CourseVR

0.1.0

Создано системой Doxygen 1.13.2

Глава 1

Алфавитный указатель пространств имен

1.1 Список пакетов

Полный список документированных пакетов.

Глава 2

Иерархический список классов

2.1 Иерархия классов

Иерархия классов.

Глава 3

Алфавитный указатель классов

3.1 Классы

Классы с их кратким описанием.

Глава 4

Список файлов

4.1 Файлы

Полный список файлов.

Глава 5

Пространства имен

5.1 Пространство имен CurseVR

Пространства имен

- namespace VoiceControl

5.2 Пространство имен CurseVR.VoiceControl

Пространства имен

- namespace Core
- namespace Editor
- namespace Input
- namespace Models
- namespace Services
- namespace Test
- namespace Utils
- namespace VAD

5.3 Пространство имен CurseVR.VoiceControl.Core

Классы

- interface IVoiceCommandService
  Interface for the voice command service that handles communication with the ASR API.

5.4 Пространство имен CurseVR.VoiceControl.Editor

Классы

- class VoiceInputManagerEditor

5.5 Пространство имен CurseVR.VoiceControl.Input

Классы

- class VoiceInputManager

  Manages voice input and integrates with Unity's Input System.

5.6 Пространство имен CurseVR.VoiceControl.Models

Классы

- class VADParameters

  Parameters for the Voice Activity Detection (VAD) system.
- class VoiceCommandData

  Data structure for voice commands received from the ASR service.
- class VoiceServiceConfig

  Configuration settings for the voice command service.

5.7 Пространство имен CurseVR.VoiceControl.Services

Классы

- class VoiceCommandService

  Implementation of the voice command service using WebSocket communication.

5.8 Пространство имен CurseVR.VoiceControl.Test

Классы

- class MockVoiceControlTest

5.9 Пространство имен CurseVR.VoiceControl.Utils

Классы

- class AudioUtils

  Utility class for audio processing operations.

5.10 Пространство имен CurseVR.VoiceControl.VAD

Классы

- class AudioClipBuffer

  Buffers audio samples and creates AudioClips when the buffer is filled or flushed.
- interface IVoiceActivityDetector

  Interface for voice activity detection functionality.
- class UnityMicrophoneProxy

  Provides an abstraction over Unity's Microphone API for simplified access to audio input.
- class VoiceActivityDetector

  Implements voice activity detection by analyzing microphone input volume.

Глава 6

Классы

6.1 Класс CurseVR.VoiceControl.VAD.AudioClipBuffer

Buffers audio samples and creates AudioClips when the buffer is filled or flushed.

Открытые члены

- AudioClipBuffer (int maxSampleLength, int frequency, int channels=1)
  Initializes a new instance of the AudioClipBuffer class.
- void AddSamples (float[] samples)
  Adds audio samples to the buffer.
- void Flush ()
  Flushes the buffer, creating an AudioClip with current data regardless of buffer fullness.

События

- Action< AudioClip > OnBufferFilled
  Event triggered when the buffer is filled or flushed with sufficient data to create an AudioClip.

Закрытые члены

- void CreateAndEmitAudioClip ()
  Creates an AudioClip from the current buffer content and triggers the OnBufferFilled event.
- void Reset ()
  Resets the buffer to its initial empty state.

Закрытые данные

- readonly float[] buffer
- readonly int channels
- readonly int frequency
- int writePosition
- bool isFull

6.1.1 Подробное описание

Buffers audio samples and creates AudioClips when the buffer is filled or flushed.

This class provides a mechanism to collect audio samples over time and create AudioClip instances when specific conditions are met (buffer full or manually flushed). It's primarily used to collect voice data detected by the voice activity detection system for further processing.

См. определение в файле AudioClipBuffer.cs строка 15

6.1.2 Конструктор(ы)

6.1.2.1 AudioClipBuffer()

```
CurseVR.VoiceControl.VAD.AudioClipBuffer.AudioClipBuffer (
            int maxSampleLength,
            int frequency,
            int channels = 1)
```

Initializes a new instance of the AudioClipBuffer class.

Аргументы

maxSampleLength	Maximum number of samples (per channel) to buffer
frequency	Sample rate in Hz (e.g., 16000 for 16kHz audio)
channels	Number of audio channels (1 for mono, 2 for stereo)

The total buffer size will be maxSampleLength $*$ channels floating-point values. For voice recognition, typical values might be 16000 samples at 16kHz for 1 second of audio.

См. определение в файле AudioClipBuffer.cs строка 42

6.1.3 Методы

6.1.3.1 AddSamples()

```
void CurseVR.VoiceControl.VAD.AudioClipBuffer.AddSamples (
            float[] samples)
```

Adds audio samples to the buffer.

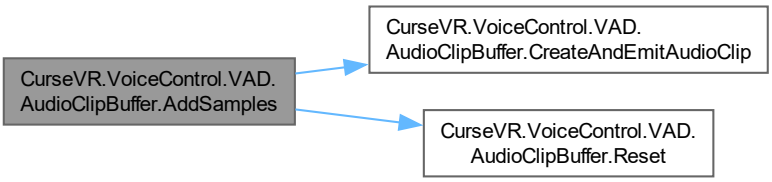
Аргументы

samples	Array of audio samples to add

This method adds audio data to the internal buffer. If the buffer becomes full, it automatically creates an AudioClip, fires the OnBufferFilled event, and resets the buffer. The samples should be in the correct format (interleaved if multi-channel) matching the buffer's configuration.

См. определение в файле AudioClipBuffer.cs строка 61

Граф вызовов:



6.1.3.2 CreateAndEmitAudioClip()

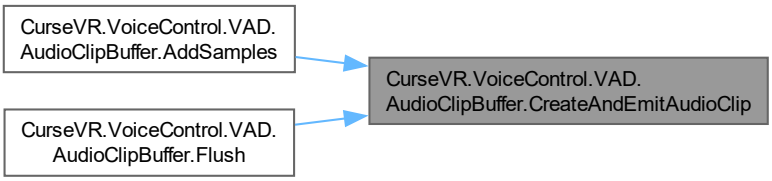
void CurseVR.VoiceControl.VAD.AudioClipBuffer.CreateAndEmitAudioClip () [private]

Creates an AudioClip from the current buffer content and triggers the OnBufferFilled event.

Creates a new Unity AudioClip with the appropriate sample rate and channel count, copies the buffered data to it, and notifies subscribers via the OnBufferFilled event.

См. определение в файле AudioClipBuffer.cs строка 102

Граф вызова функции:



6.1.3.3 Flush()

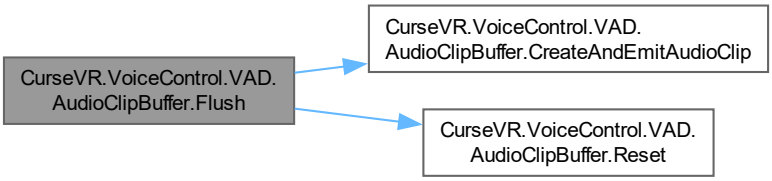
void CurseVR.VoiceControl.VAD.AudioClipBuffer.Flush ()

Flushes the buffer, creating an AudioClip with current data regardless of buffer fullness.

This is typically called when voice activity ends to process any remaining audio that hasn't filled the buffer yet. If the buffer is empty (writePosition == 0), no AudioClip will be created.

См. определение в файле AudioClipBuffer.cs строка 86

Граф вызовов:



6.1.3.4 Reset()

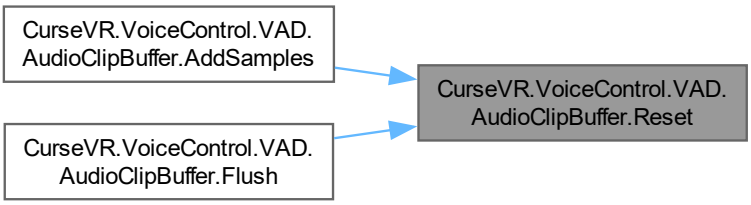
void CurseVR.VoiceControl.VAD.AudioClipBuffer.Reset () [private]

Resets the buffer to its initial empty state.

Clears all data in the buffer, resets the write position, and marks the buffer as not full. Called automatically after an AudioClip is created and the OnBufferFilled event is triggered.

См. определение в файле AudioClipBuffer.cs строка 128

Граф вызова функции:



6.1.4 Данные класса

6.1.4.1 buffer

readonly float [] CurseVR.VoiceControl.VAD.AudioClipBuffer.buffer [private]

См. определение в файле AudioClipBuffer.cs строка 17

6.1.4.2 channels

readonly int CurseVR.VoiceControl.VAD.AudioClipBuffer.channels [private]

См. определение в файле AudioClipBuffer.cs строка 18

6.1.4.3 frequency

readonly int CurseVR.VoiceControl.VAD.AudioClipBuffer.frequency [private]

См. определение в файле AudioClipBuffer.cs строка 19

6.1.4.4 isFull

bool CurseVR.VoiceControl.VAD.AudioClipBuffer.isFull [private]

См. определение в файле AudioClipBuffer.cs строка 21

6.1.4.5 writePosition

int CurseVR.VoiceControl.VAD.AudioClipBuffer.writePosition [private]

См. определение в файле AudioClipBuffer.cs строка 20

6.1.5 События

6.1.5.1 OnBufferFilled

Action<AudioClip> CurseVR.VoiceControl.VAD.AudioClipBuffer.OnBufferFilled

Event triggered when the buffer is filled or flushed with sufficient data to create an AudioClip.

Subscribers will receive the newly created AudioClip containing the buffered audio data. This is typically used to process voice data for recognition or transmission.

См. определение в файле AudioClipBuffer.cs строка 30

Объявления и описания членов класса находятся в файле:

- Assets/Scripts/VoiceControl/VAD/AudioClipBuffer.cs

6.2 Класс CurseVR.VoiceControl.Utils.AudioUtils

Utility class for audio processing operations.

Открытые статические члены

- static byte[] AudioClipToBytes (AudioClip clip)
  Converts an AudioClip to a byte array in 16-bit PCM format.
- static AudioClip BytesToAudioClip (byte[] audioData, int channels, int frequency)
  Converts raw PCM data to an AudioClip.
- static float CalculateRMSVolume (float[] samples)
  Calculates the RMS volume of an audio buffer.
- static void ApplyNoiseGate (float[] samples, float threshold)
  Applies a simple noise gate to the audio data.

6.2.1 Подробное описание

Utility class for audio processing operations.

См. определение в файле AudioUtils.cs строка 8

6.2.2 Методы

6.2.2.1 ApplyNoiseGate()

```
static void CurseVR.VoiceControl.Utils.AudioUtils.ApplyNoiseGate (
                float[] samples,
                float threshold)   [static]
```

Applies a simple noise gate to the audio data.

См. определение в файле AudioUtils.cs строка 68

6.2.2.2 AudioClipToBytes()

```
static byte[] CurseVR.VoiceControl.Utils.AudioUtils.AudioClipToBytes (
                AudioClip clip)   [static]
```

Converts an AudioClip to a byte array in 16-bit PCM format.

См. определение в файле AudioUtils.cs строка 13

6.2.2.3 BytesToAudioClip()

```
static AudioClip CurseVR.VoiceControl.Utils.AudioUtils.BytesToAudioClip (
                byte[] audioData,
                int channels,
                int frequency)   [static]
```

Converts raw PCM data to an AudioClip.

См. определение в файле AudioUtils.cs строка 34

6.2.2.4 CalculateRMSVolume()

static float CurseVR.VoiceControl.Utils.AudioUtils.CalculateRMSVolume (
float[] samples) [static]

Calculates the RMS volume of an audio buffer.

См. определение в файле AudioUtils.cs строка 54

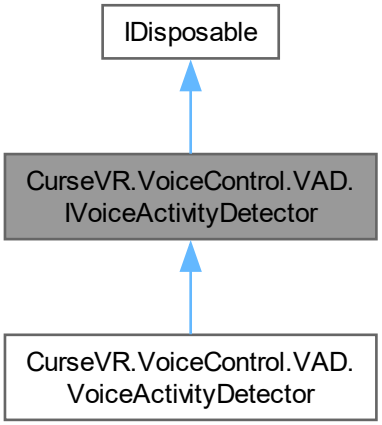
Объявления и описания членов класса находятся в файле:

- Assets/Scripts/VoiceControl/Utils/AudioUtils.cs

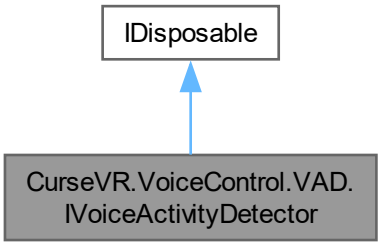
6.3 Интерфейс CurseVR.VoiceControl.VAD.IVoiceActivityDetector

Interface for voice activity detection functionality.

Граф наследования:CurseVR.VoiceControl.VAD.IVoiceActivityDetector:



Граф связей класса CurseVR.VoiceControl.VAD.IVoiceActivityDetector:



Открытые члены

- void Update ()
  Updates the voice activity detection state.

Свойства

- bool IsActive [get]
  Gets a value indicating whether voice activity is currently detected.

События

- Action< bool > OnVoiceActivityChanged
  Event triggered when voice activity state changes.

6.3.1 Подробное описание

Interface for voice activity detection functionality.

This interface defines the contract for components that detect when a user is speaking by analyzing audio input. Implementations should handle microphone data processing and determine speaking/silence transitions based on volume thresholds and timing parameters.

См. определение в файле IVoiceActivityDetector.cs строка 14

6.3.2 Методы

6.3.2.1 Update()

void CurseVR.VoiceControl.VAD.IVoiceActivityDetector.Update ()

Updates the voice activity detection state.

This method should be called regularly (typically once per frame) to process new audio data from the microphone and update the activity state. It handles reading new microphone data, analyzing volume levels, and triggering state change events when appropriate.

Замещается в CurseVR.VoiceControl.VAD.VoiceActivityDetector.

6.3.3 Полный список свойств

6.3.3.1 IsActive

bool CurseVR.VoiceControl.VAD.IVoiceActivityDetector.IsActive [get]

Gets a value indicating whether voice activity is currently detected.

True when voice is active, false when silent

Замещается в CurseVR.VoiceControl.VAD.VoiceActivityDetector.

См. определение в файле IVoiceActivityDetector.cs строка 20

6.3.4 События

6.3.4.1 OnVoiceActivityChanged

Action<bool> CurseVR.VoiceControl.VAD.IVoiceActivityDetector.OnVoiceActivityChanged

Event triggered when voice activity state changes.

The boolean parameter indicates the new state: true when voice becomes active, false when voice becomes inactive. Subscribers can use this to start/stop recording or processing audio based on detected speech.

См. определение в файле IVoiceActivityDetector.cs строка 30

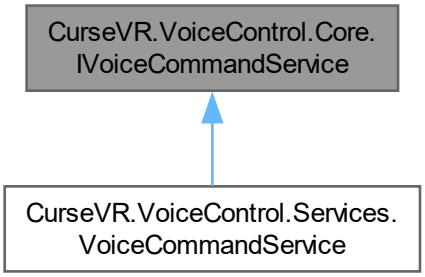
Объявления и описания членов интерфейса находятся в файле:

- Assets/Scripts/VoiceControl/VAD/IVoiceActivityDetector.cs

6.4 Интерфейс CurseVR.VoiceControl.Core.IVoiceCommandService

Interface for the voice command service that handles communication with the ASR API.

Граф наследования:CurseVR.VoiceControl.Core.IVoiceCommandService:



Открытые члены

- Task InitializeAsync (VoiceServiceConfig config)

  Initializes the voice command service with the provided configuration.
- Task SendAudioDataAsync (byte[] audioData)

  Sends audio data to the ASR service for processing.
- Task ConnectAsync ()

  Establishes a connection to the voice recognition service.
- Task DisconnectAsync ()

  Terminates the connection to the voice recognition service.

Свойства

- bool IsConnected [get]

  Gets a value indicating whether the service is currently connected to the ASR endpoint.

События

- Action< VoiceCommandData > OnCommandRecognized

  Event triggered when a voice command is recognized by the ASR service.
- Action< bool > OnConnectionStatusChanged

  Event triggered when the connection status with the ASR service changes.

6.4.1 Подробное описание

Interface for the voice command service that handles communication with the ASR API.

This interface defines the contract for communicating with an Automatic Speech Recognition (ASR) service over WebSockets. It handles initialization, connection management, and data transmission.

