

OpenGL++

Naomasa Matsubayashi

自己紹介

松林 尚理



@fadis_

第4回Boost.勉強会



2011年5月22日日曜日

まずことの発端について少し。第4回Boost.勉強会でOpenALの発表を行いました。この勉強会は基本的にC++の勉強会な訳ですが、★OpenALは残念ながらC言語のライブラリです。★OpenALはC言語ライブラリなので、と言った瞬間、★会場が凍り付くのを感じました。

第4回Boost.勉強会



OpenALは

C言語

2011年5月22日日曜日

まずことの発端について少し。第4回Boost.勉強会でOpenALの発表を行いました。この勉強会は基本的にC++の勉強会な訳ですが、★OpenALは残念ながらC言語のライブラリです。★OpenALはC言語ライブラリなので、と言った瞬間、★会場が凍り付くのを感じました。

第4回Boost.勉強会



OpenALは
C言語

C++
じゃないのか



2011年5月22日日曜日

まずことの発端について少し。第4回Boost.勉強会でOpenALの発表を行いました。この勉強会は基本的にC++の勉強会な訳ですが、★OpenALは残念ながらC言語のライブラリです。★OpenALはC言語ライブラリなので、と言った瞬間、★会場が凍り付くのを感じました。

OpenALにも C++ラッパはある

2011年5月22日 日曜日

で、OpenALのC++ラッパはないのかと言うと、あります。★が、トップページをよく見てみると☆、「新しいバージョンのOpenAL++がリリースされました」★★2004年?★これ、開発がかなり昔から止まってるんですね。従ってOpenALのC++ラッパはあるというよりは、★かつてあったといった状況です。

OpenALにも C++ラッパはある

[News](#)
[What is OpenAL++?](#)
[Documentation](#)
[FAQ](#)
[Downloads](#)
[CVS info](#)
[Bug reports](#)
[Contact info](#)
[Links](#)

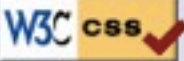
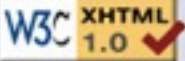
News

[New version](#) *vr-anders - 2004-11-11 00:48* - [OpenAL++](#)
A new version of OpenAL++ is available.
[\[Read More/Comment\]](#)

[New version](#) *vr-anders - 2004-03-02 00:12* - [OpenAL++](#)
OpenAL++ used to depend on CommonC++, this was a problem because CommonC++ is GPL and OpenAL++ is LGPL. So now the CommonC++ library is replaced partially by OpenThreads (LGPL). This also means that streaming support over the net is no longer in OpenAL++.
[\[Read More/Comment\]](#)

[New Version of OpenAL++](#) *vr-anders - 2003-12-11 22:53* - [OpenAL++](#)
A new version is release of OpenAL++.
Many bugfixes, support for streaming OGG files.
Builds under .NET2003 as well as Linux (of course).
[\[Read More/Comment\]](#)

[Home page up](#) *tomashamala - 2003-01-17 05:49* - [OpenAL++](#)
A new home page for OpenAL++ has been created and uploaded (on SourceForge).
[\[Read More/Comment\]](#)
[\[News archive\]](#)



2011年5月22日 日曜日

で、OpenALのC++ラッパはないのかと言うと、あります。★が、トップページをよく見てみると☆、「新しいバージョンのOpenAL++がリリースされました」★★2004年?★これ、開発がかなり昔から止まってるんですね。従ってOpenALのC++ラッパはあるというよりは、★かつてあったといった状況です。

OpenALにも C++ラッパはある



2011年5月22日 日曜日

で、OpenALのC++ラッパはないのかと言うと、あります。★が、トップページをよく見てみると☆、「新しいバージョンのOpenAL++がリリースされました」★★2004年?★これ、開発がかなり昔から止まってるんですね。従ってOpenALのC++ラッパはあるというよりは、★かつてあったといった状況です。

OpenALにも C++ラッパはある

News

[News](#)
[What is OpenAL++?](#)
[Documentation](#)
[FAQ](#)

New version *vr-anders* - **2004-11-11 00:48** - [OpenAL++](#)
A new version of OpenAL++ is available.

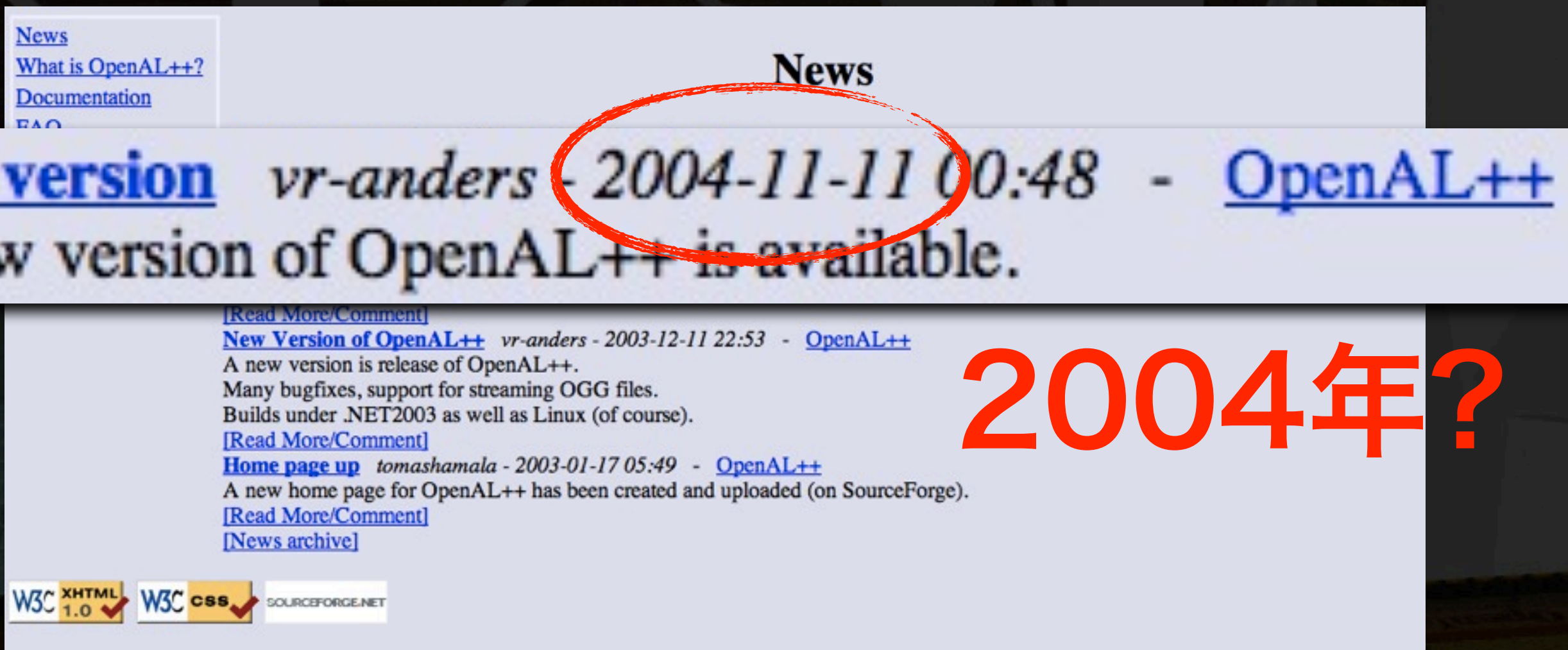
[\[Read More/Comment\]](#)
New Version of OpenAL++ *vr-anders* - 2003-12-11 22:53 - [OpenAL++](#)
A new version is release of OpenAL++.
Many bugfixes, support for streaming OGG files.
Builds under .NET2003 as well as Linux (of course).
[\[Read More/Comment\]](#)
Home page up *tomashamala* - 2003-01-17 05:49 - [OpenAL++](#)
A new home page for OpenAL++ has been created and uploaded (on SourceForge).
[\[Read More/Comment\]](#)
[\[News archive\]](#)

W3C XHTML 1.0 W3C CSS SOURCEFORGE.NET

2011年5月22日 日曜日

で、OpenALのC++ラッパはないのかと言うと、あります。★が、トップページをよく見てみると☆、「新しいバージョンのOpenAL++がリリースされました」★★2004年?★これ、開発がかなり昔から止まってるんですね。従ってOpenALのC++ラッパはあるというよりは、★かつてあったといった状況です。

