Audio Visualizer Tool

Dog Eat Dog Games

Support and more stuff at: [www.dogeatdoggames.com](http://www.dogeatdoggames.com)

Like us: [www.facebook.com/dogeatdoggames](http://www.facebook.com/dogeatdoggames)

Follow us: [www.twitter.com/dogeatdoggames](http://www.twitter.com/dogeatdoggames)

Tutorials/Videos: <https://www.youtube.com/watch?v=toKinLCuOhU&list=PLc2O4sFLm5sTMFejOz20XMo0nvrHTMjbJ>

Contents

What you can do

Quick start

Scenes

Script References

References

License info

**Thank you for your purchase!**

**What you can do**

Add audio visualization effects to your game/application.

* Use the beat to:
  + Call any custom event of your own!
  + Move objects back and forth.
  + Scale/shrink objects.
  + Fade between two materials.
  + Fade between two colors.
* Show audio waveforms
  + Use Unity’s new UI system to display waveforms on panels.
  + Arc Waveform
  + Line Renderer waveforms.
  + Pad waveforms.
  + Circular waveforms.
  + Spherical waveforms.
  + Object position waveforms.
  + Object scale waveforms.
* New Features (v2.0)
  + Pre-record tracks for
    - Beat anticipation
    - Improved Performance
  + Use live microphone input to drive beat detection and audio waveforms.
* New Features (v2.1)
  + Runtime beat anticipation. Make games like rockband in runtime.
  + Significantly reduced file size for audio recordings.

**Quick Start**

1. Open AudioWaveforms > Scenes. Run each scene to see the different things you can do!
2. Replace music in each scene with your own audio clips.
   1. In the heirarchy window you should see a gamobject called “AudioSamples” Replace the audioclip on the AudioSource component with your own music.
3. Adjust AudioListener > AudioEventListener parameters until you’re happy with the results (see script references under AudioEventListener for more details)

**Scenes**

* ArcWaveform
  + A waveform using sprites placed around in a circular arc.
* AudioPanel
  + A Unity 4.6 UI style canvas, with waveforms on it. Affected by volume.
* Beat Anticipation
  + Anticipate beats using a pre-recorded track.
* Beat Based Game – Pre-Recorded
  + Use pre-recorded tracks to create a beat based game like RockBand or AudioSurf.
* Beat Based Game – Runtime
  + Use runtime tracks to create a beat based game like RockBand or AudioSurf.
* Beat Detection
  + Examples of how to detect those beats!
* Circle
  + A circular waveform with multiple effects reacting to the music in different ways.
* City
  + A city that comes to live with the music.
* DiscoBall
  + Multiple waveform examples that react to the music in different ways.
* Lights
  + Control light intensity and color with the audio.
* MicrophoneInput
  + Use live microphone input to drive waveforms and beat detection.
* Object Rotation Waveform
  + Rotate objects based on the audio.
* Rainbow
  + A combination of pad waveforms and line waveforms.
* Record Audio
  + Record frequency and beat data from tracks. Use the output files for beat anticipation, or improved performance.
* Sidescroller
  + A scrolling waveform that could be used as a background.
* Sphere
  + A spherical group of waveforms that react to the music in different ways.

**Script References – Core**

These are the main scripts used to create audio waveforms.

