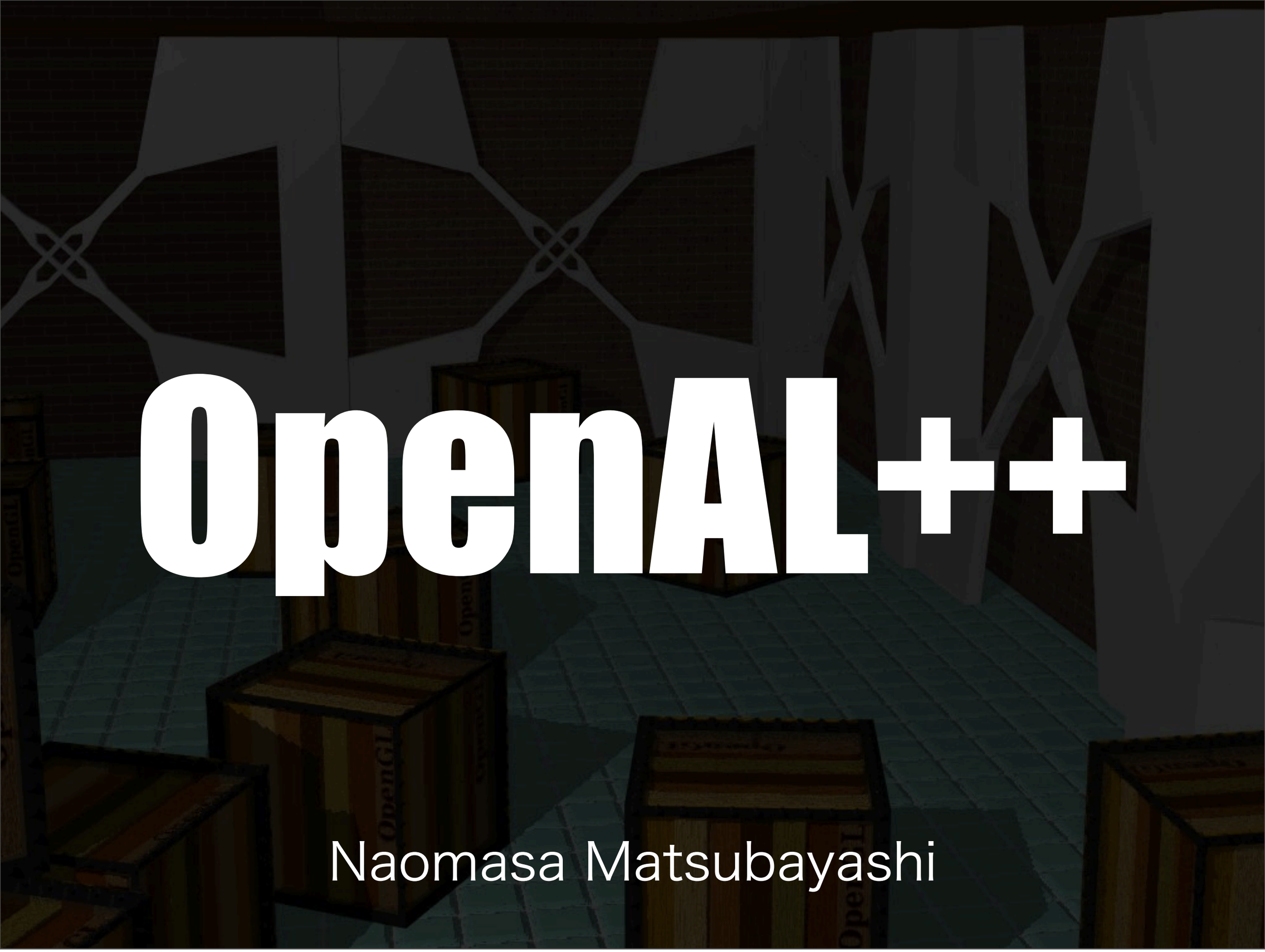
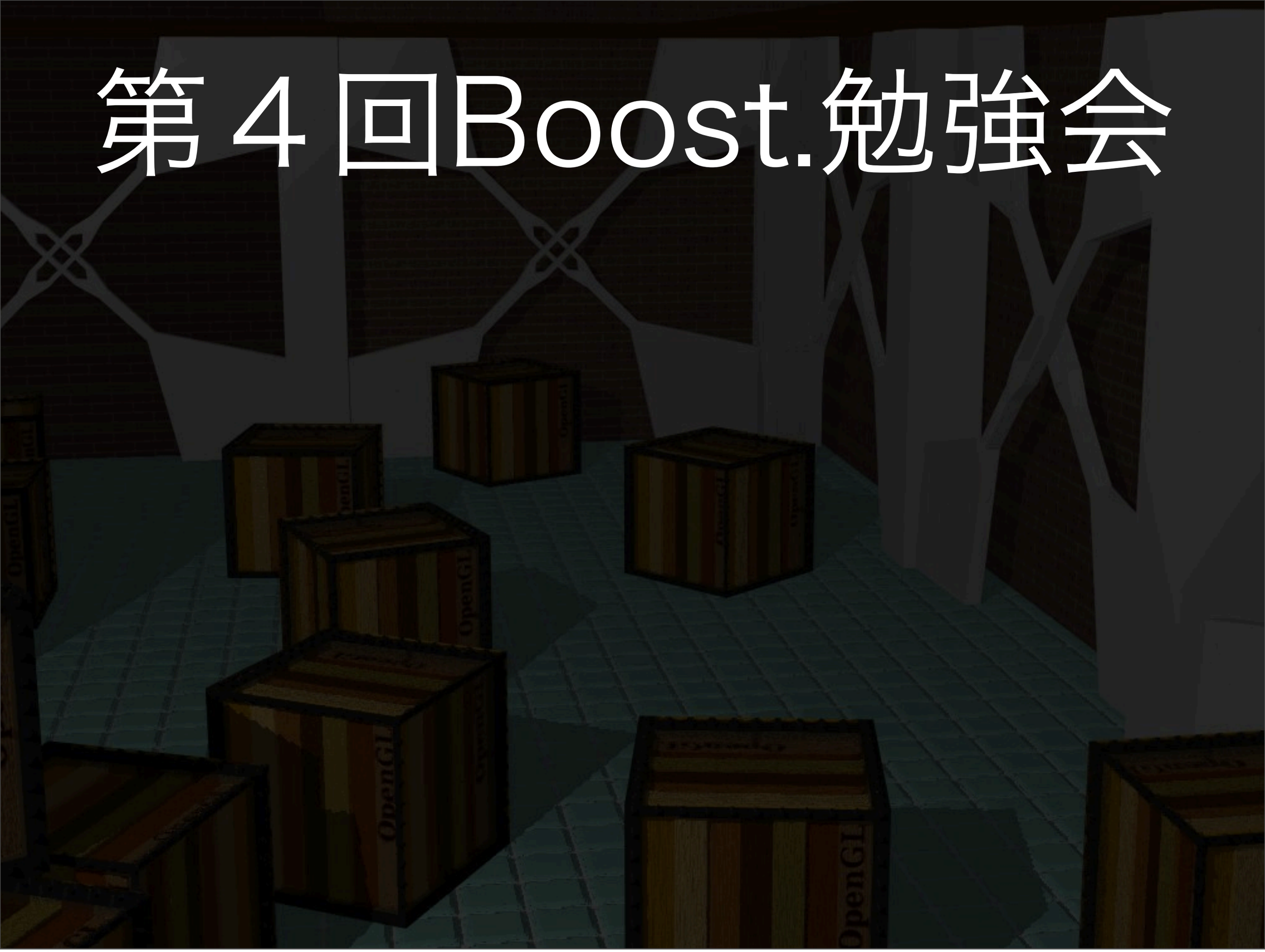

