肖添翼 作品集

游戏方向作品集 2023.7 （基本信息请见简历）

目录

1 **Unity 效果**

基于Unity Shader等功能实现的各类效果，多数与天气相关

2 **课程项目**

基于C++、OpenGL、Qt实现的两个项目

3 **Unity/UE 游戏**

多个使用Unity/UE引擎制作的3D游戏

4 **Maya插件**

多个基于Mel及C++ Maya API实现的Maya插件

5 ***其他***

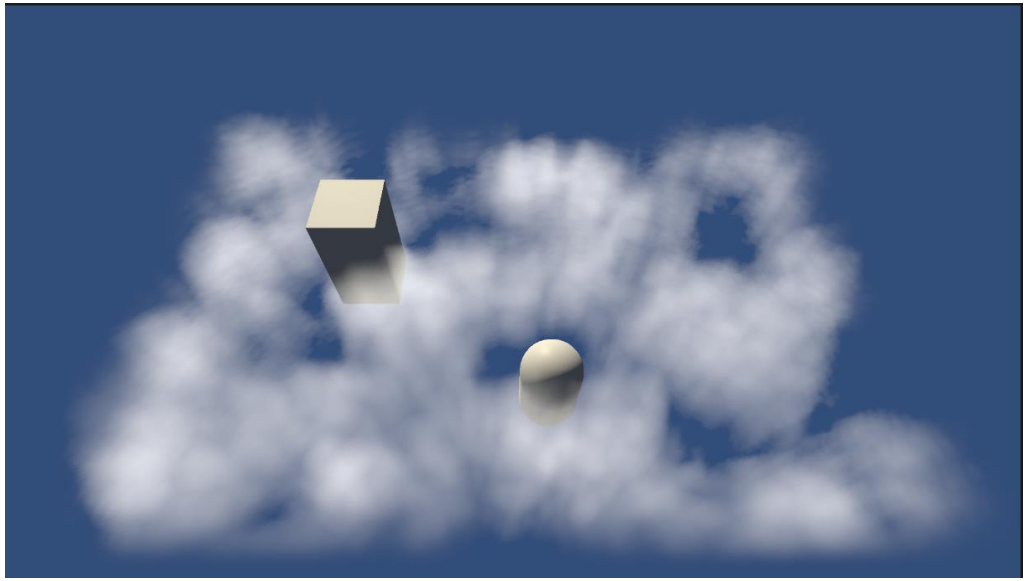
包括光线追踪离线渲染器，Maya建模作品，基于PBR流程的渲染作品等

1

Unity 效果

基于Built-in管线在Unity实现了多种效果，
包括但不限于各类天气相关效果

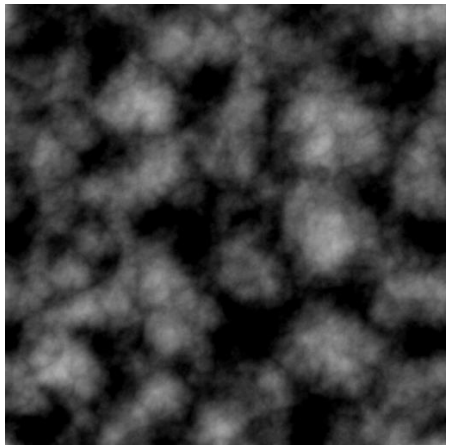
1. 体积云

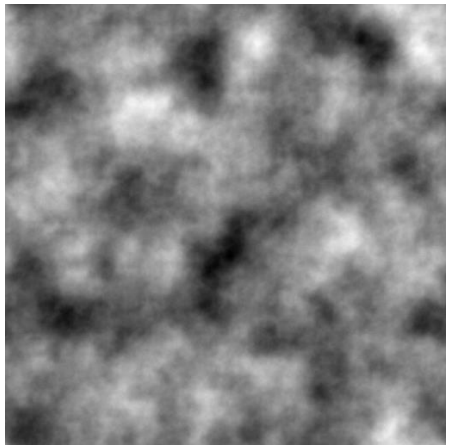


基于后处理渲染：

- 对从相机射向近裁剪平面四角的四条射线进行插值，在片元着色器中获得每个片元对应的视射线
- 判断射线是否与体积云的包围盒相交，并根据深度贴图判断是否云雾前有物体遮挡
- 如通过测试，则进行RayMarching，在射线与云包围盒相交的线段上做多次步进，采样每个采样点的密度和光照颜色并相乘，得到采样点颜色，累计相加得到最终结果
- 将云层颜色与原本场景颜色相加，得到最终渲染结果

1. 体积云





（接上文）

密度采样：

- 根据2D Perlin-Worley噪音贴图(compute shader计算生成)，决定云朵的水平形状，并结合采样点高度与噪音贴图，实现云朵高层疏底层密的效果；
- 同时根据采样点与包围盒xz平面边缘的距离，降低云密度，避免包围盒边缘的云过于突兀
- 额外再采样噪音贴图，增加云层细节

光照采样：

- 从采样点向阳光方向做步进，采样并累计得到与阳光之间的云密度总和；从而计算云层的透明度，并根据参数计算光照颜色

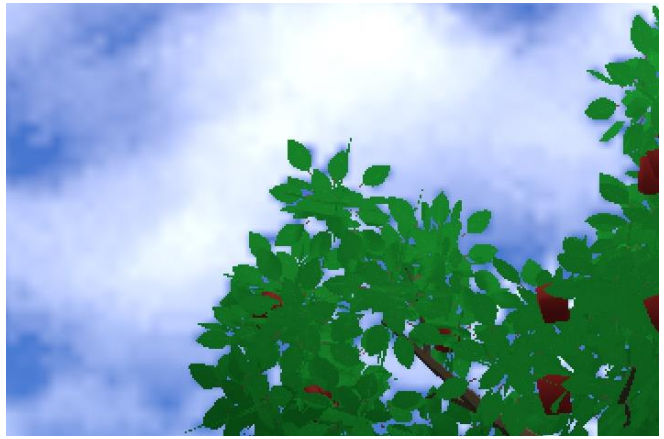
1. 体积云 – 问题与优化

问题1：步进次数较少时，会因为误差导致云分层明显，但较高的步进次数又会造成极大的性能开销

优化：

- 1. 采样云朵形状噪音贴图之前，先采样蓝噪音贴图，并根据结果移动uv，通过增加噪点来减少分层情况
- 2. 将云朵结果down sample，渲染到小尺寸的RenderTexture中，再输出到屏幕上，实现简洁的模糊效果

1. 体积云 – 问题与优化



问题2：将体积云加入场景中后，场景中与云朵重叠的部分出现明显的锯齿

优化：

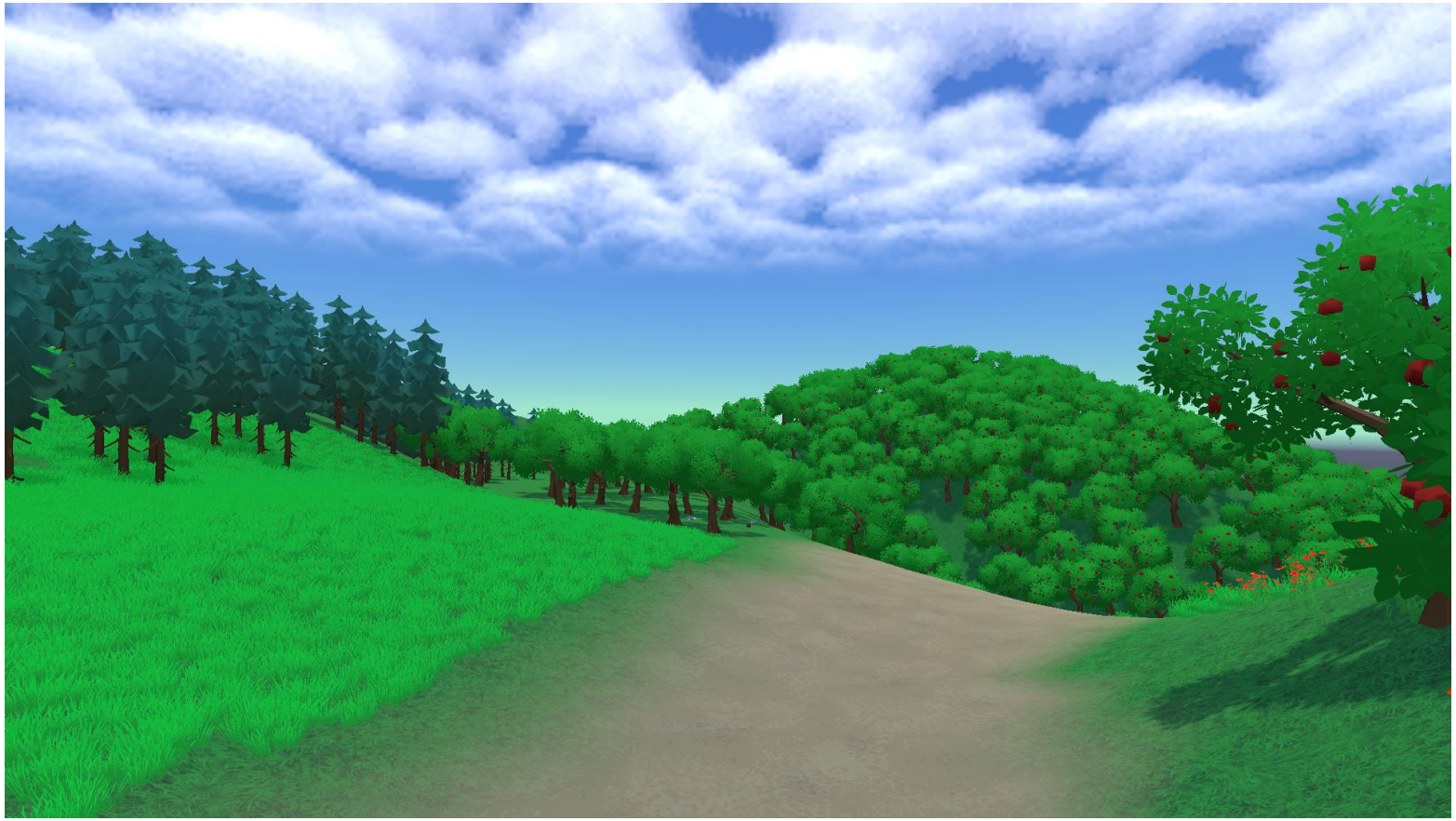
- 检测后，发现是由于down sample所导致，down sample中场景物体与云朵一同模糊。
- 将云与场景的混合放到额外的pass中。先渲染云朵图案并进行模糊处理，降低分层效果，再用一个pass比较depth与云盒距离，判断场景是否遮挡云朵。

