class template 부분 특수화를 사용해 mesh type에 맞는 builder class 생성 자동화

작성자: 이정모

게임 엔진 사용자가 factory를 생성해서 builder에 데이터를 채운 뒤
CreateObject() 함수만 호출하면 object를 생성해주는
builder-factory 패턴을 구현하던 중이었습니다.

저는 처음에 object가 non mesh, mesh, skinned mesh로 나뉘는지 모르는 상태에서 builder를 제작했습니다.

```
public:
    void SetHandleDevice(EResourceType deviceType);
    void SetHandleDeviceContext(EResourceType deviceContextType);

    void SetHandleVertexShader(EResourceType vertexShaderType);
    void SetHandlePixelShader(EResourceType pixelShaderType);
    void SetHandleInputLayout(EResourceType inputLayoutType);

    void SetHandleVertexBuffer(EResourceType VertexBufferType);
    void SetHandleIndexBuffer(EResourceType IndexBufferType);
    void SetHandleConstantBuffer(EResourceType ConstantBufferType);
    void SetIndexBufferSize(EIndexBufferSizeType eIndexBufferSizeType);

    // none mesh object들은 텍스처나 샘플러가 필요 없는데
    // 이것들을 굳이 세팅해야 하나
    void SetHandleTextureDiffuseMap(EResourceType textureDiffuseMapType);
    void SetHandleSampler(EResourceType samplerType);
```

제작하다보니 object마다 mesh type이 다르다는 것을 알게 되었고
mesh type이 다르면 필요한 자원이 다르다는 것을 알게 되었습니다.

그런데 저는 builder class 하나가 모든 object의 생성에 대응하고 있었고

texture, sampler가 none mesh object에는 필요하지 않지만

mesh object에는 필요하기 때문에 모든 멤버 함수와 변수를 다 만들어야 하는 상황에 봉착했습니다.

저는 object마다 필요한 자원이 다른데 하나의 큰 builder가 불필요하게 멤버 함수와 멤버 변수를 다 들고 있는 것이 비합리적이라 생각했고

none mesh object에 대응하는 builder는 texture나 sampler를 세팅하는 멤버 함수나 멤버 변수를 가지고 있을 필요가 없는데

굳이 가지고 있으면 “이게 왜 있지?”라고 한번 더 신경 써야 하는 것도 비합리적이라고 생각했습니다.

object의 mesh type이 다르면 결국 세팅해줘야 하는 자원이 달라지고

이는 결국 builder가 가져야 할 멤버의 signature가 다르다는 것을 의미합니다.

저는 이 순간 “아, 이건 signature가 다르기 때문에 분리해서 각각 따로 만들어야겠다”고 생각했습니다.

그래서 생각한 것이 builder class template 부분 특수화입니다.

```
// main class template인 Builder<ObjectType, MeshType>을
// Builder<ObjectType, Mesh>로 부분 특수화한 class template
template <typename ObjectType>
class Builder<ObjectType, Mesh>
{
public:
    Builder<ObjectType, Mesh>(ResourceManager* pResourceManager);
    ~Builder<ObjectType, Mesh>();
```

object type이 있고 이 object가 사용하는 mesh type이 있을텐데

object type은 무엇이 들어오더라도 특정 mesh type에 대한 전용 builder를 컴파일러가 자동으로 만들게 하자고 생각했습니다.

factory는 일반화가 되어있는 상태였기 때문에 ObjectType을 받고 있었고

builder를 factory 내부에 넣게 되면 이 ObjectType을 받을 수 있게 되고

결국 mesh type만 알아내면 되는 것이었습니다.

