

UE4 Landscape 修改

Landscape主要由 高度、Layer(混合比)、材质等数据组成。此文我们重点讨论如何快速修改高度与混合比。

一.地形创建

1.创建地形与创建普通Actor并没有区别，通过SpawnActor方法实现,创建时可以指定 Transform.

```
LandscapeMaskActor = World->SpawnActor<ALandscape>(NewLandscapeLocation, NewLandscapeTransform, this);
```

2.设置材质

```
FString NewLandscapeMaterialName = TEXT("/Game/Materials/PCG_LandscapeMask.PCG_LandscapeMaterial");  
ULandscapeMaterial* NewLandscapeMaterial = LoadObject<ULandscapeMaterial>(NULL, *NewLandscapeMaterialName,  
LandscapeMaskActor->LandscapeMaterial = NewLandscapeMaterial;
```

3.导入高度、Layer数据, 指定地形范围及Sections设置。

```
LandscapeMaskActor->Import(FGuid::NewGuid(), MinX, MinY, MaxX, MaxY, NumSubsections, HeightData, LayerData, bUseHeightData, bUseLayerData);
```

4.设置LayerObject

只有设置了 ULandscapeLayerInfoObject 后，地形的Layer数据才可以被修改，地形才能正常显示。

ULandscapeLayerInfoObject可以提前创建保存为资源，被多个地形重复使用。

```
void AssignLayerInfoObjects(ALandscapeProxy* Landscape, int32 LayerIndex)  
{  
    FSoftObjectPath AssetPath(FString::Printf(TEXT("/Game/PCGPipeline/Mask/SpeedGame/Assets/LayerInfo/LayerInfo_%.2d", LayerIndex)));  
    FAssetData AssetData;  
    UAssetManager::Get().GetAssetDataForPath(AssetPath, AssetData);  
    UObject* Object = AssetData.GetAsset();  
    if (Object)  
    {  
        Landscape->SetLayerInfoObject(LayerIndex, Object);  
    }  
}
```

```

        ULandscapeInfo* LandscapeInfo = Landscape->GetLandscapeInfo();
        ULandscapeLayerInfoObject* LandscapeInfoObject = const_cast<ULandscapeLayer

        FLandscapeInfoLayerSettings& LayerSettings = LandscapeInfo->Layers[LayerInd
        LayerSettings.LayerInfoObj = LandscapeInfoObject;
        LandscapeInfo->CreateLayerEditorSettingsFor(LandscapeInfoObject);
    }
}

```

二.地形高度修改

1.格式与高度转换

地形高度数据一般又称高度图，每个顶点的高度由一个uint16来表示，UE4高度转换算法把uint16映射到 [-256, 256]. 然后再和 DrawScale 一起决定了地形的最终渲染效果。

代码实现可以参考

[LandscapeDataAccess.h](#) 中 GetLocalHeight 函数的实现

```

#define LANDSCAPE_ZSCALE    (1.0f/128.0f)
const int32 MaxValue = 65535;
const float MidValue = 32768.f;
FORCEINLINE float GetLocalHeight(uint16 Height)
{
    return ((float)Height - MidValue) * LANDSCAPE_ZSCALE;
}

```

2.获取高度法线数据

个人一般通过 [FLandscapeEditDataInterface](#) 类的 GetHeightDataFast 方法来获取高度及法线数据。其中法线数据是可选项，可以不获取。

```

TArray<uint16> HeightData;

FLandscapeEditDataInterface LandscapeEdit(InLandscape->GetLandscapeInfo());
LandscapeEdit.GetHeightDataFast(MinX, MinY, MaxX, MaxY, HeightData.GetData(), 0);

```

MinX, MinY, MaxX, MaxY 定义下获取高度的范围。以 MinX = 0, MinY = 0, MaxX =16, MaxY = 16] 为例，要获取的范围为 16x16, 共有17x17个顶点，自然最终返回的数据也是17x17共289个高度值。

对于多Level的World, 场景中会存在多个 ALandscapeProxy*, 所有的这些ALandscapeProxy会共用同一个ULandscapeInfo, 如果我们只想获取某一个ALandscapeProxy的高度数据, 可以通过其 LandscapeComponents变量获取范围,然后再通过范围获取高度数据。

```
bool GetLandscapeExtent(int32& MinX, int32& MinY, int32& MaxX, int32& MaxY, ALandscapeProxy* Proxy)
{
    MinX = MAX_int32;
    MinY = MAX_int32;
    MaxX = MIN_int32;
    MaxY = MIN_int32;

    // Find range of entire landscape
    for (const ULandscapeComponent* Comp : InLandscape->LandscapeComponents)
    {
        Comp->GetComponentExtent(MinX, MinY, MaxX, MaxY);
    }

    return (MinX != MAX_int32);
}
```