См. определение в файле IVoiceCommandService.cs строка 15

6.4.2 Методы

6.4.2.1 ConnectAsync()

Task CurseVR.VoiceControl.Core.IVoiceCommandService.ConnectAsync ()

Establishes a connection to the voice recognition service.

Возвращает

A task representing the asynchronous connection operation

This method should be called after initialization and before sending audio data. It establishes a WebSocket connection to the ASR service endpoint specified in the configuration.

Исключения

InvalidOperationException	Thrown when service is not initialized

Замещается в CurseVR.VoiceControl.Services.VoiceCommandService.

6.4.2.2 DisconnectAsync()

Task CurseVR.VoiceControl.Core.IVoiceCommandService.DisconnectAsync ()

Terminates the connection to the voice recognition service.

Возвращает

A task representing the asynchronous disconnection operation

This method should be called when the application no longer needs to process voice commands, such as during application shutdown or when switching scenes.

Замещается в CurseVR.VoiceControl.Services.VoiceCommandService.

6.4.2.3 InitializeAsync()

Task CurseVR.VoiceControl.Core.IVoiceCommandService.InitializeAsync (
VoiceServiceConfig config)

Initializes the voice command service with the provided configuration.

Аргументы

config	Configuration parameters for the voice service connection

Возвращает

A task representing the asynchronous initialization operation

Исключения

ArgumentNullException	Thrown when config is null
InvalidOperationException	Thrown when initialization fails

Замещается в CurseVR.VoiceControl.Services.VoiceCommandService.

6.4.2.4 SendAudioDataAsync()

Task CurseVR.VoiceControl.Core.IVoiceCommandService.SendAudioDataAsync (
byte[] audioData)

Sends audio data to the ASR service for processing.

Аргументы

audioData	Raw audio data bytes to be sent for recognition

Возвращает

A task representing the asynchronous send operation

The audio data should be in the format specified in the VoiceServiceConfig (sample rate, channels, etc.). The data will be sent over WebSocket to the ASR service.

Исключения

InvalidOperationException	Thrown when service is not connected

Замещается в CurseVR.VoiceControl.Services.VoiceCommandService.

6.4.3 Полный список свойств

6.4.3.1 IsConnected

bool CurseVR.VoiceControl.Core.IVoiceCommandService.IsConnected [get]

Gets a value indicating whether the service is currently connected to the ASR endpoint.

True if connected, false otherwise

This property should be checked before attempting to send audio data to prevent errors.

Замещается в CurseVR.VoiceControl.Services.VoiceCommandService.

См. определение в файле IVoiceCommandService.cs строка 84

6.4.4 События

6.4.4.1 OnCommandRecognized

Action<VoiceCommandData> CurseVR.VoiceControl.Core.IVoiceCommandService.OnCommandRecognized

Event triggered when a voice command is recognized by the ASR service.

Subscribers will receive a VoiceCommandData object containing the recognized command details including the command text, confidence level, and any additional parameters.

См. определение в файле IVoiceCommandService.cs строка 24

6.4.4.2 OnConnectionStatusChanged

Action<bool> CurseVR.VoiceControl.Core.IVoiceCommandService.OnConnectionStatusChanged

Event triggered when the connection status with the ASR service changes.

The boolean parameter indicates whether the connection is active (true) or inactive (false). This event should be used to update UI elements or system state based on connection status.

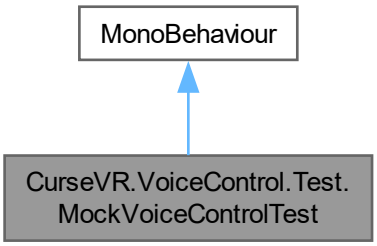
См. определение в файле IVoiceCommandService.cs строка 33

Объявления и описания членов интерфейса находятся в файле:

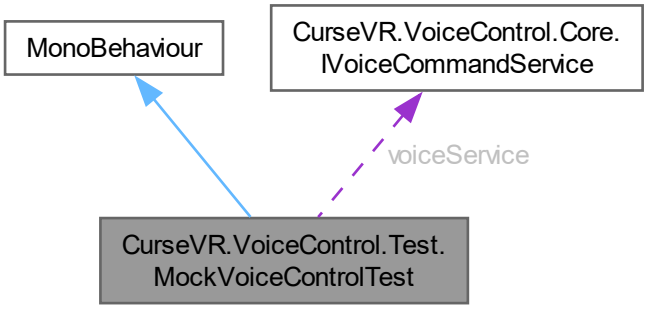
- Assets/Scripts/VoiceControl/Core/IVoiceCommandService.cs

6.5 Класс CurseVR.VoiceControl.Test.MockVoiceControlTest

Граф наследования:CurseVR.VoiceControl.Test.MockVoiceControlTest:



Граф связей класса CurseVR.VoiceControl.Test.MockVoiceControlTest:



Закрытые члены

- void Start ()
- void InitializeVoiceService ()
- void Update ()
- void HandleInteraction ()
- void SelectObject (GameObject obj)
- void DeselectObject ()
- IEnumerator ProcessMockVoiceCommand ()
- IEnumerator MonitorAudioPlayback ()
- void HandleVoiceCommand (VoiceCommandData commandData)
- IEnumerator MoveObjectRight ()
- void OnApplicationQuit ()
- void OnEnable ()
- void OnDisable ()
- void OnDestroy ()
- void CleanupVoiceService ()

Закрытые данные

- AudioClip testVoiceClip
- Material highlightMaterial
- Material defaultMaterial
- float moveDistance = 5f
- InputActionReference interactAction
- Camera mainCamera
- LayerMask interactableLayer = -1
- AudioSource voiceAudioSource
- bool playAudioOnSelect = true
- float audioVolume = 1f
- bool debugAudio = true
- bool offlineMode = false
- float offlineModeDelay = 1f
- bool debugInput = true

- GameObject selectedObject
- Material originalMaterial
- IVoiceCommandService voiceService
- bool isProcessing
- bool isQuitting
- Key interactKey = Key.E

6.5.1 Подробное описание

См. определение в файле MockVoiceControlTest.cs строка 10

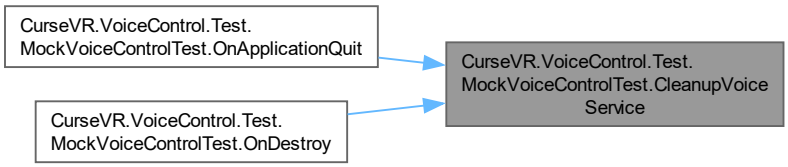
6.5.2 Методы

6.5.2.1 CleanupVoiceService()

void CurseVR.VoiceControl.Test.MockVoiceControlTest.CleanupVoiceService () [private]

См. определение в файле MockVoiceControlTest.cs строка 371

Граф вызова функции:

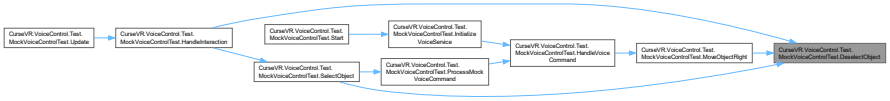


6.5.2.2 DeselectObject()

void CurseVR.VoiceControl.Test.MockVoiceControlTest.DeselectObject () [private]

См. определение в файле MockVoiceControlTest.cs строка 194

Граф вызова функции:

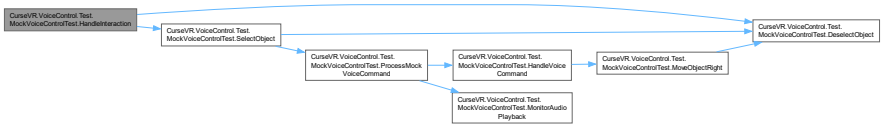


6.5.2.3 HandleInteraction()

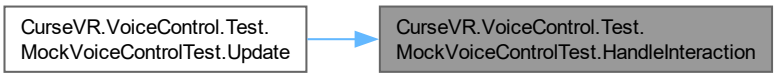
void CurseVR.VoiceControl.Test.MockVoiceControlTest.HandleInteraction () [private]

См. определение в файле MockVoiceControlTest.cs строка 131

Граф вызовов:



Граф вызова функции:



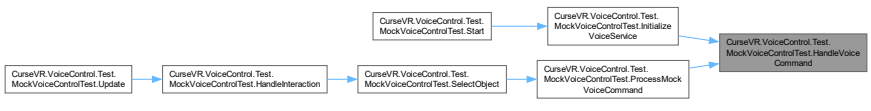
6.5.2.4 HandleVoiceCommand()

void CurseVR.VoiceControl.Test.MockVoiceControlTest.HandleVoiceCommand (
VoiceCommandData commandData) [private]

См. определение в файле MockVoiceControlTest.cs строка 302

Граф вызовов:

Граф вызова функции:



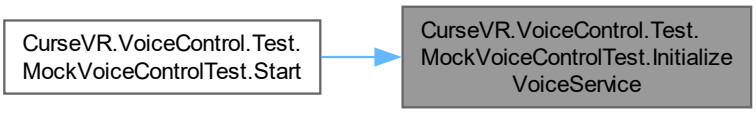
6.5.2.5 InitializeVoiceService()

void CurseVR.VoiceControl.Test.MockVoiceControlTest.InitializeVoiceService () [private]

См. определение в файле MockVoiceControlTest.cs строка 96

Граф вызовов:

Граф вызова функции:



6.5.2.6 MonitorAudioPlayback()

IEnumerator CurseVR.VoiceControl.Test.MockVoiceControlTest.MonitorAudioPlayback () [private]

См. определение в файле MockVoiceControlTest.cs строка 285

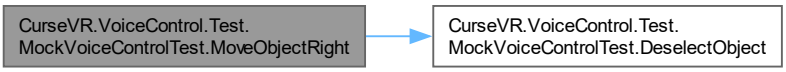
Граф вызова функции:

6.5.2.7 MoveObjectRight()

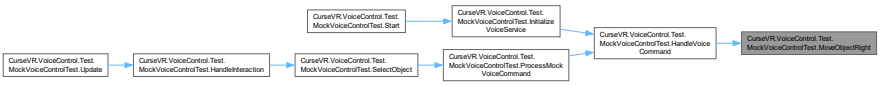
IEnumerator CurseVR.VoiceControl.Test.MockVoiceControlTest.MoveObjectRight () [private]

См. определение в файле MockVoiceControlTest.cs строка 315

Граф вызовов:



Граф вызова функции:

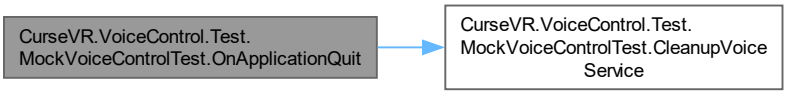


6.5.2.8 OnApplicationQuit()

void CurseVR.VoiceControl.Test.MockVoiceControlTest.OnApplicationQuit () [private]

См. определение в файле MockVoiceControlTest.cs строка 339

Граф вызовов:

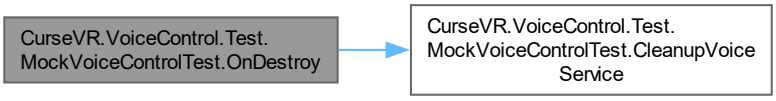


6.5.2.9 OnDestroy()

void CurseVR.VoiceControl.Test.MockVoiceControlTest.OnDestroy () [private]

См. определение в файле MockVoiceControlTest.cs строка 363

Граф вызовов:



6.5.2.10 OnDisable()

void CurseVR.VoiceControl.Test.MockVoiceControlTest.OnDisable () [private]

См. определение в файле MockVoiceControlTest.cs строка 354

6.5.2.11 OnEnable()

void CurseVR.VoiceControl.Test.MockVoiceControlTest.OnEnable () [private]

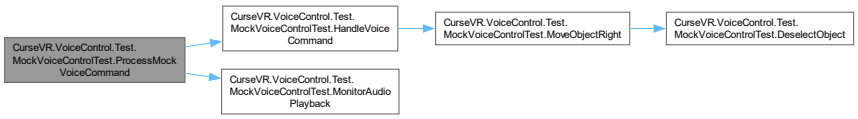
См. определение в файле MockVoiceControlTest.cs строка 345

6.5.2.12 ProcessMockVoiceCommand()

IEnumerator CurseVR.VoiceControl.Test.MockVoiceControlTest.ProcessMockVoiceCommand () [private]

См. определение в файле MockVoiceControlTest.cs строка 208

Граф вызовов:



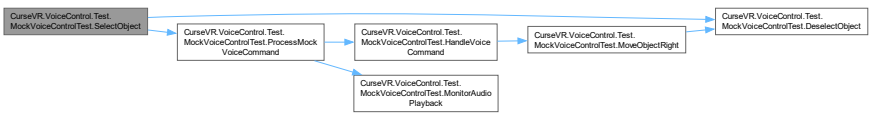
Граф вызова функции:

6.5.2.13 SelectObject()

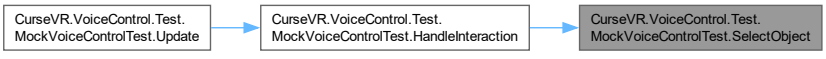
void CurseVR.VoiceControl.Test.MockVoiceControlTest.SelectObject (
GameObject obj) [private]

См. определение в файле MockVoiceControlTest.cs строка 169

Граф вызовов:



Граф вызова функции:



6.5.2.14 Start()

void CurseVR.VoiceControl.Test.MockVoiceControlTest.Start () [private]

См. определение в файле MockVoiceControlTest.cs строка 39

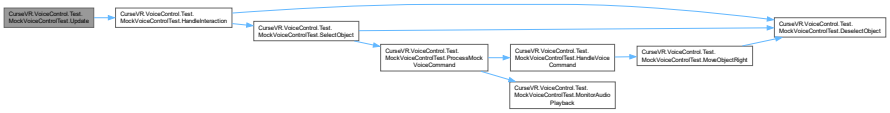
Граф вызовов:

6.5.2.15 Update()

void CurseVR.VoiceControl.Test.MockVoiceControlTest.Update () [private]

См. определение в файле MockVoiceControlTest.cs строка 118

Граф вызовов:



6.5.3 Данные класса

6.5.3.1 audioVolume

float CurseVR.VoiceControl.Test.MockVoiceControlTest.audioVolume = 1f [private]

См. определение в файле MockVoiceControlTest.cs строка 24

6.5.3.2 debugAudio

bool CurseVR.VoiceControl.Test.MockVoiceControlTest.debugAudio = true [private]

См. определение в файле MockVoiceControlTest.cs строка 25

6.5.3.3 debugInput

bool CurseVR.VoiceControl.Test.MockVoiceControlTest.debugInput = true [private]

См. определение в файле MockVoiceControlTest.cs строка 30

6.5.3.4 defaultMaterial

Material CurseVR.VoiceControl.Test.MockVoiceControlTest.defaultMaterial [private]

См. определение в файле MockVoiceControlTest.cs строка 15

6.5.3.5 highlightMaterial

Material CurseVR.VoiceControl.Test.MockVoiceControlTest.highlightMaterial [private]

См. определение в файле MockVoiceControlTest.cs строка 14

6.5.3.6 interactableLayer

LayerMask CurseVR.VoiceControl.Test.MockVoiceControlTest.interactableLayer = -1 [private]

См. определение в файле MockVoiceControlTest.cs строка 19

6.5.3.7 interactAction

InputActionReference CurseVR.VoiceControl.Test.MockVoiceControlTest.interactAction [private]

См. определение в файле MockVoiceControlTest.cs строка 17

6.5.3.8 interactKey

Key CurseVR.VoiceControl.Test.MockVoiceControlTest.interactKey = Key.E [private]

См. определение в файле MockVoiceControlTest.cs строка 37

6.5.3.9 isProcessing

bool CurseVR.VoiceControl.Test.MockVoiceControlTest.isProcessing [private]

См. определение в файле MockVoiceControlTest.cs строка 35

6.5.3.10 isQuitting

bool CurseVR.VoiceControl.Test.MockVoiceControlTest.isQuitting [private]

См. определение в файле MockVoiceControlTest.cs строка 36

6.5.3.11 mainCamera

Camera CurseVR.VoiceControl.Test.MockVoiceControlTest.mainCamera [private]

См. определение в файле MockVoiceControlTest.cs строка 18

6.5.3.12 moveDistance

float CurseVR.VoiceControl.Test.MockVoiceControlTest.moveDistance = 5f [private]