问题3：云朵整体形状呈方形，不够自然

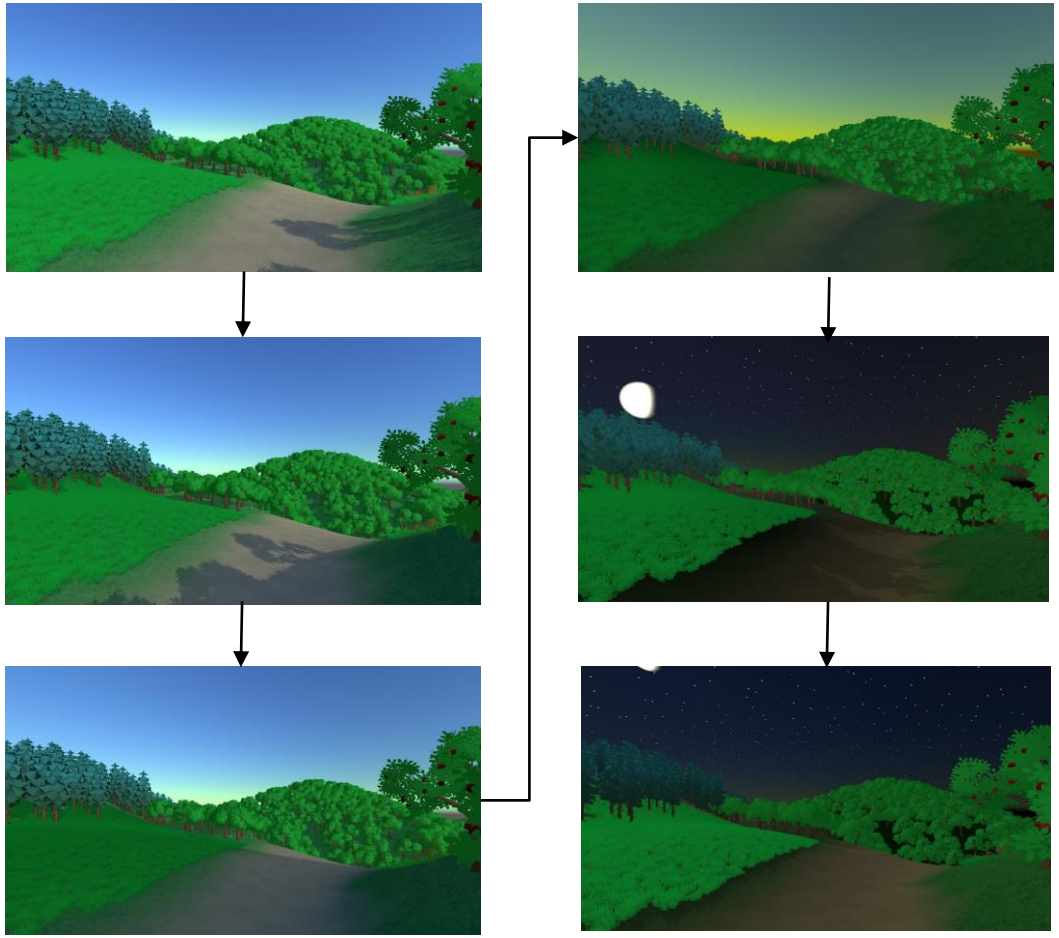
优化：

- 根据射线离云盒距离时，使云朵逐渐消失，不再根据云盒边缘情况

1. 体积云 – 最终效果



2. 昼夜变换-天空盒&光照阴影



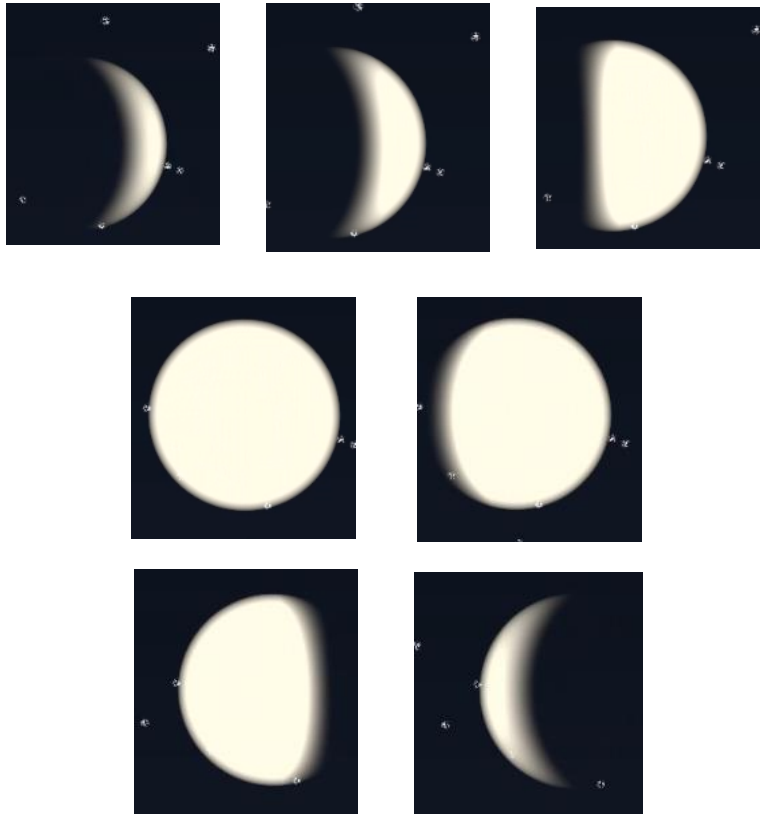
天空盒：

- 基于Unity官方天空盒中的大气散射算法，通过输入参数计算光的波长，根据散射原理朝太阳光方向做步进，累计得到天空盒颜色。每次步进时根据视射线角度、步进距离与太阳光角度计算散射强度，进而得到该采样点的光照颜色。

光照与阴影：

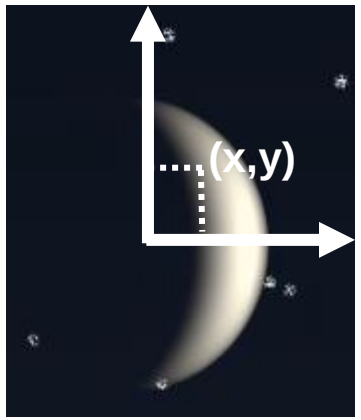
- 为保证从白天转换到夜晚时，阴影及场景接收到的光照平滑过渡，在场景中放置两个方向相反的平行光源，并用脚本控制两个平行光在昼夜交接时光照强度及阴影强度的变化，使得阴影贴图能正确渲染，光照能被正确接收。

2. 昼夜变换-月相变换



月相变换：

- 基于月光的方向，确定月球中心的位置，并将视射线从世界空间变换到月球空间下。
- 根据月相参数构造SDF，通过计算在月球空间下距离原点的水平距离和垂直距离得到决定月球形状的SDF，继而根据SDF实现月亮边缘从暗到亮的光滑过渡。



当月亮为左开口弯月时:

sdf =m * sqrt(r^2 – y^2) - x

sdf小于0时采样点在月牙内部，r为月球半径，m控制月牙的粗细且>0，由月相参数决定

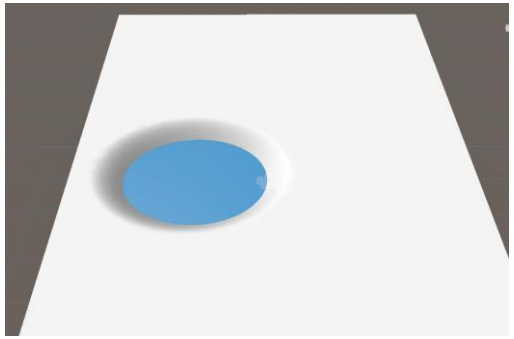
2. 昼夜变换-月相变换

星空效果:

- 将图片均匀分格，在每个小格内随机选取一个点作为星星出现的位置，从而绘制星空贴图。通过分格的大小控制星空的疏密，同时保证星星分布相对均匀。

- 渲染时，根据uv和时间计算星星的亮度，实现星空的闪烁效果

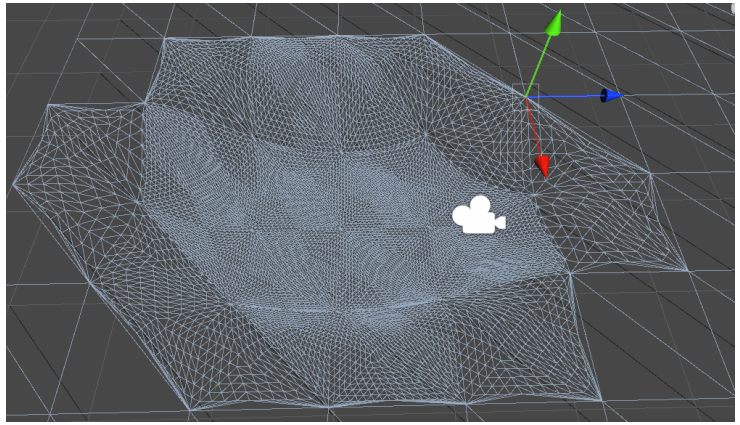
3.雪地轨迹-凹陷效果



（用椭圆代替深度图进行测试）

雪地凹陷效果：

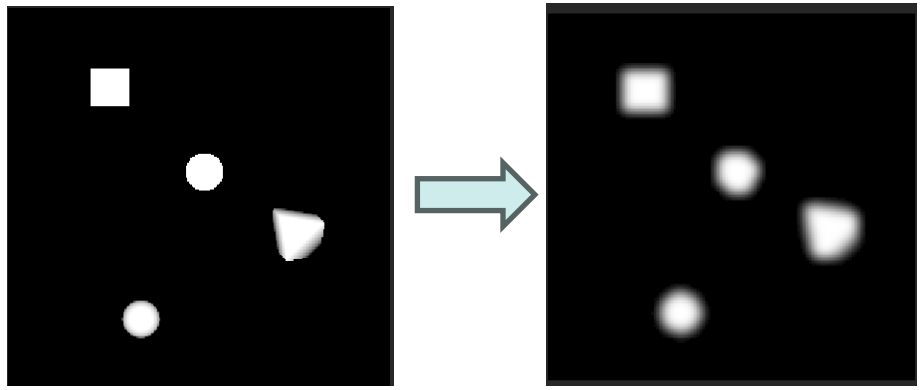
- 利用曲面细分着色器，根据雪面深度将平面进行不同程度的细分，再移动顶点位置，保证过渡的平滑，并采样周边点位的深度代替求导，估算切线与副切线，最终得到法线方向。

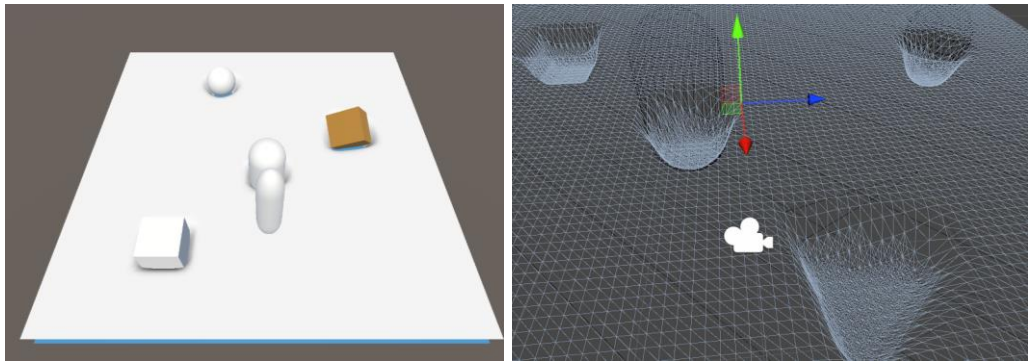


```
TrianglePatchTess ComputeTessFactor(InputPatch<v2h, 3> patch, uint patchId : SV_PrimitiveID)
{
    TrianglePatchTess output;
    float depth0 = getDepth(patch[0].uv);
    float depth1 = getDepth(patch[1].uv);
    float depth2 = getDepth(patch[2].uv);
    output.edgeTess[0] = max(int(_TessDivisionFactor * max(depth1, depth2)), 1);
    output.edgeTess[1] = max(int(_TessDivisionFactor * max(depth0, depth2)), 1);
    output.edgeTess[2] = max(int(_TessDivisionFactor * max(depth0, depth1)), 1);
    output.insideTess = max(int(_TessDivisionFactor * max(max(depth0, depth1), depth2)), 1);
    return output;
}
```

```
float uvDiff = 0.001f;
float3 tan = float3(2 * ratio * uvDiff, getDepth(uv + float2(uvDiff, 0)) - getDepth(uv - float2(uvDiff, 0)), 0);
float3 bit = float3(0, getDepth(uv + float2(0, uvDiff)) - getDepth(uv - float2(0, uvDiff)), 2 * ratio * uvDiff);
```

