ゲームプログラミング
- 応用編 -

神奈川工科大学 情報メディア学科
白井暁彦

KAIT KANAGAWA INSTITUTE of TECHNOLOGY ShiraiLab

http://blog.shirai.la

“自己効力感”を向上させよう

ShiraiLab

* 就職活動でグループワークがあったりする
  * 課題「この中でうちの会社に一番向いてそうな人を推薦して」
  * 協調性だけでは絶対に残れない高度な課題！

* 「自己効力感」
  * 自分の人生にどれだけコミットできるか？（Commit=貢献）
  * 事故などの不可抗力ではなく、自分の人生に自分で寄与する力
  * ゼロ(0)をイチ(1)にすることの積み上げであり、
    ある日突然ジュウ(10)になったりしない
  * ゼロをイチにすることがこの時期（3年生）でどれだけ大変か！
* 協調性も大事だが、やれることを増やしていこう

第10回課題 応用

いままでのゲームエンジンの知識を使ってゲームを作ってみよう
- オリジナルでなくてかまいません
- プログラムを提出しなくてよい
- UGC奨励プログラムを使いこなそう

http://open.channel.or.jp/

1. PowerPointで仕様を書く
2. 今までの知識技術を使って実装してみる
3. 動画にして公開できるようにがんばってみる

以上の期限が1週間

- 当たり判定のルールなどをまず明記
- オリジナルでなくてもよい（目コピ歓迎）
  - たとえば…「2048」とか「クッキークリッカー」
  - グラフィックに凝りすぎない。アスキーアートでもよい
  - 画像素材、音素材を使った場合は、ライセンスかURLを明記
- 提出できるのはURLが2つだけ。

1. 動画のURL(Youtube,nicovideo, mp4などのfile)，
2. ソースやPowerPoint資料等のZIP圧縮をURLで。

第10回課題 応用

■評価のポイント

- 応用として「ゲームが作れるか？」 → 巨大なものを作ろうとしすぎる！
- 時間内に作れるか？ → やれれば、やれるほど、やりたいことが増えてくる！
  → 何が終わりなのか、自分でもわからない！！
- 「学んだ技術を生かしているか？」
- バンダイナムコ カタログIPオープン化プロジェクト
http://open.channel.or.jp/terms_user.php
- 株式会社バンダイナムコエンターテインメント クリエイター奨励プログラム
親作品動画 http://nico.ms/1432620077
- 任天堂株式会社の著作物利用について
http://faq.nicovideo.jp/EokpControl?&tid=289717&event=FE0006
実は任天堂のタイトルもかなり幅広く使えます。
ドット絵を利用する場合は作業者に感謝の気持ちも伝えたい
http://babsika.cocolog-nifty.com/…/20…/04/mame_icons_09.html
http://park12.wakwak.com/~non/atelier/XEVI.htm

提出URLはいつもの通り
http://j.mp/KAIT2015GP
提出期限は6/28(日) 20:59 とします（講義準備の都合上、それ以降は評価・講評の保障なし）。
なお、第11回は6/29(月),第12回は7/6(月)、第13回は7/9(木)の補講になる見込みです。

