

 jkammerl / cardboard-unity

Public

forked from googlevr/gvr-unity-sdk

<> Code

 Pull requests

 Actions

 Projects

 Wiki

 Security

 Insights

 master ▾

...

cardboard-unity / Cardboard / Editor / CardboardAudioSourceEditor.cs / <> Jump to ▾

 smdol Version 0.6

History

1 contributor

186 lines (163 sloc) | 8.29 KB

1

// Copyright 2015 Google Inc. All rights reserved.

2

//

3

// Licensed under the Apache License, Version 2.0 (the "License");

4

// you may not use this file except in compliance with the License.

5

// You may obtain a copy of the License at

6

//

7

// http://www.apache.org/licenses/LICENSE-2.0

8

//

9

// Unless required by applicable law or agreed to in writing, software

10

// distributed under the License is distributed on an "AS IS" BASIS,

11

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

12

// See the License for the specific language governing permissions and

13

// limitations under the License.

14

15

using UnityEngine;

16

using UnityEditor;

17

using System.Collections;

18

19

/// A custom editor for properties on the CardboardAudioSource script. This appears in the Inspector

20

/// window of a CardboardAudioSource object.

21

[CustomEditor(typeof(CardboardAudioSource))]

22

[CanEditMultipleObjects]

23

public class CardboardAudioSourceEditor : Editor {

24

private SerializedProperty clip = null;

25

private SerializedProperty loop = null;

26

private SerializedProperty mute = null;

27

private SerializedProperty pitch = null;

28

private SerializedProperty playOnAwake = null;

29

private SerializedProperty volume = null;

30

private SerializedProperty rolloffMode = null;

31

private SerializedProperty maxDistance = null;

32

private SerializedProperty minDistance = null;

33

private SerializedProperty directivityAlpha = null;

