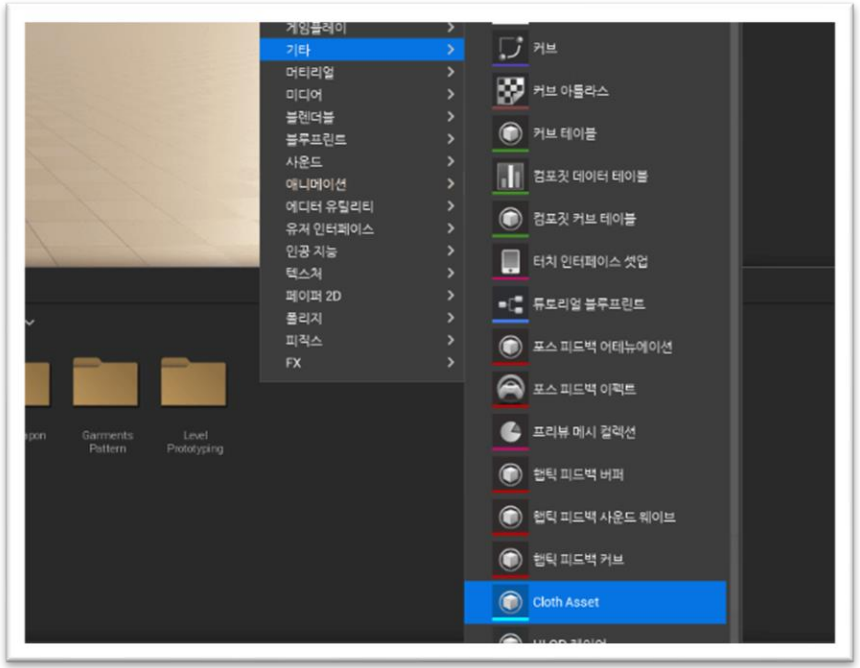
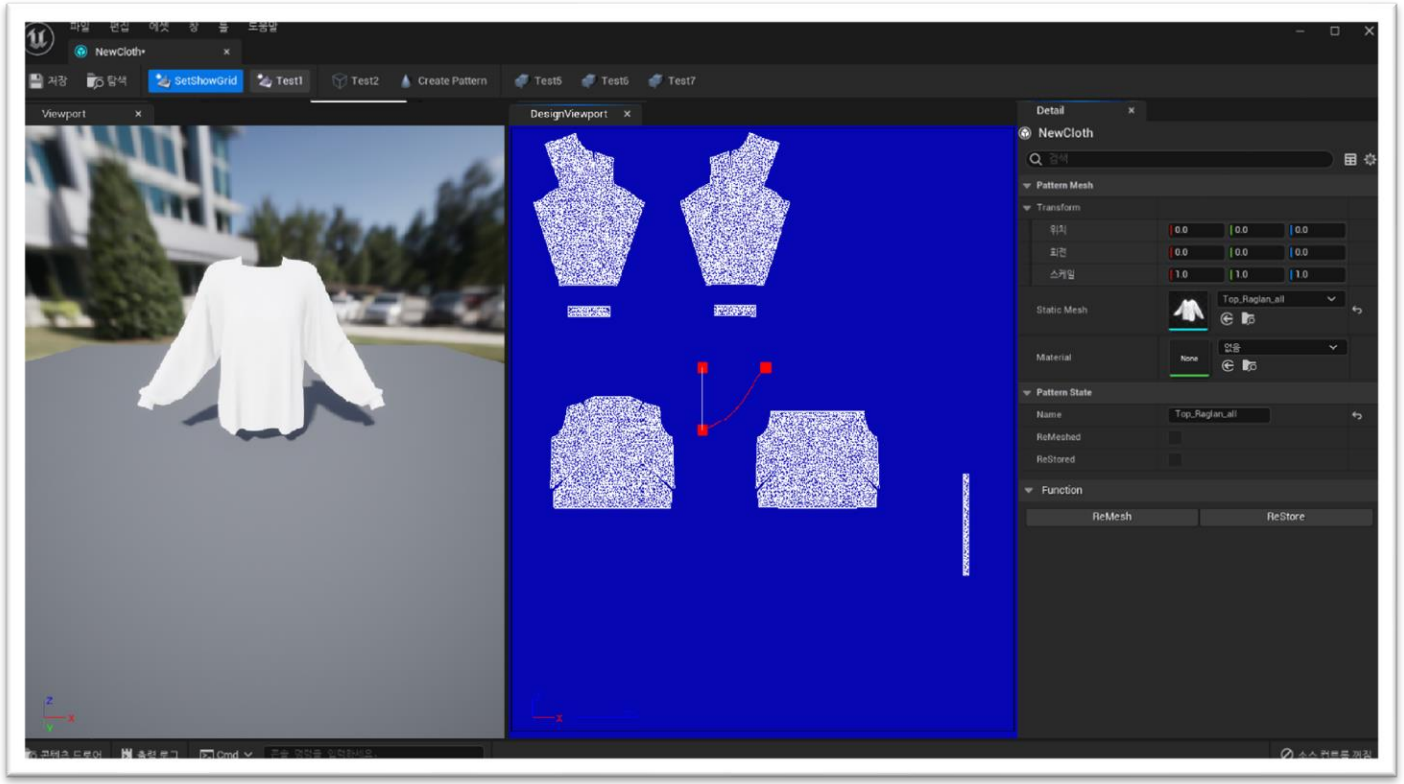
[목차]

