion. This detected emotion is crucial as it directly influences the memory and relationship between the player and the NPC. The SophiaMemory module is responsible for storing both the speech and the associated emotional value. Concurrently, the RelationshipSystem adjusts the affection level, modifying the relationship based on this emotional interaction. The NPC's continuous learning is managed by the SophiaRNN, which adapts character behavior through reinforcement learning techniques. In response to the player's speech, SophiaTTS generates a verbal reply, which is converted into audio by WavUtility. Finally, the entire interaction history, including emotions, memories, and the NPC's state, is preserved by the SaveData module, ensuring each future interaction is richer and more personalized.

4

Navmesh

A Navmesh, or Navigation Mesh, is a data structure used in artificial intelligence for pathfinding and spatial reasoning, particularly in video games and simulations. It represents a walkable area within a virtual environment, allowing AI-controlled characters to navigate around obstacles seamlessly. The Navmesh is typically made up of interconnected polygons that outline the traversable spaces, which the AI uses to determine the most efficient path from one point to another. During a Navmesh demonstration, developers can showcase how AI characters dynamically adjust their paths in real-time, reacting to changes in the environment such as moving objects or blocked passages. This demonstration highlights the robustness and flexibility of Navmesh in creating realistic and intelligent behaviors in virtual worlds.

Um Navmesh, ou Malha de Navegação, é uma estrutura de dados usada em inteligência artificial para busca de caminhos e raciocínio espacial, especialmente em videogames e simulações. Ele representa uma área caminhável dentro de um ambiente virtual, permitindo que personagens controlados por IA naveguem ao redor de obstáculos de forma fluida. O Navmesh é tipicamente composto por polígonos interconectados que delineiam os espaços transitáveis, que a IA utiliza para determinar o caminho mais eficiente de um ponto a outro. Durante uma demonstração de Navmesh, os desenvolvedores podem mostrar como os personagens de IA ajustam dinamicamente seus caminhos em tempo real, reagindo a mudanças no ambiente, como objetos em movimento ou passagens bloqueadas. Esta demonstração destaca a robustez e flexibilidade do Navmesh na criação de comportamentos realistas e inteligentes em mundos virtuais.



5

DRL e RNN - RNN and DRL

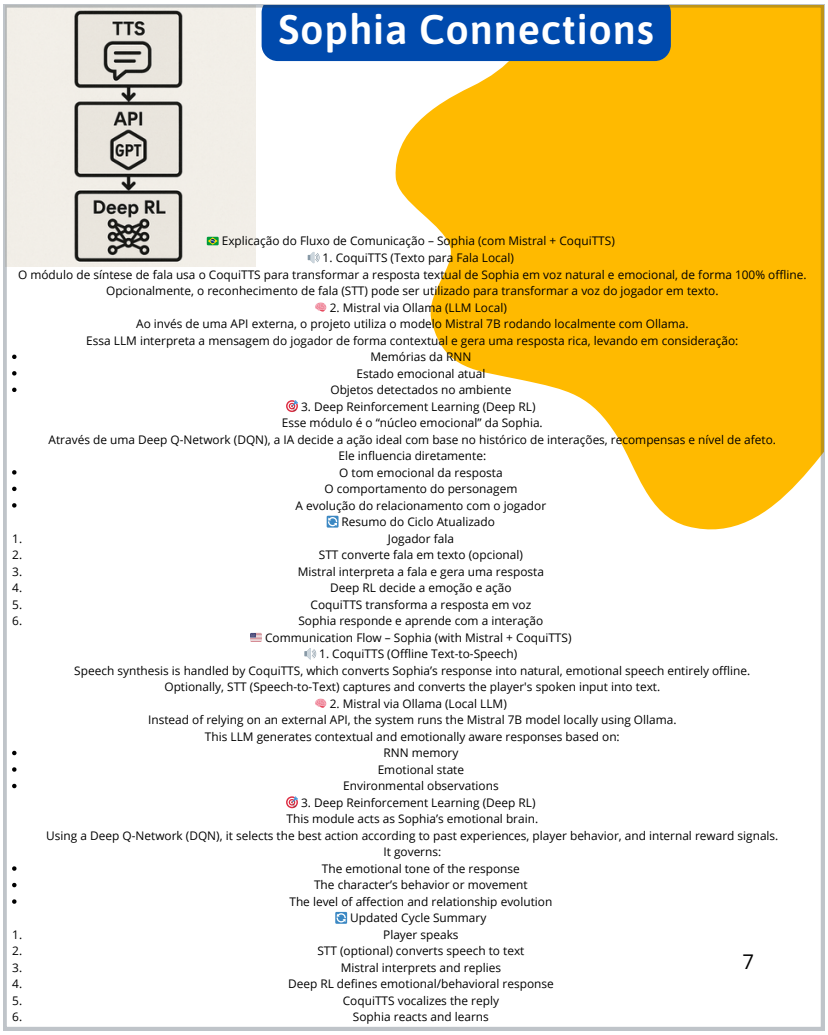
🇧🇷 Projeto Sophia – Inteligência Autônoma com DRL, RNN, Mistral e CoquiTTS