OpenAL++
Naomasa Matsubayashi

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。

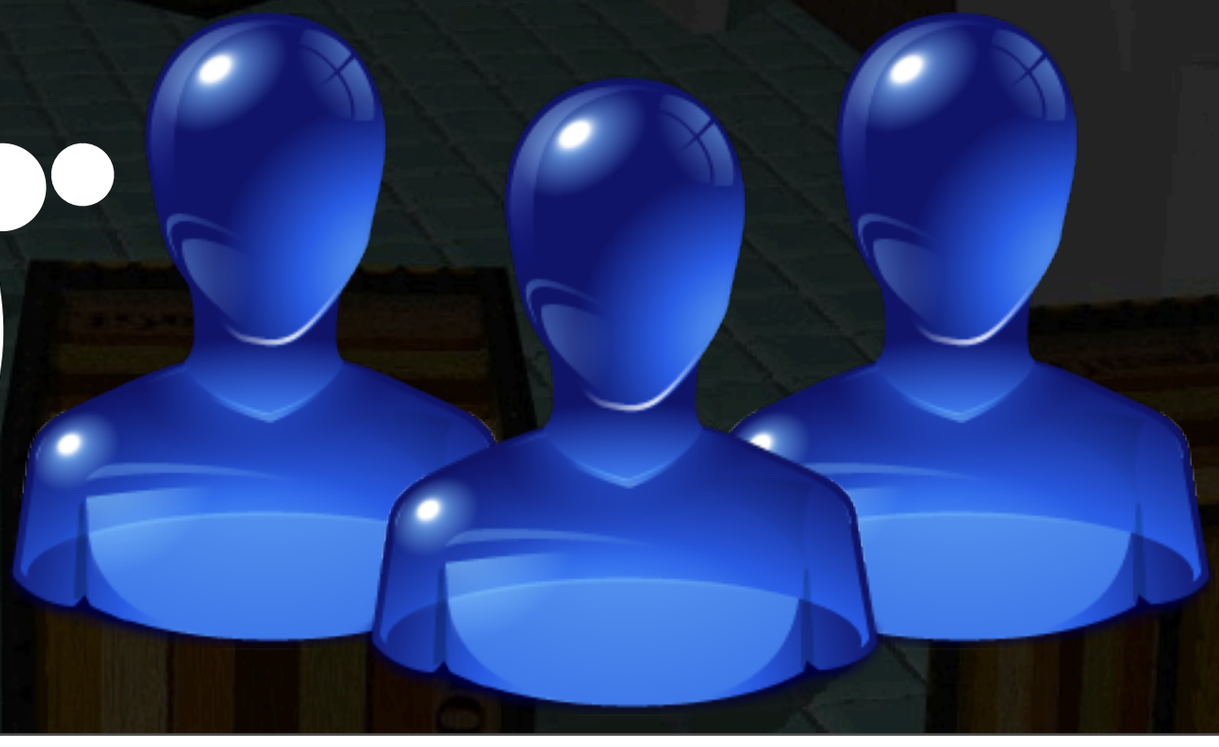
第４回Boost.勉強会

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。

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

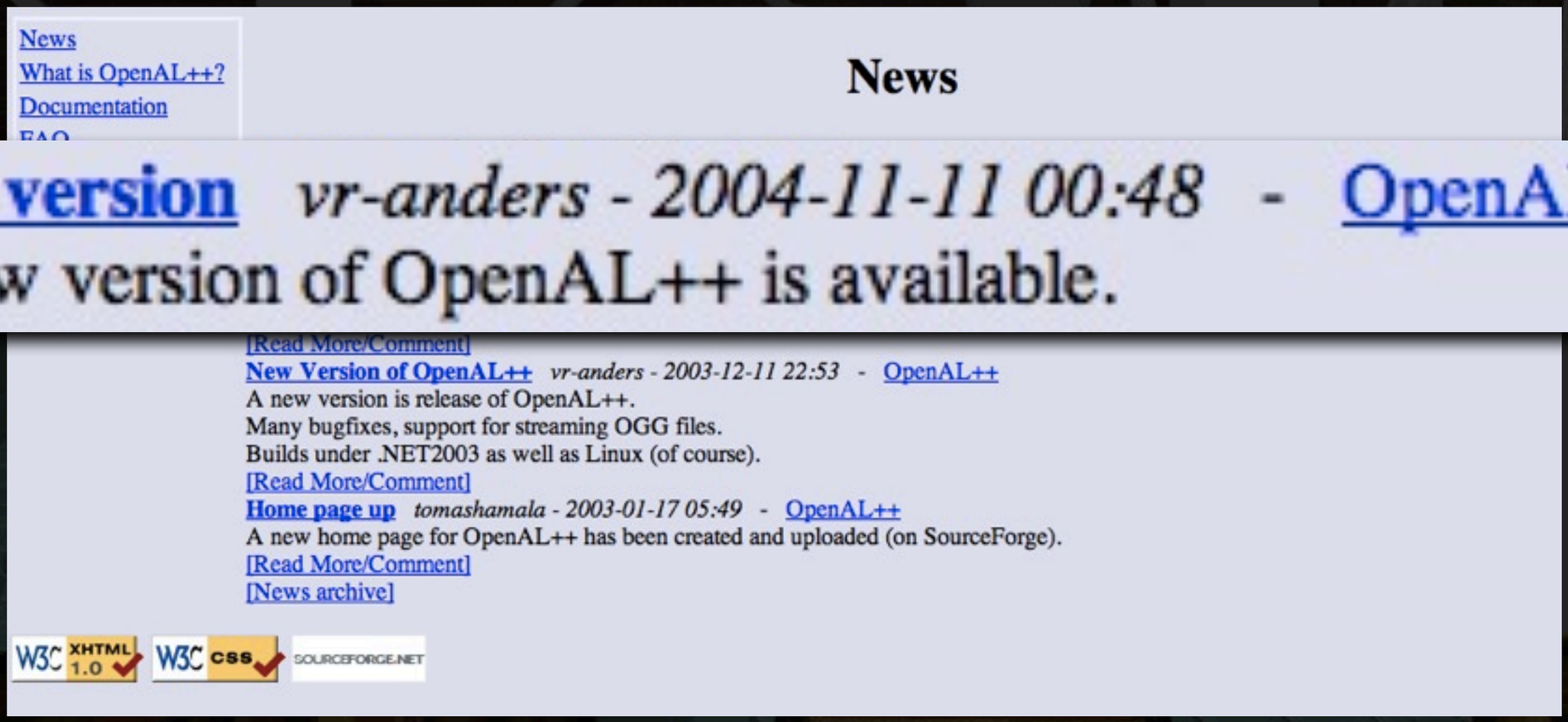
C++ラッパはある

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

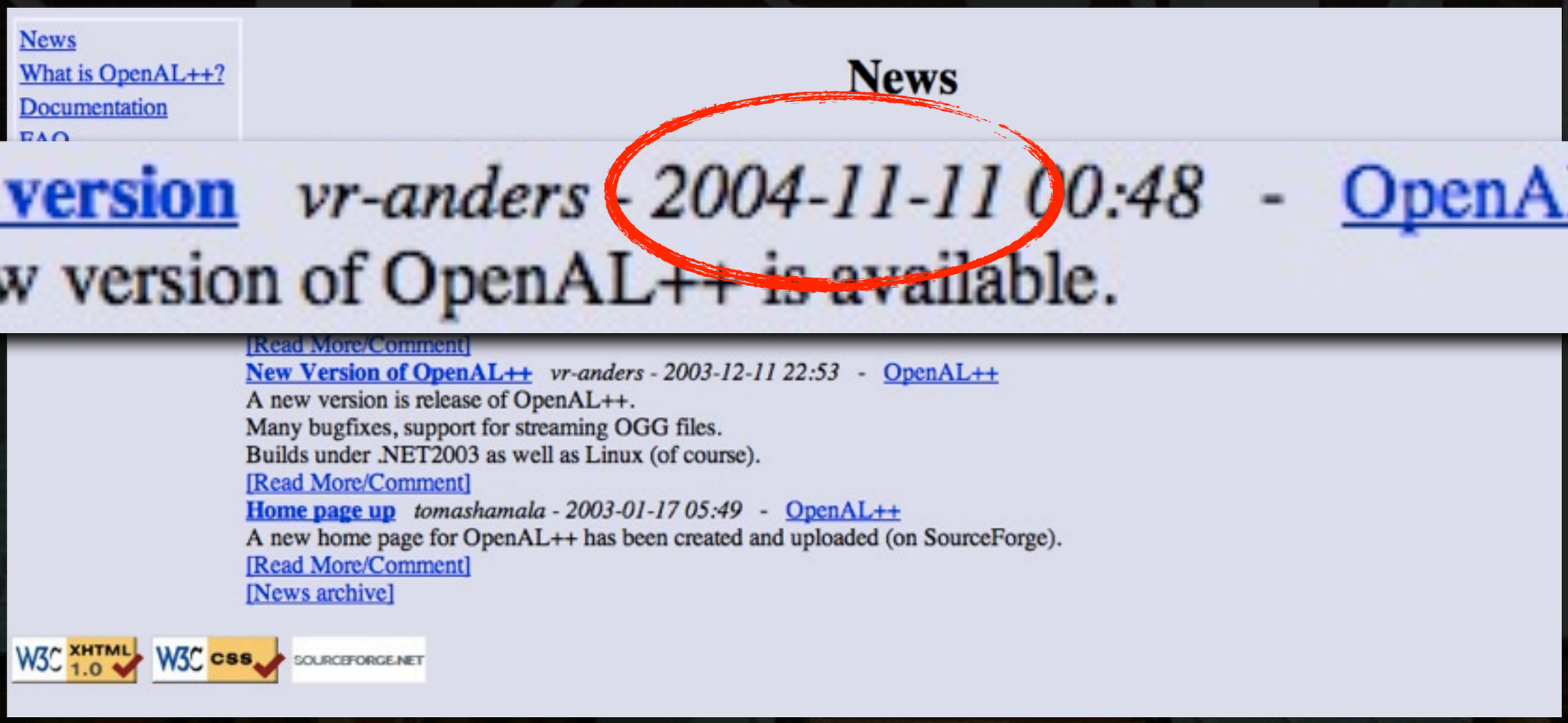


2011年5月22日日曜日

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

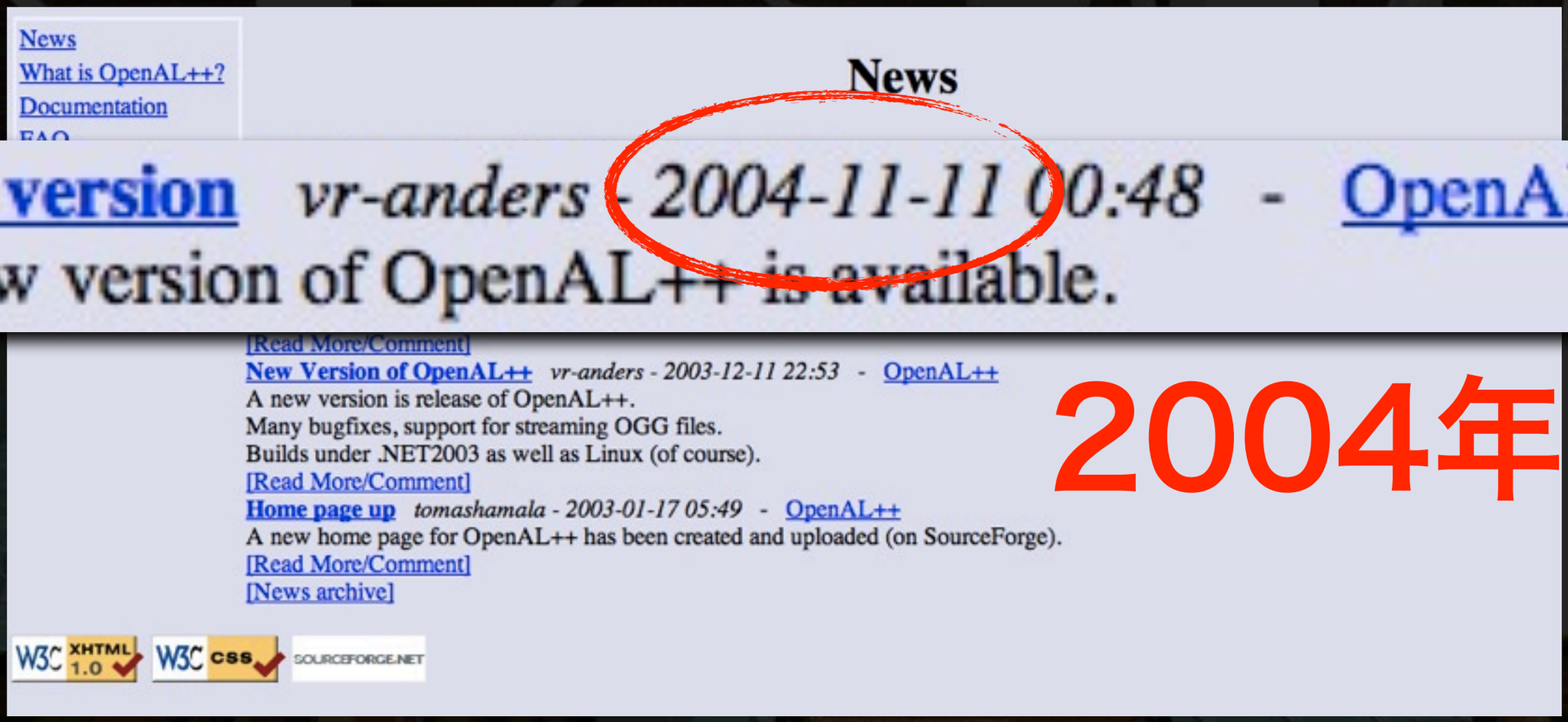


2011年5月22日日曜日

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



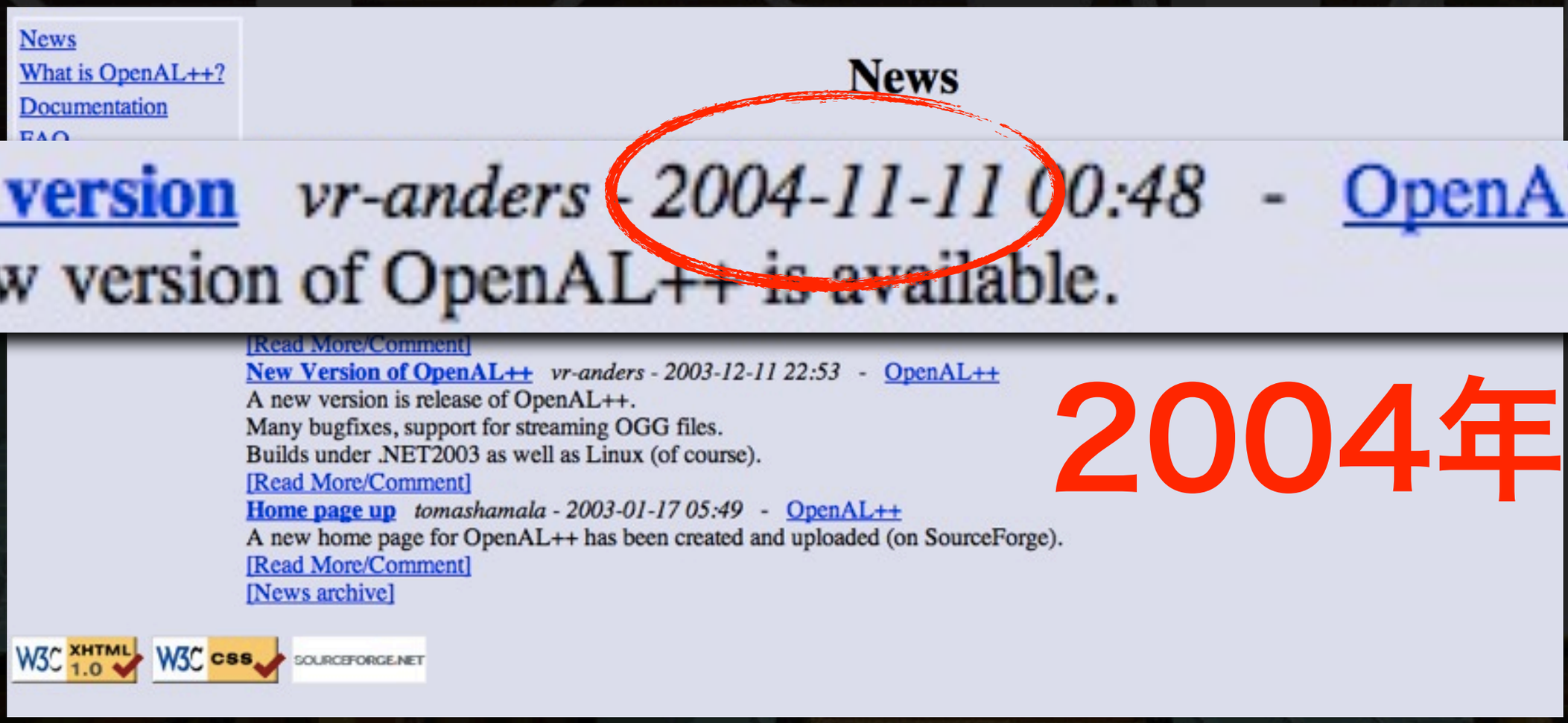
2004年?

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



2004年?