3.设置高度数据

相对应的, 我们通过 FPCGTerrainEditInterface 的 SetHeightData方法来设置高度数据。

FLandscapeEditDataInterface::SetHeightData 方法主要操作有:

1. 查找位于范围内的的所有LandscapeComponents
2. 根据传入的高度数据计算法线, 或者通过传入的法线数据更新法线。其中顶点法线计算方法为: 所有顶点相关的面法线求和取平均:

$$V_n = \frac{1}{n} \sum_{i=0}^n F_n(i)$$

i : 面索引

n : 顶点相关面数量

Vn : 顶点法线

Fn : 面法线函数

3. 更新高度数据与法线数据, UE4只用一个像素就存储了高度及法线数据, 高度因为精度要求为uint16分别占用了RG通道, 法线因为是单位向量, 所以用BA通道分别存储了法线的X,Y值, Z值由 $\sqrt{1-XX-YY}$ 求出。,(推测: 因为高度数据与地形位置的X,Z数据是分开保存的, 则用于地形绘制的索引及顶点位置的XZ数据应该是所有地形Component共用的, 只有一份)。

```
// Update the texture
TexData.R = Height >> 8;
TexData.G = Height & 255;

// Update normals if we're not on an edge vertex
FVector Normal = VertexNormals[NormalDataIndex].GetSafeNormal();
TexData.B = FMath::RoundToInt(127.5f * (Normal.X + 1.0f));
TexData.A = FMath::RoundToInt(127.5f * (Normal.Y + 1.0f));
```

3. 更新包围盒
4. 所有组件创建高度MipMaps(LOD)
5. 更新碰撞数据(物理), 如果碰撞数据修改影响寻路, 则更新寻路数据

警告

SetHeightData 函数只修改了非边缘顶点的法线, 如果要更新当前范围内所有顶点的法线, 必须扩大传入的范围。

地形Layer数据修改

获取Layer数据

Layer数据的获取与高度数据的获取类似, 区别主要有几点

1. Layer数据格式为 uint8, 高度数据格式为 uint16
2. 每个地形拥有多层Layer, 所以GetWeightDataFast 函数需要 ULandscapeLayerInfoObject* 对象做为输入。

```
// Get LayerData
bool GetLandscapeLayerData(int32& MinX, int32& MinY, int32& MaxX, int32& MaxY, TArray<
{
    ULandscapeInfo* LandscapeInfo = InLandscape->GetLandscapeInfo();
    ULandscapeLayerInfoObject* InfoObject = LandscapeInfo->GetLayerInfoByName(*Layer
    if (InfoObject)
    {
        MinX = MAX_int32;
        MinY = MAX_int32;
        MaxX = -MAX_int32;
        MaxY = -MAX_int32;
        if (GetLandscapeExtent(MinX, MinY, MaxX, MaxY, InLandscape->LandscapeCompon
        {
            FLandscapeEditDataInterface LandscapeEdit(InLandscape->GetLandscapeInfo
            LayerData.AddZeroed((MaxX - MinX + 1) * (MaxY - MinY + 1));
            LandscapeEdit.GetWeightDataFast(InfoObject, MinX, MinY, MaxX, MaxY, Lay
```

```

        return true;
    }
}

return false;
}

```

3. 对于WeightBlendLayer，每个顶点相关的所有Layer值相加为1，各Layer间形成一种此消彼长的关系。当某Layer数据增加时，其它Layer数据会根据当前值按比例缩小。

```
Weight = FMath::Clamp<uint8>(FMath::RoundToInt((float)(255 - NewWeight) * (float)We
```

设置Layer数据

Layer数据的修改通过 FLandscapeEditDataInterface 的方法 SetAlphaData来实现。

```

void OverrideLandscapeWeightData(int32 MinX, int32 MinY, int32 MaxX, int32 MaxY, co
{
    if (InLandscapeInfo && InLandscapeInfoObject)
    {
        FLandscapeEditDataInterface LandscapeEdit(InLandscapeInfo);
        LandscapeEdit.SetAlphaData(InLandscapeInfoObject, MinX, MinY, MaxX, MaxY, W
    }
}

