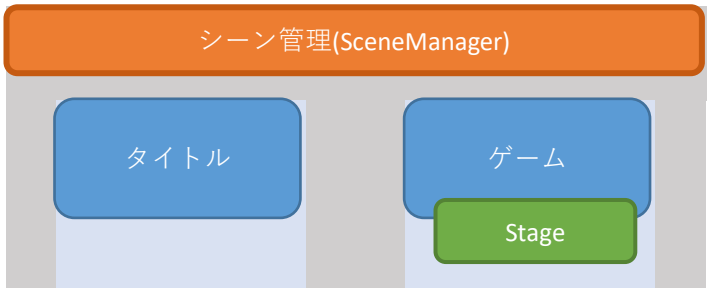
３Ｄモデルの表示

これまでは、球体等のプリミティブと呼ばれる３Ｄオブジェクトを
描画してきました。
ここでいうプリミティブとは、プログラム上で生成できる簡易的な３Ｄ形状
のことです。
実際のゲームに登場するようなキャラクターやステージを描画するには、
画像のように、外部ファイルを読み込んで、画面描画していく手順を取ります。

試しにこちらで用意した３Ｄステージを画面に描画していきたいと思います。

GameSceneクラスを作り、シーン遷移できるようにして、
３Ｄステージを描画するためのStageクラスを作成していきます。



３階層の作り方、覚えてますでしょうか？

Stage.h

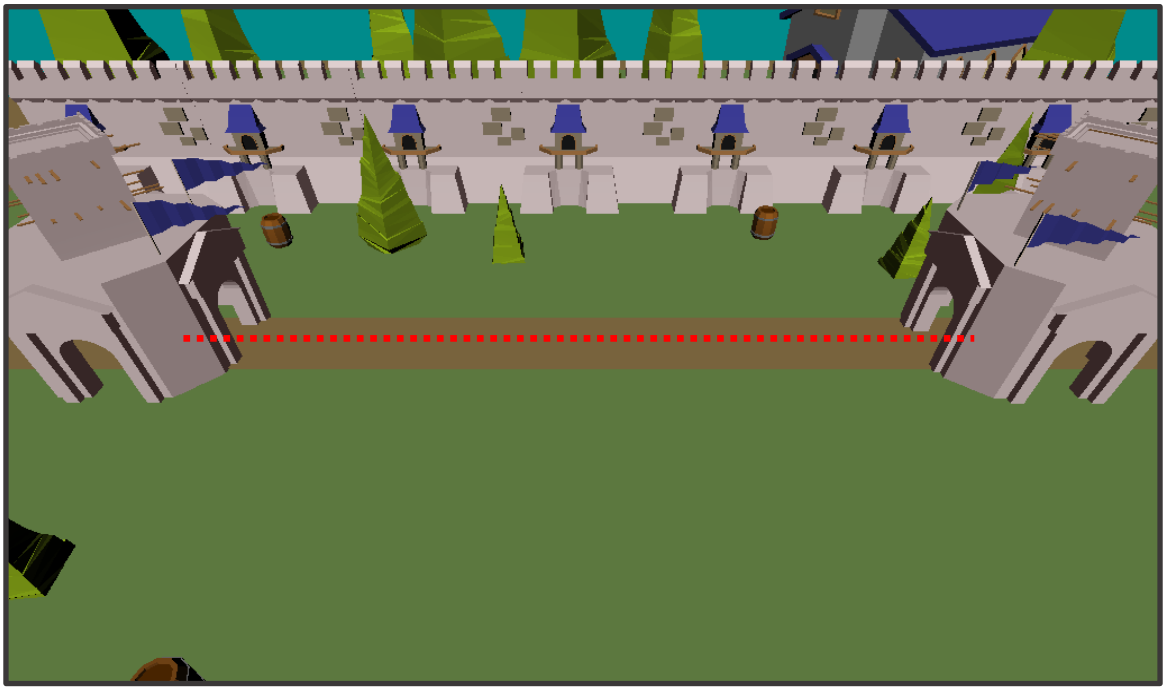
```
#pragma once
class Stage
{

public:

    // コンストラクタ
    Stage(void);

    // デストラクタ
    ~Stage(void);

    void Init(void);
    void Update(void);
    void Draw(void);
    void Release(void);
```

```
private:

    // ３Ｄモデルのハンドル ID
    int modelId_;

};
```