См. определение в файле MockVoiceControlTest.cs строка 16

6.5.3.13 offlineMode

bool CurseVR.VoiceControl.Test.MockVoiceControlTest.offlineMode = false [private]

См. определение в файле MockVoiceControlTest.cs строка 28

6.5.3.14 offlineModeDelay

float CurseVR.VoiceControl.Test.MockVoiceControlTest.offlineModeDelay = 1f [private]

См. определение в файле MockVoiceControlTest.cs строка 29

6.5.3.15 originalMaterial

Material CurseVR.VoiceControl.Test.MockVoiceControlTest.originalMaterial [private]

См. определение в файле MockVoiceControlTest.cs строка 33

6.5.3.16 playAudioOnSelect

bool CurseVR.VoiceControl.Test.MockVoiceControlTest.playAudioOnSelect = true [private]

См. определение в файле MockVoiceControlTest.cs строка 23

6.5.3.17 selectedObject

GameObject CurseVR.VoiceControl.Test.MockVoiceControlTest.selectedObject [private]

См. определение в файле MockVoiceControlTest.cs строка 32

6.5.3.18 testVoiceClip

AudioClip CurseVR.VoiceControl.Test.MockVoiceControlTest.testVoiceClip [private]

См. определение в файле MockVoiceControlTest.cs строка 13

6.5.3.19 voiceAudioSource

AudioSource CurseVR.VoiceControl.Test.MockVoiceControlTest.voiceAudioSource [private]

См. определение в файле MockVoiceControlTest.cs строка 22

6.5.3.20 voiceService

IVoiceCommandService CurseVR.VoiceControl.Test.MockVoiceControlTest.voiceService [private]

См. определение в файле MockVoiceControlTest.cs строка 34

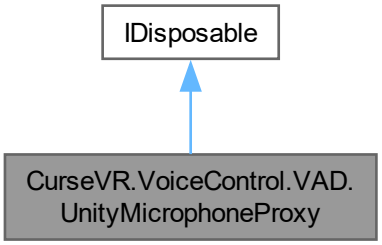
Объявления и описания членов класса находятся в файле:

- Assets/Scripts/VoiceControl/Test/MockVoiceControlTest.cs

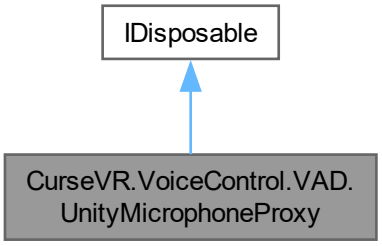
6.6 Класс CurseVR.VoiceControl.VAD.UnityMicrophoneProxy

Provides an abstraction over Unity's Microphone API for simplified access to audio input.

Граф наследования:CurseVR.VoiceControl.VAD.UnityMicrophoneProxy:



Граф связей класса CurseVR.VoiceControl.VAD.UnityMicrophoneProxy:



Открытые члены

- UnityMicrophoneProxy (string deviceName=null, int frequency=44100)
  Initializes a new instance of the UnityMicrophoneProxy class.
- void Dispose ()
  Disposes of resources used by the microphone proxy.

Свойства

- AudioClip AudioClip [get]
  Gets the AudioClip containing the microphone input data.
- int SampleRate [get]
  Gets the sample rate of the audio recording.

Закрытые члены

- void InitializeMicrophone ()
  Initializes the microphone and begins recording.

Закрытые данные

- readonly string deviceName
- AudioClip audioClip
- readonly int frequency
- readonly int sampleRate

6.6.1 Подробное описание

Provides an abstraction over Unity's Microphone API for simplified access to audio input.

This class encapsulates the initialization, management, and cleanup of Unity's microphone system. It creates a continuous recording AudioClip that can be accessed for real-time audio processing. The proxy pattern allows for easier testing and a more controlled interface to the Unity API.

См. определение в файле UnityMicrophoneProxy.cs строка 14

6.6.2 Конструктор(ы)

6.6.2.1 UnityMicrophoneProxy()

```
CurseVR.VoiceControl.VAD.UnityMicrophoneProxy.UnityMicrophoneProxy (
            string deviceName = null,
            int frequency = 44100)
```

Initializes a new instance of the UnityMicrophoneProxy class.

Аргументы

deviceName	Name of the microphone device to use, or null for default device
frequency	Requested sample rate for recording in Hz

If deviceName is null, the first available microphone device will be used. The default frequency of 44100Hz is CD quality, but can be reduced for voice recognition (e.g., 16000Hz is common for speech processing).

Исключения

InvalidOperationException	Thrown when no microphone devices are available

См. определение в файле UnityMicrophoneProxy.cs строка 48

Граф вызовов:

6.6.3 Методы

6.6.3.1 Dispose()

void CurseVR.VoiceControl.VAD.UnityMicrophoneProxy.Dispose ()

Disposes of resources used by the microphone proxy.

Stops any active recording and destroys the AudioClip to prevent memory leaks. This should be called when the proxy is no longer needed, typically during application cleanup or scene transitions.

См. определение в файле UnityMicrophoneProxy.cs строка 90

6.6.3.2 InitializeMicrophone()

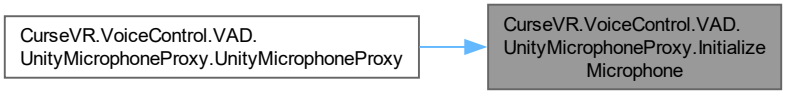
void CurseVR.VoiceControl.VAD.UnityMicrophoneProxy.InitializeMicrophone () [private]

Initializes the microphone and begins recording.

This method stops any existing recording, creates a new AudioClip for recording, and starts the microphone in loop mode. It waits until recording has actually started before returning.

См. определение в файле UnityMicrophoneProxy.cs строка 70

Граф вызова функции:



6.6.4 Данные класса

6.6.4.1 audioClip

AudioClip CurseVR.VoiceControl.VAD.UnityMicrophoneProxy.audioClip [private]

См. определение в файле UnityMicrophoneProxy.cs строка 17

6.6.4.2 deviceName

readonly string CurseVR.VoiceControl.VAD.UnityMicrophoneProxy.deviceName [private]

См. определение в файле UnityMicrophoneProxy.cs строка 16

6.6.4.3 frequency

readonly int CurseVR.VoiceControl.VAD.UnityMicrophoneProxy.frequency [private]

См. определение в файле UnityMicrophoneProxy.cs строка 18

6.6.4.4 sampleRate

readonly int CurseVR.VoiceControl.VAD.UnityMicrophoneProxy.sampleRate [private]

См. определение в файле UnityMicrophoneProxy.cs строка 19

6.6.5 Полный список свойств

6.6.5.1 AudioClip

AudioClip CurseVR.VoiceControl.VAD.UnityMicrophoneProxy.AudioClip [get]

Gets the AudioClip containing the microphone input data.

The AudioClip recording from the microphone

This is a continuous recording in a circular buffer. Use Microphone.GetPosition to determine the current recording position within the clip.

См. определение в файле UnityMicrophoneProxy.cs строка 29

6.6.5.2 SampleRate

int CurseVR.VoiceControl.VAD.UnityMicrophoneProxy.SampleRate [get]

Gets the sample rate of the audio recording.

Sample rate in Hz

См. определение в файле UnityMicrophoneProxy.cs строка 35

Объявления и описания членов класса находятся в файле:

- Assets/Scripts/VoiceControl/VAD/UnityMicrophoneProxy.cs

6.7 Класс CurseVR.VoiceControl.Models.VADParameters

Parameters for the Voice Activity Detection (VAD) system.

Открытые атрибуты

- int BufferSize = 2048
  Size of the audio buffer in samples.
- float MaxActiveDurationSeconds = 10f
  Maximum duration of continuous voice activity in seconds.
- float MaxQueueingTimeSeconds = 0.1f
  Maximum time to queue audio data before processing in seconds.
- float MinQueueingTimeSeconds = 0.05f
  Minimum time to queue audio data before processing in seconds.
- float ActiveVolumeThreshold = 0.1f
  Volume threshold for voice activity detection.
- float ActivationRateThreshold = 0.6f
  Rate threshold for activation.
- float InactivationRateThreshold = 0.4f
  Rate threshold for inactivation.
- float ActivationIntervalSeconds = 0.1f
  Time interval for activation check in seconds.
- float InactivationIntervalSeconds = 0.3f
  Time interval for inactivation check in seconds.

6.7.1 Подробное описание

Parameters for the Voice Activity Detection (VAD) system.

This class contains all configuration parameters for the voice activity detection system. It controls audio buffer sizes, thresholds for detecting when speech begins and ends, timing parameters, and other settings that affect VAD sensitivity and responsiveness. These values can be adjusted in the Unity Inspector to fine-tune voice detection for different environments and microphones.

См. определение в файле VADParameters.cs строка 16

6.7.2 Данные класса

6.7.2.1 ActivationIntervalSeconds

float CurseVR.VoiceControl.Models.VADParameters.ActivationIntervalSeconds = 0.1f

Time interval for activation check in seconds.

The minimum time that must elapse between voice activation events. This prevents rapid toggling of the active state due to fluctuating audio levels near the threshold.

См. определение в файле VADParameters.cs строка 111

6.7.2.2 ActivationRateThreshold

float CurseVR.VoiceControl.Models.VADParameters.ActivationRateThreshold = 0.6f

Rate threshold for activation.

The proportion of audio frames that must exceed the volume threshold within a detection window to trigger voice activation. Higher values make activation more resistant to brief spikes in volume.

См. определение в файле VADParameters.cs строка 88

6.7.2.3 ActiveVolumeThreshold

float CurseVR.VoiceControl.Models.VADParameters.ActiveVolumeThreshold = 0.1f

Volume threshold for voice activity detection.

The RMS (Root Mean Square) volume level that must be exceeded for audio to be considered speech rather than background noise. Higher values require louder speech but reduce false activations. This should be calibrated based on the microphone and ambient noise levels.

См. определение в файле VADParameters.cs строка 76

6.7.2.4 BufferSize

int CurseVR.VoiceControl.Models.VADParameters.BufferSize = 2048

Size of the audio buffer in samples.

Determines how much audio data is processed at once. Larger values may improve detection accuracy but increase latency. At typical speech sample rates (16kHz), a value of 2048 represents about 128ms of audio.

См. определение в файле VADParameters.cs строка 28

6.7.2.5 InactivationIntervalSeconds

float CurseVR.VoiceControl.Models.VADParameters.InactivationIntervalSeconds = 0.3f

Time interval for inactivation check in seconds.

The minimum time that must elapse between voice deactivation events. A higher value makes the system more tolerant of brief pauses in speech, preventing a single utterance from being split into multiple activations.

См. определение в файле VADParameters.cs строка 122

6.7.2.6 InactivationRateThreshold

float CurseVR.VoiceControl.Models.VADParameters.InactivationRateThreshold = 0.4f

Rate threshold for inactivation.

The proportion of audio frames that must fall below the volume threshold within a detection window to trigger voice deactivation. Lower values make the system more responsive to pauses in speech.

См. определение в файле VADParameters.cs строка 100

6.7.2.7 MaxActiveDurationSeconds

float CurseVR.VoiceControl.Models.VADParameters.MaxActiveDurationSeconds = 10f

Maximum duration of continuous voice activity in seconds.

Sets a limit on how long a single voice activation can last before being forcibly terminated. This prevents runaway recordings in noisy environments or when the deactivation threshold isn't met.

См. определение в файле VADParameters.cs строка 40

6.7.2.8 MaxQueueingTimeSeconds

float CurseVR.VoiceControl.Models.VADParameters.MaxQueueingTimeSeconds = 0.1f

Maximum time to queue audio data before processing in seconds.

The upper limit on how long audio data will be queued before being processed. Helps ensure that even in difficult detection scenarios, data will eventually be processed.

См. определение в файле VADParameters.cs строка 51

6.7.2.9 MinQueueingTimeSeconds

float CurseVR.VoiceControl.Models.VADParameters.MinQueueingTimeSeconds = 0.05f

Minimum time to queue audio data before processing in seconds.

The lower limit on how long audio data must be queued before processing starts. This helps prevent processing very small chunks of audio which could reduce recognition accuracy.

См. определение в файле VADParameters.cs строка 62

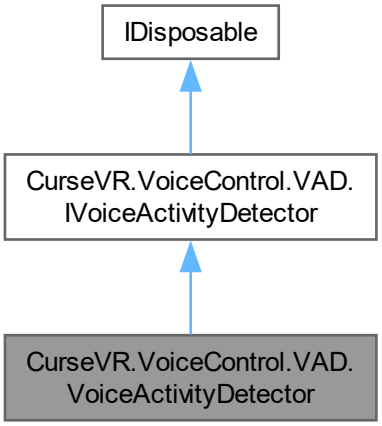
Объявления и описания членов класса находятся в файле:

- Assets/Scripts/VoiceControl/Models/VADParameters.cs

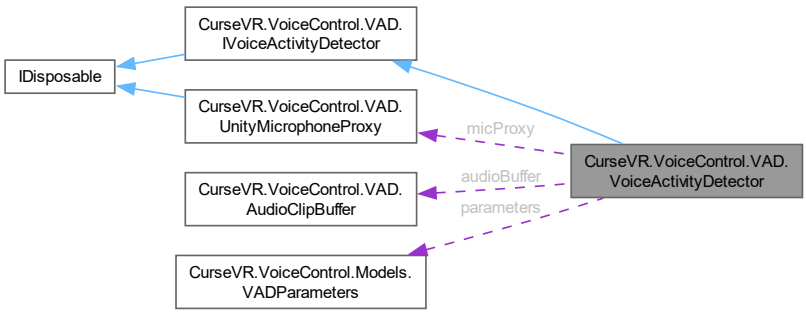
6.8 Класс CurseVR.VoiceControl.VAD.VoiceActivityDetector

Implements voice activity detection by analyzing microphone input volume.

Граф наследования:CurseVR.VoiceControl.VAD.VoiceActivityDetector:



Граф связей класса CurseVR.VoiceControl.VAD.VoiceActivityDetector:



Открытые члены

- VoiceActivityDetector (UnityMicrophoneProxy micProxy, AudioClipBuffer audioBuffer, VADParameters parameters)

  Initializes a new instance of the VoiceActivityDetector class.
- void Update ()

  Updates the voice activity detection state.
  This method should be called regularly (typically once per frame) to process new audio data from the microphone and update the activity state. It handles reading new microphone data, analyzing volume levels, and triggering state change events when appropriate.
- void Dispose ()

Свойства

- bool IsActive [get]

  Gets a value indicating whether voice activity is currently detected.
  True when voice is active, false when silent

События

- Action< bool > OnVoiceActivityChanged

События унаследованные от CurseVR.VoiceControl.VAD.IVoiceActivityDetector

- Action< bool > OnVoiceActivityChanged

  Event triggered when voice activity state changes.

Закрытые члены

- int GetSamplesToRead (int currentPosition)

  Calculates how many new audio samples should be read from the microphone.
- void ProcessAudioData (float[] samples, int sampleCount)

  Processes a batch of audio data to detect voice activity.
- float CalculateVolume (float[] samples, int sampleCount)

  Calculates the Root Mean Square (RMS) volume level of an audio sample batch.

Закрытые данные

- readonly UnityMicrophoneProxy micProxy
- readonly AudioClipBuffer audioBuffer
- readonly VADParameters parameters
- readonly float[] sampleBuffer
- float lastActiveTime
- float lastInactiveTime
- bool isActive
- int lastReadPosition

6.8.1 Подробное описание

Implements voice activity detection by analyzing microphone input volume.

This class monitors audio input from a microphone to detect when a user starts and stops speaking. It uses volume thresholds and timing parameters to determine state transitions and prevent rapid toggling between active and inactive states. When voice activity is detected, audio data is captured and stored in a buffer for further processing.

См. определение в файле VoiceActivityDetector.cs строка 16

6.8.2 Конструктор(ы)

6.8.2.1 VoiceActivityDetector()

```
CurseVR.VoiceControl.VAD.VoiceActivityDetector.VoiceActivityDetector (
            UnityMicrophoneProxy micProxy,
            AudioClipBuffer audioBuffer,
            VADParameters parameters)
```

Initializes a new instance of the VoiceActivityDetector class.

Аргументы

micProxy	Proxy for the Unity microphone system
audioBuffer	Buffer to store captured audio samples
parameters	Configuration parameters for voice activity detection

Исключения

ArgumentNullException	Thrown if any parameter is null

The constructor initializes internal state and prepares the detector for processing. Initial timestamps are set to ensure proper debouncing at startup.