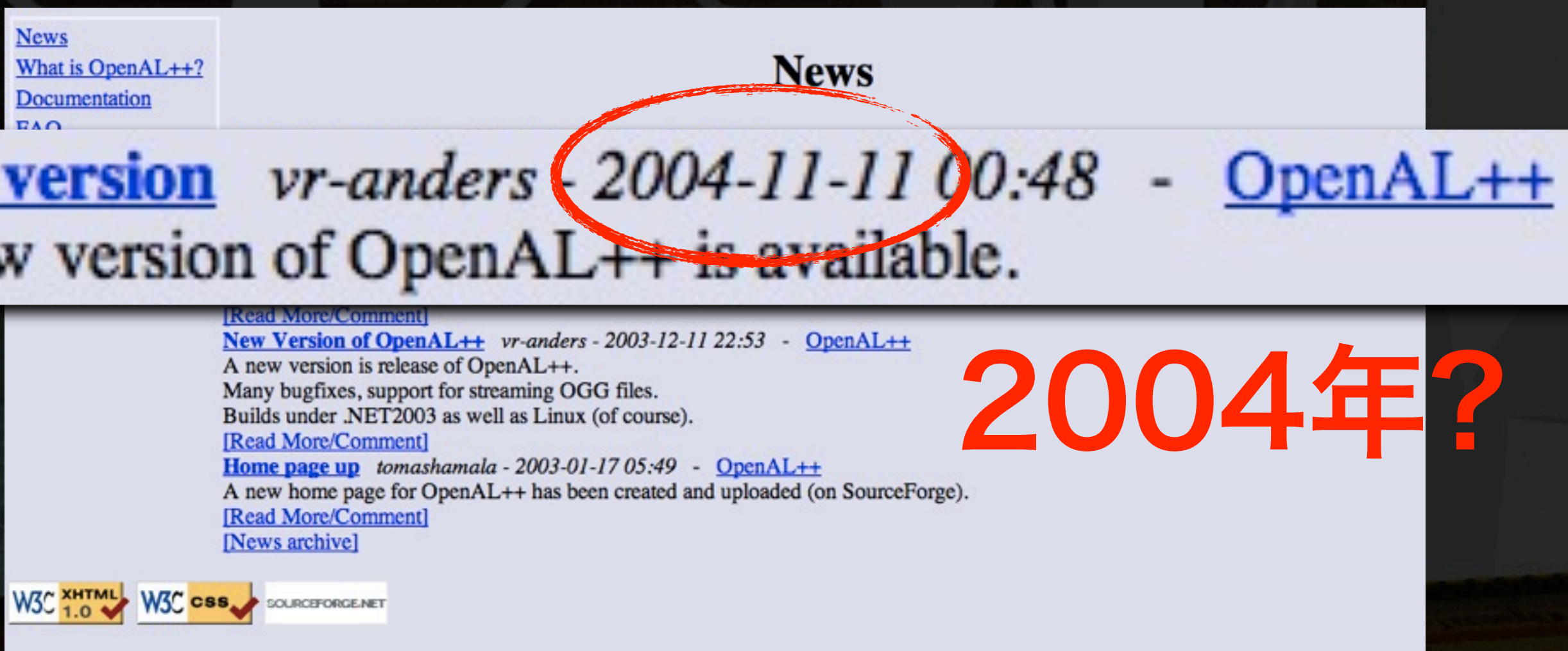
OpenALにも C++ラッパはある



2011年5月22日 日曜日

で、OpenALのC++ラッパはないのかと言うと、あります。★が、トップページをよく見てみると☆、「新しいバージョンのOpenAL++がリリースされました」★★2004年?★これ、開発がかなり昔から止まってるんですね。従ってOpenALのC++ラッパはあるというよりは、★かつてあったといった状況です。

OpenALにも C++ラッパはある



開発が止まっている

2011年5月22日 日曜日

で、OpenALのC++ラッパはないのかと言うと、あります。★が、トップページをよく見てみると☆、「新しいバージョンのOpenAL++がリリースされました」★★2004年?★これ、開発がかなり昔から止まってるんですね。従ってOpenALのC++ラッパはあるというよりは、★かつてあったといった状況です。

OpenALにも C++ラッパはあ



[News](#)
[What is OpenAL++?](#)
[Documentation](#)
[FAQ](#)

News

New version *vr-anders* - 2004-11-11 00:48 - [OpenAL++](#)
A new version of OpenAL++ is available.

[\[Read More/Comment\]](#)

New Version of OpenAL++ *vr-anders* - 2003-12-11 22:53 - [OpenAL++](#)

A new version is release of OpenAL++.

Many bugfixes, support for streaming OGG files.

Builds under .NET2003 as well as Linux (of course).

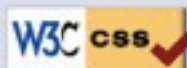
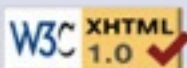
[\[Read More/Comment\]](#)

Home page up *tomashamala* - 2003-01-17 05:49 - [OpenAL++](#)

A new home page for OpenAL++ has been created and uploaded (on SourceForge).

[\[Read More/Comment\]](#)

[\[News archive\]](#)



SOURCEFORGE.NET

2004年?

開発が止まっている

2011年5月22日 日曜日

で、OpenALのC++ラッパはないのかと言うと、あります。★が、トップページをよく見てみると☆、「新しいバージョンのOpenAL++がリリースされました」★★2004年?★これ、開発がかなり昔から止まってるんですね。従ってOpenALのC++ラッパはあるというよりは、★かつてあったといった状況です。

UNIX文化

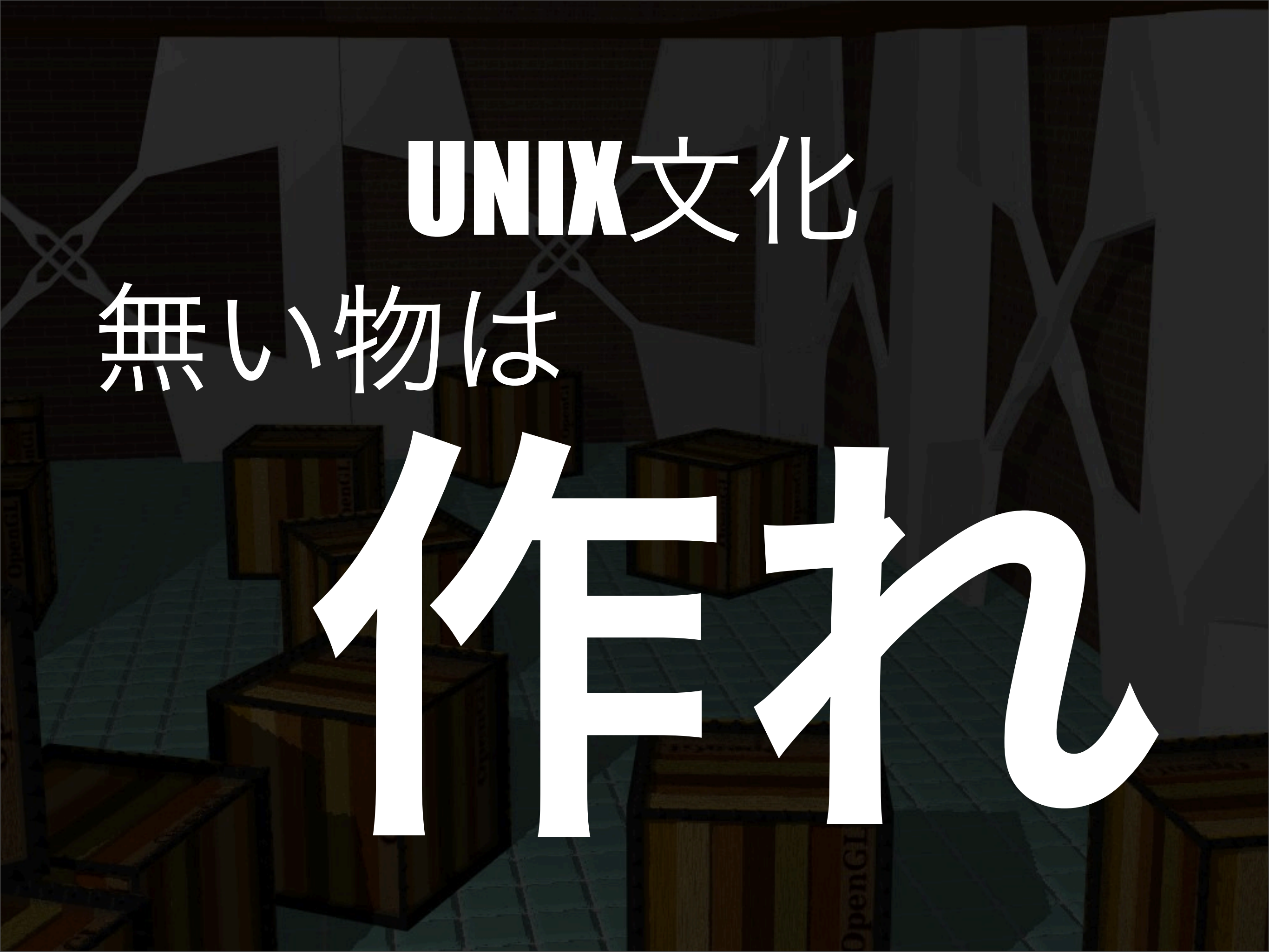
2011年5月22日日曜日

UNIX文化では☆ないものは、★作れます。

UNIX文化 無い物は

2011年5月22日日曜日

UNIX文化では☆ないものは、★作れます。



UNIX文化

無い物は

作れ

2011年5月22日日曜日

UNIX文化では☆ないものは、★作れます。

No meetings. No paper.
NO KIDDING!

DOWNLOAD
a Free Trial Now

CodeCollaborator

SOURCEforge

Find Open Source Software

Register Log In

OpenAL++ by lufeiz

Summary Files Reviews Support Develop Tracker Mailing Lists Forums Code

A modern C++ wrapper for OpenAL.

Project Home
alxx.sf.net

Recommend Me
[Show some love](#)

SF

Download
alxx-1.0.1.tar.gz

Develop
sf.net/projects/alxx/develop

Last Update
4 hours ago

Other Versions
[Browse all files](#)

Support
sf.net/projects/alxx/support

More Detail
[Hide](#)

Registered2011-04-19

Release Date2011-05-20

AVS Video Converter

DVD, VOB, DV, MOV, AVI, DivX, 3GP, XviD, VRO, MP4, WMV, ASF, MPEG, H.263, H.264, RM, RMVB, 3G2, SWF, DVR-MS, QT, ASX, MKV, OGM, FLV...