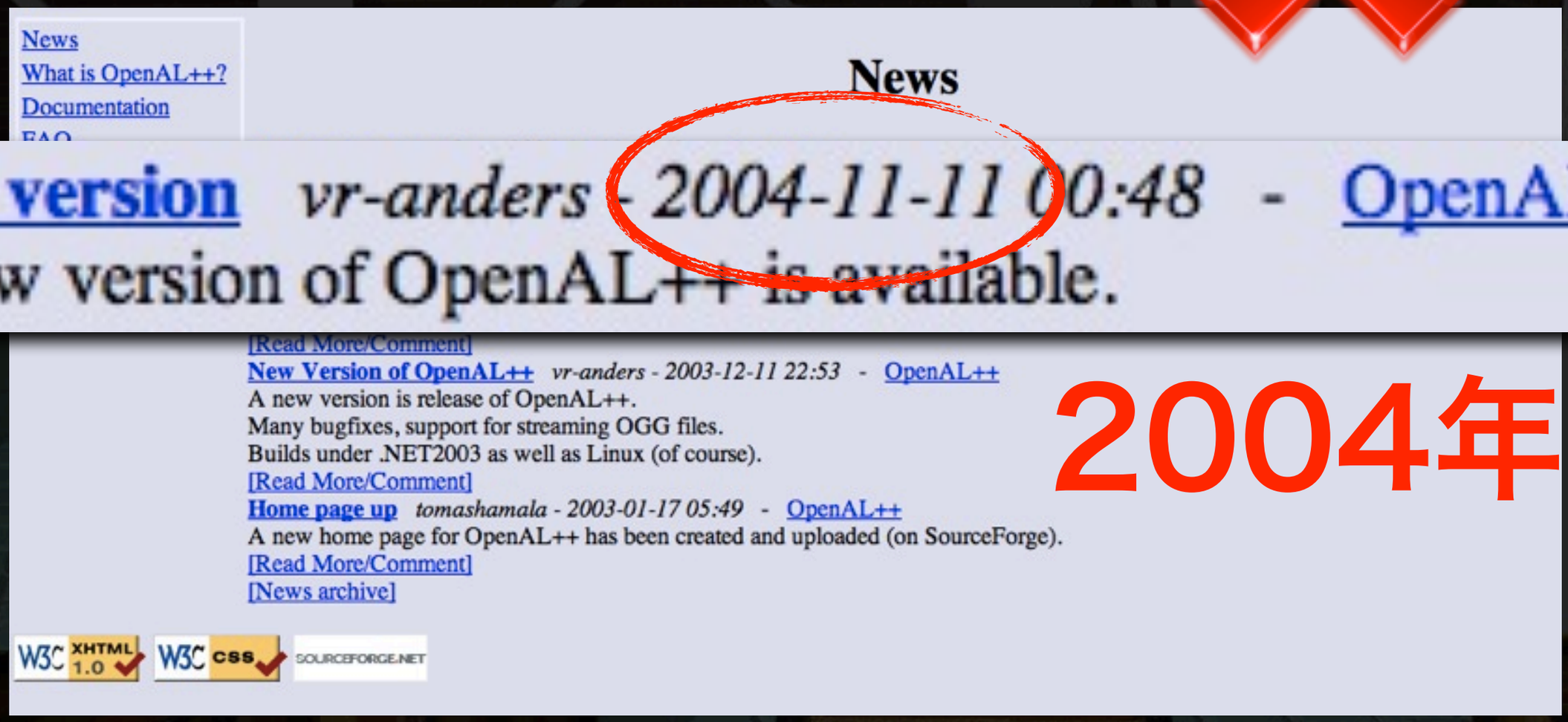
開発が止まっている

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。

OpenALにも

C++ラッパはあ❌ら



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。

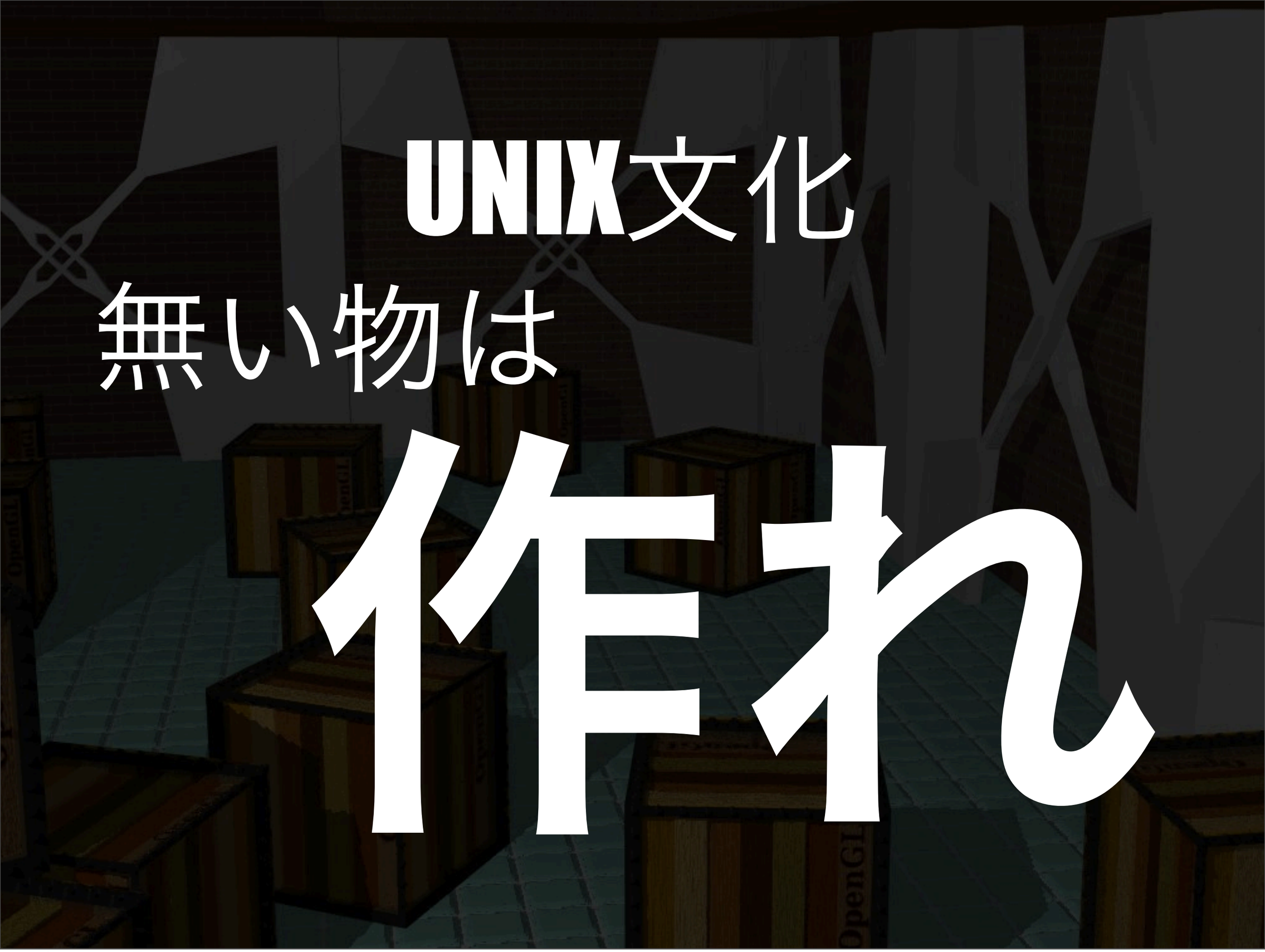
UNIX文化

2011年5月22日日曜日

UNIX文化では☆ないものは、★作れです。

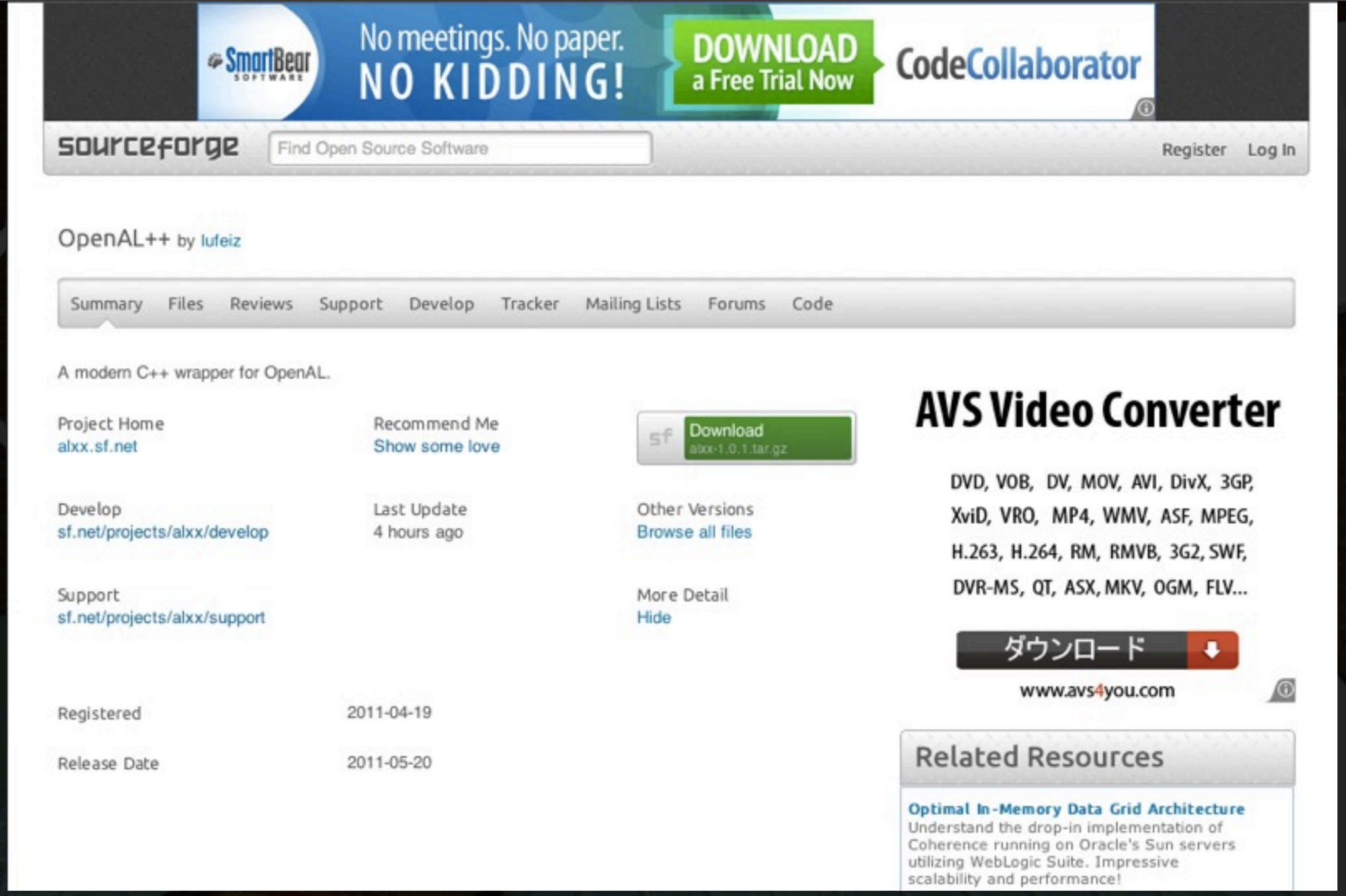
2011年5月22日日曜日

UNIX文化では☆ないものは、★作れです。



2011年5月22日日曜日

UNIX文化では☆ないものは、★作れです。

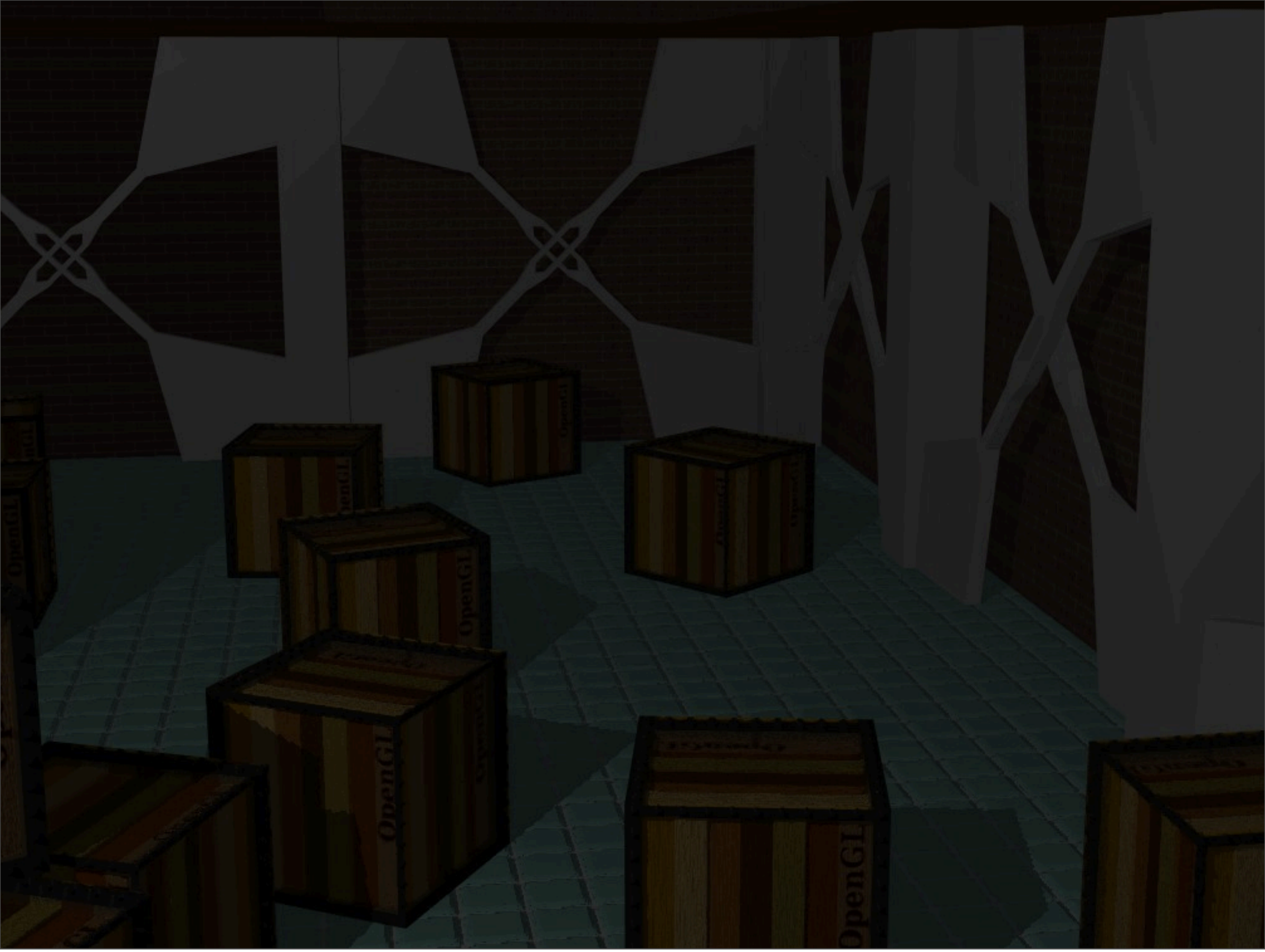


http://sourceforge.net/projects/alxx/

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

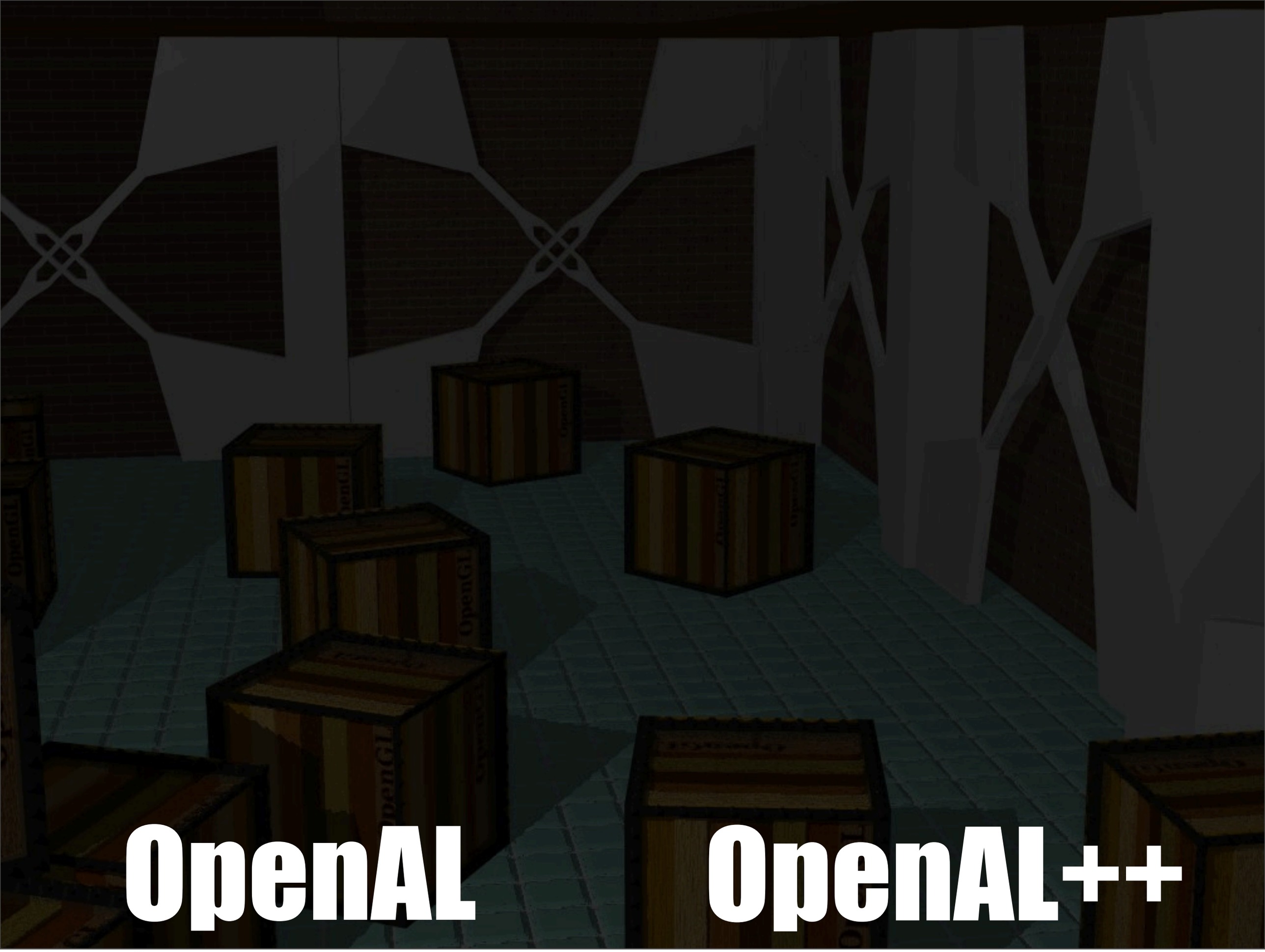
紛らわしいので見分け方ですが、★略称がalppなヤツが開発が止まってるOpenAL++で、★alxxなヤツが今回作ったOpenAL++です。

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。

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。



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 );
}
```

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();
}
```

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;
  alGenSo    1, &   urce );
  ALuint buff ;
  alGenBuf   1, &b   ffer );
  alBufferData
    buff    AL_FORMAT_MONO16, epian ,
    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