См. определение в файле VoiceActivityDetector.cs строка 45

6.8.3 Методы

6.8.3.1 CalculateVolume()

```
float CurseVR.VoiceControl.VAD.VoiceActivityDetector.CalculateVolume (
            float[] samples,
            int sampleCount)   [private]
```

Calculates the Root Mean Square (RMS) volume level of an audio sample batch.

Аргументы

samples	Array of audio samples
sampleCount	Number of samples to analyze

Возвращает

RMS volume level, typically between 0 and 1

См. определение в файле VoiceActivityDetector.cs строка 142

Граф вызова функции:

6.8.3.2 Dispose()

```
void CurseVR.VoiceControl.VAD.VoiceActivityDetector.Dispose ()
```

См. определение в файле VoiceActivityDetector.cs строка 154

6.8.3.3 GetSamplesToRead()

int CurseVR.VoiceControl.VAD.VoiceActivityDetector.GetSamplesToRead (
int currentPosition) [private]

Calculates how many new audio samples should be read from the microphone.

Аргументы

currentPosition	Current position in the microphone buffer

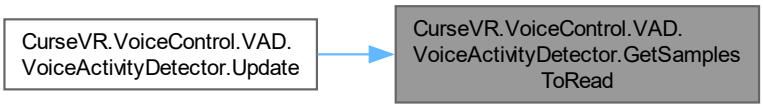
Возвращает

Number of samples to read

Handles the circular buffer wrapping that occurs in Unity's microphone system.

См. определение в файле VoiceActivityDetector.cs строка 83

Граф вызова функции:



6.8.3.4 ProcessAudioData()

```
void CurseVR.VoiceControl.VAD.VoiceActivityDetector.ProcessAudioData (
                float[] samples,
                int sampleCount)   [private]
```

Processes a batch of audio data to detect voice activity.

Аргументы

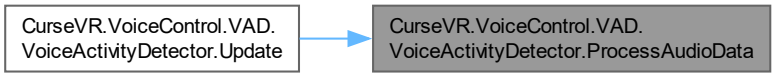
samples	Array of audio samples
sampleCount	Number of samples to process

This method analyzes volume levels to determine if voice is active, handles state transitions with appropriate timing intervals, and stores active audio in the buffer for later processing.

См. определение в файле VoiceActivityDetector.cs строка 103

Граф вызовов:

Граф вызова функции:



6.8.3.5 Update()

void CurseVR.VoiceControl.VAD.VoiceActivityDetector.Update ()

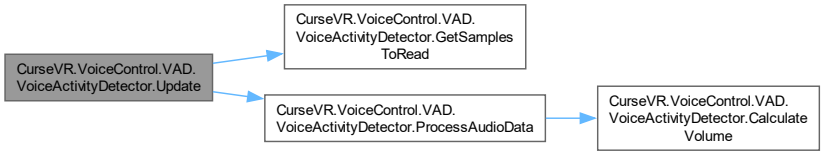
Updates the voice activity detection state.

This method should be called regularly (typically once per frame) to process new audio data from the microphone and update the activity state. It handles reading new microphone data, analyzing volume levels, and triggering state change events when appropriate.

Замещает CurseVR.VoiceControl.VAD.IVoiceActivityDetector.

См. определение в файле VoiceActivityDetector.cs строка 57

Граф вызовов:



6.8.4 Данные класса

6.8.4.1 audioBuffer

readonly AudioClipBuffer CurseVR.VoiceControl.VAD.VoiceActivityDetector.audioBuffer [private]

См. определение в файле VoiceActivityDetector.cs строка 19

6.8.4.2 isActive

bool CurseVR.VoiceControl.VAD.VoiceActivityDetector.isActive [private]

См. определение в файле VoiceActivityDetector.cs строка 25

6.8.4.3 lastActiveTime

float CurseVR.VoiceControl.VAD.VoiceActivityDetector.lastActiveTime [private]

См. определение в файле VoiceActivityDetector.cs строка 23

6.8.4.4 lastInactiveTime

float CurseVR.VoiceControl.VAD.VoiceActivityDetector.lastInactiveTime [private]

См. определение в файле VoiceActivityDetector.cs строка 24

6.8.4.5 lastReadPosition

int CurseVR.VoiceControl.VAD.VoiceActivityDetector.lastReadPosition [private]

См. определение в файле VoiceActivityDetector.cs строка 26

6.8.4.6 micProxy

readonly UnityMicrophoneProxy CurseVR.VoiceControl.VAD.VoiceActivityDetector.micProxy [private]

См. определение в файле VoiceActivityDetector.cs строка 18

6.8.4.7 parameters

readonly VADParameters CurseVR.VoiceControl.VAD.VoiceActivityDetector.parameters [private]

См. определение в файле VoiceActivityDetector.cs строка 20

6.8.4.8 sampleBuffer

readonly float [] CurseVR.VoiceControl.VAD.VoiceActivityDetector.sampleBuffer [private]

См. определение в файле VoiceActivityDetector.cs строка 21

6.8.5 Полный список свойств

6.8.5.1 IsActive

bool CurseVR.VoiceControl.VAD.VoiceActivityDetector.IsActive [get]

Gets a value indicating whether voice activity is currently detected.

True when voice is active, false when silent

Замещает CurseVR.VoiceControl.VAD.IVoiceActivityDetector.

См. определение в файле VoiceActivityDetector.cs строка 29

6.8.6 События

6.8.6.1 OnVoiceActivityChanged

Action<bool> CurseVR.VoiceControl.VAD.VoiceActivityDetector.OnVoiceActivityChanged

См. определение в файле VoiceActivityDetector.cs строка 32

Объявления и описания членов класса находятся в файле:

- Assets/Scripts/VoiceControl/VAD/VoiceActivityDetector.cs

6.9 Класс CurseVR.VoiceControl.Models.VoiceCommandData

Data structure for voice commands received from the ASR service.

Открытые члены

- VoiceCommandData ()
  Initializes a new instance of the VoiceCommandData class.

Свойства

- string CommandType [get, set]
  Gets or sets the type of command recognized.
- string RawText [get, set]
  Gets or sets the raw text transcribed from speech.
- float Confidence [get, set]
  Gets or sets the confidence score of the recognition.
- Dictionary< string, object > Parameters [get, set]
  Gets or sets additional parameters associated with the command.
- DateTime Timestamp [get, set]
  Gets or sets the timestamp when the command was recognized.

6.9.1 Подробное описание

Data structure for voice commands received from the ASR service.

This class represents a processed voice command after recognition by the ASR service. It contains the recognized command type, the original transcribed text, confidence level, and any parameters extracted from the command. This object is passed to subscribers of the OnCommandRecognized event in the voice command service.

См. определение в файле VoiceCommandData.cs строка 17

6.9.2 Конструктор(ы)

6.9.2.1 VoiceCommandData()

CurseVR.VoiceControl.Models.VoiceCommandData.VoiceCommandData ()

Initializes a new instance of the VoiceCommandData class.

Creates a new voice command data object with an empty parameter dictionary and sets the timestamp to the current UTC time.

См. определение в файле VoiceCommandData.cs строка 81

6.9.3 Полный список свойств

6.9.3.1 CommandType

string CurseVR.VoiceControl.Models.VoiceCommandData.CommandType [get], [set]

Gets or sets the type of command recognized.

A string identifier for the command category (e.g., "move", "select", "open")

This field represents the classified intent of the voice command after natural language processing. It is used to determine which action should be performed in response to the command.

См. определение в файле VoiceCommandData.cs строка 28

6.9.3.2 Confidence

float CurseVR.VoiceControl.Models.VoiceCommandData.Confidence [get], [set]

Gets or sets the confidence score of the recognition.

A float between 0.0 and 1.0 representing recognition confidence

Higher values indicate greater confidence in the accuracy of the transcription. This can be used to filter out potentially misheard commands or to request confirmation from the user when confidence is low.

См. определение в файле VoiceCommandData.cs строка 50

6.9.3.3 Parameters

Dictionary<string, object> CurseVR.VoiceControl.Models.VoiceCommandData.Parameters [get], [set]

Gets or sets additional parameters associated with the command.

A dictionary mapping parameter names to their values

These parameters are extracted from the speech by the natural language understanding component. For example, in "move forward three meters", "direction" might be "forward" and "distance" might be 3.0. Parameter values can be of various types (string, number, boolean).

См. определение в файле VoiceCommandData.cs строка 61

6.9.3.4 RawText

string CurseVR.VoiceControl.Models.VoiceCommandData.RawText [get], [set]

Gets or sets the raw text transcribed from speech.

The unprocessed text as transcribed by the ASR service

This is the direct output from the speech-to-text process before any intent recognition or parameter extraction. Useful for debugging or displaying exactly what was heard.

См. определение в файле VoiceCommandData.cs строка 39

6.9.3.5 Timestamp

DateTime CurseVR.VoiceControl.Models.VoiceCommandData.Timestamp [get], [set]

Gets or sets the timestamp when the command was recognized.

DateTime indicating when the command was processed

This timestamp can be used to implement command timeout or to synchronize commands with other events in the application. It is set to UTC time when the object is created.

См. определение в файле VoiceCommandData.cs строка 72

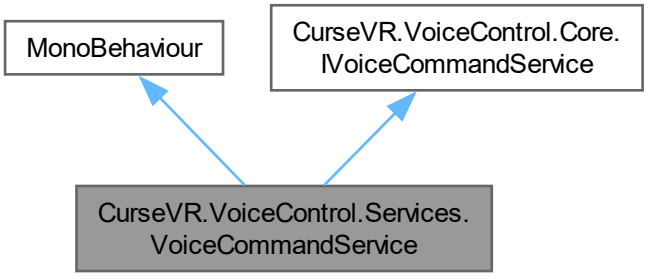
Объявления и описания членов класса находятся в файле:

- Assets/Scripts/VoiceControl/Models/VoiceCommandData.cs

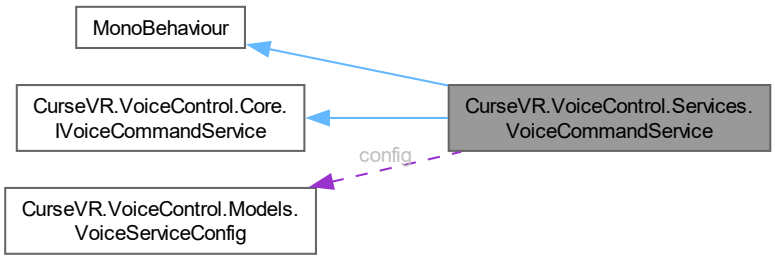
6.10 Класс CurseVR.VoiceControl.Services.VoiceCommandService

Implementation of the voice command service using WebSocket communication.

Граф наследования:CurseVR.VoiceControl.Services.VoiceCommandService:



Граф связей класса CurseVR.VoiceControl.Services.VoiceCommandService:



Открытые члены

- async Task InitializeAsync (VoiceServiceConfig config)

  Initializes the voice command service with the provided configuration.
- async Task ConnectAsync ()

  Establishes a connection to the voice recognition service.
- async Task DisconnectAsync ()

  Terminates the connection to the voice recognition service.
- async Task SendAudioDataAsync (byte[] audioData)

  Sends audio data to the ASR service for processing.

Свойства

- bool IsConnected [get]

  Gets a value indicating whether the service is currently connected to the ASR endpoint.

События

- Action< VoiceCommandData > OnCommandRecognized
- Action< bool > OnConnectionStatusChanged

События унаследованные от CurseVR.VoiceControl.Core.IVoiceCommandService

- Action< VoiceCommandData > OnCommandRecognized

  Event triggered when a voice command is recognized by the ASR service.
- Action< bool > OnConnectionStatusChanged

  Event triggered when the connection status with the ASR service changes.

Закрытые члены

- async void TryReconnect ()
- void ProcessWebSocketMessage (string message)
- void Update ()
- void OnDestroy ()

Закрытые данные

- WebSocket webSocket
- VoiceServiceConfig config
- bool isInitialized
- int reconnectAttempts

6.10.1 Подробное описание

Implementation of the voice command service using WebSocket communication.

См. определение в файле VoiceCommandService.cs строка 14

6.10.2 Методы

6.10.2.1 ConnectAsync()

async Task CurseVR.VoiceControl.Services.VoiceCommandService.ConnectAsync ()

Establishes a connection to the voice recognition service.

Возвращает

A task representing the asynchronous connection operation

This method should be called after initialization and before sending audio data. It establishes a WebSocket connection to the ASR service endpoint specified in the configuration.

Исключения

InvalidOperationException	Thrown when service is not initialized

Замещает CurseVR.VoiceControl.Core.IVoiceCommandService.

См. определение в файле VoiceCommandService.cs строка 77

Граф вызова функции:

6.10.2.2 DisconnectAsync()

async Task CurseVR.VoiceControl.Services.VoiceCommandService.DisconnectAsync ()

Terminates the connection to the voice recognition service.

Возвращает

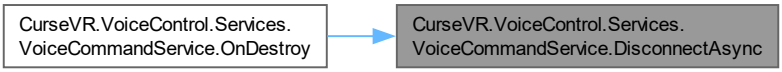
A task representing the asynchronous disconnection operation

This method should be called when the application no longer needs to process voice commands, such as during application shutdown or when switching scenes.

Замещает CurseVR.VoiceControl.Core.IVoiceCommandService.

См. определение в файле VoiceCommandService.cs строка 92

Граф вызова функции:



6.10.2.3 InitializeAsync()

async Task CurseVR.VoiceControl.Services.VoiceCommandService.InitializeAsync (
VoiceServiceConfig config)

Initializes the voice command service with the provided configuration.

Аргументы

config	Configuration parameters for the voice service connection

Возвращает

A task representing the asynchronous initialization operation

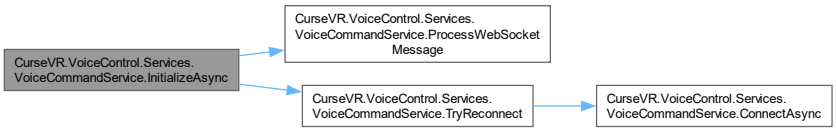
Исключения

ArgumentNullException	Thrown when config is null
InvalidOperationException	Thrown when initialization fails

Замещает CurseVR.VoiceControl.Core.IVoiceCommandService.

См. определение в файле VoiceCommandService.cs строка 26

Граф вызовов:

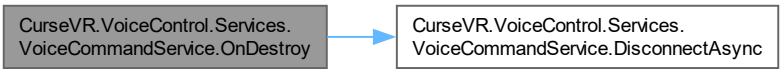


6.10.2.4 OnDestroy()

void CurseVR.VoiceControl.Services.VoiceCommandService.OnDestroy () [private]

См. определение в файле VoiceCommandService.cs строка 152

Граф вызовов:

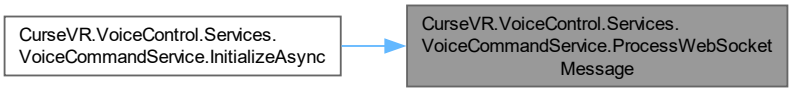


6.10.2.5 ProcessWebSocketMessage()

void CurseVR.VoiceControl.Services.VoiceCommandService.ProcessWebSocketMessage (
string message) [private]

См. определение в файле VoiceCommandService.cs строка 126

Граф вызова функции:



6.10.2.6 SendAudioDataAsync()

async Task CurseVR.VoiceControl.Services.VoiceCommandService.SendAudioDataAsync (
byte[] audioData)

Sends audio data to the ASR service for processing.

Аргументы

audioData	Raw audio data bytes to be sent for recognition

Возвращает

A task representing the asynchronous send operation

The audio data should be in the format specified in the VoiceServiceConfig (sample rate, channels, etc.). The data will be sent over WebSocket to the ASR service.

Исключения

InvalidOperationException	Thrown when service is not connected

Замещает CurseVR.VoiceControl.Core.IVoiceCommandService.