ダウンロード

www.avs4you.com

Related Resources

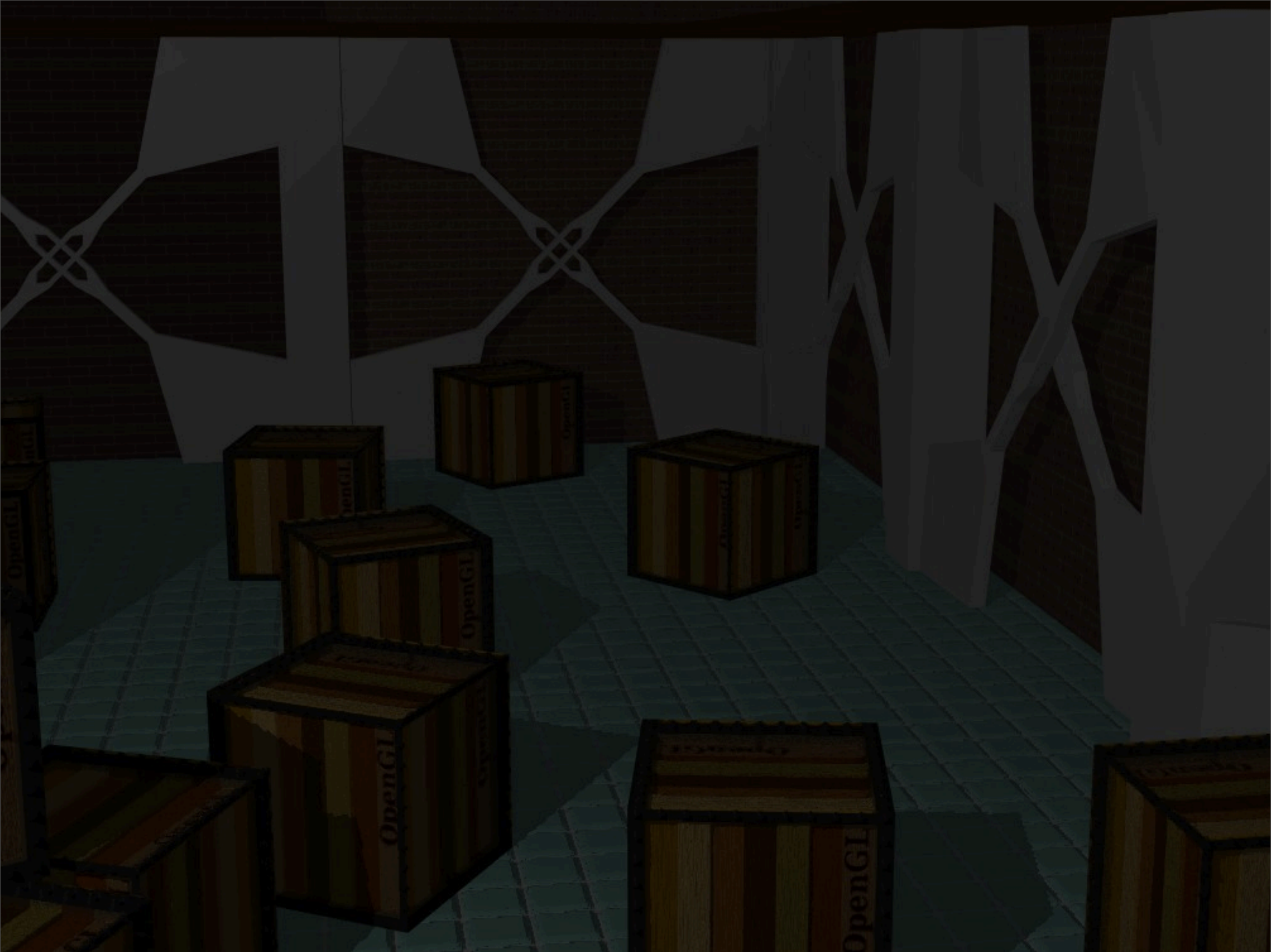
Optimal In-Memory Data Grid Architecture
Understand the drop-in implementation of Coherence running on Oracle's Sun servers utilizing WebLogic Suite. Impressive scalability and performance!

<http://sourceforge.net/projects/alxx/>

OpenAL++

2011年5月22日 日曜日

というわけで作りました。新しいOpenAL++です。



2011年5月22日日曜日

紛らわしいので見分け方ですが、★略称がalppなヤツが開発が止まってるOpenAL++で、★alxxなヤツが今回作ったOpenAL++です。



alpp

開発が止まっている

2011年5月22日 日曜日

紛らわしいので見分け方ですが、★略称がalppなヤツが開発が止まっているOpenAL++で、★alxxなヤツが今回作ったOpenAL++です。



alpp

開発が止まってる

alxx

今回作った

2011年5月22日日曜日

紛らわしいので見分け方ですが、★略称がalppなヤツが開発が止まってるOpenAL++で、★alxxなヤツが今回作ったOpenAL++です。



OpenAL

OpenAL++

2011年5月22日日曜日

C++バインダを使うことでどのくらい楽ができるかを見てみましょう。★これは、単純にバッファに読み込んだ音を再生するだけのOpenALで書かれたコードです。★一方これがOpenAL++で同じことをするコードです。OpenALだと★31行にもなるこのコードが、OpenAL++なら★15行です。


```
#include <boost/thread.hpp>
#include <AL/al.h>
#include <AL/alc.h>
#include "epiano.h"
int main(
) {
    const ALCchar *devices =
        alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context =
        alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    ALuint source;
    alGenSources( 1, &source );
    ALuint buffer;
    alGenBuffers( 1, &buffer );
    alBufferData(
        buffer, AL_FORMAT_MONO16, epiano,
        EPIANO_LENGTH * 2, 48000 );
    alSourceQueueBuffers( source, 1, &buffer );
    alSourcePlay( source );
    ALint stat;
    do {
        alGetSourcei( source, AL_SOURCE_STATE, &stat );
        boost::thread::yield();
    } while( stat == AL_PLAYING );
    alDeleteSources( 1, &source );
    alDeleteBuffers( 1, &buffer );
    alcMakeContextCurrent( NULL );
    alcDestroyContext( context );
    alcCloseDevice ( device );
}
```

OpenAL

OpenAL++

2011年5月22日日曜日

C++バインダを使うことでどのくらい楽ができるかを見てみましょう。★これは、単純にバッファに読み込んだ音を再生するだけのOpenALで書かれたコードです。★一方これがOpenAL++で同じことをするコードです。OpenALだと★31行にもなるこのコードが、OpenAL++なら★15行です。

```
#include <boost/thread.hpp>
#include <AL/al.h>
#include <AL/alc.h>
#include "epiano.h"
int main(
) {
    const ALCchar *devices =
        alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context =
        alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    ALuint source;
    alGenSources( 1, &source );
    ALuint buffer;
    alGenBuffers( 1, &buffer );
    alBufferData(
        buffer, AL_FORMAT_MONO16, epiano,
        EPIANO_LENGTH * 2, 48000 );
    alSourceQueueBuffers( source, 1, &buffer );
    alSourcePlay( source );
    ALint stat;
    do {
        alGetSourcei( source, AL_SOURCE_STATE, &stat );
        boost::thread::yield();
    } while( stat == AL_PLAYING );
    alDeleteSources( 1, &source );
    alDeleteBuffers( 1, &buffer );
    alcMakeContextCurrent( NULL );
    alcDestroyContext( context );
    alcCloseDevice ( device );
}
```

```
#include <al++/al++.hpp>
#include "epiano.h"
int main() {
    using namespace al::playback;
    DeviceList device_list;
    Device device( device_list.front() );
    Context context( device );
    Source source( context );
    Buffer buffer( context );
    buffer.setData( epiano,
        epiano + EPIANO_LENGTH, 48000 );
    source.queueBuffer( buffer );
    source.play();
    source.wait();
}
```

OpenAL

OpenAL++

2011年5月22日日曜日

C++バインダを使うことでどのくらい楽ができるかを見てみましょう。★これは、単純にバッファに読み込んだ音を再生するだけのOpenALで書かれたコードです。★一方これがOpenAL++で同じことをするコードです。OpenALだと★31行にもなるこのコードが、OpenAL++なら★15行です。


```

#include <boost/thread.hpp>
#include <AL/al.h>
#include <AL/alc.h>
#include "epiano.h"
int main(
) {
    const ALCchar *devices =
        alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context =
        alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    ALuint source;
    alGenSources( 1, &source );
    ALuint buffer;
    alGenBuffers( 1, &buffer );
    alBufferData(
        buffer, AL_FORMAT_MONO16, epiano,
        EPIANO_LENGTH * 2, 48000 );
    alSourceQueueBuffers( source, 1, &buffer );
    alSourcePlay( source );
    ALint stat;
    do {
        alGetSourcei( source, AL_SOURCE_STATE, &stat );
        boost::thread::yield();
    } while( stat == AL_PLAYING );
    alDeleteSources( 1, &source );
    alDeleteBuffers( 1, &buffer );
    alcMakeContextCurrent( NULL );
    alcDestroyContext( context );
    alcCloseDevice ( device );
}

```

```

#include <al++/al++.hpp>
#include "epiano.h"
int main() {
    using namespace al::playback;
    DeviceList device_list;
    Device device( device_list.front() );
    Context context( device );
    Source source( context );
    Buffer buffer( context );
    buffer.setData( epiano,
        epiano + EPIANO_LENGTH, 48000 );
    source.queueBuffer( buffer );
    source.play();
    source.wait();
}