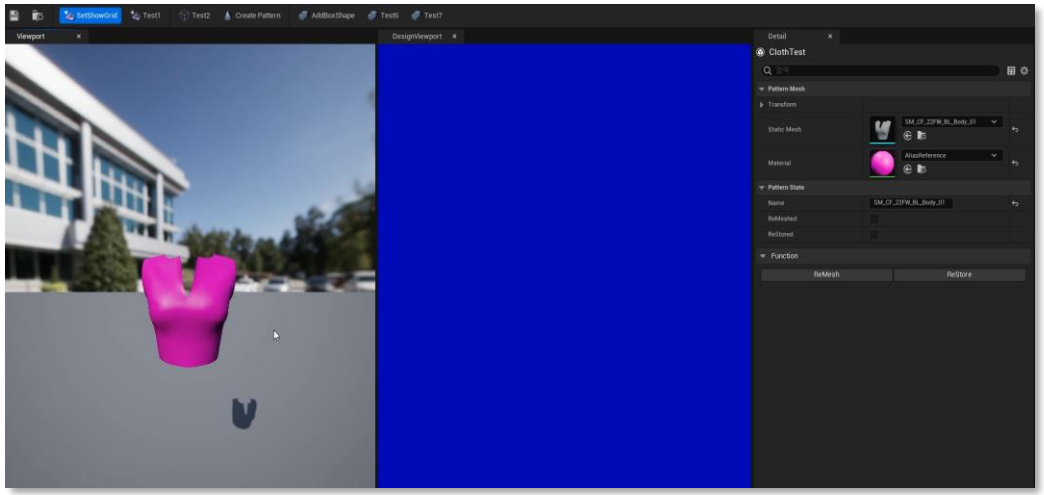
1. 프로젝트 개요





언리얼 엔진의 Slate 라는 Custom User Interface Framework 를 사용해
Custom Asset Editor 를 개발했습니다.
Custom Asset 을 생성할 수 있고, Asset 열면 Editor 가 열리는 구조입니다.

2. 주요 기능 및 구현 내용 – 3D Viewport



3D Viewport :

에디터의 우측 Detail Pannel 에서
Asset의 Mesh 와 Material 을 설정할 수 있습니다.

```
TSharedRef<SDockTab> FClothEditor::SpawnTab_Detail(const FSpawnTabArgs& Args)
{
    check(Args.GetTabId() == FClothEditorTabs::DetailTabId)
        return SNew(SDockTab)
        [
            SNew(SVerticalBox)
            + SVerticalBox::Slot() // SVerticalBox::FSlot::FSlotArguments
            .AutoHeight()
            .Padding( Uniform: 5) // SVerticalBox::FSlot::FSlotArguments&
            [
                DetailsView.ToSharedRef()
            ]

            + SVerticalBox::Slot() // SVerticalBox::FSlot::FSlotArguments
            .AutoHeight()
            .Padding( Uniform: 5) // SVerticalBox::FSlot::FSlotArguments&
            [
                SNew(SExpandableArea)
                .InitiallyCollapsed(false)
                .BorderImage(FCoreStyle::Get().GetBrush( PropertyName: "DetailsView.CollapsedCategory"))
                .Padding(8.0f) // SExpandableArea::FArguments&
                 HeaderContent() // NamedSlotProperty<SExpandableArea::FArguments>
```

- DetailPannel 디자인 코드

Detail Pannel :

Slate 의 TabManager 를 사용해 Detail Pannel 를 생성하고,
Pannel의 디자인은 Slate 문법을 사용해 구현했습니다.

또한, Custom Slate Widget 을 만들어
Name 이라는 Text Block 을 모듈화 하여 생성했습니다.

2. 주요 기능 및 구현 내용 – 3D Viewport

```cpp
void UCustomAsset::PostEditChangeProperty(FPropertyChangedEvent& PropertyChangedEvent)
{
    if (PropertyChangedEvent.Property != nullptr)
    {
        OnPropertyChanged.Broadcast([&]PropertyChangedEvent);
        UpdateMeshInfo();
    }
}
```

- Property 변경 시 Delegate로 위 함수 호출

StaticMesh 와 Material 의 설정은 Delegate를 이용해 구현했습니다.
PostEditChangeProperty 함수 안에서 직접 Static Mesh를 변경해도 되지만,
3D Viewport를 변경된 Mesh로 업데이트하는 작업을 수행할 수 없었습니다.