* ArcWaveform
  + Displays a circular waveform, made up of sprites.
  + Parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - FrequencyRange - the frequency range of the audio we’re sampling.
    - sensitivity  - how sensitive is this script to the audio.
    - sprite - sprite to use for each cell in the waveform.
    - Angle – the angle of the arc, 0-360 degrees.
    - Radius – the radius of the circular arc.
    - Height – the amplitude of the waveform.
    - numColumns – number of columns in the waveform.
    - numRows - number of rows in the waveform.
    - spacingX - spacing between columns.
    - spacingY - spacing between rows.
    - bottomColor - color of sprites at the bottom, when audio levels are low.
    - topColor - color of sprites at the top, when audio levels are high.
    - lerpSpeed - how fast the waveform moves.
    - Use audio file – should we ready from an audio file?
    - Abs – absolute value, if true we will only get positive values.
* AudioData
  + An audio data object, for storing pre-analyzed track data.
  + Use AudioRecorder.cs to fill in this data object.
  + Parameters
    - clipLength – the length of the audio clip that was recorded
    - bufferSize – the size of each recorded sample array
    - abs – was this data recorded with the absolute value flag?
    - List<T> - lists used to store recorded beat and frequency data.
  + Methods
    - AudioData (AudioRecorder recorder, float clipLength)
      * Constructor for the class.
    - RecordBeat(float time, float volume)
      * Records a beat into the List<Beat> beats parameter.
    - RecordSamples(FrequencyRange freqRange, float[] data)
      * Record the passed in “data” that belongs to the passed in “freqRange”
    - GetSamples(FrequencyRange freqRange,float time)
      * Grab samples out of the data container, belonging to the given freqRange and at a given time into the track.
    - GetSampleArray(FrequencyRange freqRange)
      * Grab the entire sample matrix in a given frequency range.
* AudioEventListener
  + Listens to the beat, calls public method in the public OnBeat event.
  + parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - preBeatOffset - OnBeat events will trigger "preBeatOffset" seconds before the beat occurs, used for beat anticipation. Use this in conjuction with “SilentAudio” in the AudioSampler.
    - FrequencyRange - the frequency range of the audio we’re sampling.
    - Sample Buffer Size – buffer this many audio samples, used for beat detection.
    - Beat Threshold - adjusted per song. Lower if you’re not receiving events, raise if you’re receiving too many events.
    - Automatic Threshold - automatically adjust beat threshold by tracking audio from the last “sampleBuffer” frames.
    - Beat Limiter - consecutive beats below this time limit will not be regsitered
    - Automatic Limiter – automatically adjust the beat limiter for good results.
    - OnBeat() - public UnityEvents can get added here, and are called when a beat is detected.
    - OnFrequencyChanged
      * OnChange – hook in public dynamic float variables here. These values will be changed according to the audio frequency.
      * Min/Max value. Every float hooked in to the OnChange listener, will be changed between these min/max values according to the audio frequency.
* AudioFileEventListener
  + Similar to AudioEventListener, but uses pre-recorded tracks instead of live ones.
  + Parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - preBeatOffset - OnBeat events will trigger "preBeatOffset" seconds before the beat occurs, used for beat anticipation.
    - frequencyRange – the frequency range you’re sampling from. Note the audioFile you reference needs to have that frequency data recorded. See AudioRecorder for more info.
    - OnBeatRecognized – a static beat event that classes can subscribe to in order to call a method every time a beat is recognized.
* AudioRecorder
  + Records tracks into serialized json files that can later be used for beat anticipation or improved performance.
  + Parameters
    - audioFileName - the name of the file we'll record data into.
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample.
    - Ranges – a list of frequency ranges we want to record data for.
    - sampleBufferSize – the size of each sample array recorded every frame.
    - recordBeats – flag indicating if we’re currently recording or not
    - abs – flag indicating if we record Mathf.Abs() of each value or not.
    - Debug – display debug info during recording.
  + Methods
    - RecordBeat()
      * Record a beat into the data file, this is called from AudioEventListener’s “BeatDetected” event.
    - RecordSamples()
      * Records samples from the frequencyRanges in the “ranges” list into the audio file.
* AudioSampler
  + A singleton instance that samples the audio.
  + parameters
    - instance - public static instance of the Audio Sampler
    - Audio Sources - list of audio sources that you want to sample.
      * By default this will grab an AudioSource attached to the same GameObject. This allows easier setup if you just have one audio source you want to sample.
      * If you want multiple audio sources just add them to the list here.
    - Audio Files – list of pre recorded audio files that will be used to drive waveforms instead of during runtime.
    - SilentAudio – used for runtime beat anticipation. This audio is muted, beat detection is run on the muted audio. Other audio is played with a delay so you can trigger future events to be in sync with the music. This delay is set up using the “pre beat offset” parameter on the AudioEventListener and AudioFileEventListener.
      * Audio Sources – list of sources you want to be samples for beat detection.
      * Silent Mixer Group – AudioMixerGroup used to mute the audio sources. Use the “SilentMixer” mixer group provided under the AudioVisualizerFolder.
      * UseSilentAudio – flag indicating whether or not silent audio should be used.
    - Debug - if true, shows audio data being sampled.
  + methods
    - GetAudioSamples(int audioSourceIndex)
      * AudioSourceIndex - which audio data in the AudioSampler are you getting samples from.
      * returns a float[] of the samples taken (multiplied by the audio volume)
    - GetAudioSamples(int audioSourceIndex, int numBins, bool absoluteVal)
      * Like the above method, but returns an array of size ‘numBins’, and potentially takes the absolute value of each sample.
    - GetAvg(int audioSourceIndex, int numSamples, float sensitivity, bool abs)
      * AudioSourceIndex - see above
      * NumSamples - see above
      * Sensitivity - multiplied by the average
      * abs - use absolute value of samples or not (decibal levels samples can be positive or negative).
    - GetRMS() - root means squared
    - GetInstantEnergy() - square and sum audio samples.
    - GetFrequencyVol() - get current volume, within a given frequency range.
    - GetFrequencyData() - return the raw spectrum data in the given frequency range.
    - GetFreqForRange() - return the frequency range values to listen for, with the passed in enum.
* CircleWaveform - move objects in a circle, and in and out using the music.
  + Moves objects in a circle, and up and down with the music.
  + parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - FrequencyRange - the frequency range of the audio we’re sampling.
    - sensitivity - how sensitive is this script to the audio.
    - objects - The objects you’re going to move around in a circle. Objects should exist in the scene. Typically these are objects with trail renderers and particle systems.
