

Optispeech 2

This is the documentation for the Optispeech 2 program created by the Speech Production Lab at UT Dallas. You can check out the manual for resources on using or extending this program or the Scripting API for information on each script file in this project. Links to both sections are in the header above.

For more information including related research papers, see the website for the [Speech Production Lab at UT Dallas](#).

Resources

This program was written with the help of the following resources:

- [Carstens EMA manual](#)
- [NDI Wave manual rev 7](#)

Note that this rev 7 and only documentation. At the speech production lab we're currently using software that came with rev 4 in the along with a demo application and source code, which can be found in the ndigital folder in Program Files.

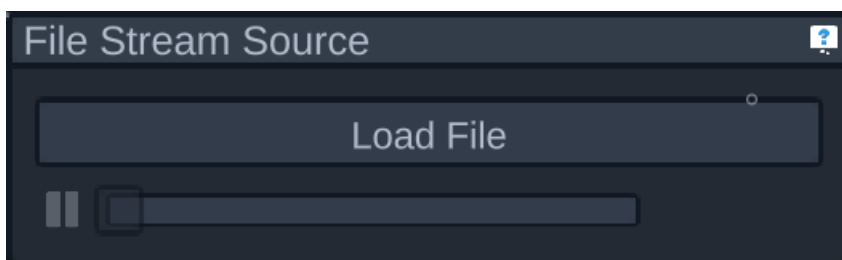
Getting Started

This guide is intended to bring a researcher with no experience using OptiSpeech 2 to a point where they can use the software and most of its features comfortably. For more detailed explanations on specific parts of the tool, please use the links in the sidebar.

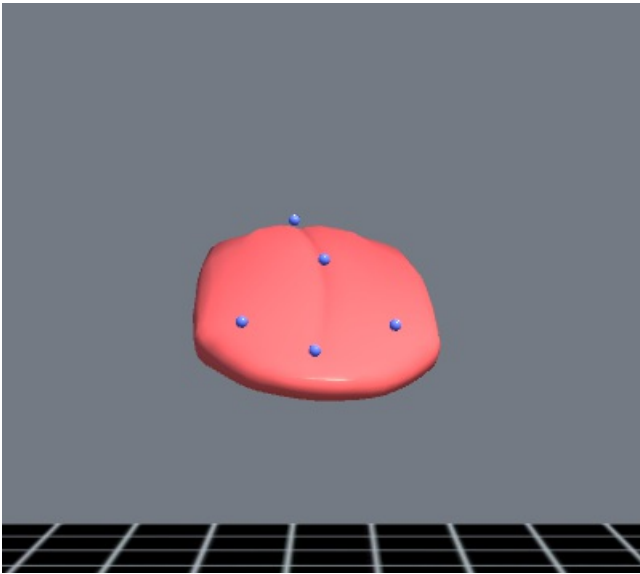
When you first open up OptiSpeech, you will see a large message that there is no data source selected. On the left you'll see a number of panels, including an open one with a list of data sources you can choose from. Some may be disabled if the system doesn't currently support that data source, in which case they'll have a red X on the side. If you're playing back a previously recorded sweep, you'll want to choose the "Read from File" data source. Otherwise choose whichever source corresponds to the hardware you plan on using.



If you choose the "Read from File" data source, you'll have to select the file to read from by clicking the "Load File" button in the new set of panels on the right side of the screen.



The tongue model should now be moving to visualize the data being read in from the data source you've selected.



If you need a different camera angle you can hold down the right mouse button and use the WASD keys to move the camera. Moving your mouse while holding down the right mouse button will also change the direction the camera is pointing in.

When you're ready to start recording data, you're going to use the "Sweeps" panel. You can choose a folder to put the data into, such as a flash drive or a folder specific to your research. You can then write a sweep name. When starting a sweep it'll automatically append an incrementing number if there's already a sweep with that name. You can then click Start Sweep to begin recording data, and the button will then turn into a Stop Sweep button that will end the recording. By default it'll record the raw data for playback in OptiSpeech 2 as well as a file with all the sensor data after its been "transformed", which means after its been placed onto the tongue model. This is a .tsv file, and when you're performing your data analysis this is probably the file you want. By default an audio file is also saved which contains anything recorded from the microphone during the sweep. There are checkboxes to enable or disabled which files you'd like to be written.

Sweeps

Folder Path

C:\Users\LawnAnthonyReilley\Docu

Sweep Name

sweep

Stop After (s)

Never

☒ Save Raw

☒ Save Transformed

☐ Save Transformed Without Offsets

☒ Save Synced Audio

Start Sweep

That's enough to get you started! You now know how to collect data using OptiSpeech 2. For more advanced usage you can explore the other panels to find out what they do, or use the sidebar to read the documtation on each of them. Each panel also has a convenient help button that'll open its documentation page in your browser.

Profile Selector



All the settings from where the camera is, to sweep settings, are saved in "profiles". In the profile selector you can add new profiles, delete old ones, and switch between them at will. There are two main reasons for having different profiles:

- Different experiments may require different setups, and this allows you to switch between them as needed
- A single experiment may require testing under different conditions, and storing those conditions as profiles makes it easy to switch between them

To create a new profile, type the name in the text field at the top of the panel and click "Create". This will create a new profile with the default settings.

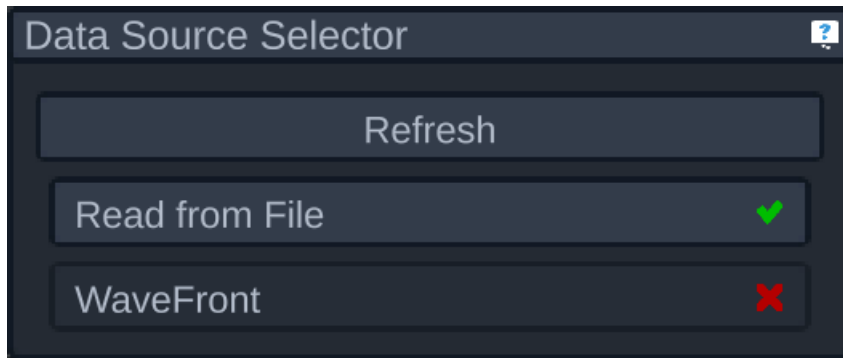
Clicking a profile (but not on one of the three buttons) will select that profile. The currently selected profile will be highlighted in green. Additionally, the name of the panel will include the name of the active profile.

The three buttons have the following purposes:

- The first button toggles rename mode. When in rename mode, you can type the new name of the profile and either press enter or the rename button again (which will now appear as a checkmark) to save the name. To cancel, press escape or click off the textfield.
- The second button will duplicate that profile
- The third button deletes that profile. Careful, there's no way to undo this besides recreating that profile from scratch!

Please be considerate of other researches and don't write over their profiles! You can always duplicate their profile first if you like their settings.

Data Source Selector



The data sources selector panel allows you to select from which source the sensors will pull data from. Each source may be enabled or disabled depending on the hardware currently connected to the machine. In some cases it may take some time to determine if the hardware is available, in which case you'll see the following icon while its working:



Additional Info on Each Data Source

- [File Data Source](#)
- [WaveFront Data Source](#)

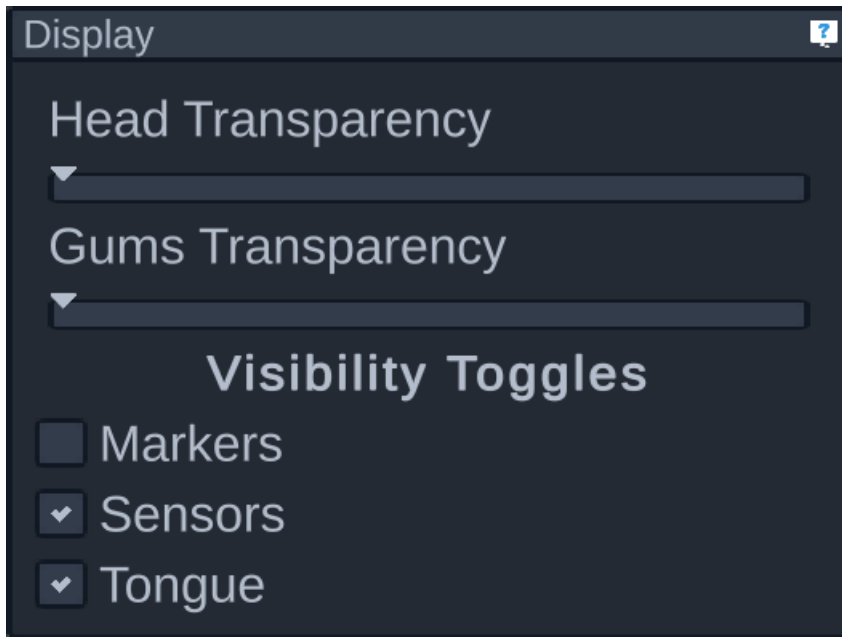
Camera Panel



The camera panel shows a list of recommended camera positions and angles.

If you'd like a custom camera angle, you can hold the right mouse button down and move it to change the direction of the camera. While looking around you can also use the WASD keys on your keyboard to move the camera forward, backward, left, and right; and Q and E to move the camera up and down. Hold down shift while moving to move faster.

Display panel



This panel allows you to configure how the tongue and avatar appear.

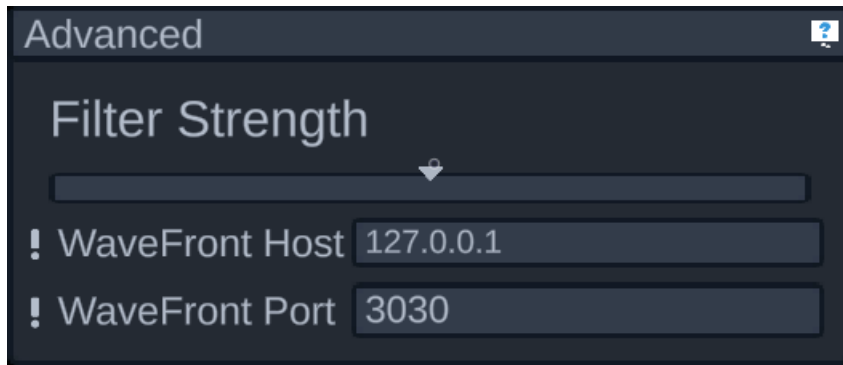
The head transparency slider affects how visible versus see-through the avatar appears. All the way to the right means completely visible.

The gums transparency slider works the same way but for the avatar's gums.

Underneath the transparency sliders are visibility toggles. These allow you to choose whether or not certain markers appear:

- Markers refer to the reference sensors like the forehead, ears, and jaw
- Sensors refer to all the sensors on the tongue
- Toggling the tongue visibility affects the appearance of the tongue model itself

Advanced panel



This panel includes advanced settings that don't belong in other panels.

The filter strength affects how strong of a low pass filter to apply to data as its rendered on the model, where the slider being all the way to the left indicates no filter should be applied. A low pass filter prevents spikes in data from having as much of an affect, effectively "smoothing" out the data. This isn't ideal for data analysis, however, so the data written to file will NOT be affected by this setting. This only affects how it appears on the model.

The next two values are for setting up the WaveFront data source. The default values are written according to specification and these will probably not needed to be changed. If they do, however, it'll be because of the hardware/network setup on the machine the program is running on. Because of that, these properties are stored independently from profiles. Because of that, they have a warning that these will affect all profiles.

Sensors List

Sensors List

?

Reset Resting Position

Sensor 0

Forehead

Offsets: X 0 Y 0 Z 0

Sensor 1

Left Ear

Offsets: X 0 Y 0 Z 0

Sensor 2

Right Ear

Offsets: X 0 Y 0 Z 0

Sensor 3

Tongue Tip

Offsets: X 0 Y 0 Z 0

Sensor 4

Tongue Dorsum

Offsets: X 0 Y 0 Z 0

Sensor 5

Tongue Right

Offsets: X 0 Y 0 Z 0

Sensor 6

Tongue Left

Offsets: X 0 Y 0 Z 0

Sensor 7

Tongue Back

Offsets: X 0 Y 0 Z 0

Sensor 8

Jaw

Offsets: X 0 Y 0 Z 0

Sensor 9

Other

Offsets: X 0 Y 0 Z 0

When a data source is active, this panel shows a list of all the current sensors and their statuses. The sensors status is shown via a colored background behind the sensor ID, and comes from the data source so the user cannot add or remove sensors or change their status.

Each sensor can one of the following statuses:

- OK - green, everything's okay
- BAD FIT - red, a status WaveFront may send under certain conditions
- OUT OF RANGE - yellow, the sensor is outside of the detectable area
- PROCESSING ERROR - blue, something went wrong processing the data for this sensor
- UNKNOWN - black, when the status cannot be determined

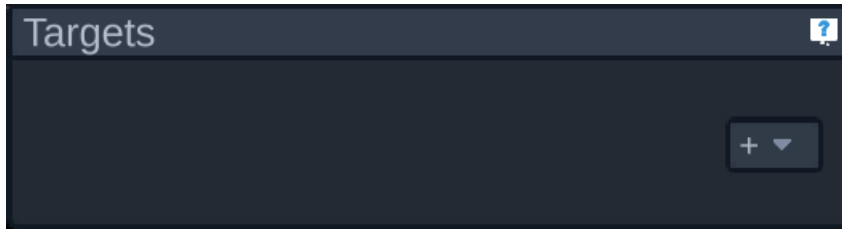
In addition to the sensor status, the user can change each sensor's settings. This includes which position each sensor is in, by using the dropdown. You can also add an offset to each non-reference sensor, which will move where that sensor appears on the avatar by a set distance in each of the three axes. You can drag the label for each axis to change the value and see how it affects the sensor on the model. This is useful for situations where you couldn't place the sensor exactly where it needed to be, for example if a gag reflex prevented placing a sensor as far back in the mouth as it needed to be.

Finally, above the sensors is a button to reset the resting position. The jaw uses a resting position to determine when its open or closed. If the jaw or tongue ever appears to be out of alignment, ask the patient to rest their face - that is, have it closed and not be consciously using any muscles. Then clicking this button will make the jaw appear closed at this position. The initial resting position is wherever the sensors were when the current data source was selected.

NOTE

Some data sources prevent affecting the sensors' settings, such as the file data source, in which case everything in the sensors list is "read-only", and can't be edited.

Targets Panel

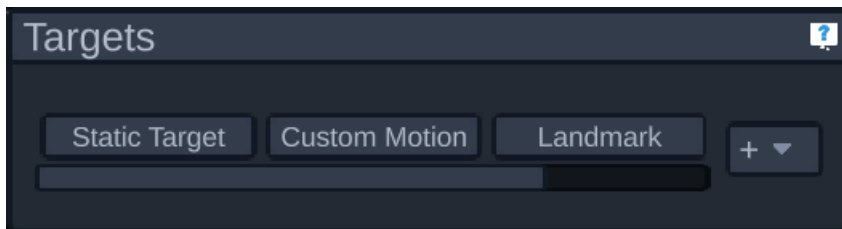


The targets panel shows the current targets and allows them to be added, removed, and configured. Initially it'll appear empty as shown above, and they can be added using the dropdown.

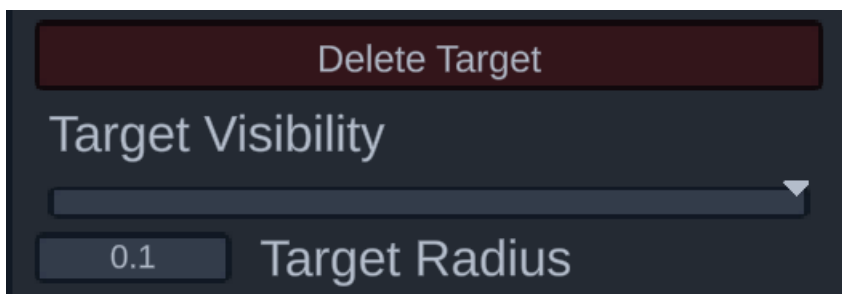
The dropdown allows you to specify the type of target to add:

- Custom Motion: Follows the position of a sensor from an old sweep
- Landmark: Follows the position from one of the current sensors
- Oscillating Target: Moves back and forth between two points
- Static Target: Stays in one place

Added targets will appear to the left of the add target button, with a scrollbar if necessary, like so:



Clicking on a target will allow you to delete or configure that target. The fields shown will vary depending on the target type. Some fields, however, will appear on all target types:



- The delete target button will remove this target
- The target visibility slider affects the transparency of the target, where all the way to the right is completely opaque
- The target radius affects the size of the target marker, and how accuracy is calculated (see the [accuracy panel](#) for details)

Each type of target then has additional settings:

Custom Motion:

Targets 

Custom Motion  

Custom Motion

Delete Target

Target Visibility 

0.1 Target Radius

Load Motion Path From Sweep 

 Sensor ID

1 Playback Speed

☒ Alternate Direction

Offset:

X Y Z

- The custom motion config shows a button to load a previously saved sweep to get motion path data from. When waiting to select a sweep it will show a magnifying glass. If a file is chosen but it fails to find any sweep data, that'll change to a red cross. If it succeeds, a green checkmark will appear.
- Once a motion path is successfully loaded, the Sensor ID dropdown can be used to select which sensor from the motion path the target should follow.
- Playback speed will affect the rate at which the sweep's data will play.
- Alternate direction will make the motion path play back in reverse after playing back forward, and flip-flopping. This prevents the data from "jumping" when the sweep ends and resets.
- Offset will change the position of the sweep data by the specified three axis values.

Landmark:

Targets

Landmark

+

▼

Landmark

Delete Target

Target Visibility

0.1

Target Radius

4 - Tongue Tip

Sensor ID

- The sensor ID is the ID of one of the active sensors that this target should follow

Oscillating Target:

Targets

Oscillating Target

+

▼

Oscillating Target

Delete Target

Target Visibility

0.1

Target Radius

Start Position:

X

0

Y

0.5

Z

-1.2

End Position:

X

0

Y

0.7

Z

-1

1

Frequency

- The start and end positions are 3D points that the target will move between, back and forth. Dragging each X, Y, and Z label will allow you to change the number as if it were a slider.
- Frequency is how many times the target will make a full "cycle" per second, where one cycle is the travel from the start position, to the end position, and back to the start position.

Static Target:

Targets

Static Target

+ ▼

Static Target

Delete Target

Target Visibility

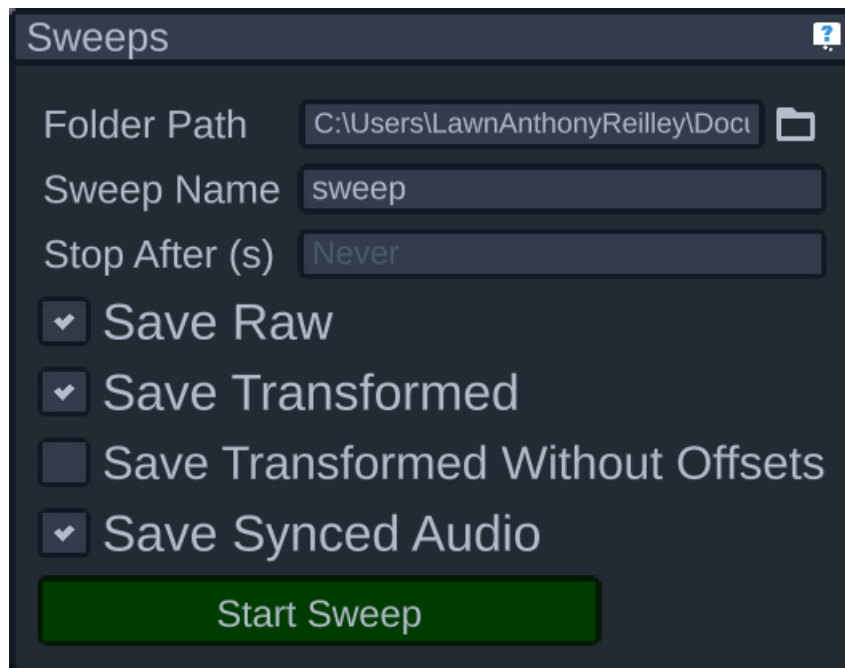
0.1 Target Radius

Position:

X0Y0.5Z-0.8

- The position is a 3D point where this target should be

Sweeps Panel

The image shows a software window titled "Sweeps" with a dark theme. It contains several input fields and checkboxes. The "Folder Path" field is set to "C:\Users\LawnAnthonyReilley\Docu" with a folder icon to its right. The "Sweep Name" field contains the text "sweep". The "Stop After (s)" field is set to "Never". Below these are four checkboxes: "Save Raw" (checked), "Save Transformed" (checked), "Save Transformed Without Offsets" (unchecked), and "Save Synced Audio" (checked). At the bottom is a large green button labeled "Start Sweep".

Sweeps

Folder Path

Sweep Name

Stop After (s)

☒ Save Raw

☒ Save Transformed

☐ Save Transformed Without Offsets

☒ Save Synced Audio

The sweeps panel is where you can configure and then start and stop sweeps. The first two fields allow you to specify where to save your sweep data, and what to call it. If there's already a sweep with that same name, it'll automatically increment a number to make a unique sweep name (e.g. sweep 2, then sweep 3, etc.)

The stop after field allows you to configure a time to automatically stop the sweep. If left blank the sweep will not stop until you manually tell it to. If you have given this a value, you can still manually stop the sweep before the duration has passed.

The 4 toggles allow you to configure what data gets saved in each sweep:

- Raw will save a tab-separated values (tsv) file with all the sensor data completely unprocessed, as well as contain information on what data source was used, the name of the audio file if any, and how each sensor and target was configured. This can be read back by OptiSpeech 2 to have the exact same sensors, targets, and sensor data as when the sweep was recorded, making this the recommended format for future playback
- Transformed will save a tab-separated values (tsv) file with all the sensor data after its been transformed to fit onto the head. This will normalize any head movement and positioning, allowing for data from different sweeps to be more congruent. This format loses any sensor or target information, so its recommended to use this solely for data analysis
- Transformed without offsets will save a file exactly like the one above, but without any configured sensor offsets. In most common cases there won't be any configured sensor offsets, so this format is rarely useful
- Audio will save any audio coming from the default microphone if any. This will also be played back when the raw or transformed data is being played back

By default, all but the transformed without offsets format will be selected, which is the recommended setup for most cases.

Accuracy Panel

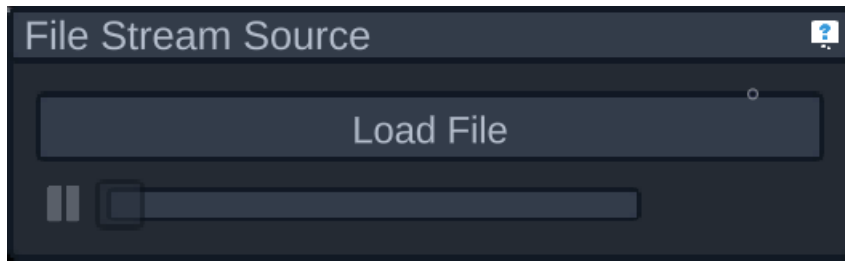
When a data source is active with at least one target, an accuracy panel will appear at the bottom center of the screen and show how accurately the tongue tip's sensor matches the targets.

Accuracy is calculated based on the radius of the targets. If the sensor is completely within the target's radius, it is 100% accurate. 0% accuracy is when the sensor is a second radius away from the target, and anywhere in between has a percent accuracy between 0 and 100. The target marker will change color to represent its individual accuracy at any given moment.

The accuracy percentages are shown over two different type of durations:

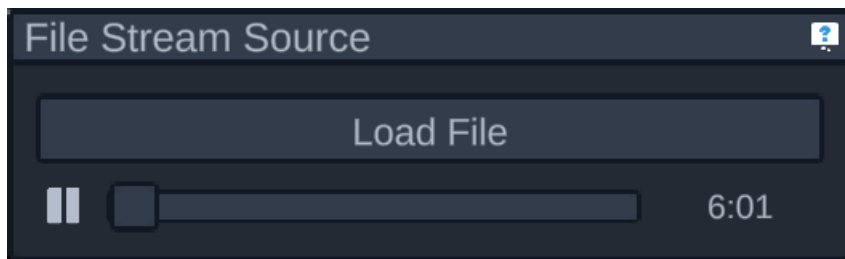
- One "cycle", which is the lowest common multiple of each target's duration. That is, if one takes 2 seconds and another takes 5, then one cycle would be 10 seconds. The duration of a target is how long it takes to get back to its initial position
- One sweep. During a sweep this will be the accuracy since the beginning of the sweep. Once the sweep ends it'll continue displaying the average accuracy of the entire sweep until another is started

File Data Source



This data source loads sweep data from a file and plays it back. The file can be either a WaveFront sweep file or an OptiSpeech sweep file (including the OptiSpeech 1, and including both raw and transformed data from OptiSpeech 2).