3.雪地轨迹-雪地深度图





雪地深度图的获取：

- 在雪面底部设置正交相机，覆盖雪面范围。调整正交相机的近裁剪平面与地面重合，远裁剪平面与雪面重合，使得正交相机的深度图与雪面深度匹配，利用后处理使正交相机渲染出深度图。

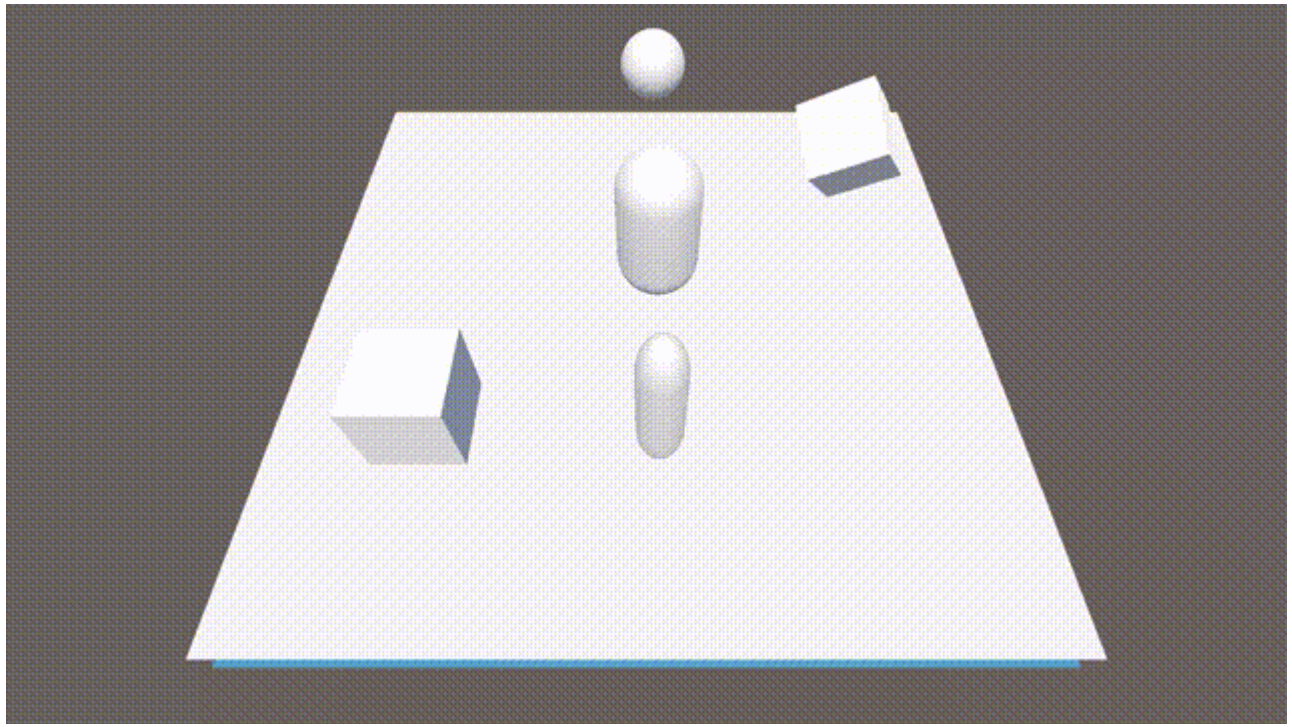
优化：

- 当前深度图黑白过渡过于尖锐，不够自然。因此，利用down sample与高斯模糊，使深度图的黑白边界平滑过渡。

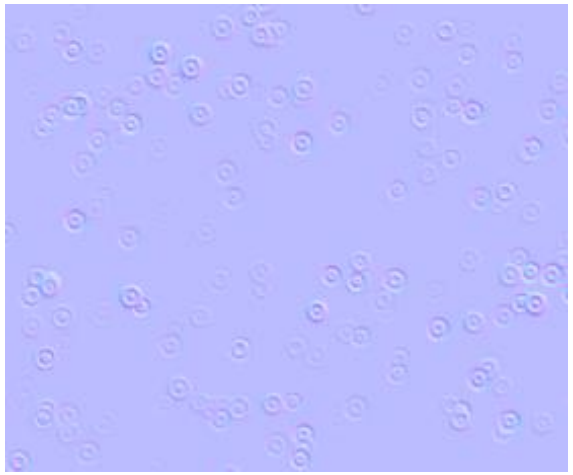
保留轨迹：

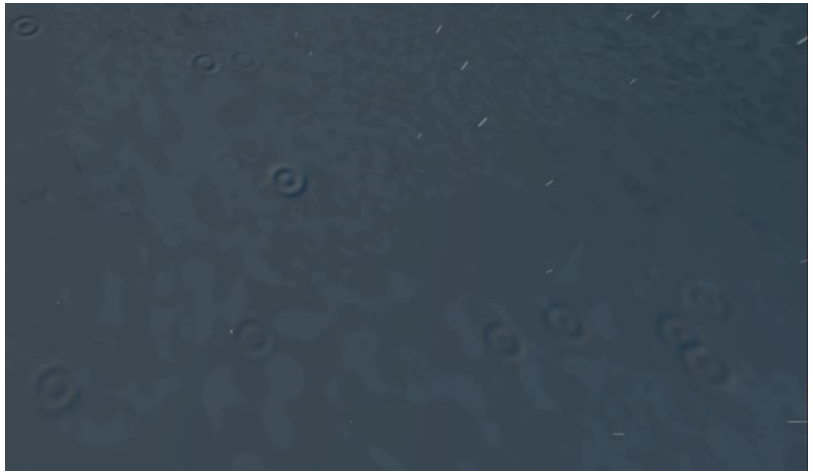
- 模糊后额外用一个pass，取当前深度图和上一帧深度图的最大值，从而保留雪地轨迹，同时避免将上一帧深度图结果再次模糊，从而产生误差。

3.雪地轨迹 – 最终效果



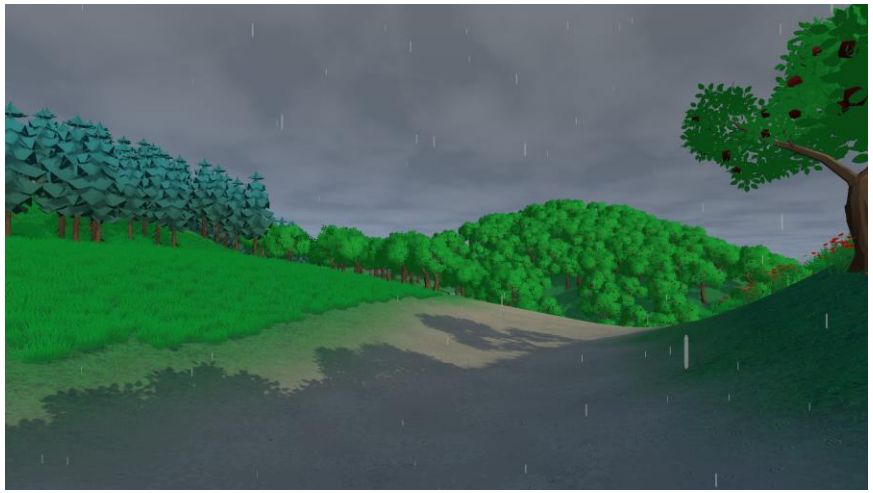
4.雨天效果-地面雨滴





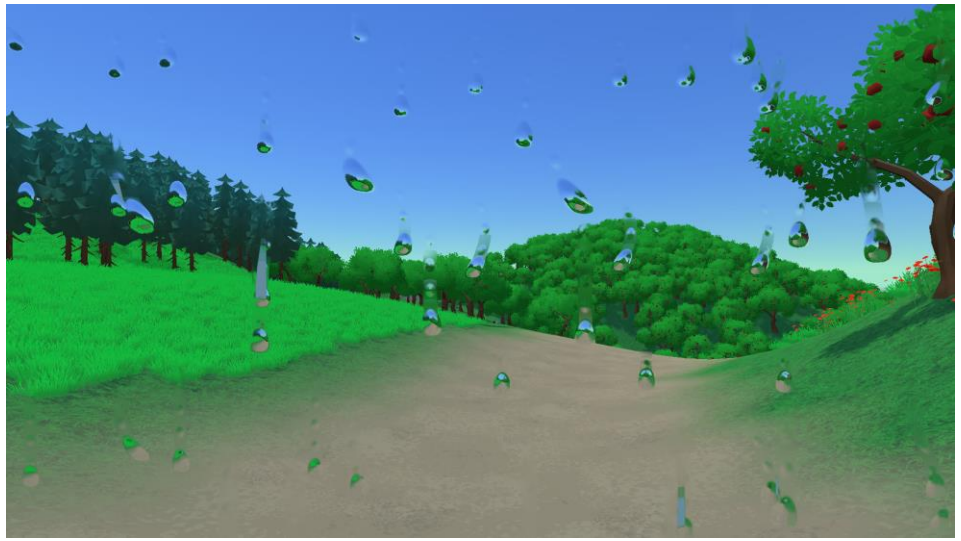
- 通过ComputeShader，更新地面上雨滴坐标变换矩阵和当前雨滴经过的时间，将矩阵和时间通过ComputeBuffer传出。
- 通过CommandBuffer.DrawMeshInstanced(),传入从buffer中获取的变换矩阵数组，从而在构造好的mesh上绘制雨滴地面normal texture。
- 绘制时，在shader中根据instanceID获取每个雨滴的相应时间，从而根据时间实现雨滴波纹的扩散效果；并根据时间控制输出颜色的透明度，进而通过透明度混合，让波纹随时间逐渐消逝。

4.雨天效果 – 雨滴及贴图云



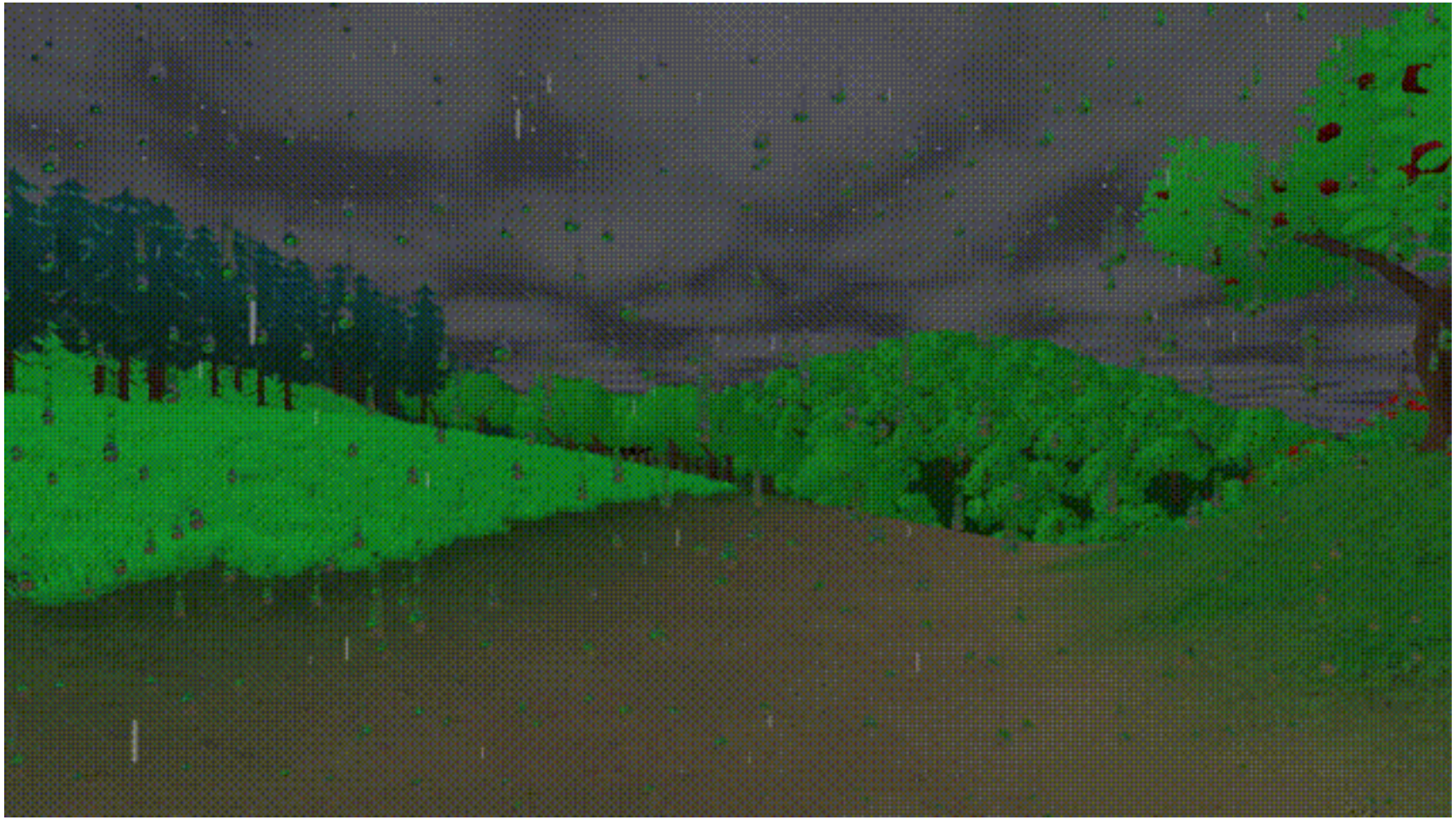
- 通过粒子效果，制作空中的雨滴效果，并使用雨滴的贴图代替默认的粒子贴图。
- 在天空盒上根据视角射线方向计算uv，读取云朵贴图。根据云雾浓度控制星星是否能透过云层。