```

OpenAL

OpenAL++

2011年5月22日日曜日

C++バインダを使うことでどのくらい楽ができるかを見てみましょう。★これは、単純にバッファに読み込んだ音を再生するだけのOpenALで書かれたコードです。★一方これがOpenAL++で同じことをするコードです。OpenALだと★31行にもなるこのコードが、OpenAL++なら★15行です。

```

#include <boost/thread.hpp>
#include <AL/al.h>
#include <AL/alc.h>
#include "epiano.h"
int main(
) {
    const ALCchar *devices =
        alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context =
        alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    ALuint source;
    alGenSources( 1, &source );
    ALuint buffer;
    alGenBuffers( 1, &buffer );
    alBufferData(
        buffer, AL_FORMAT_MONO16, epiano,
        EPIANO_LENGTH * 2, 48000 );
    alSourceQueueBuffers( source, 1, &buffer );
    alSourcePlay( source );
    ALint stat;
    do {
        alGetSourcei( source, AL_SOURCE_STATE, &stat );
        boost::thread::yield();
    } while( stat == AL_PLAYING );
    alDeleteSources( 1, &source );
    alDeleteBuffers( 1, &buffer );
    alcMakeContextCurrent( NULL );
    alcDestroyContext( context );
    alcCloseDevice ( device );
}

```

31行

OpenAL

```

#include <al++/al++.hpp>
#include "epiano.h"
int main() {
    using namespace al::playback;
    DeviceList device_list;
    Device device( device_list.front() );
    Context context( device );
    Source source( context );
    Buffer buffer( context );
    buffer.setData( epiano,
        epiano + EPIANO_LENGTH, 48000 );
    source.queueBuffer( buffer );
    source.play();
    source.wait();
}

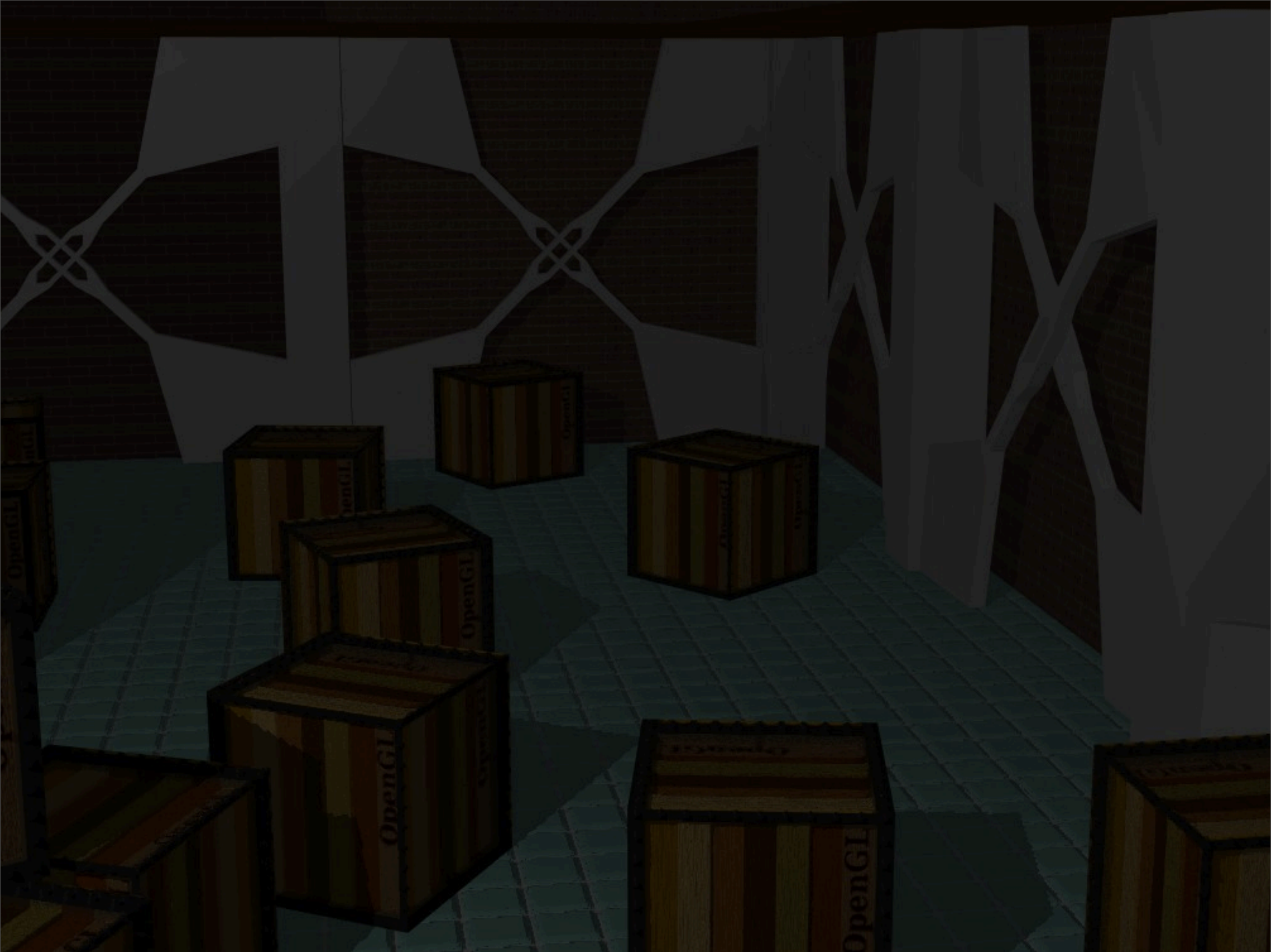
```

15行

OpenAL++

2011年5月22日日曜日

C++バインダを使うことでどのくらい楽ができるかを見てみましょう。★これは、単純にバッファに読み込んだ音を再生するだけのOpenALで書かれたコードです。★一方これがOpenAL++で同じことをするコードです。OpenALだと★31行にもなるこのコードが、OpenAL++なら★15行です。



2011年5月22日日曜日

OpenALの使用手順はBoost.勉強会#4での発表の方を参照していただきたいのですが、あのめんどくさいOpenALの手順がどう簡単になっているか見てみましょう。★この2行でデバイス一覧から最初のデバイスを取得し、★続いて必要なリソースを用意し、★バッファを登録して再生、待機といったようにOpenALの通常の手順を☆より簡潔に書けるようになっています。

```
DeviceList device_list;  
Device device( device_list.front() );
```

デバイス一覧から、1つ目のデバイスを選択する



2011年5月22日日曜日

OpenALの使用手順はBoost.勉強会#4での発表の方を参照していただきたいのですが、あのめんどくさいOpenALの手順がどう簡単になっているか見てみましょう。★この2行でデバイス一覧から最初のデバイスを取得し、★続いて必要なリソースを用意し、★バッファを登録して再生、待機といったようにOpenALの通常の手順を☆より簡潔に書けるようになっています。


```
DeviceList device_list;  
Device device( device_list.front() );
```

デバイス一覧から、1つ目のデバイスを選択する

```
Context context( device );  
Source source( context );  
Buffer buffer( context );
```

コンテキスト、ソース、バッファを作る

2011年5月22日 日曜日

OpenALの使用手順はBoost.勉強会#4での発表の方を参照していただきたいのですが、あのめんどくさいOpenALの手順がどう簡単になっているか見てみましょう。★この2行でデバイス一覧から最初のデバイスを取得し、★続いて必要なリソースを用意し、★バッファを登録して再生、待機といったようにOpenALの通常の手順を☆より簡潔に書けるようになっています。

```
DeviceList device_list;  
Device device( device_list.front() );
```

デバイス一覧から、1つ目のデバイスを選択する

```
Context context( device );  
Source source( context );  
Buffer buffer( context );
```

コンテキスト、ソース、バッファを作る

```
buffer.setData( epiano,  
    epiano + EPIANO_LENGTH, 48000 );  
source.queueBuffer( buffer );  
source.play();  
source.wait();
```

バッファに値を書き込んで、ソースに登録して再生、
その後再生が終わるまで待つ

2011年5月22日 日曜日

OpenALの使用手順はBoost.勉強会#4での発表の方を参照していただきたいのですが、あのめんどくさいOpenALの手順がどう簡単になっているか見てみましょう。★この2行でデバイス一覧から最初のデバイスを取得し、★続いて必要なリソースを用意し、★バッファに登録して再生、待機といったようにOpenALの通常の手順を☆より簡潔に書けるようになっています。


```
DeviceList device_list;  
Device device( device_list.front() );
```

デバイス一覧から、1つ目のデバイスを選択する

```
Context context( device );  
Source source( context );  
Buffer buffer( context );
```

コンテキスト、ソース、バッファを作る

```
buffer.setData( epiano,  
    epiano + EPIANO_LENGTH, 48000 );  
source.queueBuffer( buffer );  
source.play();  
source.wait();
```

バッファに値を書き込んで、ソースに登録して再生、
その後再生が終わるまで待つ

2011年5月22日日曜日

OpenALの使用手順はBoost.勉強会#4での発表の方を参照していただきたいのですが、あのめんどくさいOpenALの手順がどう簡単になっているか見てみましょう。★この2行でデバイス一覧から最初のデバイスを取得し、★続いて必要なリソースを用意し、★バッファに登録して再生、待機といったようにOpenALの通常の手順を☆より簡潔に書けるようになっています。