To select the sweep data to load, press the "Load File" button in the panel that appears and select a file from the file browser. The file will then be loaded, which may take a little while. Once its complete the UI elements below the Load File button will be interactable. If they aren't, there was an issue loading the file you selected. Here's what the panel looks like when a file is successfully loaded:



Below the Load File button are three UI elements. The first is a button to pause or resume playback of the loaded file. Next is a progress bar showing how far along the file the playback is. You can move the progress bar to go forward and backwards through the file. Finally, when there's a file loaded, the duration of the sweep will appear on the right side of the progress bar.

WaveFront Data Source

This data source reads data from compatible [NDI Wave System](#) hardware through WaveFront's RealTime API (RTAPI). This requires WaveFront to be running.

If the WaveFront data source is not available, its possible the hardware or network configurations on the computer are different from the default. In that case you can change the host and port OptiSpeech 2 will search for WaveFront's RTAPI in the [Advanced Settings Panel](#)

NOTE

Some versions of WaveFront may support different features, like starting sweeps on WaveFront at the same time as they're started in OptiSpeech 2

Development Environment

You can clone this project using the [git](#) version control system. This will also track each person's changes and ease the process of merging changes from different developers. Merging changes in Unity scenes and prefabs is pretty complicated and messy, so its recommended to only edit prefabs for the specific changes being made, whenever possible. When you do need to merge changes in the same scene or prefab, it's recommended to use [Smart Merge](#).

Unity Version

2020.01.0b15 (or potentially higher, but make sure all developers are using the same version at all times)

You'll also probably want to use Visual Studio for editing scripts.

Documentation

To generate documentation, you'll need to [install docFX](#) and make sure its on your system's PATH environment variable ([Windows Guide](#)).

Data Sources

Data Sources allow the program to read data frames from various different source. This document will discuss how to add a new data source to OptiSpeech 2. Typically this'll need to be done to support a new EMA system. If you just want to support loading in a new type of file, you should instead create a new [File Reader](#).

Implement `DataSourceReader`

The core of the new data source is the actual data reader itself, usually saved in `Assets/Scripts/Data/Sources`. It has the following abstract methods that'll need to be implemented:

- `DataSourceReaderStatus GetCurrentStatus()` - This function is needed to determine when the data source is available. This should be where you determine if any hardware or software requirements of this data source are met. You can pass `DataSourceReaderStatus.UNKNOWN` and later invoke `statusChangeEvent` if you need to asynchronously check for requirements.
- `DataFrame ReadFrame()` - This is the function that should return the next frame from the data source. This runs in a separate thread so waiting until the next frame is available is allowed. Several fields in the `DataFrame` are filled in while processing each frame. The only fields required to be set are `sensorData`, and `timestamp` if the data source reader returns `true` from `IsTimestampProvided()`.

Click each link to see more specific information on implementing each function.

Additionally, there are several virtual functions that can be overridden for more advanced control of the data source:

- `bool IsTimestampProvided()` - By default the data source expects `DataFrame`'s coming from `ReadFrame` to contain timestamps, but returning `false` in this function will make the `DataSourceReader` set the timestamp itself
- `bool AreTargetsConfigurable()` - By default targets can be added, modified, and removed by the researcher. Returning `false` in this function will make all the target configs read-only
- `bool AreSensorsConfigurable()` - By default sensors can have their roles and post-offsets changed by the researcher. Returning `false` in this function will make all the sensors read-only
- `SensorConfiguration[] GetDefaultSensorConfigurations()` - Returns the default sensor configurations to have initially
- `void StartThread()` - Function that's called when the data source is selected. Make sure to still call `base.StartThread()`. This can be used to, e.g., tell the data source to start sending data frames
- `void Cleanup()` - Function that's called when the data source is de-selected.
- `void StartSweep(string folderPath, string sweepName)` - Function that's called whenever a sweep starts, which can be used to, e.g., start a sweep on the data source so they're synced
- `void StopSweep()` - Function that's called whenever a sweep ends

If any part is confusing, it's recommended to read through any of the existing data source readers to use as a reference. Notably, there's a utility class `TcpClientController` that is recommended for any data source reader that uses a TCP connection.

Create a `DataSourceDescription`

This is a scriptable object that will tell the program about this new data source. It must be inside a folder called `Data Source Descriptions` inside of any `Resources` folder in `Assets`. To create the `DataSourceDescription`, go to `Optispeech > Data Source` in the create menu. The create menu is accessed through the plus sign in the project panel or by right clicking in a folder and hovering over `Create`. You must then specify the name of the data source and give it a prefab. The only requirement for this prefab is that it include the implementation of `DataSourceReader` made in the previous step. It can contain anything else you need for this data source. This prefab will be instantiated in the Source Settings Accordion, so any `TogglePanel`s in the prefab will automatically be in that accordion.

Once this is setup, the data source should be detected and appear in the Data Sources panel.

Add documentation for the new data source

Make sure to add a page in the Researcher Manual describing when and how to use the new data source!

Add a File Reader

If this data source has its own concept of "sweeps", consider adding a [FileReader](#) to read that sweep data in the [FileDataSource](#) as well. See [File Readers](#) for details on adding a new one.

File Readers

File readers are a way for [FrameReader](#) to read data frames from different file formats. This is used in the [FileDataSource](#) and [CustomMotionTargetController](#). Each format should have its own file reader. This document will discuss how to add a new file reader to OptiSpeech 2.

Implement [FileReader](#)

Your implementation must implement the following abstract methods:

- `void ReadFrames(StreamReader file, UnityAction<DataFrame> addDataFrame, UnityAction<bool> finish)` - Given a file, read any data frames from it and call `addDataFrame` on each one. Call `finish` before exiting the function. You can pass `true` if the file was read successfully, and `false` otherwise. This function will be called in a background thread.
- `SensorConfiguration[] GetSensorConfigurations(DataFrame dataframe)` - Given an already create DataFrame generated by this file reader, get the sensor configurations for that frame. In some cases the data frame may be ignored and the sensor configurations are determined by the header of the file, or may even be the same for all files this reader supports.

Click each link to see more specific information on implementing each function.

Additionally, there's a `string GetAudioFile(string filename)` function that's called to determine where the audio file is for a given data file, if it exists. By default it just replaced the file extension with `.wav`, but this function can be overridden if there's a file format-specific way to find the audio file. It is not required to check if the audio file does in fact exist, this function just reports where it *would* exist, if it does.

The constructor can be created however you'd like. Since you'll be manually writing the code to instantiate this file reader (due to reasons described in the section below), you have full control over giving it the information it needs to be constructed. This may involve parsing the first so many lines of the file, as a "header", so that the first line next to be read is a data frame line.

Several file formats can have similar designs or patterns, so [FrameReader](#) contains the following utility functions to handle common use cases:

- `bool ReadSingleLineFrames(StreamReader file, UnityAction<DataFrame> addDataFrame, FrameReader.ReadFrameFromString readFrame)`
 - This can help when reading a file that will have a single line for each data frame. It takes the file, the `addDataFrame` delegate, and your own function that can be a valid `FrameReader.ReadFrameFromString` that takes a single-line string and returns a dataframe. It'll return whether or not it read through the whole file successfully.
- `bool HandleFailure(string message, ref int numFailures)` - This can handle writing messages to the console and checking if enough errors have occurred to determine if this file is invalid. Note that you pass in the number of failures, and the function will increment it and check its value for you

If any part is confusing, it's recommended to read through any of the existing file readers to use as a reference.

Update [FrameReader.GetFileReader](#)

Currently there is no standardized way to determine which file reader to use for a given file. The logic to determine which to use is written inside the aforementioned function. Whenever adding a new file reader that function will need to be updated to add logic for when to use the new file reader. This may (hopefully) change in the future to dynamically discovering file reader implementations and call a function on each to determine if that file reader can understand the given file.

Target Types

Targets are a way to display a desired location in 3D space for the patient to try to place their tongue tip near. Different types of targets can handle determining where that 3D location is in different ways. This document will discuss how to add a new target type to OptiSpeech 2.

Implement TargetController

The main part of the target is the controller, this is the class that determines the position of the target and contains any information this type of target uses. It has the following methods that'll need to be implemented:

- `Vector3 GetTargetPosition(long currTime)` - Return where the target should be at the given point in time in milliseconds.
- `long GetCycleDuration()` - Return how long it takes for the target to return to its initial position, in milliseconds. If the target doesn't move, this should just be 0.

If you have any settings (apart from target transparency and radius, which are in the base class) for your target, you'll also need to override these methods:

- `void ApplyConfigFromString(string config)` - Take a tab-separated values string and read this target's settings from it. The string will always be generated via this controller's `ToString()` method so you have complete control over how its formatted.
- `string ToString()` - Writes this target's settings to a tab-separated values string.

Click each link to see more specific information on implementing each function.

Additionally, you can override `void UpdateTarget(Vector3 targetPosition, SensorData? tongueTipSensor)` if you'd like to customize how the target gets updated every frame.

You can set default values for everything like you would with any other MonoBehaviour. Using the `ToString` and `ApplyConfigFromString` methods, values will automatically be loaded and saved as needed.

TIP

The base class already has implementations for target transparency and radius, so you don't need to add those yourself! Just call `base.ApplyConfigFromString(config)` and `base.ToString()` as appropriate. You can use the `TargetController.NUM_BASE_CONFIG_VALUES` constant to look up how many tab separated values are written in `TargetController.ToString()`.

If you choose *not* to use those functions, you can disregard the fact that the settings strings are supposed to be tab-separated values. That is, if you never call `base.ApplyConfigFromString(config)` or `base.ToString()` then you can safely format the strings however you please

If any part is confusing, it's recommended to read through any of the existing target controller implementations to use as a reference.

Implement TargetConfig

The config script handles all the UI elements for updating the targets' values, when that target is selected in the target's panel. Any additional information stored in the `TargetController` implementation should have a corresponding UI element to update it here. Hooking the UI elements up will require overriding the appropriate methods out of the following:

- `void Init(TargetsPanel panel, TargetController controller)` - This function is called when the target is opened in the targets panel. The config object is instantiated and then this function is called. You should setup listeners on each UI element to update `controller`, and set the elements' current values to the ones present in `controller`. You'll probably

want to cast `controller` to the implementation of `TargetController` you made in the previous step. Whenever changing a value on `controller`, make sure to call `panel.SaveTargetsToPrefs();` so the new value is saved.

- `void SetInteractable(bool interactable)` - The targets are only sometimes allowed to be configured by the user, so this function is called to update whether the UI elements should be interactable or not.
- `void OnOpen()` - This is called whenever the target config is opened/the panel is opened while this target is selected
- `void OnClose()` - This is called whenever the target config is closed/the panel is closed while this target is selected

Make sure to call `base.Init(panel, controller)` and `base.SetInteractable(interactable)` in their respective overrides!

If any part is confusing, it's recommended to read through any of the existing target config implementations to use as a reference.

Create a `TargetDescription`

This is a scriptable object that will tell the program about this new target type. It must be inside a folder called `Target Type Descriptions` inside of any `Resources` folder in `Assets`. To create the `TargetDescription`, go to `Optispeech > Target Type` in the create menu. The create menu is accessed through the plus sign in the project panel or by right clicking in a folder and hovering over `Create`. You must then specify the name of the target type and give it two prefabs.

The first prefab is for the target controller. The only requirement for this prefab is that it include the implementation of `TargetController` made in the first step, and have a mesh renderer somewhere in the prefab to represent the target. It can contain anything else you need for this target type, although that's uncommon.

The second prefab is for the target config. The only requirement for this prefab is that it include the implementation of `TargetConfig` made in the previous step. It should also include any UI elements the script needs, naturally.

To make this easier, it's recommended to make variants of the "base" target prefabs. Those are located in the `Prefabs > Targets` folder, and you can create the variants by right clicking them and selecting `Create > Prefab Variant`.

Once this is setup, the target type should be detected and appear in the add target dropdown in the targets panel.

Add documentation for the new target type

Make sure to add a section to the `Targets Panel` documentation describing how this new target type works, and how to configure it!

Testing

This is a big TODO. Unit testing will be super important, and should ideally involve testing each feature with various different profiles, with mocking for each of the different data sources. See [here](#) on how to get started with the [Unity Test Framework](#).

Namespace Optispeech.Accuracy

Classes

[AccuracyDisplayController](#)

Renders the accuracy in the last cycle and sweep.

[AccuracyManager](#)

Tracks how closely the data matches any registered targets over the last cycle and the current sweep. A cycle is a period of time equal to the Lowest Common Multiple of the durations of each target. For example if you have a target that takes 2 seconds to loop and one that takes 3, then the cycle would be each 6 seconds

Class AccuracyDisplayController

Renders the accuracy in the last cycle and sweep.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
AccuracyDisplayController

Namespace: [Optispeech.Accuracy](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class AccuracyDisplayController : MonoBehaviour
```

Fields

cycleLabel

Label used to display the accuracy in the last cycle

Declaration

```
[SerializeField]  
TextMeshProUGUI cycleLabel
```

Field Value

TYPE	DESCRIPTION
TMPro.TextMeshProUGUI	

inSweep

Flag used to track whether a sweep is currently in progress. This information is used to determine whether to update the sweep label, like its supposed to every frame to show the accuracy up to this point in the sweep

Declaration

```
bool inSweep
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

sweepEnded

Since sweeps end on a separate thread, this flag is used to store the fact that a sweep has ended until the next Update, at which point the sweep animator can be told to start fading out the sweep accuracy label

Declaration

```
bool sweepEnded
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

sweepLabel

Label used to display the accuracy in the last sweep

Declaration

[SerializeField] TextMeshProUGUI sweepLabel
--

Field Value

TYPE	DESCRIPTION
TMPro.TextMeshProUGUI	

Methods

UpdateAccuracyLabel(TextMeshProUGUI, Single)

Utility function used to update an accuracy label with a given accuracy. Handles formatting the accuracy and changing the label's text color depending on how good of an accuracy was given

Declaration

void UpdateAccuracyLabel(TextMeshProUGUI label, float accuracy)

Parameters

TYPE	NAME	DESCRIPTION
TMPro.TextMeshProUGUI	label	The accuracy label to update
System.Single	accuracy	The accuracy to display in said label, from 0 to 1 where 1 means perfectly in the target the entire duration

Class AccuracyManager

Tracks how closely the data matches any registered targets over the last cycle and the current sweep. A cycle is a period of time equal to the Lowest Common Multiple of the durations of each target. For example if you have a target that takes 2 seconds to loop and one that takes 3, then the cycle would be each 6 seconds

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
AccuracyManager

Namespace: [Optispeech.Accuracy](#)
Assembly: Assembly-CSharp.dll

Syntax

```
public class AccuracyManager : MonoBehaviour
```

Fields

currTime

How far we are in the current cycle

Declaration

```
long currTime
```

Field Value

TYPE	DESCRIPTION
System.Int64	

cycleDuration

How long one cycle lasts

Declaration

```
long cycleDuration
```

Field Value

TYPE	DESCRIPTION
System.Int64	

cycleEnded

Since frames are processed in a different thread, this is used to store the fact the cycle has ended until the next Update, where the cycle ending can be handled properly

Declaration

```
bool cycleEnded
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Instance

Static member to access the singleton instance of this class

Declaration

```
public static AccuracyManager Instance
```

Field Value

TYPE	DESCRIPTION
AccuracyManager	

lastFrame

Reference to the previous frame, to calculate how much time has passed (note that the current data source's "lastFrame" is effectively our current frame)

Declaration

```
DataFrame? lastFrame
```

Field Value

TYPE	DESCRIPTION
System.Nullable<DataFrame>	

measurementsCount

How many accuracies have been added into sumAccuracy. We store this so we can divide the sumAccuracy to get our average accuracy, even though we don't know how many measurements we'll have until the cycle ends

Declaration

```
int measurementsCount
```

Field Value

TYPE	DESCRIPTION
System.Int32	

onCycleEnd

Event that fires whenever a cycle ends

Declaration

```
[HideInInspector]  
public UnityEvent<float> onCycleEnd
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent<System.Single>	

sumAccuracy

A sum of the accuracies of all targets thus far in the cycle

Declaration

float sumAccuracy

Field Value

TYPE	DESCRIPTION
System.Single	

sweepMeasurementsCount

How many accuracies have been added into sweepSumAccuracy.

Declaration

int sweepMeasurementsCount

Field Value

TYPE	DESCRIPTION
System.Int32	

sweepSumAccuracy

A sum of the accuracies of all targets thus far in the sweep

Declaration

float sweepSumAccuracy

Field Value

TYPE	DESCRIPTION
System.Single	

Methods

CalculateDurationLCM()

Calculates and updates the the duration of a cycle by finding the Lowest Common Multiple of all the targets' cycles

Declaration

public void CalculateDurationLCM()

GCD(Int64, Int64)

Utility function for calculating the Greatest Common Denominator of two numbers, which is used when calculating LCM

Declaration

```
long GCD(long a, long b)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int64	a	
System.Int64	b	

Returns

TYPE	DESCRIPTION
System.Int64	The greatest common denominator of <code>a</code> and <code>b</code>

GetSweepAccuracy()

Utility function for safely calculating sweep accuracy even if there is no data

Declaration

```
public float GetSweepAccuracy()
```

Returns

TYPE	DESCRIPTION
System.Single	The average sweep accuracy from 0 to 1

ProcessFrame(DataFrame)

"Processes" a frame by calculating the accuracy for each target and adding them to the current sweep and cycle sumAccuracy counts, as well as updating the measurementsCount for each

Declaration

```
void ProcessFrame(DataFrame frame)
```

Parameters

TYPE	NAME	DESCRIPTION
DataFrame	frame	The data frame to process

Namespace Optispeech.Advanced

Classes

[AdvancedPanel](#)

Panel to update the filter strength on the low pass filter and advanced data source settings

[FilterManager](#)

Manager singleton for applying a low pass filter to DataFrame objects as they're read

Class AdvancedPanel

Panel to update the filter strength on the low pass filter and advanced data source settings

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
AdvancedPanel

Namespace: [Optispeech.Advanced](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class AdvancedPanel : MonoBehaviour
```

Fields

hostField

Number field for the WaveFront data source host

Declaration

```
[SerializeField]  
TMP_InputField hostField
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

portField

Number field for the WaveFront data source port

Declaration

```
[SerializeField]  
TMP_InputField portField
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

strengthField

Slider for the user to change the strength of the low pass filter

Declaration

```
[SerializeField]  
Slider strengthField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Slider	

Class FilterManager

Manager singleton for applying a low pass filter to DataFrame objects as they're read

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
FilterManager

Namespace: [Optispeech.Advanced](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class FilterManager : MonoBehaviour
```

Fields

defaultStrength

Store initial strength to use as default for new profiles. 0 is no affect

Declaration

```
public int defaultStrength
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Instance

Static member to access the singleton instance of this class

Declaration

```
public static FilterManager Instance
```

Field Value

TYPE	DESCRIPTION
FilterManager	

previousFrame

Used to store the previous frame if it exists. Whenever we process a data frame we'll set this, and set it to null whenever switching sources. Additionally, a data source implementation may choose to reset the previousFrame to null when they want, such as when its config changes

Declaration

```
public DataFrame? previousFrame
```

Field Value

TYPE	DESCRIPTION
System.Nullable< DataFrame >	

Methods

ApplyFilter(DataFrame, Nullable<DataFrame>)

Applies the low pass filter to the given data frame based on the previous frame. Optionally, a previous frame can be passed to override [previousFrame](#). If overridden, then [previousFrame](#) will NOT be set to `frame`.

Declaration

```
public DataFrame ApplyFilter(DataFrame frame, DataFrame? prevFrame = default(DataFrame? ))
```

Parameters

TYPE	NAME	DESCRIPTION
DataFrame	frame	The frame to have the filter applied to
System.Nullable< DataFrame >	prevFrame	The previous data frame if overriding previousFrame

Returns

TYPE	DESCRIPTION
DataFrame	<code>frame</code> , after aplying the filter

ApplyFilter(Nullable<SensorData>, Nullable<SensorData>, Single)

Applies a low pass filter to a specific sensor data

Declaration

```
SensorData? ApplyFilter(SensorData? rawData, SensorData? previousData, float alpha)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Nullable< SensorData >	rawData	The sensor data to apply the filter to
System.Nullable< SensorData >	previousData	The previous data for the same sensor
System.Single	alpha	The strength of the filter to apply

Returns

TYPE	DESCRIPTION
System.Nullable< SensorData >	The filtered sensor data

GetFilteredValue(Single, Single, Single)

Applies a low pass filter to a number

Declaration

```
float GetFilteredValue(float rawValue, float previousValue, float alpha)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	rawValue	The current number to apply the filter to
System.Single	previousValue	The previous value for that number
System.Single	alpha	The strength of the filter to apply

Returns

TYPE	DESCRIPTION
System.Single	The filtered number

GetFilteredValue(Quaternion, Quaternion, Single)

Applies a low pass filter to a quaternion value

Declaration

```
Quaternion GetFilteredValue(Quaternion rawValue, Quaternion previousValue, float alpha)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Quaternion	rawValue	The quaternion to apply the filter to
UnityEngine.Quaternion	previousValue	The previous data for that quaternion
System.Single	alpha	The strength of the filter to apply

Returns

TYPE	DESCRIPTION
UnityEngine.Quaternion	The filtered quaternion value

GetFilteredValue(Vector3, Vector3, Single)

Applies a low pass filter to a vector value

Declaration

```
Vector3 GetFilteredValue(Vector3 rawValue, Vector3 previousValue, float alpha)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Vector3	rawValue	The vector to apply the filter to
UnityEngine.Vector3	previousValue	The previous data for that vector
System.Single	alpha	The strength of the filter to apply

Returns

TYPE	DESCRIPTION
UnityEngine.Vector3	The filtered vector value

Namespace Optispeech.Avatar

Classes

[AvatarController](#)

Controls the avatar and handles updating it based on the current dataframe

[RotationCopyCat](#)

Utility behaviour to match this object's rotation to another. Used on our "inner" head (which is used to help occlude body parts to help the avatar have a smooth fade out when the camera is close, as opposed to just clipping through) to make its jaw match the "outer" head's jaw

[TongueController](#)

Controls the avatar's tongue, based on the rigged joints in the tongue model. Since these are the rigged joints, we can't have more or less of them without updating the model/rig.