Stage.cpp

```
#include <DxLib.h>
#include "../Application.h"
#include "Stage.h"

Stage::Stage(void)
{
}

Stage::~Stage(void)
{
}

void Stage::Init(void)
{

    // 外部ファイルの３Ｄモデルをロード
    modelId_ = MV1LoadModel(
                (Application::PATH_MODEL + "Stage/Stage.mv1").c_str());

    // ３Ｄモデルの大きさを設定(引数は、x, y, zの倍率)
    MV1SetScale(modelId_, { 1.0f, 1.0f, 1.0f });

    // ３Ｄモデルの位置(引数は、３Ｄ座標)
    MV1SetPosition(modelId_, { 0.0f, 0.0f, 0.0f });

    // ３Ｄモデルの向き(引数は、x, y, zの回転量。単位はラジアン。)
    MV1SetRotationXYZ(modelId_, { 0.0f, DX_PI_F, 0.0f });

}

void Stage::Update(void)
```

```
{
}

void Stage::Draw(void)
{
    // ロードされた３Ｄモデルを画面に描画
    MV1DrawModel(modelId_);
}

void Stage::Release(void)
{
    // ロードされた３Ｄモデルをメモリから解放
    MV1DeleteModel(modelId_);
}
```



こんな感じでステージモデルが描画されたらＯＫです。
このステージを使って簡単なラインディフェンスゲームを作りたいと思います。

赤点線のゲームラインがしっかり見えるように、
カメラ調整を行ってから、本格的にゲーム作りに入っていきましょう。

カメラの位置や角度を調整したいのですが、
プログラム内の数字を変更して、実行して、確認して、
イマイチだったら、また変更して、実行して、確認して。。。を
何度も繰り返すことになりますので、フリーカメラの機能を追加して、
座標や角度を可視化していきたいと思います。

カメラにSTATEデザインパターンとして、MODEを追加します。

Camera.h

```
public:

    // カメラモード
    enum class MODE
    {
        NONE,
        FIXED_POINT,      // 定点カメラ
        FREE,             // フリーモード
    };

    ～ 省略 ～

    void SetBeforeDraw(void);
    void SetBeforeDrawFixedPoint(void);
    void SetBeforeDrawFree(void);

    ～ 省略 ～

    // カメラモードの変更
    void ChangeMode(MODE mode);

private:

    // カメラモード
    MODE mode_;

    ～ 省略 ～
```

Camera.cpp

```
void Camera::Init(void)
{
    // カメラの位置
    pos_ = { 0.0f, 500.0f, -500.0f };

    // カメラの角度
    angles_ = { 40.0f * DX_PI_F / 180.0f, 0.0f, 0.0f };

    // 定点カメラを初期状態にする
    ChangeMode(MODE::FREE);
}

void Camera::SetBeforeDraw(void)
{

    // クリップ距離を設定する(SetDrawScreenでリセットされる)
    SetCameraNearFar(10.0f, 30000.0f);

    switch (mode_)
    {
    case Camera::MODE::FIXED_POINT:
        SetBeforeDrawFixedPoint();
        break;
    case Camera::MODE::FREE:
        SetBeforeDrawFree();
        break;
    }

    // カメラの設定(位置と角度による制御)
    SetCameraPositionAndAngle(
        pos_,
        angles_.x,
        angles_.y,
        angles_.z
    );

}
```

最終的には、
ここの位置と角度の
数字を変える。

STATE
デザインパターン

```
void Camera::SetBeforeDrawFixedPoint(void)
{
    // 何もしない
}

void Camera::SetBeforeDrawFree(void)
{

    auto& ins = InputManager::GetInstance();

    // WASDでカメラの位置を変える
    float movePow = 3.0f;
    if (ins.IsNew(KEY_INPUT_W)) { ？？？ }
    if (ins.IsNew(KEY_INPUT_A)) { ？？？ }
    if (ins.IsNew(KEY_INPUT_S)) { ？？？ }
    if (ins.IsNew(KEY_INPUT_D)) { ？？？ }
    if (ins.IsNew(KEY_INPUT_Q)) { ？？？ }
    if (ins.IsNew(KEY_INPUT_E)) { ？？？ }

    // 矢印キーでカメラの角度を変える
    float rotPow = 1.0f * DX_PI_F / 180.0f;
    if (ins.IsNew(KEY_INPUT_DOWN))       { ？？？ }
    if (ins.IsNew(KEY_INPUT_UP))         { ？？？ }
    if (ins.IsNew(KEY_INPUT_RIGHT))      { ？？？ }
    if (ins.IsNew(KEY_INPUT_LEFT))       { ？？？ }

}
```