OpenAL

OpenAL++

2011年5月22日日曜日

★これは、システムがサポートするOpenALの拡張機能の一覧をOpenALで取得するコードです。★
一方これは同じことをするOpenAL++のコードです。 OpenALだと★35行にもなるこのコードが、
OpenAL++なら★14行です。


```
#include <iostream>
#include <boost/foreach.hpp>
#include <boost/spirit/include/qi.hpp>
#include <AL/al.h>
#include <AL/alc.h>
int main( int _argc, char **_argv ) {
    const ALCchar *devices =
        alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context = alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    const ALCchar *alc_begin =
        alcGetString( device, ALC_EXTENSIONS );
    const ALCchar *alc_end;
    for( alc_end = alc_begin; *alc_end; alc_end++ );
    std::vector< std::string > alc_extensions;
    using namespace boost::spirit;
    qi::parse( alc_begin, alc_end,
        +ascii::graph % ' ', alc_extensions );
    std::cout << "== ALC Extensions ==" << std::endl;
    BOOST_FOREACH( const std::string &elem, alc_extensions )
        std::cout << elem << std::endl;
    const ALchar *al_begin = alGetString( AL_EXTENSIONS );
    const ALchar *al_end;
    for( al_end = al_begin; *al_end; al_end++ );
    std::vector< std::string > al_extensions;
    qi::parse( al_begin, al_end,
        +ascii::graph % ' ', al_extensions );
    std::cout << "== AL Extensions ==" << std::endl;
    BOOST_FOREACH( const std::string &elem, al_extensions )
        std::cout << elem << std::endl;
    alcMakeContextCurrent( NULL );
    alcDestroyContext( context );
    alcCloseDevice ( device );
}
```

OpenAL

OpenAL++

2011年5月22日日曜日

★これは、システムがサポートするOpenALの拡張機能の一覧をOpenALで取得するコードです。★
一方これは同じことをするOpenAL++のコードです。 OpenALだと★35行にもなるこのコードが、
OpenAL++なら★14行です。

```

#include <iostream>
#include <boost/foreach.hpp>
#include <boost/spirit/include/qi.hpp>
#include <AL/al.h>
#include <AL/alc.h>
int main( int _argc, char **_argv ) {
    const ALCchar *devices =
        alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context = alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    const ALCchar *alc_begin =
        alcGetString( device, ALC_EXTENSIONS );
    const ALCchar *alc_end;
    for( alc_end = alc_begin; *alc_end; alc_end++ );
    std::vector< std::string > alc_extensions;
    using namespace boost::spirit;
    qi::parse( alc_begin, alc_end,
        +ascii::graph % ' ', alc_extensions );
    std::cout << "== ALC Extensions ==" << std::endl;
    BOOST_FOREACH( const std::string &elem, alc_extensions )
        std::cout << elem << std::endl;
    const ALchar *al_begin = alGetString( AL_EXTENSIONS );
    const ALchar *al_end;
    for( al_end = al_begin; *al_end; al_end++ );
    std::vector< std::string > al_extensions;
    qi::parse( al_begin, al_end,
        +ascii::graph % ' ', al_extensions );
    std::cout << "== AL Extensions ==" << std::endl;
    BOOST_FOREACH( const std::string &elem, al_extensions )
        std::cout << elem << std::endl;
    alcMakeContextCurrent( NULL );
    alcDestroyContext( context );
    alcCloseDevice ( device );
}

```

```

#include <iostream>
#include <al++/al++.hpp>
int main() {
    using namespace al::playback;
    DeviceList device_list;
    Device device( device_list.front() );
    Context context( device );
    ALCExtensionList alcel( device );
    std::cout << "== ALC Extensions ==" << std::endl;
    std::cout << alcel;
    ALExtensionList alel( context );
    std::cout << "== AL Extensions ==" << std::endl;
    std::cout << alel;
}

```

OpenAL

OpenAL++

2011年5月22日日曜日

★これは、システムがサポートするOpenALの拡張機能の一覧をOpenALで取得するコードです。★
 一方これは同じことをするOpenAL++のコードです。 OpenALだと★35行にもなるこのコードが、
 OpenAL++なら★14行です。


```

#include <iostream>
#include <boost/foreach.hpp>
#include <boost/spirit/include/qi.hpp>
#include <AL/al.h>
#include <AL/alc.h>
int main( int _argc, char **_argv ) {
    const ALCchar *devices =
        alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context = alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    const ALCchar *alc_begin =
        alcGetString( device, ALC_EXTENSIONS );
    const ALCchar *alc_end;
    for( alc_begin = alc_end; alc_end = alc_end + 1; );
    std::vector< std::string > alc_extensions;
    using namespace boost::spirit;
    qi::parse( alc_begin, alc_end,
        +ascii::grapheme, alc_extensions );
    std::cout << "== ALC Extensions ==" << std::endl;
    BOOST_FOREACH( const std::string &elem, alc_extensions )
        std::cout << elem << std::endl;
    const ALchar *al_begin = alGetString( AL_EXTENSIONS );
    const ALchar *al_end;
    for( al_end = al_begin; *al_end; al_end++ );
    std::vector< std::string > al_extensions;
    qi::parse( al_begin, al_end,
        +ascii::grapheme, al_extensions );
    std::cout << "== AL Extensions ==" << std::endl;
    BOOST_FOREACH( const std::string &elem, al_extensions )
        std::cout << elem << std::endl;
    alcMakeContextCurrent( NULL );
    alcDestroyContext( context );
    alcCloseDevice ( device );
}

```

```

#include <iostream>
#include <al++/al++.hpp>
int main() {
    using namespace al::playback;
    DeviceList device_list;
    Device device( device_list.front() );
    Context context( device );
    ALCExtensionList alcel( device );
    std::cout << "== ALC Extensions ==" << std::endl;
    std::cout << alcel;
    ALExtensionList alel( context );
    std::cout << "== AL Extensions ==" << std::endl;
    std::cout << alel;
}

```

OpenAL

OpenAL++

2011年5月22日日曜日

★これは、システムがサポートするOpenALの拡張機能の一覧をOpenALで取得するコードです。★
 一方これは同じことをするOpenAL++のコードです。 OpenALだと★35行にもなるこのコードが、
 OpenAL++なら★14行です。

```

#include <iostream>
#include <boost/foreach.hpp>
#include <boost/spirit/include/qi.hpp>
#include <AL/al.h>
#include <AL/alc.h>
int main( int _argc, char **_argv ) {
    const ALCchar *devices =
        alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context = alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    const ALCchar *alc_begin =
        alcGetString( device, ALC_EXTENSIONS );
    const ALCchar *alc_end;
    for( alc_begin = alc_end; alc_end = alc_end + 1; );
    std::vector< std::string > alc_extensions;
    using namespace boost::spirit;
    qi::parse( alc_begin, alc_end,
        +ascii::grapheme, alc_extensions );
    std::cout << "== ALC Extensions ==" << std::endl;
    BOOST_FOREACH( const std::string &elem, alc_extensions )
        std::cout << elem << std::endl;
    const ALchar *al_begin = alGetString( AL_EXTENSIONS );
    const ALchar *al_end;
    for( al_end = al_begin; *al_end; al_end++ );
    std::vector< std::string > al_extensions;
    qi::parse( al_begin, al_end,
        +ascii::grapheme, al_extensions );
    std::cout << "== AL Extensions ==" << std::endl;
    BOOST_FOREACH( const std::string &elem, al_extensions )
        std::cout << elem << std::endl;
    alcMakeContextCurrent( NULL );
    alcDestroyContext( context );
    alcCloseDevice ( device );
}

```

35行

```

#include <iostream>
#include <al++/al++.hpp>
int main() {
    using namespace al::platform;
    DeviceList device_list;
    Device device( device_list.front() );
    Context context( device );
    ALCExtensionList alc_ext_list( device );
    std::cout << "== ALC Extensions ==" << std::endl;
    std::cout << alc_ext_list;
    ALExtensionList al_ext_list( context );
    std::cout << "== AL Extensions ==" << std::endl;
    std::cout << al_ext_list;
}