4.雨天效果 – 屏幕雨滴

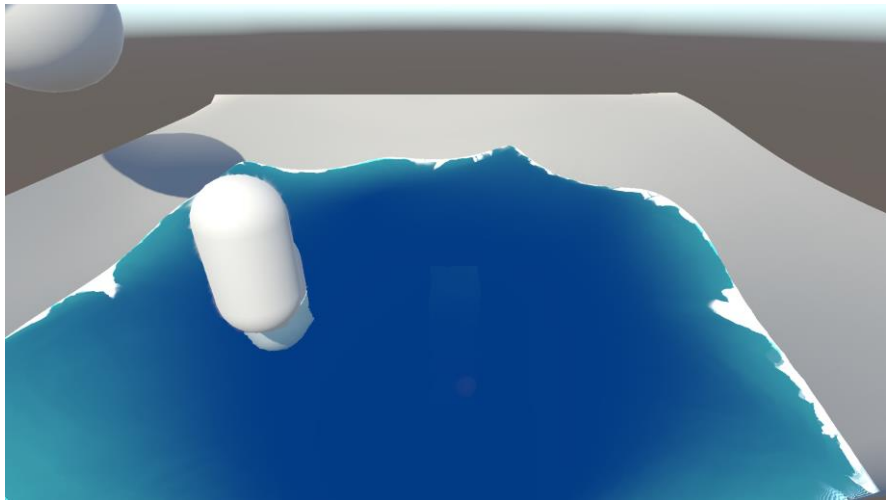


- 在后处理中，通过将屏幕分格，在每个格子内部随机选取雨滴位置，继而决定雨滴拖尾效果以及雨滴拖尾带来的屏幕雾效果。
- 根据效果强度，在采样主纹理时进行uv便宜，并通过采样mipmap直接实现雨滴拖尾雾的模糊效果。

4.雨天效果 -最终效果

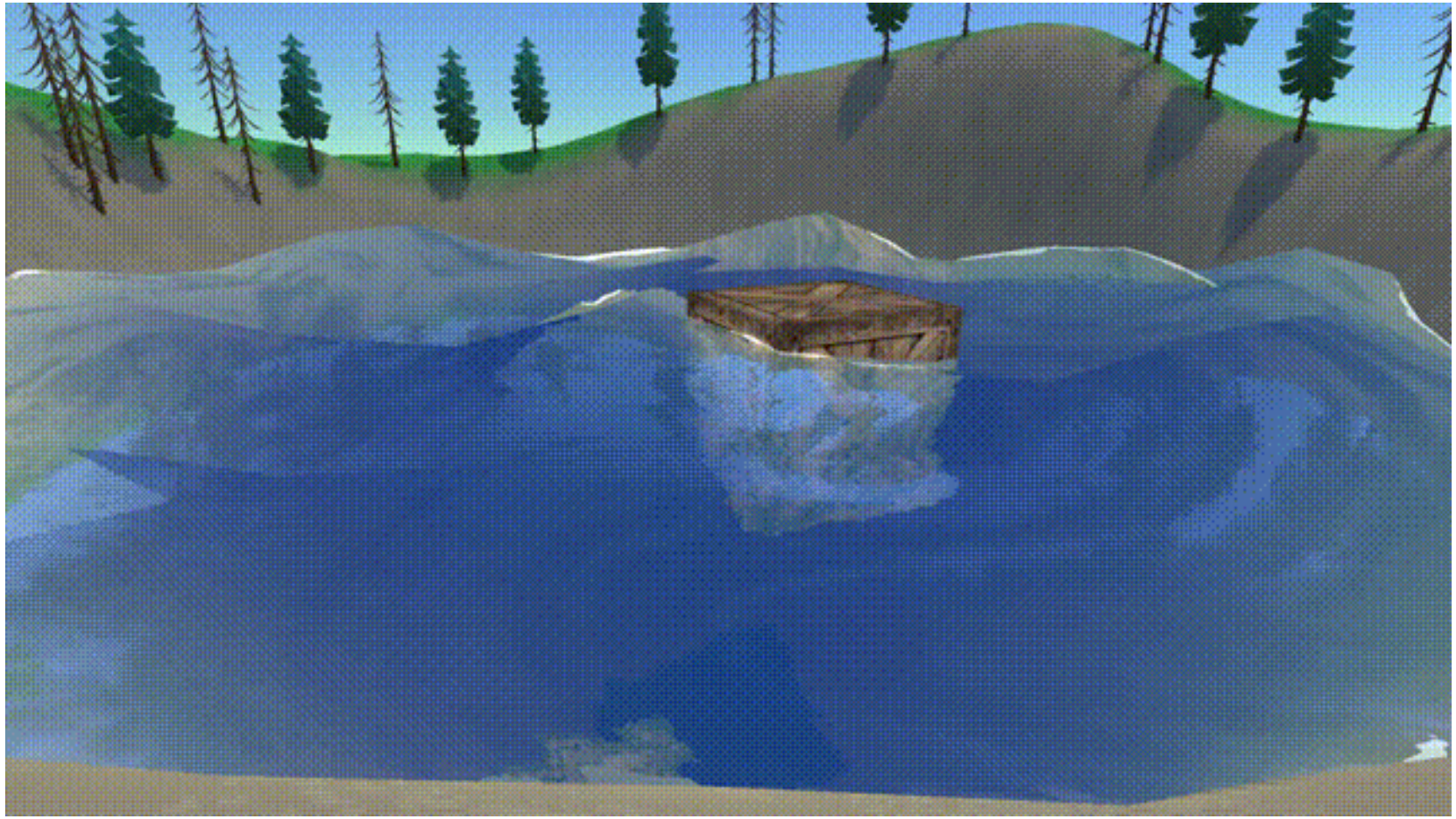


5. 水面波动及渲染

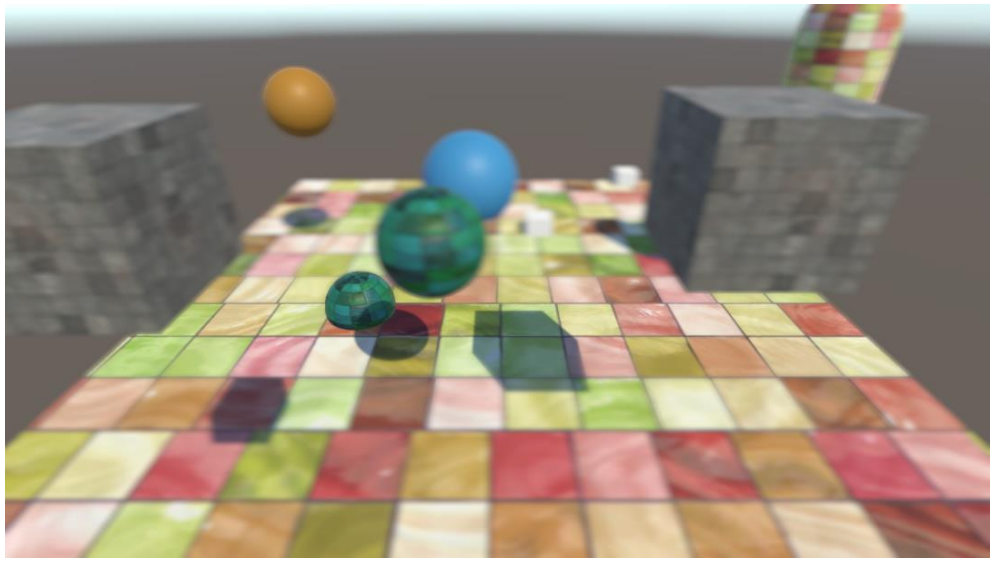


- 在顶点着色器中，通过叠加多个Gerstner波，实现水波顶点动画，并重新计算每个顶点的法线及切线。
- 根据流动贴图和normal贴图计算normal，继而计算光照和折射及反射方向。根据折射方向偏移uv，读取偏移后的水面深度，来决定水面颜色及水面透明度。
- 同时根据折射方向偏移屏幕坐标，读取GrabPass贴图，获取折射颜色。若偏移后所得深度小于水面到相机的距离，说明水面上方物体被折射，则无需偏移。
- 同理，从反射相机的RenderTexture中读取反射颜色。根据菲涅尔项控制反射强度。
- 根据深度判断水面边缘，在边缘处实现浮沫效果，并采样噪音贴图使浮沫