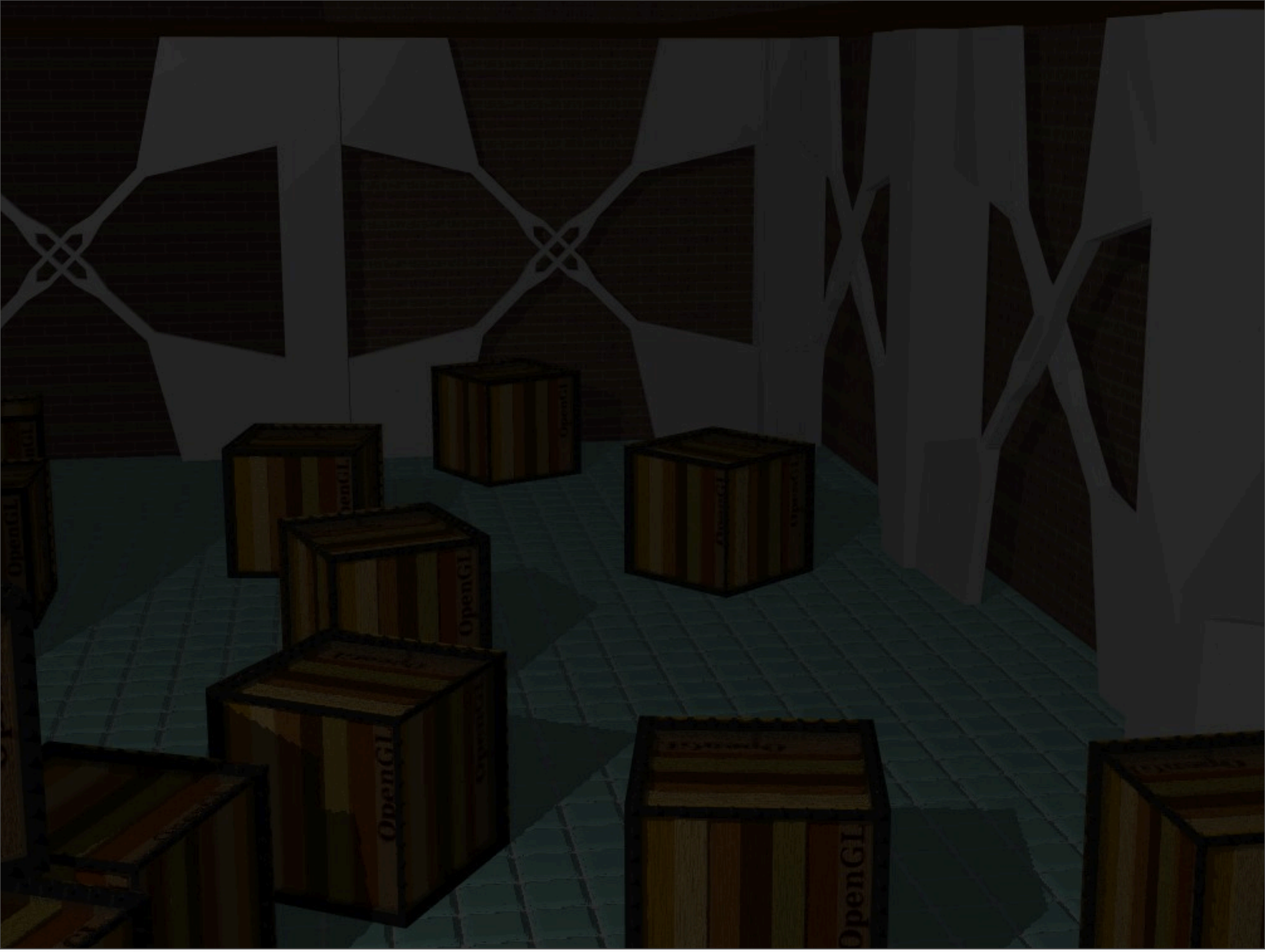
```
#include <al++/al++.hpp>
#include "epiano.h"
int main() {
  using namespace al::playback;
  DeviceList device_list;
  Devic  devic    device_list.front() );
  Contex  conte  t( device );
  Source  sourc   co   ext );
  Buffer  uffer( con   ext );
  buffer  setDat   ano,
    epiano + EPIANO_LENGT , 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

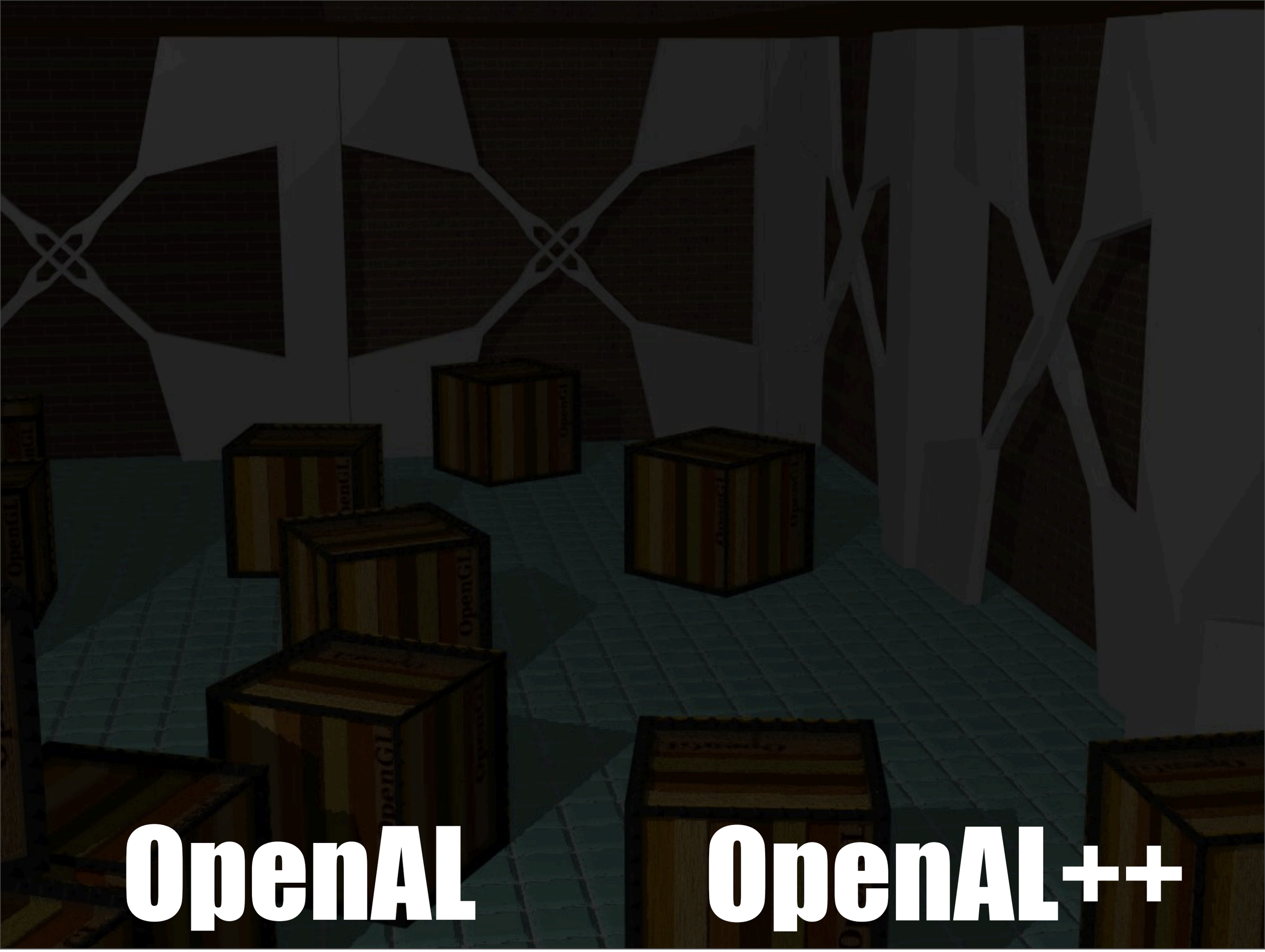
```
buffer.setData( epiano,
  epiano + EPIANO_LENGTH, 48000 );
source.queueBuffer( buffer );
source.play();
source.wait();
```

EASY TO USE

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。



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
  std::vec
  using name
  qi::parse(
    +ascii::graph
  std::co
  BOOST_FO
    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 );
}
```

35行

OpenAL

```
#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++

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
  std::vec
  using name
  qi::parse(
    +ascii::graph
  std::co
  BOOST_FO
    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 );
}
```

35行

```
#include <iostream>
#include <al++/al++.hpp>
int main() {
  using
  DeviceLis
  Device de
  Context c
  ALCExtens
  std::cout
  std::cout
  ALExtensionList alel( context );
  std::cout << "== AL Extensions ==" << std::endl;
  std::cout << alel;
}
```