34

private SerializedProperty directivitySharpness = null;

```
35 private Texture2D directivityTexture = null;
36 private SerializedProperty gainDb = null;
37 private SerializedProperty hrtfEnabled = null;
38 private SerializedProperty occlusionEnabled = null;
39
40 private GUIContent clipLabel = new GUIContent("AudioClip",
41     "The AudioClip asset played by the CardboardAudioSource.");
42 private GUIContent loopLabel = new GUIContent("Loop",
43     "Sets the source to loop.");
44 private GUIContent muteLabel = new GUIContent("Mute",
45     "Mutes the sound.");
46 private GUIContent pitchLabel = new GUIContent("Pitch",
47     "Sets the frequency of the sound. Use this to slow down or speed up the sound.");
48 private GUIContent volumeLabel = new GUIContent("Volume",
49     "Sets the overall volume of the sound.");
50 private GUIContent rolloffModeLabel = new GUIContent("Volume Rolloff",
51     "Which type of rolloff curve to use.");
52 private GUIContent maxDistanceLabel = new GUIContent("Max Distance",
53     "Max distance is the distance a sound stops attenuating at.");
54 private GUIContent minDistanceLabel = new GUIContent("Min Distance",
55     "Within the min distance, the volume will stay at the loudest possible. " +
56     "Outside this min distance it will begin to attenuate.");
57 private GUIContent playOnAwakeLabel = new GUIContent("Play On Awake",
58     "Play the sound when the scene loads.");
59 private GUIContent directivityLabel = new GUIContent("Directivity",
60     "Controls the pattern of sound emission of the source. This can change the perceived " +
61     "loudness of the source depending on which way it is facing relative to the listener. " +
62     "Patterns are aligned to the 'forward' direction of the parent object.");
63 private GUIContent directivityAlphaLabel = new GUIContent("Alpha",
64     "Controls the balance between dipole pattern and omnidirectional pattern for source " +
65     "emission. By varying this value, differing directivity patterns can be formed.");
66 private GUIContent directivitySharpnessLabel = new GUIContent("Sharpness",
67     "Sets the sharpness of the directivity pattern. Higher values will result in increased " +
68     "directivity.");
69 private GUIContent gainLabel = new GUIContent("Gain (dB)",
70     "Applies a gain to the source for adjustment of relative loudness.");
71 private GUIContent hrtfEnabledLabel = new GUIContent("Enable HRTF",
72     "Sets HRTF binaural rendering for the source. Note that this setting has no effect when " +
73     "global low quality mode is selected.");
74 private GUIContent occlusionLabel = new GUIContent("Enable Occlusion",
75     "Sets whether the sound of the source should be occluded when passing through objects " +
76     "between the source and the listener.");
77
78 void OnEnable () {
79     clip = serializedObject.FindProperty("sourceClip");
80     loop = serializedObject.FindProperty("sourceLoop");
81     mute = serializedObject.FindProperty("sourceMute");
82     pitch = serializedObject.FindProperty("sourcePitch");
83     playOnAwake = serializedObject.FindProperty("playOnAwake");
84     volume = serializedObject.FindProperty("sourceVolume");
85     rolloffMode = serializedObject.FindProperty("rolloffMode");
86     maxDistance = serializedObject.FindProperty("sourceMaxDistance");
87     minDistance = serializedObject.FindProperty("sourceMinDistance");
88     directivityAlpha = serializedObject.FindProperty("directivityAlpha");
```

```
89     directivitySharpness = serializedObject.FindProperty("directivitySharpness");
90 #if UNITY_4_5
91     directivityTexture = Texture2D.CreateExternalTexture(0, 0, TextureFormat.ARGB32, false, false,
92                                                         System.IntPtr.Zero);
93 #else
94     directivityTexture = Texture2D.blackTexture;
95 #endif
96     gainDb = serializedObject.FindProperty("gainDb");
97     hrtfEnabled = serializedObject.FindProperty("hrtfEnabled");
98     occlusionEnabled = serializedObject.FindProperty("occlusionEnabled");
99 }
100
101 /// @cond
102 public override void OnInspectorGUI () {
103     serializedObject.Update();
104
105     EditorGUILayout.PropertyField(clip, clipLabel);
106
107     EditorGUILayout.Separator();
108
109     EditorGUILayout.PropertyField(mute, muteLabel);
110     EditorGUILayout.PropertyField(playOnAwake, playOnAwakeLabel);
111     EditorGUILayout.PropertyField(loop, loopLabel);
112
113     EditorGUILayout.Separator();
114
115     EditorGUILayout.PropertyField(volume, volumeLabel);
116
117     EditorGUILayout.Separator();
118
119     EditorGUILayout.PropertyField(pitch, pitchLabel);
120
121     EditorGUILayout.Separator();
122
123     EditorGUILayout.Slider(gainDb, CardboardAudio.minGainDb, CardboardAudio.maxGainDb, gainLabel);
124
125     EditorGUILayout.Separator();
126
127     EditorGUILayout.PropertyField(rolloffMode, rolloffModeLabel);
128     ++EditorGUI.indentLevel;
129     EditorGUILayout.PropertyField(minDistance, minDistanceLabel);
130     EditorGUILayout.PropertyField(maxDistance, maxDistanceLabel);
131     --EditorGUI.indentLevel;
132     if (rolloffMode.enumValueIndex == (int)AudioRolloffMode.Custom) {
133         EditorGUILayout.HelpBox("Custom rolloff mode is not supported.", MessageType.Error);
134     }
135
136     EditorGUILayout.Separator();
137
138     EditorGUILayout.BeginHorizontal();
139     EditorGUILayout.BeginVertical();
140     GUILayout.Label(directivityLabel);
141     ++EditorGUI.indentLevel;
142     EditorGUILayout.Slider(directivityAlpha, 0.0f, 1.0f, directivityAlphaLabel);
```

```
143 EditorGUILayout.Slider(directionalitySharpness, 1.0f, 10.0f, directionalitySharpnessLabel);
144 --EditorGUI.indentLevel;
145 EditorGUILayout.EndVertical();
146 DrawDirectivityPattern((int)(3.0f * EditorGUIUtility.singleLineHeight));
147 EditorGUILayout.EndHorizontal();
148 EditorGUILayout.PropertyField(occlusionEnabled, occlusionLabel);
149
150 EditorGUILayout.Separator();
151
152 // HRTF toggle can only be modified through the Inspector in Edit mode.
153 GUI.enabled = !EditorApplication.isPlaying;
154 EditorGUILayout.PropertyField(hrtfEnabled, hrtfEnabledLabel);
155 GUI.enabled = true;
156
157 serializedObject.ApplyModifiedProperties();
158 }
159 /// @endcond
160
161 private void DrawDirectivityPattern (int size) {
162     directivityTexture.Resize(size, size);
163     // Draw the axes.
164     Color axisColor = 0.5f * Color.black;
165     for (int i = 0; i < size; ++i) {
166         directivityTexture.SetPixel(i, size / 2, axisColor);
167         directivityTexture.SetPixel(size / 2, i, axisColor);
168     }
169     // Draw the 2D polar directivity pattern.
170     Color cardioidColor = 0.75f * Color.blue;
171     float offset = 0.5f * size;
172     float cardioidSize = 0.45f * size;
173     Vector2[] vertices = CardboardAudio.Generate2dPolarPattern(directionalityAlpha.floatValue,
174                                                                 directivitySharpness.floatValue,
175                                                                 180);
176     for (int i = 0; i < vertices.Length; ++i) {
177         directivityTexture.SetPixel((int)(offset + cardioidSize * vertices[i].x),
178                                     (int)(offset + cardioidSize * vertices[i].y),
179                                     cardioidColor);
180     }
181     directivityTexture.Apply();
182     // Show the texture.
183     GUILayout.Box(directivityTexture);
184 }
185
186 }
```