5. 水面波动及渲染 – 最终效果

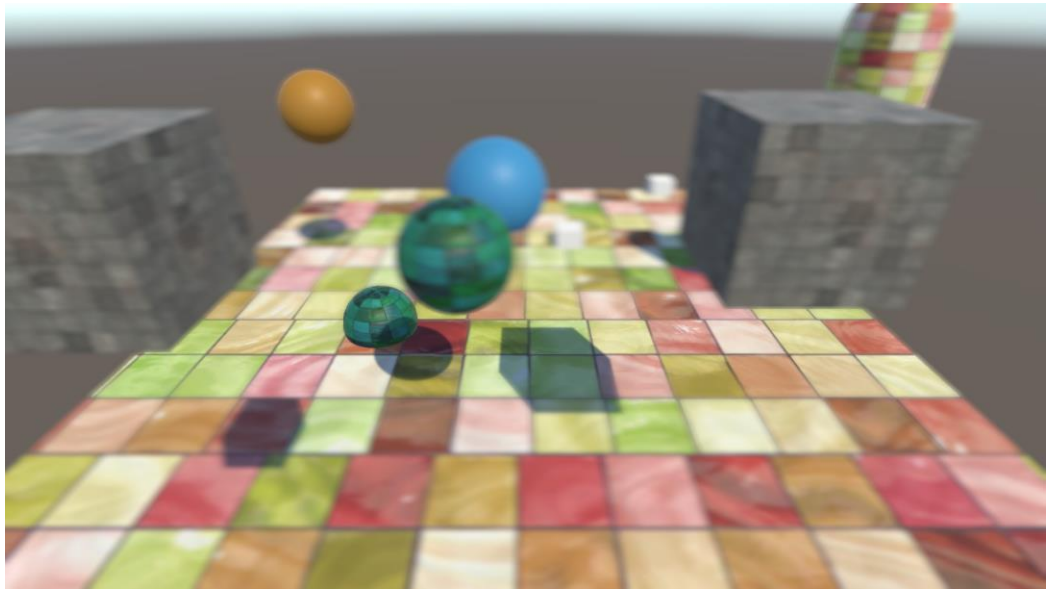


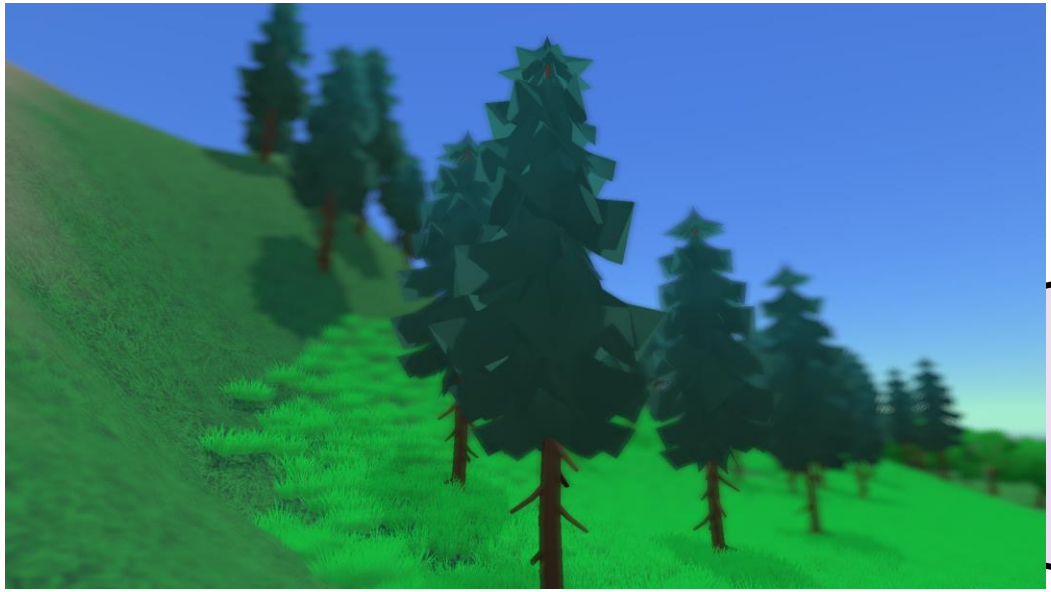
6.后处理景深模糊



- Pass0：根据深度贴图和参数(模糊范围，模糊距离等)输出模糊贴图。
- Pass1：读取模糊贴图，在圆盘范围内的预设采样点采样，分别计算前景和后景的模糊程度和模糊颜色，防止前后景一同混合。
- Pass2：将模糊结果进一步做高斯模糊，使模糊效果更为自然美观
- Pass3：将Pass2得到的模糊结果与主帖图进行混合，混合比重根据前后景的模糊程度和模糊贴图决定，得到最终效果。

6.后处理景深模糊



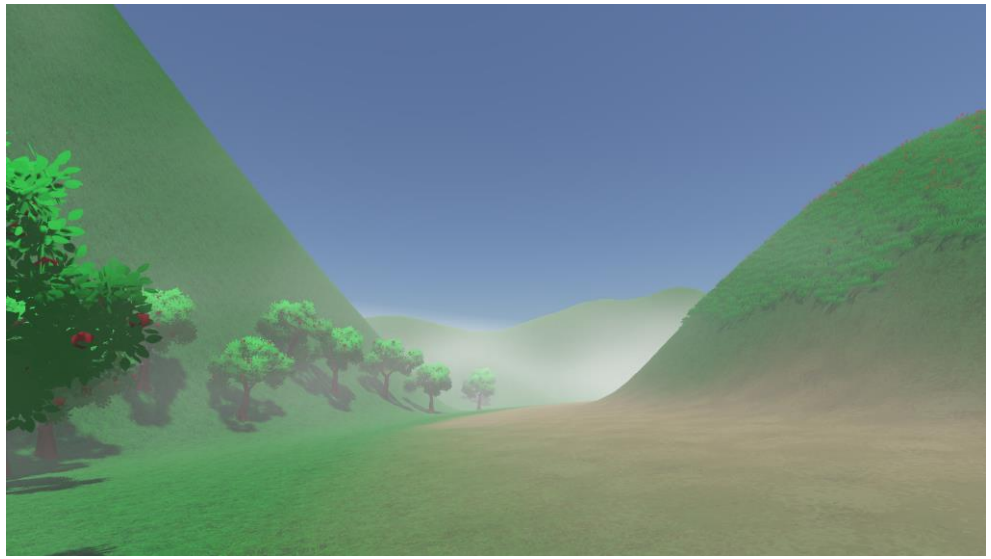


7. 其他效果

风吹树叶

树叶随风摆动, 利用Perlin噪音保证树叶摆动效果错落自然

后处理雾



根据深度图获取高度和深度，继而根据高度和深度控制雾的浓度。

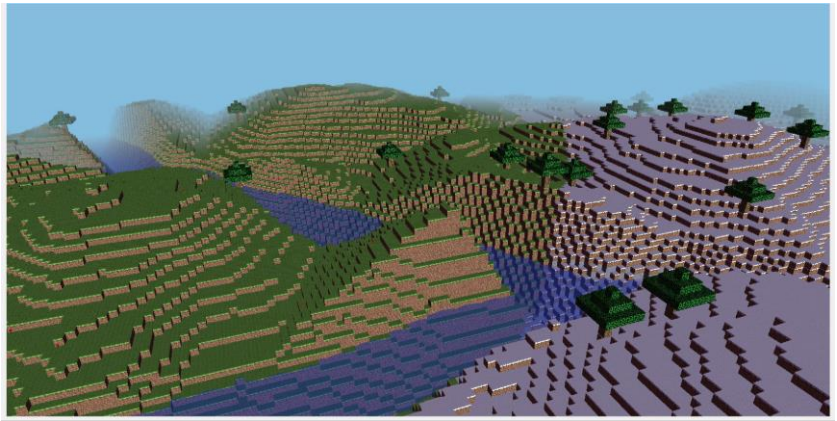
2

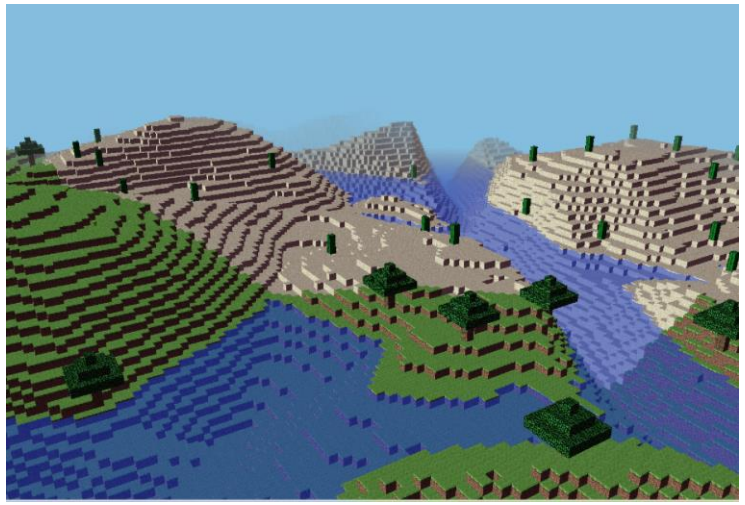
课程项目

Mini-Minecraft & Micro-maya

使用C++，Qt及OpenGL实现

1. Mini-Minecraft



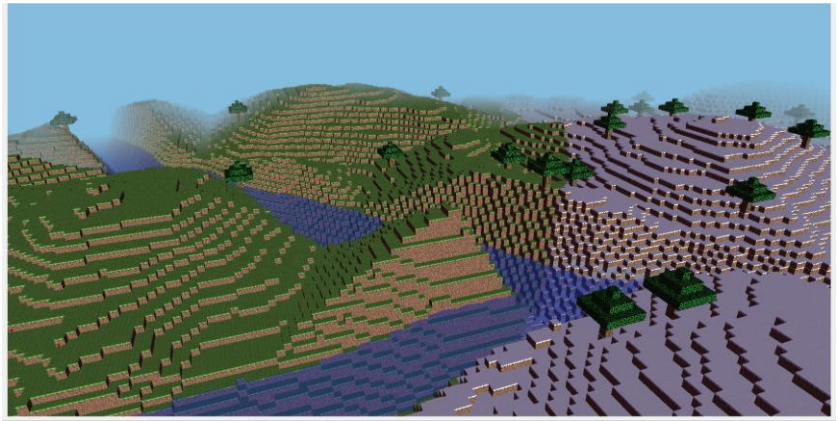


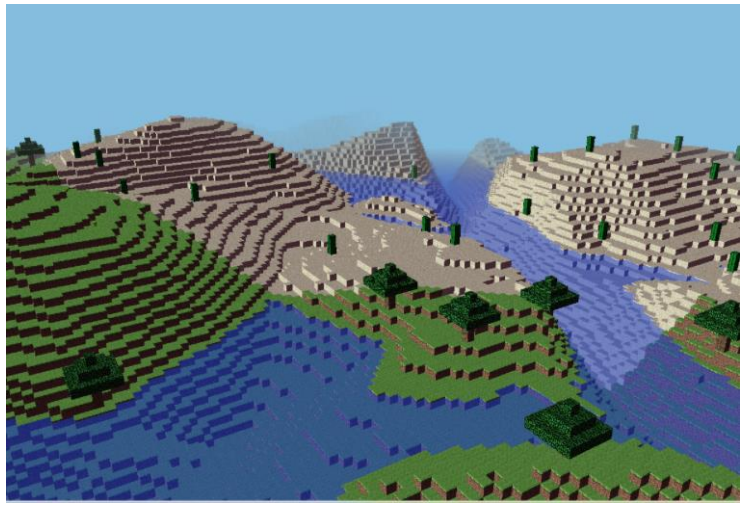
多线程随机化地形生成：

- 将屏幕内可见地形划分为多个16x16的区块，每个区块调用一个线程来生成地形；从而利用多线程实现异步生成地形，防止卡机，同时加快生成速度

- 根据xz坐标，利用worley及perlin噪音函数，计算每一个地点的湿度值以及温度值，从草地，沙漠，山地，雪地中选取一种地形类型。同时计算各个类型对应的地形高度，再根据湿度值及温度值做插值，保证不同地形之间高度分布方式不同，且高度差能够平滑过渡。