Structs

[TongueController.PositionAndRotation](#)

Utility struct for saving and loading position and rotation from a transform

Class AvatarController

Controls the avatar and handles updating it based on the current dataframe

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- AvatarController

Namespace: [Optispeech.Avatar](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class AvatarController : MonoBehaviour
```

Fields

forehead

A reference point at the front of the head. If the patient is given glasses this sensor can be attached to the bridge of the glasses

Declaration

```
[Header("Head Rig Reference Points")]  
[SerializeField]  
Transform forehead
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

foreheadPosition

Pre-"calculated" position of the forehead reference point, used when creating the transformation matrix

Declaration

```
Vector3 foreheadPosition
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

jaw

A reference point on the bottom of the head. Since heads have different sizes the jaw is used to offset the points so the tongue appears in the right place relative to the avatar's jaw

Declaration

```
[SerializeField]  
Transform jaw
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

jawPosition

Pre-"calculated" position of the jaw reference point, used when creating the transformation matrix

Declaration

Vector3 jawPosition

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

lastTimestamp

Reference to the timestamp of the most recently displayed data frame. Since a new Update call doesn't necessarily correspond to a new data frame, and since its not useful to update for a data frame more than once per Update, this is used to check if the current dataframe is a new one, in the Update method. There is a slight possibility of this not updating when it should if two data frames have the exact same timestamp, which would require them to have been recorded in the same millisecond, or more likely be a result from changing data sources that coincidentally gives a frame with the same timestamp. In those cases, the problem will only last until the next data frame comes and fixes it, though

Declaration

long lastTimestamp

Field Value

TYPE	DESCRIPTION
System.Int64	

leftEar

A reference point on the left side of the head on the same plane as the forehead sensor. If the patient is given glasses this sensor can be attached above the left ear

Declaration

[SerializeField] Transform leftEar

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

leftEarPosition

Pre-"calculated" position of the left ear reference point, used when creating the transformation matrix

Declaration

Vector3 leftEarPosition

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

Main

Technically this class doesn't need to be a singleton, but dataframes will use the positions of some avatar's reference points (forehead, ears, and jaw) to calculate the transformation matrix to map raw data onto the avatar, so this is a static member to access the "main" instance (as opposed to the singleton instance, which won't be enforced)

Declaration

```
public static AvatarController Main
```

Field Value

TYPE	DESCRIPTION
AvatarController	

markerPrefab

This prefab is used to instantiate a dynamic number of sensor markers for each sensor with type "OTHER"

Declaration

```
[SerializeField]  
GameObject markerPrefab
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

markersParent

This container is where any markers for sensors with type "OTHER" will be instantiated into

Declaration

```
[Header("Sensor Markers")]  
[SerializeField]  
Transform markersParent
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

otherSensorMarkers

List of currently instantiated markers for sensors with type "OTHER". Will be recreated whenever the number of such sensors changes

Declaration

```
GameObject[] otherSensorMarkers
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject[]	

rightEar

A reference point on the right side of the head on the same plane as the forehead sensor. If the patient is given glasses this sensor can be attached above the right ear

Declaration

```
[SerializeField]  
Transform rightEar
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

rightEarPosition

Pre-"calculated" position of the right ear reference point, used when creating the transformation matrix

Declaration

```
Vector3 rightEarPosition
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

rigLookDir

Pre-calculated rotation looking from the left ear to the right ear. This will also be calculated for each dataframe to create the transform matrix to place the raw data onto the avatar.

Declaration

```
Quaternion rigLookDir
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Quaternion	

tongueController

The tongue has its own controller due to its own complexity, and this reference is used to give it the data it needs to update

Declaration

```
[SerializeField]  
TongueController tongueController
```

Field Value

TYPE	DESCRIPTION
TongueController	

Methods

GetReal2RigMatrix(Nullable<SensorData>, Nullable<SensorData>, Nullable<SensorData>, Nullable<SensorData>)

Create transformation matrix using the reference markers on the left, right, and/or front sides of the head The matrix will transform points such that the tongue sensors' raw positions can map onto the avatar's tongue

Declaration

```
public Matrix4x4 GetReal2RigMatrix(SensorData? foreheadSensor, SensorData? jawSensor, SensorData?  
leftEarSensor, SensorData? rightEarSensor)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Nullable< SensorData >	foreheadSensor	The sensor at the front of the head, if exists
System.Nullable< SensorData >	jawSensor	The sensor at the bottom of the head, if exists
System.Nullable< SensorData >	leftEarSensor	The sensor on the left side of the head, if exists
System.Nullable< SensorData >	rightEarSensor	The sensor on the right side of the head, if exists

Returns

TYPE	DESCRIPTION
UnityEngine.Matrix4x4	A transformation matrix that will map sensor data onto the avatar

SetRestingPosition(SensorData[])

Calculates the offset needed to make the tongue appear the correct distance from the avatar's jaw, to account for the size or shape of the patient not matching the avatar

Declaration

```
void SetRestingPosition(SensorData[] sensorData)
```

Parameters

TYPE	NAME	DESCRIPTION
SensorData []	sensorData	The raw data from one data frame

Class RotationCopyCat

Utility behaviour to match this object's rotation to another. Used on our "inner" head (which is used to help occlude body parts to help the avatar have a smooth fade out when the camera is close, as opposed to just clipping through) to make its jaw match the "outer" head's jaw

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- RotationCopyCat

Namespace: [Optispeech.Avatar](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class RotationCopyCat : MonoBehaviour
```

Fields

rotationSource

The target transform to copy the rotation from every frame

Declaration

```
[SerializeField]  
Transform rotationSource
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

Class TongueController

Controls the avatar's tongue, based on the rigged joints in the tongue model. Since these are the rigged joints, we can't have more of less of them without updating the model/rig.

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- TongueController

Namespace: [Optispeech.Avatar](#)
Assembly: Assembly-CSharp.dll

Syntax

```
public class TongueController : MonoBehaviour
```

Fields

backSensor

The sensor marker on the back of the tongue. Also has three children that influence the rig

Declaration

```
[SerializeField]  
Transform backSensor
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

centerJoint

The rigged joint at the center of the tongue model

Declaration

```
[SerializeField]  
Transform centerJoint
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

dorsumSensor

The sensor marker on the tongue's dorsum. Also has two children that influence the rig

Declaration

```
[SerializeField]  
Transform dorsumSensor
```


Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

farLowerJoint

The rigged joint slightly above the base of the tongue model

Declaration

<code>[SerializeField]</code> <code>Transform farLowerJoint</code>

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

farUpperJoint

The rigged joint slightly below the tip of the tongue model

Declaration

<code>[SerializeField]</code> <code>Transform farUpperJoint</code>

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

hasMoved

Tracks whether or not we've been moved from our default state. If told to reset and we haven't moved, we'll just do nothing

Declaration

<code>bool</code> hasMoved

Field Value

TYPE	DESCRIPTION
System.Boolean	

initialBackSensor

Saved position and rotation of the tongue back sensor marker in our default state

Declaration

<code>TongueController.PositionAndRotation</code> initialBackSensor

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialCenterJoint

Saved position and rotation of the center joint in our default state

Declaration

<code>TongueController.PositionAndRotation initialCenterJoint</code>
--

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialDorsumSensor

Saved position and rotation of the tongue dorsum sensor marker in our default state

Declaration

<code>TongueController.PositionAndRotation initialDorsumSensor</code>

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialFarLowerJoint

Saved position and rotation of the far lower joint in our default state

Declaration

<code>TongueController.PositionAndRotation initialFarLowerJoint</code>
--

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialFarUpperJoint

Saved position and rotation of the far upper joint in our default state

Declaration

<code>TongueController.PositionAndRotation initialFarUpperJoint</code>
--

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialJawRot

Saved position and rotation of the jaw rotation joint in our default state

Declaration

```
TongueController.PositionAndRotation initialJawRot
```

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialLeftJoint

Saved position and rotation of the left joint in our default state

Declaration

```
TongueController.PositionAndRotation initialLeftJoint
```

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialLeftSensor

Saved position and rotation of the tongue left sensor marker in our default state

Declaration

```
TongueController.PositionAndRotation initialLeftSensor
```

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialLowerJoint

Saved position and rotation of the lower joint in our default state

Declaration

```
TongueController.PositionAndRotation initialLowerJoint
```

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialRightJoint

Saved position and rotation of the right joint in our default state

Declaration

```
TongueController.PositionAndRotation initialRightJoint
```

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialRightSensor

Saved position and rotation of the tongue right sensor marker in our default state

Declaration

```
TongueController.PositionAndRotation initialRightSensor
```

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialRootJoint

Saved position and rotation of the root joint in our default state

Declaration

```
TongueController.PositionAndRotation initialRootJoint
```

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialTipJoint

Saved position and rotation of the tip joint in our default state

Declaration

```
TongueController.PositionAndRotation initialTipJoint
```

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialTipSensor

Saved position and rotation of the tongue tip sensor marker in our default state

Declaration

```
TongueController.PositionAndRotation initialTipSensor
```

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

initialUpperJoint

Saved position and rotation of the upper joint in our default state

Declaration

<code>TongueController.PositionAndRotation initialUpperJoint</code>

Field Value

TYPE	DESCRIPTION
TongueController.PositionAndRotation	

jawRot

The rigged joint that, when rotated, controls the angle the jaw makes

Declaration

<code>[SerializeField] Transform jawRot</code>
--

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

jawSensor

The sensor marker at the jaw's rotation pivot

Declaration

<code>[SerializeField] Transform jawSensor</code>

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

leftJoint

The rigged joint left of the center of the tongue model

Declaration

<code>[SerializeField] Transform leftJoint</code>

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

leftSensor

The sensor marker on the left of the tongue. Also has two children that influence the rig

Declaration

[SerializeField] Transform leftSensor
--

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

lowerJoint

The rigged joint slightly below the center of the tongue model

Declaration

[SerializeField] Transform lowerJoint
--

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

rightJoint

The rigged joint right of the center of the tongue model

Declaration

[SerializeField] Transform rightJoint
--

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

rightSensor

The sensor marker on the right of the tongue. Also has two children that influence the rig

Declaration

[SerializeField] Transform rightSensor

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

rootJoint

The rigged joint at the base of the tongue model

Declaration

```
[Header("Joints")]
[SerializeField]
Transform rootJoint
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

tipJoint

The rigged joint at the tip of the tongue model

Declaration

```
[SerializeField]
Transform tipJoint
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

tipSensor

The sensor marker on the tip of the tongue. Also has three children the influence the rig

Declaration

```
[Header("Sensor Markers")]
[SerializeField]
Transform tipSensor
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

upperJoint

The rigged joint slightly above the center of the tongue model

Declaration

```
[SerializeField]
Transform upperJoint
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

Methods

Reset()

If the rig has moved out of its default state, apply all the initial positions and rotations to revert back

Declaration

```
public void Reset()
```

UpdateRig(TransformedData)

Updates the model so the tongue rig matches the data given

Declaration

```
public void UpdateRig(TransformedData data)
```

Parameters

TYPE	NAME	DESCRIPTION
TransformedData	data	The transformed data to apply to the tongue rig

Struct TongueController.PositionAndRotation

Utility struct for saving and loading position and rotation from a transform

Namespace: [Optispeech.Avatar](#)

Assembly: Assembly-CSharp.dll

Syntax

```
struct PositionAndRotation
```

Constructors

PositionAndRotation(Transform)

Constructor that will take the position and rotation from the provided transform

Declaration

```
public PositionAndRotation(Transform transform)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Transform	transform	The transform to copy the position and rotation from

Fields

position

Position to save

Declaration

```
public Vector3 position
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

rotation

Rotation to save

Declaration

```
public Quaternion rotation
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Quaternion	

Methods

Apply(Transform)

Applies the saved position and rotation to the target transform

Declaration

```
public void Apply(Transform transform)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Transform	transform	The transform to copy the position and rotation to

Namespace Optispeech.Data

Classes

[DataSourceDescription](#)

A scriptable object that contains information about a specific data source. The data source list will look for all instances of this scriptable object in any "Data Source Descriptions" folder inside of a Resources folder anywhere in Assets

[DataSourceList](#)

This panel shows a list of data sources, whether or not they can be made active, and allows the user to see and change which data source is active

[DataSourceManager](#)

Tracks the active data source and handles changing or deselecting the active data source

[DataSourceReader](#)

A abstract source for data frames

[DataSourceSelector](#)

This is a UGUI object that represents a data source in the data source list. It has a toggle to change whether its the active data source or not, and communicates to the user the current status of this data source

[TcpClientController](#)

Multiple data source readers may need to setup a TCP connection with very similar settings, so this class can be used as a utility to set them all up appropriately

Structs

[DataFrame](#)

A single "frame" of data from a data source. A "frame" is a single reading from the data source, and contains the timestamp it was either given by the data source or when it was received by this program, as well as the data recorded for each sensor in this frame. The frame will also be processed and then given the data once transformed to fit on the avatar model as well as the positions of each of the targets at this timestamp

[SensorData](#)

The data for a single sensor inside of a data frame

[TransformedData](#)

Struct to hold the transformed data for a given data frame. Stores specific sensors by role, and includes the transformation matrix used to map the raw sensors to their transformed versions

Enums

[DataSourceReader.DataSourceReaderStatus](#)

The different statuses a data source reader may be in

[SensorStatus](#)

Different data sources may have different ways of reporting sensor statuses, so we map them as best as we can to these different status categories (based on the WaveFront statuses). With these we can display sensor statuses to the user without any data source-specific code.

Struct DataFrame

A single "frame" of data from a data source. A "frame" is a single reading from the data source, and contains the timestamp it was either given by the data source or when it was received by this program, as well as the data recorded for each sensor in this frame. The frame will also be processed and then given the data once transformed to fit on the avatar model as well as the positions of each of the targets at this timestamp

Namespace: [Optispeech.Data](#)
Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public struct DataFrame
```

Fields

sensorData

An array of raw sensor data directly from the data source

Declaration

```
public SensorData[] sensorData
```

Field Value

TYPE	DESCRIPTION
SensorData[]	

targetPositions

A dictionary of positions for each active target. Since many targets can have different positions over time, this is the position that target has at this frame's timestamp

Declaration

```
public Dictionary<string, Vector3> targetPositions
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, UnityEngine.Vector3>	

timestamp

The timestamp for this data frame. This will either be given by the data source directly if possible, or calculated directly by this program.

Declaration

```
public long timestamp
```

Field Value

TYPE	DESCRIPTION
System.Int64	

transformedData

A processed version of the sensor data that's been mapped onto the avatar model

Declaration

```
public TransformedData transformedData
```

Field Value

TYPE	DESCRIPTION
TransformedData	

Methods

GetSensorData(SensorData[], SensorConfiguration)

Utility function used while processing a data frame that finds the first sensor that fills a specified role, if any. Also applies the pre-offset configured for that sensor, and adds the configured post offset to the sensor data.

Declaration

```
static SensorData? GetSensorData(SensorData[] sensorData, SensorConfiguration sensorConfig)
```

Parameters

TYPE	NAME	DESCRIPTION
SensorData[]	sensorData	The raw sensor data from a data source
SensorConfiguration	sensorConfig	The configuration for the sensor role to search for

Returns

TYPE	DESCRIPTION
System.Nullable<SensorData>	The found sensor data, if any

GetTransformedData(SensorData[])

Static method for processing raw sensor data to map them onto an avatar model. This will use the raw data to construct an appropriate transformation matrix, find the first sensor data for each "role", and store each of their transformed versions

Declaration

```
public static TransformedData GetTransformedData(SensorData[] sensorData)
```

Parameters

TYPE	NAME	DESCRIPTION
SensorData[]	sensorData	The raw sensor data from a data source

Returns

TYPE	DESCRIPTION
TransformedData	A struct of processed data mapping each sensor role onto the avatar model

GetTransformedSensorData(Nullable<SensorData>, Vector3, Matrix4x4)

Utility function used while processing a data frame that takes a sensor data as well as the forehead position and the transformation matrix to map sensors onto the avatar model, and applies the matrix to the sensor data to return the transformed data

Declaration

```
static SensorData? GetTransformedSensorData(SensorData? sensor, Vector3 foreheadOffset, Matrix4x4 transformMatrix)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Nullable<SensorData>	sensor	The sensor data to transform
UnityEngine.Vector3	foreheadOffset	The forehead position, used to position the sensor relative to the forehead
UnityEngine.Matrix4x4	transformMatrix	The transformation matrix that maps sensors onto the avatar model

Returns

TYPE	DESCRIPTION
System.Nullable<SensorData>	The sensor data but mapped onto the avatar model

Class DataSourceDescription

A scriptable object that contains information about a specific data source. The data source list will look for all instances of this scriptable object in any "Data Source Descriptions" folder inside of a Resources folder anywhere in Assets

Inheritance

System.Object
UnityEngine.Object
UnityEngine.ScriptableObject
DataSourceDescription

Namespace: [Optispeech.Data](#)

Assembly: Assembly-CSharp.dll

Syntax

```
[CreateAssetMenu(menuName = "Optispeech/Data Source")]  
public class DataSourceDescription : ScriptableObject
```

Fields

readerPrefab

The prefab that will be instantiated whenever this data source is the active one

Declaration

```
public DataSourceReader readerPrefab
```

Field Value

TYPE	DESCRIPTION
DataSourceReader	

sourceName

The name of this data source, as it should appear in the data source list

Declaration

```
public string sourceName
```

Field Value

TYPE	DESCRIPTION
System.String	

Class DataSourceList

This panel shows a list of data sources, whether or not they can be made active, and allows the user to see and change which data source is active

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
DataSourceList

Namespace: [Optispeech.Data](#)
Assembly: Assembly-CSharp.dll

Syntax

```
[RequireComponent(typeof(RectTransform))]  
public class DataSourceList : MonoBehaviour
```

Fields

activeToggle

The toggle on the data source selector that represents the currently active data source. This is stored so when the active data source changes the toggle's isOn property can be updated

Declaration

```
Toggle activeToggle
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

contentContainer

A container to place each data source selector inside of

Declaration

```
[SerializeField]  
RectTransform contentContainer
```

Field Value

TYPE	DESCRIPTION
UnityEngine.RectTransform	

dataSourceSelectorPrefab

A prefab with a DataSourceSelector MonoBehaviour on it that represents a single data source in the list. This class should have a RectTransform with a static height

Declaration


```
[SerializeField]  
GameObject dataSourceSelectorPrefab
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

refreshSourcesButton

A button that will tell each data source to re-check whether or not it can be made active

Declaration

```
[SerializeField]  
Button refreshSourcesButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

Methods

IsActiveToggle(Toggle)

A function intended to be called by the data source selectors to check whether they are currently the active toggle

Declaration

```
public bool IsActiveToggle(Toggle newToggle)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.UI.Toggle	newToggle	The toggle to check

Returns

TYPE	DESCRIPTION
System.Boolean	Whether or not <code>newToggle</code> is the active toggle

Refresh()

Removes the list of data source selectors if it exists and re-creates it

Declaration

```
public void Refresh()
```

SetActiveToggle(Toggle)

A function intended to be called by the data source selectors that updates the active toggle, while turning off the current active toggle if it exists

Declaration

```
public void SetActiveToggle(Toggle newToggle)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.UI.Toggle	newToggle	The toggle to store as the active toggle

Class DataSourceManager

Tracks the active data source and handles changing or deselecting the active data source

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
DataSourceManager

Namespace: [Optispeech.Data](#)
Assembly: Assembly-CSharp.dll

Syntax

```
public class DataSourceManager : MonoBehaviour
```

Fields

activeDataSource

The instantiated object of the active data source reader

Declaration

```
GameObject activeDataSource
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

dataSourceName

The name of the active data source reader, as specified by its DataSourceDescription

Declaration

```
[HideInInspector]  
public string dataSourceName
```

Field Value

TYPE	DESCRIPTION
System.String	

dataSourceReader

The active data source reader

Declaration

```
[HideInInspector]  
public DataSourceReader dataSourceReader
```

Field Value

TYPE	DESCRIPTION
DataSourceReader	

Instance

Static member to access the singleton instance of this class

Declaration

```
public static DataSourceManager Instance
```

Field Value

TYPE	DESCRIPTION
DataSourceManager	

noSourceSelectedCanvas

Technically this doesn't need to be a canvas. This is an object that will be set active or inactive to tell the player when there is NO active data source. This is intended to notify someone why the tongue model is not moving, so they know what they need to do to change that. This is additionally important for new users, so they know what the first thing they need to do is. To this end, the object may also want to provide basic getting started information to the user.

Declaration

```
[SerializeField]  
GameObject noSourceSelectedCanvas
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

onFrameRead

Event that fires whenever a data frame is read by the active data source

Declaration

```
[HideInInspector]  
public UnityEvent<DataFrame> onFrameRead
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent< DataFrame >	

onSourceChanged

Event that fires whenever the active data source changes

Declaration

```
[HideInInspector]  
public UnityEvent<DataSourceReader> onSourceChanged
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent< DataSourceReader >	

sourceSettingsAccordion

Some settings are only useful to have visible when there is an active data source, so this accordion appears on the opposite side of the screen as the universal setting accordion. The active data source reader will be instantiated as a child of this accordion, so it may add additional panels with source-specific settings.

Declaration

<code>[SerializeField]</code> <code>Accordion sourceSettingsAccordion</code>

Field Value

TYPE	DESCRIPTION
Accordion	

Methods

DelayAccordion(String)

Waits a frame and then sets up the sensors panel, and open the source settings accordion

Declaration

<code>IEnumerator DelayAccordion(string sourceName)</code>
--

Parameters

TYPE	NAME	DESCRIPTION
System.String	sourceName	The name of the data source, used to load sensors configurations for that source specifically

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	A coroutine that waits a frame before actually setting up the sensors panel

ResetTargets(ProfileManager.Profile)

Clears all targets and then recreates the targets (and sensors) panels later, if applicable

Declaration

<code>void ResetTargets(ProfileManager.Profile profile)</code>
--

Parameters

TYPE	NAME	DESCRIPTION
ProfileManager.Profile	profile	

SetActiveDataSourceReader(DataSourceReader, String)

Changes the active data source readerto the specified one, or just deselects the active data source if null is passed

Declaration

```
public void SetActiveDataSourceReader(DataSourceReader dataSourceReader, string sourceName = "")
```

Parameters

TYPE	NAME	DESCRIPTION
DataSourceReader	dataSourceReader	The data source to make active, or null
System.String	sourceName	The name of the data source, used to load targets for this data source. Ignored if dataSourceReader is null

Class DataSourceReader

A abstract source for data frames

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- DataSourceReader
- [FileDataSource](#)
- [WaveFrontDataSource](#)

Namespace: [Optispeech.Data](#)
Assembly: Assembly-CSharp.dll

Syntax

```
public abstract class DataSourceReader : MonoBehaviour
```

Fields

dataQueue

This FIFO queue stores data frames in a thread-safe way so that this class can add new frames, and the other parts of the program can read/consume them

Declaration

```
public ConcurrentQueue<Dataframe> dataQueue
```

Field Value

TYPE	DESCRIPTION
System.Collections.Concurrent.ConcurrentQueue< Dataframe >	

isActive

This flag gets read by the new thread to tell it when to stop

Declaration

```
bool isActive
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

lastFrame

This variable tracks the last frame added to the queue It will be used by the avatar, which only ever cares about the most recent sensor states

Declaration

```
public Dataframe lastFrame
```

Field Value

TYPE	DESCRIPTION
DataFrame	

statusChangeEvent

This event can be used to subscribe to status changes. If determining the status can take awhile, its recommended to send [UNKNOWN](#) in GetCurrentStatus, use another thread to determine the actual status, and call this delegate once the actual status is known

Declaration

<pre>public UnityEvent<DataSourceReader.DataSourceReaderStatus> statusChangeEvent</pre>

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent< DataSourceReader.DataSourceReaderStatus >	

stopwatch

This stopwatch is used to generate precise timestamps if they aren't provided by the data source itself

Declaration

<pre>Stopwatch stopwatch</pre>

Field Value

TYPE	DESCRIPTION
System.Diagnostics.Stopwatch	

Methods

AreSensorsConfigurable()

Implementations should return true unless they want to handle sensors themselves, e.g. FileDataSource. This'll prevent us from saving/loading sensors from our profile and will make all fields readonly in the Sensors panel

Declaration

<pre>public virtual bool AreSensorsConfigurable()</pre>

Returns

TYPE	DESCRIPTION
System.Boolean	Whether or not sensors can be changed by the user

AreTargetsConfigurable()

Implementations should return true unless they want to handle targets themselves, e.g. FileDataSource. This'll prevent us from saving/loading targets from our profile and will make all fields readonly in the Targets panel

Declaration


```
public virtual bool AreTargetsConfigurable()
```

Returns

TYPE	DESCRIPTION
System.Boolean	Whether or not targets can be changed by the user

Cleanup()

Optional method that implementations can override to cleanup anything whenever a data source is un-selected

Declaration

```
protected virtual void Cleanup()
```

GetCurrentStatus()

Implementations should return AVAILABLE if this source can currently be used Can be determined by e.g. searching if a type of process is currently running or detect the hardware directly. In that case this function should return UNKOWN and use statusChangeEvent later

Declaration

```
public abstract DataSourceReader.DataSourceReaderStatus GetCurrentStatus()
```

Returns

TYPE	DESCRIPTION
DataSourceReader.DataSourceReaderStatus	The current availability status of the data source

GetDefaultSensorConfigurations()

If the sensors are configurable, then this method should be overridden to provide a sensible default list of sensor configs

Declaration

```
public virtual SensorConfiguration[] GetDefaultSensorConfigurations()
```

Returns

TYPE	DESCRIPTION
SensorConfiguration[]	A set of sensible default sensor configurations for this data source

IsTimestampProvided()

Implementations should probably just return a constant without needing to perform any calculations. The source should have an inherent value for this - either the data it reads comes with a timestamp or it doesn't. This function returning different values at different times may lead to undefined behavior. If this returns false, a timestamp will be calculated immediately following a data frame being received

Declaration

```
protected virtual bool IsTimestampProvided()
```

Returns

TYPE	DESCRIPTION
System.Boolean	Whether or not the data frames produced by this data source have a valid timestamp value

OnEnable()

This function gets called automatically whenever a gameobject with this monobehaviour is added to the scene (as well as whenever the code gets re-compiled automatically while in play mode).