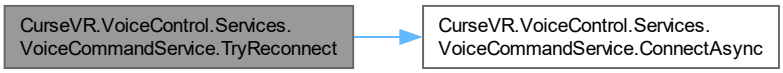
См. определение в файле VoiceCommandService.cs строка 100

6.10.2.7 TryReconnect()

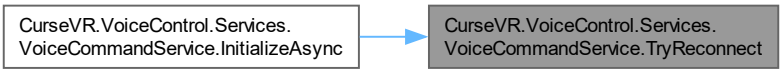
async void CurseVR.VoiceControl.Services.VoiceCommandService.TryReconnect () [private]

См. определение в файле VoiceCommandService.cs строка 111

Граф вызовов:



Граф вызова функции:



6.10.2.8 Update()

void CurseVR.VoiceControl.Services.VoiceCommandService.Update () [private]

См. определение в файле VoiceCommandService.cs строка 142

6.10.3 Данные класса

6.10.3.1 config

VoiceServiceConfig CurseVR.VoiceControl.Services.VoiceCommandService.config [private]

См. определение в файле VoiceCommandService.cs строка 17

6.10.3.2 isInitialized

bool CurseVR.VoiceControl.Services.VoiceCommandService.isInitialized [private]

См. определение в файле VoiceCommandService.cs строка 18

6.10.3.3 reconnectAttempts

int CurseVR.VoiceControl.Services.VoiceCommandService.reconnectAttempts [private]

См. определение в файле VoiceCommandService.cs строка 19

6.10.3.4 webSocket

WebSocket CurseVR.VoiceControl.Services.VoiceCommandService.webSocket [private]

См. определение в файле VoiceCommandService.cs строка 16

6.10.4 Полный список свойств

6.10.4.1 IsConnected

bool CurseVR.VoiceControl.Services.VoiceCommandService.IsConnected [get]

Gets a value indicating whether the service is currently connected to the ASR endpoint.

True if connected, false otherwise

This property should be checked before attempting to send audio data to prevent errors.

Замещает CurseVR.VoiceControl.Core.IVoiceCommandService.

См. определение в файле VoiceCommandService.cs строка 24

6.10.5 События

6.10.5.1 OnCommandRecognized

Action<VoiceCommandData> CurseVR.VoiceControl.Services.VoiceCommandService.OnCommandRecognized

См. определение в файле VoiceCommandService.cs строка 21

6.10.5.2 OnConnectionStatusChanged

Action<bool> CurseVR.VoiceControl.Services.VoiceCommandService.OnConnectionStatusChanged

См. определение в файле VoiceCommandService.cs строка 22

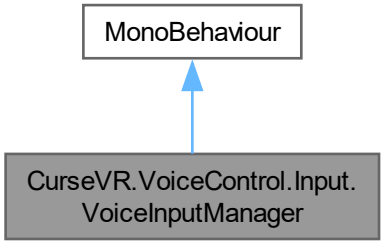
Объявления и описания членов класса находятся в файле:

- Assets/Scripts/VoiceControl/Services/VoiceCommandService.cs

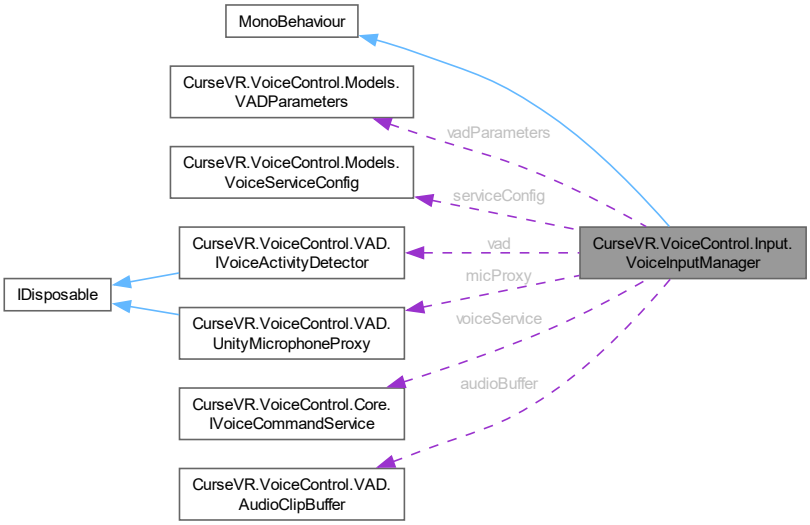
6.11 Класс CurseVR.VoiceControl.Input.VoiceInputManager

Manages voice input and integrates with Unity's Input System.

Граф наследования:CurseVR.VoiceControl.Input.VoiceInputManager:



Граф связей класса CurseVR.VoiceControl.Input.VoiceInputManager:



Закрытые члены

- void Start ()
- async void InitializeVoiceService ()
- void InitializeVAD ()
- void InitializeInputActions ()
- void HandleVoiceInactive (AudioClip clip)
- void HandleVoiceCommand (VoiceCommandData commandData)
- System.Collections.IEnumerator ToggleAction (InputAction action)
- void HandleConnectionStatus (bool isConnected)
- void Update ()
- void OnDestroy ()

Закрытые данные

- InputActionAsset inputActions
- VADParameters vadParameters
- AudioSource debugAudioSource
- VoiceServiceConfig serviceConfig
- IVoiceActivityDetector vad
- UnityMicrophoneProxy micProxy
- IVoiceCommandService voiceService
- AudioClipBuffer audioBuffer
- bool isProcessingVoice
- Dictionary< string, InputActionReference > actionReferences = new Dictionary<string, Input↩ ActionReference>()

6.11.1 Подробное описание

Manages voice input and integrates with Unity's Input System.

См. определение в файле VoiceInputManager.cs строка 16

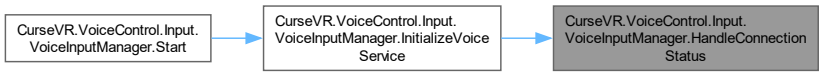
6.11.2 Методы

6.11.2.1 HandleConnectionStatus()

void CurseVR.VoiceControl.Input.VoiceInputManager.HandleConnectionStatus (
bool isConnected) [private]

См. определение в файле VoiceInputManager.cs строка 152

Граф вызова функции:



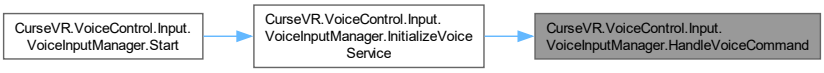
6.11.2.2 HandleVoiceCommand()

void CurseVR.VoiceControl.Input.VoiceInputManager.HandleVoiceCommand (
VoiceCommandData commandData) [private]

См. определение в файле VoiceInputManager.cs строка 114

Граф вызовов:

Граф вызова функции:

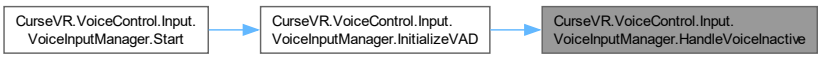


6.11.2.3 HandleVoiceInactive()

void CurseVR.VoiceControl.Input.VoiceInputManager.HandleVoiceInactive (
AudioClip clip) [private]

См. определение в файле VoiceInputManager.cs строка 86

Граф вызова функции:

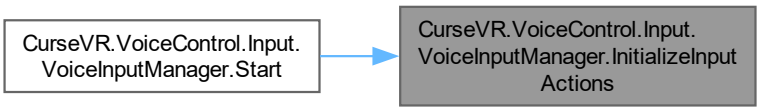


6.11.2.4 InitializeInputActions()

void CurseVR.VoiceControl.Input.VoiceInputManager.InitializeInputActions () [private]

См. определение в файле VoiceInputManager.cs строка 69

Граф вызова функции:

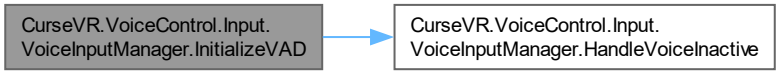


6.11.2.5 InitializeVAD()

void CurseVR.VoiceControl.Input.VoiceInputManager.InitializeVAD () [private]

См. определение в файле VoiceInputManager.cs строка 49

Граф вызовов:



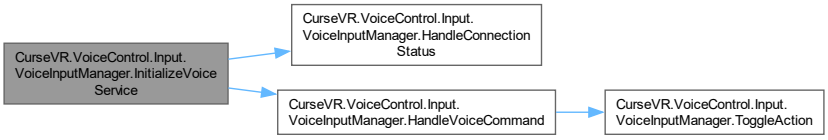
Граф вызова функции:

6.11.2.6 InitializeVoiceService()

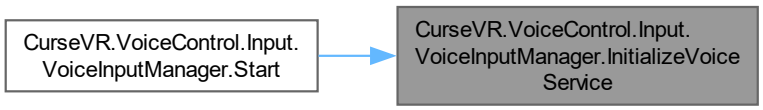
async void CurseVR.VoiceControl.Input.VoiceInputManager.InitializeVoiceService () [private]

См. определение в файле VoiceInputManager.cs строка 39

Граф вызовов:



Граф вызова функции:



6.11.2.7 OnDestroy()

void CurseVR.VoiceControl.Input.VoiceInputManager.OnDestroy () [private]

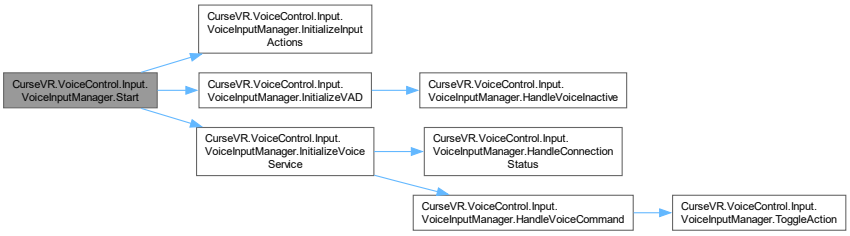
См. определение в файле VoiceInputManager.cs строка 163

6.11.2.8 Start()

void CurseVR.VoiceControl.Input.VoiceInputManager.Start () [private]

См. определение в файле VoiceInputManager.cs строка 32

Граф вызовов:

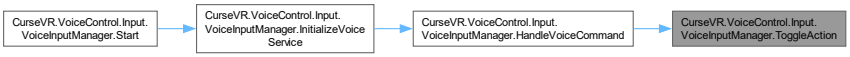


6.11.2.9 ToggleAction()

System.Collections.IEnumerator CurseVR.VoiceControl.Input.VoiceInputManager.ToggleAction (
InputAction action) [private]

См. определение в файле VoiceInputManager.cs строка 141

Граф вызова функции:



6.11.2.10 Update()

void CurseVR.VoiceControl.Input.VoiceInputManager.Update () [private]

См. определение в файле VoiceInputManager.cs строка 158

6.11.3 Данные класса

6.11.3.1 actionReferences

Dictionary<string, InputActionReference> CurseVR.VoiceControl.Input.VoiceInputManager.actionReferences = new Dictionary<string, InputActionReference>() [private]

См. определение в файле VoiceInputManager.cs строка 30

6.11.3.2 audioBuffer

AudioClipBuffer CurseVR.VoiceControl.Input.VoiceInputManager.audioBuffer [private]

См. определение в файле VoiceInputManager.cs строка 26

6.11.3.3 debugAudioSource

AudioSource CurseVR.VoiceControl.Input.VoiceInputManager.debugAudioSource [private]

См. определение в файле VoiceInputManager.cs строка 20

6.11.3.4 inputActions

InputActionAsset CurseVR.VoiceControl.Input.VoiceInputManager.inputActions [private]

См. определение в файле VoiceInputManager.cs строка 18

6.11.3.5 isProcessingVoice

bool CurseVR.VoiceControl.Input.VoiceInputManager.isProcessingVoice [private]

См. определение в файле VoiceInputManager.cs строка 27

6.11.3.6 micProxy

UnityMicrophoneProxy CurseVR.VoiceControl.Input.VoiceInputManager.micProxy [private]

См. определение в файле VoiceInputManager.cs строка 24

6.11.3.7 serviceConfig

VoiceServiceConfig CurseVR.VoiceControl.Input.VoiceInputManager.serviceConfig [private]

См. определение в файле VoiceInputManager.cs строка 21

6.11.3.8 vad

IVoiceActivityDetector CurseVR.VoiceControl.Input.VoiceInputManager.vad [private]

См. определение в файле VoiceInputManager.cs строка 23

6.11.3.9 vadParameters

VADParameters CurseVR.VoiceControl.Input.VoiceInputManager.vadParameters [private]

См. определение в файле VoiceInputManager.cs строка 19

6.11.3.10 voiceService

IVoiceCommandService CurseVR.VoiceControl.Input.VoiceInputManager.voiceService [private]

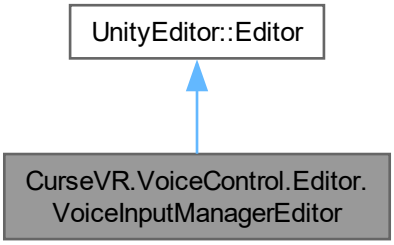
См. определение в файле VoiceInputManager.cs строка 25

Объявления и описания членов класса находятся в файле:

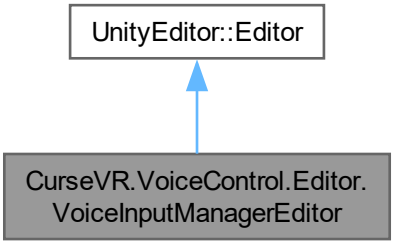
- Assets/Scripts/VoiceControl/Input/VoiceInputManager.cs

6.12 Класс CurseVR.VoiceControl.Editor.VoiceInputManagerEditor

Граф наследования:CurseVR.VoiceControl.Editor.VoiceInputManagerEditor:



Граф связей класса CurseVR.VoiceControl.Editor.VoiceInputManagerEditor:



Открытые члены

- override void OnInspectorGUI ()

Закрытые члены

- void TestMicrophone ()
- void TestWebSocketConnection ()

Закрытые данные

- bool showDebugSettings = false
- bool showServiceConfig = false
- bool showVADSettings = false

6.12.1 Подробное описание

См. определение в файле VoiceInputManagerEditor.cs строка 8

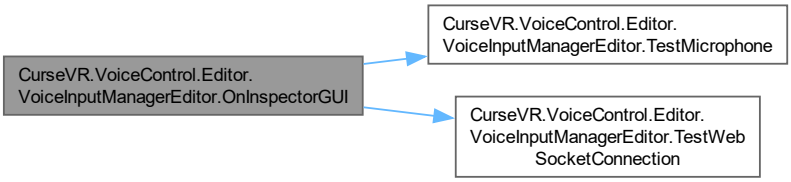
6.12.2 Методы

6.12.2.1 OnInspectorGUI()

override void CurseVR.VoiceControl.Editor.VoiceInputManagerEditor.OnInspectorGUI ()

См. определение в файле VoiceInputManagerEditor.cs строка 14

Граф вызовов:

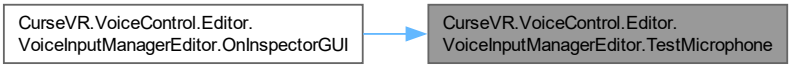


6.12.2.2 TestMicrophone()

void CurseVR.VoiceControl.Editor.VoiceInputManagerEditor.TestMicrophone () [private]

См. определение в файле VoiceInputManagerEditor.cs строка 71

Граф вызова функции:

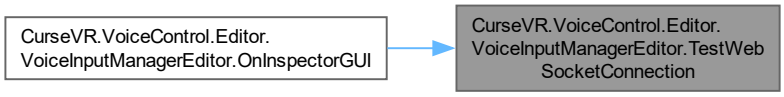


6.12.2.3 TestWebSocketConnection()

void CurseVR.VoiceControl.Editor.VoiceInputManagerEditor.TestWebSocketConnection () [private]

См. определение в файле VoiceInputManagerEditor.cs строка 90

Граф вызова функции:



6.12.3 Данные класса

6.12.3.1 showDebugSettings

bool CurseVR.VoiceControl.Editor.VoiceInputManagerEditor.showDebugSettings = false [private]

См. определение в файле VoiceInputManagerEditor.cs строка 10

6.12.3.2 showServiceConfig

bool CurseVR.VoiceControl.Editor.VoiceInputManagerEditor.showServiceConfig = false [private]

См. определение в файле VoiceInputManagerEditor.cs строка 11

6.12.3.3 showVADSettings

bool CurseVR.VoiceControl.Editor.VoiceInputManagerEditor.showVADSettings = false [private]

См. определение в файле VoiceInputManagerEditor.cs строка 12

Объявления и описания членов класса находятся в файле:

- Assets/Scripts/VoiceControl/Editor/VoiceInputManagerEditor.cs

6.13 Класс CurseVR.VoiceControl.Models.VoiceServiceConfig

Configuration settings for the voice command service.