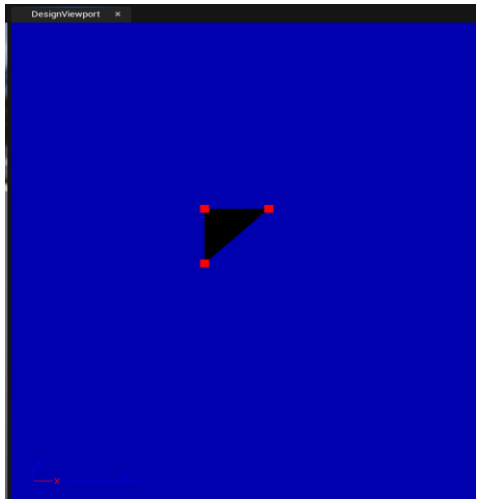
```
"Modules" :
[
    {
        "Name" : "CustomAssetEditor",
        "Type" : "Editor",
        "LoadingPhase" : "Default",
        "AdditionalDependencies": [
          "Paper2D"
        ]
    }
],
"Plugins": [
    {
        "Name" : "CustomAsset",
        "Enabled" : true
    },
```

```
"Modules" :
[
    {
        "Name" : "CustomAsset",
        "Type" : "Runtime",
        "LoadingPhase" : "PreDefault"
    }
]
```

- CustomAsset 은 Runtime Module

그 이유는,
Custom Asset 은 Runtime 모듈이기 때문에
Editor Module에 종속성을 가지면 추 후에
Editor Module을 Plugin이나 독립적인 모듈로 구현할 수 없기 때문입니다.

2. 주요 기능 및 구현 내용 – 2D Viewport



2D Viewport :
Orthographic Camera 를 사용해 삼각형을 보여줍니다.
3D Vertex 를 3개 배치해 삼각형을 그리고 있습니다.

```
void FClothDesignViewportClient::Draw(const FSceneView* View, FPrimitiveDrawInterface* PDI)
{
    //Draw Triangle And Point
    FVector PaperAxisX(1.0f, 0.0f, 0.0f);
    FVector PaperAxisY(0.0f, 0.0f, 1.0f);
    FVector PaperAxisZ(0.0f, 1.0f, 0.0f);

    UMaterialInterface* DefaultMaterial = LoadObject<UMaterialInterface>(Outer:nullptr, Name:TEXT("/Script/Engine.Material'/Game/StarterContent/Materials/M_Basic_Wall.M_
    FMaterialRenderProxy* MP = DefaultMaterial->GetRenderProxy();

    TArray<FDynamicMeshVertex> MeshVertices;
    MeshVertices.Add(Item:FDynamicMeshVertex(InPosition:FVector3f(InX:0, InY:0, InZ:0), InTangentX:(FVector3f)PaperAxisX, InTangentZ:(FVector3f)PaperAxisZ, InTexCoord:FVector2f(InX
    MeshVertices.Add(Item:FDynamicMeshVertex(InPosition:FVector3f(InX:0, InY:0, InZ:100), InTangentX:(FVector3f)PaperAxisX, InTangentZ:(FVector3f)PaperAxisZ, InTexCoord:FVector2f(
    MeshVertices.Add(Item:FDynamicMeshVertex(InPosition:FVector3f(InX:100, InY:0, InZ:100), InTangentX:(FVector3f)PaperAxisX, InTangentZ:(FVector3f)PaperAxisZ, InTexCoord:FVector2f
    FDynamicMeshBuilder MeshBuilder(View->GetFeatureLevel());

    int vi[3];
    for (int i = 0; i < 3; i++)
    {
        vi[i] = MeshBuilder.AddVertex(MeshVertices[i]);
    }

    for (int32 VertexIndex = 0; VertexIndex < MeshVertices.Num(); VertexIndex++)
    {
        // 정점 렌더링
        const FVector3f& Vertex = MeshVertices[VertexIndex].Position;
        PDI->DrawPoint(Position:FVector(Vertex), FColor::Red, PointSize:10.0f, DepthPriorityGroup:SDPG_World);
    }
```

- 2D Viewport 삼각형 렌더링

Mesh 의 Vertex 의 위치를 수정하거나 제거할 수 있도록 구현하기 위해, FDynamicMeshBuilder 를 사용해 Vertex 들을 관리했습니다.

실제로 2D 렌더링은 PDI 를 통해 수행합니다.
FCanvasItem, DynamicMeshBuilder, PDI 등 여러 방식이 존재했지만,

월드 좌표에 생성 되고, Viewport 를 이동했을 때 같이 움직이는 장점 때문에 선택하게 되었습니다.