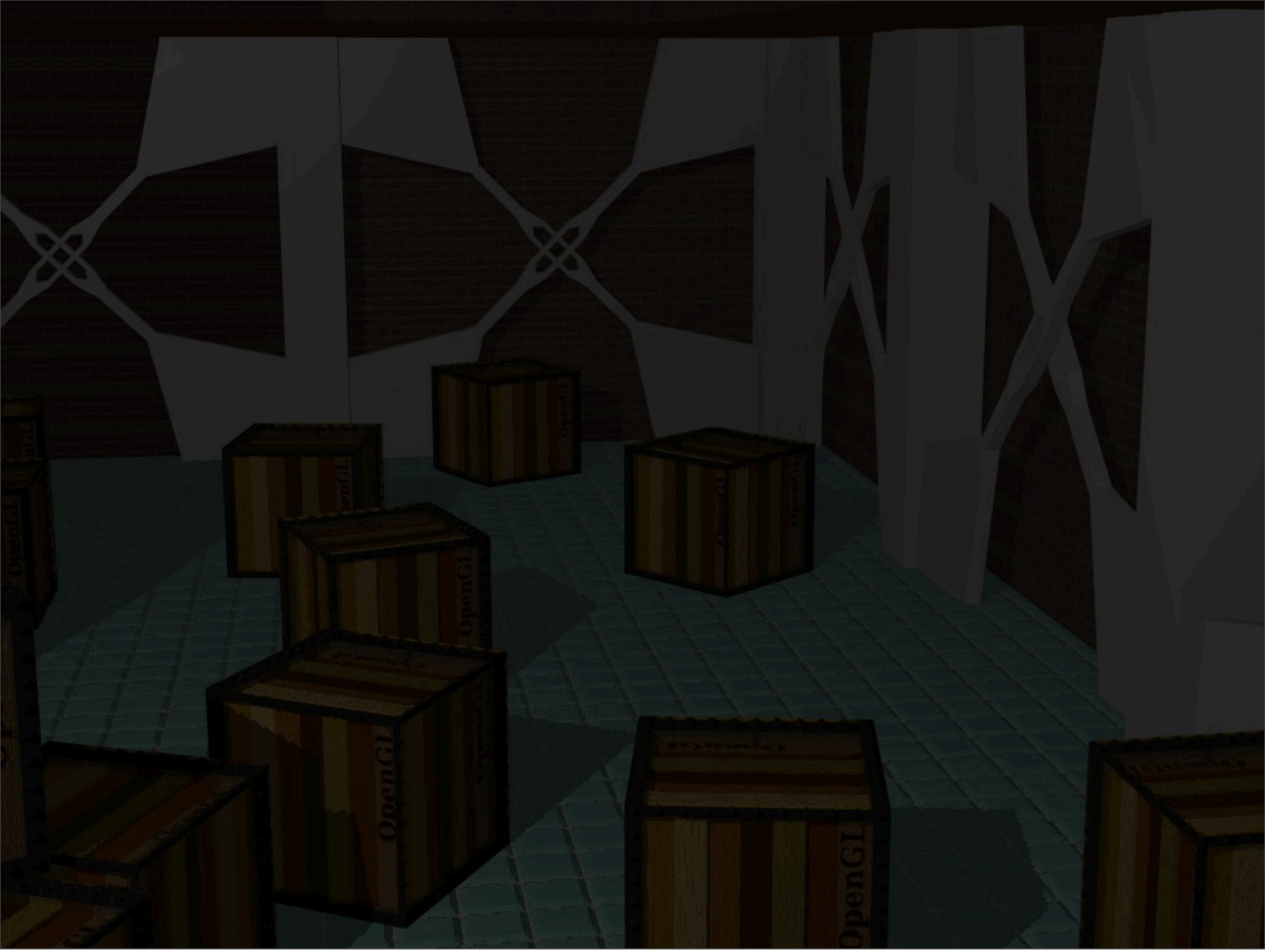
14行

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す。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す。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す。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 );
```

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す。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す。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

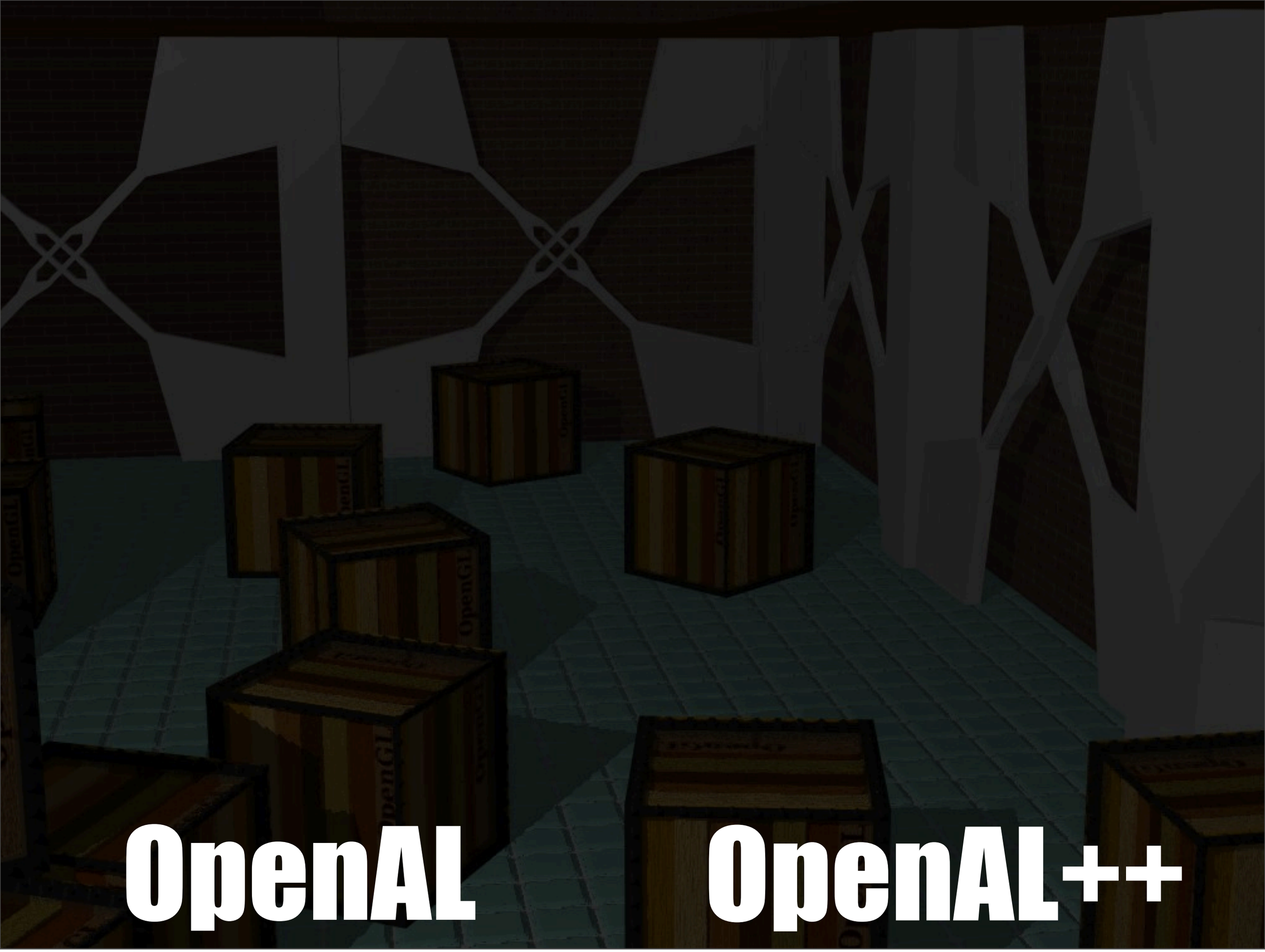
```
std::cout << alel;
```

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す。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！



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 += 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 != 10000; 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();
}
```