O Projeto Sophia integra múltiplas tecnologias de IA para criar uma NPC emocional e responsiva. Utilizando Deep Reinforcement Learning (DRL), a Sophia aprende com interações passadas, recebendo recompensas por respostas positivas. Sua Rede Neural Recorrente (RNN) fornece memória contextual de curto e longo prazo, permitindo diálogos coerentes e emocionalmente ajustados. As respostas textuais são geradas localmente usando o modelo Mistral 7B via Ollama, que interpreta o estado do jogador, memórias anteriores e objetos no ambiente. Por fim, a voz da Sophia é sintetizada com o CoquiTTS, transformando as respostas em fala natural dentro da Unity, de forma completamente offline.

🇺🇸 Sophia Project – Autonomous Intelligence with DRL, RNN, Mistral and CoquiTTS

The Sophia Project combines cutting-edge AI to create an emotional and responsive NPC. Through Deep Reinforcement Learning (DRL), Sophia learns from past interactions by receiving rewards for positive outcomes. Her Recurrent Neural Network (RNN) simulates both short- and long-term memory, enabling coherent, emotionally-aware dialogue. Responses are generated locally using the Mistral 7B model via Ollama, which interprets player state, previous memories, and environment objects. Finally, Sophia's voice is synthesized using CoquiTTS, converting the output into natural speech in Unity — all running fully offline.

6


TTS
API
GPT
Deep RL
Sophia Connections
🇧🇷 Explicação do Fluxo de Comunicação – Sophia (com Mistral + CoquiTTS)
🔊 1. CoquiTTS (Texto para Fala Local)
O módulo de síntese de fala usa o CoquiTTS para transformar a resposta textual de Sophia em voz natural e emocional, de forma 100% offline.
Opcionalmente, o reconhecimento de fala (STT) pode ser utilizado para transformar a voz do jogador em texto.
🧠 2. Mistral via Ollama (LLM Local)
Ao invés de uma API externa, o projeto utiliza o modelo Mistral 7B rodando localmente com Ollama.
Essa LLM interpreta a mensagem do jogador de forma contextual e gera uma resposta rica, levando em consideração:
Memórias da RNN
Estado emocional atual
Objetos detectados no ambiente
🎯 3. Deep Reinforcement Learning (Deep RL)
Esse módulo é o "núcleo emocional" da Sophia.
Através de uma Deep Q-Network (DQN), a IA decide a ação ideal com base no histórico de interações, recompensas e nível de afeto.
Ele influencia diretamente:
O tom emocional da resposta
O comportamento do personagem
A evolução do relacionamento com o jogador
🔁 Resumo do Ciclo Atualizado
Jogador fala
STT converte fala em texto (opcional)
Mistral interpreta a fala e gera uma resposta
Deep RL decide a emoção e ação
CoquiTTS transforma a resposta em voz
Sophia responde e aprende com a interação
🇺🇸 Communication Flow – Sophia (with Mistral + CoquiTTS)
🔊 1. CoquiTTS (Offline Text-to-Speech)
Speech synthesis is handled by CoquiTTS, which converts Sophia's response into natural, emotional speech entirely offline.
Optionally, STT (Speech-to-Text) captures and converts the player's spoken input into text.
🧠 2. Mistral via Ollama (Local LLM)
Instead of relying on an external API, the system runs the Mistral 7B model locally using Ollama.
This LLM generates contextual and emotionally aware responses based on:
RNN memory
Emotional state
Environmental observations
🎯 3. Deep Reinforcement Learning (Deep RL)
This module acts as Sophia's emotional brain.
Using a Deep Q-Network (DQN), it selects the best action according to past experiences, player behavior, and internal reward signals.
It governs:
The emotional tone of the response
The character's behavior or movement
The level of affection and relationship evolution
🔁 Updated Cycle Summary
Player speaks
STT (optional) converts speech to text
Mistral interprets and replies
Deep RL defines emotional/behavioral response
CoquiTTS vocalizes the reply
Sophia reacts and learns
7