```

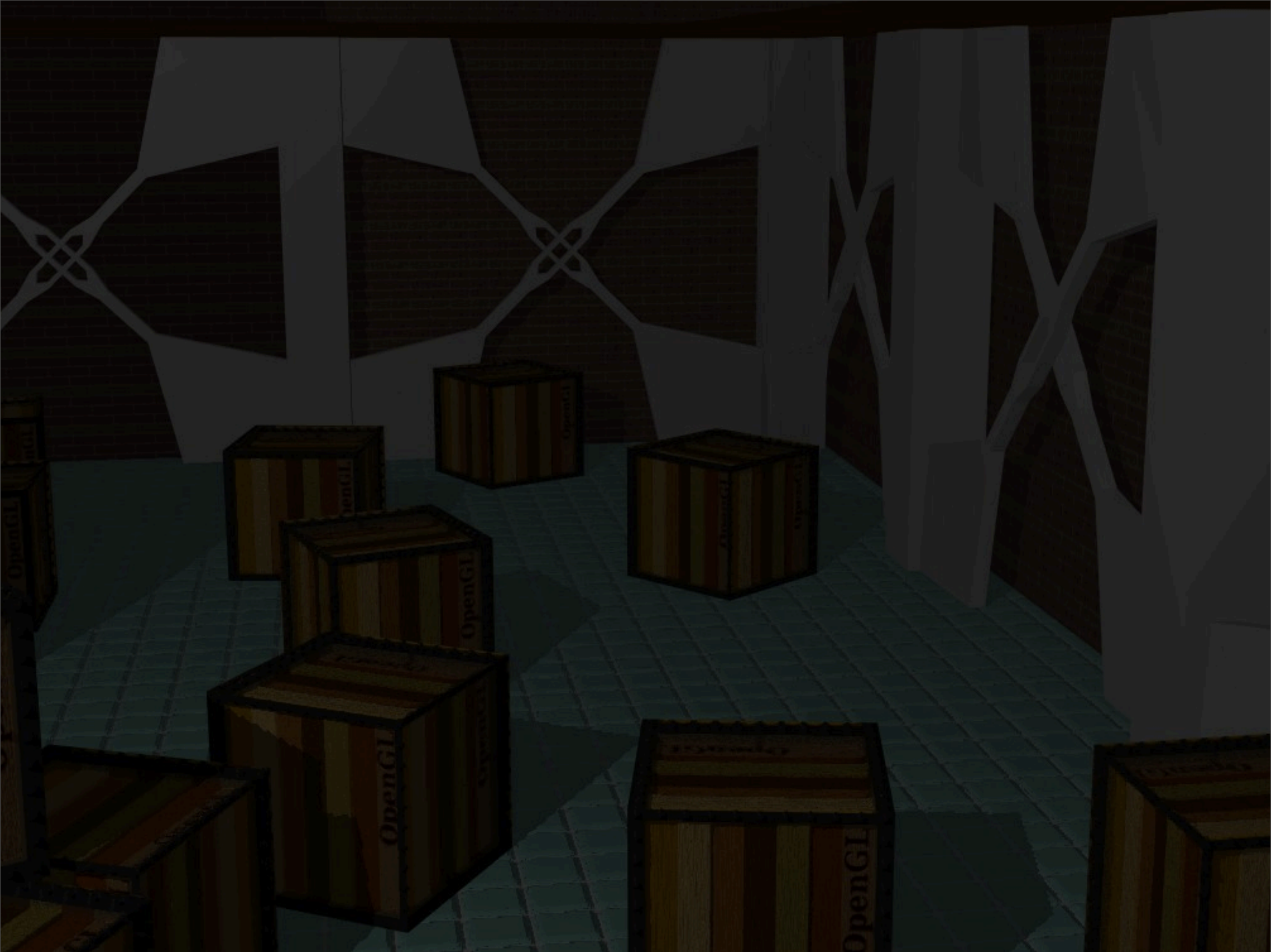
14行

OpenAL

OpenAL++

2011年5月22日日曜日

★これは、システムがサポートするOpenALの拡張機能の一覧をOpenALで取得するコードです。★
 一方これは同じことをするOpenAL++のコードです。 OpenALだと★35行にもなるこのコードが、
 OpenAL++なら★14行です。



2011年5月22日日曜日

★まずは先ほどと同じようにデバイスを開きます。★★OpenAL++では拡張一覧はSTLコンテナライクなアクセスが可能なクラスで調べることが出来るようになっていました。このクラスはそのまま ostream にダンプできるようになっているので★出力はこの1行です。☆簡単ですね！

```
DeviceList device_list;  
Device device( device_list.front() );
```

デバイス一覧から、1つ目のデバイスを選択する



2011年5月22日日曜日

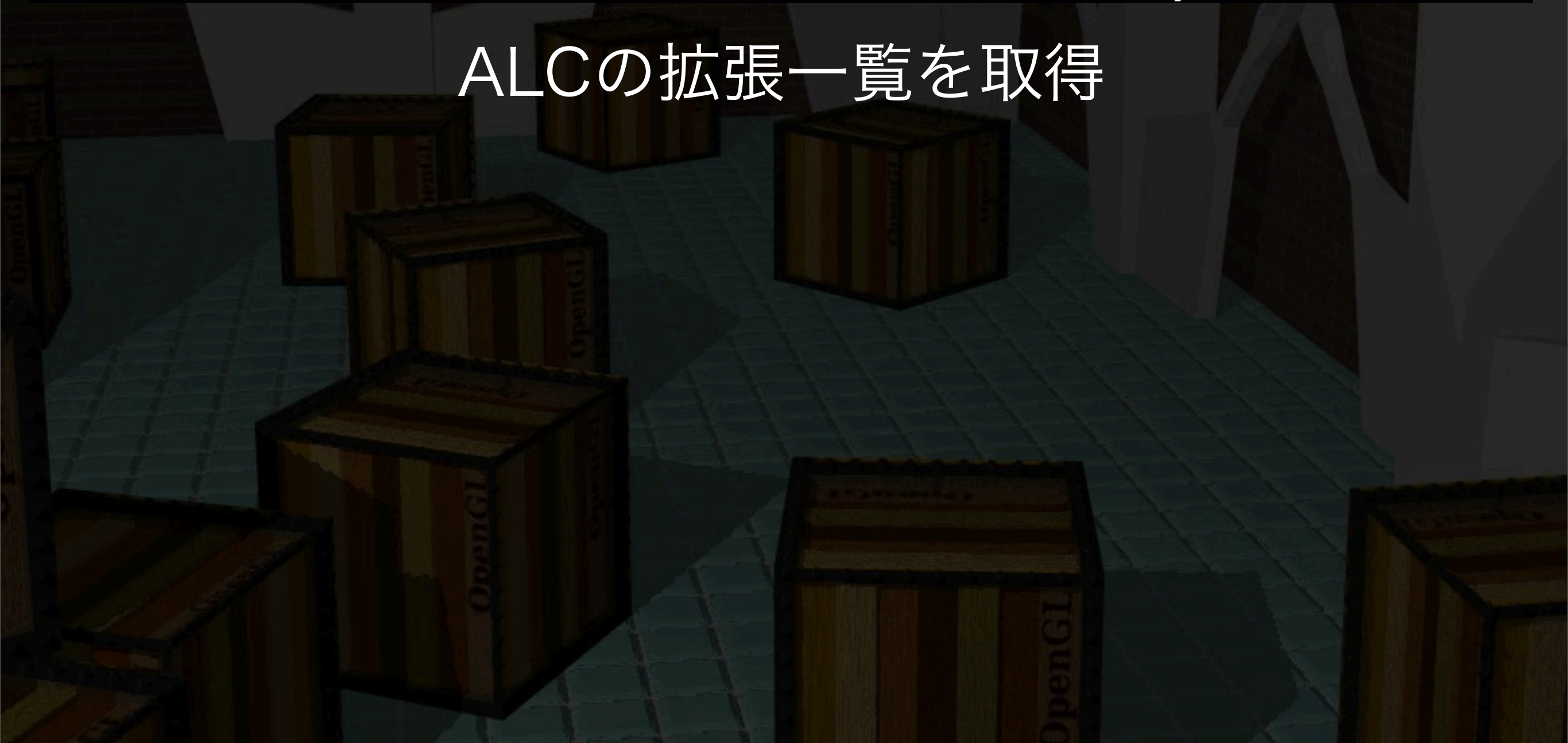
★まずは先ほどと同じようにデバイスを開きます。★★OpenAL++では拡張一覧はSTLコンテナライクなアクセスが可能なクラスで調べることが出来るようになっていました。このクラスはそのまま ostream にダンプできるようになっているので★出力はこの1行です。☆簡単ですね！


```
DeviceList device_list;  
Device device( device_list.front() );
```

デバイス一覧から、1つ目のデバイスを選択する

```
Context context( device );  
ALCExtensionList alcel( device );
```

ALCの拡張一覧を取得



2011年5月22日日曜日

★まずは先ほどと同じようにデバイスを開きます。★★OpenAL++では拡張一覧はSTLコンテナライクなアクセスが可能なクラスで調べることが出来るようになっていました。このクラスはそのまま ostream にダンプできるようになっているので★出力はこの1行です。☆簡単ですね！

```
DeviceList device_list;  
Device device( device_list.front() );
```

デバイス一覧から、1つ目のデバイスを選択する

```
Context context( device );  
ALCExtensionList alcel( device );
```

ALCの拡張一覧を取得

```
ALExtensionList alel( context );
```

ALの拡張一覧を取得

2011年5月22日日曜日

★まずは先ほどと同じようにデバイスを開きます。★★OpenAL++では拡張一覧はSTLコンテナライクなアクセスが可能なクラスで調べることが出来るようになっていました。このクラスはそのまま ostream にダンプできるようになっているので★出力はこの1行です。☆簡単ですね！


```
DeviceList device_list;  
Device device( device_list.front() );
```

デバイス一覧から、1つ目のデバイスを選択する

```
Context context( device );  
ALCExtensionList alcel( device );
```

ALCの拡張一覧を取得

```
ALExtensionList alel( context );
```

ALの拡張一覧を取得

```
std::cout << alel;
```

std::ostream対応

2011年5月22日日曜日

★まずは先ほどと同じようにデバイスを開きます。★★OpenAL++では拡張一覧はSTLコンテナライクなアクセスが可能なクラスで調べることが出来るようになっていました。このクラスはそのままostreamにダンプできるようになっているので★出力はこの1行です。☆簡単ですね！

```
DeviceList device_list;  
Device device( device_list.front() );
```

デバイス一覧から、1つ目のデバイスを選択する

```
Context context( device );  
ALCExtensionList alcel( device );
```

ALCの拡張一覧を取得

```
ALExtensionList alel( context );
```

ALの拡張一覧を取得

```
std::cout << alel;
```

std::ostream対応

**EASY
TO
USE**

2011年5月22日日曜日

★まずは先ほどと同じようにデバイスを開きます。★★OpenAL++では拡張一覧はSTLコンテナライクなアクセスが可能なクラスで調べることが出来るようになっていました。このクラスはそのままostreamにダンプできるようになっているので★出力はこの1行です。☆簡単ですね！