Свойства

- string WebSocketUrl = "ws://localhost:8000/ws/asr/" [get, set]
  Gets or sets the base URL for the ASR WebSocket service.
- string ClientId = "unity_client" [get, set]
  Gets or sets the client identifier for this connection.
- int SampleRate = 16000 [get, set]
  Gets or sets the sample rate for audio recording.
- int Channels = 1 [get, set]
  Gets or sets the number of audio channels.
- int BufferSize = 2048 [get, set]
  Gets or sets the size of the audio buffer in samples.
- int MaxReconnectAttempts = 3 [get, set]
  Gets or sets the maximum reconnection attempts.
- int ReconnectDelayMs = 1000 [get, set]
  Gets or sets the delay between reconnection attempts in milliseconds.

6.13.1 Подробное описание

Configuration settings for the voice command service.

This serializable class contains all the connection and audio parameters needed to establish and maintain a connection with an Automatic Speech Recognition (ASR) WebSocket service. It can be configured in the Unity Inspector and saved as a ScriptableObject or directly in a MonoBehaviour component.

См. определение в файле VoiceServiceConfig.cs строка 16

6.13.2 Полный список свойств

6.13.2.1 BufferSize

int CurseVR.VoiceControl.Models.VoiceServiceConfig.BufferSize = 2048 [get], [set]

Gets or sets the size of the audio buffer in samples.

Buffer size in samples (per channel)

This determines how much audio data is collected before sending to the ASR service. Larger buffers introduce more latency but may improve recognition accuracy. A value of 2048 at 16 kHz represents about 128ms of audio.

См. определение в файле VoiceServiceConfig.cs строка 68

6.13.2.2 Channels

int CurseVR.VoiceControl.Models.VoiceServiceConfig.Channels = 1 [get], [set]

Gets or sets the number of audio channels.

1 for mono, 2 for stereo

For voice recognition, mono (1 channel) is typically sufficient and reduces bandwidth requirements. The ASR service must support the specified channel count.

См. определение в файле VoiceServiceConfig.cs строка 57

6.13.2.3 ClientId

string CurseVR.VoiceControl.Models.VoiceServiceConfig.ClientId = "unity_client" [get], [set]

Gets or sets the client identifier for this connection.

A unique string identifying this client instance

The client ID can be used by the server to track different connections and maintain session state. In multi-user environments, this should be unique per user.

См. определение в файле VoiceServiceConfig.cs строка 37

6.13.2.4 MaxReconnectAttempts

int CurseVR.VoiceControl.Models.VoiceServiceConfig.MaxReconnectAttempts = 3 [get], [set]

Gets or sets the maximum reconnection attempts.

Number of reconnection attempts before giving up

If the WebSocket connection is lost, the system will automatically attempt to reconnect this many times before requiring manual intervention.

См. определение в файле VoiceServiceConfig.cs строка 78

6.13.2.5 ReconnectDelayMs

int CurseVR.VoiceControl.Models.VoiceServiceConfig.ReconnectDelayMs = 1000 [get], [set]

Gets or sets the delay between reconnection attempts in milliseconds.

Delay time in milliseconds

After a failed connection attempt, the system will wait this many milliseconds before trying again. This helps prevent overloading the server with rapid connection requests.

См. определение в файле VoiceServiceConfig.cs строка 89

6.13.2.6 SampleRate

int CurseVR.VoiceControl.Models.VoiceServiceConfig.SampleRate = 16000 [get], [set]

Gets or sets the sample rate for audio recording.

Sample rate in Hz (samples per second)

16000 Hz (16 kHz) is a common sample rate for speech recognition systems. This must match the expected input format of the ASR service.

См. определение в файле VoiceServiceConfig.cs строка 47

6.13.2.7 WebSocketUrl

string CurseVR.VoiceControl.Models.VoiceServiceConfig.WebSocketUrl = "ws://localhost:8000/ws/asr/" [get], [set]

Gets or sets the base URL for the ASR WebSocket service.

WebSocket URL in the format "ws://hostname:port/path"

For local development, the default "ws://localhost:8000/ws/asr/" connects to a service running on the same machine. For production, this should be changed to the actual server address.

См. определение в файле VoiceServiceConfig.cs строка 27

Объявления и описания членов класса находятся в файле:

- Assets/Scripts/VoiceControl/Models/VoiceServiceConfig.cs

Глава 7

Файлы

7.1 Файл Assets/Scripts/VoiceControl/Core/IVoiceCommandService.cs

Классы

- interface CurseVR.VoiceControl.Core.IVoiceCommandService
  Interface for the voice command service that handles communication with the ASR API.

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.Core

7.2 IVoiceCommandService.cs

См. документацию.

```
00001 using System;
00002 using System.Threading.Tasks;
00003 using UnityEngine;
00004 using CurseVR.VoiceControl.Models;
00005
00006 namespace CurseVR.VoiceControl.Core
00007 {
00015     public interface IVoiceCommandService
00016     {
00024         event Action<VoiceCommandData> OnCommandRecognized;
00025
00033         event Action<bool> OnConnectionStatusChanged;
00034
00042         Task InitializeAsync(VoiceServiceConfig config);
00043
00054         Task SendAudioDataAsync(byte[] audioData);
00055
00065         Task ConnectAsync();
00066
00075         Task DisconnectAsync();
00076
00084         bool IsConnected { get; }
00085     }
00086 }
```

7.3 Файл Assets/Scripts/VoiceControl/Editor/VoiceInputManagerEditor.cs

Классы

- class CurseVR.VoiceControl.Editor.VoiceInputManagerEditor

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.Editor

7.4 VoiceInputManagerEditor.cs

См. документацию.

```
00001 using UnityEngine;
00002 using UnityEditor;
00003 using CurseVR.VoiceControl.Input;
00004
00005 namespace CurseVR.VoiceControl.Editor
00006 {
00007     [CustomEditor(typeof(VoiceInputManager))]
00008     public class VoiceInputManagerEditor : UnityEditor.Editor
00009     {
00010         private bool showDebugSettings = false;
00011         private bool showServiceConfig = false;
00012         private bool showVADSettings = false;
00013
00014         public override void OnInspectorGUI()
00015         {
00016             var manager = (VoiceInputManager)target;
00017
00018             EditorGUILayout.Space(10);
00019             EditorGUILayout.LabelField("Voice Input Manager Settings", EditorStyles.boldLabel);
00020
00021             // Input Actions
00022             EditorGUILayout.PropertyField(serializedObject.FindProperty("inputActions"));
00023
00024             // Service Configuration
00025             showServiceConfig = EditorGUILayout.Foldout(showServiceConfig, "Service Configuration");
00026             if (showServiceConfig)
00027             {
00028                 EditorGUI.indentLevel++;
00029                 EditorGUILayout.PropertyField(serializedObject.FindProperty("serviceConfig"));
00030                 EditorGUI.indentLevel--;
00031             }
00032
00033             // VAD Settings
00034             showVADSettings = EditorGUILayout.Foldout(showVADSettings, "Voice Activity Detection Settings");
00035             if (showVADSettings)
00036             {
00037                 EditorGUI.indentLevel++;
00038                 EditorGUILayout.PropertyField(serializedObject.FindProperty("vadParameters"));
00039                 EditorGUI.indentLevel--;
00040             }
00041
00042             // Debug Settings
00043             showDebugSettings = EditorGUILayout.Foldout(showDebugSettings, "Debug Settings");
00044             if (showDebugSettings)
00045             {
00046                 EditorGUI.indentLevel++;
00047                 EditorGUILayout.PropertyField(serializedObject.FindProperty("debugAudioSource"));
00048                 EditorGUI.indentLevel--;
00049             }
00050
00051             serializedObject.ApplyModifiedProperties();
00052
00053             // Add test buttons in debug mode
00054             if (showDebugSettings)
00055             {
00056                 EditorGUILayout.Space(10);
```

```
00057             EditorGUILayout.LabelField("Debug Tools", EditorStyles.boldLabel);
00058
00059             if (GUILayout.Button("Test Microphone"))
00060             {
00061                 TestMicrophone();
00062             }
00063
00064             if (GUILayout.Button("Test WebSocket Connection"))
00065             {
00066                 TestWebSocketConnection();
00067             }
00068         }
00069     }
00070
00071     private void TestMicrophone()
00072     {
00073         var devices = Microphone.devices;
00074         if (devices.Length == 0)
00075         {
00076             EditorUtility.DisplayDialog("Microphone Test",
00077                 "No microphone devices found!", "OK");
00078             return;
00079         }
00080
00081         string deviceList = "Available Microphones:\n\n";
00082         foreach (var device in devices)
00083         {
00084             deviceList += $"- {device}\n";
00085         }
00086
00087         EditorUtility.DisplayDialog("Microphone Test", deviceList, "OK");
00088     }
00089
00090     private void TestWebSocketConnection()
00091     {
00092         // This would need to be implemented to actually test the connection
00093         EditorUtility.DisplayDialog("WebSocket Test",
00094             "WebSocket connection test not implemented in editor.", "OK");
00095     }
00096 }
00097 }
```

7.5 Файл Assets/Scripts/VoiceControl/Input/VoiceInputManager.cs

Классы

- class CurseVR.VoiceControl.Input.VoiceInputManager

  Manages voice input and integrates with Unity's Input System.

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.Input

7.6 VoiceInputManager.cs

См. документацию.

```
00001 using UnityEngine;
00002 using UnityEngine.InputSystem;
00003 using UnityEngine.InputSystem.LowLevel;
00004 using CurseVR.VoiceControl.Core;
00005 using CurseVR.VoiceControl.Models;
00006 using CurseVR.VoiceControl.Services;
00007 using CurseVR.VoiceControl.VAD;
00008 using System.Threading.Tasks;
00009 using System.Collections.Generic;
00010
00011 namespace CurseVR.VoiceControl.Input
```

```
00012 {
00016     public class VoiceInputManager : MonoBehaviour
00017     {
00018         [SerializeField] private InputActionAsset inputActions;
00019         [SerializeField] private VADParameters vadParameters;
00020         [SerializeField] private AudioSource debugAudioSource;
00021         [SerializeField] private VoiceServiceConfig serviceConfig;
00022 
00023         private IVoiceActivityDetector vad;
00024         private UnityMicrophoneProxy micProxy;
00025         private IVoiceCommandService voiceService;
00026         private AudioClipBuffer audioBuffer;
00027         private bool isProcessingVoice;
00028 
00029         // Dictionary to store action references by command type
00030         private Dictionary<string, InputActionReference> actionReferences = new Dictionary<string,
      InputActionReference>();
00031 
00032         private void Start()
00033         {
00034             InitializeVoiceService();
00035             InitializeVAD();
00036             InitializeInputActions();
00037         }
00038 
00039         private async void InitializeVoiceService()
00040         {
00041             voiceService = gameObject.AddComponent<VoiceCommandService>();
00042             await voiceService.InitializeAsync(serviceConfig);
00043             await voiceService.ConnectAsync();
00044 
00045             voiceService.OnCommandRecognized += HandleVoiceCommand;
00046             voiceService.OnConnectionStatusChanged += HandleConnectionStatus;
00047         }
00048 
00049         private void InitializeVAD()
00050         {
00051             micProxy = new UnityMicrophoneProxy();
00052 
00053             audioBuffer = new AudioClipBuffer(
00054                 maxSampleLength: (int)(vadParameters.MaxActiveDurationSeconds * micProxy.SampleRate),
00055                 frequency: micProxy.SampleRate);
00056 
00057             audioBuffer.OnBufferFilled += HandleVoiceInactive;
00058 
00059             vad = new VoiceActivityDetector(micProxy, audioBuffer, vadParameters);
00060 
00061             if (debugAudioSource != null)
00062             {
00063                 debugAudioSource.clip = micProxy.AudioClip;
00064                 debugAudioSource.loop = true;
00065                 debugAudioSource.Play();
00066             }
00067         }
00068 
00069         private void InitializeInputActions()
00070         {
00071             var voiceActionMap = inputActions.FindActionMap("Voice");
00072             if (voiceActionMap != null)
00073             {
00074                 voiceActionMap.Enable();
00075 
00076                 // Create action references for all actions in the map
00077                 foreach (var action in voiceActionMap)
00078                 {
00079                     var actionRef = ScriptableObject.CreateInstance<InputActionReference>();
00080                     actionRef.Set(action);
00081                     actionReferences[action.name] = actionRef;
00082                 }
00083             }
00084         }
00085 
00086         private void HandleVoiceInactive(AudioClip clip)
00087         {
00088             if (!isProcessingVoice || clip == null) return;
00089 
00090             // Convert AudioClip to byte array
00091             float[] samples = new float[clip.samples * clip.channels];
00092             clip.GetData(samples, 0);
00093 
00094             // Convert to 16-bit PCM
00095             byte[] audioData = new byte[samples.Length * 2];
00096             for (int i = 0; i < samples.Length; i++)
00097             {
00098                 short value = (short)(samples[i] * 32767f);
00099                 audioData[i * 2] = (byte)(value & 0xff);
00100                 audioData[i * 2 + 1] = (byte)((value » 8) & 0xff);
```

```
00101           }
00102
00103           // Send to voice service
00104           _ = voiceService.SendAudioDataAsync(audioData);
00105
00106           // Play debug audio if needed
00107           if (debugAudioSource != null)
00108           {
00109               debugAudioSource.clip = clip;
00110               debugAudioSource.Play();
00111           }
00112       }
00113
00114       private void HandleVoiceCommand(VoiceCommandData commandData)
00115       {
00116           Debug.Log($"Voice command recognized: {commandData.CommandType}");
00117
00118           if (actionReferences.TryGetValue(commandData.CommandType, out var actionRef))
00119           {
00120               var action = actionRef.action;
00121               if (action != null)
00122               {
00123                   Debug.Log($"Triggering input action: {action.name}");
00124
00125                   // Simply enable the action - this will trigger any subscribers
00126                   // that are monitoring for this action's state changes
00127                   action.Enable();
00128
00129                   // This is a workaround - Unity's InputSystem doesn't provide
00130                   // a public API to directly trigger an action from code
00131                   // We're just toggling the action's state via enable/disable
00132                   StartCoroutine(ToggleAction(action));
00133               }
00134           }
00135           else
00136           {
00137               Debug.LogWarning($"No action found for command type: {commandData.CommandType}");
00138           }
00139       }
00140
00141       private System.Collections.IEnumerator ToggleAction(InputAction action)
00142       {
00143           // Wait a frame to ensure the enable event is processed
00144           yield return null;
00145
00146           // Disable and re-enable the action to trigger another change
00147           action.Disable();
00148           yield return null;
00149           action.Enable();
00150       }
00151
00152       private void HandleConnectionStatus(bool isConnected)
00153       {
00154           Debug.Log($"Voice service connection status: {isConnected}");
00155           isProcessingVoice = isConnected;
00156       }
00157
00158       private void Update()
00159       {
00160           vad?.Update();
00161       }
00162
00163       private void OnDestroy()
00164       {
00165           vad?.Dispose();
00166           micProxy?.Dispose();
00167           _ = voiceService?.DisconnectAsync();
00168
00169           // Clean up action references
00170           foreach (var actionRef in actionReferences.Values)
00171           {
00172               if (actionRef != null)
00173               {
00174                   Destroy(actionRef);
00175               }
00176           }
00177           actionReferences.Clear();
00178       }
00179   }
00180 }
```

7.7 Файл Assets/Scripts/VoiceControl/Models/VADParameters.cs

Классы

- class CurseVR.VoiceControl.Models.VADParameters
  Parameters for the Voice Activity Detection (VAD) system.

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.Models

7.8 VADParameters.cs

См. документацию.

```
00001 using UnityEngine;
00002
00003 namespace CurseVR.VoiceControl.Models
00004 {
00015     [System.Serializable]
00016     public class VADParameters
00017     {
00026         [Header("Audio Settings")]
00027         [Tooltip("Size of the audio buffer in samples")]
00028         public int BufferSize = 2048;
00029
00038         [Header("Timing")]
00039         [Tooltip("Maximum duration of voice activity in seconds")]
00040         public float MaxActiveDurationSeconds = 10f;
00041
00050         [Tooltip("Maximum time to queue audio data before processing")]
00051         public float MaxQueueingTimeSeconds = 0.1f;
00052
00061         [Tooltip("Minimum time to queue audio data before processing")]
00062         public float MinQueueingTimeSeconds = 0.05f;
00063
00073         [Header("Detection")]
00074         [Tooltip("Volume threshold for voice activity detection")]
00075         [Range(0f, 1f)]
00076         public float ActiveVolumeThreshold = 0.1f;
00077
00086         [Tooltip("Rate threshold for activation")]
00087         [Range(0f, 1f)]
00088         public float ActivationRateThreshold = 0.6f;
00089
00098         [Tooltip("Rate threshold for inactivation")]
00099         [Range(0f, 1f)]
00100         public float InactivationRateThreshold = 0.4f;
00101
00109         [Header("Intervals")]
00110         [Tooltip("Time interval for activation check")]
00111         public float ActivationIntervalSeconds = 0.1f;
00112
00121         [Tooltip("Time interval for inactivation check")]
00122         public float InactivationIntervalSeconds = 0.3f;
00123     }
00124 }
```

7.9 Файл Assets/Scripts/VoiceControl/Models/VoiceCommandData.cs

Классы

- class CurseVR.VoiceControl.Models.VoiceCommandData
  Data structure for voice commands received from the ASR service.