Reconhecimento de voz - Voice Recognition

🇧🇷 Reconhecimento de Voz (Speech-to-Text – STT) no Projeto Sophia

🪄 O que é?

O módulo de reconhecimento de voz permite que a Sophia compreenda comandos e frases ditas pelo jogador em tempo real. A fala captada é transcrita em texto e enviada para os módulos de decisão emocional (Deep RL) e geração de resposta com LLM local (Mistral).

🧩 Como funciona na prática?

1. O jogador fala com Sophia por meio do microfone (ex: "Oi Sophia, está tudo bem?")
2. O sistema utiliza uma biblioteca de STT local, como o DictationRecognizer do Unity ou o Whisper.cpp rodando offline.
3. A fala é transcrita para texto e enviada ao modelo Mistral (via Ollama).
4. A resposta gerada passa por análise do Deep RL, que decide a emoção e ação.
5. O texto final é convertido em fala usando o CoquiTTS e reproduzido como resposta sonora.

💡 Vantagens do STT na Sophia

- Imersão total: o jogador interage por voz com uma NPC que escuta e responde emocionalmente.
- 100% offline: todos os módulos (STT, LLM e TTS) podem funcionar localmente.
- Análise emocional futura: possível integrar prosódia para que Sophia detecte tristeza, empolgação ou irritação na voz do jogador.

🇺🇸 Voice Recognition (Speech-to-Text – STT) in the Sophia Project

🪄 What is it?

The voice recognition module lets Sophia understand spoken messages from the player. The captured audio is transcribed into text and processed by the local emotional AI (Deep RL) and the LLM (Mistral).

🧩 How it works in Sophia

1. The player speaks through a microphone (e.g., "Hi Sophia, are you okay?")
2. The system uses a local STT engine like Unity's DictationRecognizer or Whisper.cpp.
3. The transcribed text is sent to the Mistral model (via Ollama) for language processing.
4. The Deep RL module determines the emotional and behavioral response.
5. The final response is converted to voice using CoquiTTS and played back.

💡 STT Benefits in Sophia

- Fully immersive: talk to Sophia naturally, as if she were alive.
- Offline support: all modules (STT, LLM, and TTS) work without an internet connection.
- Future integration: voice tone can be analyzed to detect emotional states and adjust responses accordingly.

8

Como a memória e percepção funcionam - How memory and perception works

🇧🇷 Memória e Percepção na Sophia

🧠 Memória (RNN - Rede Neural Recorrente)

📌 Como funciona:

A memória da Sophia é baseada em uma Rede Neural Recorrente (RNN) que simula memória de curto e longo prazo. Isso permite à NPC manter o contexto entre interações, lembrando:

- Frases anteriores ditas pelo jogador
- Eventos importantes da narrativa
- Estados emocionais detectados

A RNN recebe como entrada o histórico de texto e decisões anteriores, atualizando o estado interno da personagem para manter coerência no diálogo e na emoção.

🧠 Exemplo:

Se o jogador disser "tô meio pra baixo hoje", e minutos depois perguntar "o que você acha?", a Sophia pode lembrar do estado emocional e responder com empatia.

👁️🗨️ Percepção Ambiental Virtual

📌 Como funciona:

A Sophia possui um módulo de percepção ambiental implementado na Unity, que simula sentidos virtuais usando:

- Raycasting para detectar obstáculos
- Colisores 2D e Triggers para eventos próximos
- NavMesh para navegação e mapeamento do ambiente
- Sensores de proximidade e entradas de visão computacional (futuramente com OpenCV)

As informações captadas são transformadas em estados que alimentam a Deep Q-Network (DQN), otimizando decisões emocionais e comportamentais.