Declaration

```
protected void OnEnable()
```

ReadFrame()

Implementations should have most of their logic in this function Since this will be run in a separate thread from the one Unity is using, implementations can freely wait until the next data frame is available

Declaration

```
protected abstract DataFrame ReadFrame()
```

Returns

TYPE	DESCRIPTION
DataFrame	The next data frame

Remarks

Implementations do NOT need to fill in any data for [transformedData](#) or [targetPositions](#); that will be calculated automatically. [timestamp](#) is only required if [IsTimestampProvided\(\)](#) returns `false`.

Run()

This is the function that gets run in the new thread It detects if data should still be read, processes the next data frame, and repeats

Declaration

```
void Run()
```

SetupStopwatch()

Setup a stopwatch to take very precise timestamp measurements, based on [these recommendations](#)

Declaration

```
public static Stopwatch SetupStopwatch()
```

Returns

TYPE	DESCRIPTION
System.Diagnostics.Stopwatch	A setup stopwatch

StartSweep(String, String)

If supported, send message to data source reader to start a sweep

Declaration

```
public virtual void StartSweep(string folderPath, string sweepName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	folderPath	The folder path to store any sweep data in
System.String	sweepName	The name of this sweep

StartThread()

Starts the thread that will read data frames in the background. This method is virtual so implementations can add their own code that needs to run when the thread is started, such as initializing a connection to the data source. If that is done, the implementation should still call

```
base.StartThread()
```

Declaration

```
protected virtual void StartThread()
```

StopSweep()

If supported, send message to data source reader to stop a sweep

Declaration

```
public virtual void StopSweep()
```

Enum DataSourceReader.DataSourceReaderStatus

The different statuses a data source reader may be in

Namespace: [Optispeech.Data](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public enum DataSourceReaderStatus
```

Fields

NAME	DESCRIPTION
AVAILABLE	
UNAVAILABLE	
UNKNOWN	

Class DataSourceSelector

This is a UGUI object that represents a data source in the data source list. It has a toggle to change whether its the active data source or not, and communicates to the user the current status of this data source

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- DataSourceSelector

Namespace: [Optispeech.Data](#)
Assembly: Assembly-CSharp.dll

Syntax

```
[RequireComponent(typeof(Toggle))]  
public class DataSourceSelector : MonoBehaviour
```

Fields

availableStatusIcon

The sprite to show when the data source represented by this data selector is currently available to be selected

Declaration

```
[SerializeField]  
Sprite availableStatusIcon
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

description

The description of the data source represented by this selector

Declaration

```
DataSourceDescription description
```

Field Value

TYPE	DESCRIPTION
DataSourceDescription	

failedStatusIcon

The sprite to show when the data source represented by this data selector is currently not available to be selected

Declaration

```
[SerializeField]  
Sprite failedStatusIcon
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

nameLabel

A label to show the name of the data source represented by this selector

Declaration

[SerializeField] TextMeshProUGUI nameLabel

Field Value

TYPE	DESCRIPTION
TMPro.TextMeshProUGUI	

sourceList

A reference to the list of data source selectors this selector is apart of. This is used to communicate back to the list whenever this data source is toggled

Declaration

DataSourceList sourceList

Field Value

TYPE	DESCRIPTION
DataSourceList	

sourceReader

The DataSourceReader component of the prefab of the data source represented by this selector

Declaration

DataSourceReader sourceReader

Field Value

TYPE	DESCRIPTION
DataSourceReader	

statusIndicator

An image that will change sprite dependent on the current status of the data source represented by this selector

Declaration

[SerializeField] Image statusIndicator

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Image	

toggle

The toggle used to change whether the data source represented by this selector is the active data source or not

Declaration

Toggle toggle

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

waitingStatusIcon

The sprite to show when the data source represented by this data selector is currently determining its availability

Declaration

[SerializeField] Sprite waitingStatusIcon
--

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

Methods

Init(DataSourceList, DataSourceDescription)

Initializes this selector so it will represent the data source described to it

Declaration

public void Init(DataSourceList sourceList, DataSourceDescription description)

Parameters

TYPE	NAME	DESCRIPTION
DataSourceList	sourceList	The data source list this selector is apart of
DataSourceDescription	description	The description of the data source to represent

OnStatusChange(DataSourceReader.DataSourceReaderStatus)

Callback method given to the data source reader that will update the status icon appropriately whenever the source's status changes, as well as force the toggle to be off unless the data source is currently available

Declaration

```
void OnStatusChange(DataSourceReader.DataSourceReaderStatus status)
```

Parameters

TYPE	NAME	DESCRIPTION
DataSourceReader.DataSourceReaderStatus	status	The new status to represent

Select(Boolean)

Handles the toggle changing state by making the active data source this one or null

Declaration

```
void Select(bool isOn)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	isOn	Whether or not the toggle's state is on

Struct SensorData

The data for a single sensor inside of a data frame

Namespace: [Optispeech.Data](#)

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public struct SensorData
```

Fields

id

The ID of the sensor this data is for

Declaration

```
public int id
```

Field Value

TYPE	DESCRIPTION
System.Int32	

position

The recorded position for this sensor this frame

Declaration

```
public Vector3 position
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

postOffset

The offset configured by the user. This gets stored separately from the position because the position may be changed while processing the data frame, and the post offset should be applied after all that

Declaration

```
public Vector3 postOffset
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

rotation

The recorded rotation for this sensor this frame

Declaration

```
public Quaternion rotation
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Quaternion	

status

The status of this sensor in this frame

Declaration

```
public SensorStatus status
```

Field Value

TYPE	DESCRIPTION
SensorStatus	

Enum SensorStatus

Different data sources may have different ways of reporting sensor statuses, so we map them as best as we can to these different status categories (based on the WaveFront statuses). With these we can display sensor statuses to the user without any data source-specific code.

Namespace: [Optispeech.Data](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public enum SensorStatus
```

Fields

NAME	DESCRIPTION
BAD_FIT	
OK	
OUT_OF_VOLUME	
PROCESSING_ERROR	
UNKNOWN	

Class TcpClientController

Multiple data source readers may need to setup a TCP connection with very similar settings, so this class can be used as a utility to set them all up appropriately

Inheritance

System.Object

TcpClientController

Namespace: [Optispeech.Data](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class TcpClientController
```

Fields

client

The internal TCP client from .NET this utility class is a wrapper for

Declaration

```
public TcpClient client
```

Field Value

TYPE	DESCRIPTION
System.Net.Sockets.TcpClient	

connectTask

An asynchronous task to connect to the TCP host

Declaration

```
Task connectTask
```

Field Value

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

flipEndian

Some data sources may use the opposite endianness as this program expects, so this flag will determine whether or not all incoming numbers should be flipped before being parsed

Declaration

```
public bool flipEndian
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

onFail

A callback event that fires whenever the connection fails

Declaration

```
public UnityEvent onFail
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent	

onSuccess

A callback event that fires whenever the connection succeeds

Declaration

```
public UnityEvent onSuccess
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent	

stream

The network stream created by the TCP client through which this program can communicate with the data source

Declaration

```
public NetworkStream stream
```

Field Value

TYPE	DESCRIPTION
System.Net.Sockets.NetworkStream	

timeoutCancellationTokenSource

A timeout token used to cancel [connectTask](#) if the connection takes too long. (This is most likely to occur if the host the connection is trying to reach is not available)

Declaration

```
CancellationTokenSource timeoutCancellationTokenSource
```

Field Value

TYPE	DESCRIPTION
System.Threading.CancellationTokenSource	

Methods

Connect(Int32, Int32, String)

Attempts to asynchronously connect to the TCP host at the specified location, and will call either [onSuccess](#) or [onFail](#) as

appropriate

Declaration

```
public void Connect(int port, int connectTimeout = 1000, string host = "127.0.0.1")
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	port	The port to connect to
System.Int32	connectTimeout	How long, in ms, to wait before assuming the host is not available
System.String	host	The host to connect to, defaulting to the localhost

ReadInt32()

Reads the next 32-bit int from the network stream

Declaration

```
public int ReadInt32()
```

Returns

TYPE	DESCRIPTION
System.Int32	The parsed value

ReadInt64()

Reads the next 64-bit int from the network stream

Declaration

```
public long ReadInt64()
```

Returns

TYPE	DESCRIPTION
System.Int64	The parsed value

ReadSingle()

Reads the next 32-bit float ("single") from the network stream

Declaration

```
public float ReadSingle()
```

Returns

TYPE	DESCRIPTION
System.Single	The parsed value

Struct TransformedData

Struct to hold the transformed data for a given data frame. Stores specific sensors by role, and includes the transformation matrix used to map the raw sensors to their transformed versions

Namespace: [Optispeech.Data](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public struct TransformedData
```

Fields

forehead

The sensor data for the first sensor configured as the forehead, if any

Declaration

```
public SensorData? forehead
```

Field Value

TYPE	DESCRIPTION
System.Nullable< SensorData >	

jaw

The sensor data for the first sensor configured as the jaw, if any

Declaration

```
public SensorData? jaw
```

Field Value

TYPE	DESCRIPTION
System.Nullable< SensorData >	

leftEar

The sensor data for the first sensor configured as the left ear, if any

Declaration

```
public SensorData? leftEar
```

Field Value

TYPE	DESCRIPTION
System.Nullable< SensorData >	

otherSensors

The sensor data for all sensors configured as "Other"

Declaration


```
public SensorData[] otherSensors
```

Field Value

TYPE	DESCRIPTION
SensorData[]	

rightEar

The sensor data for the first sensor configured as the right ear, if any

Declaration

```
public SensorData? rightEar
```

Field Value

TYPE	DESCRIPTION
System.Nullable<SensorData>	

tongueBack

The sensor data for the first sensor configured as the back of the tongue, if any

Declaration

```
public SensorData? tongueBack
```

Field Value

TYPE	DESCRIPTION
System.Nullable<SensorData>	

tongueDorsum

The sensor data for the first sensor configured as the center of the tongue, if any

Declaration

```
public SensorData? tongueDorsum
```

Field Value

TYPE	DESCRIPTION
System.Nullable<SensorData>	

tongueLeft

The sensor data for the first sensor configured as the left of the tongue, if any

Declaration

```
public SensorData? tongueLeft
```

Field Value

TYPE	DESCRIPTION
System.Nullable< SensorData >	

tongueRight

The sensor data for the first sensor configured as the right of the tongue, if any

Declaration

```
public SensorData? tongueRight
```

Field Value

TYPE	DESCRIPTION
System.Nullable< SensorData >	

tongueTip

The sensor data for the first sensor configured as the tip of the tongue, if any

Declaration

```
public SensorData? tongueTip
```

Field Value

TYPE	DESCRIPTION
System.Nullable< SensorData >	

transformMatrix

The transformation matrix used to map this frame's raw sensor data to fit onto the avatar model

Declaration

```
public Matrix4x4 transformMatrix
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Matrix4x4	

Namespace Optispeech.Data.FileReaders

Classes

[FrameReader](#)

This class reads sweep data from a file

[FrameReader.FileReader](#)

Abstract class for reading sweep data in different file formats

[LegacyFileReader](#)

This file reader reads tsv files from the previous iteration of OptiSpeech

[OptiSpeechFileReader](#)

This file reader reads XXXX_raw.tsv files from OptiSpeech 2

[WaveFrontFileReader](#)

This file reader reads tsv files from NDI WaveFront

Delegates

[FrameReader.ReadFrameFromString](#)

Delegate used for creating utility functions for common patterns across FrameReaders

Class FrameReader

This class reads sweep data from a file

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
FrameReader

Namespace: [Optispeech.Data.FileReaders](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class FrameReader : MonoBehaviour
```

Remarks

While originally written for the FileDataSource this code can also be used to read sweep data elsewhere, such as for the custom motion target type

Fields

Instance

Static member to access the singleton instance of this class

Declaration

```
public static FrameReader Instance
```

Field Value

TYPE	DESCRIPTION
FrameReader	

Methods

GetFileReader(StreamReader, Boolean)

Takes a file and determines which kind of file it has and returns the appropriate [FrameReader.FileReader](#) implementation

Declaration

```
FrameReader.FileReader GetFileReader(StreamReader file, bool loadTargets)
```

Parameters

TYPE	NAME	DESCRIPTION
System.IO.StreamReader	file	The file to be read
System.Boolean	loadTargets	Whether or not to add any targets present in the file

Returns

TYPE	DESCRIPTION
FrameReader.FileReader	The appropriate FileReader implementation for this type of file, or null if none found

HandleFailure(String, ref Int32)

Utility function for handling errors reading from the file this will tick the number of failures and return a bool over whether to continue processing that file

Declaration

```
public static bool HandleFailure(string message, ref int numFailures)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	The error message to log
System.Int32	numFailures	The number of failures this file has

Returns

TYPE	DESCRIPTION
System.Boolean	Whether or not to cancel reading the file

ReadFile(String, Boolean, UnityAction<FrameReader.FileReader, List<DataFrame>>, UnityAction)

Creates a coroutine that reads the specific file and then calls the appropriate callback. This function will determine what kind of file it is and attempt to read it.

Declaration

```
public IEnumerator ReadFile(string filename, bool loadTargets = true, UnityAction<FrameReader.FileReader, List<DataFrame>> onSuccess = null, UnityAction onFailure = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	filename	The filename of the file to load
System.Boolean	loadTargets	Whether or not to load any targets present in the file
UnityEngine.Events.UnityAction< FrameReader.FileReader , System.Collections.Generic.List< DataFrame >>	onSuccess	Callback that's called when the file is successfully read, with the FileReader used and the frames read

TYPE	NAME	DESCRIPTION
UnityEngine.Events.UnityAction	onFailure	Callback that's called when the file is not successfully read

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	Coroutine

ReadSingleLineFrames(StreamReader, UnityAction<DataFrame>, FrameReader.ReadFrameFromString)

Utility function for frame readers that want to read a single data frame from each line until the end of the file

Declaration

```
public static bool ReadSingleLineFrames(StreamReader file, UnityAction<DataFrame> addDataFrame,
FrameReader.ReadFrameFromString readFrame)
```

Parameters

TYPE	NAME	DESCRIPTION
System.IO.StreamReader	file	The file to read
UnityEngine.Events.UnityAction<DataFrame>	addDataFrame	The addDataFrame delegate given to the frame reader
FrameReader.ReadFrameFromString	readFrame	Delegate to handle reading a data frame from a single line of the file

Returns

TYPE	DESCRIPTION
System.Boolean	Whether or not the file was successfully read

Class FrameReader.FileReader

Abstract class for reading sweep data in different file formats

Inheritance

- System.Object
- FrameReader.FileReader
- LegacyFileReader
- OptiSpeechFileReader
- WaveFrontFileReader

Namespace: [Optispeech.Data.FileReaders](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public abstract class FileReader
```

Methods

GetAudioFile(String)

Takes a filename and returns a file that will hold the corresponding audio for this file, if it exists Note that it doesn't need to check if the file exists, just return the filename it would be at if it does exist

Declaration

```
public virtual string GetAudioFile(string filename)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	filename	

Returns

TYPE	DESCRIPTION
System.String	

GetSensorConfigurations(DataFrame)

When reading files, the sensor configurations are generally stored in the file, or can be assumed based on the file format. This function should return the sensor configurations present in the passed data frame

Declaration

```
public abstract SensorConfiguration[] GetSensorConfigurations(DataFrame dataFrame)
```

Parameters

TYPE	NAME	DESCRIPTION
DataFrame	dataFrame	The data frame to find the sensor configurations for

Returns

TYPE	DESCRIPTION
SensorConfiguration []	The sensor configurations present or assumed from the passed data frame

ReadFrames(StreamReader, UnityAction<DataFrame>, UnityAction<Boolean>)

This function will read a file and call a delegate on every data frame parsed from the file, and eventually call the finish delegate when done working, returning a bool representing whether or not the file was successfully read

Declaration

```
public abstract void ReadFrames(StreamReader file, UnityAction<DataFrame> addDataFrame, UnityAction<bool> finish)
```

Parameters

TYPE	NAME	DESCRIPTION
System.IO.StreamReader	file	The file to read data frames from
UnityEngine.Events.UnityAction< DataFrame >	addDataFrame	A delegate to call with any data frames read from the file
UnityEngine.Events.UnityAction<System.Boolean>	finish	A delegate to call when done working, returning true if successful

Delegate FrameReader.ReadFrameFromString

Delegate used for creating utility functions for common patterns across FrameReaders

Namespace: [Optispeech.Data.FileReaders](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate DataFrame ReadFrameFromString(string frame);
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	frame	A string to read a data frame from

Returns

TYPE	DESCRIPTION
DataFrame	The data frame containing the data from <code>frame</code>

Class LegacyFileReader

This file reader reads tsv files from the previous iteration of OptiSpeech

Inheritance

System.Object
FrameReader.FileReader
LegacyFileReader

Inherited Members

FrameReader.FileReader.ReadFrames(StreamReader, UnityAction<DataFrame>, UnityAction<Boolean>)
FrameReader.FileReader.GetSensorConfigurations(DataFrame)
FrameReader.FileReader.GetAudioFile(String)

Namespace: [Optispeech.Data.FileReaders](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class LegacyFileReader : FrameReader.FileReader
```

Constructors

LegacyFileReader(String, Boolean)

When creating the file reader it takes a header row to determine which types of targets are in the file, and wherethey are in each frame of data

Declaration

```
public LegacyFileReader(string headerRow, bool loadTargets)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	headerRow	The first line of the file to read
System.Boolean	loadTargets	Whether or not to add the targets in the file

Remarks

Note that there are various different versions of the legacy optispeech program, and if more target types are found they'll need to be handled in here as well

Fields

firstTargetIndex

When parsing the header row, this is the index of the first target

Declaration

```
int firstTargetIndex
```

Field Value

TYPE	DESCRIPTION
System.Int32	

landmarkTargetIndex

When parsing the header row, this is the index of the landmark target

Declaration

int landmarkTargetIndex

Field Value

TYPE	DESCRIPTION
System.Int32	

loadTargets

Whether to add targets from the file or not

Declaration

bool loadTargets

Field Value

TYPE	DESCRIPTION
System.Boolean	

oscillatingTargetIndex

When parsing the header row, this is the index of the oscillating target

Declaration

int oscillatingTargetIndex

Field Value

TYPE	DESCRIPTION
System.Int32	

staticTargetType

Cached value of the static target type, which we use to display the different targets.

Declaration

TargetDescription staticTargetType

Field Value

TYPE	DESCRIPTION
TargetDescription	

Class OptiSpeechFileReader

This file reader reads XXXX_raw.tsv files from OptiSpeech 2

Inheritance

System.Object
FrameReader.FileReader
OptiSpeechFileReader

Inherited Members

FrameReader.FileReader.ReadFrames(StreamReader, UnityAction<DataFrame>, UnityAction<Boolean>)
FrameReader.FileReader.GetSensorConfigurations(DataFrame)
FrameReader.FileReader.GetAudioFile(String)

Namespace: [Optispeech.Data.FileReaders](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class OptiSpeechFileReader : FrameReader.FileReader
```

Constructors

OptiSpeechFileReader(StreamReader, Boolean)

When creating this file reader the stream reader is passed so the entire header can be parsed

Declaration

```
public OptiSpeechFileReader(StreamReader file, bool loadTargets)
```

Parameters

TYPE	NAME	DESCRIPTION
System.IO.StreamReader	file	The file to be read
System.Boolean	loadTargets	Whether or not to add targets in the file

Fields

audioFile

The filename of the audio file recorded with this sweep, if it exists

Declaration

```
string audioFile
```

Field Value

TYPE	DESCRIPTION
System.String	

sensorConfigurations

The sensor configurations stored in the file

Declaration

SensorConfiguration[] sensorConfigurations

Field Value

TYPE	DESCRIPTION
SensorConfiguration[]	

Class WaveFrontFileReader

This file reader reads tsv files from NDI WaveFront

Inheritance

System.Object

[FrameReader.FileReader](#)

WaveFrontFileReader

Inherited Members

[FrameReader.FileReader.ReadFrames\(StreamReader, UnityAction<DataFrame>, UnityAction<Boolean>\)](#)

[FrameReader.FileReader.GetSensorConfigurations\(DataFrame\)](#)

[FrameReader.FileReader.GetAudioFile\(String\)](#)

Namespace: [Optispeech.Data.FileReaders](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class WaveFrontFileReader : FrameReader.FileReader
```

Namespace Optispeech.Data.Sources

Classes

[FileDataSource](#)

A data source reader that reads from a local file. This reader will read a file in a variety of different formats, load the whole file, and then allow the user to playback that data, including a panel where they can pause or jump around the file

[WaveFrontDataSource](#)

A data source reader that reads from the WaveFront real-time API. Written to specification from the Real-Time API section of the [WaveFront manual revision 7](#).

Class FileDataSource

A data source reader that reads from a local file. This reader will read a file in a variety of different formats, load the whole file, and then allow the user to playback that data, including a panel where they can pause or jump around the file

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- [DataSourceReader](#)
- FileDataSource

Inherited Members

- [DataSourceReader.SetupStopwatch\(\)](#)
- [DataSourceReader.statusChangeEvent](#)
- [DataSourceReader.dataQueue](#)
- [DataSourceReader.lastFrame](#)
- [DataSourceReader.isActive](#)
- [DataSourceReader.stopwatch](#)
- [DataSourceReader.OnEnable\(\)](#)
- [DataSourceReader.GetCurrentStatus\(\)](#)
- [DataSourceReader.IsTimestampProvided\(\)](#)
- [DataSourceReader.AreTargetsConfigurable\(\)](#)
- [DataSourceReader.AreSensorsConfigurable\(\)](#)
- [DataSourceReader.GetDefaultSensorConfigurations\(\)](#)
- [DataSourceReader.ReadFrame\(\)](#)
- [DataSourceReader.StartThread\(\)](#)
- [DataSourceReader.Cleanup\(\)](#)
- [DataSourceReader.StartSweep\(String, String\)](#)
- [DataSourceReader.StopSweep\(\)](#)
- [DataSourceReader.Run\(\)](#)

Namespace: [Optispeech.Data.Sources](#)

Assembly: Assembly-CSharp.dll

Syntax

```
[RequireComponent(typeof(AudioSource))]  
public class FileDataSource : DataSourceReader
```

Fields

audio

An audio source used to playback any loaded audio files

Declaration

```
AudioSource audio
```

Field Value

TYPE	DESCRIPTION
UnityEngine.AudioSource	

durationLabel

A label used to show how long the loaded file is

Declaration

```
[SerializeField]
TextMeshProUGUI durationLabel
```

Field Value

TYPE	DESCRIPTION
TMPro.TextMeshProUGUI	

frames

The frames read from the file

Declaration

```
List<DataFrame> frames
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List< DataFrame >	

loadFile

A button to open the file dialog to choose a file to load

Declaration

```
[SerializeField]
Button loadFile
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

loadFileLabel

The label on the load file button, used to change its text when its currently loading a file

Declaration

```
[SerializeField]
TextMeshProUGUI loadFileLabel
```

Field Value

TYPE	DESCRIPTION
TMPro.TextMeshProUGUI	

nextFrame

The index in [frames](#) of the next data frame

Declaration

```
int nextFrame
```

Field Value

TYPE	DESCRIPTION
System.Int32	

paused

Whether or not the playback is currently paused

Declaration

```
bool paused
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

pauseSprite

The sprite to show on the [togglePauseButton](#) when the button will pause the playback

Declaration

```
[SerializeField]  
Sprite pauseSprite
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

progressSlider

A progress slider used to show how far along the file the current playback is, which can be changed by the user to move forward or back along the file

Declaration

```
[SerializeField]  
Slider progressSlider
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Slider	

reader

The file reader used to read the currently selected file

Declaration

```
FrameReader.FileReader reader
```

Field Value

TYPE	DESCRIPTION
FrameReader.FileReader	

resumeSprite

The sprite to show on the [togglePauseButton](#) when the button will resume the playback

Declaration

```
[SerializeField]  
Sprite resumeSprite
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

togglePauseButton

The button used to toggle whether the file is being played back at the moment

Declaration

```
[SerializeField]  
Button togglePauseButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

togglePauseImage

The image on [togglePauseButton](#), used to change what sprite its using based on the state of [paused](#)

Declaration

```
Image togglePauseImage
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Image	

Methods

ReadFile()

Opens a file dialog and reads the selected file

Declaration

```
void ReadFile()
```

Reset()

Resets the loaded file so a new one can be loaded

Declaration

```
void Reset()
```

SetTime(Single)

Sets how far along the file playback is

Declaration

```
void SetTime(float value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	value	The time in ms to change playback to

StartPlayback(String, FrameReader.FileReader, List<DataFrame>)

Starts playing back a file. Returns a coroutine so it can attempt to load the audio file for this data file

Declaration

```
IEnumerator StartPlayback(string filename, FrameReader.FileReader reader, List<DataFrame> frames)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	filename	The location of the file to play back
FrameReader.FileReader	reader	The reader used to read the file
System.Collections.Generic.List< DataFrame >	frames	The data frames read from the file

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	A coroutine that will attempt to load the audio file for this file

TogglePause()

Toggles whether or not ReadFrame is currently stalling or returning frames

Declaration

```
void TogglePause()
```

Class WaveFrontDataSource

A data source reader that reads from the WaveFront real-time API. Written to specification from the Real-Time API section of the [WaveFront manual revision 7](#).

