● CPPGame01D「ゲームループを作ろう！」（課題データをコピーすること）

ゲームループを作成し、文字列を描画する処理を追加せよ。

文字列の仕様

No.	サイズ	座標	描画する文字列	その他の指定
1	L	(320, 160)	″GAME だよ″	色：白、書式：センタリング
2	M	初期位置：(640, 300) X 座標は毎回 2 減らす。	″文字列が動きます！うりゃあああああ！！″	色：黄、書式：左詰め

手順１．ゲームループの作成（ファイル「WinMain.cpp」）

① ファイル「WinMain.cpp」に**リスト１**を入力する。
② ビルド・実行し、ゲーム用ウインドウが表示されるのを確認する（ウインドウ右上の×ボタン、もしくは[ESC]キーで終了できる）。

手順２．文字列 No.1 の描画（ファイル「WinMain.cpp」）

① 描画処理に文字列 No.1 の描画処理を作成する。
② ビルド・実行し、画面に「GAME だよ」と表示されるのを確認する。

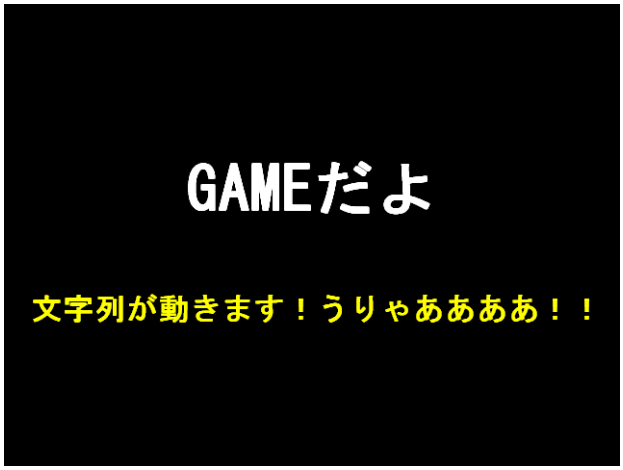
手順３．文字列 No.2 の描画（ファイル「WinMain.cpp」）

① 文字列 No.2 は、X 座標が変化するので、X 座標を表す変数をあらかじめ宣言しておく。
② 更新処理に X 座標を計算する処理、描画処理に文字列描画処理を作成する。
③ ビルド・実行し、画面に文字列 No.2 が表示&移動するのを確認する（ただし表示はおかしい）。
④ 文字列の描画がおかしくなるので、画面クリアを追加する。
⑤ ビルド・実行し、文字列 No.2 の表示が正常になったのを確認する。

リスト１：ゲームループの作成（ファイル「WinMain.cpp」）

```
// インクルード
#include ".\GL\GL.h"
// 実体宣言
ここで必要な変数を宣言する

// WinMain関数（エントリポイント）
int APIENTRY WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow )
{
    // 初期設定
    GL::Init(_T("ゲームプログラミング"), 640, 480, false);

    // メインループ
    while (GL::GameLoop()) {
        // 入力処理
        GL::Input();

        // 更新処理
        ここに更新処理（変数の値を変更する処理）を書く

        // 描画処理
        GL::BeginScene();               // 描画開始
        ここに描画処理を書く
        GL::EndScene();                 // 描画終了＆表示
    }

    // 終了処理
    GL::End();
    return 0;
}
```