👁 Exemplo:

Se a Sophia percebe um obstáculo no caminho, ela pode evitá-lo. Se o jogador estiver sorrindo (em versões com reconhecimento facial), ela pode responder com uma fala mais animada, gerada via CoquiTTS.

🇺🇸 Memory and Perception in Sophia

🧠 Memory (RNN - Recurrent Neural Network)

📌 How it works:

Sophia's memory is simulated by an RNN, enabling her to maintain both short- and long-term context across interactions. She remembers:

- Previous dialogue with the player
- Important story moments
- Detected emotional states

The RNN processes dialogue history and internal variables, updating Sophia's state so she can respond coherently and naturally.

🧠 Example:

If the player says "I'm feeling down today" and later asks "What do you think?", Sophia can remember that and answer with compassion.

👁️🗨️ Environmental Perception

📌 How it works:

Sophia uses a virtual perception system built in Unity, acting as artificial senses. It includes:

- Raycasting for obstacle detection
- 2D Colliders and Triggers for interactive objects
- NavMesh for navigation and mapping
- Proximity sensors and optionally OpenCV for vision-based input

These inputs are converted into states used by her Deep Reinforcement Learning (DQN) module to make context-aware decisions.

👁 Example:

If a dangerous object is detected nearby, Sophia might avoid it. If the player is close and smiling (in a future version with facial recognition), she may respond cheerfully using CoquiTTS.

9

How To Implement in Your Project - Como Implementar no seu projeto

mplementar a Sophia no seu jogo é um processo detalhado que envolve vários passos, desde a criação do NPC na Unity até a personalização da memória residual. Primeiramente, você precisará criar ou importar um modelo de personagem 2D, como a Sophia, e configurar um controlador de animações para gerenciar transições de estados como falar, andar e ficar parado. Adicionar um colisor permitirá a nteração com o jogador.

Após isso, importe os scripts do framework Sophia necessários para a implementação, como SophiaController.cs, VoiceRecognition.cs, e outros relacionados ao reconhecimento de voz e aprendizado de máquina. Crie um GameObject chamado "Sophia" e configure os componentes necessários, incluindo URLs para serviços de geração de texto e síntese de voz.

A memória residual adaptativa da Sophia pode ser ajustada para diferentes tipos de NPCs, como um dragão maligno ou um cavaleiro da morte, usando o script SophiaMemory.cs para definir o contexto inicial e os estados emocionais. O fluxo de comunicação é essencial: a fala do jogador é capturada, processada e interpretada, e a Sophia responde de maneira contextualizada, atualizando sua memória e percepção ambiental.

Por fim, personalize a Sophia para criar experiências únicas no jogo, ajustando seus níveis de afinidade e memórias iniciais para cada cenário específico.Certainly! Here's the guide in English:

Steps to Integrate Sophia in Your Game

1. Create the NPC in Unity

- **Model:** Design or import a 2D character (e.g., Sophia, Dragon, Knight).
- **Animator Controller:** Set animations and transitions (Idle, Speak, Walk).
- **Collider:** Add a `BoxCollider2D` or `CircleCollider2D` for interactions.
- **NavMesh Agent (optional):** Use for autonomous movement on the map if needed.

2. Import the Sophia Framework

- Required scripts:
  - `SophiaController.cs`
  - `SophiaMistral.cs`
  - `VoiceRecognition.cs`
  - `RelationshipSystem.cs`
  - `ReinforcementLearning.cs`
  - `SophiaMemory.cs`
  - `SophiaPerception.cs`
  - `WAV.cs`

3. Attach and Configure Components

- Create a GameObject named "Sophia" and add the scripts and `AudioSource`.
- Set the URLs:
  - `http://localhost:11434/api/generate` (Mistral)
  - `http://localhost:5002/api/tts` (CoquiTTS)

4. Residual Memory Adaptation

Sophia's RNN-based memory can be customized for any NPC type:

- **Evil Dragon:** Configure memories of destructive events, aggressive phrases, and relationships based on submission and challenge.
- **Death Knight:** Store moral judgments, dark responses, and interactions of honor.
- **Priest:** Record spiritual themes, requests for blessings, and pacifist or ethical decisions.

Use the script `SophiaMemory.cs` to adjust initial context, emotional states, and previous records.

