Sound with Cinder

Andrea Cuius

andrea@nocte.co.uk
www.nocte.co.uk
@q_says / @noctestudio

nocte

Cinder APIs

Cinder offers built-in support for sound, the APIs allow to playback a track, generate sound, interface with the system audio device and perform basic audio analysis.

The current APIs are fully supported on OSX, but not on Windows. Luckily there is a new cross-platform CinderBlock available for alpha testing.

Cinder-Audio2

https://github.com/richardeakin/Cinder-Audio2

Cinder APIs - load & play

```
#include "cinder/audio/Io.h"
#include "cinder/audio/Output.h"

class BasicApp : public AppNative {
	...
	audio::TrackRef			mTrack;
}

// load and add(play) a track, this call immediately play the sound
void BasicApp::setup() {
	mTrack = audio::Output::addTrack( audio::load( loadAsset( “test.mp3” ) ) );
}
```

Cinder APIs - load, then play

```
class BasicApp : public AppNative {
	...
	audio::SourceRef	mAudioSource;
}

void BasicApp::setup() {
	mAudioSource = audio::load( loadAsset( “test.mp3” ) );	// only load the sound
}

void AudioPlaybackApp::mouseDown( MouseEvent event )
{
	audio::Output::play( mAudioSource );			// play the sound, how to stop it?
}
```

Cinder APIs - load, play and stop

```
class BasicApp : public AppNative {
	...
	audio::SourceRef	mAudioSource;
	audio::TrackRef		mTrack;
}

void BasicApp::setup() {
	mAudioSource	= audio::load( loadAsset( “test.mp3” ) );		// load the sound
	mTrackRef		= ci::audio::Output::addTrack( mAudioRef, false );	// create a track
}
```

Cinder APIs - load, play and stop

```
...

void AudioPlaybackApp::mouseDown( MouseEvent event )
{
	if ( mTrackRef->isPlaying() )
		mTrackRef->stop();
	else
		mTrackRef->play();
}
```

Cinder APIs - sound analysis

```
...
#include "cinder/audio/FftProcessor.h"
#include "cinder/audio/PcmBuffer.h"

class BasicApp : public AppNative {
	...
	audio::TrackRef			mTrack;
	audio::PcmBuffer32fRef		mPcmBuffer;
}

void BasicApp::setup() {
	mTrack = audio::Output::addTrack( audio::load( loadAsset( “test.mp3” ) ) );
}
```

Cinder APIs - sound analysis PCM

```
void BasicApp::drawPcm() {
	audio::Buffer32fRef	leftBuffer, rightBuffer;
	// get the PCM buffer for right and left channel
	leftBuffer	= mPcmBuffer->getChannelData( audio::CHANNEL_FRONT_LEFT );
	rightBuffer	= mPcmBuffer->getChannelData( audio::CHANNEL_FRONT_RIGHT );

	for( int i = 0; i < PcmBuffer->getSampleCount(); i++ ) {
		float leftValue = leftBuffer->mData[i];
		float rightValue = leftBuffer->mData[i];
		…
	}
}
```

Cinder APIs - sound analysis FFT

```
void BasicApp::drawFft() {
	audio::Buffer32fRef	leftBuffer;

	// get the left channel buffer
	leftBuffer = mPcmBuffer->getChannelData( audio::CHANNEL_FRONT_LEFT );

	// calculate the Fft
	std::shared_ptr<float> fftRef = audio::calculateFft( leftBuffer, bandCount );

	for( int i = 0; i < ( bandCount ); i++ ) {	//draw the bands
		float val = fftRef.get()[i];
		…
```

Cinder APIs - sound input

```
#include "cinder/audio/Input.h"

audio::Input mInput;
const std::vector<audio::InputDeviceRef>& devices = audio::Input::getDevices();

for( size_t k=0; devices.size(); k++ )
        console() << devices[k]->getName() << std::endl;

mInput = audio::Input();                    //initialize the audio Input, using the default input device
mInput.start();                             //tell the input to start capturing audio

mPcmBuffer = mInput.getPcmBuffer();
```

Cinder-Audio2 - Nodes

```
auto ctx      = audio2::Context::master();          // a Context is required for making new audio Node's.
mGen          = ctx->makeNode( new audio2::GenSine );
mGain         = ctx->makeNode( new audio2::Gain );

mGen->setFreq( 220 );
mGain->setValue( 0.5f );

mGen >> mGain >> ctx->getOutput();    // connections can be made this way or with connect().
mGen->start();                        // Node's need to be enabled to process audio

ctx->start();                         // Context also must be started
```

Cinder-Audio2 - Play a track

```
#include "cinder/audio2/Voice.h"
#include "cinder/audio2/Source.h"

audio2::VoiceRef mVoice;
mVoice = audio2::Voice::create( audio2::load( loadAsset( “test.mp3” ) ) );

mVoice->setVolume( volume );
mVoice->setPan( pan );

if( mVoice->isPlaying() )
	mVoice->stop();
else
	mVoice->play();
```

Cinder-Audio2 - Input Analyzer

```
audio2::LineInRef           mLineIn;
audio2::ScopeSpectralRef    mScopeSpectral;

auto ctx                    = audio2::Context::master();
mLineIn                     = ctx->createLineIn();                          // get the default line-in
auto scopeFmt               = audio2::ScopeSpectral::Format().fftSize( 2048 ).windowSize( 1024 );
mScopeSpectral = ctx->makeNode( new audio2::ScopeSpectral( scopeFmt ) );

mLineIn >> mScopeSpectral;                  // scope spectral is the end point to get PCM and Fft
mLineIn->start();

ctx->start();
```

Andrea Cuius

andrea@nocte.co.uk
www.nocte.co.uk
@q_says / @noctestudio

nocte