課題完成時の画面



● GL 仕様書

ヘッダファイル

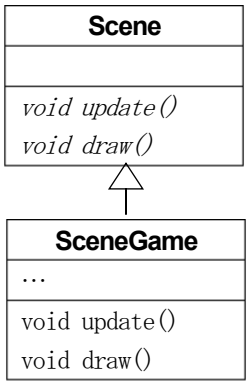
```
#include ″./GL/GL.h″
```

ゲームループ・描画関連の関数（使い方は CPPGame01D のリスト 1 を参照）

関数	説明
GL::Init(_T(″文字列″), 横幅, 高さ, モード)	ライブラリ（Windows、DirectX）の初期設定を行う。 ″文字列″は、タイトルバーに表示される文字列。 横幅、高さは、スクリーンの解像度。 モードは true：フルスクリーン、false：ウインドウモード。
GL::GameLoop()	ゲームループを終了させるかどうかのチェックを行う。 （実態は Windows のメッセージ処理）
GL::End()	ライブラリの終了処理を行う。
GL::Input()	入力用変数を更新する（更新する変数の説明は次回）。
GL::BeginScene()	描画処理を開始する。
GL::EndScene()	描画処理を終了し、画面を表示する。
GL::ClearScene()	画面をクリアする。
GL::DrawStringL(X 座標, Y 座標, 文字列, 色, 書式)	L サイズ（64 ドット）の文字列を描画する。色と書式は省略可。 色（R:G:B=8:8:8 で指定、以下のラベルも指定可能、省略時は白） COLOR_BLACK：黒　COLOR_GRAY：灰色　COLOR_WHITE：白 COLOR_RED：赤　COLOR_GREEN：緑　COLOR_BLUE：青 COLOR_YELLOW：黄色　COLOR_MAGENTA：紫　COLOR_CYAN：水色 書式（省略時は左詰め） false：左詰め true：センタリング なお、これ以外にも以下の関数がある。 GL::DrawStringS 関数：S サイズ（16 ドット）文字列を描画 GL::DrawStringM 関数：M サイズ（32 ドット）文字列を描画

● CPPGame01C「Scene クラスを作ろう！」（前回の続きで作成）

ゲームのシーンを表すScene クラス（抽象クラス）およびその派生クラスとしてSceneGame クラスを作成し、ゲームループ内にある更新処理と描画処理をSceneGame に移せ。なお必要に応じてメンバ変数を追加すること。またゲームループ内の処理はScene クラスへのポインタを使って行うこと。



SceneGame クラスの仕様（新規ファイル「SceneGame.h」「SceneGame.cpp」）

メンバ関数	処理
コンストラクタ	メンバ変数の初期設定をする
void update()	ゲームループ内の更新処理を持ってくる
void draw()	ゲームループ内の描画処理（GL::BeginScene()とGL::EndScene()に挟まれた部分）を持ってくる

手順１．Scene クラスの定義（新規ファイル「Scene.h」）

③ ヘッダファイル「Scene.h」を新規作成し、Scene クラスを定義する。

手順２．SceneGame クラスの作成（新規ファイル「SceneGame.h」「SceneGame.cpp」）

① ヘッダファイル「SceneGame.h」を新規作成し、SceneGame クラスを定義する。
② ソースファイル「SceneGame.cpp」を新規作成し、SceneGame クラスのメンバ関数を作成する。

手順３．SceneGame クラスの呼び出し（ファイル「WinMain.cpp」）

① SceneGame クラスの実体とScene クラスへのポインタを宣言する。
② Scene クラスのポインタを使ってSceneGame の更新処理と描画処理が実行されるようにゲームループを修正する。
③ ビルド・実行し、CPPGame01D と同じ画面が出るかどうか確認する。

● CPPGame01B「シーンを切り替えよう！」（前回の続きで作成）

タイトル（SceneTitle クラス）を追加し、タイトル→ゲーム→タイトルと切り替わる処理を作成せよ。必要に応じて SceneGame クラス、WinMain 関数なども改造すること。

SceneTitle クラスの仕様（新規ファイル「SceneTitle.h」「SceneTitle.cpp」）

メンバ関数	処理
void update()	[スペース]キーが押されていたらゲーム（SceneGame クラス）へ切り替える。
void draw()	文字列「TITLE です」と「スペースキーを押してね」を描画する（座標や色、サイズは各自で設定）。