1. Mini-Minecraft



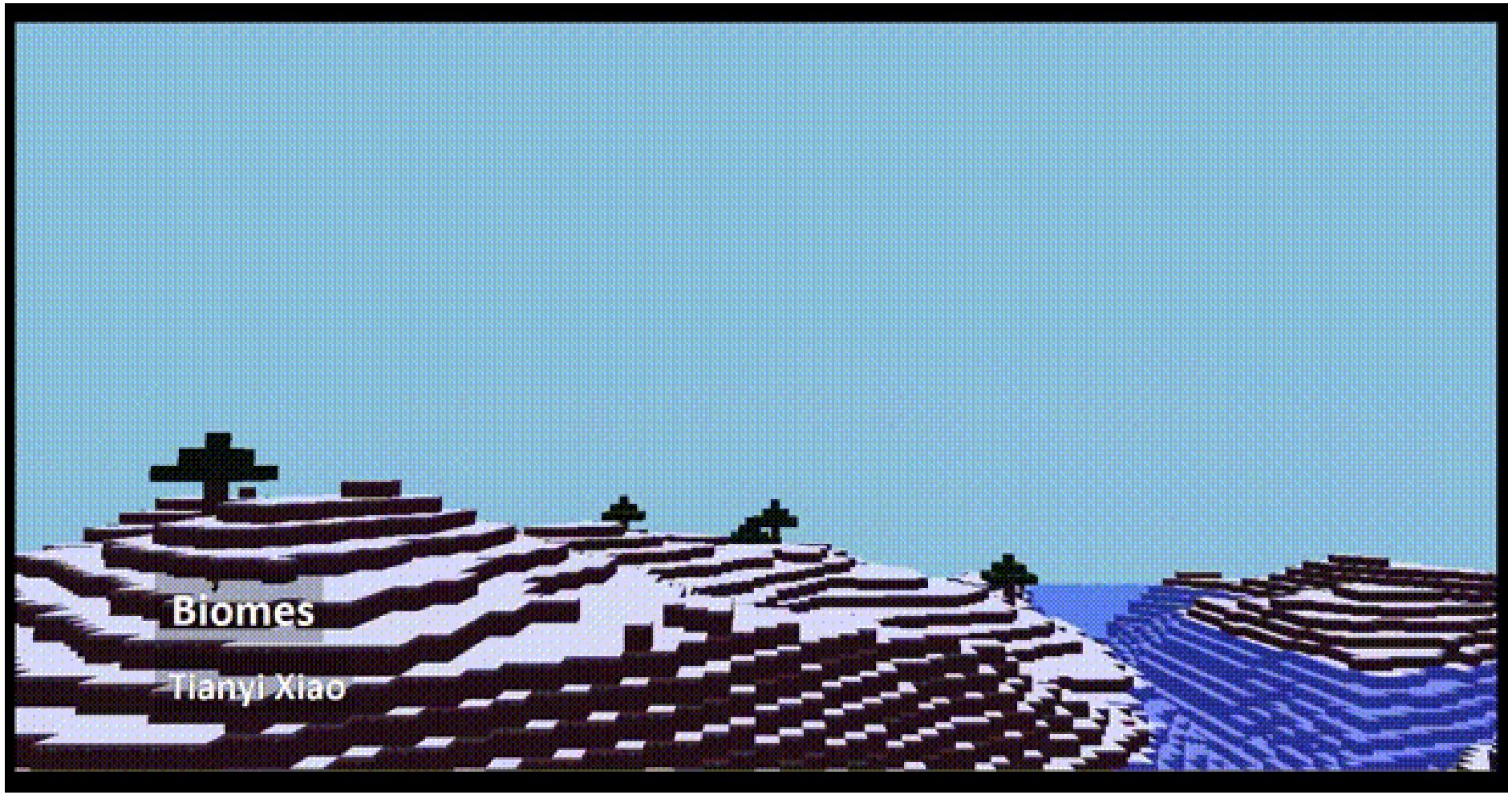


河流生成：

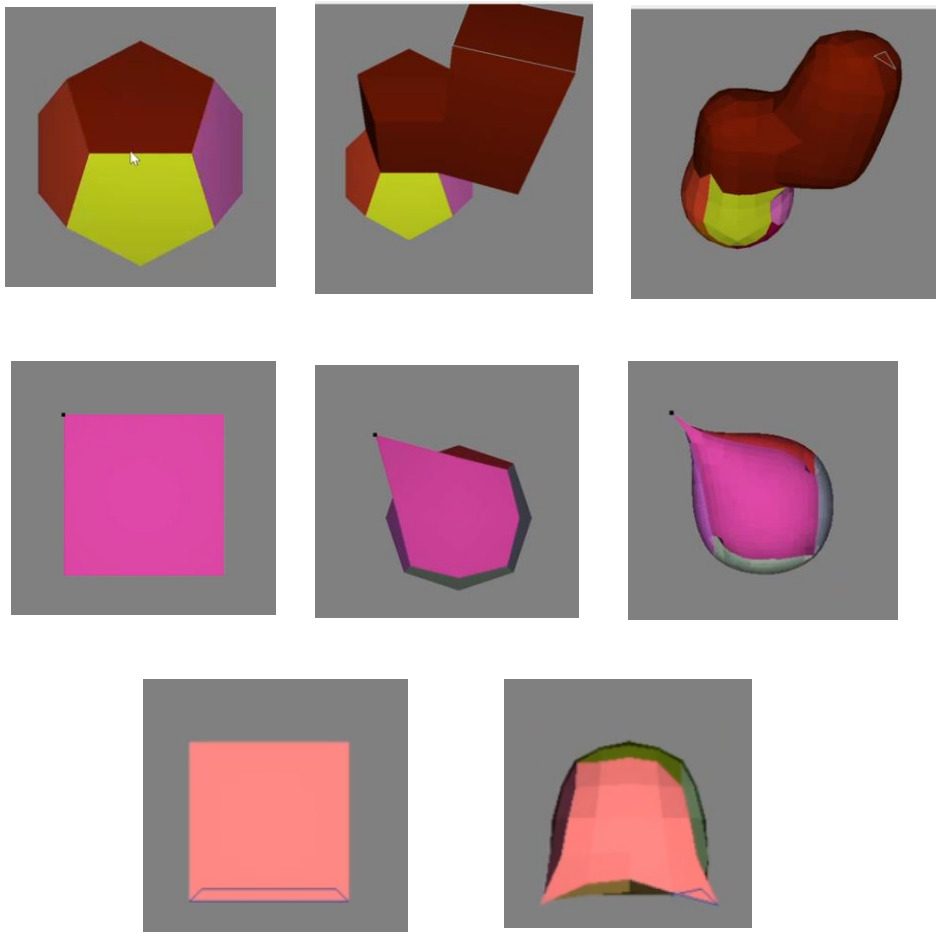
- 利用L-System生成分叉线段，表示河流形状。根据距离线段的距离对地面高度进行插值，使高度过渡较为自然。

- 使用unordered_map，以线段所覆盖的区块作为key储存线段数据，便于地形计算时查找线段数据，节约时间。

1. Mini-Minecraft

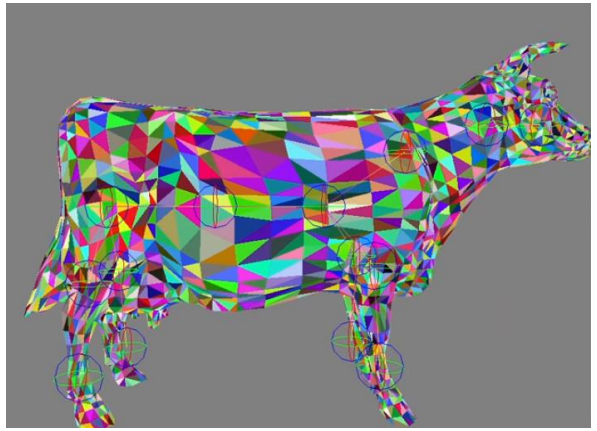


2. Micro-Maya

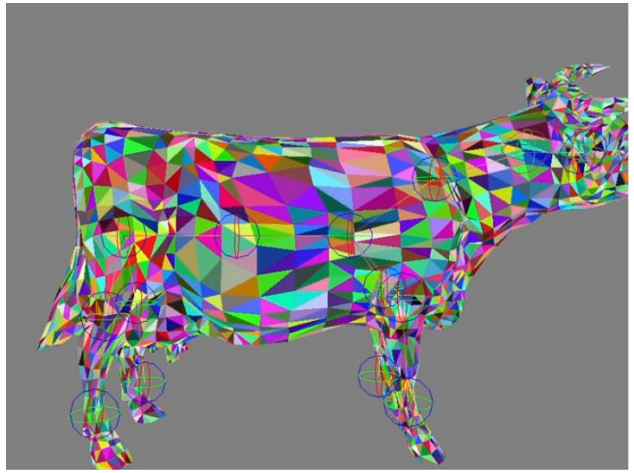


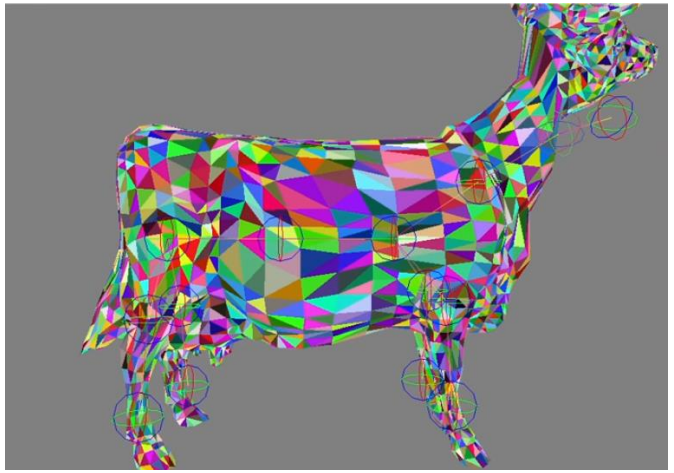
- 突出所选平面
- 可将物体进一步细分，细分时可以设置尖锐点/线/面，从而调整细分效果

2. Micro-Maya

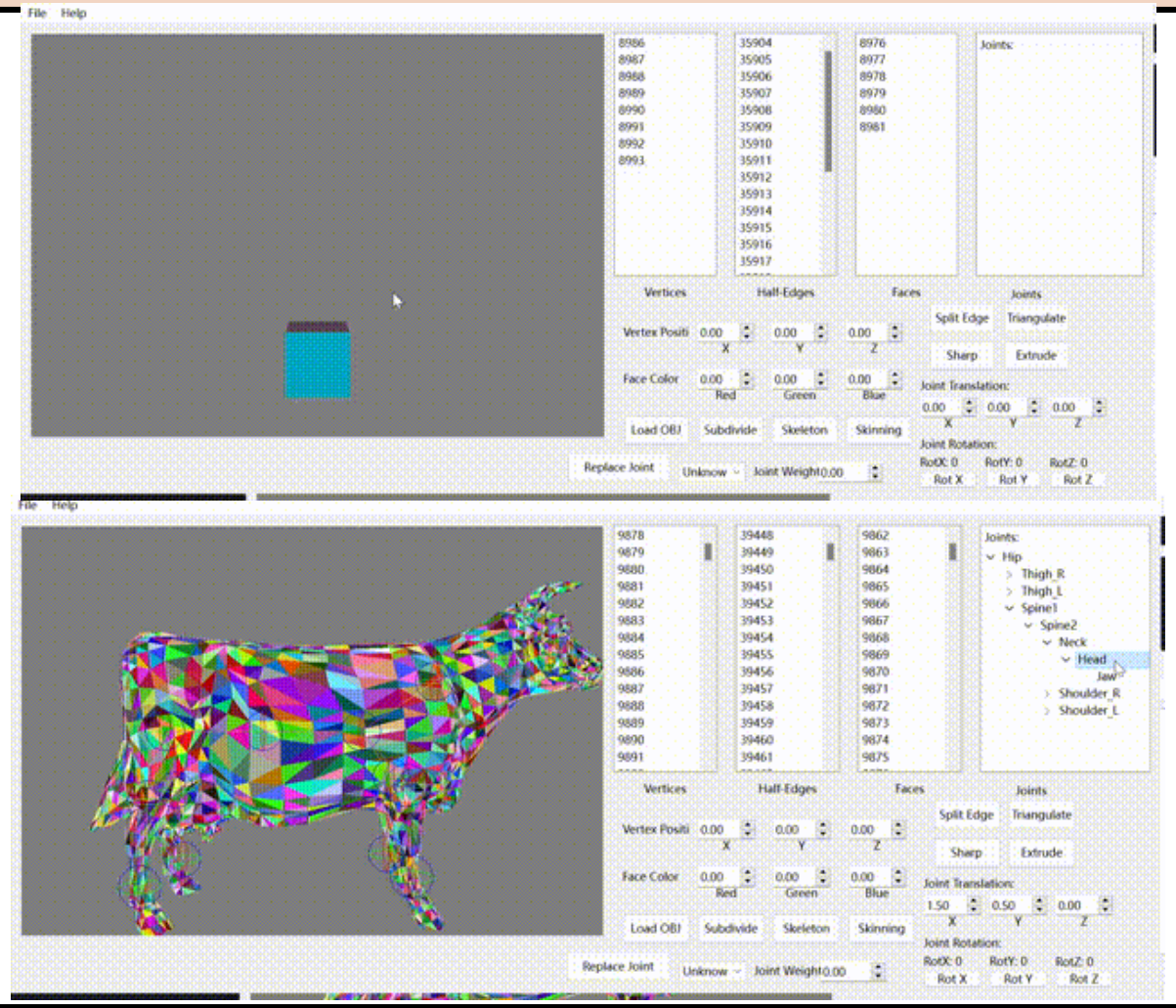


- 可读取骨骼数据，并进行骨骼绑定，根据顶点距离骨骼节点的距离实现简单的蒙皮动画效果





2. Micro-Maya


File Help
Vertices
Half-Edges
Faces
Joints
Split Edge
Triangulate
Sharp
Extrude
Face Color
Red
Green
Blue
Joint Translation:
Joint Rotation:
Load OBJ
Subdivide
Skeleton
Skinning
Replace Joint
Unknow
Joint Weight
Hip
Thigh_R
Thigh_L
Spine1
Spine2
Neck
Head
Jaw
Shoulder_R
Shoulder_L