OpenAL

OpenAL++

2011年5月22日日曜日

★これはストリーミング再生を行うOpenALで書かれたコードです。★そしてこちらが同じことをするOpenAL++で書かれたコードです。 OpenALだと★49行にもなるこのコードが、OpenAL++なら★27行です。

```
#include <iostream>
#include <boost/thread.hpp>
#include <AL/al.h>
#include <AL/alc.h>
int main() {
    const ALCchar *devices = alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context = alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    ALuint source;
    alGenSources( 1, &source );
    ALuint buffers[ 4 ] = { 0, 0, 0, 0 };
    alGenBuffers( 4, buffers );
    unsigned int count;
    double time = 0.0;
    ALshort raw_data[ 2048 ];
    unsigned int index;
    for( count = 1; count != 4; count++ ) {
        for( index = 0; index != 2048; index++, time += 1.0 ) {
            raw_data[ index ] = 0;
        }
        alBufferData( buffers[ count % 4 ], AL_FORMAT_MONO16, raw_data,
            2048 * sizeof( ALshort ), 48000 );
        alSourceQueueBuffers( source, 1, buffers + count % 4 );
    }
    alSourcePlay( source );
    for( count = 0; count++ ) {
        ALint state;
        alGetSourcei( source, AL_SOURCE_STATE, &state );
        if( state != AL_PLAYING ) {
            alSourcePlay( source );
        }
        ALint unqueued = 0;
        while( true ) {
            alGetSourcei( source, AL_BUFFERS_PROCESSED, &unqueued );
            if( unqueued )
                break;
            boost::thread::yield();
        }
        ALuint next;
        alSourceUnqueueBuffers( source, 1, &next );
        for( index = 0; index != 2048; index++, time += 0.1 ) {
            raw_data[ index ] = sin( time ) * 32767.0;
        }
        alBufferData( next, AL_FORMAT_MONO16, raw_data,
            2048 * sizeof( ALshort ), 48000 );
        alSourceQueueBuffers( source, 1, &next );
    }
}
```

OpenAL

OpenAL++

2011年5月22日日曜日

★これはストリーミング再生を行うOpenALで書かれたコードです。★そしてこちらが同じことをするOpenAL++で書かれたコードです。 OpenALだと★49行にもなるこのコードが、OpenAL++なら★27行です。


```

#include <iostream>
#include <boost/thread.hpp>
#include <AL/al.h>
#include <AL/alc.h>
int main() {
    const ALCchar *devices = alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context = alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    ALuint source;
    alGenSources( 1, &source );
    ALuint buffers[ 4 ] = { 0, 0, 0, 0 };
    alGenBuffers( 4, buffers );
    unsigned int count;
    double time = 0.0;
    ALshort raw_data[ 2048 ];
    unsigned int index;
    for( count = 1; count != 4; count++ ) {
        for( index = 0; index != 2048; index++, time += 1.0 ) {
            raw_data[ index ] = 0;
        }
        alBufferData( buffers[ count % 4 ], AL_FORMAT_MONO16, raw_data,
            2048 * sizeof( ALshort ), 48000 );
        alSourceQueueBuffers( source, 1, buffers + count % 4 );
    }
    alSourcePlay( source );
    for( count = 0; count++ ) {
        ALint state;
        alGetSourcei( source, AL_SOURCE_STATE, &state );
        if( state != AL_PLAYING ) {
            alSourcePlay( source );
        }
        ALint unqueued = 0;
        while( true ) {
            alGetSourcei( source, AL_BUFFERS_PROCESSED, &unqueued );
            if( unqueued )
                break;
            boost::thread::yield();
        }
        ALuint next;
        alSourceUnqueueBuffers( source, 1, &next );
        for( index = 0; index != 2048; index++, time += 0.1 ) {
            raw_data[ index ] = sin( time ) * 32767.0;
        }
        alBufferData( next, AL_FORMAT_MONO16, raw_data,
            2048 * sizeof( ALshort ), 48000 );
        alSourceQueueBuffers( source, 1, &next );
    }
}

```

```

#include <al++/al++.hpp>
#include <cmath>
struct Feeder {
    void operator()( al::playback::Buffer &_buffer ) {
        std::vector< ALshort > temp;
        temp.resize( 109 * 10 );
        std::vector< ALshort >::iterator iter;
        float pos;
        for( iter = temp.begin(), pos = 0.0f;
            iter != temp.end();
            iter++, pos += 2.0f * M_PI / 109.0f ) {
            *iter = sinf( pos ) * 32767.0f;
        }
        _buffer.setData( temp.begin(), temp.end(), 48000 );
    }
};
int main() {
    using namespace al::playback;
    DeviceList device_list;
    Device device( device_list.front() );
    ContextAttribute attr;
    Context context( device, attr );
    boost::shared_ptr< al::TaskRemapper > trm( new al::TaskRemapper );
    Stream stream1( context, trm, Feeder(), 10 );
    stream1.play();
    stream1.wait();
}

```

OpenAL

OpenAL++

2011年5月22日日曜日

★これはストリーミング再生を行うOpenALで書かれたコードです。★そしてこちらが同じことをするOpenAL++で書かれたコードです。 OpenALだと★49行にもなるこのコードが、OpenAL++なら★27行です。

```
#include <iostream>
#include <boost/thread.hpp>
#include <AL/al.h>
#include <AL/alc.h>
int main() {
    const ALCchar *devices = alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context = alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    ALuint source;
    alGenSources( 1, &source );
    ALuint buffers[ 4 ] = { 0, 0, 0, 0 };
    alGenBuffers( 4, buffers );
    unsigned int count;
    double time = 0.0;
    ALshort raw_data[ 2048 ];
    unsigned int index;
    for( count = 1; count != 4; count++ ) {
        for( index = 0; index != 2048; index++, time += 0.1 ) {
            raw_data[ index ] = 0;
        }
        alBufferData( buffers[ count % 4 ], AL_FORMAT_MONO16, raw_data,
            2048 * sizeof( ALshort ), 48000 );
        alSourceQueueBuffers( source, 1, buffers + count % 4 );
    }
    alSourcePlay( source );
    for( count = 0; count != 4; count++ ) {
        ALint state;
        alGetSourcei( source, AL_SOURCE_STATE, &state );
        if( state != AL_PLAYING ) {
            alSourcePlay( source );
        }
        ALint unqueued = 0;
        while( true ) {
            alGetSourcei( source, AL_BUFFERS_PROCESSED, &unqueued );
            if( unqueued )
                break;
            boost::thread::yield();
        }
        ALuint next;
        alSourceUnqueueBuffers( source, 1, &next );
        for( index = 0; index != 2048; index++, time += 0.1 ) {
            raw_data[ index ] = sin( time ) * 32767.0;
        }
        alBufferData( next, AL_FORMAT_MONO16, raw_data,
            2048 * sizeof( ALshort ), 48000 );
        alSourceQueueBuffers( source, 1, &next );
    }
}
```

```
#include <al++/al++.hpp>
#include <cmath>
struct Feeder {
    void operator()( al::playback::Buffer &_buffer ) {
        std::vector< ALshort > temp;
        temp.resize( 109 * 10 );
        std::vector< ALshort >::iterator iter;
        float pos;
        for( iter = temp.begin(), pos = 0.0f;
            iter != temp.end();
            iter++, pos += 2.0f * M_PI / 109.0f ) {
            *iter = sinf( pos ) * 32767.0f;
        }
        _buffer.setData( temp.begin(), temp.end(), 48000 );
    }
};
int main() {
    using namespace al::playback;
    DeviceList device_list;
    Device device( device_list.front() );
    ContextAttribute attr;
    Context context( device, attr );
    boost::shared_ptr< al::TaskRemapper > trm( new al::TaskRemapper );
    Stream stream1( context, trm, Feeder(), 10 );
    stream1.play();
    stream1.wait();
}
```

OpenAL

OpenAL++

2011年5月22日日曜日

★これはストリーミング再生を行うOpenALで書かれたコードです。★そしてこちらが同じことをするOpenAL++で書かれたコードです。 OpenALだと★49行にもなるこのコードが、OpenAL++なら★27行です。


```
#include <iostream>
#include <boost/thread.hpp>
#include <AL/al.h>
#include <AL/alc.h>
int main() {
    const ALCchar *devices = alcGetString( NULL, ALC_DEVICE_SPECIFIER );
    ALCdevice *device = alcOpenDevice ( devices );
    ALCcontext *context = alcCreateContext( device, NULL );
    alcMakeContextCurrent( context );
    ALuint source;
    alGenSources( 1, &source );
    ALuint buffers[ 4 ] = { 0, 0, 0, 0 };
    alGenBuffers( 4, buffers );
    unsigned int count;
    double time = 0.0;
    ALshort raw_data[ 2048 ];
    unsigned int index;
    for( count = 1; count != 4; count++ ) {
        for( index = 0; index != 2048; index++, time += 0.1 ) {
            raw_data[ index ] = 0;
        }
        alBufferData( buffers[ count % 4 ], AL_FORMAT_MONO16, raw_data,
            2048 * sizeof( ALshort ), 48000 );
        alSourceQueueBuffers( source, 1, buffers + count % 4 );
    }
    alSourcePlay( source );
    for( count = 0; count != 4; count++ ) {
        ALint state;
        alGetSourcei( source, AL_SOURCE_STATE, &state );
        if( state != AL_PLAYING ) {
            alSourcePlay( source );
        }
        ALint unqueued = 0;
        while( true ) {
            alGetSourcei( source, AL_BUFFERS_PROCESSED, &unqueued );
            if( unqueued )
                break;
            boost::thread::yield();
        }
        ALuint next;
        alSourceUnqueueBuffers( source, 1, &next );
        for( index = 0; index != 2048; index++, time += 0.1 ) {
            raw_data[ index ] = sin( time ) * 32767.0;
        }
        alBufferData( next, AL_FORMAT_MONO16, raw_data,
            2048 * sizeof( ALshort ), 48000 );
        alSourceQueueBuffers( source, 1, &next );
    }
}
```