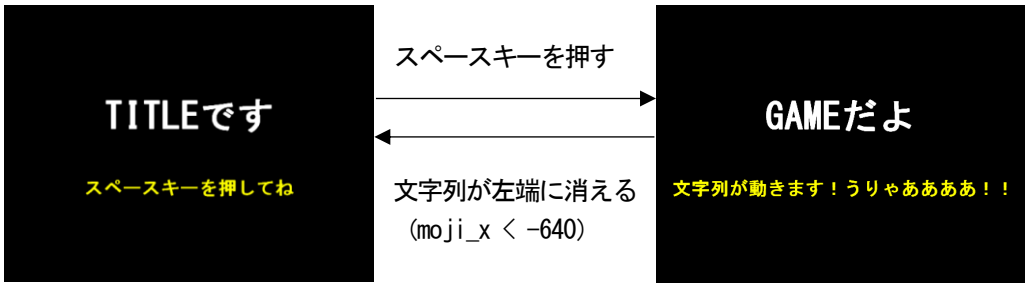
シーンの仕様

シーン No.	ラベル	実行する処理
0	SCENE_TITLE	タイトルを実行
1	SCENE_GAME	ゲームを実行

シーン切り換え関数（ファイル「WinMain.cpp」に作成）

書式	処理
void setScene(int scene)	scene で指定されたシーンに切り替える。

シーン切り替えの仕様



● GL 仕様書

入力処理（GL::Input 関数）で設定される入力情報

変数	説明
int pad_state	押されているキーの情報。ひとつのビットにひとつのキーが割り当てられている。 PAD_UP :[↑]キー　PAD_DOWN :[↓]キー PAD_LEFT :[←]キー　PAD_RIGHT :[→]キー PAD_START :[スペース]キー　PAD_SELECT :[F1]キー PAD_TRG1 :[Z]キー　PAD_TRG2 :[X]キー PAD_TRG3 :[C]キー
int pad_trg	キーが押された瞬間だけビットが設定される（トリガー入力）。キー割り当ては pad_state と同じ。

例：スペースキーが押されているかどうかのチェック

```
if (pad_state & PAD_START) {
    // スペースキーが押されているときの処理
}
```

手順１．SceneTitle クラスの作成（新規ファイル「SceneTitle.h」「SceneTitle.cpp」）

④ ヘッダファイル「SceneTitle.h」を新規作成し、SceneTitle クラスを定義する（**リスト１**）。

⑤ ソースファイル「SceneTitle.cpp」を新規作成し、SceneTitle クラスのメンバ関数を作成する（**リスト２**）。ただし update 関数のコードは**手順３**で実装する。

手順２．シーン切り換えの準備（新規ファイル「WinMain.h」、「WinMain.cpp」）

① ヘッダファイル「WinMain.h」を新規作成し、シーンラベルを定義する。

② 「WinMain.cpp」に setScene 関数を作成し（**リスト３**）、「WinMain.h」でプロトタイプ宣言する。

手順３．シーン切り換え処理の作成（ファイル「SceneTitle.cpp」「SceneGame.cpp」）

① SceneTitle の更新処理にシーン切り替え処理を追加する（**リスト２**）。

② SceneGame の更新処理にシーン切り替え処理を追加する。

手順４．ゲームループのシーン切り替え対応（ファイル「WinMain.cpp」）

① シーン切り替えに対応するようにゲームループを修正する（**リスト３**）。

② ビルド・実行して、タイトル→ゲーム→タイトルと切り替わることを確認する（ただしこのあと再度ゲームへ切り替えようとしても切り替わらない）。

手順５．不具合の修正（ファイル「Scene.h」「SceneGame.*」「WinMain.cpp」）

③ Scene クラスに初期設定関数（void init()）を追加し、SceneGame でオーバーライドする。

④ シーンが切り換わったときに init 関数を呼ぶようにゲームループを修正する（**リスト３**）。

⑤ ビルド・実行して、２回目以降のゲーム処理も正常に実行されることを確認する。

リスト１：SceneTitle クラス（新規ファイル「SceneTitle.h」）

```
#include "Scene.h"
SceneTitleクラスを定義する
```

リスト２：SceneTitle クラスのメンバ関数（新規ファイル「SceneTitle.cpp」）

```
#include "./GL/GL.h"
必要なヘッダファイルをインクルードする

void SceneTitle::update()
{
    スペースキー（PAD_START）が押されたらゲームへ切り換え（コードは手順３で実装する）
}

void SceneTitle::draw()
{
    GL::ClearScene();
    GL::DrawStringL(320, 160, "TITLEです", COLOR_WHITE, true);
    GL::DrawStringM(320, 300, "スペースキーを押してね", COLOR_YELLOW, true);
}
```