    - rotationSpeed - how fast should the objects rotate, value can be negative.
    - radius - radius of the circle.
    - lerpSpeed - lerp speed related to movement around the circle.
    - useWaveform - move up and down relative to the waveform of the music.
  + methods
    - Boost(mulitplier) - for .1 seconds, boost the rotationSpeed by the passed in multiplier
    - Bump(bool switchSign) - Get the avg decibal level of the audio, and move the radius to equal startRadius\*avg. If ‘switchSign’ is true, the sign of the radius we bump to, will switch between + and -.
* ColorChange - change a material’s colors based on the music.
  + parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - FrequencyRange - the frequency range of the audio we’re sampling.
    - lowColor - when music decibal level is low, material is this color.
    - highColor - when music decibal level is high, material is this color.
    - sensitivity  - how sensitive is this script to the audio.
    - lerpSpeed - rate of color change.
* CurveWaveform – Child of LineWaveform: display an audio waveform using a line renderer, and an input curve.
* LineWaveform - display the waveform using a line renderer.
  + parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - FrequencyRange - the frequency range of the audio we’re sampling.
    - points - draw a line between each of these points in order.
    - lineAtt - lineRenderer attributes, like color, width, material, etc.
    - amplitude - height of the waveform.
    - Gizmos size – how big is the gizmos sphere drawn in the Scene view around each point.
    - abs - take the abosulte value of audio samples.
    - OrientPoints() – make each point look at the next point in the list.
    - RenamePoints – rename and number all the points in our points list.
* MaterialChange - lerp between two materials, using the music. (BlendTex shader required)
  + parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - FrequencyRange - the frequency range of the audio we’re sampling.
    - sensitivity  - how sensitive is this script to the audio.
    - Note: you don’t need lowMat/highMat if the gameobject has a \_Blend attribute in it’s material.
      * lowMat - when music decibal level is low, use this material.
      * highMat - when music decibal level is high, use this material.
    - lerpSpeed - rate of material change.
* MicrophoneInput – streams microphone data into an AudioSource component.
  + Parameters
    - currentAudioInput – name of the microphone device being used.
    - DeviceNum – index into the array of microphone that are detected on the system. Use this to determine which microphone is used.
* Object Position Waveform - move objects up and down, to create a waveform.
  + parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - FrequencyRange - the frequency range of the audio we’re sampling.
    - objects - objects to move up and down.
    - positionAxis - move the objects along this axis.
    - maxHeight - move objects to this max height.
    - sensitivity  - how sensitive is this script to the audio.
    - lerpSpeed - rate of movement.
    - absoluteVal - take the abosulte value of audio samples.
* Object Rotation Waveform – move objects to the music.
  + - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - FrequencyRange - the frequency range of the audio we’re sampling.
    - sensitivity  - how sensitive is this script to the audio.
    - objects - objects to rotate
    - rotationAxis – rotate the objects around this axis
    - minAngle – minimum rotation
    - maxAngle – maximum rotation.
    - lerpSpeed - rate of movement.
    - LocalRotation – rotate each object around it’s own local music?
      * If false, the transform of the ObjectRotationWaveform script is used as the pivot point.
    - RandomAxis – if true, rotation axis is ignored and each objects get’s it’s own random rotation axis.
    - Use Audio File – use a pre-recorded audio file?
* Object Scale Waveform - scale objects to create a waveform.
  + parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - FrequencyRange - the frequency range of the audio we’re sampling.
    - sensitivity  - how sensitive is this script to the audio.
    - objects - objects to scale.
    - scaleAxis - scalethe objects along this axis.
    - maxHeight - move objects to this max height.
    - lerpSpeed - rate of scaling.
    - absoluteVal - take the abosulte value of audio samples.
* Pad Waveform - a 3D waveform made of line-renderers in concentric rings.
  + parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - numLines - number of lines/rings on the pad.
    - radius - radius of the pad.
    - maxHeight - max height of the pad effects, either ripples or bounces.
    - updateRate - how often the pad effects are updated. Once every ‘updateRate’ frames.
    - rippleColor - color of the ripple waves.
    - rippleWidth – how many lines are in each ripple. Typically 3-5.
    - lineAttributes - lineRenderer attributes, like color, width, material, etc.
    - padType
      * Ripple – animate the inner ring. This state is typically paired with SendRipple() method, which can be called from an AudioEventListener.
      * DampWave - wave played across pad, damped by distance.
      * Wave - wave across the pad.
      * Bounce - bounce rings up and down.
  + Methods
    - SendRipple(float propegationTime) – send a ripple down the pad, that takes “propegationTime” to reach the end of the pad. The ripple height will be determined by “maxHeight” and the current audio frequency.
* Panel Waveform - display a waveform using sprites on a UI panel.
  + parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - FrequencyRange - the frequency range of the audio we’re sampling.
    - sensitivity  - how sensitive is this script to the audio.
    - sprite - sprite to use for each cell in the waveform.
    - numColumns - columns of the waveform.
    - numRows - rows of the waveform.
    - spacingX - spacing between columns.
    - spacingY - spacing between rows.
    - bottomColor - color of sprites at the bottom, when audio levels are low.
    - topColor - color of sprites at the top, when audio levels are high.
    - lerpSpeed - how fast the waveform moves.
    - Use audio file – should we ready from an audio file?
* Sphere Waveform - similar to circle waveform, but with a sphere! Move objects around a sphere.
  + parameters
    - audioIndex - index into audioSampler audioSources or audioFiles list. Determines which audio source we want to sample
    - FrequencyRange - the frequency range of the audio we’re sampling.
    - sensitivity  - how sensitive is this script to the audio.
    - objects - objects to move around a sphere.
    - rotationSpeed - speed at which objects are rotated around the sphere.
    - rotationAxis - axis of rotation.
    - radius - radius of sphere
    - lerpSpeed - rate of scaling
    - useWaveform - move the radius of this object up and down relative to the music.
    - rotationType
      * Uniform - rotate around rotation axis
      * Rand - rotate around a random axis
      * Cross - use a cross product of this objects position to center, cross the rotation axis.
  + methods
    - Boost(mulitplier) - for .1 seconds, boost the rotationSpeed by the passed in multiplier
    - Bump(bool switchSign) - Get the avg decibal level of the audio, and move the radius to equal startRadius\*avg. If ‘switchSign’ is true, the sign of the radius we bump to, will switch between + and -.