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.Models

7.10 VoiceCommandData.cs

См. документацию.

```
00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 namespace CurseVR.VoiceControl.Models
00006 {
00016     [Serializable]
00017     public class VoiceCommandData
00018     {
00028         public string CommandType { get; set; }
00029
00039         public string RawText { get; set; }
00040
00050         public float Confidence { get; set; }
00051
00061         public Dictionary<string, object> Parameters { get; set; }
00062
00072         public DateTime Timestamp { get; set; }
00073
00081         public VoiceCommandData()
00082         {
00083             Parameters = new Dictionary<string, object>();
00084             Timestamp = DateTime.UtcNow;
00085         }
00086     }
00087 }
```

7.11 Файл Assets/Scripts/VoiceControl/Models/VoiceServiceConfig.cs

Классы

- class CurseVR.VoiceControl.Models.VoiceServiceConfig

  Configuration settings for the voice command service.

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.Models

7.12 VoiceServiceConfig.cs

См. документацию.

```
00001 using System;
00002 using UnityEngine;
00003
00004 namespace CurseVR.VoiceControl.Models
00005 {
00015     [Serializable]
00016     public class VoiceServiceConfig
00017     {
```

```
00027         public string WebSocketUrl { get; set; } = "ws://localhost:8000/ws/asr/";
00028
00037         public string ClientId { get; set; } = "unity_client";
00038
00047         public int SampleRate { get; set; } = 16000;
00048
00057         public int Channels { get; set; } = 1;
00058
00068         public int BufferSize { get; set; } = 2048;
00069
00078         public int MaxReconnectAttempts { get; set; } = 3;
00079
00089         public int ReconnectDelayMs { get; set; } = 1000;
00090     }
00091 }
```

7.13 Файл Assets/Scripts/VoiceControl/README.md

7.14 Файл Assets/Scripts/VoiceControl/Services/VoiceCommandService.cs

Классы

- class CurseVR.VoiceControl.Services.VoiceCommandService

  Implementation of the voice command service using WebSocket communication.

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.Services

7.15 VoiceCommandService.cs

См. документацию.

```
00001 using System;
00002 using System.Threading.Tasks;
00003 using UnityEngine;
00004 using NativeWebSocket;
00005 using CurseVR.VoiceControl.Core;
00006 using CurseVR.VoiceControl.Models;
00007 using Newtonsoft.Json;
00008
00009 namespace CurseVR.VoiceControl.Services
00010 {
00014     public class VoiceCommandService : MonoBehaviour, IVoiceCommandService
00015     {
00016         private WebSocket webSocket;
00017         private VoiceServiceConfig config;
00018         private bool isInitialized;
00019         private int reconnectAttempts;
00020
00021         public event Action<VoiceCommandData> OnCommandRecognized;
00022         public event Action<bool> OnConnectionStatusChanged;
00023
00024         public bool IsConnected => webSocket?.State == WebSocketState.Open;
00025
00026         public async Task InitializeAsync(VoiceServiceConfig config)
00027         {
00028             this.config = config ?? throw new ArgumentNullException(nameof(config));
00029
00030             string fullUrl = $"{config.WebSocketUrl}{config.ClientId}";
00031             webSocket = new WebSocket(fullUrl);
00032
```

```
00033           var tcs = new TaskCompletionSource<bool>();
00034
00035           webSocket.OnOpen += () =>
00036           {
00037               Debug.Log("Connected to ASR service");
00038               OnConnectionStatusChanged?.Invoke(true);
00039               reconnectAttempts = 0;
00040               tcs.TrySetResult(true);
00041           };
00042
00043           webSocket.OnClose += (e) =>
00044           {
00045               Debug.Log($"Disconnected from ASR service: {e}");
00046               OnConnectionStatusChanged?.Invoke(false);
00047               TryReconnect();
00048               tcs.TrySetResult(false);
00049           };
00050
00051           webSocket.OnMessage += (bytes) =>
00052           {
00053               var message = System.Text.Encoding.UTF8.GetString(bytes);
00054               ProcessWebSocketMessage(message);
00055           };
00056
00057           webSocket.OnError += (e) =>
00058           {
00059               Debug.LogError($"WebSocket error: {e}");
00060               tcs.TrySetException(new Exception($"WebSocket error: {e}"));
00061           };
00062
00063           try
00064           {
00065               await webSocket.Connect();
00066               await tcs.Task;
00067           }
00068           catch (Exception ex)
00069           {
00070               Debug.LogError($"Failed to initialize WebSocket: {ex.Message}");
00071               throw;
00072           }
00073
00074           isInitialized = true;
00075       }
00076
00077       public async Task ConnectAsync()
00078       {
00079           if (!isInitialized)
00080           {
00081               throw new InvalidOperationException("Service must be initialized before connecting");
00082           }
00083
00084           if (webSocket.State == WebSocketState.Open)
00085           {
00086               return;
00087           }
00088
00089           await webSocket.Connect();
00090       }
00091
00092       public async Task DisconnectAsync()
00093       {
00094           if (webSocket != null && webSocket.State == WebSocketState.Open)
00095           {
00096               await webSocket.Close();
00097           }
00098       }
00099
00100       public async Task SendAudioDataAsync(byte[] audioData)
00101       {
00102           if (!IsConnected)
00103           {
00104               Debug.LogWarning("Cannot send audio data: WebSocket is not connected");
00105               return;
00106           }
00107
00108           await webSocket.Send(audioData);
00109       }
00110
00111       private async void TryReconnect()
00112       {
00113           if (reconnectAttempts >= config.MaxReconnectAttempts)
00114           {
00115               Debug.LogError("Max reconnection attempts reached");
00116               return;
00117           }
00118
00119           reconnectAttempts++;
```

```
00120             Debug.Log($"Attempting to reconnect ({reconnectAttempts}/{config.MaxReconnectAttempts})...");
00121
00122             await Task.Delay(config.ReconnectDelayMs);
00123             await ConnectAsync();
00124         }
00125
00126         private void ProcessWebSocketMessage(string message)
00127         {
00128             try
00129             {
00130                 var commandData = JsonConvert.DeserializeObject<VoiceCommandData>(message);
00131                 if (commandData != null)
00132                 {
00133                     OnCommandRecognized?.Invoke(commandData);
00134                 }
00135             }
00136             catch (Exception e)
00137             {
00138                 Debug.LogError($"Error processing WebSocket message: {e.Message}");
00139             }
00140         }
00141
00142         private void Update()
00143         {
00144             if (webSocket != null)
00145             {
00146                 #if !UNITY_WEBGL || UNITY_EDITOR
00147                 webSocket.DispatchMessageQueue();
00148                 #endif
00149             }
00150         }
00151
00152         private void OnDestroy()
00153         {
00154             DisconnectAsync().GetAwaiter().GetResult();
00155         }
00156     }
00157 }
```

7.16 Файл Assets/Scripts/VoiceControl/Test/MockVoiceControlTest.cs

Классы

- class CurseVR.VoiceControl.Test.MockVoiceControlTest

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.Test

7.17 MockVoiceControlTest.cs

См. документацию.

```
00001 using UnityEngine;
00002 using UnityEngine.InputSystem;
00003 using System.Collections;
00004 using CurseVR.VoiceControl.Core;
00005 using CurseVR.VoiceControl.Models;
00006 using CurseVR.VoiceControl.Services;
00007
00008 namespace CurseVR.VoiceControl.Test
00009 {
00010     public class MockVoiceControlTest : MonoBehaviour
00011     {
00012         [Header("Test Settings")]
00013         [SerializeField] private AudioClip testVoiceClip;
00014         [SerializeField] private Material highlightMaterial;
00015         [SerializeField] private Material defaultMaterial;
```

```
00016       [SerializeField] private float moveDistance = 5f;
00017       [SerializeField] private InputActionReference interactAction;
00018       [SerializeField] private Camera mainCamera;
00019       [SerializeField] private LayerMask interactableLayer = -1;
00020
00021       [Header("Audio Settings")]
00022       [SerializeField] private AudioSource voiceAudioSource;
00023       [SerializeField] private bool playAudioOnSelect = true;
00024       [SerializeField] private float audioVolume = 1f;
00025       [SerializeField] private bool debugAudio = true;
00026
00027       [Header("Debug Settings")]
00028       [SerializeField] private bool offlineMode = false;
00029       [SerializeField] private float offlineModeDelay = 1f;
00030       [SerializeField] private bool debugInput = true;
00031
00032       private GameObject selectedObject;
00033       private Material originalMaterial;
00034       private IVoiceCommandService voiceService;
00035       private bool isProcessing;
00036       private bool isQuitting;
00037       private Key interactKey = Key.E;
00038
00039       private void Start()
00040       {
00041           if (mainCamera == null)
00042           {
00043               mainCamera = Camera.main;
00044               if (mainCamera == null)
00045               {
00046                   Debug.LogError("No main camera found! Please assign a camera in the inspector.");
00047                   enabled = false;
00048                   return;
00049               }
00050           }
00051
00052           if (testVoiceClip == null)
00053           {
00054               Debug.LogError("Test voice clip is not assigned! Please assign an audio clip in the inspector.");
00055           }
00056
00057           if (voiceAudioSource == null)
00058           {
00059               voiceAudioSource = GetComponent<AudioSource>();
00060               if (voiceAudioSource == null)
00061               {
00062                   Debug.LogError("No AudioSource component found! Please add an AudioSource component to this
      GameObject or assign one in the inspector.");
00063                   return;
00064               }
00065           }
00066
00067           voiceAudioSource.clip = testVoiceClip;
00068           voiceAudioSource.playOnAwake = false;
00069           voiceAudioSource.volume = audioVolume;
00070
00071           if (debugAudio)
00072           {
00073               Debug.Log($"Audio source configured: Volume={voiceAudioSource.volume}, Clip={testVoiceClip?.name ??
      "None"}");
00074           }
00075
00076           if (!offlineMode)
00077           {
00078               InitializeVoiceService();
00079           }
00080           else
00081           {
00082               Debug.Log("Running in offline mode - voice service disabled");
00083           }
00084
00085           if (interactAction != null && interactAction.action != null)
00086           {
00087               interactAction.action.Enable();
00088               Debug.Log($"Input action enabled: {interactAction.action.name}");
00089           }
00090           else
00091           {
00092               Debug.LogError("Interact action reference is not assigned!");
00093           }
00094       }
00095
00096       private void InitializeVoiceService()
00097       {
00098           voiceService = gameObject.AddComponent<VoiceCommandService>();
00099           var config = new VoiceServiceConfig
00100           {
```

```
00101             WebSocketUrl = "ws://localhost:8000/ws/asr/",
00102             ClientId = "unity_test_client",
00103             SampleRate = 16000,
00104             Channels = 1
00105         };
00106
00107         voiceService.InitializeAsync(config).ContinueWith(_ =>
00108         {
00109             if (!isQuitting)
00110             {
00111                 voiceService.ConnectAsync();
00112             }
00113         });
00114
00115         voiceService.OnCommandRecognized += HandleVoiceCommand;
00116     }
00117
00118     private void Update()
00119     {
00120         // Check for input using both methods for reliability
00121         bool shouldInteract = (Keyboard.current != null && Keyboard.current[interactKey].wasPressedThisFrame) ||
00122                               (interactAction != null && interactAction.action != null && interactAction.action.triggered);
00123
00124         if (shouldInteract)
00125         {
00126             if (debugInput) Debug.Log("Interact key was pressed this frame");
00127             HandleInteraction();
00128         }
00129     }
00130
00131     private void HandleInteraction()
00132     {
00133         if (isProcessing)
00134         {
00135             if (debugInput) Debug.Log("Interaction skipped - still processing previous command");
00136             return;
00137         }
00138
00139         if (mainCamera == null)
00140         {
00141             Debug.LogError("No camera assigned!");
00142             return;
00143         }
00144
00145         Vector2 mousePosition = Mouse.current.position.ReadValue();
00146         Ray ray = mainCamera.ScreenPointToRay(mousePosition);
00147
00148         if (debugInput)
00149         {
00150             Debug.Log($"Casting ray from: {ray.origin} in direction: {ray.direction}");
00151             Debug.DrawRay(ray.origin, ray.direction * 100f, Color.red, 1f);
00152         }
00153
00154         RaycastHit hit;
00155         bool didHit = Physics.Raycast(ray, out hit, 100f, interactableLayer);
00156
00157         if (debugInput) Debug.Log($"Raycast hit something: {didHit}" + (didHit ? $", Object:
    {hit.collider.gameObject.name}, Distance: {hit.distance}" : ""));
00158
00159         if (didHit)
00160         {
00161             SelectObject(hit.collider.gameObject);
00162         }
00163         else
00164         {
00165             DeselectObject();
00166         }
00167     }
00168
00169     private void SelectObject(GameObject obj)
00170     {
00171         if (debugAudio) Debug.Log($"Selecting object: {obj.name}");
00172
00173         // Deselect previous object if any
00174         DeselectObject();
00175
00176         // Select new object
00177         selectedObject = obj;
00178         var renderer = selectedObject.GetComponent<MeshRenderer>();
00179         if (renderer != null)
00180         {
00181             originalMaterial = renderer.material;
00182             renderer.material = highlightMaterial;
00183             Debug.Log("Applied highlight material");
00184         }
00185         else
00186         {
```

```
00187               Debug.LogWarning("Selected object has no MeshRenderer component!");
00188           }
00189
00190           // Start mock voice command process
00191           StartCoroutine(ProcessMockVoiceCommand());
00192       }
00193
00194       private void DeselectObject()
00195       {
00196           if (selectedObject != null)
00197           {
00198               Debug.Log($"Deselecting object: {selectedObject.name}");
00199               var renderer = selectedObject.GetComponent<MeshRenderer>();
00200               if (renderer != null)
00201               {
00202                   renderer.material = originalMaterial;
00203               }
00204               selectedObject = null;
00205           }
00206       }
00207
00208       private IEnumerator ProcessMockVoiceCommand()
00209       {
00210           if (testVoiceClip == null && !offlineMode)
00211           {
00212               Debug.LogError("Test voice clip is not assigned!");
00213               yield break;
00214           }
00215
00216           isProcessing = true;
00217           Debug.Log("Starting mock voice command process");
00218
00219           if (offlineMode && playAudioOnSelect)
00220           {
00221               if (voiceAudioSource != null && testVoiceClip != null)
00222               {
00223                   if (debugAudio) Debug.Log($"Starting audio playback: {testVoiceClip.name}, length:
      {testVoiceClip.length}s");
00224
00225                   voiceAudioSource.time = 0;
00226
00227                   voiceAudioSource.Play();
00228
00229                   if (debugAudio)
00230                   {
00231                       Debug.Log($"Audio state: isPlaying={voiceAudioSource.isPlaying}, time={voiceAudioSource.time},
      volume={voiceAudioSource.volume}");
00232                       StartCoroutine(MonitorAudioPlayback());
00233                   }
00234
00235                   while (voiceAudioSource.isPlaying)
00236                   {
00237                       yield return null;
00238                   }
00239
00240                   if (debugAudio) Debug.Log("Audio playback completed");
00241               }
00242               else
00243               {
00244                   Debug.LogError($"Cannot play audio: AudioSource={voiceAudioSource != null}, Clip={testVoiceClip !=
      null}");
00245                   yield return new WaitForSeconds(1f);
00246               }
00247
00248               if (testVoiceClip != null)
00249               {
00250                   // Convert AudioClip to byte array and send to service
00251                   float[] samples = new float[testVoiceClip.samples * testVoiceClip.channels];
00252                   testVoiceClip.GetData(samples, 0);
00253
00254                   byte[] audioData = new byte[samples.Length * 2];
00255                   for (int i = 0; i < samples.Length; i++)
00256                   {
00257                       short value = (short)(samples[i] * 32767f);
00258                       audioData[i * 2] = (byte)(value & 0xff);
00259                       audioData[i * 2 + 1] = (byte)((value >> 8) & 0xff);
00260                   }
00261
00262                   if (debugAudio) Debug.Log($"Sending audio data, size: {audioData.Length} bytes");
00263                   yield return voiceService.SendAudioDataAsync(audioData);
00264               }
00265           }
00266           else if (offlineMode)
00267           {
00268               Debug.Log($"Offline mode: simulating voice command processing for {offlineModeDelay} seconds");
00269               yield return new WaitForSeconds(offlineModeDelay);
00270
```

```
00271             var mockCommand = new VoiceCommandData
00272             {
00273                 CommandType = "move_right",
00274                 RawText = "передвинуть этот предмет на 5 метров вправо",
00275                 Confidence = 1.0f
00276             };
00277 
00278             HandleVoiceCommand(mockCommand);
00279         }
00280 
00281         isProcessing = false;
00282         Debug.Log("Mock voice command process completed");
00283     }
00284 
00285     private IEnumerator MonitorAudioPlayback()
00286     {
00287         float startTime = Time.time;
00288         while (voiceAudioSource != null && voiceAudioSource.isPlaying)
00289         {
00290             Debug.Log($"Audio playing: time={voiceAudioSource.time:F2}/{testVoiceClip.length:F2},
    volume={voiceAudioSource.volume}");
00291             yield return new WaitForSeconds(0.1f);
00292 
00293             // Timeout after twice the clip length
00294             if (Time.time - startTime > testVoiceClip.length * 2)
00295             {
00296                 Debug.LogWarning("Audio playback timeout!");
00297                 break;
00298             }
00299         }
00300     }
00301 
00302     private void HandleVoiceCommand(VoiceCommandData commandData)
00303     {
00304         if (selectedObject == null) return;
00305 
00306         Debug.Log($"Received command: {commandData.CommandType}, Raw text: {commandData.RawText}");
00307 
00308         if (commandData.RawText.ToLower().Contains("передвинуть") &&
00309             commandData.RawText.ToLower().Contains("вправо"))
00310         {
00311             StartCoroutine(MoveObjectRight());
00312         }
00313     }
00314 
00315     private IEnumerator MoveObjectRight()
00316     {
00317         if (selectedObject == null) yield break;
00318 
00319         Debug.Log($"Moving object {selectedObject.name} right by {moveDistance} units");
00320 
00321         Vector3 startPos = selectedObject.transform.position;
00322         Vector3 endPos = startPos + Vector3.right * moveDistance;
00323         float duration = 1f;
00324         float elapsed = 0f;
00325 
00326         while (elapsed < duration)
00327         {
00328             elapsed += Time.deltaTime;
00329             float t = elapsed / duration;
00330             selectedObject.transform.position = Vector3.Lerp(startPos, endPos, t);
00331             yield return null;
00332         }
00333 
00334         selectedObject.transform.position = endPos;
00335         Debug.Log("Movement completed");
00336         DeselectObject();
00337     }
00338 
00339     private void OnApplicationQuit()
00340     {
00341         isQuitting = true;
00342         CleanupVoiceService();
00343     }
00344 
00345     private void OnEnable()
00346     {
00347         if (interactAction != null && interactAction.action != null)
00348         {
00349             interactAction.action.Enable();
00350             if (debugInput) Debug.Log($"Input action enabled in OnEnable: {interactAction.action.name}");
00351         }
00352     }
00353 
00354     private void OnDisable()
00355     {
00356         if (interactAction != null && interactAction.action != null)
```

```
00357           {
00358               interactAction.action.Disable();
00359               if (debugInput) Debug.Log($"Input action disabled in OnDisable: {interactAction.action.name}");
00360           }
00361       }
00362
00363       private void OnDestroy()
00364       {
00365           if (!isQuitting && !offlineMode)
00366           {
00367               CleanupVoiceService();
00368           }
00369       }
00370
00371       private void CleanupVoiceService()
00372       {
00373           Debug.Log("Cleaning up voice service");
00374           if (voiceService != null)
00375           {
00376               try
00377               {
00378                   voiceService.DisconnectAsync().GetAwaiter().GetResult();
00379                   Debug.Log("Voice service disconnected successfully");
00380               }
00381               catch (System.Exception e)
00382               {
00383                   Debug.LogError($"Error disconnecting voice service: {e}");
00384               }
00385           }
00386       }
00387   }
00388 }
```