49行

OpenAL

```
#include <al++/al++.hpp>
#include <cmath>
struct Feeder {
    void operator()( al::playback::Buffer &_buffer ) {
        std::vector< ALshort > temp;
        temp.resize( 109 * 10 );
        std::vector< ALshort >::iterator iter;
        float pos;
        for( iter = temp.begin(); iter != temp.end(); ++iter ) {
            *iter = sinf( pos ) * 32767.0f;
            pos += 2.0f * M_PI / 9.0f;
        }
        _buffer.setData( temp.begin(), temp.end(), 48000 );
    }
};
int main() {
    using namespace al::playback;
    Device device( device_list );
    Device device( device_list.front() );
    ContextAttribute attr;
    Context context( device, attr );
    boost::shared_ptr< al::TaskRemapper > trm( new al::TaskRemapper );
    Stream stream1( context, trm, Feeder(), 10 );
    stream1.play();
    stream1.wait();
}
```

27行

OpenAL++

2011年5月22日日曜日

★これはストリーミング再生を行うOpenALで書かれたコードです。★そしてこちらが同じことをするOpenAL++で書かれたコードです。 OpenALだと★49行にもなるこのコードが、OpenAL++なら★27行です。

ストリーミング再生専用クラス

Stream

2011年5月22日 日曜日

OpenALにはコールバックがなくストリーミングが非常に煩わしいので、コールバックでストリーミング再生を行うクラス、Streamを用意しました。★再生バッファが減ってきたら★この関数オブジェクトが呼ばれるので、この中でバッファを補充するだけでストリーミングができます。★簡単ですね！

ストリーミング再生専用クラス

Stream

```
Stream stream1( context, trm, Feeder(), 10 );
```

再生バッファが減ってきたら

2011年5月22日 日曜日

OpenALにはコールバックがなくストリーミングが非常に煩わしいので、コールバックでストリーミング再生を行うクラス、Streamを用意しました。★再生バッファが減ってきたら★この関数オブジェクトが呼ばれるので、この中でバッファを補充するだけでストリーミングができます。★簡単ですね！

ストリーミング再生専用クラス

Stream

```
Stream stream1( context, trm, Feeder(), 10 );
```

再生バッファが減ってきたら



この関数オブジェクトが
呼ばれる

2011年5月22日 日曜日

OpenALにはコールバックがなくストリーミングが非常に煩わしいので、コールバックでストリーミング再生を行うクラス、Streamを用意しました。★再生バッファが減ってきたら★この関数オブジェクトが呼ばれるので、この中でバッファを補充するだけでストリーミングができます。★簡単ですね！

ストリーミング再生専用クラス

Stream

```
Stream stream1( context, trm, Feedback, 1000 );
```

再生バッファが減ってきたら

この関数オブジェクトが
呼ばれる

**EASY
TO
USE**

2011年5月22日 日曜日

OpenALにはコールバックがなくストリーミングが非常に煩わしいので、コールバックでストリーミング再生を行うクラス、Streamを用意しました。★再生バッファが減ってきたら★この関数オブジェクトが呼ばれるので、この中でバッファを補充するだけでストリーミングができます。★簡単ですね！



OpenAL

OpenAL++

2011年5月22日 日曜日

OpenALとOpenAL++を使って、★★煩わしいコードの要らない幸せなオーディオプログラミングを始めましょう。

The background is a dark, atmospheric scene featuring a stone archway in the distance. In the foreground and middle ground, there are several wooden crates stacked on a cobblestone floor. The crates have the text 'OpenGL' written on them in a stylized font. The lighting is dim, creating a sense of mystery and depth.

OpenAL

+ OpenAL++

2011年5月22日日曜日

OpenALとOpenAL++を使って、★★煩わしいコードの要らない幸せなオーディオプログラミングを始めましょう。



OpenAL

+ OpenAL++

HAPPY!

2011年5月22日 日曜日

OpenALとOpenAL++を使って、★★煩わしいコードの要らない幸せなオーディオプログラミングを始めましょう。

問題点



2011年5月22日日曜日

さて、とはいえ、OpenAL++にも色々既知の問題があります。★まずMacでStreamが不思議な踊りを踊ります。★これはどうもBoost.Threadのバグを踏んでる臭い挙動をしているのでもう少し突っ込んで調べてみないと、という状況です。★録音機能には対応していません。★これについては当面对応する予定はありませんが、パッチは大歓迎です。★Windowsで試していません。★基本的にクロス

問題点

Macで不思議な踊りを踊る



2011年5月22日日曜日

さて、とはいえ、OpenAL++にも色々既知の問題があります。★まずMacでStreamが不思議な踊りを踊ります。★これはどうもBoost.Threadのバグを踏んでる臭い挙動をしているのでもう少し突っ込んで調べてみないと、という状況です。★録音機能には対応していません。★これについては当面对応する予定はありませんが、パッチは大歓迎です。★Windowsで試していません。★基本的にクロス

問題点

Macで不思議な踊りを踊る

Boost.Threadのバグを踏んでる？

2011年5月22日日曜日

さて、とはいえ、OpenAL++にも色々既知の問題があります。★まずMacでStreamが不思議な踊りを踊ります。★これはどうもBoost.Threadのバグを踏んでる臭い挙動をしているのでもう少し突っ込んで調べてみないと、という状況です。★録音機能には対応していません。★これについては当面对応する予定はありませんが、パッチは大歓迎です。★Windowsで試していません。★基本的にクロス

問題点

Macで不思議な踊りを踊る

Boost.Threadのバグを踏んでる？

録音機能未対応

2011年5月22日 日曜日

さて、とはいえ、OpenAL++にも色々既知の問題があります。★まずMacでStreamが不思議な踊りを踊ります。★これはどうもBoost.Threadのバグを踏んでる臭い挙動をしているのでもう少し突っ込んで調べてみないと、という状況です。★録音機能には対応していません。★これについては当面对応する予定はありませんが、パッチは大歓迎です。★Windowsで試していません。★基本的にクロス

問題点

Macで不思議な踊りを踊る

Boost.Threadのバグを踏んでる？

録音機能未対応

今のところ対応予定なし

2011年5月22日日曜日

さて、とはいえ、OpenAL++にも色々既知の問題があります。★まずMacでStreamが不思議な踊りを踊ります。★これはどうもBoost.Threadのバグを踏んでる臭い挙動をしているのでもう少し突っ込んで調べてみないと、という状況です。★録音機能には対応していません。★これについては当面对応する予定はありませんが、パッチは大歓迎です。★Windowsで試していません。★基本的にクロス

問題点

Macで不思議な踊りを踊る

Boost.Threadのバグを踏んでる？

録音機能未対応

今のところ対応予定なし

Windowsで試していない

2011年5月22日 日曜日

さて、とはいえ、OpenAL++にも色々既知の問題があります。★まずMacでStreamが不思議な踊りを踊ります。★これはどうもBoost.Threadのバグを踏んでる臭い挙動をしているのでもう少し突っ込んで調べてみないと、という状況です。★録音機能には対応していません。★これについては当面对応する予定はありませんが、パッチは大歓迎です。★Windowsで試していません。★基本的にクロス

問題点

Macで不思議な踊りを踊る

Boost.Threadのバグを踏んでる？

録音機能未対応

今のところ対応予定なし

Windowsで試していない

動いたら奇跡だと思います

2011年5月22日 日曜日

さて、とはいえ、OpenAL++にも色々既知の問題があります。★まずMacでStreamが不思議な踊りを踊ります。★これはどうもBoost.Threadのバグを踏んでる臭い挙動をしているのでもう少し突っ込んで調べてみないと、という状況です。★録音機能には対応していません。★これについては当面对応する予定はありませんが、パッチは大歓迎です。★Windowsで試していません。★基本的にクロス

問題点

Macで不思議な踊りを踊る

Boost.Threadのバグを踏んでる？

録音機能未対応

今のところ対応予定なし

Windowsで試してない

動いたら奇跡だと思います

C++ かわいい

2011年5月22日 日曜日

さて、とはいえ、OpenAL++にも色々既知の問題があります。★まずMacでStreamが不思議な踊りを踊ります。★これはどうもBoost.Threadのバグを踏んでる臭い挙動をしているのでもう少し突っ込んで調べてみないと、という状況です。★録音機能には対応していません。★これについては当面对応する予定はありませんが、パッチは大歓迎です。★Windowsで試していません。★基本的にクロス

問題点

Macで不思議な踊りを踊る

Boost.Threadのバグを踏んでる？

録音機能未対応

今のところ対応予定なし

Windowsで試してない

動いたら奇跡だと思います

C++ かわいい

C++はかわいいらしい

2011年5月22日 日曜日

さて、とはいえ、OpenAL++にも色々既知の問題があります。★まずMacでStreamが不思議な踊りを踊ります。★これはどうもBoost.Threadのバグを踏んでる臭い挙動をしているのでもう少し突っ込んで調べてみないと、という状況です。★録音機能には対応していません。★これについては当面对応する予定はありませんが、パッチは大歓迎です。★Windowsで試していません。★基本的にクロス



復習資料掲載予定URL

<http://bit.ly/eQKvdE>



Thank you for listening