```

FLandscapeEditDataInterface::SetHeightData 方法主要操作有：

1. 若对应LayerIdx的贴图不存在，则创建新的WeightTexture。
2. 调整其它层Weight，保证所有层数值相加为255。着色器中会映射[0-255]为[0-1]
3. 更新WeightmapMips(推测为地形LOD 渲染时，提高纹理采样效率，降低带宽占用)
4. 更新Layer相关的物理数据; 物理材质是与Layer绑定的，如摩擦力等。
5. 清空Layer，若某Layer所有数据全为0，则删除WeightTexture。

更新植被显示

除了FoliageSystem 外，UE4中还可以通过材质配置草的渲染，其中草的生成依赖于地形Layer数据。在地形Layer数据修改后，需要调用以下代码更新草的显示。

```
// Invalid Component Data
TSet<ULandscapeComponent*> ComponentsSet(LandscapeActor->LandscapeComponents);
ALandscapeProxy::InvalidateGeneratedComponentData(ComponentsSet);
```

四.性能优化

UE4中地形修改，其性能瓶颈主要有两点

1. HeightmapTexture的创建
2. WeightmapTexture的创建与删除

1. 分块修改

对于地形的修改，大部分情况下，我们只修改了一部分数据的数据。以1009x1009地形为例，以16x16块大小进行分块，则可以得到63 X 63 共 3969 个Tiles.

如果一条河只影响了其中的63个Tile, 那么只有 1.6% 左右的地形顶点需要通过调用 SetHeightData 与 SetAlphaData进行数据修改。

性能数据

```
TimeProfiler FPCGCookLinker::ApplyLayers+PCGCookLinker.cpp"+108 ,38.537473s, 99.999
TimeProfiler FPCGTerrainSyncHelper::ImportLayers+PCGTerrainSyncHelper.cpp"+300
TimeProfiler FPCGTerrainSyncHelper::AddLayerWeight+PCGTerrainSyncHelper.cpp"+34
TimeProfiler FPCGTerrainSyncHelper::AddLayerWeight+PCGTerrainSyncHelper.cpp"+35
TimeProfiler FPCGTerrainSyncHelper::AddLayerWeight+PCGTerrainSyncHelper.cpp"+35
```

单Level应用一层纹理耗时约等于 8秒,

纹理加载、数据计算、InvalidComponentData 对性能影响较低; 主要的耗时函数为 FLandscapeEditDataInterface::SetAlphaData, 耗时占比超过98%。

分块后 应用River_Layer, 耗时在1秒以内。

Tile大小对效率的影响：

单Level, Layer修改总耗时

8x8 7.996s

16x16 4.373s

32x32 4.634s

64x64 6.828s

其中 16x16是比较好的选择

2. 避免使用 FillLayer

FillLayer会把当前Layer置1，导致其它Layer的所有WeightmapTexture被删除，严重影响编辑效率。

3. 并行计算

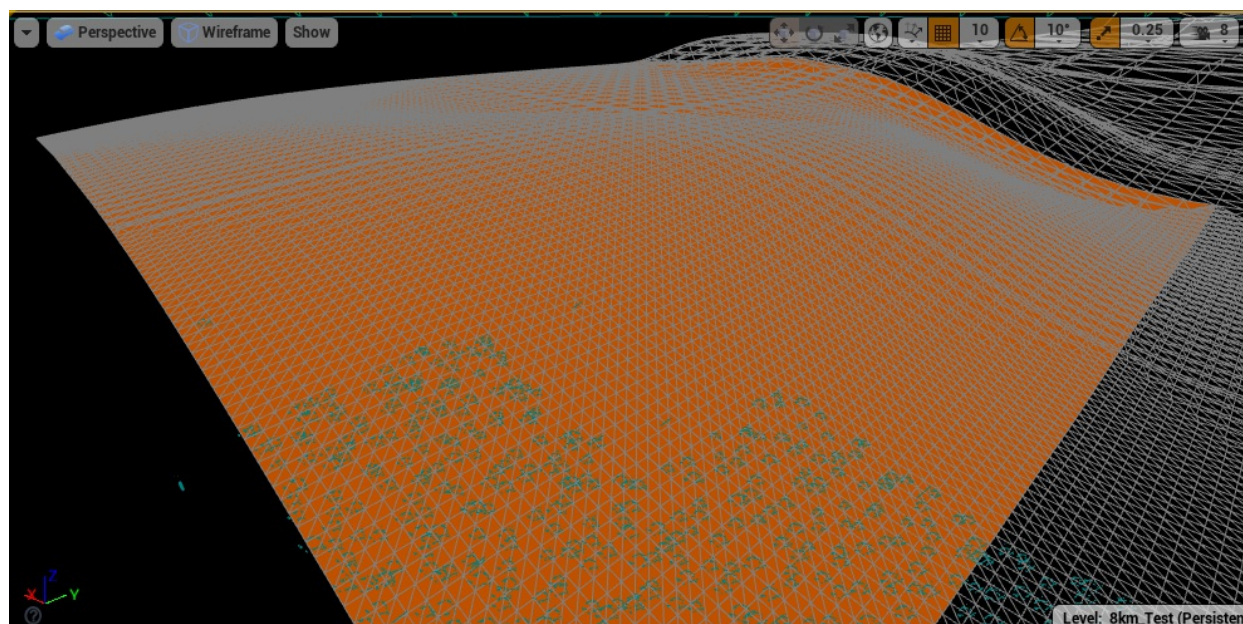
对数据的分块操作以及各种计算，其数据都是只读的，且不相互影响。可以使用ParallelFor 进行多线程并行处理，提升效率。