49行

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 != 10000; 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 = sin( pos ) * 32767;
    }
    _buffer.set( 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();
}
```

49行

27行

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。

ストリーミング再生専用クラス

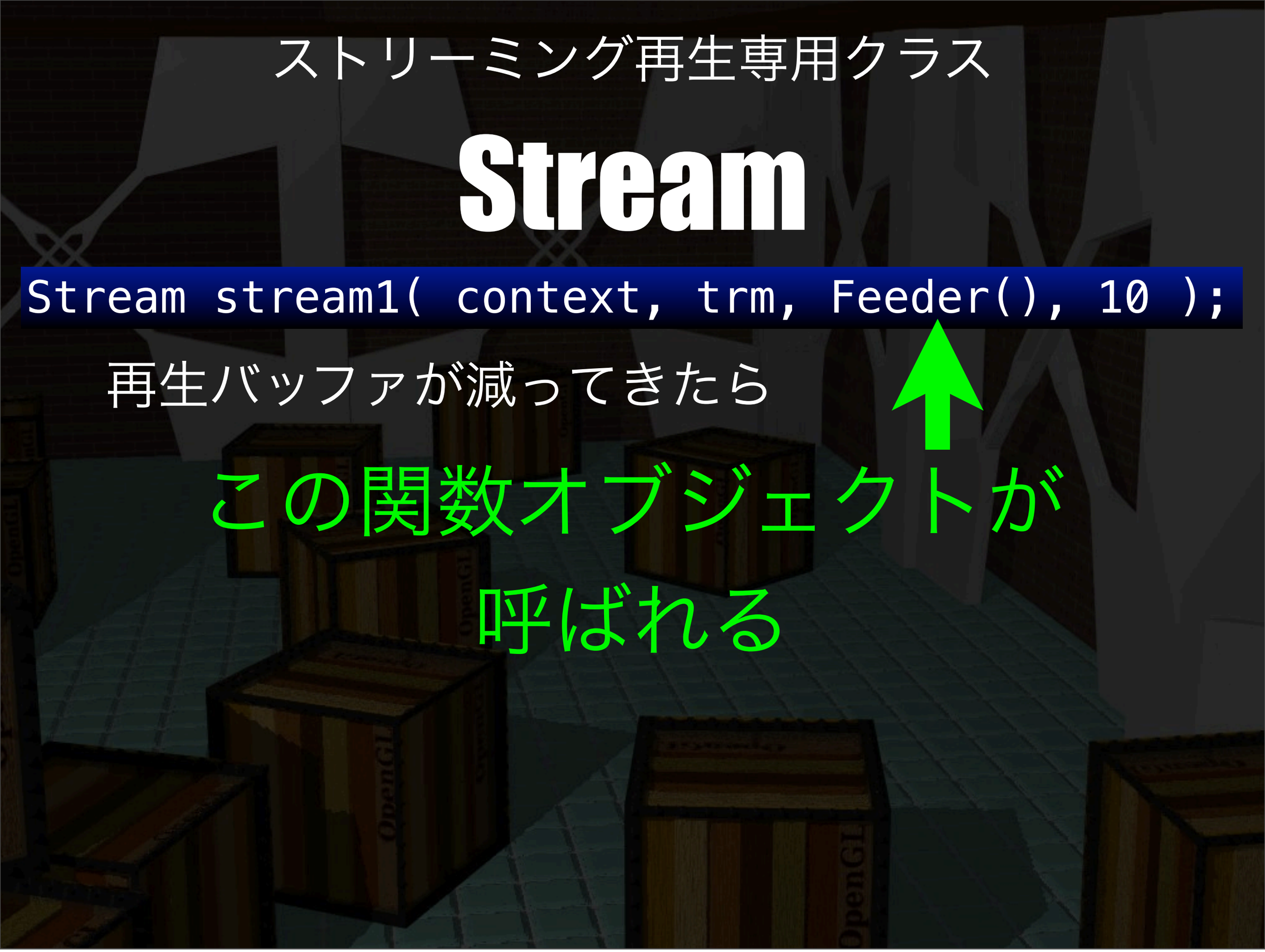
Stream

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！

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！



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！

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！

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。

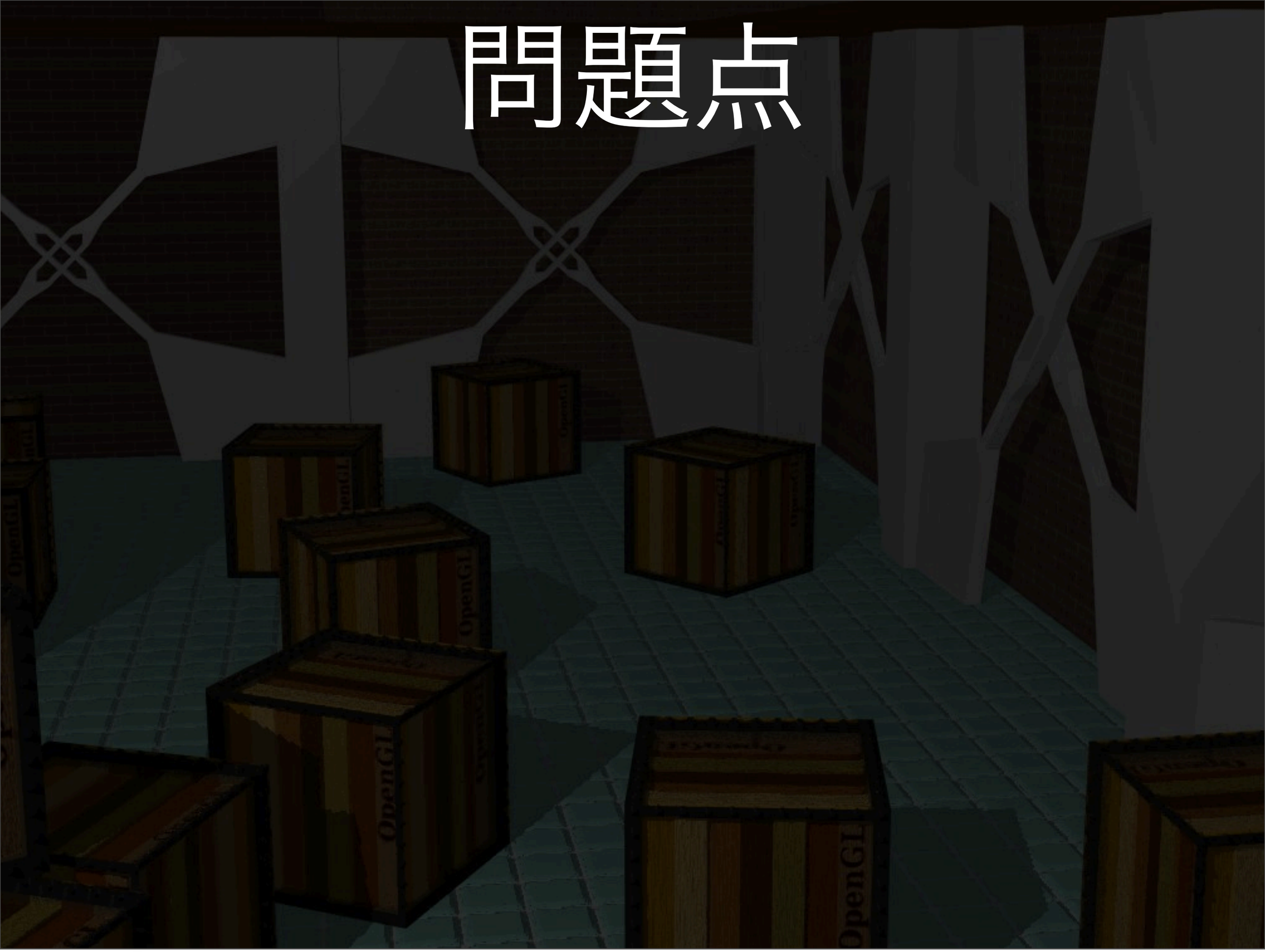
2011年5月22日日曜日

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。

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対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対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

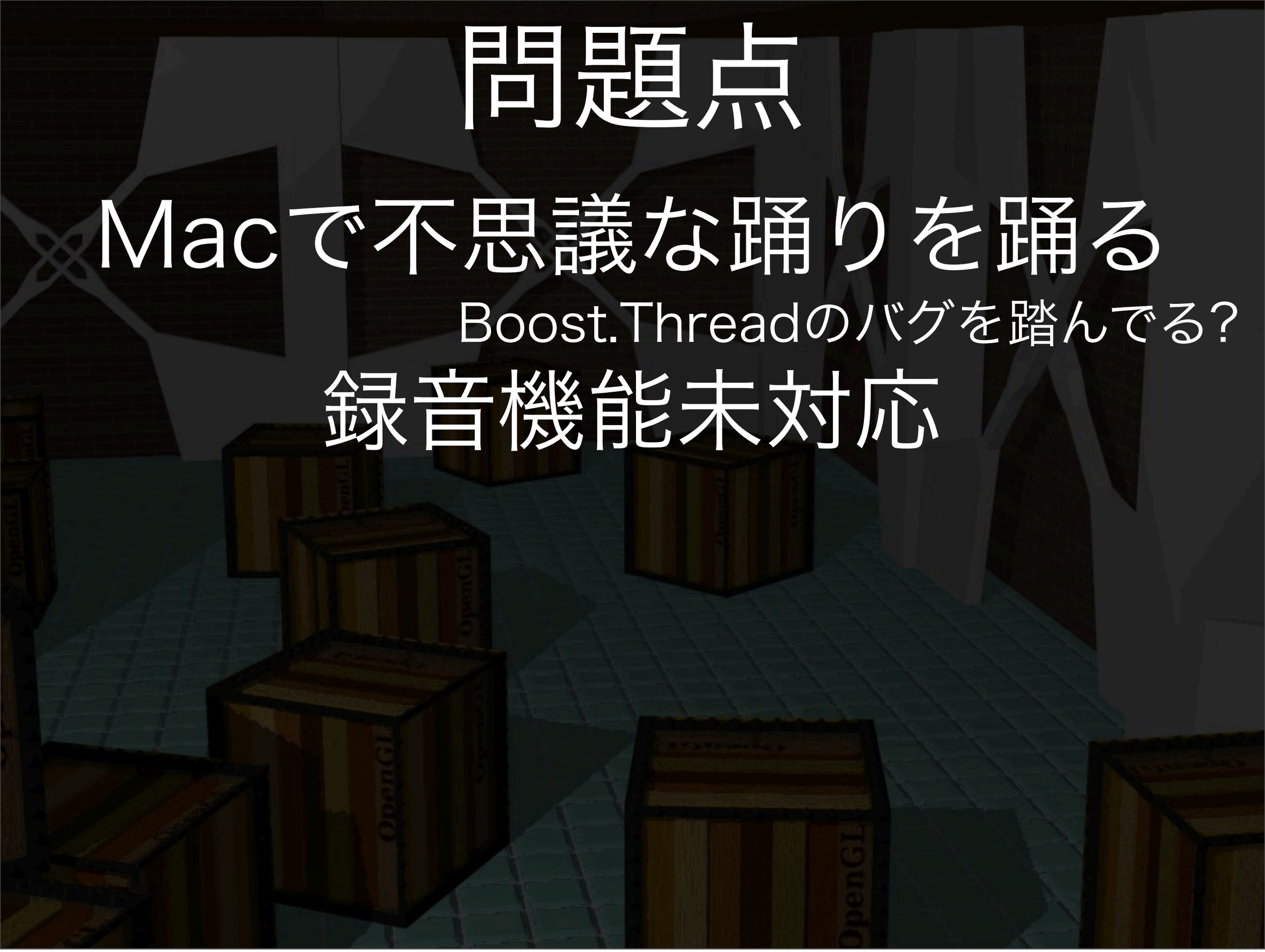
問題点

Macで不思議な踊りを踊る

Boost.Threadのバグを踏んでる?

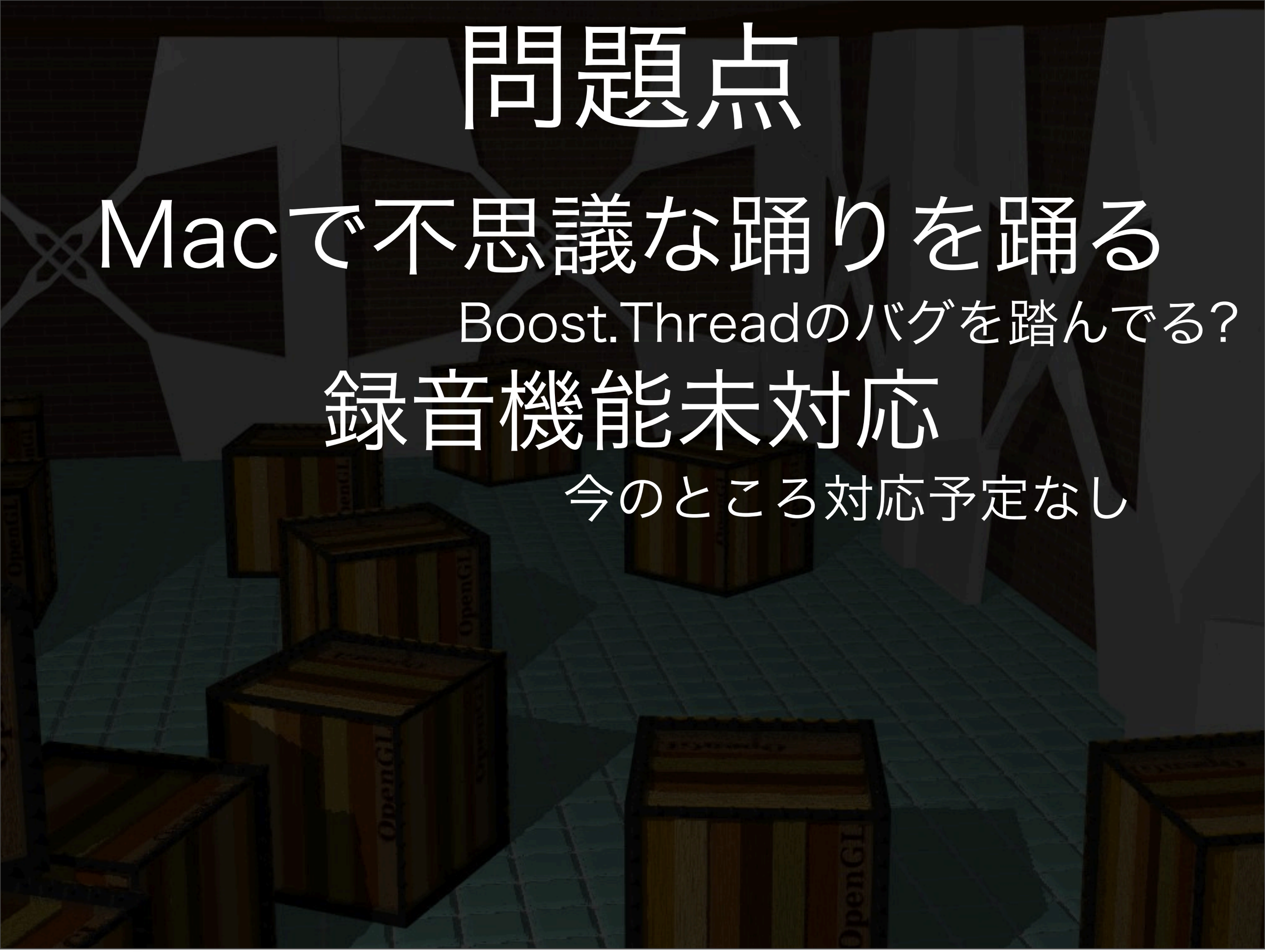
