Chapter-7 Adapterパターン

7.1 【ハンズオン】Adapterパターンを使ってみる。

　では、AdapterパターンをつかってDirectX10～12の描画APIに対応した3Dモデル描画処理の疑似サンプルを作ってみましょう。Adapter_01/Adapter_01.slnを開いてください。

　このサンプルはゲーム開始時に、使用する描画APIを選択することができます。現在は、DirectX11とDirectX12のみ実装済みで、DirectX10は未実装です。そのため、現在はゲーム開始時にDirectX10を選択すると、プログラムはクラッシュします。

　このサンプルでは、3Dモデルを描画するインターフェースを定義したIModelクラスを継承して、DirectX11向けとDirectX12向けのクラスを実装しています。ですので、DirectX10を利用する場合もこの設計に合わせたクラスを実装する必要があります。しかし、DirectX10は別のプロジェクトで作成している、modelRenderDx10クラスがあります。開発効率を上げるため、車輪の再発明を避けるためにも、是非ともこれを利用したい(今更DirectX10のような古い描画APIを勉強するのはモチベーションも上がりませんよね)。しかし、modelRenderDx10クラスは別のプロジェクトで利用しているライブラリであるため、IModelクラスは継承していない。そのため、そのままこのプロジェクトで利用はできません。

　modelRenderDx10クラスを改造すれば、問題は解決しますが、このクラスは別プロジェクトで使用されているクラスで、今後もバグフィクスなどのバージョンアップが行われる可能性があります。そのため、modelRenderDx10クラスを改造するのは避けたい。

　今回のサンプルでは、「現在のプロジェクト用の設計に合わせてmodelRenderDx10クラスを利用できるようにしたい。」という欲求があります。しかし、「modelRenderDx10クラスは他プロジェクトでも利用しているため、改造することはできない(今後のバグフィクスも考えると、改造したくない)。」という問題があります。

　では、この欲求をかなえるためにAdapterパターンを利用して、DirectX10向けの描画クラスを実装していきましょう。

step-1 modelRenderDx10を利用するアダプタクラス宣言を作成する。

　まずはmodelRenderDx10のインターフェースをゲームが望んでいる形に変換するためのアダプタクラスを宣言します。このクラスがIModelクラスを継承して、ゲームが望んでいるインターフェースを定義します。しかし、step-2のアダプタクラス定義の作成で見ていくように、実装の詳細はmodelRenderDx10クラスに委譲します。そのため、メンバ変数として、modelRenderDx10クラスのオブジェクトを保持しています。では、ModelDx12.hに次のプログラムを入力してください。

[ModelDx10.h]

```
// step-1 modelRenderDx10を利用するアダプタクラス宣言を作成する。
class ModelDx10 : public IModel
{
public:
    void Init() override;
    void BeginDraw() override;
    void Draw() override;
    void EndDraw() override;
private:
```

```
    modelRenderDx10 m_modelRenderDx10;  // 実装の詳細はmodelRenderDx10に委譲する。
};
```

step-2 modelRenderDx10を利用するアダプタクラス定義を作成する。

続いて、アダプタクラスの定義を実装していきます。step-1でも説明したように、実装の詳細はmodelRenderDx10に委譲することになります。では、ModelDx10.cppに次のプログラムを入力してください。 [ModelDx10.cpp]

```
// step-2 modelRenderDx10を利用するアダプタクラス定義を作成する。
void ModelDx10::Init()
{
    // modelRenderDx10に委譲する
    m_modelRenderDx10.initialyze();
}
void ModelDx10::BeginDraw()
{
    // modelRenderDx10に委譲する
    m_modelRenderDx10.prepareDraw();
}
void ModelDx10::Draw()
{
    // modelRenderDx10に委譲する
    m_modelRenderDx10.draw();
}
void ModelDx10::EndDraw()
{
    // modelRenderDx10に委譲する
    m_modelRenderDx10.endDraw();
}
```

step-3 ModelDx10.hをインクルードする。

step-2でアダプタクラスとなる、ModelDx10クラスが完成しました。では、ModelDx10クラスを利用していきましょう。main.cppを開いてください。まずは、ModelDx10.hをインクルードします。該当するコメントの箇所に次のプログラムを入力してください。 [main.cpp]

```
// step-3 ModelDx10.hをインクルードする。
#include "ModelDx10.h"
```

step-4 ModelDx10のインスタンスを作成する。

いよいよ最後です。描画APIとしてDirectX10が選ばれた際に、ModelDx10クラスのインスタンスを作成するコードを追加します。main.cppの該当するコメントの箇所に次のプログラムを入力してください。入力出来たら、実行して、描画APIにDirectX10を選んでみてください。 [main.cpp]

```
// step-4 ModelDx10のインスタンスを作成する。
model = new ModelDx10;
```