# **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！！" | 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

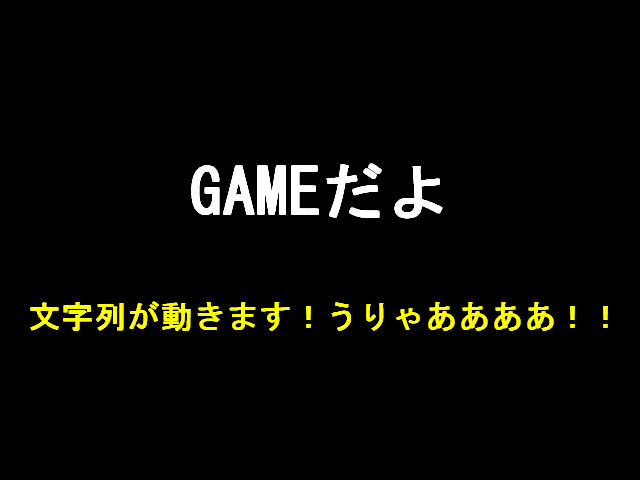
}

// 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。

|  |
| --- |
| **Scene** |
|  |
| *void update()*  *void draw()* |

|  |
| --- |
| **SceneGame** |
| … |
| void update()  void draw() |

**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」）**

1. 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）。 |

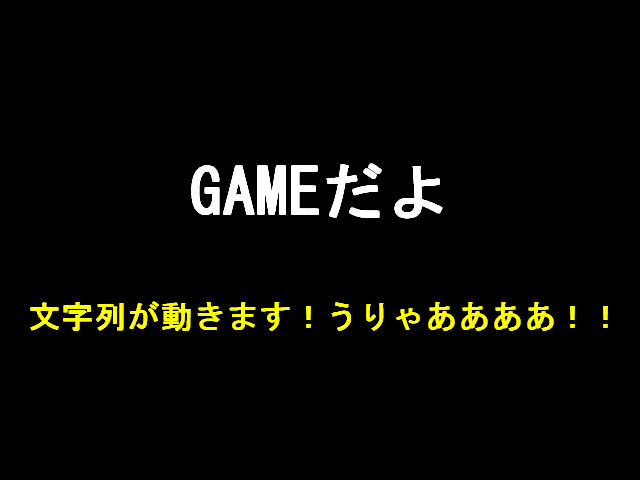
**シーンの仕様**

|  |  |  |
| --- | --- | --- |
| 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**



スペースキーを押す

文字列が左端に消える

（moji\_x < -640）

# **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」）**

1. 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**）。
2. 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」）**

1. 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。
2. 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**）。
3. 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**

|  |
| --- |
| **Scene** |
|  |
| void init()  *void update()*  *void draw()* |

|  |
| --- |
| **SceneGame** |
| … |
| void init()  void update()  void draw() |

|  |
| --- |
| **SceneTitle** |
|  |
| void update()  void draw() |

# **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。

|  |
| --- |
| **Scene** |
|  |
| init()  *update()*  *draw()* |

|  |
| --- |
| **SceneClear** |
| … |
| init()  update()  draw() |

|  |
| --- |
| **SceneOver** |
| … |
| init()  update()  draw() |

|  |
| --- |
| **SceneGame** |
| … |
| init()  update()  draw() |

|  |
| --- |
| **SceneTitle** |
|  |
| update()  draw() |

**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。

スペース

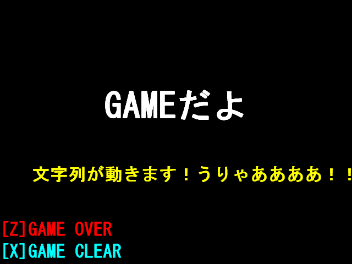
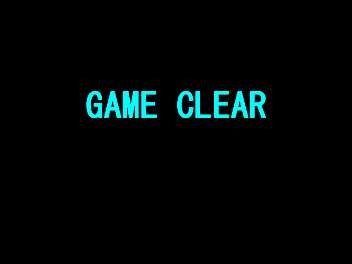
（PAD\_START）

Xキー

（PAD\_TRG2）

Zキー

（PAD\_TRG1）



ゲームオーバー

タイトル

ゲームクリア

ゲーム

5秒経過

5秒経過