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- [DataSourceReader](#)
- WaveFrontDataSource

Inherited Members

- [DataSourceReader.SetupStopwatch\(\)](#)
- [DataSourceReader.statusChangeEvent](#)
- [DataSourceReader.dataQueue](#)
- [DataSourceReader.lastFrame](#)
- [DataSourceReader.isActive](#)
- [DataSourceReader.stopwatch](#)
- [DataSourceReader.OnEnable\(\)](#)
- [DataSourceReader.GetCurrentStatus\(\)](#)
- [DataSourceReader.IsTimestampProvided\(\)](#)
- [DataSourceReader.AreTargetsConfigurable\(\)](#)
- [DataSourceReader.AreSensorsConfigurable\(\)](#)
- [DataSourceReader.GetDefaultSensorConfigurations\(\)](#)
- [DataSourceReader.ReadFrame\(\)](#)
- [DataSourceReader.StartThread\(\)](#)
- [DataSourceReader.Cleanup\(\)](#)
- [DataSourceReader.StartSweep\(String, String\)](#)
- [DataSourceReader.StopSweep\(\)](#)
- [DataSourceReader.Run\(\)](#)

Namespace: [Optispeech.Data.Sources](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class WaveFrontDataSource : DataSourceReader
```

Fields

clientController

A client controller used to communicate with the WaveFront realtime API. This is static so that when this data source is instantiated, it shares the same controller that originally made the connection from the prefab

Declaration

```
static TcpClientController clientController
```

Field Value

TYPE	DESCRIPTION
TcpClientController	

connectTimeout

How long to wait to connect to WaveFront before giving up and assuming WaveFront just isn't available

Declaration

```
[SerializeField]
int connectTimeout
```

Field Value

TYPE	DESCRIPTION
System.Int32	

dataFrameQueue

This data source gets frames from WaveFront in realtime, which may not be 1:1 with rendered frames in OptiSpeech. To handle this, any read frames are put into this queue and ReadFrame will pass the next one, or wait if the queue is empty.

Declaration

```
Queue<DataFrame> dataFrameQueue
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.Queue< DataFrame >	

host

The host to try to connect to

Declaration

```
static string host
```

Field Value

TYPE	DESCRIPTION
System.String	

mainDataSource

The [WaveFrontDataSource](#) instance on the first [DataSourceDescription](#) with a [WaveFrontDataSource](#) on it's [readerPrefab](#)

Declaration

```
static WaveFrontDataSource mainDataSource
```

Field Value

TYPE	DESCRIPTION
WaveFrontDataSource	

port

The port to try to connect to. The spec says this should always be 3030, but is configurable just in case of things like proxies or

anything else that could cause a problem

Declaration

```
static int port
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

Host

Property that provides public access to [host](#) that saves to PlayerPrefs on write

Declaration

```
public static string Host { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Port

Property that provides public access to [port](#) that saves to PlayerPrefs on write

Declaration

```
public static int Port { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

LoadAdvancedSettings()

Loads host and port values from PlayerPrefs when the scene is loaded

Declaration

```
[RuntimeInitializeOnLoadMethod(RuntimeInitializeLoadType.BeforeSceneLoad)]  
static void LoadAdvancedSettings()
```

ReadPacket()

This handles each packet received from the WaveFront realtime API

Declaration

```
void ReadPacket()
```

SendCommand(String)

Sends a message to the WaveFront realtime API

Declaration

```
void SendCommand(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	The message to send

Namespace Optispeech.Display

Classes

[CameraController](#)

Sets up the camera to be controlled by the user when holding down right click, noclip-style. Modified from [FlyCam Extended](#)

[DisplayPanel](#)

A panel controller that handles various visibility settings

[SetupCameraPanel](#)

This sets up a camera panel to include buttons to move the camera to preset positions and rotations. Each child transform of this gameObject will be a different preset

Class CameraController

Sets up the camera to be controlled by the user when holding down right click, noclip-style. Modified from [FlyCam Extended](#)

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
CameraController

Namespace: [Optispeech.Display](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class CameraController : MonoBehaviour
```

Fields

cameraSensitivity

The amount the camera will rotate as the mouse moves

Declaration

```
[SerializeField]  
float cameraSensitivity
```

Field Value

TYPE	DESCRIPTION
System.Single	

climbSpeed

How quickly the camera moves vertically

Declaration

```
[SerializeField]  
float climbSpeed
```

Field Value

TYPE	DESCRIPTION
System.Single	

fastMoveFactor

How quickly the camera moves when using WASD while holding shift

Declaration

```
[SerializeField]  
float fastMoveFactor
```

Field Value

TYPE	DESCRIPTION
System.Single	

isMoving

Whether or not we're currently moving, used to save the camera position and rotation to the active profile once the user stops moving

Declaration

```
bool isMoving
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

normalMoveSpeed

How quickly the camera moves when using WASD while not holding shift

Declaration

```
[SerializeField]  
float normalMoveSpeed
```

Field Value

TYPE	DESCRIPTION
System.Single	

slowMoveFactor

How quickly the camera moves when using WASD while holding ctrl

Declaration

```
[SerializeField]  
float slowMoveFactor
```

Field Value

TYPE	DESCRIPTION
System.Single	

Methods

LoadCameraSettings(ProfileManager.Profile)

Loads the camera position and rotation from the given profile

Declaration

```
void LoadCameraSettings(ProfileManager.Profile profile)
```

Parameters

TYPE	NAME	DESCRIPTION
ProfileManager.Profile	profile	The profile to apply the camera position and rotation from

Class DisplayPanel

A panel controller that handles various visibility settings

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- DisplayPanel

Namespace: [Optispeech.Display](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class DisplayPanel : MonoBehaviour
```

Fields

gumsMaterial

The material used on the gums part of the avatar

Declaration

```
[SerializeField]  
Material gumsMaterial
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Material	

gumsTransparencySlider

Slider for controlling the transparency of [gumsMaterial](#)

Declaration

```
[SerializeField]  
Slider gumsTransparencySlider
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Slider	

headMaterial

The material used on the head part of the avatar

Declaration

```
[SerializeField]  
Material headMaterial
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Material	

headTransparencySlider

Slider for controlling the transparency of [headMaterial](#)

Declaration

[SerializeField] Slider headTransparencySlider

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Slider	

markersParent

The container all reference markers are in

Declaration

[SerializeField] GameObject markersParent
--

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

markersVisibleToggle

Toggle for controlling the visibility of [markersParent](#)

Declaration

[SerializeField] Toggle markersVisibleToggle

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

sensorsParent

The container all sensor markers are in

Declaration

[SerializeField] GameObject sensorsParent
--

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

sensorsVisibleToggle

Toggle for controlling the visibility of [sensorsParent](#)

Declaration

[SerializeField] Toggle sensorsVisibleToggle

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

tongueParent

The container the tongue is in

Declaration

[SerializeField] GameObject tongueParent

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

tongueVisibleToggle

Toggle for controlling the visibility of [tongueParent](#)

Declaration

[SerializeField] Toggle tongueVisibleToggle
--

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

Class SetupCameraPanel

This sets up a camera panel to include buttons to move the camera to preset positions and rotations. Each child transform of this gameObject will be a different preset

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- SetupCameraPanel

Namespace: [Optispeech.Display](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class SetupCameraPanel : MonoBehaviour
```

Fields

buttonPrefab

Button prefab to create each preset button. Must have a UnityEngine.UI.Button and TMPro.TextMeshProUGUI component somewhere in the prefab (they can be in children)

Declaration

```
[SerializeField]  
GameObject buttonPrefab
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

cameraPanelContent

Reference to the content container of the camera panel, to add the preset buttons to

Declaration

```
[SerializeField]  
RectTransform cameraPanelContent
```

Field Value

TYPE	DESCRIPTION
UnityEngine.RectTransform	

Namespace Optispeech.Documentation

Classes

[DocumentationBuilder](#)

Automatically checks if docfx is installed when building the project and, if so, uses it to build the documentation and places it inside the output folder so it can be hosted by the [DocumentationServer](#)

[DocumentationServer](#)

Creates a web server and hosts the documentation website on it while the program is running. This server will, assuming the ports aren't opened, only be accessible on this computer. This allows the documentation to be a website the user can see in the browser, without the need to make it public to the world wide web.

[DocumentationServer.SimpleHTTPServer](#)

A simple http server implementation from [this Unity answer](#)

[HideInDocumentation](#)

This attribute is used to hide a certain member or method from appearing in the documentation generated by docfx. Intended use is for private methods that don't need documentation. Since our scripting API is targeted at developers who are going to be working on the source itself, as opposed to writing a script that just interfaces with this program, it made sense to allow docfx to generate documentation for private members (especially if they're decorated with the `SerializeFieldAttribute`!) However, many private methods like the ones called by messages from UnityEngine itself (Start, Update, ...) have a very straightforward purpose that doesn't really change much object to object. Instead of adding a comment to every Start method about how its called by UnityEngine at the start of the game or when the object is added to the scene, and how its being used to setup anything that needs setting up, it makes sense to assume if they're working on a Unity project like this they already know that and don't need a reminder on every MonoBehaviour. Additionally, if the documentation were to describe exactly what is being set up, the documentation would just be a redundancy of reading the actual code itself, and would likely not be maintained. Individual Start methods can still remain documented by just not decorating it with this attribute, if there are special circumstances that make it worth documenting something about that specific Start method. This should be used sparingly.

Class DocumentationBuilder

Automatically checks if docfx is installed when building the project and, if so, uses it to build the documentation and places it inside the output folder so it can be hosted by the [DocumentationServer](#)

Inheritance

System.Object

DocumentationBuilder

Namespace: [Optispeech.Documentation](#)

Assembly: Assembly-CSharp-Editor.dll

Syntax

```
public class DocumentationBuilder : IPostprocessBuildWithReport, IOrderedCallback
```

Methods

OnPostprocessBuild(BuildReport)

Callback that runs when the project finishes building, at which point it attempts to build the documentation and place it in the output folder

Declaration

```
public void OnPostprocessBuild(BuildReport report)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEditor.Build.Reporting.BuildReport	report	

Class DocumentationServer

Creates a web server and hosts the documentation website on it while the program is running. This server will, assuming the ports aren't opened, only be accessible on this computer. This allows the documentation to be a website the user can see in the browser, without the need to make it public to the world wide web.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
DocumentationServer

Namespace: [Optispeech.Documentation](#)
Assembly: Assembly-CSharp.dll

Syntax

```
public class DocumentationServer : MonoBehaviour
```

Fields

Instance

Static member to access the singleton instance of this class

Declaration

```
public static DocumentationServer Instance
```

Field Value

TYPE	DESCRIPTION
DocumentationServer	

myServer

The server the documentation is being hosted on

Declaration

```
DocumentationServer.SimpleHTTPServer myServer
```

Field Value

TYPE	DESCRIPTION
DocumentationServer.SimpleHTTPServer	

Class DocumentationServer.SimpleHTTPServer

A simple http server implementation from [this Unity answer](#)

Inheritance

System.Object

DocumentationServer.SimpleHTTPServer

Namespace: [Optispeech.Documentation](#)

Assembly: Assembly-CSharp.dll

Syntax

```
class SimpleHTTPServer
```

Constructors

SimpleHTTPServer(String)

Construct server with suitable port.

Declaration

```
public SimpleHTTPServer(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	path	Directory path to serve.

SimpleHTTPServer(String, Int32)

Construct server with given port.

Declaration

```
public SimpleHTTPServer(string path, int port)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	path	Directory path to serve.
System.Int32	port	Port of the server.

Fields

_indexFiles

List of files that represent an "index" page

Declaration

```
readonly string[] _indexFiles
```

Field Value

TYPE	DESCRIPTION
System.String[]	

_listener

The listener for all http messages to the server

Declaration

HttpListener _listener

Field Value

TYPE	DESCRIPTION
System.Net.HttpListener	

_mimeTypeMappings

Dictionary of mime types for handling various file types

Declaration

static IDictionary<string, string> _mimeTypeMappings
--

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.IDictionary<System.String, System.String>	

_port

The port the server is running on

Declaration

int _port

Field Value

TYPE	DESCRIPTION
System.Int32	

_rootDirectory

The root directory that is being served

Declaration

string _rootDirectory

Field Value

TYPE	DESCRIPTION
System.String	

_serverThread

The thread the http server is running on

Declaration

```
Thread _serverThread
```

Field Value

TYPE	DESCRIPTION
System.Threading.Thread	

Properties

Port

Property that allows public read-only access to the port the server is running on

Declaration

```
public int Port { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

Initialize(String, Int32)

Creates thread to listen for http requests on the provided port, and serves pages found in the given path

Declaration

```
void Initialize(string path, int port)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	path	The directory of files to serve
System.Int32	port	The port to listen for http requests on

Listen()

Start http listener and continuously process requests

Declaration

```
void Listen()
```

Process(HttpListenerContext)

Processes a given http request

Declaration

```
void Process(HttpListenerContext context)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.HttpListenerContext	context	The http request to process

Stop()

Stop server and dispose all functions.

Declaration

```
public void Stop()
```

Class HideInDocumentation

This attribute is used to hide a certain member or method from appearing in the documentation generated by docfx. Intended use is for private methods that don't need documentation. Since our scripting API is targeted at developers who are going to be working on the source itself, as opposed to writing a script that just interfaces with this program, it made sense to allow docfx to generate documentation for private members (especially if they're decorated with the `SerializeFieldAttribute`!) However, many private methods like the ones called by messages from UnityEngine itself (Start, Update, ...) have a very straightforward purpose that doesn't really change much object to object. Instead of adding a comment to every Start method about how its called by UnityEngine at the start of the game or when the object is added to the scene, and how its being used to setup anything that needs setting up, it makes sense to assume if they're working on a Unity project like this they already know that and don't need a reminder on every MonoBehaviour. Additionally, if the documentation were to describe exactly what is being set up, the documentation would just be a redundancy of reading the actual code itself, and would likely not be maintained. Individual Start methods can still remain documented by just not decorating it with this attribute, if there are special circumstances that make it worth documenting something about that specific Start method. This should be used sparingly.

Inheritance

System.Object

System.Attribute

HideInDocumentation

Namespace: [Optispeech.Documentation](#)

Assembly: Assembly-CSharp.dll

Syntax

```
[AttributeUsage(AttributeTargets.All | AttributeTargets.Assembly | AttributeTargets.Class |
AttributeTargets.Constructor | AttributeTargets.Delegate | AttributeTargets.Enum | AttributeTargets.Event |
AttributeTargets.Field | AttributeTargets.GenericParameter | AttributeTargets.Interface |
AttributeTargets.Method | AttributeTargets.Module | AttributeTargets.Parameter | AttributeTargets.Property |
AttributeTargets.ReturnValue | AttributeTargets.Struct, Inherited = false, AllowMultiple = true)]
public class HideInDocumentation : Attribute, _Attribute
```


Namespace Optispeech.Profiles

Classes

[ProfileManager](#)

Manages the active profile, which includes all the settings in the entire program, and handles updating settings and switching profiles. Profiles are saved to PlayerPrefs.

[ProfileManager.IntEvent](#)

Create a UnityEvent that passes along an int to each listener

[ProfileManager.ProfileEvent](#)

Create a UnityEvent that passes along a Profile to each listener

[ProfilePanel](#)

Panel for creating, deleting, renaming, and switching between the various registered profiles

[ProfileSelector](#)

A UI object that represents a profile and allows it to be selected, renamed, and deleted

[SerializableVector3](#)

A serializable version of a UnityEngine.Vector3, which will implicitly convert to/from UnityEngine.Vector3. This is used for saving UnityEngine.Vector3 values in profiles

Structs

[ProfileManager.Profile](#)

A collection of settings

Delegates

[ProfileManager.ProfileBoolUpdater](#)

Delegate to make updating profiles based on input bool callbacks easier

[ProfileManager.ProfileFloatUpdater](#)

Delegate to make updating profiles based on input float callbacks easier

[ProfileManager.ProfileIntUpdater](#)

Delegate to make updating profiles based on input int callbacks easier

[ProfileManager.ProfileStringUpdater](#)

Delegate to make updating profiles based on input string callbacks easier

Class ProfileManager

Manages the active profile, which includes all the settings in the entire program, and handles updating settings and switching profiles. Profiles are saved to PlayerPrefs.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
ProfileManager

Namespace: [Optispeech.Profiles](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class ProfileManager : MonoBehaviour
```

Fields

activeProfile

The currently active profile

Declaration

```
ProfileManager.Profile activeProfile
```

Field Value

TYPE	DESCRIPTION
ProfileManager.Profile	

activeProfileIndex

The index in [profiles](#) of the active profile

Declaration

```
[HideInInspector]  
public int activeProfileIndex
```

Field Value

TYPE	DESCRIPTION
System.Int32	

initialCameraPos

The default camera position for new profiles

Declaration

```
Vector3 initialCameraPos
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

initialCameraRot

The default camera rotation for new profiles

Declaration

Vector3 initialCameraRot

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

Instance

Static member to access the singleton instance of this class

Declaration

public static ProfileManager Instance

Field Value

TYPE	DESCRIPTION
ProfileManager	

onProfileAdded

Event that fires whenever a new profile is added to the list

Declaration

[HideInInspector] public ProfileManager.ProfileEvent onProfileAdded
--

Field Value

TYPE	DESCRIPTION
ProfileManager.ProfileEvent	

onProfileChange

Event that fires whenever the active profile changes

Declaration

[HideInInspector] public ProfileManager.ProfileEvent onProfileChange

Field Value

TYPE	DESCRIPTION
ProfileManager.ProfileEvent	

onProfileDeleted

Event that fires whenever a profile is removed from the list

Declaration

```
[HideInInspector]
public ProfileManager.IntEvent onProfileDeleted
```

Field Value

TYPE	DESCRIPTION
ProfileManager.IntEvent	

profiles

List of all registered profiles

Declaration

```
[HideInInspector]
public List<ProfileManager.Profile> profiles
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List< ProfileManager.Profile >	

Properties

ActiveProfile

Property that provides public read-only access to the active profile

Declaration

```
public ProfileManager.Profile ActiveProfile { get; }
```

Property Value

TYPE	DESCRIPTION
ProfileManager.Profile	

Methods

AddProfile(ProfileManager.Profile)

Adds the given profile to the profiles list and loads it

Declaration

```
public void AddProfile(ProfileManager.Profile profile)
```

Parameters

TYPE	NAME	DESCRIPTION
ProfileManager.Profile	profile	The profile to add to profiles

CreateProfile()

Creates a new profile

Declaration

```
public static ProfileManager.Profile CreateProfile()
```

Returns

TYPE	DESCRIPTION
ProfileManager.Profile	A default profile

Remarks

Some default values aren't known at compile time, so here we create a new profile and manually set the default values appropriately

DeleteProfile(Int32)

Deletes the profile at the specified index

Declaration

```
public void DeleteProfile(int index)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	index	The index in profiles of the profile to delete

LoadProfile(Int32)

Loads a profile from the list at the specified index

Declaration

```
public void LoadProfile(int index)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	index	The index in profiles of the profile to load

Save()

Saves [profiles](#) without invoking any listeners

Declaration

```
public void Save()
```

Remarks

Rarely to be used outside this class.

UpdateProfile(ProfileManager.Profile, Int32)

Updates a profile at the given index

Declaration

```
public void UpdateProfile(ProfileManager.Profile profile, int index = -1)
```

Parameters

TYPE	NAME	DESCRIPTION
ProfileManager.Profile	profile	The new value of the profile to save
System.Int32	index	The index in profiles to save the profile to

UpdateProfileCB(ProfileManager.ProfileBoolUpdater)

Function intended to be used as a callback for input elements for conveniently updating a profile

Declaration

```
public static UnityAction<bool> UpdateProfileCB(ProfileManager.ProfileBoolUpdater updater)
```

Parameters

TYPE	NAME	DESCRIPTION
ProfileManager.ProfileBoolUpdater	updater	Delegate to update the given profile with the given value

Returns

TYPE	DESCRIPTION
UnityEngine.Events.UnityAction<System.Boolean>	Callback to give to an input element's onValueChanged event

UpdateProfileCB(ProfileManager.ProfileFloatUpdater)

Function intended to be used as a callback for input elements for conveniently updating a profile

Declaration

```
public static UnityAction<float> UpdateProfileCB(ProfileManager.ProfileFloatUpdater updater)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
ProfileManager.ProfileFloatUpdater	updater	Delegate to update the given profile with the given value

Returns

TYPE	DESCRIPTION
UnityEngine.Events.UnityAction<System.Single>	Callback to give to an input element's onValueChanged event

UpdateProfileCB(ProfileManager.ProfileIntUpdater)

Function intended to be used as a callback for input elements for conveniently updating a profile

Declaration

```
public static UnityAction<string> UpdateProfileCB(ProfileManager.ProfileIntUpdater updater)
```

Parameters

TYPE	NAME	DESCRIPTION
ProfileManager.ProfileIntUpdater	updater	Delegate to update the given profile with the given value

Returns

TYPE	DESCRIPTION
UnityEngine.Events.UnityAction<System.String>	Callback to give to an input element's onValueChanged event

UpdateProfileCB(ProfileManager.ProfileStringUpdater)

Function intended to be used as a callback for input elements for conveniently updating a profile

Declaration

```
public static UnityAction<string> UpdateProfileCB(ProfileManager.ProfileStringUpdater updater)
```

Parameters

TYPE	NAME	DESCRIPTION
ProfileManager.ProfileStringUpdater	updater	Delegate to update the given profile with the given value

Returns

TYPE	DESCRIPTION
UnityEngine.Events.UnityAction<System.String>	Callback to give to an input element's onValueChanged event

Class ProfileManager.IntEvent

Create a UnityEvent that passes along an int to each listener

Inheritance

System.Object

UnityEngine.Events.UnityEventBase

UnityEngine.Events.UnityEvent<System.Int32>

ProfileManager.IntEvent

Namespace: [Optispeech.Profiles](#)

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]  
public class IntEvent : UnityEvent<int>, ISerializationCallbackReceiver
```


Struct ProfileManager.Profile

A collection of settings

Namespace: [Optispeech.Profiles](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public struct Profile
```