5. Communication Flow

1. Player speaks into the microphone → `VoiceRecognition.cs` (STT)
2. Text is interpreted via `SophiaMistral.cs` using the Mistral LLM model
3. Response is emotionally analyzed via `ReinforcementLearning.cs`
4. Memory is updated → `SophiaMemory.cs`
5. Object and state perception → `SophiaPerception.cs`
6. Vocal response is generated with `CoquiTTS` → `AudioSource.Play()`

6. Example of Custom NPC Initialization

```csharp
SophiaMemory.SetInitialMemory("You are an ancient dragon guarding the Fire Temple.");
RelationshipSystem.SetAffinityLevel(-30); // Starts hostile
```

10

O que você vai precisar - What will you need?

🇧🇷 O que você vai precisar

🎮 Engine:

- Unity versão 2021.3.0f1 ou superior (recomendado LTS)
- Sistema 2D Top Down habilitado

🧠 Modelo de Linguagem:

- Mistral 7B QLoRA instalado e rodando via Ollama
- Acesso local via http://localhost:11434/api/generate

🔊 Síntese de Fala (TTS):

- Coqui TTS com modelo em português (ex: tts_models/pt/cv/vits)
- Endpoint local: http://localhost:5002/api/tts

🍀 Framework Sophia:

- Scripts principais:
  - SophiaController.cs ✅
  - SophiaMistral.cs
  - VoiceRecognition.cs
  - RelationshipSystem.cs
  - ReinforcementLearning.cs
  - SophiaMemory.cs
  - SophiaPerception.cs
  - WAV.cs (leitor de áudio .wav em runtime)

🎤 Pré-requisitos de Voz:

- Microfone configurado no PC
- API de reconhecimento de voz (Unity DictationRecognizer no Windows)

📁 Bibliotecas e Dependências:

- UnityEngine.Networking (integrada na Unity 2021+)
- System.Speech não é necessário (opcional para STT avançado)
- Python 3.10 para Coqui
- Virtualenv para isolar ambiente (python3 -m venv venv)

🧠 Hardware Recomendado:

- 16 GB RAM ou mais
- Placa de vídeo com CUDA para acelerar Coqui (opcional)
- SSD para leitura rápida dos modelos

🇺🇸 What you will need

🎮 Game Engine:

- Unity version 2021.3.0f1 or higher (LTS preferred)
- Top-down 2D project setup

🧠 Language Model:

- Mistral 7B QLoRA installed and served locally using Ollama
- Local endpoint: http://localhost:11434/api/generate

🔊 Text-to-Speech:

- Coqui TTS with Brazilian Portuguese model (e.g. tts_models/pt/cv/vits)
- Local endpoint: http://localhost:5002/api/tts

🍀 Sophia Framework Scripts:

- Core components:
  - SophiaController.cs ✅
  - SophiaMistral.cs
  - VoiceRecognition.cs
  - RelationshipSystem.cs
  - ReinforcementLearning.cs
  - SophiaMemory.cs
  - SophiaPerception.cs
  - WAV.cs (runtime audio buffer reader)

🎤 Voice Requirements:

- Microphone properly configured
- DictationRecognizer API (for STT on Windows)

📁 Dependencies:

- UnityEngine.Networking (comes with Unity)
- Python 3.10 for running Coqui TTS
- Virtual environment (python3 -m venv venv)

🧠 Recommended Hardware:

- 16 GB RAM or more
- GPU with CUDA support for faster TTS (optional)
- SSD for faster loading of models

11

Compartilhe com outros devs - Share with another devs

🇧🇷 Compartilhe com outros devs

Este framework é gratuito e open-source sob a licença MIT. Se ele te ajudou, compartilhe com a comunidade e ajude a criar NPCs mais vivos, emocionais e inteligentes em jogos 2D!

📦 Acesse no Unity Asset Store:

https://assetstore.unity.com/preview/323455/1075823

📌 Contribua, aprimore e transforme o Sophia Framework no seu projeto.

🇺🇸 Share with other devs

This framework is free and open-source under the MIT license. If it helped you, share it with the community and help bring more emotional and intelligent NPCs to 2D games!

📦 Available on the Unity Asset Store:

https://assetstore.unity.com/preview/323455/1075823

📌 Contribute, improve, and make the Sophia Framework your own.

12