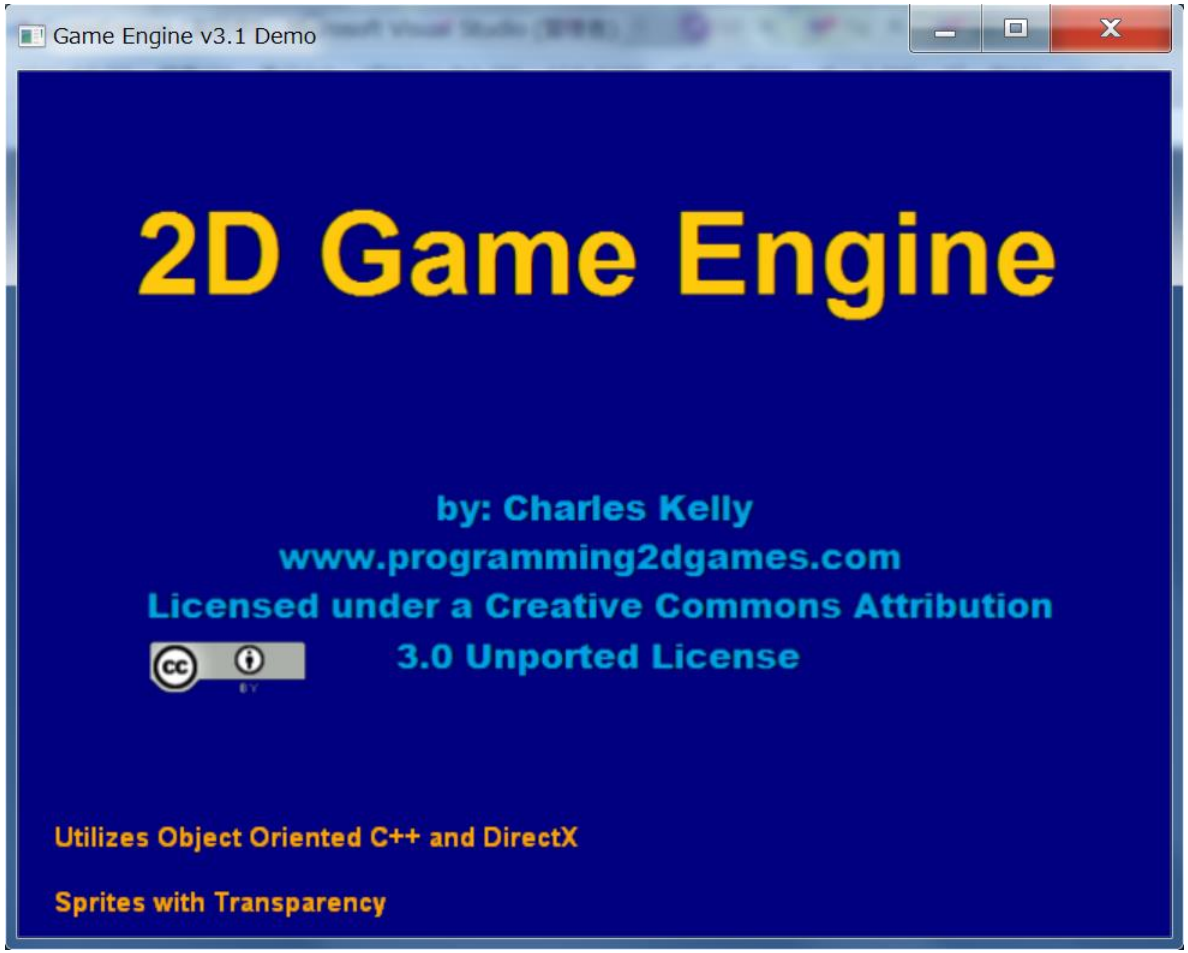
GameEngineV3.1をつかってみる

ShiraiLab

* SampleCode¥Chapter11¥GameEngine¥GameEngineV3.1
* CCライセンス
* なんか英語出るだけ

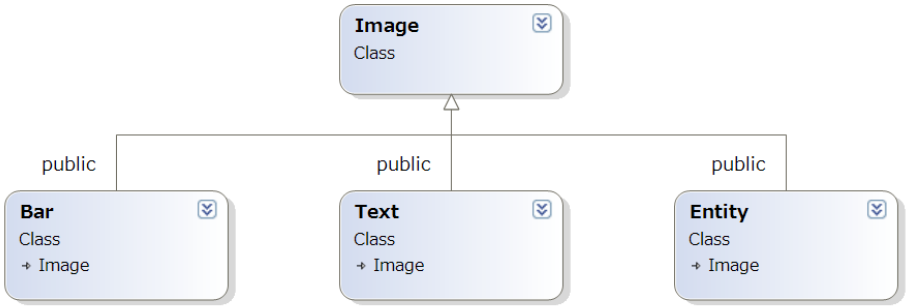
* Picturesフォルダ
* Audioフォルダ

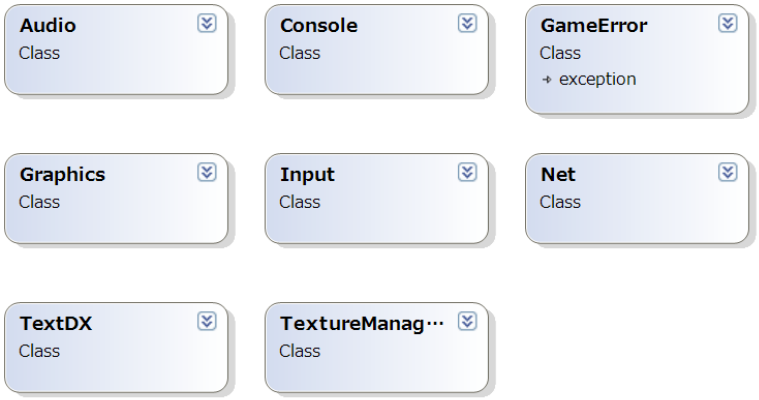
* .slnファイルを
  開いてVC起動

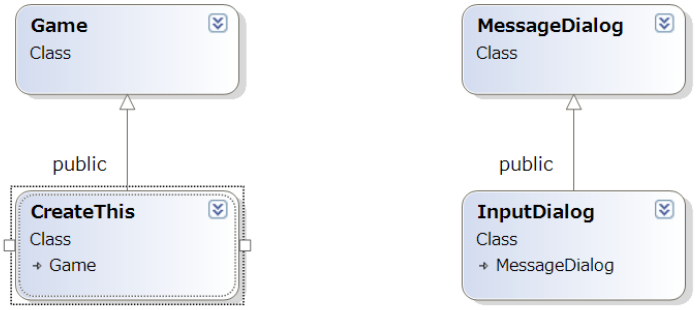


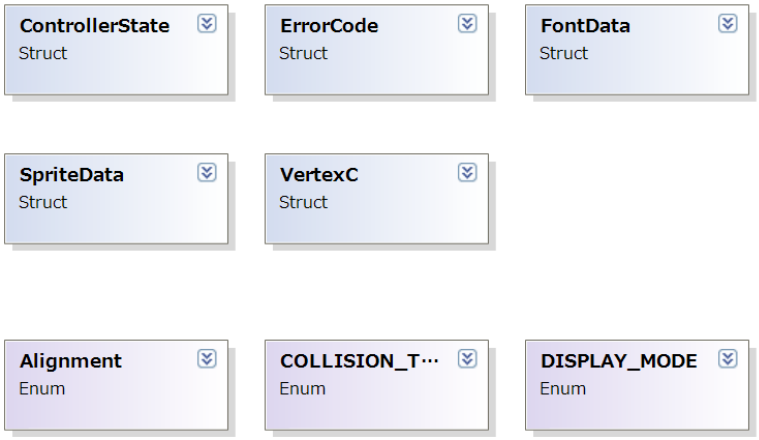
どのクラスがアプリか？

* ソリューションエクスプローラで「クラスダイアグラム」表示









* createThisClassがアプリ

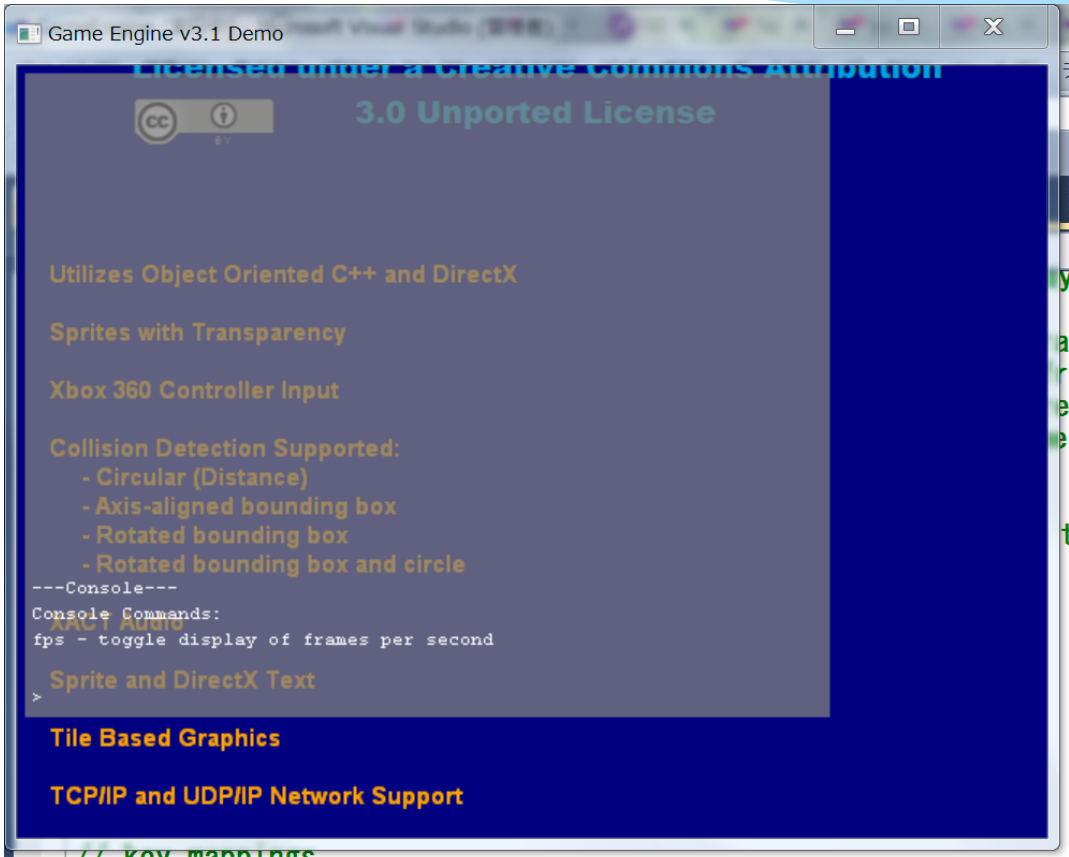
Constants.hを見ておく

* エンジンの内部に手を入れる必要はない
* 画像やサウンドバンク、タイトルなどの指定を確認

```
//=============================================================================
//                  Constants
//=============================================================================

// window
const char CLASS_NAME[] = "createThisClass";
const char GAME_TITLE[] = "Game Engine v3.1 Demo";
const bool FULLSCREEN = false;              // windowed or fullscreen
const UINT GAME_WIDTH =  640;               // width of game in pixels
const UINT GAME_HEIGHT = 480;               // height of game in pixels

// game
const bool VSYNC = false;                   // true locks display to vertical sync rate
const double PI = 3.14159265;
const float FRAME_RATE = 240.0f;                // the target frame rate (frames/sec)
const float MIN_FRAME_RATE = 10.0f;             // the minimum frame rate
const float MIN_FRAME_TIME = 1.0f/FRAME_RATE;   // minimum desired time for 1 frame
const float MAX_FRAME_TIME = 1.0f/MIN_FRAME_RATE; // maximum time used in calculations

// graphic images
const char MENU_IMAGE[] =     "pictures¥¥menu.png";      // menu texture

// audio files required by audio.cpp
// WAVE_BANK must be location of .xwb file.
const char WAVE_BANK[]  = "";
// SOUND_BANK must be location of .xsb file.
const char SOUND_BANK[] = "";

// audio cues

// key mappings
// In this game simple constants are used for key mappings. If variables were used
// it would be possible to save and restore key mappings from a data file.
const UCHAR CONSOLE_KEY  = '`';         // ` key
const UCHAR ESC_KEY      = VK_ESCAPE;   // escape key
const UCHAR ALT_KEY      = VK_MENU;     // Alt key
const UCHAR ENTER_KEY    = VK_RETURN;   // Enter key
```

