DXライブラリ講座

Presented by やる気のない副会長

今日の内容

- DXライブラリの解説と設定
- 実装部とヘッダファイルの扱い
- ゲーム動作順番の構成
- 練習課題及び自主作成

※開発環境はVisualC++2008または2010

そもそもDXライブラリとは？

DirectXを使ったソフト開発を行うときにいろいろめんどくさいことがあるけど、それをまとめてくれて超楽にプログラムを書けるようになるC++用（実質C言語）のゲームライブラリ。
ゆとり乙。

今回は主にグラフィックの描画、音の再生に関してお世話になります。本気出せばこれでネトゲも作れます。

DXライブラリの設定

http://homepage2.nifty.com/natupaji/DxLib/
にアクセス

DXライブラリでググって一番上に出てくるページ

※後々のためにブックマークしておくといいよ

DXライブラリVisualC++用をダウンロード
保存場所はどこでもおk（自分のわかる場所で）

DXライブラリの設定

- VC2010の人　→　めんどい
- VC2008の人　→　もっとめんどい

『DXライブラリの使い方』を参照

実装部とヘッダファイル

- 今回のプログラムは実装部(.cpp)とヘッダファイル(.h)を分けて作っています。

VC2010の人→実装部1、ヘッダ1

VC2008の人→実装部1、ヘッダ2

実装部とヘッダファイル

- main.cpp
  メインループでの動作を記述
  VC2010の人は各種関数も記述

- function.h(VC2008の人のみ)
  メインループで呼び出される関数

- material.h
  グローバル変数と構造体の宣言

インクルードガード

- 1つのヘッダファイルを複数のファイルでインクルードするとコンパイルエラーが発生してしまう。今回の場合はVC2008用のmaterial.hがそれに当たる。インクルードガードを行うことでコンパイルエラーを防ぐ

  VCの場合重複してインクルードされるヘッダファイルの頭に#pragma onceと記述することでインクルードガードを行う

今回のゲームプログラムの構成

1. アプリケーションに関する設定
2. 各種データの初期化

ーメインループー

1. 初期化フェイズ
2. 制御フェイズ
3. 描画フェイズ

ーメインループー

アプリケーションに関する設定

- SetMainWindowText(“タイトル”)
  タイトルの設定
- ChangeWindowMode(bool)
  フルスクリーンの設定
- SetDrawScreen(DX_SCREEN_***)
  描画対象の設定（初期設定は表）

各種データの初期化

今回使用する構造体

- Character_Condition構造体

  キャラクターに関する状態を扱う

- Bullet_Condition構造体

  弾に関する状態を扱う

- Stage_Condition構造体

  ステージに関する状態を扱う（今回はステージの画像のみ）

DXライブラリ豆知識

- ハンドル

  主にLoadGraph関数やLoadSoundMem関数などで返ってくるデータを入れておく変数のこと。これ自体はただのint型の変数なのだがここではグラフィックハンドル、サウンドハンドルなどと呼ぶ。意識付けのため？

各種データの初期化

- LoadGraph(“ファイル名”)
- LoadSoundMem(“ファイル名”)

  各種データのロード

※これらのデータはハンドルに入れて使います

メインループ

- 初期化フェイズ

  今回は描画した画面のクリアのみを行う

- 制御フェイズ

  移動、弾発射などの入力と自機や弾の状態操作

  当たり判定の確認などを行う

- 描画フェイズ

  制御フェイズで座標が確定したオブジェクトの描画を行う

メインループ

- 継続条件式の中身に注目

ProcessMessage() == 0
これは正常にプログラムが稼動している場合0が、
ウィンドウが閉じられた場合には-1が返ってくる。

もし継続条件式の中にこれを書かないでウィンドウを消すとどうなる？

→ウィンドウは閉じるがループから抜けられないのでプロセスには残り続ける。タスクマネージャーで確認できるからやりたい人はどうぞ

初期化フェイズの説明の前に

- DXライブラリの描画のイメージ

紙の裏側にオブジェクトを書く

↓

紙をひっくり返して画面として表示

↓

ひっくり返した紙の裏側を消す

↓

紙の裏側にオブジェクt(ry

初期化フェイズ

- ClsDrawScreen()
  画面に描画したものを全消去

※今回描画対象は裏側に設定してあるので表向きでは何も起きていないように見えます。

→SetDrawScreen(DX_SCREEN_BACK)

制御フェイズ

ここでやる大切なことは主に三つ

1. キャラの移動
2. 弾の発射
3. あたり判定のチェック

制御フェイズ

別にここでやらなくてもいいもの

- ステージ音の再生(今回の場合は)
- リタイアボタンのチェック

キャラの移動

- 上下左右のボタンに対応して移動するようにしたい

関数CheckHitKey(KEY_INPUT_***)

→対応するキーが入力されている場合1が返ってくる

キャラの移動

- 移動するときははみ出ないようにしないといけない

→移動範囲の設定

(今回は画面サイズに合わせています)

- 移動する量

対応するキーの方向に2ピクセル動かしています。もしアイテムとかで移動速度を早くするのであれば、変動量は構造体で宣言してもいいと思う。

弾の発射

- 弾の構造体の宣言に注目

Bullet_Condition　My_Bullet[256]

このように配列で宣言しています。

弾はひとつひとつ状態が異なるのでこのように配列で宣言するのが利口でしょう。

弾の発射

- 弾構造体の中で特に重要なもの

『bool　frag_Alive』

弾が現在使われているのか否かの状態を保持します

bool型　→　true　or　false

弾発射

- 弾発射の流れ

弾の移動と弾の画面外進入の判定

↓

自分の発射待ち時間を1減らす

↓

弾を撃てる状態でスペースキーが押されたらAliveがfalseの弾に各種情報を持たせて発射しAliveをtrueにする。

当たり判定

- 判定の取り方はいろいろあるが、今回は発射した弾の中心座標が敵の画像に触れたらあたりとみなす。

画像に触れる

→画像のサイズを求める必要がある

関数GetGraphSize(Ghandle,*sizex,*sizey)

描画フェイズ

関数DrawGraph(X,Y,Ghandle,透過指定)

指定したX,Y座標が画像の左上の頂点になるように描画される

※画像の中心座標ではないので注意

描画はどんどん上書きされる

→隠れたらまずいものは最後に描く

課題

- このプログラムでは敵が弾を撃ってきません。

↓

敵に弾を撃たせよう！
敵の弾が当たったらゲームオーバー！