Remarks

structs don't have default values so we add tons of attributes to set the default for each parameter as well as tell our json library to handle default values by "IgnoreAndPopulate", which means it'll use the default value when that property is missing from the string, which means we can use an empty object as the default profile when reading from PlayerPrefs. Additionally it'll ignore any values to serialize that are already the default, which will lead to shorter strings saved to PlayerPrefs. Some default values aren't known at compile time and are set manually

Fields

autoStopDuration

When to stop a sweep automatically (or -1 to never stop automatically), defaulting to -1

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public int autoStopDuration
```

Field Value

TYPE	DESCRIPTION
System.Int32	

cameraPos

The current position of the camera, defaulting to [initialCameraPos](#)

Declaration

```
public SerializableVector3 cameraPos
```

Field Value

TYPE	DESCRIPTION
SerializableVector3	

cameraRot

The current rotation of the camera, defaulting to [initialCameraRot](#)

Declaration

```
public SerializableVector3 cameraRot
```

Field Value

TYPE	DESCRIPTION
SerializableVector3	

gumsTransparency

The transparency of the gums material, defaulting to 0

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public float gumsTransparency
```

Field Value

TYPE	DESCRIPTION
System.Single	

headTransparency

The transparency of the head material, defaulting to 0

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public float headTransparency
```

Field Value

TYPE	DESCRIPTION
System.Single	

lpfStrength

The strength of the low pass filter, defaulting to the default value in [defaultStrength](#)

Declaration

```
public int lpfStrength
```

Field Value

TYPE	DESCRIPTION
System.Int32	

markersVisible

The visibility of the reference markers, defaulting to invisible

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public bool markersVisible
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

profileName

The name of this profile, defaulting to "Default"

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public string profileName
```

Field Value

TYPE	DESCRIPTION
System.String	

saveAudio

Whether or not to save audio data from sweeps, defaulting to true

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public bool saveAudio
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

saveRaw

Whether or not to save raw data from sweeps, defaulting to true

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public bool saveRaw
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

saveTransformed

Whether or not to save transformed data from sweeps, defaulting to true

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public bool saveTransformed
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

saveTransformedWithoutOffsets

Whether or not to save transformed data but ignoring offsets from sweeps, defaulting to false

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public bool saveTransformedWithoutOffsets
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

sensors

A dictionary of sensor IDs to config settings, defaulting to an empty dictionary

Declaration

```
public Dictionary<string, string> sensors
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, System.String>	

sensorsVisible

The visibility of the sensor markers, defaulting to visible

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public bool sensorsVisible
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

sweepFolder

The folder to save sweep data in, defaulting to the "My Documents" folder

Declaration

```
public string sweepFolder
```

Field Value

TYPE	DESCRIPTION
System.String	

sweepName

The name of the current sweep, defaulting to "sweep"

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public string sweepName
```

Field Value

TYPE	DESCRIPTION
System.String	

targets

A dictionary of target IDs to config settings, defaulting to an empty dictionary

Declaration

```
public Dictionary<string, string> targets
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, System.String>	

tongueVisible

The visibility of the tongue, defaulting to visible

Declaration

```
[JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore | DefaultValueHandling.Populate |
DefaultValueHandling.IgnoreAndPopulate)]
public bool tongueVisible
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Delegate ProfileManager.ProfileBoolUpdater

Delegate to make updating profiles based on input bool callbacks easier

Namespace: [Optispeech.Profiles](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void ProfileBoolUpdater(bool value, ref ProfileManager.Profile profile);
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	value	The input value
ProfileManager.Profile	profile	The profile to apply the value to

Class ProfileManager.ProfileEvent

Create a UnityEvent that passes along a Profile to each listener

Inheritance

System.Object

UnityEngine.Events.UnityEventBase

UnityEngine.Events.UnityEvent<[ProfileManager.Profile](#)>

ProfileManager.ProfileEvent

Namespace: [Optispeech.Profiles](#)

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]  
public class ProfileEvent : UnityEvent<ProfileManager.Profile>, ISerializationCallbackReceiver
```

Delegate ProfileManager.ProfileFloatUpdater

Delegate to make updating profiles based on input float callbacks easier

Namespace: [Optispeech.Profiles](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void ProfileFloatUpdater(float value, ref ProfileManager.Profile profile);
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	value	The input value
ProfileManager.Profile	profile	The profile to apply the value to

Delegate ProfileManager.ProfileIntUpdater

Delegate to make updating profiles based on input int callbacks easier

Namespace: [Optispeech.Profiles](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void ProfileIntUpdater(int value, ref ProfileManager.Profile profile);
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	value	The input value
ProfileManager.Profile	profile	The profile to apply the value to

Delegate ProfileManager.ProfileStringUpdater

Delegate to make updating profiles based on input string callbacks easier

Namespace: [Optispeech.Profiles](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void ProfileStringUpdater(string value, ref ProfileManager.Profile profile);
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	value	The input value
ProfileManager.Profile	profile	The profile to apply the value to

Class ProfilePanel

Panel for creating, deleting, renaming, and switching between the various registered profiles

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
ProfilePanel

Namespace: [Optispeech.Profiles](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class ProfilePanel : MonoBehaviour
```

Fields

newProfileButton

Button for creating a new profile

Declaration

```
[SerializeField]  
Button newProfileButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

profileNameInput

Input field for putting in the name of the new profile to create

Declaration

```
[SerializeField]  
TMP_InputField profileNameInput
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

profileToggleContainer

Group that all profile buttons will be added to so only one can be active at a time

Declaration

```
[SerializeField]  
ToggleGroup profileToggleContainer
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.ToggleGroup	

profileTogglePrefab

Prefab to instantiate for each profile selector

Declaration

[SerializeField] ProfileSelector profileTogglePrefab

Field Value

TYPE	DESCRIPTION
ProfileSelector	

refreshCoroutine

Active couroutine, if any, that is refreshing this panel

Declaration

IEnumerator refreshCoroutine

Field Value

TYPE	DESCRIPTION
System.Collections.IEnumerator	

selectors

List of instantiated profile selectors

Declaration

List<ProfileSelector> selectors

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List< ProfileSelector >	

togglePanel

This panel

Declaration

TogglePanel togglePanel

Field Value

TYPE	DESCRIPTION
TogglePanel	

Methods

AddProfileSelector(ProfileManager.Profile)

Adds a selector for the given profile

Declaration

```
void AddProfileSelector(ProfileManager.Profile profile)
```

Parameters

TYPE	NAME	DESCRIPTION
ProfileManager.Profile	profile	The profile to add a selector for

DelayedRefresh()

Used to wait a frame then refresh this panel a second time, so that anything we effected has time to update

Declaration

```
IEnumerator DelayedRefresh()
```

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	Coroutine

GetProfileIndex(ProfileSelector)

Returns the current index of the given profile selector in [selectors](#)

Declaration

```
public int GetProfileIndex(ProfileSelector selector)
```

Parameters

TYPE	NAME	DESCRIPTION
ProfileSelector	selector	The profile selector to find the index of

Returns

TYPE	DESCRIPTION
System.Int32	The index of <code>selector</code> in selectors

RemoveProfileSelector(Int32)

Removes the selector for the profile at the given index

Declaration

```
public void RemoveProfileSelector(int index)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	index	The index in selectors of the profile selector to remove

UpdatePanelTitle(ProfileManager.Profile)

Changes the name of the panel to reflect the name of the active profile

Declaration

```
void UpdatePanelTitle(ProfileManager.Profile profile)
```

Parameters

TYPE	NAME	DESCRIPTION
ProfileManager.Profile	profile	

Class ProfileSelector

A UI object that represents a profile and allows it to be selected, renamed, and deleted

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
ProfileSelector

Namespace: [Optispeech.Profiles](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class ProfileSelector : MonoBehaviour
```

Fields

deleteProfileButton

The button to delete the profile this selector represents

Declaration

```
[SerializeField]  
Button deleteProfileButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

duplicateProfileButton

The button to duplicate the profile this selector represents

Declaration

```
[SerializeField]  
Button duplicateProfileButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

profileNameInput

Input field used for renaming the profile this selector represents

Declaration

```
[SerializeField]  
TMP_InputField profileNameInput
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

profileNameLabel

Label for the name of the profile this selector represents

Declaration

[SerializeField] TextMeshProUGUI profileNameLabel
--

Field Value

TYPE	DESCRIPTION
TMPro.TextMeshProUGUI	

renameProfileImage

Image that shows the status of whether or not the profile this selector represents is being renamed

Declaration

[SerializeField] Image renameProfileImage
--

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Image	

renameProfileSprite

The sprite to show on [renameProfileImage](#) when the profile this selector represents is not being renamed

Declaration

[SerializeField] Sprite renameProfileSprite
--

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

renameProfileToggle

Toggle that determines if the profile this selector represents is being renamed. Saves the profile name when toggled off.

Declaration

[SerializeField] Toggle renameProfileToggle
--

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

saveProfileNameSprite

The sprite to show on [renameProfileImage](#) when the profile this selector represents is being renamed

Declaration

```
[SerializeField]  
Sprite saveProfileNameSprite
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

toggle

The toggle that changes whether or not the profile this selector represents is the active profile

Declaration

```
public Toggle toggle
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

toggleImage

The image on the toggle that changes whether or not the profile this selector represents is the active profile. This is used to change the interactivity of the toggle dependent on whether or not the profile this selector represents is being renamed

Declaration

```
[SerializeField]  
Image toggleImage
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Image	

Methods

CloseRename()

Change the selector back to normal when renaming is either cancelled or saved

Declaration

```
void CloseRename()
```

Init(ProfilePanel, ProfileManager.Profile)

Sets up this selector with the given profile

Declaration

```
public void Init(ProfilePanel panel, ProfileManager.Profile profile)
```

Parameters

TYPE	NAME	DESCRIPTION
ProfilePanel	panel	The profile panel this selector will be added to
ProfileManager.Profile	profile	The profile to represent with this selector

Class SerializableVector3

A serializable version of a UnityEngine.Vector3, which will implicitly convert to/from UnityEngine.Vector3. This is used for saving UnityEngine.Vector3 values in profiles

Inheritance

System.Object

SerializableVector3

Namespace: [Optispeech.Profiles](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class SerializableVector3
```

Constructors

SerializableVector3(Single, Single, Single)

Constructor that takes x, y, and z components of the UnityEngine.Vector3 this represents

Declaration

```
public SerializableVector3(float x, float y, float z)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	x	The x value for this vector
System.Single	y	The y value for this vector
System.Single	z	The z value for this vector

Fields

x

This vector's x component

Declaration

```
public float x
```

Field Value

TYPE	DESCRIPTION
System.Single	

y

This vector's y component

Declaration

```
public float y
```

Field Value

TYPE	DESCRIPTION
System.Single	

z

This vector's z component

Declaration

```
public float z
```

Field Value

TYPE	DESCRIPTION
System.Single	

Operators

Implicit(SerializableVector3 to Vector3)

Implicitly converts [SerializableVector3](#) to UnityEngine.Vector3

Declaration

```
public static implicit operator Vector3(SerializableVector3 v)
```

Parameters

TYPE	NAME	DESCRIPTION
SerializableVector3	v	The value to convert into a UnityEngine.Vector3

Returns

TYPE	DESCRIPTION
UnityEngine.Vector3	

Implicit(Vector3 to SerializableVector3)

Implicitly converts UnityEngine.Vector3 to [SerializableVector3](#)

Declaration

```
public static implicit operator SerializableVector3(Vector3 v)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Vector3	v	The value to convert into a SerializableVector3

Returns

TYPE	DESCRIPTION
SerializableVector3	

Namespace Optispeech.Sensors

Classes

[SensorConfiguration](#)

Stores information about each active sensor

[SensorInformationDisplay](#)

UI element that shows the current status for a single sensor, as well as configuration settings for that sensor

[SensorsList](#)

This panel displays a list of all the current sensors, with their configuration

[SensorsManager](#)

Manages the current sensors and their roles

Enums

[SensorType](#)

Enumeration of all the different "types", or "roles" of sensors

Class SensorConfiguration

Stores information about each active sensor

Inheritance

System.Object

SensorConfiguration

Namespace: [Optispeech.Sensors](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class SensorConfiguration
```

Remarks

Note that this is a class: This way it will pass by reference so it can be changed and compared appropriately.

Fields

display

Reference to this sensor's display in the [SensorsList](#) panel

Declaration

```
public SensorInformationDisplay display
```

Field Value

TYPE	DESCRIPTION
SensorInformationDisplay	

id

ID of this sensor

Declaration

```
public int id
```

Field Value

TYPE	DESCRIPTION
System.Int32	

postOffset

Post offset is configured by the user and applied after the transform matrix is applied to the sensor. Can only be applied to non-reference sensors and for reference sensors will be equal to `UnityEngine.Vector3.zero`. Intended use is for making a sensor appear further back in the mouth than it really is and similar purposes

Declaration

```
public Vector3 postOffset
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

preOffset

Pre offset is calculated and applied to the forehead and ear sensors' raw positions such that the jaw sensor will line up with the avatar's jaw when the patient is in "resting position" For other sensors it will be equal to UnityEngine.Vector3.zero

Declaration

```
public Vector3 preOffset
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

status

The current status of this sensor, which is potentially updated at each data frame

Declaration

```
public SensorStatus status
```

Field Value

TYPE	DESCRIPTION
SensorStatus	

type

This sensor's role

Declaration

```
public SensorType type
```

Field Value

TYPE	DESCRIPTION
SensorType	

Class SensorInformationDisplay

UI element that shows the current status for a single sensor, as well as configuration settings for that sensor

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SensorInformationDisplay

Namespace: [Optispeech.Sensors](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class SensorInformationDisplay : MonoBehaviour
```

Fields

configuration

The configuration for this sensor

Declaration

```
SensorConfiguration configuration
```

Field Value

TYPE	DESCRIPTION
SensorConfiguration	

idLabel

Label that shows the ID of this sensor

Declaration

```
[SerializeField]  
TextMeshProUGUI idLabel
```

Field Value

TYPE	DESCRIPTION
TMPro.TextMeshProUGUI	

isReferenceSensor

Whether or not this sensor is a reference sensor (Jaw, Forehead, and Ears). If true, the offset fields are made non-interactive

Declaration

```
bool isReferenceSensor
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

statusIndicator

Image whose sprite will indicate the current status of the sensor

Declaration

[SerializeField] Image statusIndicator

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Image	

statusTooltip

Tooltip that displays the name of the status on hover

Declaration

[SerializeField] Tooltip statusTooltip

Field Value

TYPE	DESCRIPTION
Tooltip	

typeDropdown

Dropdown of all different sensor "roles", and allows the user to change which this sensor fills

Declaration

[SerializeField] TMP_Dropdown typeDropdown

Field Value

TYPE	DESCRIPTION
TMPro.TMP_Dropdown	

xOffsetField

Number field for changing the sensor's post-offset x value

Declaration

[SerializeField] TMP_InputField xOffsetField

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

yOffsetField

Number field for changing the sensor's post-offset y value

Declaration

[SerializeField] TMP_InputField yOffsetField

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

zOffsetField

Number field for changing the sensor's post-offset z value

Declaration

[SerializeField] TMP_InputField zOffsetField

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

Methods

Init(SensorConfiguration)

Sets up this display with the given sensor

Declaration

public void Init(SensorConfiguration configuration)

Parameters

TYPE	NAME	DESCRIPTION
SensorConfiguration	configuration	The sensor this display will represent

SetInteractable(Boolean)

Changes the interactivity of the various input fields as appropriate

Declaration

public void SetInteractable(bool interactable)
--

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	interactable	Whether or not these fields should be interactable

SetSensorStatus(SensorStatus)

Changes the displayed status for this sensor

Declaration

```
public void SetSensorStatus(SensorStatus status)
```

Parameters

TYPE	NAME	DESCRIPTION
SensorStatus	status	The new status for this sensor

Class SensorsList

This panel displays a list of all the current sensors, with their configuration

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SensorsList

Namespace: [Optispeech.Sensors](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class SensorsList : MonoBehaviour
```

Fields

contentContainer

The container for all the sensors, to use as a parent and to change the panel's height appropriately

Declaration

```
[SerializeField]  
RectTransform contentContainer
```

Field Value

TYPE	DESCRIPTION
UnityEngine.RectTransform	

isInteractable

This variable is used to track whether the sensors panel can currently be edited. If not, sensor types cannot be modified by the user; Usually because there either isn't an active data source, or the data source has indicated that it'll be controlling the sensors instead of the user

Declaration

```
bool isInteractable
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

resetRestingPosition

The reset resting position button, used to manually tell the tongue controller to recalculate the sensor pre-offsets so the jaw knows where resting position is

Declaration

```
[SerializeField]  
Button resetRestingPosition
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

sensorInfoDisplayPrefab

This should be a prefab with a SensorInformationDisplay MonoBehaviour on it. Each sensor in the list will be represented with one of these. This class assumes it has a RectTransform with a static height.

Declaration

<pre>[SerializeField] GameObject sensorInfoDisplayPrefab</pre>
--

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

sensors

List of sensor displays currently instantiated

Declaration

<pre>SensorInformationDisplay[] sensors</pre>

Field Value

TYPE	DESCRIPTION
SensorInformationDisplay[]	

Properties

IsInteractable

Property that handles updating the interactivity of each sensor display

Declaration

<pre>public bool IsInteractable { get; set; }</pre>

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

RecreateList()

Remove all the old sensor displays and re-create them based off the current sensors

Declaration

<pre>void RecreateList()</pre>

Class SensorsManager

Manages the current sensors and their roles

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SensorsManager

Namespace: [Optispeech.Sensors](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class SensorsManager : MonoBehaviour
```

Fields

foreheadSensorConfig

If it exists, the first sensor with [FOREHEAD](#) type

Declaration

```
public SensorConfiguration foreheadSensorConfig
```

Field Value

TYPE	DESCRIPTION
SensorConfiguration	

Instance

Static member to access the singleton instance of this class

Declaration

```
public static SensorsManager Instance
```

Field Value

TYPE	DESCRIPTION
SensorsManager	

jawSensorConfig

If it exists, the first sensor with [JAW](#) type

Declaration

```
public SensorConfiguration jawSensorConfig
```

Field Value

TYPE	DESCRIPTION
SensorConfiguration	

leftEarSensorConfig

If it exists, the first sensor with LEFT_EAR type

Declaration

```
public SensorConfiguration leftEarSensorConfig
```

Field Value

TYPE	DESCRIPTION
SensorConfiguration	

offsetsDirty

Flag to tell tongue controller whether or not to re-calculate pre-offsets

Declaration

```
[HideInInspector]  
public bool offsetsDirty
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

onListUpdate

Event that fires whenever the sensors list changes

Declaration

```
[HideInInspector]  
public UnityEvent onListUpdate
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent	

otherSensorConfigs

All sensors with OTHER type

Declaration

```
public IEnumerable<SensorConfiguration> otherSensorConfigs
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.IEnumerable< SensorConfiguration >	

panel

The panel with a list of all sensors, used to handle displaying and updating sensor configurations

Declaration

```
public SensorsList panel
```

Field Value

TYPE	DESCRIPTION
SensorsList	

rightEarSensorConfig

If it exists, the first sensor with [RIGHT_EAR](#) type

Declaration

```
public SensorConfiguration rightEarSensorConfig
```

Field Value

TYPE	DESCRIPTION
SensorConfiguration	

sensors

List of all current sensors in the most recent data frame

Declaration

```
public SensorConfiguration[] sensors
```

Field Value

TYPE	DESCRIPTION
SensorConfiguration []	

sensorsDirty

Flag representing whether the sensors have changed

Declaration

```
bool sensorsDirty
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

tongueBackSensorConfig

If it exists, the first sensor with [TONGUE_BACK](#) type

Declaration

```
public SensorConfiguration tongueBackSensorConfig
```

Field Value

TYPE	DESCRIPTION
SensorConfiguration	

tongueDorsumSensorConfig

If it exists, the first sensor with [TONGUE_DORSUM](#) type

Declaration

```
public SensorConfiguration tongueDorsumSensorConfig
```

Field Value

TYPE	DESCRIPTION
SensorConfiguration	

tongueLeftSensorConfig

If it exists, the first sensor with [TONGUE_LEFT](#) type

Declaration

```
public SensorConfiguration tongueLeftSensorConfig
```

Field Value

TYPE	DESCRIPTION
SensorConfiguration	

tongueRightSensorConfig

If it exists, the first sensor with [TONGUE_RIGHT](#) type

Declaration

```
public SensorConfiguration tongueRightSensorConfig
```

Field Value

TYPE	DESCRIPTION
SensorConfiguration	

tongueTipSensorConfig

If it exists, the first sensor with [TONGUE_TIP](#) type

Declaration

```
public SensorConfiguration tongueTipSensorConfig
```

Field Value

TYPE	DESCRIPTION
SensorConfiguration	

Methods

ChangeSensorType(SensorConfiguration, SensorType)

Changes the type of a sensor and re-calculates the relevant roles

Declaration

```
public void ChangeSensorType(SensorConfiguration sensor, SensorType newType)
```

Parameters

TYPE	NAME	DESCRIPTION
SensorConfiguration	sensor	The sensor whose type has changed
SensorType	newType	The new type for this sensor

Reset()

Removes all sensor configurations

Declaration

```
public void Reset()
```

SaveSensors()

Saves the sensor configurations to the active profile, so they can be re-loaded when switching to the same data source reader

Declaration

```
public void SaveSensors()
```

SetSensors(SensorConfiguration[])

Updates list of sensors and assigns them to the appropriate roles

Declaration

```
public void SetSensors(SensorConfiguration[] sensors)
```

Parameters

TYPE	NAME	DESCRIPTION
SensorConfiguration []	sensors	The new array of sensors

Remarks

Note: A flag is used to know when to recreate the sensors list in Update because this method is intended to be called on a data source reader's thread, which will throw an error upon accessing a game object, transform, etc.

Enum SensorType

Enumeration of all the different "types", or "roles" of sensors

Namespace: [Optispeech.Sensors](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public enum SensorType
```

Fields

NAME	DESCRIPTION
FOREHEAD	
IGNORED	
JAW	
LEFT_EAR	
OTHER	
RIGHT_EAR	
TONGUE_BACK	
TONGUE_DORSUM	
TONGUE_LEFT	
TONGUE_RIGHT	
TONGUE_TIP	

Namespace Optispeech.Sweeps

Classes

[FileWritingManager](#)

Handles writing data from sweeps

[SavWav](#)

Saves wav data from an UnityEngine.AudioClip to file

[SweepsPanel](#)

Panel for starting and stopping sweeps and configuring where and what to write during the sweep

Class FileWritingManager

Handles writing data from sweeps

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
FileWritingManager

Namespace: [Optispeech.Sweeps](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class FileWritingManager : MonoBehaviour
```

Fields

audio

Audioclip that stores any recorded audio during the sweep

Declaration

```
AudioClip audio
```

Field Value

TYPE	DESCRIPTION
UnityEngine.AudioClip	

elapsedTime

How long has elapsed since the start of the sweep

Declaration

```
[HideInInspector]  
public TimeSpan elapsedTime
```

Field Value

TYPE	DESCRIPTION
System.TimeSpan	

filename

The base filename to write sweep data to

Declaration

```
string filename
```

Field Value

TYPE	DESCRIPTION
System.String	

finishing

Flag used to track when a sweep ends to handle the sweep ending on the main thread

Declaration

```
bool finishing
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Instance

Static member to access the singleton instance of this class

Declaration

```
public static FileWritingManager Instance
```

Field Value

TYPE	DESCRIPTION
FileWritingManager	

onSweepEnd

Event that fires whenever a sweep ends

Declaration

```
[HideInInspector]  
public UnityEvent onSweepEnd
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent	

onSweepStart

Event that fires whenever a sweep starts

Declaration

```
[HideInInspector]  
public UnityEvent onSweepStart
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent	

raw

Writer that streams raw data to file

Declaration

StreamWriter raw

Field Value

TYPE	DESCRIPTION
System.IO.StreamWriter	

recordingSweep

Flag representing whether or not a sweep is being recorded

Declaration

[HideInInspector] public bool recordingSweep

Field Value

TYPE	DESCRIPTION
System.Boolean	

remainingFrames

When a sweep ends this is used to track how many frames are remaining in the queue and still need to be processed before the writing to file can officially stop

Declaration

[HideInInspector] public int remainingFrames

Field Value

TYPE	DESCRIPTION
System.Int32	

sweepsPanel

The sweeps panel that starts and stops writing data to file, as well as handles what to write and where to write it to

Declaration

[SerializeField] SweepsPanel sweepsPanel

Field Value

TYPE	DESCRIPTION
SweepsPanel	

transformed

Writer that streams transformed data to file

Declaration

StreamWriter transformed

Field Value

TYPE	DESCRIPTION
System.IO.StreamWriter	

transformedWithoutOffsets

Writer that streams transformed data, but without post-offsets, to file

Declaration

StreamWriter transformedWithoutOffsets
--

Field Value

TYPE	DESCRIPTION
System.IO.StreamWriter	

Methods

GetSensorString(Nullable<SensorData>, Boolean)

Creates a config string of sensor data in a single frame

Declaration

<code>string GetSensorString(SensorData? sensorData, bool includePostOffsets = true)</code>

Parameters

TYPE	NAME	DESCRIPTION
System.Nullable< SensorData >	sensorData	The sensor data to create a string out of
System.Boolean	includePostOffsets	Whether or not to include post-offsets

Returns

TYPE	DESCRIPTION
System.String	A string representation of this sensor's data this frame

MakeRelativePath(String, String)

Creates a relative path from one file or folder to another.