4. 异步创建 Actor（未实现）

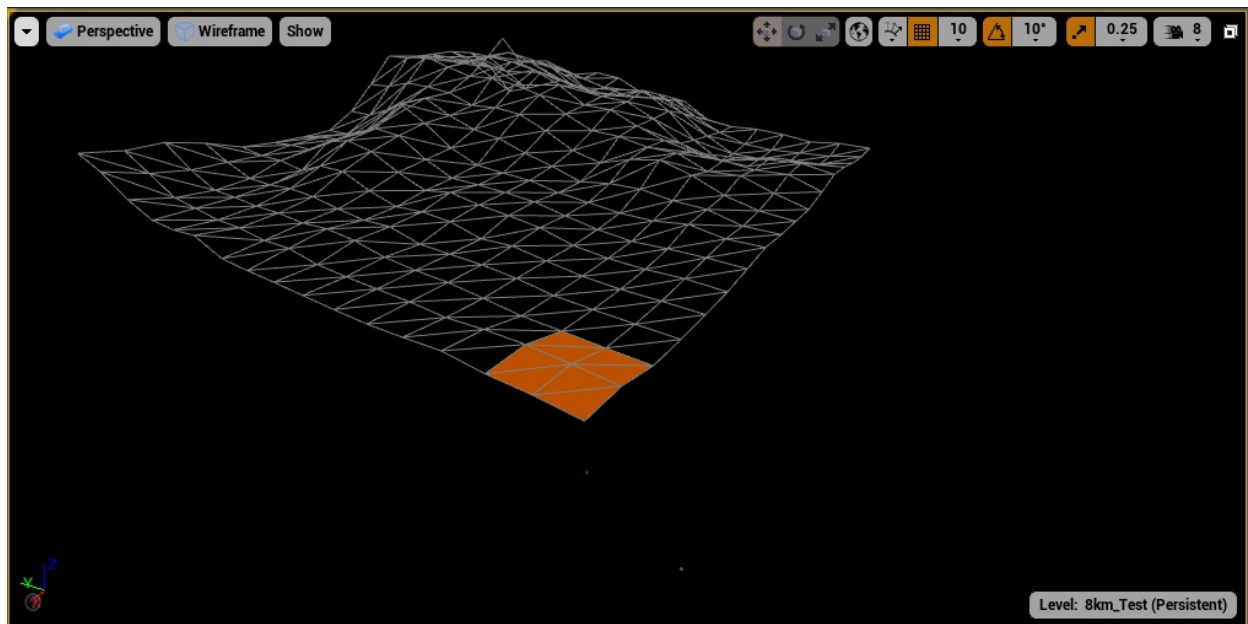
目前用于Mask编辑的地形创建耗时较长，可以尝试异步方式创建Actor,避免阻塞主线程。

待续

1. 地形LOD的实现细节, 最低级LOD面数为最高级高LOD面数的 $1/3969$.



每个Component有4个SubSection组成，在最高LOD下，每个SubSection由63x63x2个三角形面(黄色区域是一个Compont).



每个Component有4个SubSection组成，在最低LOD下，每个SubSection由1x1x2个三角形面(黄色区域是一个Component)

2. 高度数据(HightmapTexture)与Layer数据(WeightmapTextures)在渲染中的使用细节

参考

1. Epic Games (1998-2019) . [Landscape Outdoor Terrain](#)