コマンドモードなんかもある

* [`] @+Shift
* help, fps表示




```
// key mappings
// In this game simple constants are used for key mappings. If variables were
// it would be possible to save and restore key mappings from a data file.
const UCHAR CONSOLE_KEY  = '`';      // ` key
const UCHAR ESC_KEY      = VK_ESCAPE;  // escape key
const UCHAR ALT_KEY      = VK_MENU;    // Alt key
const UCHAR ENTER_KEY    = VK_RETURN;  // Enter key
```

やりたいことをもんやり列挙

ShiraiLab

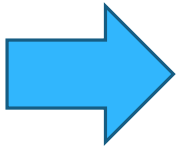
* 自由にスプライト表示したい
* 音を鳴らしたい
* スクロールとかシューティングゲームっぽいことをしたい
* あまり複雑なことをしたくない

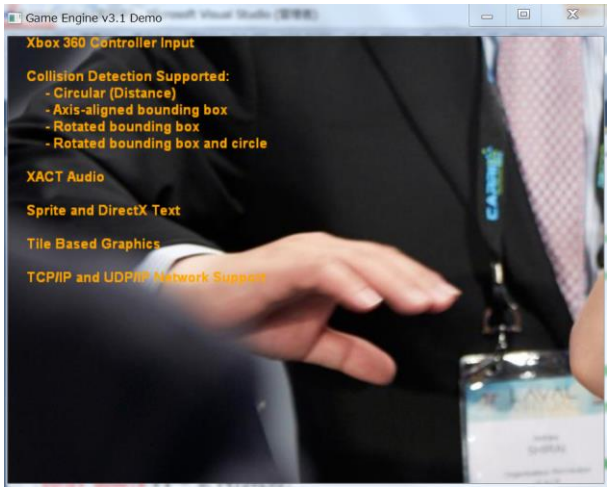
まずはJPEGファイルを読込表示

ShiraiLab

* Picturesフォルダに希望のJPEG(PNGでも可)ファイルを置く
* タイトル画像の代わりにJPEG表示Constants.h

```
// graphic images
//const char MENU_IMAGE[] =    "pictures¥¥menu.png";    // menu texture
const char MENU_IMAGE[] =    "pictures¥¥LV2015akiL.jpg";
```