Declaration

```
public static string MakeRelativePath(string fromPath, string toPath)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	fromPath	Contains the directory that defines the start of the relative path.
System.String	toPath	Contains the path that defines the endpoint of the relative path.

Returns

TYPE	DESCRIPTION
System.String	The relative path from the start directory to the end path or <code>toPath</code> if the paths are not related.

Exceptions

TYPE	CONDITION
System.ArgumentNullException	
System.UriFormatException	
System.InvalidOperationException	

Record()

Function that runs in new thread to constantly write frames while a sweep is ongoing

Declaration

```
void Record()
```

StartAudio()

Starts recording microphone audio to [audio](#)

Declaration

```
void StartAudio()
```

StartRaw()

Creates System.IO.StreamWriter and writes header for the raw sweep data

Declaration

```
void StartRaw()
```

StartTransformed()

Creates System.IO.StreamWriter and writes header for the transformed sweep data

Declaration

```
void StartTransformed()
```

StartTransformedWithoutOffsets()

Creates System.IO.StreamWriter and writes header for the transformed sweep data without offsets

Declaration

```
void StartTransformedWithoutOffsets()
```

ToggleSweep()

Handles starting and stopping a sweep. Sweeps can be toggled to start from the main thread only, but can be stopped from any

Declaration

```
public void ToggleSweep()
```

WriteFrameToFile(DataFrame)

Writes a frame to file

Declaration

```
void WriteFrameToFile(DataFrame frame)
```

Parameters

TYPE	NAME	DESCRIPTION
DataFrame	frame	The data frame to write to file

Class SavWav

Saves wav data from an UnityEngine.AudioClip to file

Inheritance

System.Object

SavWav

Namespace: [Optispeech.Sweeps](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public static class SavWav
```

Fields

HEADER_SIZE

Size of the header in a wav file

Declaration

```
const int HEADER_SIZE = 44
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Methods

ConvertAndWrite(FileStream, AudioClip)

Converts the audio data from an UnityEngine.AudioClip and writes it to a System.IO.FileStream

Declaration

```
static void ConvertAndWrite(FileStream fileStream, AudioClip clip)
```

Parameters

TYPE	NAME	DESCRIPTION
System.IO.FileStream	fileStream	The file stream to write audio data to
UnityEngine.AudioClip	clip	The audio clip to get data from

CreateEmpty(String)

Create a System.IO.FileStream to the given filepath with a header filled with empty bytes

Declaration

```
static FileStream CreateEmpty(string filepath)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	filepath	Filepath to create the System.IO.FileStream at

Returns

TYPE	DESCRIPTION
System.IO.FileStream	The file stream with empty header

Save(String, AudioClip)

Saves the data from an audioclip to a file

Declaration

```
public static bool Save(string filepath, AudioClip clip)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	filepath	The file location to write the audio data to
UnityEngine.AudioClip	clip	The audioclip that contains the data to write

Returns

TYPE	DESCRIPTION
System.Boolean	True if the file is successfully written

TrimSilence(List<Single>, Single, Int32, Int32)

Calls [TrimSilence\(List<Single>, Single, Int32, Int32, Boolean\)](#) with streaming set to false

Declaration

```
public static AudioClip TrimSilence(List<float> samples, float min, int channels, int hz)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.List<System.Single>	samples	The audio samples to create an audio clip out of
System.Single	min	The minimum volume threshold that won't get filtered out

TYPE	NAME	DESCRIPTION
System.Int32	channels	The number of channels to give the audioclip
System.Int32	hz	The frequency to give the audioclip

Returns

TYPE	DESCRIPTION
UnityEngine.AudioClip	An audioclip with trimmed silence with the given samples, channels, and frequency

See Also

[TrimSilence\(AudioClip, Single\)](#)

[TrimSilence\(List<Single>, Single, Int32, Int32, Boolean\)](#)

TrimSilence(List<Single>, Single, Int32, Int32, Boolean)

Creates an audio clip, but with the silence trimmed from both ends, from a given list of audio samples and volume threshold to determine what constitutes silent.

Declaration

```
public static AudioClip TrimSilence(List<float> samples, float min, int channels, int hz, bool stream)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.List<System.Single>	samples	The audio samples to create an audio clip out of
System.Single	min	The minimum volume threshold that won't get filtered out
System.Int32	channels	The number of channels to give the audioclip
System.Int32	hz	The frequency to give the audioclip
System.Boolean	stream	The stream value to give the audioclip

Returns

TYPE	DESCRIPTION
UnityEngine.AudioClip	An audioclip with trimmed silence with the given samples, channels, frequency, and stream values

See Also

[TrimSilence\(AudioClip, Single\)](#)
[TrimSilence\(List<Single>, Single, Int32, Int32\)](#)

TrimSilence(AudioClip, Single)

Takes an audioclip and trims the beginning and end for as long as they're below a given volume threshold

Declaration

```
public static AudioClip TrimSilence(AudioClip clip, float min)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.AudioClip	clip	The audioclip to trim
System.Single	min	The minimum volume threshold that won't get filtered

Returns

TYPE	DESCRIPTION
UnityEngine.AudioClip	A new audio clip with trimmed silence from either end

See Also

[TrimSilence\(List<Single>, Single, Int32, Int32\)](#)
[TrimSilence\(List<Single>, Single, Int32, Int32, Boolean\)](#)

WriteHeader(FileStream, AudioClip)

Fills a System.IO.FileStream's header with data from an UnityEngine.AudioClip

Declaration

```
static void WriteHeader(FileStream fileStream, AudioClip clip)
```

Parameters

TYPE	NAME	DESCRIPTION
System.IO.FileStream	fileStream	File stream to write the header to
UnityEngine.AudioClip	clip	The audio clip to get header data from

Class SweepsPanel

Panel for starting and stopping sweeps and configuring where and what to write during the sweep

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SweepsPanel

Namespace: [Optispeech.Sweeps](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class SweepsPanel : MonoBehaviour
```

Fields

autoStopDurationInput

Input field to set the duration before a sweep automatically stops

Declaration

```
[SerializeField]  
TMP_InputField autoStopDurationInput
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

browserButton

Button that opens a file browser to find a folder to save

Declaration

```
[SerializeField]  
Button browserButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

durationText

During a sweep, this label will display how long its recorded for

Declaration

```
[SerializeField]  
TextMeshProUGUI durationText
```

Field Value

TYPE	DESCRIPTION
TMPro.TextMeshProUGUI	

folderPathInput

Input field to set the folder to write sweep data into

Declaration

```
[SerializeField]
TMP_InputField folderPathInput
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

pollFrequency

How long to wait (in ms) before checking for a new data frame while recording sweeps if set too high then it'll process frames until it catches up

Declaration

```
public int pollFrequency
```

Field Value

TYPE	DESCRIPTION
System.Int32	

saveRawToggle

Toggle to determine whether or not to save raw sweep data

Declaration

```
[SerializeField]
Toggle saveRawToggle
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

saveSyncedAudioToggle

Toggle to determine whether or not to save audio data

Declaration

```
[SerializeField]
Toggle saveSyncedAudioToggle
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

saveTransformedToggle

Toggle to determine whether or not to save transformed sweep data

Declaration

[SerializeField] Toggle saveTransformedToggle
--

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

saveTransformedWithoutOffsetsToggle

Toggle to determine whether or not to save transformed sweep data without post-offsets

Declaration

[SerializeField] Toggle saveTransformedWithoutOffsetsToggle
--

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

sweepNameInput

Input field to set the name of the sweep, which is used as part of the filename each sweep data file is written to

Declaration

[SerializeField] TMP_InputField sweepNameInput

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

toggleSweepButton

Button to start and stop sweep recordings

Declaration

[SerializeField] Button toggleSweepButton
--

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

toggleSweepImage

Toggle to determine whether or not to save raw sweep data

Declaration

[SerializeField] Image toggleSweepImage
--

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Image	

toggleSweepText

Label to display if [toggleSweepButton](#) will start or stop a sweep

Declaration

[SerializeField] TextMeshProUGUI toggleSweepText

Field Value

TYPE	DESCRIPTION
TMPro.TextMeshProUGUI	

Methods

BrowseFolderPath()

Opens a file browser window and sets [folderPathInput](#) to the chosen folder path

Declaration

void BrowseFolderPath()

DisplayFinishing()

Updates the panel to display that we're finishing a sweep

Declaration

public void DisplayFinishing()

DisplayRecording()

Updates the panel to display that we're currently recording

Declaration

public void DisplayRecording()

ResetDisplay()

Resets the panel to non-recording, non-finishing state

Declaration

```
public void ResetDisplay()
```

UpdateDuration(TimeSpan)

Updates the time display with the new amount of elapsed time

Declaration

```
public void UpdateDuration(TimeSpan elapsedTime)
```

Parameters

TYPE	NAME	DESCRIPTION
System.TimeSpan	elapsedTime	The amount of time since the start of the sweep

Namespace Optispeech.Targets

Classes

[TargetConfig](#)

Manages a set of target configuration UI elements

[TargetController](#)

Controls a target

[TargetDescription](#)

ScriptableObject that describes a single target type

[TargetsManager](#)

Managers the configured targets

[TargetsPanel](#)

Panel that shows the current targets and allows them to be added, removed, and edited

Class TargetConfig

Manages a set of target configuration UI elements

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- TargetConfig
- CustomMotionTargetConfig
- LandmarkTargetConfig
- OscillatingTargetConfig
- StaticTargetConfig

Namespace: [Optispeech.Targets](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public abstract class TargetConfig : MonoBehaviour
```

Fields

deleteTargetButton

Button to delete this target

Declaration

```
[SerializeField]  
Button deleteTargetButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

radiusField

Number field for the radius of the target marker

Declaration

```
[SerializeField]  
TMP_InputField radiusField
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

targetIDLabel

Label to show the ID of the target

Declaration

```
[SerializeField]
TextMeshProUGUI targetIDLabel
```

Field Value

TYPE	DESCRIPTION
TMPro.TextMeshProUGUI	

visibilitySlider

Slider to show how visible the target should be

Declaration

```
[SerializeField]
Slider visibilitySlider
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Slider	

Methods

Init(TargetsPanel, TargetController)

Sets up this config with the given target controller

Declaration

```
public virtual void Init(TargetsPanel panel, TargetController controller)
```

Parameters

TYPE	NAME	DESCRIPTION
TargetsPanel	panel	The panel this config will appear in
TargetController	controller	The target controller this config is for

OnClose()

Optional function for doing custom things whenever the config menu is closed

Declaration

```
public virtual void OnClose()
```

Remarks

Note these also fire when closing the targets panel

OnOpen()

Optional function for doing custom things whenever the config menu is opened

Declaration


```
public virtual void OnOpen()
```

Remarks

Note these also fire when opening the targets panel

SetInteractable(Boolean)

Changes the interactivity of the UI elements

Declaration

```
public virtual void SetInteractable(bool interactable)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	interactable	Whether or not the config UI elements should be interactable

Class TargetController

Controls a target

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
TargetController

[CurvedTargetController](#)
[CustomMotionTargetController](#)
[LandmarkTargetController](#)
[OscillatingTargetController](#)
[StaticTargetController](#)

Namespace: [Optispeech.Targets](#)

Assembly: Assembly-CSharp.dll

Syntax

```
[RequireComponent(typeof(MeshRenderer))]  
[RequireComponent(typeof(Tooltip))]  
[RequireComponent(typeof(TrailRenderer))]  
public abstract class TargetController : MonoBehaviour
```

Fields

configToggle

The toggle for opening/closing this target's config

Declaration

```
[HideInInspector]  
public GameObject configToggle
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

description

The description of this target, which includes a type name and prefab with this controller on it, and the prefab for the config to add to the targets panel

Declaration

```
[HideInInspector]  
public TargetDescription description
```

Field Value

TYPE	DESCRIPTION
TargetDescription	

mesh

The renderer for the target marker, for setting radius and color

Declaration

```
[HideInInspector]
public MeshRenderer mesh
```

Field Value

TYPE	DESCRIPTION
UnityEngine.MeshRenderer	

NUM_BASE_CONFIG_VALUES

This is the number of config values written to a config string This is stored in a constant so implementations can know to offset their own values by this amount

Declaration

```
protected static readonly int NUM_BASE_CONFIG_VALUES
```

Field Value

TYPE	DESCRIPTION
System.Int32	

radius

The radius of the target marker as well as how close the user needs to be to be considered "in" the target

Declaration

```
public float radius
```

Field Value

TYPE	DESCRIPTION
System.Single	

targetId

The name of this target

Declaration

```
[HideInInspector]
public string targetId
```

Field Value

TYPE	DESCRIPTION
System.String	

tooltip

Tooltip used to display this target's name on hover

target.TargetType == TargetType.Tongue

Declaration

<code>[HideInInspector]</code> <code>Tooltip tooltip</code>
--

Field Value

TYPE	DESCRIPTION
<code>Tooltip</code>	

trail

The trail renderer used to show this target's history, when the config is open

Declaration

<code>[HideInInspector]</code> <code>TrailRenderer trail</code>
--

Field Value

TYPE	DESCRIPTION
<code>UnityEngine.TrailRenderer</code>	

visibility

How visible this target appears

Declaration

<code>public float visibility</code>

Field Value

TYPE	DESCRIPTION
<code>System.Single</code>	

Methods

ApplyConfigFromString(String)

Sets up this target based on a target configuration string from a raw export

Declaration

<code>public virtual void ApplyConfigFromString(string config)</code>

Parameters

TYPE	NAME	DESCRIPTION
<code>System.String</code>	<code>config</code>	The configuration of this target as a tab separated values string

GetCycleDuration()

When displaying how accurate the tongue was at matching the targets, the accuracy is shown in terms of "cycles". This describes

how long a cycle is for this target, in ms

Declaration

```
public abstract long GetCycleDuration()
```

Returns

TYPE	DESCRIPTION
System.Int64	The duration of one cycle for this target

GetTargetPosition(Int64)

Returns the current position of this target based on timestamp

Declaration

```
public abstract Vector3 GetTargetPosition(long currTime)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int64	currTime	The current time of this data frame

Returns

TYPE	DESCRIPTION
UnityEngine.Vector3	The position of this target at the given timestamp

OnClose()

Function that's called whenever this target's config is closed

Declaration

```
public virtual void OnClose()
```

OnOpen()

Function that's called whenever this target's config is opened

Declaration

```
public virtual void OnOpen()
```

ToString()

Creates a tab separated values string of all the configs for this target

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
System.String	Tab separated values string

Overrides
UnityEngine.Object.ToString()

UpdateTarget(Vector3, Nullable<SensorData>)

Updates the target marker to appear in the specified position, with a color determined by the distance from the given sensor (if provided)

Declaration

```
public virtual void UpdateTarget(Vector3 targetPosition, SensorData? tongueTipSensor)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Vector3	targetPosition	The position the marker should appear at
System.Nullable< SensorData >	tongueTipSensor	The sensor, if it exists, this target is tracking

Class TargetDescription

ScriptableObject that describes a single target type

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.ScriptableObject
- TargetDescription

Namespace: [Optispeech.Targets](#)
Assembly: Assembly-CSharp.dll

Syntax

```
[CreateAssetMenu(menuName = "Optispeech/Target Type")]  
public class TargetDescription : ScriptableObject
```

Fields

targetConfigPrefab

A prefab that shows the configuration UI elements for targets of this type

Declaration

```
public TargetConfig targetConfigPrefab
```

Field Value

TYPE	DESCRIPTION
TargetConfig	

targetPrefab

A prefab with the target controller to use for these types of targets

Declaration

```
public TargetController targetPrefab
```

Field Value

TYPE	DESCRIPTION
TargetController	

typeName

The name of this target type. Shown in the add target dropdown, and is used to identify the types of targets in their config strings

Declaration

```
public string typeName
```

Field Value

TYPE	DESCRIPTION
System.String	

Class TargetsManager

Managers the configured targets

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- TargetsManager

Namespace: [Optispeech.Targets](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class TargetsManager : MonoBehaviour
```

Fields

closeColor

The color a target marker should be when the sensor is within the target radius

Declaration

```
public Color closeColor
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Color	

farColor

The color a target marker should be when the sensor is far away from the target

Declaration

```
public Color farColor
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Color	

Instance

Static member to access the singleton instance of this class

Declaration

```
public static TargetsManager Instance
```

Field Value

TYPE	DESCRIPTION
TargetsManager	

panel

The targets panel, used for clearing it when clearing all targets

Declaration

```
public TargetsPanel panel
```

Field Value

TYPE	DESCRIPTION
TargetsPanel	

targetDescriptions

A list of all the different target type descriptions, which is used to load the targets for the specific data source whenever its changed

Declaration

```
TargetDescription[] targetDescriptions
```

Field Value

TYPE	DESCRIPTION
TargetDescription[]	

targets

Dictionary of target IDs to their controllers

Declaration

```
[HideInInspector]  
public Dictionary<string, TargetController> targets
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, TargetController>	

Methods

AddTarget(TargetDescription, Boolean)

Adds a target with the given description

Declaration

```
public string AddTarget(TargetDescription description, bool manuallyAdded = true)
```

Parameters

TYPE	NAME	DESCRIPTION
TargetDescription	description	The description of the target to add
System.Boolean	manuallyAdded	Whether this target was loaded or created by the user just now

Returns

TYPE	DESCRIPTION
System.String	The ID of the created target

RemoveTarget(TargetController)

Removes a target from the list of targets and from the targets panel

Declaration

```
void RemoveTarget(TargetController controller)
```

Parameters

TYPE	NAME	DESCRIPTION
TargetController	controller	Target to remove

ResetTargets()

Updates targets to the ones in the ActiveProfile

Declaration

```
public void ResetTargets()
```

Class TargetsPanel

Panel that shows the current targets and allows them to be added, removed, and edited

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
TargetsPanel

Namespace: [Optispeech.Targets](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class TargetsPanel : MonoBehaviour
```

Fields

activeConfig

Which target config is currently open

Declaration

```
GameObject activeConfig
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

activeController

When [activeConfig](#) is not null, this will be the controller that config is associated with

Declaration

```
TargetController activeController
```

Field Value

TYPE	DESCRIPTION
TargetController	

addedPanelEvent

Flag to ensure the panelToggledEvent event listener is only added once

Declaration

```
bool addedPanelEvent
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

addTargetDropdown

Dropdown to select the type of the new target to add

Declaration

[SerializeField] TMP_Dropdown addTargetDropdown
--

Field Value

TYPE	DESCRIPTION
TMPro.TMP_Dropdown	

configContainer

Container to place the active config prefab inside

Declaration

[SerializeField] RectTransform configContainer

Field Value

TYPE	DESCRIPTION
UnityEngine.RectTransform	

isInteractable

This variable is used to track whether the targets panel can currently be edited. If not, targets cannot be added, removed, or modified by the user; Usually because there either isn't an active data source, or the data source has indicated that it'll be controlling the targets instead of the user

Declaration

bool isInteractable

Field Value

TYPE	DESCRIPTION
System.Boolean	

panel

This panel

Declaration

TogglePanel panel

Field Value

TYPE	DESCRIPTION
TogglePanel	

targets

List of all the current target descriptions

Declaration

<code>TargetDescription[] targets</code>
--

Field Value

TYPE	DESCRIPTION
TargetDescription[]	

targetToggleContainer

Each target gets a toggle to open/close its config; this is the container all those toggles are placed in

Declaration

<code>[SerializeField] ToggleGroup targetToggleContainer</code>

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.ToggleGroup	

targetTogglePrefab

Prefab that is instantiated for each target. Each toggles which target is currently being configured

Declaration

<code>[SerializeField] Toggle targetTogglePrefab</code>

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

Properties

IsInteractable

Property that handles updating the interactivity of the active target config

Declaration

<code>public bool IsInteractable { get; set; }</code>

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

AddTarget(TargetController, Boolean)

Adds a target toggle to the list

Declaration

```
public void AddTarget(TargetController controller, bool manuallyAdded = true)
```

Parameters

TYPE	NAME	DESCRIPTION
TargetController	controller	Target this toggle belongs to
System.Boolean	manuallyAdded	Whether or not this target has been loaded or created

AddTarget(Int32)

Adds a target with the type specified by the given index to the targets manager

Declaration

```
void AddTarget(int index)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	index	The index in targets of the target type this new target should have

CloseConfig()

Closes the currently open target config, if any

Declaration

```
public void CloseConfig()
```

DelayedOpen(TargetController)

Creates a coroutine that waits a frame before opening a target config, so this panel can start closing if necessary, because otherwise there are issues with the height being slightly miscalculated

Declaration

```
IEnumerator DelayedOpen(TargetController controller)
```

Parameters

TYPE	NAME	DESCRIPTION
TargetController	controller	The target to open the config for

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	A coroutine

DelayedRefresh()

Creates a coroutine that waits a frame then refreshes this panel a second time, when all the UI elements have finished updating (for some reason it wouldn't update the height correctly before a frame passed)

Declaration

IEnumerator DelayedRefresh()

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	A coroutine

OpenConfig(TargetController)

Opens a specified target config

Declaration

void OpenConfig(TargetController controller)
--

Parameters

TYPE	NAME	DESCRIPTION
TargetController	controller	The target to open the config for

SaveTargetsToPrefs()

Create config strings for each target and save it to the active profile

Declaration

public void SaveTargetsToPrefs()

Namespace Optispeech.Targets.Configs

Classes

[CustomMotionTargetConfig](#)

Configuration interface for [CustomMotionTargetControllers](#)

[LandmarkTargetConfig](#)

Configuration interface for [LandmarkTargetControllers](#)

[OscillatingTargetConfig](#)

Configuration interface for [OscillatingTargetControllers](#)

[StaticTargetConfig](#)

Configuration interface for [StaticTargetControllers](#)

Class CustomMotionTargetConfig

Configuration interface for [CustomMotionTargetControllers](#)

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- [TargetConfig](#)
- CustomMotionTargetConfig

Inherited Members

- [TargetConfig.targetIDLabel](#)
- [TargetConfig.deleteTargetButton](#)
- [TargetConfig.visibilitySlider](#)
- [TargetConfig.radiusField](#)
- [TargetConfig.Init\(TargetsPanel, TargetController\)](#)
- [TargetConfig.SetInteractable\(Boolean\)](#)
- [TargetConfig.OnOpen\(\)](#)
- [TargetConfig.OnClose\(\)](#)

Namespace: [Optispeech.Targets.Configs](#)
Assembly: Assembly-CSharp.dll

Syntax

```
public class CustomMotionTargetConfig : TargetConfig
```

Fields

alternateDirectionToggle

Toggle for [alternateDirection](#)

Declaration

```
[SerializeField]  
Toggle alternateDirectionToggle
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Toggle	

idDropdown

Dropdown for [id](#)

Declaration

```
[SerializeField]  
TMP_Dropdown idDropdown
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_Dropdown	

loadFile

Button to load a new file to use for the custom motion

Declaration

[SerializeField] Button loadFile

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

local

Reference to en-US locale for making the ID dropdown in title case

Declaration

static readonly TextInfo local

Field Value

TYPE	DESCRIPTION
System.Globalization.TextInfo	

motionController

Store the currently active controller so [config](#) can be set to null when this is closed