7.18 Файл Assets/Scripts/VoiceControl/Utils/AudioUtils.cs

Классы

- class CurseVR.VoiceControl.Utils.AudioUtils

  Utility class for audio processing operations.

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.Utils

7.19 AudioUtils.cs

См. документацию.

```
00001 using UnityEngine;
00002
00003 namespace CurseVR.VoiceControl.Utils
00004 {
00008     public static class AudioUtils
00009     {
00013         public static byte[] AudioClipToBytes(AudioClip clip)
00014         {
00015             if (clip == null) return null;
00016
00017             float[] samples = new float[clip.samples * clip.channels];
00018             clip.GetData(samples, 0);
00019
00020             byte[] audioData = new byte[samples.Length * 2];
00021             for (int i = 0; i < samples.Length; i++)
00022             {
00023                 short value = (short)(samples[i] * 32767f);
00024                 audioData[i * 2] = (byte)(value & 0xff);
00025                 audioData[i * 2 + 1] = (byte)((value >> 8) & 0xff);
00026             }
```

```
00027
00028             return audioData;
00029         }
00030
00034         public static AudioClip BytesToAudioClip(byte[] audioData, int channels, int frequency)
00035         {
00036             if (audioData == null) return null;
00037
00038             float[] samples = new float[audioData.Length / 2];
00039             for (int i = 0; i < samples.Length; i++)
00040             {
00041                 short value = (short)((audioData[i * 2 + 1] « 8) | audioData[i * 2]);
00042                 samples[i] = value / 32768f;
00043             }
00044
00045             AudioClip clip = AudioClip.Create("ConvertedClip", samples.Length / channels, channels, frequency, false);
00046             clip.SetData(samples, 0);
00047
00048             return clip;
00049         }
00050
00054         public static float CalculateRMSVolume(float[] samples)
00055         {
00056             float sum = 0f;
00057             for (int i = 0; i < samples.Length; i++)
00058             {
00059                 sum += samples[i] * samples[i];
00060             }
00061
00062             return Mathf.Sqrt(sum / samples.Length);
00063         }
00064
00068         public static void ApplyNoiseGate(float[] samples, float threshold)
00069         {
00070             for (int i = 0; i < samples.Length; i++)
00071             {
00072                 if (Mathf.Abs(samples[i]) < threshold)
00073                 {
00074                     samples[i] = 0f;
00075                 }
00076             }
00077         }
00078     }
00079 }
```

7.20 Файл Assets/Scripts/VoiceControl/VAD/AudioClipBuffer.cs

Классы

- class CurseVR.VoiceControl.VAD.AudioClipBuffer

  Buffers audio samples and creates AudioClips when the buffer is filled or flushed.

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.VAD

7.21 AudioClipBuffer.cs

См. документацию.

```
00001 using System;
00002 using UnityEngine;
00003
00004 namespace CurseVR.VoiceControl.VAD
00005 {
00015     public class AudioClipBuffer
00016     {
00017         private readonly float[] buffer;
```

```
00018         private readonly int channels;
00019         private readonly int frequency;
00020         private int writePosition;
00021         private bool isFull;
00022 
00030         public event Action<AudioClip> OnBufferFilled;
00031 
00042         public AudioClipBuffer(int maxSampleLength, int frequency, int channels = 1)
00043         {
00044             this.buffer = new float[maxSampleLength * channels];
00045             this.frequency = frequency;
00046             this.channels = channels;
00047             this.writePosition = 0;
00048             this.isFull = false;
00049         }
00050 
00061         public void AddSamples(float[] samples)
00062         {
00063             if (samples == null || samples.Length == 0) return;
00064 
00065             int samplesToWrite = Math.Min(samples.Length, buffer.Length - writePosition);
00066             Array.Copy(samples, 0, buffer, writePosition, samplesToWrite);
00067 
00068             writePosition += samplesToWrite;
00069 
00070             if (writePosition >= buffer.Length)
00071             {
00072                 isFull = true;
00073                 CreateAndEmitAudioClip();
00074                 Reset();
00075             }
00076         }
00077 
00086         public void Flush()
00087         {
00088             if (writePosition > 0)
00089             {
00090                 CreateAndEmitAudioClip();
00091                 Reset();
00092             }
00093         }
00094 
00102         private void CreateAndEmitAudioClip()
00103         {
00104             int sampleCount = isFull ? buffer.Length : writePosition;
00105             if (sampleCount == 0) return;
00106 
00107             var clip = AudioClip.Create(
00108                 "VoiceBuffer",
00109                 sampleCount / channels,
00110                 channels,
00111                 frequency,
00112                 false);
00113 
00114             float[] data = new float[sampleCount];
00115             Array.Copy(buffer, data, sampleCount);
00116             clip.SetData(data, 0);
00117 
00118             OnBufferFilled?.Invoke(clip);
00119         }
00120 
00128         private void Reset()
00129         {
00130             writePosition = 0;
00131             isFull = false;
00132             Array.Clear(buffer, 0, buffer.Length);
00133         }
00134     }
00135 }
```

7.22 Файл Assets/Scripts/VoiceControl/VAD/IVoiceActivityDetector.cs

Классы

- interface CurseVR.VoiceControl.VAD.IVoiceActivityDetector

  Interface for voice activity detection functionality.

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.VAD

7.23 IVoiceActivityDetector.cs

См. документацию.

```
00001 using System;
00002 using UnityEngine;
00003
00004 namespace CurseVR.VoiceControl.VAD
00005 {
00014     public interface IVoiceActivityDetector : IDisposable
00015     {
00020         bool IsActive { get; }
00021
00030         event Action<bool> OnVoiceActivityChanged;
00031
00041         void Update();
00042     }
00043 }
```

7.24 Файл Assets/Scripts/VoiceControl/VAD/UnityMicrophoneProxy.cs

Классы

- class CurseVR.VoiceControl.VAD.UnityMicrophoneProxy

  Provides an abstraction over Unity's Microphone API for simplified access to audio input.

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.VAD

7.25 UnityMicrophoneProxy.cs

См. документацию.

```
00001 using System;
00002 using UnityEngine;
00003
00004 namespace CurseVR.VoiceControl.VAD
00005 {
00014     public class UnityMicrophoneProxy : IDisposable
00015     {
00016         private readonly string deviceName;
00017         private AudioClip audioClip;
00018         private readonly int frequency;
00019         private readonly int sampleRate;
00020
00029         public AudioClip AudioClip => audioClip;
00030
00035         public int SampleRate => sampleRate;
00036
00048         public UnityMicrophoneProxy(string deviceName = null, int frequency = 44100)
00049         {
00050             if (Microphone.devices.Length == 0)
00051             {
```

```
00052             throw new InvalidOperationException("No microphone devices available");
00053         }
00054
00055         this.deviceName = deviceName ?? Microphone.devices[0];
00056         this.frequency = frequency;
00057         this.sampleRate = frequency;
00058
00059         InitializeMicrophone();
00060     }
00061
00070     private void InitializeMicrophone()
00071     {
00072         if (Microphone.IsRecording(deviceName))
00073         {
00074             Microphone.End(deviceName);
00075         }
00076
00077         audioClip = Microphone.Start(deviceName, true, 1, frequency);
00078
00079         while (!(Microphone.GetPosition(deviceName) > 0)) { }
00080     }
00081
00090     public void Dispose()
00091     {
00092         if (Microphone.IsRecording(deviceName))
00093         {
00094             Microphone.End(deviceName);
00095         }
00096
00097         if (audioClip != null)
00098         {
00099             UnityEngine.Object.Destroy(audioClip);
00100             audioClip = null;
00101         }
00102     }
00103 }
00104 }
```

7.26 Файл Assets/Scripts/VoiceControl/VAD/VoiceActivityDetector.cs

Классы

- class CurseVR.VoiceControl.VAD.VoiceActivityDetector

  Implements voice activity detection by analyzing microphone input volume.

Пространства имен

- namespace CurseVR
- namespace CurseVR.VoiceControl
- namespace CurseVR.VoiceControl.VAD

7.27 VoiceActivityDetector.cs

См. документацию.

```
00001 using System;
00002 using UnityEngine;
00003 using CurseVR.VoiceControl.Models;
00004
00005 namespace CurseVR.VoiceControl.VAD
00006 {
00016     public class VoiceActivityDetector : IVoiceActivityDetector
00017     {
00018         private readonly UnityMicrophoneProxy micProxy;
00019         private readonly AudioClipBuffer audioBuffer;
00020         private readonly VADParameters parameters;
00021         private readonly float[] sampleBuffer;
00022
00023         private float lastActiveTime;
00024         private float lastInactiveTime;
```

```
00025       private bool isActive;
00026       private int lastReadPosition;
00027
00029       public bool IsActive => isActive;
00030
00032       public event Action<bool> OnVoiceActivityChanged;
00033
00045       public VoiceActivityDetector(UnityMicrophoneProxy micProxy, AudioClipBuffer audioBuffer, VADParameters
    parameters)
00046       {
00047           this.micProxy = micProxy ?? throw new ArgumentNullException(nameof(micProxy));
00048           this.audioBuffer = audioBuffer ?? throw new ArgumentNullException(nameof(audioBuffer));
00049           this.parameters = parameters ?? throw new ArgumentNullException(nameof(parameters));
00050
00051           this.sampleBuffer = new float[parameters.BufferSize];
00052           this.lastActiveTime = -parameters.ActivationIntervalSeconds;
00053           this.lastInactiveTime = -parameters.InactivationIntervalSeconds;
00054       }
00055
00057       public void Update()
00058       {
00059           if (micProxy.AudioClip == null) return;
00060
00061           int currentPosition = Microphone.GetPosition(null);
00062           if (currentPosition < 0) return;
00063
00064           if (currentPosition == lastReadPosition) return;
00065
00066           int samplesToRead = GetSamplesToRead(currentPosition);
00067           if (samplesToRead <= 0) return;
00068
00069           micProxy.AudioClip.GetData(sampleBuffer, 0);
00070           ProcessAudioData(sampleBuffer, samplesToRead);
00071
00072           lastReadPosition = currentPosition;
00073       }
00074
00083       private int GetSamplesToRead(int currentPosition)
00084       {
00085           if (currentPosition < lastReadPosition)
00086           {
00087               return (micProxy.AudioClip.samples - lastReadPosition) + currentPosition;
00088           }
00089
00090           return currentPosition - lastReadPosition;
00091       }
00092
00103       private void ProcessAudioData(float[] samples, int sampleCount)
00104       {
00105           float volume = CalculateVolume(samples, sampleCount);
00106           float time = Time.time;
00107
00108           if (!isActive && volume > parameters.ActiveVolumeThreshold)
00109           {
00110               if (time - lastActiveTime >= parameters.ActivationIntervalSeconds)
00111               {
00112                   isActive = true;
00113                   lastActiveTime = time;
00114                   OnVoiceActivityChanged?.Invoke(true);
00115               }
00116           }
00117           else if (isActive && volume < parameters.ActiveVolumeThreshold)
00118           {
00119               if (time - lastInactiveTime >= parameters.InactivationIntervalSeconds)
00120               {
00121                   isActive = false;
00122                   lastInactiveTime = time;
00123                   OnVoiceActivityChanged?.Invoke(false);
00124                   audioBuffer.Flush();
00125               }
00126           }
00127
00128           if (isActive)
00129           {
00130               float[] activeData = new float[sampleCount];
00131               Array.Copy(samples, activeData, sampleCount);
00132               audioBuffer.AddSamples(activeData);
00133           }
00134       }
00135
00142       private float CalculateVolume(float[] samples, int sampleCount)
00143       {
00144           float sum = 0f;
00145           for (int i = 0; i < sampleCount; i++)
00146           {
00147               sum += samples[i] * samples[i];
00148           }
```

```
00149
00150             return Mathf.Sqrt(sum / sampleCount);
00151         }
00152
00154         public void Dispose()
00155         {
00156             micProxy?.Dispose();
00157         }
00158     }
00159 }
```

Предметный указатель