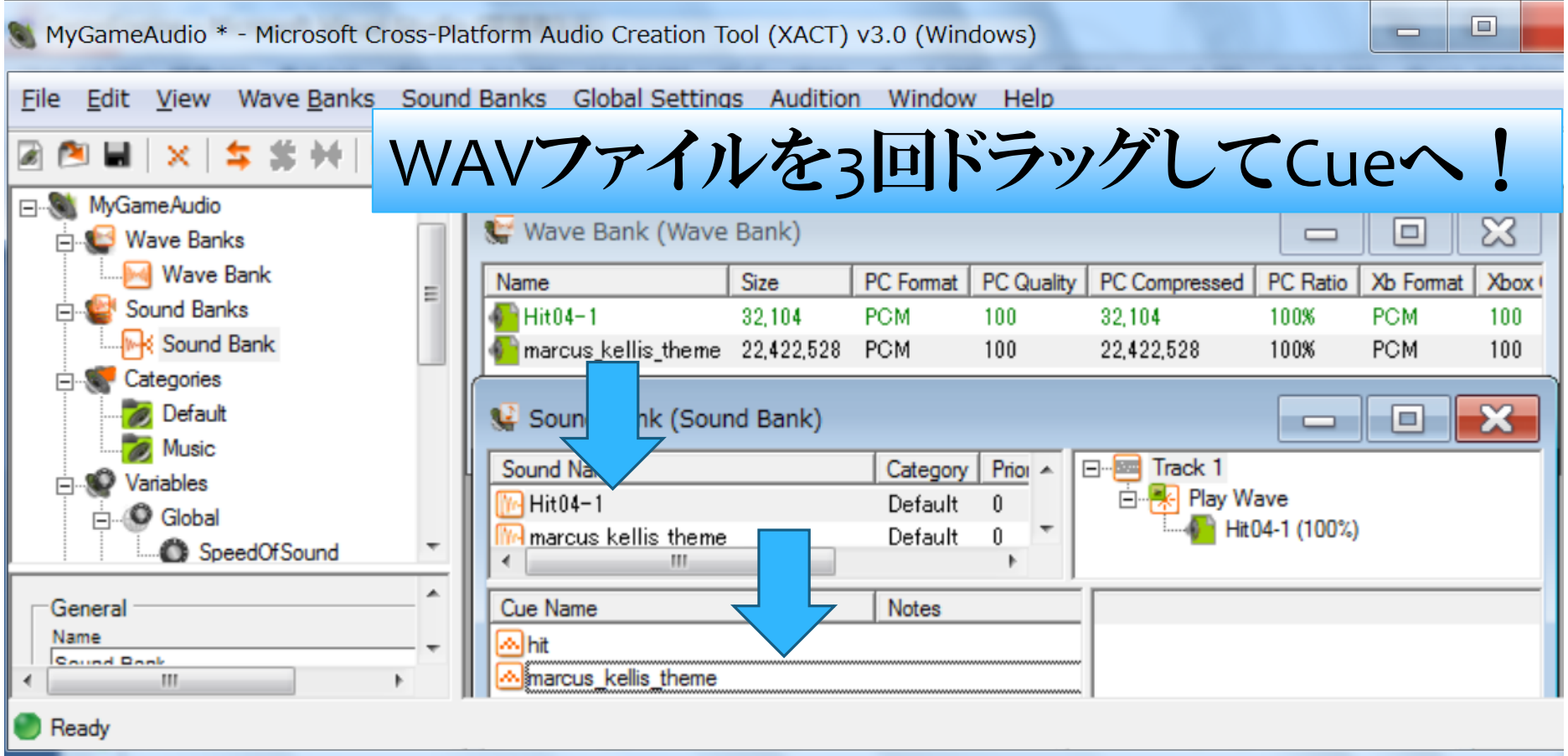


BGM等の取り込み

* Audacityを使ってMP3ファイルからWAVに変換
* XACTを使って新しいバンクを作成
* ソースコード(constants.h)に追記

```
// audio files required by audio.cpp
// WAVE_BANK must be location of .xwb file.
const char WAVE_BANK[]  = "audio¥¥Win¥¥Wave Bank.xwb";
// SOUND_BANK must be location of .xsb file.
const char SOUND_BANK[] = "audio¥¥Win¥¥Sound Bank.xsb";

// audio cues
const char BGM[] = "marcus_kellis_theme"; //marcus_kellis_theme
```

* なおここでは“marcus_kellis_theme” を利用
  Dance Like There's Marcus Kellis In Your Pants Themesong, Future Boy




```
// audio files required by audio.cpp
// WAVE_BANK must be location of .xwb file.
const char WAVE_BANK[]  = "audio¥¥Win¥¥Wave Bank.xwb";
// SOUND_BANK must be location of .xsb file.
const char SOUND_BANK[] = "audio¥¥Win¥¥Sound Bank.xsb";

// audio cues
const char HIT[]    = "hit";
const char BGM[]    = "marcus_kellis_theme"; //サウンドバンクのCuesの名前
```

せっかくなのでBGM鳴らす

* CreateThis::initialize()
* **最後のほうに**

audio->playCue(BGM); **を追加**

```
    // initialize DirectX font
    // 18 pixel high Arial
    if(dxFont->initialize(graphics, 18, true, false, "Ar
        throw(GameError(gameErrorNS::FATAL_ERROR, "Erro

    menu.setDegrees(300);
    menu.setScale(0.002861f);

    message = "¥n¥n¥nUtilizes Object Oriented C++ and D
    message += "Sprites with Transparency¥n¥n";
    message += "Xbox 360 Controller Input¥n¥n";
    message += "Collision Detection Supported:¥n";
    message += "      - Circular (Distance)¥n";
    message += "      - Axis-aligned bounding box¥n";
    message += "      - Rotated bounding box¥n";
    message += "      - Rotated bounding box and circle¥r
    message += "XACT Audio¥n¥n";
    message += "Sprite and DirectX Text¥n¥n";
    message += "Tile Based Graphics¥n¥n";
    message += "TCP/IP and UDP/IP Network Support¥n¥n";
    messageY = GAME_HEIGHT;

    audio->playCue(BGM);

    return;
}
```

* **正しくロードできていれば、起動時に再生されます**

せっかくなのでループで鳴らす

ShiraiLab

* CreateThis::update()

```
//=============================================================
// Update all game items
//=============================================================
void CreateThis::update()
{
    if(menu.getDegrees() > 0)
    {
        menu.setDegrees(menu.getDegrees() - frameTime * 120);
        menu.setScale(menu.getScale() + frameTime * 0.4f);
    }
    else if(messageY > -500)
    {
        menu.setDegrees(0);
        menu.setY(menu.getY() - frameTime * 50);
        messageY -= frameTime * 50;
    }
    else    // start over
    {
        menu.setDegrees(300);
        menu.setScale(0.002861f);
        menu.setY(0);
        messageY = GAME_HEIGHT;
        audio->stopCue(BGM);
        audio->playCue(BGM);
    }
}
```

* 正しくロードできていれば、ループごとに再生されます

せっかくなのでSEを足す

* Music is VFR　CCライセンス
  http://musicisvfr.com/free/se/hit01.html
* AudacityでMP3からWAVに変換
* WAVEバンク、サウンドバンクに追加してビルドしなおすこと

“へぇボタン”のSEも足したいが

ShiraiLab

* ニコニ・コモンズ
  * http://commons.nicovideo.jp/material/nc69286
* ニコニコ動画に公開するなら可能
* この素材の場合は
  別のサイトで再配布する権利はない