Declaration

CustomMotionTargetController motionController

Field Value

TYPE	DESCRIPTION
CustomMotionTargetController	

offsetXInput

Field for [offset](#)'s x component

Declaration

[SerializeField] TMP_InputField offsetXInput

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

offsetYInput

Field for [offset](#)'s y component

Declaration

[SerializeField] TMP_InputField offsetYInput

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

offsetZInput

Field for [offset](#)'s z component

Declaration

[SerializeField] TMP_InputField offsetZInput

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

panel

The panel this config is on

Declaration

TargetsPanel panel

Field Value

TYPE	DESCRIPTION
TargetsPanel	

playbackSpeedInput

Number field for [playbackSpeed](#)

Declaration

[SerializeField] TMP_InputField playbackSpeedInput

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

preparingStatusIcon

Sprite to show when **status** is PREPARING

Declaration

[SerializeField] Sprite preparingStatusIcon
--

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

readyStatusIcon

Sprite to show when **status** is READY

Declaration

[SerializeField] Sprite readyStatusIcon
--

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

statusIndicator

Image that displays the current **status**

Declaration

[SerializeField] Image statusIndicator

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Image	

unavailableStatusIcon

Sprite to show when **status** is UNAVAILABLE

Declaration

[SerializeField] Sprite unavailableStatusIcon
--

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

Methods

SetStatus(CustomMotionTargetController.MotionPathStatus)

Handles updating the motion path status indicator

Declaration

```
public void SetStatus(CustomMotionTargetController.MotionPathStatus status)
```

Parameters

TYPE	NAME	DESCRIPTION
CustomMotionTargetController.MotionPathStatus	status	The new motion path status

Class LandmarkTargetConfig

Configuration interface for [LandmarkTargetControllers](#)

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- [TargetConfig](#)
- LandmarkTargetConfig

Inherited Members

- [TargetConfig.targetIDLabel](#)
- [TargetConfig.deleteTargetButton](#)
- [TargetConfig.visibilitySlider](#)
- [TargetConfig.radiusField](#)
- [TargetConfig.Init\(TargetsPanel, TargetController\)](#)
- [TargetConfig.SetInteractable\(Boolean\)](#)
- [TargetConfig.OnOpen\(\)](#)
- [TargetConfig.OnClose\(\)](#)

Namespace: [Optispeech.Targets.Configs](#)
Assembly: Assembly-CSharp.dll

Syntax

```
public class LandmarkTargetConfig : TargetConfig
```

Fields

idDropdown

Dropdown for [id](#)

Declaration

```
[SerializeField]  
TMP_Dropdown idDropdown
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_Dropdown	

local

Reference to en-US locale for making the ID dropdown in title case

Declaration

```
static readonly TextInfo local
```

Field Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Globalization.TextInfo	

Methods

SetupDropdown()

Sets up the dropdown with options for each active sensor ID

Declaration

```
void SetupDropdown()
```

Class OscillatingTargetConfig

Configuration interface for [OscillatingTargetControllers](#)

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- [TargetConfig](#)
- OscillatingTargetConfig

Inherited Members

- [TargetConfig.targetIDLabel](#)
- [TargetConfig.deleteTargetButton](#)
- [TargetConfig.visibilitySlider](#)
- [TargetConfig.radiusField](#)
- [TargetConfig.Init\(TargetsPanel, TargetController\)](#)
- [TargetConfig.SetInteractable\(Boolean\)](#)
- [TargetConfig.OnOpen\(\)](#)
- [TargetConfig.OnClose\(\)](#)

Namespace: [Optispeech.Targets.Configs](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class OscillatingTargetConfig : TargetConfig
```

Fields

endXPosField

Field for [endPosition](#)'s x component

Declaration

```
[SerializeField]  
TMP_InputField endXPosField
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

endYPosField

Field for [endPosition](#)'s y component

Declaration

```
[SerializeField]  
TMP_InputField endYPosField
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

endZPosField

Field for [endPosition](#)'s z component

Declaration

[SerializeField] TMP_InputField endZPosField

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

frequencyField

Field for [frequency](#)

Declaration

[SerializeField] TMP_InputField frequencyField

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

startXPosField

Field for [startPosition](#)'s x component

Declaration

[SerializeField] TMP_InputField startXPosField

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

startYPosField

Field for [startPosition](#)'s y component

Declaration

[SerializeField] TMP_InputField startYPosField

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

startZPosField

Field for [startPosition](#)'s z component

Declaration

[SerializeField] TMP_InputField startZPosField

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

Class StaticTargetConfig

Configuration interface for [StaticTargetControllers](#)

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- [TargetConfig](#)
- StaticTargetConfig

Inherited Members

- [TargetConfig.targetIDLabel](#)
- [TargetConfig.deleteTargetButton](#)
- [TargetConfig.visibilitySlider](#)
- [TargetConfig.radiusField](#)
- [TargetConfig.Init\(TargetsPanel, TargetController\)](#)
- [TargetConfig.SetInteractable\(Boolean\)](#)
- [TargetConfig.OnOpen\(\)](#)
- [TargetConfig.OnClose\(\)](#)

Namespace: [Optispeech.Targets.Configs](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class StaticTargetConfig : TargetConfig
```

Fields

xPosField

Field for [position](#)'s x component

Declaration

```
[SerializeField]  
TMP_InputField xPosField
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

yPosField

Field for [position](#)'s y component

Declaration

```
[SerializeField]  
TMP_InputField yPosField
```

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

zPosField

Field for [position](#)'s z component

Declaration

[SerializeField] TMP_InputField zPosField
--

Field Value

TYPE	DESCRIPTION
TMPro.TMP_InputField	

Namespace Optispeech.Targets.Controllers

Classes

[CurvedTargetController](#)

Controls an oscillating target, which takes two points and oscillates between them with a given frequency

[CustomMotionTargetController](#)

Controls a custom motion target, which loads sweep data and plays it back as a target

[LandmarkTargetController](#)

Controls a landmark target, which follows a given sensor

[OscillatingTargetController](#)

Controls an oscillating target, which takes two points and oscillates between them with a given frequency

[StaticTargetController](#)

Controls a target that stays in one spot

Enums

[CustomMotionTargetController.MotionPathStatus](#)

Defines the different statuses possible for the motion path. Initially [UNAVAILABLE](#), [PREPARING](#) while loading the path, and finally [READY](#) once successfully loaded, or back to [UNAVAILABLE](#) if unsuccessful

Class CurvedTargetController

Controls an oscillating target, which takes two points and oscillates between them with a given frequency

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- TargetController
- CurvedTargetController

Inherited Members

- TargetController.NUM_BASE_CONFIG_VALUES
- TargetController.description
- TargetController.targetId
- TargetController.mesh
- TargetController.configToggle
- TargetController.visibility
- TargetController.radius
- TargetController.trail
- TargetController.tooltip
- TargetController.GetTargetPosition(Int64)
- TargetController.GetCycleDuration()
- TargetController.ApplyConfigFromString(String)
- TargetController.UpdateTarget(Vector3, Nullable<SensorData>)
- TargetController.ToString()
- TargetController.OnOpen()
- TargetController.OnClose()

Namespace: [Optispeech.Targets.Controllers](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class CurvedTargetController : TargetController
```

Fields

endPosition

The second point to oscillate between

Declaration

```
public Vector3 endPosition
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

frequency

How many oscillations per second

Declaration

```
public float frequency
```

Field Value

TYPE	DESCRIPTION
System.Single	

startPosition

The first point to oscillate between

Declaration

```
public Vector3 startPosition
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

Methods

pointOnEllipse(Vector3, Vector3, Single)

Declaration

```
Vector3 pointOnEllipse(Vector3 center, Vector3 axes, float angle)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Vector3	center	
UnityEngine.Vector3	axes	
System.Single	angle	

Returns

TYPE	DESCRIPTION
UnityEngine.Vector3	

Class CustomMotionTargetController

Controls a custom motion target, which loads sweep data and plays it back as a target

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- TargetController
- CustomMotionTargetController

Inherited Members

- TargetController.NUM_BASE_CONFIG_VALUES
- TargetController.description
- TargetController.targetId
- TargetController.mesh
- TargetController.configToggle
- TargetController.visibility
- TargetController.radius
- TargetController.trail
- TargetController.tooltip
- TargetController.GetTargetPosition(Int64)
- TargetController.GetCycleDuration()
- TargetController.ApplyConfigFromString(String)
- TargetController.UpdateTarget(Vector3, Nullable<SensorData>)
- TargetController.ToString()
- TargetController.OnOpen()
- TargetController.OnClose()

Namespace: [Optispeech.Targets.Controllers](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class CustomMotionTargetController : TargetController
```

Fields

alternateDirection

Whether or not the loaded motion should loop, or "ping pong", where it plays back in reverse then normal again, and so on

Declaration

```
public bool alternateDirection
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

config

The active config, when its for this controller, to display the current motion path status

Declaration


```
[HideInInspector]
public CustomMotionTargetConfig config
```

Field Value

TYPE	DESCRIPTION
CustomMotionTargetConfig	

cycleDuration

The duration of the loaded sweep data

Declaration

```
long cycleDuration
```

Field Value

TYPE	DESCRIPTION
System.Int64	

frames

The data frames read from the loaded sweep data

Declaration

```
[HideInInspector]
public List<DataFrame> frames
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DataFrame>	

Remarks

Note: when alternateDirection is false, the last frame is ignored and only used to determine how long the second to last frame should last before looping back

id

The ID of the sensor from the sweep data to load the motion path from

Declaration

```
public int id
```

Field Value

TYPE	DESCRIPTION
System.Int32	

loading

Whether or not sweep data is currently being loaded

Declaration

```
[HideInInspector]  
public bool loading
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

motionPath

The file path of the sweep data to load the motion path from

Declaration

```
public string motionPath
```

Field Value

TYPE	DESCRIPTION
System.String	

offset

A static offset to add to the motion path

Declaration

```
public Vector3 offset
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

playbackSpeed

The speed at which to playback the motion data

Declaration

```
public float playbackSpeed
```

Field Value

TYPE	DESCRIPTION
System.Single	

prevFrame

The previous data frame, if it exists. Used for applying a low pass filter to the sweep data being read

Declaration

```
DataFrame? prevFrame
```

Field Value

TYPE	DESCRIPTION
System.Nullable< DataFrame >	

reader

The file reader used to read the sweep data. Stored because it can provide the sensor configuration to add the appropriate data into the ID dropdown

Declaration

```
[HideInInspector]
public FrameReader.FileReader reader
```

Field Value

TYPE	DESCRIPTION
FrameReader.FileReader	

status

The current status of the motion path

Declaration

```
[HideInInspector]
public CustomMotionTargetController.MotionPathStatus status
```

Field Value

TYPE	DESCRIPTION
CustomMotionTargetController.MotionPathStatus	

Methods

Finish(CustomMotionTargetController.MotionPathStatus)

Handles loading sweep data ending, either successfully or not

Declaration

```
void Finish(CustomMotionTargetController.MotionPathStatus status)
```

Parameters

TYPE	NAME	DESCRIPTION
CustomMotionTargetController.MotionPathStatus	status	The resulting status of the load attempt

LoadMotionPath(Boolean)

Attempts to load sweep data

Declaration

```
public void LoadMotionPath(bool resetId = true)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	resetId	Whether or not to set id to -1 before attempting to read the sweep data

StopLoading()

Stops loading any sweep data and resets motion path

Declaration

```
public void StopLoading()
```

Enum CustomMotionTargetController.MotionPathStatus

Defines the different statuses possible for the motion path. Initially [UNAVAILABLE](#), [PREPARING](#) while loading the path, and finally [READY](#) once successfully loaded, or back to [UNAVAILABLE](#) if unsuccessful

Namespace: [Optispeech.Targets.Controllers](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public enum MotionPathStatus
```

Fields

NAME	DESCRIPTION
PREPARING	
READY	
UNAVAILABLE	

Class LandmarkTargetController

Controls a landmark target, which follows a given sensor

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- TargetController
- LandmarkTargetController

Inherited Members

- TargetController.NUM_BASE_CONFIG_VALUES
- TargetController.description
- TargetController.targetId
- TargetController.mesh
- TargetController.configToggle
- TargetController.visibility
- TargetController.radius
- TargetController.trail
- TargetController.tooltip
- TargetController.GetTargetPosition(Int64)
- TargetController.GetCycleDuration()
- TargetController.ApplyConfigFromString(String)
- TargetController.UpdateTarget(Vector3, Nullable<SensorData>)
- TargetController.ToString()
- TargetController.OnOpen()
- TargetController.OnClose()

Namespace: [Optispeech.Targets.Controllers](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class LandmarkTargetController : TargetController
```

Fields

id

The ID of the sensor to use as the landmark

Declaration

```
public int id
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Class OscillatingTargetController

Controls an oscillating target, which takes two points and oscillates between them with a given frequency

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- TargetController
- OscillatingTargetController

Inherited Members

- TargetController.NUM_BASE_CONFIG_VALUES
- TargetController.description
- TargetController.targetId
- TargetController.mesh
- TargetController.configToggle
- TargetController.visibility
- TargetController.radius
- TargetController.trail
- TargetController.tooltip
- TargetController.GetTargetPosition(Int64)
- TargetController.GetCycleDuration()
- TargetController.ApplyConfigFromString(String)
- TargetController.UpdateTarget(Vector3, Nullable<SensorData>)
- TargetController.ToString()
- TargetController.OnOpen()
- TargetController.OnClose()

Namespace: [Optispeech.Targets.Controllers](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class OscillatingTargetController : TargetController
```

Fields

endPosition

The second point to oscillate between

Declaration

```
public Vector3 endPosition
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

frequency

How many oscillations per second

Declaration

```
public float frequency
```

Field Value

TYPE	DESCRIPTION
System.Single	

startPosition

The first point to oscillate between

Declaration

```
public Vector3 startPosition
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

Class StaticTargetController

Controls a target that stays in one spot

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- TargetController
- StaticTargetController

Inherited Members

- TargetController.NUM_BASE_CONFIG_VALUES
- TargetController.description
- TargetController.targetId
- TargetController.mesh
- TargetController.configToggle
- TargetController.visibility
- TargetController.radius
- TargetController.trail
- TargetController.tooltip
- TargetController.GetTargetPosition(Int64)
- TargetController.GetCycleDuration()
- TargetController.ApplyConfigFromString(String)
- TargetController.UpdateTarget(Vector3, Nullable<SensorData>)
- TargetController.ToString()
- TargetController.OnOpen()
- TargetController.OnClose()

Namespace: [Optispeech.Targets.Controllers](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class StaticTargetController : TargetController
```

Fields

position

The position of this target

Declaration

```
public Vector3 position
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

Namespace Optispeech.UI

Classes

Accordion

This class looks for TogglePanels inside it and handles keeping the panels appear stacked and automatically closes panels to fit within the height of the program

NumberField

Makes a label for a TMPro.TMP_InputField support dragging to change the input field's numeric value, simila to Unity's number fields

TogglePanel

This class goes on a Panel and allows us to toggle it open or closed by clicking on the titlebar. When closed only the titlebar will be visible

Tooltip

Creates a tooltip that appears when hovering over whatever this behaviour is attached to

Class Accordion

This class looks for TogglePanels inside it and handles keeping the panels appear stacked and automatically closes panels to fit within the height of the program

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
Accordion

Namespace: [Optispeech.UI](#)
Assembly: Assembly-CSharp.dll

Syntax

```
[RequireComponent(typeof(CanvasGroup))]  
public class Accordion : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler, IEventSystemHandler
```

Fields

cameraMoveTransparency

Affects how transparent the panels appear when the camera is being moved, where 1 is completely opaque and 0 is completely transparent

Declaration

```
[SerializeField]  
[Range(0F, 1F)]  
float cameraMoveTransparency
```

Field Value

TYPE	DESCRIPTION
System.Single	

cameraStillTransparency

Affects how transparent the panels appear when the camera is still, where 1 is completely opaque and 0 is completely transparent

Declaration

```
[SerializeField]  
[Range(0F, 1F)]  
float cameraStillTransparency
```

Field Value

TYPE	DESCRIPTION
System.Single	

canvas

Used for ensuring the accordion stays within the height of the screen

Declaration

Canvas canvas

Field Value

TYPE	DESCRIPTION
UnityEngine.Canvas	

canvasGroup

A canvas group attached to this game object to set the transparency of the whole accordion

Declaration

CanvasGroup canvasGroup

Field Value

TYPE	DESCRIPTION
UnityEngine.CanvasGroup	

isMouseOver

Flag that represents whether the mouse is currently over this accordion

Declaration

bool isMouseOver

Field Value

TYPE	DESCRIPTION
System.Boolean	

mouseOverTransparency

Affects how transparent the panels appear when the mouse is hovering over the accordion, where 1 is completely opaque and 0 is completely transparent

Declaration

```
[SerializeField]  
[Range(0F, 1F)]  
float mouseOverTransparency
```

Field Value

TYPE	DESCRIPTION
System.Single	

openPanelIndices

List of the indices of the currently open panels

Declaration

List<int> openPanelIndices

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	

panels

List of panels in this accordion

Declaration

[SerializeField] TogglePanel[] panels
--

Field Value

TYPE	DESCRIPTION
TogglePanel[]	

panelTransforms

List of each panels' transform, used for positioning

Declaration

[SerializeField] RectTransform[] panelTransforms

Field Value

TYPE	DESCRIPTION
UnityEngine.RectTransform[]	

remainingHeight

The height between the top of the accordion and the top of the screen

Declaration

float remainingHeight

Field Value

TYPE	DESCRIPTION
System.Single	

screenHeight

The height of the screen

Declaration

float screenHeight

Field Value

TYPE	DESCRIPTION
System.Single	

titleHeight

How tall the title of each panel is, used for positioning

Declaration

```
float titleHeight
```

Field Value

TYPE	DESCRIPTION
System.Single	

transitionSpeed

How quickly the transparency changes when changing between states, where 1 means moving from completely opaque to completely transparent in one frame

Declaration

```
[SerializeField]  
float transitionSpeed
```

Field Value

TYPE	DESCRIPTION
System.Single	

Methods

ChangeAlpha(Single)

Creates a coroutine to transition the accordion from the current transparency to the target

Declaration

```
IEnumerator ChangeAlpha(float target)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	target	The target transparency to transition to

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	A coroutine

EnsureAccordionHeight()

Ensures the height of the accordion is under the height of the screen by closing panels

Declaration

```
void EnsureAccordionHeight()
```

Class NumberField

Makes a label for a TmPro.TMP_InputField support dragging to change the input field's numeric value, simila to Unity's number fields

Inheritance

System.Object

UnityEngine.Object

UnityEngine.Component

UnityEngine.Behaviour

UnityEngine.MonoBehaviour

NumberField

Namespace: [Optispeech.UI](#)

Assembly: Assembly-CSharp.dll

Syntax

```
public class NumberField : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler, IBeginDragHandler, IDragHandler, IEndDragHandler, IEventSystemHandler
```

Fields

blocker

A gameobject that prevents input events from being passed to things except this class, while the label is being dragged

Declaration

```
GameObject blocker
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

inputField

The input field this label will control

Declaration

```
[SerializeField]  
TMP_InputField inputField
```

Field Value

TYPE	DESCRIPTION
TmPro.TMP_InputField	

isDragging

Whether or not the user is currently dragging the label

Declaration

```
bool isDragging
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

isHovering

Whether or not the user is currently hovering over the label

Declaration

```
bool isHovering
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

partial

Used to track fractional value when sliding integers

Declaration

```
float partial
```

Field Value

TYPE	DESCRIPTION
System.Single	

slider

The first child of [sliderCanvas](#), which will be moved to the mouse cursor each frame while [sliderCanvas](#) is active

Declaration

```
RectTransform slider
```

Field Value

TYPE	DESCRIPTION
UnityEngine.RectTransform	

sliderCanvas

A canvas that should be this gameObject's first child and contains an indicator that the label can be dragged

Declaration

```
Transform sliderCanvas
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

slideSpeed

How quickly the numeric value will change as the label is dragged

Declaration

```
[SerializeField]  
float slideSpeed
```

Field Value

TYPE	DESCRIPTION
System.Single	

Class TogglePanel

This class goes on a Panel and allows us to toggle it open or closed by clicking on the titlebar. When closed only the titlebar will be visible

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
TogglePanel

Namespace: [Optispeech.UI](#)
Assembly: Assembly-CSharp.dll

Syntax

```
[RequireComponent(typeof(RectTransform))]  
[RequireComponent(typeof(ContentSizeFitter))]  
public class TogglePanel : MonoBehaviour
```

Fields

fitter

Used to find the preferred size of the panel when its open

Declaration

```
ContentSizeFitter fitter
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.ContentSizeFitter	

heightChangeEvent

Event that fires whenever the height of the panel changes. Passes two floats: the current height and then the new height

Declaration

```
public UnityEvent<float, float> heightChangeEvent
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent<System.Single, System.Single>	

isOpen

Whether or not this panel is currently open

Declaration

```
[HideInInspector]  
public bool isOpen
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

maxHeight

The maximum height this panel is allowed to be

Declaration

<code>float</code> maxHeight

Field Value

TYPE	DESCRIPTION
System.Single	

panelToggledEvent

Event that fires whenever the panel is toggled open or closed. Passed whether or not the panel is now open, and the amount the height has changed

Declaration

<code>public</code> UnityEvent< <code>bool</code> , <code>float</code> > panelToggledEvent
--

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent<System.Boolean, System.Single>	

rect

This panel's rect transform, for setting its size

Declaration

<code>RectTransform</code> rect

Field Value

TYPE	DESCRIPTION
UnityEngine.RectTransform	

startOpen

Whether or not this panel should be open initially

Declaration

<code>public</code> <code>bool</code> startOpen

Field Value

TYPE	DESCRIPTION
System.Boolean	

titleBar

This is the titlebar that'll act as the toggle for the panel The panel will shrink to the titlebar's size

Declaration

```
public Button titleBar
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.Button	

titleRect

This panel's title bar, for getting its size

Declaration

```
RectTransform titleRect
```

Field Value

TYPE	DESCRIPTION
UnityEngine.RectTransform	

transitionSpeed

How many canvas pixels to move every frame

Declaration

```
[SerializeField]  
float transitionSpeed
```

Field Value

TYPE	DESCRIPTION
System.Single	

Methods

AnimateHeightTo(Single)

Creates a coroutine that animates the height of this panel to the target height

Declaration

```
IEnumerator AnimateHeightTo(float height)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	height	The height to transition to

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	A coroutine

Refresh()

Recalculates preferred height and transitions to that height

Declaration

```
public void Refresh()
```

SetMaxHeight(Single)

Changes the maximum height the panel can reach

Declaration

```
public void SetMaxHeight(float maxHeight)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	maxHeight	The new upper limit

Toggle()

Toggles whether this panel is open or not and transitions to the new height

Declaration

```
public void Toggle()
```

Class Tooltip

Creates a tooltip that appears when hovering over whatever this behaviour is attached to

Inheritance

- System.Object
- UnityEngine.Object
- UnityEngine.Component
- UnityEngine.Behaviour
- UnityEngine.MonoBehaviour
- Tooltip

Namespace: [Optispeech.UI](#)
Assembly: Assembly-CSharp.dll

Syntax

```
public class Tooltip : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler, IEventSystemHandler
```

Fields

showTooltip

Whether or not the tooltip is currently visible

Declaration

```
bool showTooltip
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

text

The label to display the tooltip's text

Declaration

```
TextMeshProUGUI text
```

Field Value

TYPE	DESCRIPTION
TMPPro.TextMeshProUGUI	

tooltip

The first child of [tooltipCanvas](#), which will be placed at the mouse's position every frame

Declaration

```
RectTransform tooltip
```

Field Value

TYPE	DESCRIPTION
UnityEngine.RectTransform	

tooltipCanvas

The first child of this gameObject, which will only be visible on hover

Declaration

Transform tooltipCanvas

Field Value

TYPE	DESCRIPTION
UnityEngine.Transform	

Methods

SetText(String)

Change the text that appears in this tooltip

Declaration

public void SetText(string text)

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The new tooltip text