저는 이를 using 문을 사용한 타입 재정의로 해결했습니다.

```
class Mesh : public IObject
{
    // Mesh를 상속받은 object는 MeshType이라 재정의된 이름으로 부모의 Mesh Type에 접근 가능
public:
    using MeshType = Mesh;
```

Mesh class가 스스로의 타입을 MeshType이라 재정의하고

```
class Box : public Mesh
{
    // 부모 클래스에서 MeshType이 무엇인지 using문을 사용해서 재정의 했고
    // 자식 클래스에서 부모의 MeshType을 가져와서 같은 이름으로 using문을 사용해 재정의 했음
    // 본인의 mesh type을 밖에서도 참조할 수 있도록 using문을 통해 타입을 재정의한 것이 핵심임
    // 이를 통해 Object Type을 알면 object가 상속받은 MeshType을 가져와서 전용 builder를 생성할 수 있음
public:
    using MeshType = MeshType; // Box::MeshType = Mesh::MeshType
```

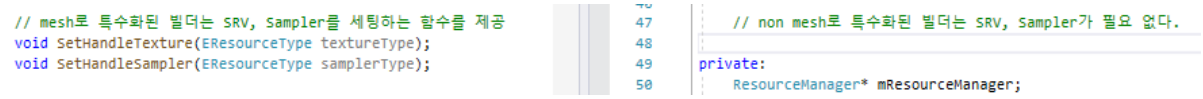
mesh class를 상속한 box class는 부모의 MeshType을 가져와서 이름 그대로 MeshType 이라 다시 재정의 했습니다.

```
47
48      // ObjectType 내부에 using문을 사용하여 자신이 상속받은 mesh type을 재정의 해둔 것을 가져옴
49      // 이로써 factory가 object type을 받으면 이를 기준으로 mesh type을 가져와서 전용 builder를 자동으로 생성
50      Builder<ObjectType, typename ObjectType::MeshType>* mBuilder;
51
52      // ObjectType::MeshType과 같이 타입에 종속(::)적인 이름은 컴파일러가 변수로 해석할 수 있기 때문에 typename 키워드로 명시하여 타입임을 알려줌
```

결국 Builder는 factory가 생성될 때 ObjectType을 알 수 있고

ObjectType 내부에 MeshType을 재정의 했기 때문에 MeshType도 알 수 있게 되어

MeshType이 none mesh인지, mesh인지, skinned mesh인지에 따라 그에 맞는 전용 builder가 자동으로 생성되도록 만들었습니다.



mesh로 특수화된 builder는 texture나 sampler를 세팅하는 멤버 함수와 멤버 변수가 있지만

none mesh로 특수화된 builder는 없습니다.

```
// grid로 생성한 팩토리 내부에서 자동으로 생성된 builder class는
// 클래스 템플릿 부분 특수화를 통해 애초에 texture나 sampler를 세팅하는 함수와 변수를 만들지 않았음
// mGridFactory->GetBuilder()->SetHandleTexture(EResourceType::None); // error // 함수 존재 x
// mGridFactory->GetBuilder()->SetHandleSampler(EResourceType::None); // error // 함수 존재 x
```

grid object는 none mesh type이기 때문에 builder에 texture, sampler를 세팅하는 멤버 함수가 아예 존재하지 않지만

```
// mesh를 가지는 box로 생성된 factory의 builder는 Texture와 Sampler를 세팅하는 함수가 존재함
// 클래스 템플릿 부분 특수화를 통해 mesh를 가진 object에 대한 builder는 Texture와 Sampler를 세팅하는 함수를 만들어 주었음
mBoxFactory->GetBuilder()->SetHandleTexture(EResourceType::TexDarkBrick);
mBoxFactory->GetBuilder()->SetHandleSampler(EResourceType::SamplerLinear);
```

box object는 mesh type이기 때문에 builder에 texture나 sampler를 세팅하는 멤버 함수가 있습니다.

마무리하자면

처음에는 builder class가 하나만 있었고 이 builder가 모든 object의 mesh type에 대응해야 했기 때문에 불필요한 멤버 함수나 멤버 변수를 가지고 있게 되었습니다.

저는 이것이 비합리적이라 생각했고 class template 부분 특수화를 사용하였습니다.

object type이 어떤 것이 들어오더라도 none mesh, mesh, skinned mesh에 대해서 builder class를 부분 특수화 함으로써 각각에 맞는 builder가 자동으로 생성되도록 하여 문제를 해결했습니다.

이상입니다.