キーボードを押すとSEが鳴る

ShiraiLab

* Constants.hに定義を追加

```
// audio cues
const char BGM[] = "marcus_kellis_theme"; //marcus_kellis_theme
const char HIT[] = "Hit03-1"; // from http://musicisvfr.com/free/se/hit01.html

// key mappings
// In this game simple constants are used for key mappings. If variables were used
// it would be possible to save and restore key mappings from a data file.
const UCHAR CONSOLE_KEY  = '`';          // ` key
const UCHAR ESC_KEY      = VK_ESCAPE;    // escape key
const UCHAR ALT_KEY      = VK_MENU;      // Alt key
const UCHAR ENTER_KEY    = VK_RETURN;    // Enter key
//add by aki
const UCHAR SHIP_LEFT_KEY    = VK_LEFT;     // left arrow
const UCHAR SHIP_RIGHT_KEY   = VK_RIGHT;    // right arrow
const UCHAR SHIP_UP_KEY      = VK_UP;       // up arrow
const UCHAR SHIP_DOWN_KEY    = VK_DOWN;     // down arrow
const UCHAR SHIP_SPACE_KEY   = VK_SPACE;    // スペースキー
```

* SpaceキーでHIT音を鳴らす

```
void CreateThis::update()
{
    if(input->isKeyDown(SHIP_SPACE_KEY)) {
        audio->playCue(HIT);
    }
```

へぇボタン的なものを作る

ShiraiLab

* キーが押されたら2つ目のスプライトを表示する
* CreateThis.Classhに
  Texture Manager
  　　heeTexture;
  Image hee;
  を追加
* 「へぇ画像」を用意
* Constants.hにHEE_IMAGE追加

```
//=============================================
// This class is the core of the game
//=============================================
class CreateThis : public Game
{
private:
    // game items
    TextureManager menuTexture; // textures
    Image   menu;               // menu image
    //2つ目のスプライトオブジェクトを用意
    TextureManager heeTexture; // textures
    Image   hee;               // menu image


    TextDX  *dxFont;            // DirectX font
    std::string  message;
    float messageY;
```

```
// graphic images
//const char MENU_IMAGE[] =     "pictures¥¥menu.png";      // menu texture
const char MENU_IMAGE[] =    "pictures¥¥bg.jpg"; // My Background Texture
const char HEE_IMAGE[]  =    "pictures¥¥hee.jpg";// hee texture
```

インタラクションを追加する

ShiraiLab

* CreateThis::initialize(HWND hwnd)にロード、初期化を追加
* Update()に
  キーボードイベントを
  追加する
* Render()に
  追加draw()を追加

```
// へぇ画像のロード、上の4行を参考にオブジェクト名を変えただけ
if (!heeTexture.initialize(graphics,HEE_IMAGE))
    throw(GameError(gameErrorNS::FATAL_ERROR, "Error initializing hee textu
// menu image
if (!hee.initialize(graphics,0,0,0,&heeTexture))
    throw(GameError(gameErrorNS::FATAL_ERROR, "Error initializing hee"));
```

```
//=============================================================
// Update all game items
//=============================================================
void CreateThis::update()
{
    //へぇキー（スペース）が押されたら、音を鳴らして画像を表示
    if (input->isKeyDown(HEE_KEY)) {
        audio->playCue(HIT);
        hee.setX(GAME_WIDTH/2); //ここはまかせます
        hee.setY(GAME_HEIGHT/2); //ここもまかせます
        hee.setScale(0.8);// 調整に便利
        hee.setVisible(true); //表示
    } else {
        hee.setVisible(false); //非表示
    }
```

```
//=============================
// Render game items
//=============================
void CreateThis::render()
{
    graphics->spriteBegin();

    menu.draw();
    hee.draw(); //これがないとへぇ画像は表示されない
    dxFont->setFontColor(graphicsNS::ORANGE);
    dxFont->print(message,20,(int)messageY);

    graphics->spriteEnd();                  // end dra
}
```