3

Unity/UE 游戏

多个使用Unity/UE引擎制作的3D游戏

1. Escape

Escape – Unity 3D 冒险类游戏

- 玩家需要操纵主角探索宇宙飞船，解决机关，通过平台跳跃挑战并打败敌人，从而抵达终点，逃离飞船。玩家可以在UI菜单中使用药水回复生命及精力(跑/跳会消耗精力值)。

2. Kung-Fu Master

Kung-Fu Master – Unity VR 体感游戏

- 在该游戏中，玩家需要佩戴VR设备，尽量准确的模仿 NPC 的武术动作，来通过不同的游戏关卡。
- 通过检测玩家佩戴的传感器的空间位置，系统会判定玩家姿态是否与武术动作向匹配。游戏中共有三种武术动作，包括：静态动作，动态动作（玩家需要参照NPC动画来跟随NPC动作）及武器动作（玩家需要拾取选取正确的武器，并对准武器的动作）。

2. Kung-Fu Master

3. Inertial Antics

Inertial Antics – UE 3D解谜游戏

- 在该游戏中，玩家可以附身于不同物体上，利用每个物体的特殊功能进行移动，或者触发机关，进行解谜，从而最终抵达关卡的终点。

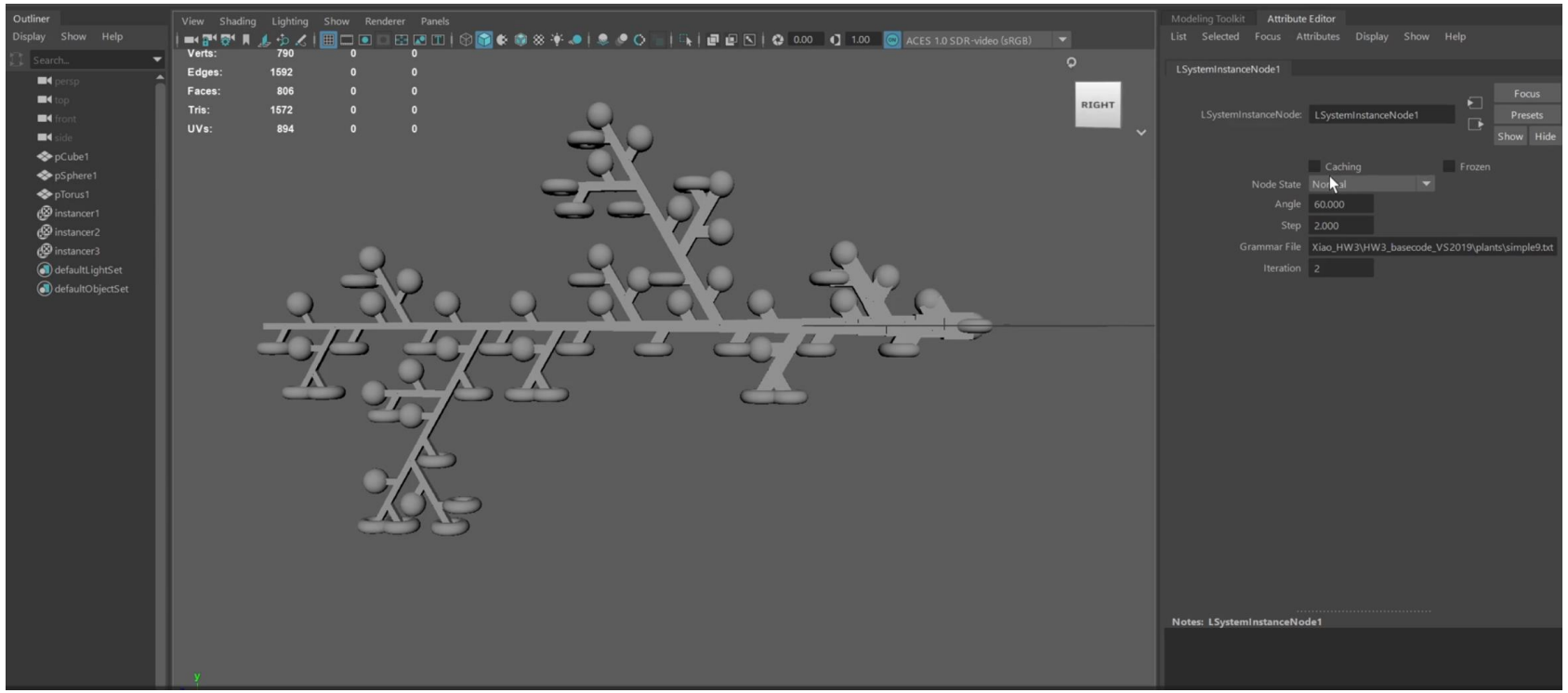
3. Inertial Antics

4

Maya插件

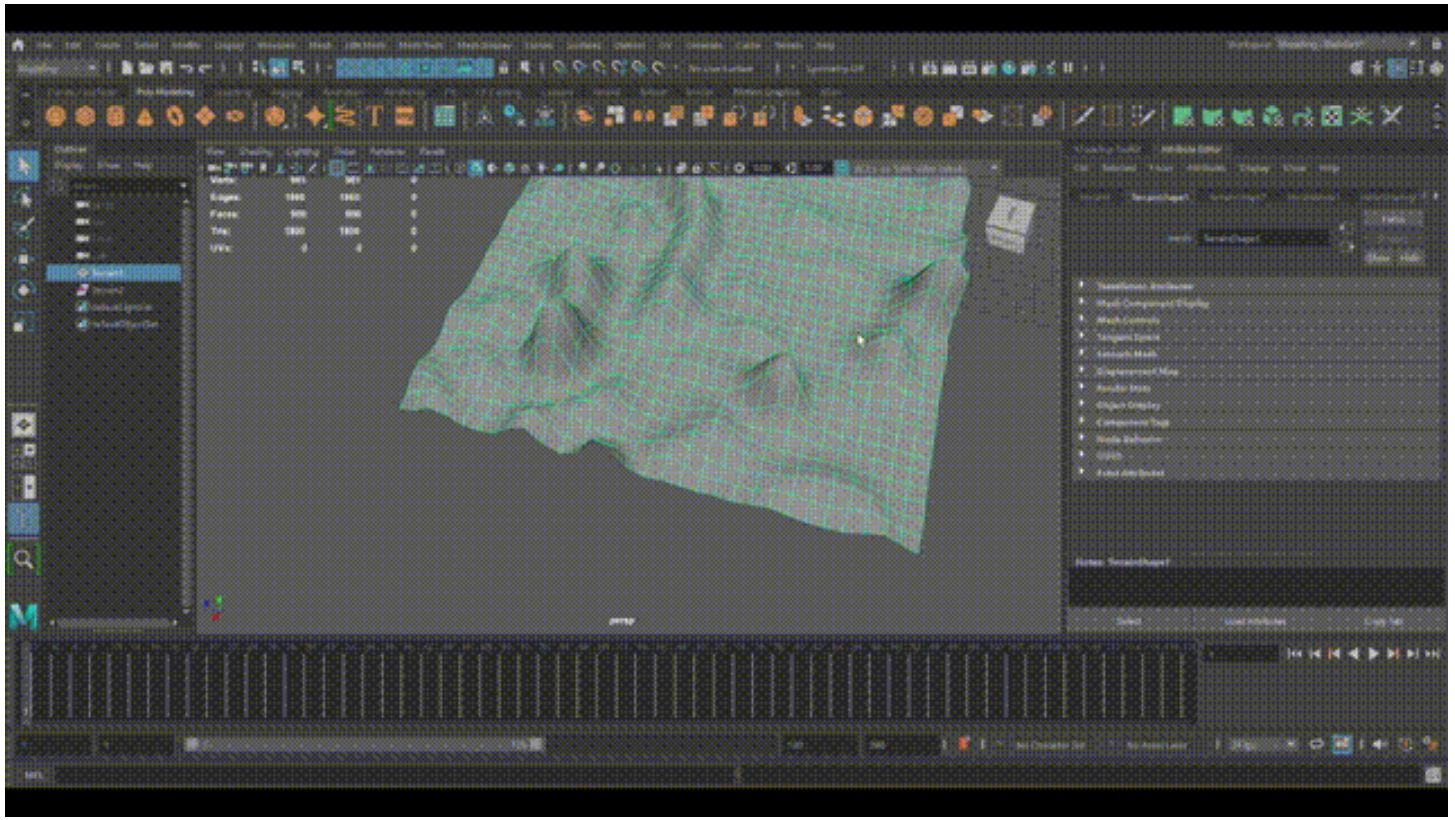
多个基于Mel及C++ Maya API实现的Maya插件

1. L-System插件



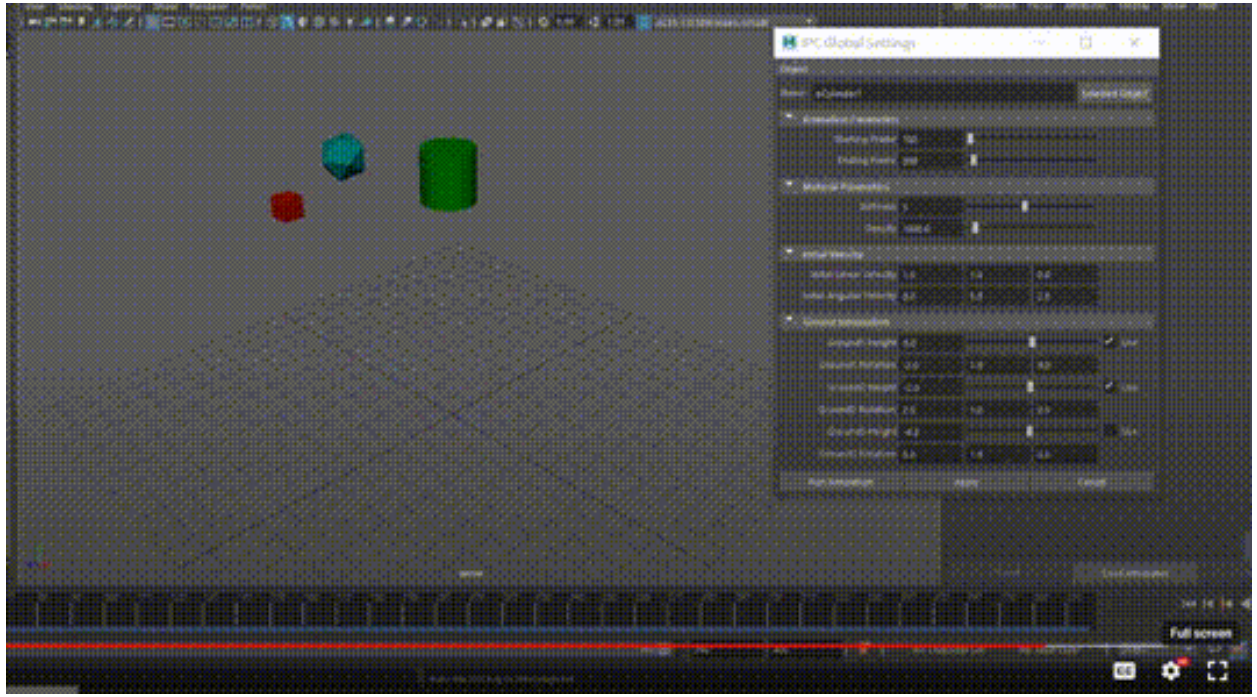
可用于生成树木结构模型，及各种花纹装饰。
通过参数控制L-System算法步长及旋转角度。