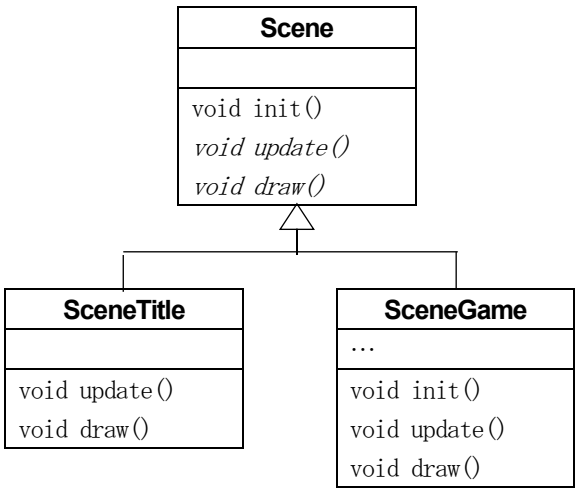
リスト３：シーンの切り替え（ファイル「WinMain.cpp」）

```
// インクルード
必要に応じてインクルードを追加する
// 実体宣言
必要に応じて変数宣言する
```

```
int APIENTRY WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow )
{
    // 初期設定
    GL::Init(_T("ゲームプログラミング"), 640, 480, false);
    setScene関数で最初に実行するシーンを設定する

    // メインループ
    while (GL::GameLoop()) {
            ～
        // 更新処理
        シーン切り替えを行う
        pScene->update();
        // 描画処理
            ～
    }
            ～
}
```

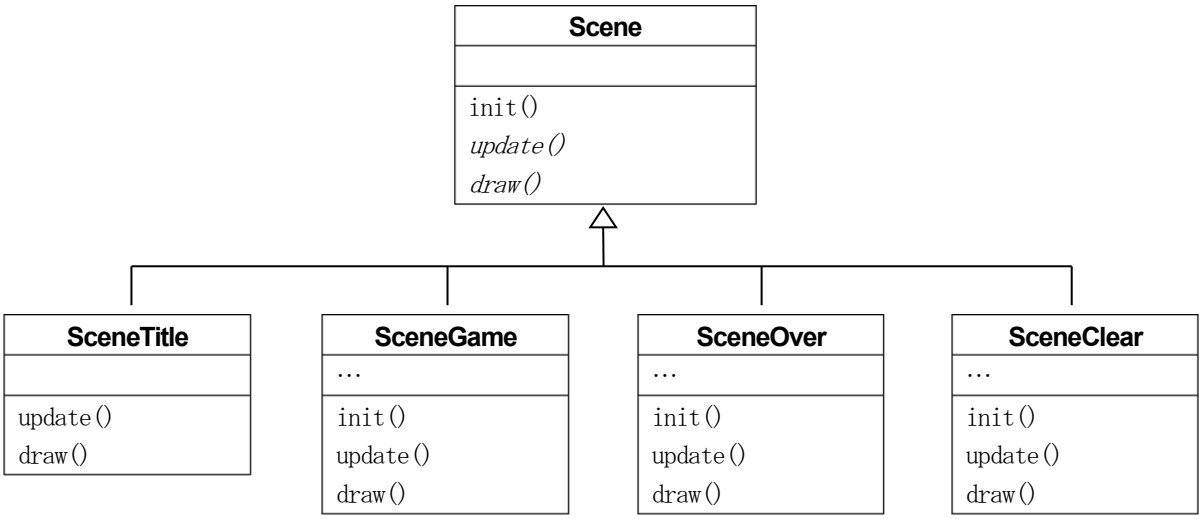
```
// シーンの切り替え
void setScene(int scene)
{
    作成すること（手順２）
}
```

課題完成時のクラス構成



● CPPGame01A「ゲームの雛形を作ろう！」（前回の続きで作成）

ゲームオーバー（SceneOver クラス）とゲームクリア（SceneClear クラス）を追加せよ。なお、必要に応じてメンバ変数を追加すること。



シーンの仕様

シーン	仕様
タイトル (SCENE_TITLE)	（追加作成なし）
ゲーム (SCENE_GAME)	**※以下の仕様を追加する。** Z キー（PAD_TRG1）が押されたら、ゲームオーバーへ切り換える。 X キー（PAD_TRG2）が押されたら、ゲームクリアへ切り換える。
ゲームオーバー (SCENE_OVER)	"GAME OVER"と表示する（点滅させること）。 5 秒経過したらタイトルへ切り換える。
ゲームクリア (SCENE_CLEAR)	"GAME CLEAR"と表示する（点滅させること）。 5 秒経過したらタイトルへ切り換える。

※　表示する文字列の座標・色・大きさは各自で設定すること。