？？？を埋めて、
カメラの座標や角度を動かすよ

InputManager への
各種キー登録を忘れずに

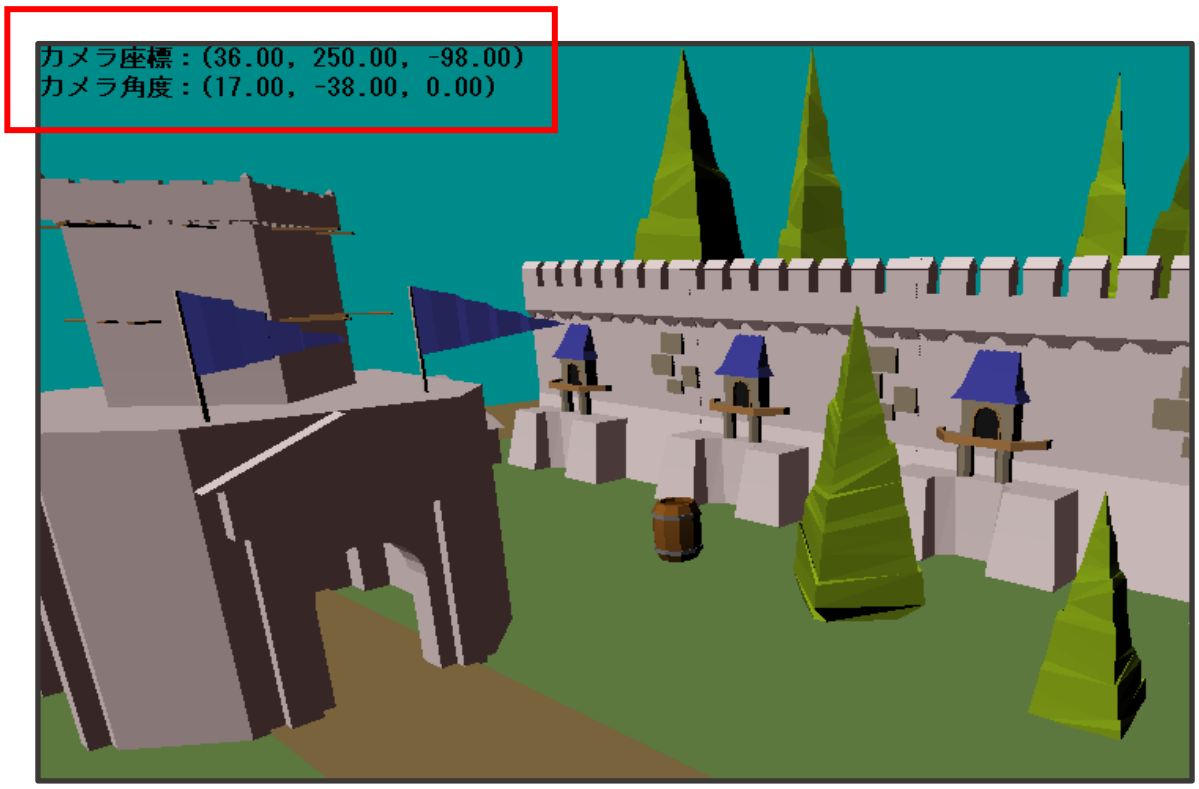
※ カメラの移動について

Wキーを押すと、ワールドZ軸の正方向にカメラが移動、
Dキーを押すと、ワールドX軸の正方向にカメラが移動する形で大丈夫です
本来であれば、Wキーを押すと、ワールドではなく、カメラの前方方向に
移動するのが正しいのですが、もう少し、授業が進んでから紹介します。
どうしてもという方は、個別にお尋ねください。

```
void Camera::Draw(void)
{
    DrawFormatString(0,  0, 0x000000,
        カメラ座標：(%.2f, %.2f, %.2f), pos_.x, pos_.y, pos_.z);
    DrawFormatString(0, 20, 0x000000, "カメラ角度：(%.2f, %.2f, %.2f)",
        angles_.x * 180.0f / DX_PI_F,
        angles_.y * 180.0f / DX_PI_F,
        angles_.z * 180.0f / DX_PI_F);
}

void Camera::Release(void)
{
}

void Camera::ChangeMode(MODE mode)
{

    // カメラモードの変更
    mode_ = mode;

    // 変更時の初期化処理
    switch (mode_)
    {
    case Camera::MODE::FIXED_POINT:
        break;
    case Camera::MODE::FREE:
        break;
    }

}
```

カメラの機能追加が終わったら、
SceneManagerからゲッター経由でCameraポインタを取得できるようにして、
ゲームシーンのInitでカメラモードをフリーモードに設定しましょう。

```
void GameScene::Init(void)
{

    stage_ = new Stage();
    stage_->Init();

    // カメラをフリーモードにする
    SceneManager::GetInstance().GetCamera()->ChangeMode(Camera::MODE::FREE);

}
```



カメラ移動できるようになっていると思います。
(上図のカメラ設定では、ゲームラインが見えないので使えませんが。。)

自分が見やすいカメラにして、画面に表示されている座標や角度を、
カメラクラスの初期値として設定しましょう。