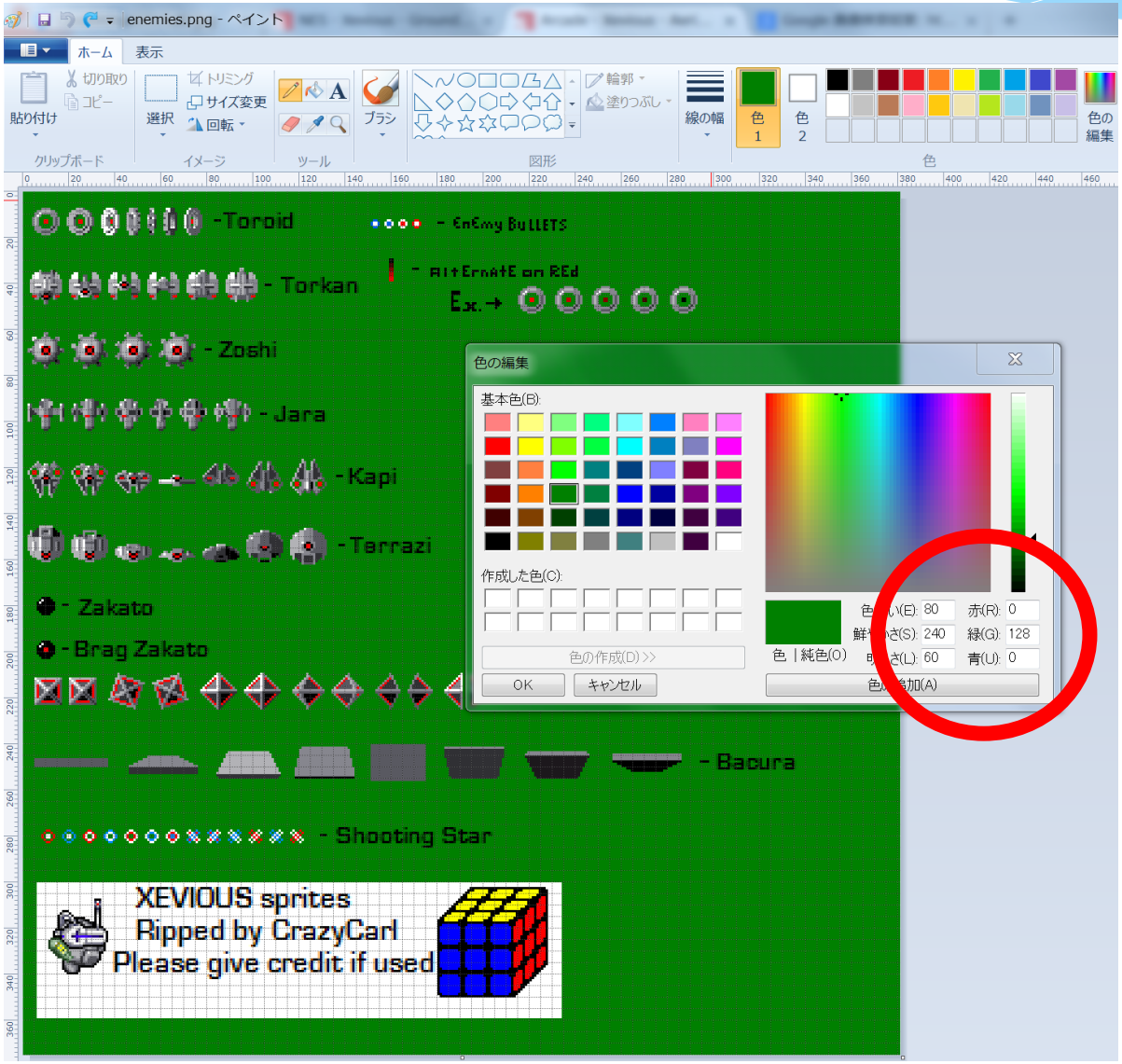
ここで再度,仕様について考える

ShiraiLab

* せめてゲームっぽいスプライト表示したい
* 音を鳴らせるようになったのでオトゲーにしたい
* スクロールとかシューティングゲームっぽいことをしたい
* あまり複雑なことをしたくない
* せめてスコアファイルぐらい読めないとゲームにならないかも
* 参考：Processingをつかってmp3ファイルからスコア作成
  * Example:minim: FrequencyEnergy
  * I/O:SaveFile2
  * mp3ファイルを再生しながら、Kick, Snare, Hatの打楽器データを生成
  * CSVファイルに書き出し → テキストエディタやExcelで修正
* 以下来週の予告！

スプライトファイルを頂いてくる

* http://www.spriters-resource.com/arcade/xevious/sheet/42387/



Thanks CrazyCarl,

抜き色は
ペイントのスポイトで
調べることができる
→ (0,128,0)

SpaceshipControlを参考に…

* createThisClass.hにメンバ追加

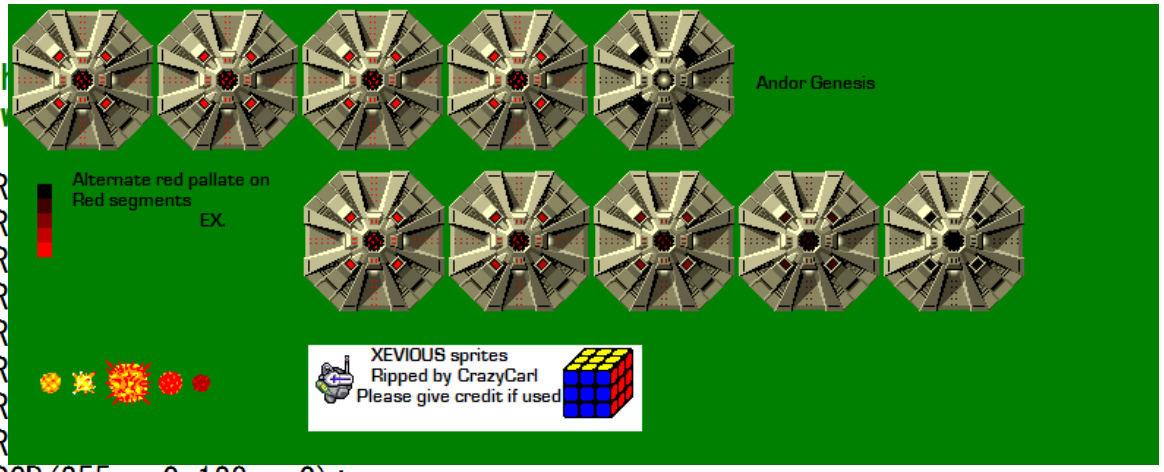
```
TextureManager shipTexture;    // ship texture
Image   ship;                  // ship image
```

