● 授業のポイント

① ゲームプログラムの構造（入力処理、更新処理、描画処理の繰り返し）について学習します。
② シーン切り替え処理について学習します。
③ ゲームの雛形を作成します。

※ シーンの切り替え処理をひと通り作成しますが、現状は仮バージョンです。あとの課題でブラッシュアップします。

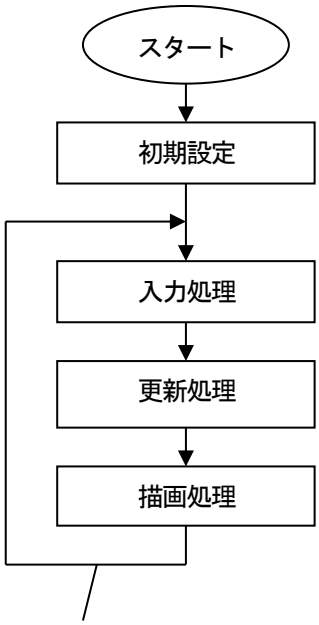
● CPPGame01D「ゲームループを作ろう！」

1．GL ライブラリについて

ゲーム用ライブラリ（GL ライブラリ）を使ってゲームプログラムを作成していきます。フォルダ「GL」にあるヘッダファイル「GL.h」をインクルードすれば使うことができます。ただしライブラリの関数を使うのは最初のころのみで、ゲームが出来上がるにつれ意識しなくてもいいようになってきます。

2．ゲームプログラムの構造

ゲームプログラムの構造、および各処理の概要は以下のようになっています。



・初期設定　ウインドウの作成やDirectX の初期設定など
・入力処理　キーボード、マウス、ジョイパッドなどの状態を調べる
・更新処理　ゲームの情報（画像の座標など）を更新する
・描画処理　画像を描画する

ゲームループ（またはメインループ）。60 分の 1 秒（約 16.7 ミリ秒）で 1 周する。ループ 1 回分を 1 フレームと呼ぶ。

キーボードやマウス、ジョイスティックなどの状態を調べ（入力処理）、その情報に基づいてゲーム情報（例えばプレイヤーの座標など）を更新し（更新処理）、更新された情報に基づいて画像を描画する（描画処理）という流れです。これを繰り返すことによってゲームが進行します。少しずつ異なる画像を高速で表示して動いているように見せる原理はアニメーションと同じですが、ゲームの場合は毎フレーム、入力に基づいて画像を更新して描画・表示していきます。

「入力処理」「更新処理」「描画処理」の繰り返し部分をゲームループ（またはメインループ）と呼びます。家庭用ゲーム機ではテレビの表示タイミング（リフレッシュレート）と同じ 60 分の 1 秒周期となっています。パソコンではリフレッシュレートの設定によって変わりますが、授業では 60 分の 1 秒周期ということで進めていきます。

３．入力処理

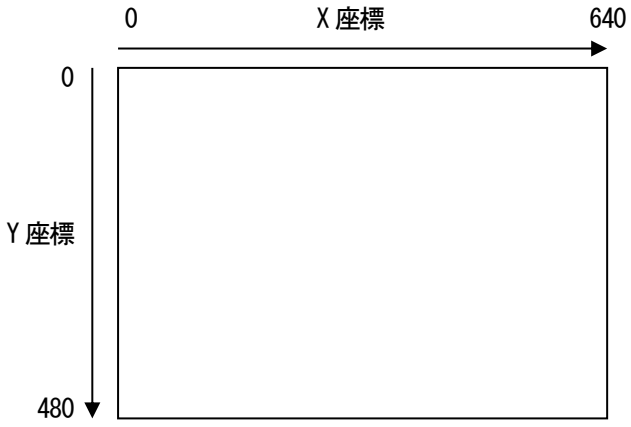
入力処理はライブラリにまかせます。ゲームプログラマはライブラリが設定した情報を使います。

４．更新処理

入力情報にもとづいて、画像の座標といったゲームの情報を更新します。ただし、ゲームループの中の処理なので、ループ 1 回分（1 フレーム、16.7 ミリ秒）での変化量を記述します。例えば、X 座標の値を 2 減らすと記述すると、ループで何度も呼び出されるので、時間とともに値が減っていく（動く）ことになります。

５．描画処理

更新処理で更新されたゲーム情報にもとづいて、画像の描画を行います。GL;;BeginScene 関数と GL::EndScene 関数の間に描画処理を記述します。文字列の座標系はスクリーン座標（右図）を使います。解像度は 640×480 です。Y 座標は下方向がプラスとなります。




```
GL::BeginScene();
      ～
（ここで描画）
      ～
GL::EndScene();
```

GL::EndScene 関数を呼び出した段階で描画した画面をモニターに表示します。このとき垂直同期信号（VSync、60Hz）を待って表示するため、ゲームループは 60 分の 1 秒でループします。

ちなみに描画用バッファには前回描画した画像が残っています。よって描画処理では通常、GL::BeginScene()のあとに画面クリア処理（GL::ClearScene()）を行います。

６．その他

コンソールアプリケーションのエントリポイントは main 関数ですが、ウインドウズアプリケーションのエントリポイントは WinMain 関数となります。授業ではウインドウズアプリケーションとしてゲームプログラムを作成するので、ゲームプログラムのエントリポイントも WinMain 関数となっています。

● CPPGame01B「シーンを切り替えよう！」

１．シーン切り替え

前回課題で pScene に設定されたシーンが実行されるようになっています。よって pScene に設定するシーンを変更できるようにすればタイトルやゲームの切り替えができます。ただしここで大切なのは更新処理と描画処理はセットで実行しないといけない点です。更新処理の中で pScene を変更してしまうと、直後の描画処理は変更された pScene の描画処理が呼び出されます。今回の課題では特に問題は発生しませんが、場合によっては初期化されていない画像を描画しようとしてシステムがクラッシュすることもあります。

そこで今回の課題では更新処理の中で直接 pScene を変更するのではなく、次に実行するシーン（pNext）を設定するだけにしています。これによって直後の描画処理は更新処理と同じシーンのものが実行されます。シーンの切り替えは次のループの更新処理を実行する直前に行われます（pNext が設定されていたら pScene を変更する）。

２．コンストラクタと初期化処理

SceneGame の初期化（メンバ変数の初期設定など）はコンストラクタで行っています。しかしシーンの切り替えが何度も発生すると（ゲームでは日常茶飯事）、2 回目以降は初期設定されないため正常に動作しなくなります。よってコンストラクタとは別に初期化処理（init 関数）を準備して、シーン切り替えが発生したタイミングで呼び出すようにします。

シーンの実体を静的に宣言するのではなく、シーン切り替えが発生するたびに動的に new・delete すれば毎回コンストラクタが呼び出されますが、この場合 delete されたシーンの情報はすべて失われてしまいます。ゲームの場合、失われてもいい情報（例えばプレイヤーのスコアなど）だけでなく失われてはいけない情報（例えばハイスコアなど）もあるため、何らかの対処が必要になります。静的に実体宣言している場合は、コンストラクタ（起動時）でハイスコアを初期化して、init 関数（シーン切り換え時）でプレイヤースコアを初期化するというように使い分けができます。

というよりも不特定多数の敵キャラや容量が大きいアセットデータなどで動的メモリ確保するのはわかるが、一つしか存在しない（しかもプログラムコードの容量なんてしれてる）シーンを動的に確保しても・・・。