**Script References – Miscellaneous**

These are small scripts used in the demo scenes.

* CameraCircle - rotate the camera around a target
  + parameters
    - target - transform we rotate around
    - rotationSpeed - speed of rotaion
    - rotaitonAxis - axis of rotation
* CameraMovement - Moves the camera right in the Sidescroller scene.
  + parameters
    - speed - movement speed
    - lerpSpeed - lerp between current and desired position at this rate
* Object Circle- place objects evenly in a circle’s radius.
  + parameters
    - objectsToplace - objects to move around a sphere, typically particles or objects with trail renderers.
    - radius - radius of the sphere.
* Object Sphere - place objects evenly in a sphere’s radius.
  + parameters
    - objectsToplace - objects to move around a sphere, typically particles or objects with trail renderers.
    - radius - radius of the sphere.
* Particle Controller - call particle system.play at a given rate
  + parameters
    - particleSystems - the particle systems we want to use.
    - updateRate - how often effects are played. Once every ‘updateRate’ frames.
* Rotate - rotate this object out of it’s up axis.
  + parameters
    - speed - rotation rate.

**Tutorials/Videos**

<https://www.youtube.com/watch?v=toKinLCuOhU&list=PLc2O4sFLm5sTMFejOz20XMo0nvrHTMjbJ>

**Credits**

Programming and Effects: Kurt Hollowell

Audio: Austin Williams, Devin Williams

BlendTexture shaders:<http://wiki.unity3d.com/index.php?title=Blend_2_Textures>

City Model: <http://www.turbosquid.com/3d-models/cartoony-buildings-max-free/730644>

**License Info**

Subject to Unity’s EULA: <http://unity3d.com/legal/as_terms>

Music composed, arranged, and produced by:

Austin Williams. © 2015 Austin Williams

* A Restful Town
* Abandoned Places
* Celestial Visitor

Devin Williams

* Family Fugh

Bensound [**http://www.bensound.com**](http://www.bensound.com/)

* Bensound-energy

Bendsound-popdance