* CreateThis::render()に追加

```
//=============================================================================
// Render game items
//=============================================================================
void CreateThis::render()
{
    graphics->spriteBegin();                // begin drawing sprites

    menu.draw();
    ship.draw();                            // add by Aki
    dxFont->setFontColor(graphicsNS::ORANGE);
    dxFont->print(message,20,(int)messageY);

    graphics->spriteEnd();                  // end drawing sprites
```

抜き色指定

```
namespace graphicsNS
{
    // Some common colors
    // ARGB numbers range from 0 through
    // A = Alpha channel (transparency
    // R = Red, G = Green, B = Blue
    const COLOR_ARGB ORANGE  = D3DCOLOR_AR
    const COLOR_ARGB BROWN   = D3DCOLOR_AR
    const COLOR_ARGB LTGRAY  = D3DCOLOR_AR
    const COLOR_ARGB GRAY    = D3DCOLOR_AR
    const COLOR_ARGB OLIVE   = D3DCOLOR_AR
    const COLOR_ARGB PURPLE  = D3DCOLOR_AR
    const COLOR_ARGB MAROON  = D3DCOLOR_AR
    const COLOR_ARGB TEAL    = D3DCOLOR_AR
    const COLOR_ARGB GREEN   = D3DCOLOR_ARGB(255,   0, 128,   0);
    const COLOR_ARGB NAVY    = D3DCOLOR_ARGB(255,   0,   0, 128);
    const COLOR_ARGB WHITE   = D3DCOLOR_ARGB(255, 255, 255, 255);
    const COLOR_ARGB YELLOW  = D3DCOLOR_ARGB(255, 255, 255,   0);
    const COLOR_ARGB MAGENTA = D3DCOLOR_ARGB(255, 255,   0, 255);
    const COLOR_ARGB RED     = D3DCOLOR_ARGB(255, 255,   0,   0);
    const COLOR_ARGB CYAN    = D3DCOLOR_ARGB(255,   0, 255, 255);
    const COLOR_ARGB LIME    = D3DCOLOR_ARGB(255,   0, 255,   0);
    const COLOR_ARGB BLUE    = D3DCOLOR_ARGB(255,   0,   0, 255);
    const COLOR_ARGB BLACK   = D3DCOLOR_ARGB(255,   0,   0,   0);
    const COLOR_ARGB FILTER  = D3DCOLOR_ARGB(  0,   0,   0,   0); // use to specify drawing with colorFilter
    const COLOR_ARGB ALPHA25 = D3DCOLOR_ARGB( 64, 255, 255, 255); // AND with color to get 25% alpha
    const COLOR_ARGB ALPHA50 = D3DCOLOR_ARGB(128, 255, 255, 255); // AND with color to get 50% alpha
    const COLOR_ARGB BACK_COLOR = NAVY;                           // background color of game
//    const COLOR_ARGB TRANSCOLOR = MAGENTA;                          // transparent color
    const COLOR_ARGB TRANSCOLOR = D3DCOLOR_ARGB(255,0, 128,   0); // for XEVIOUS sprites (ARGB)

    enum DISPLAY_MODE{TOGGLE, FULLSCREEN, WINDOW};
}
```

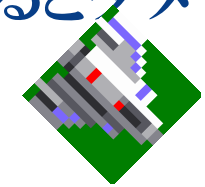
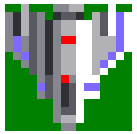
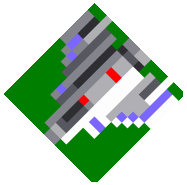
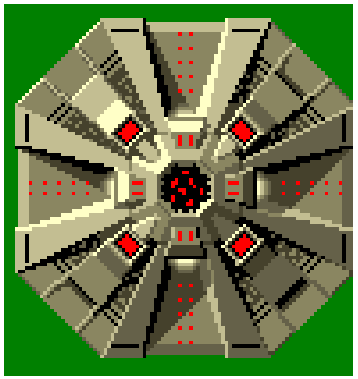


Graphics.hで抜き色指定を
MANGENTA から
ARGB(255,0,128,0) に変更

そろそろ何がやりたいか決める

ShiraiLab

* レトロゲーム好きが楽しめそうな音ゲー(リズムゲーム)
* 自分がボスになって敵機を蹴散らすゲーム
* 音楽に合わせて敵機が飛んでくる
* ←, ↑, → キーで3方向を打つ(タイミングだけ)
* タイミングミスるとダメージ
* 曲の
  終わりまで
  耐えるだけのゲーム

とにかく応用！自分を前進させる

ShiraiLab

* 前回と同じ提出形式
  * 動画：横位置, 学籍番号+11.mp4
  * 資料ZIP：PowerPoint仕様／作業メモ／スクリーンショット, ソース
  * Dropboxがお勧めです（更新できるという意味でも…）
* がんばれる人は・・・
  * ファイルのロード
  * コリジョン
  * 得点とハイスコア
  * シーン遷移
  * ネットワーク対応・・・！？

* 資料・ソースはこちら
  https://github.com/kaitas/ShiraiLabOpen/tree/master/GamePro/GameEngine