2011年5月22日日曜日

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対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



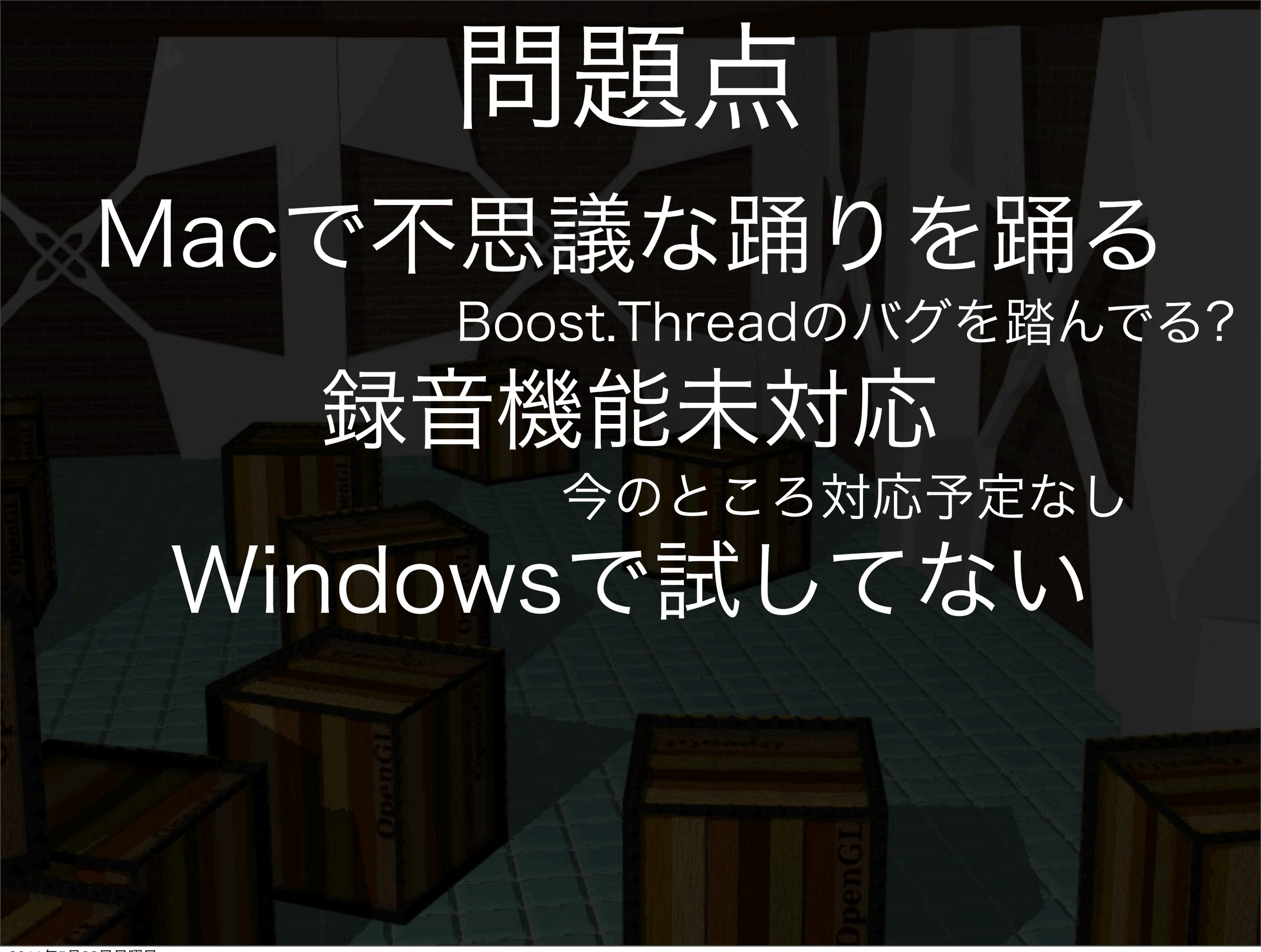
2011年5月22日日曜日

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対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



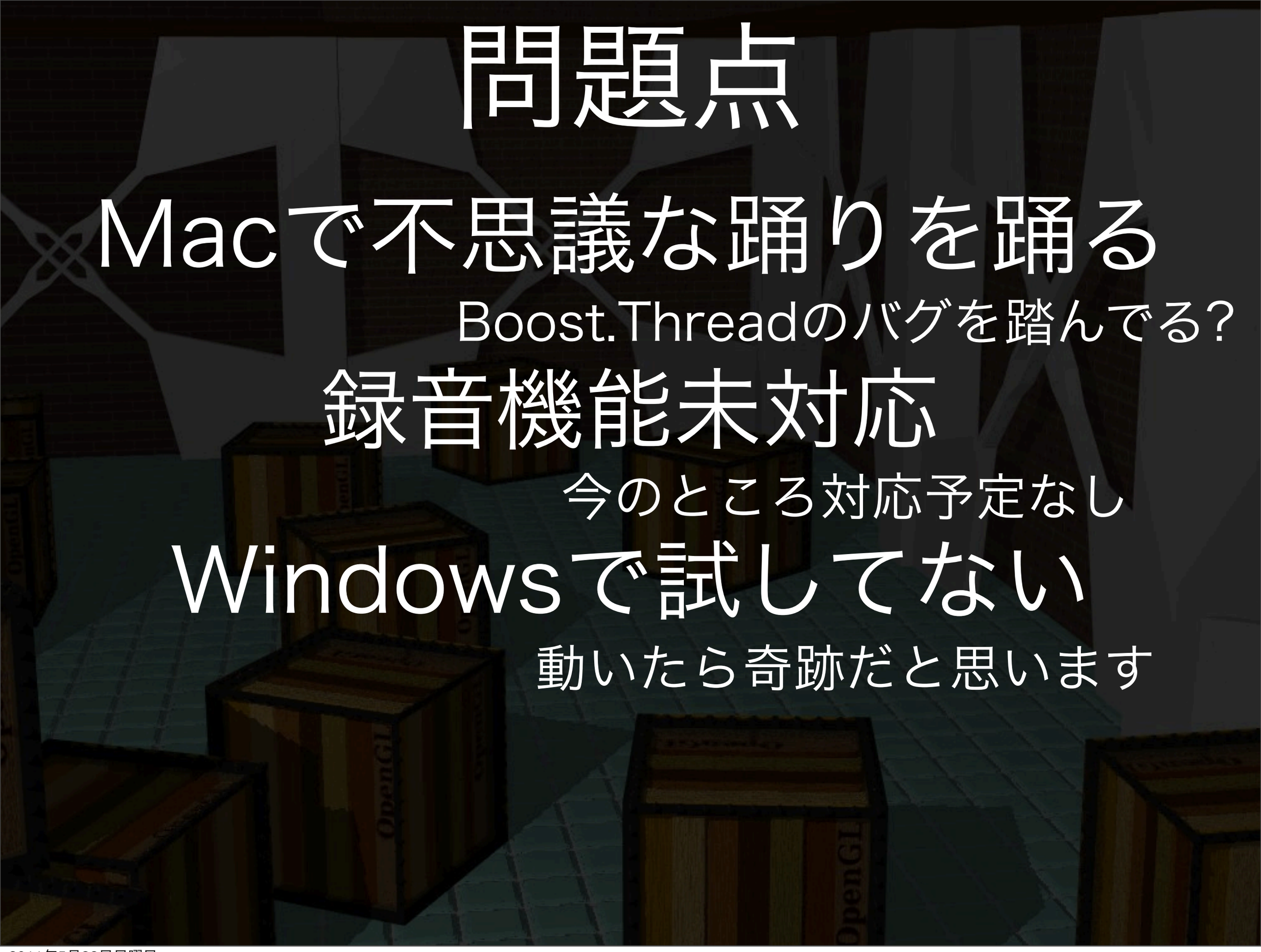
2011年5月22日日曜日

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対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



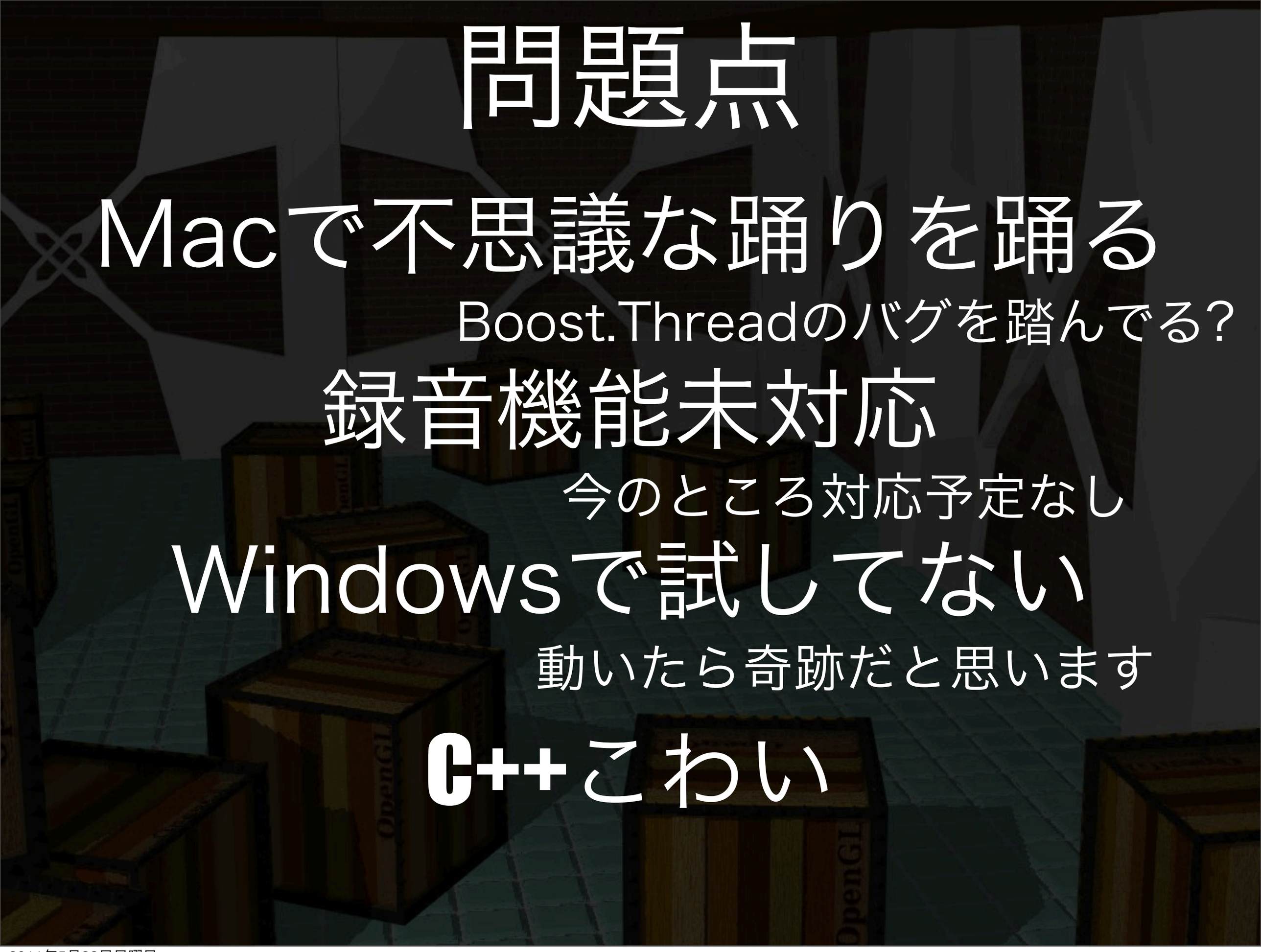
2011年5月22日日曜日

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対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



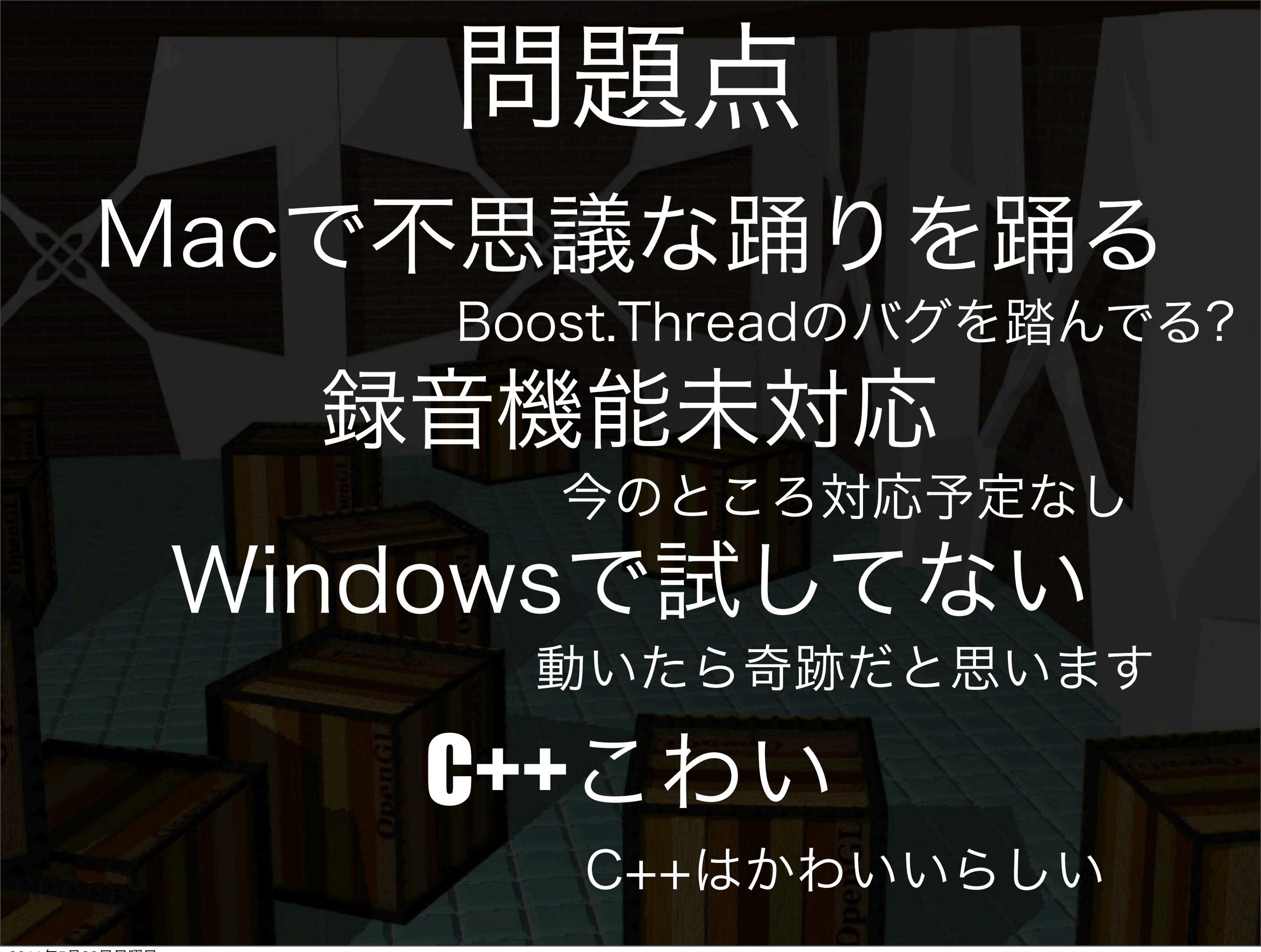
2011年5月22日日曜日

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対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



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対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



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対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

2011年5月22日日曜日


Thank you for listening