Supplements (1)

```
std::unordered_map<std::string, std::unique_ptr<Material>> mMaterials;

void LitColumnsApp::BuildMaterials() {
   auto bricks0 = std::make_unique<Material>();
   bricks0->Name = "bricks0";
   bricks0->MatCBIndex = 0;
   bricks0->DiffuseSrvHeapIndex = 0;
   bricks0->DiffuseAlbedo = XMFLOAT4(Colors::ForestGreen);
   bricks0->FresnelR0 = XMFLOAT3(0.02f, 0.02f, 0.02f);
   bricks0->Roughness = 0.1f;

   auto stone0 = std::make_unique<Material>();
   stone0->Name = "stone0";
   stone0->MatCBIndex = 1;
   stone0->DiffuseSrvHeapIndex = 1;
   stone0->DiffuseAlbedo = XMFLOAT4(Colors::LightSteelBlue);
   stone0->FresnelR0 = XMFLOAT3(0.05f, 0.05f, 0.05f);
   stone0->Roughness = 0.3f;
```

Supplements (2)

```
    auto tile0 = std::make_unique<Material>();
    tile0->Name = "tile0";
    tile0->MatCBIndex = 2;
    tile0->DiffuseSrvHeapIndex = 2;
    tile0->DiffuseAlbedo = XMFLOAT4(Colors::LightGray);
    tile0->FresnelR0 = XMFLOAT3(0.02f, 0.02f, 0.02f);
    tile0->Roughness = 0.2f;

    auto skullMat = std::make_unique<Material>();
    skullMat->Name = "skullMat";
    skullMat->MatCBIndex = 3;
    skullMat->DiffuseSrvHeapIndex = 3;
    skullMat->DiffuseAlbedo = XMFLOAT4(1.0f, 1.0f, 1.0f, 1.0f);
    skullMat->FresnelR0 = XMFLOAT3(0.05f, 0.05f, 0.05);
    skullMat->Roughness = 0.3f;

    mMaterials["bricks0"] = std::move(bricks0);
    mMaterials["stone0"] = std::move(stone0);
    mMaterials["tile0"] = std::move(tile0);
    mMaterials["skullMat"] = std::move(skullMat);
}
```

Supplements (3)

```
// FrameResource.h
struct FrameResource {
public:

    FrameResource(ID3D12Device* device, UINT passCount,
       UINT objectCount, UINT materialCount);
    FrameResource(const FrameResource& rhs) = delete;
    FrameResource& operator=(const FrameResource& rhs) = delete;
    ~FrameResource();

    Microsoft::WRL::ComPtr<ID3D12CommandAllocator> CmdListAlloc;

    std::unique_ptr<UploadBuffer<PassConstants>> PassCB = nullptr;
    std::unique_ptr<UploadBuffer<MaterialConstants>> MaterialCB = nullptr;
    std::unique_ptr<UploadBuffer<ObjectConstants>> ObjectCB = nullptr;

    UINT64 Fence = 0;
};
```

Supplements (4)

```
void LitColumnsApp::UpdateMaterialCBs(const GameTimer& gt) {
    auto currMaterialCB = mCurrFrameResource->MaterialCB.get();
    for(auto& e : mMaterials) {
        Material* mat = e.second.get();
        if(mat->NumFramesDirty > 0) {
            XMMATRIX matTransform = XMLoadFloat4x4(&mat->MatTransform);

            MaterialConstants matConstants;
            matConstants.DiffuseAlbedo = mat->DiffuseAlbedo;
            matConstants.FresnelR0 = mat->FresnelR0;
            matConstants.Roughness = mat->Roughness;
            XMStoreFloat4x4(&matConstants.MatTransform,
                XMMatrixTranspose(matTransform));

            currMaterialCB->CopyData(mat->MatCBIndex, matConstants);

            mat->NumFramesDirty--;
        }
    }
}
```

Supplements (5)

```
void LitColumnsApp::UpdateMainPassCB(const GameTimer& gt) {
    XMMATRIX view = XMLoadFloat4x4(&mView);
    XMMATRIX proj = XMLoadFloat4x4(&mProj);

    // …
    mMainPassCB.TotalTime = gt.TotalTime();
    mMainPassCB.DeltaTime = gt.DeltaTime();
    mMainPassCB.AmbientLight = { 0.25f, 0.25f, 0.35f, 1.0f };
    mMainPassCB.Lights[0].Direction = { 0.57735f, -0.57735f, 0.57735f };
    mMainPassCB.Lights[0].Strength = { 0.6f, 0.6f, 0.6f };
    mMainPassCB.Lights[1].Direction = { -0.57735f, -0.57735f, 0.57735f };
    mMainPassCB.Lights[1].Strength = { 0.3f, 0.3f, 0.3f };
    mMainPassCB.Lights[2].Direction = { 0.0f, -0.707f, -0.707f };
    mMainPassCB.Lights[2].Strength = { 0.15f, 0.15f, 0.15f };

    auto currPassCB = mCurrFrameResource->PassCB.get();
    currPassCB->CopyData(0, mMainPassCB);
}
```