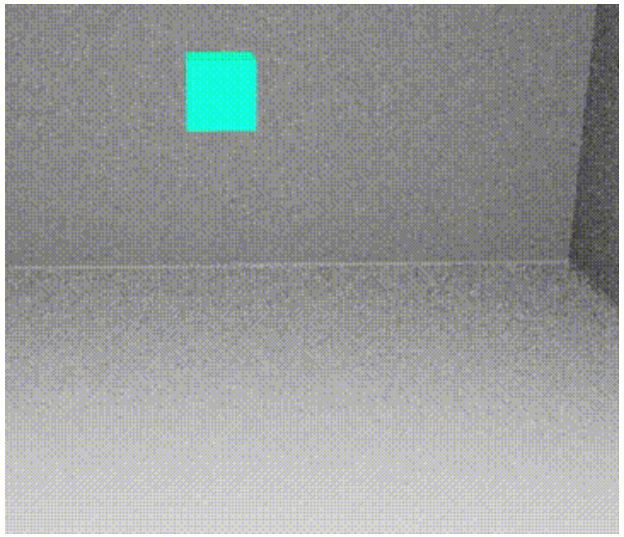
2. 地形插件



基于 Worley & Perlin 算法的地形生成插件

3. IPC物理模拟算法插件





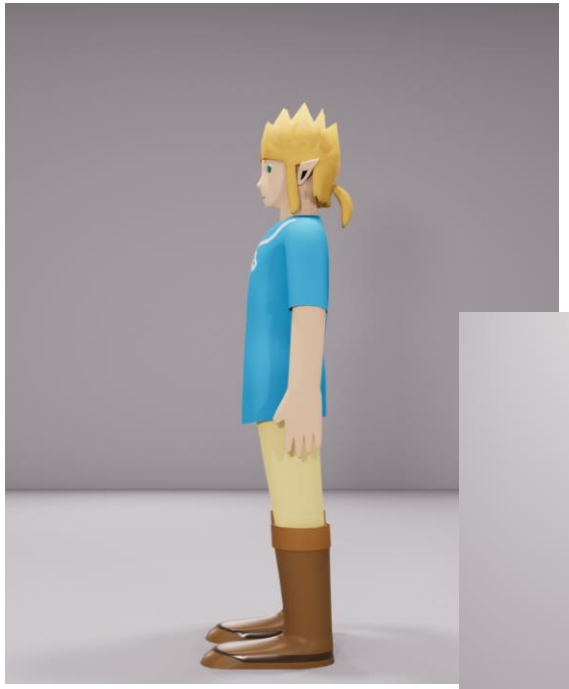
IPC (Incremental Potential Contact)算法插件，可基于 IPC 算法计算并生成弹性物体与固定平面的碰撞动画；

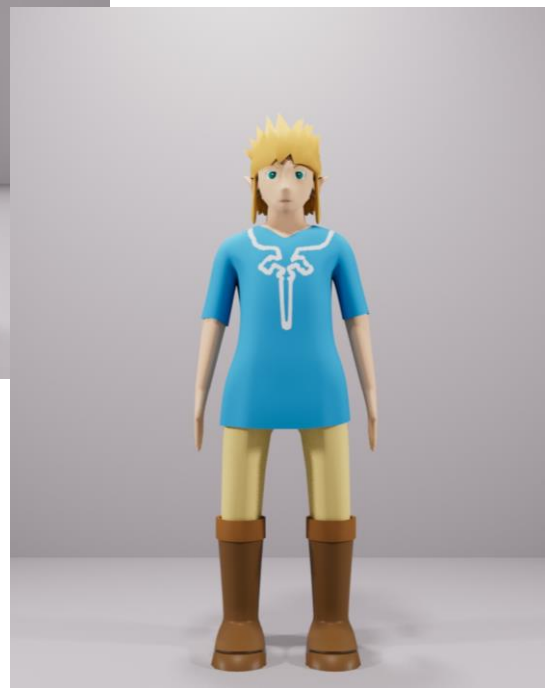
5

其他

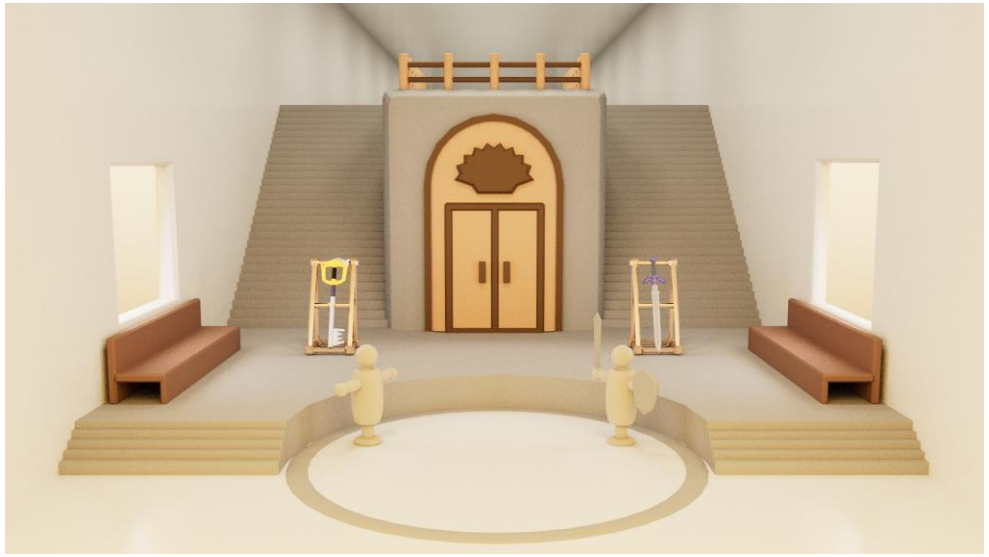
包括Maya建模作品，光线追踪离线渲染器，
基于PBR流程的渲染作品等

1. Maya建模合集

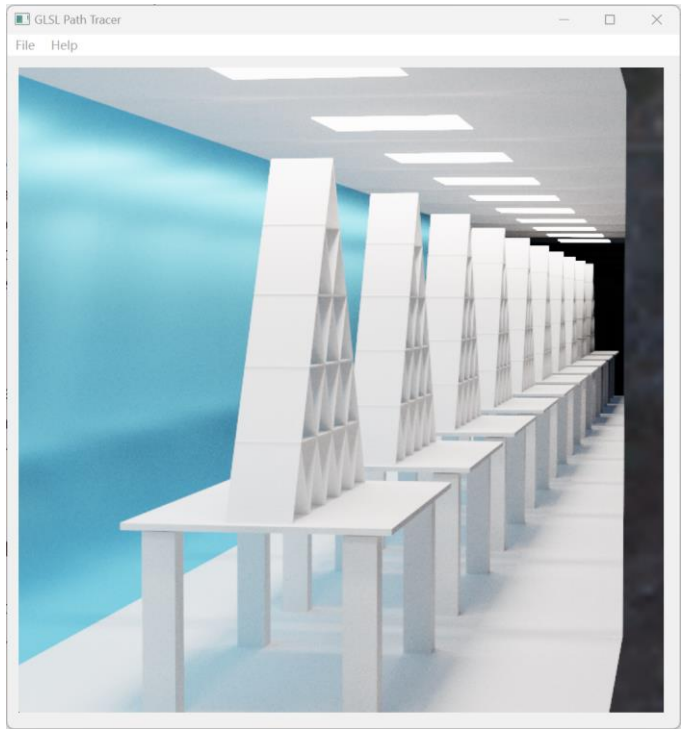


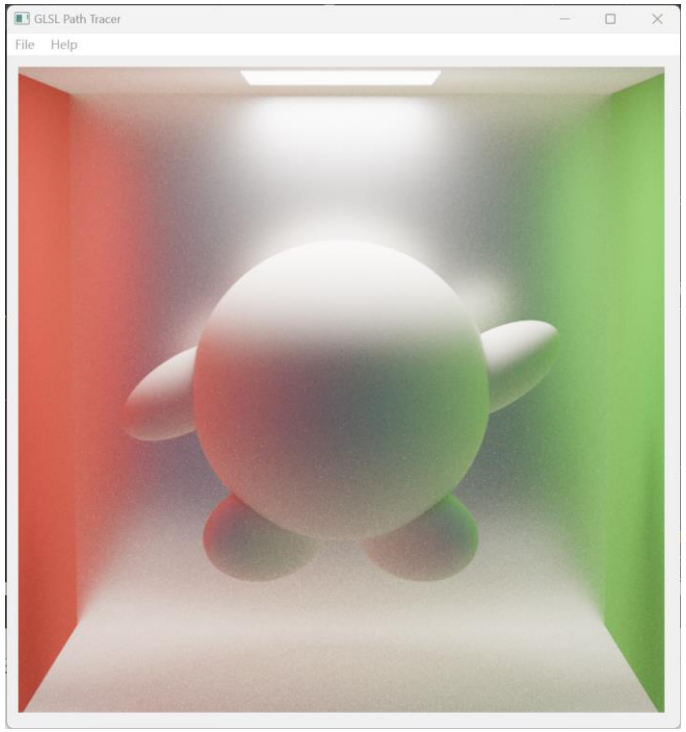


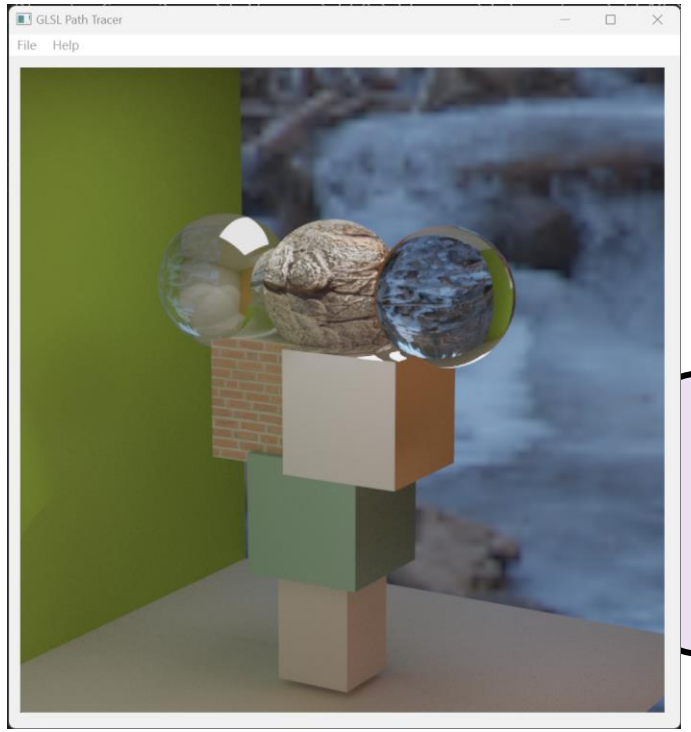
林克-抽象欢乐风格



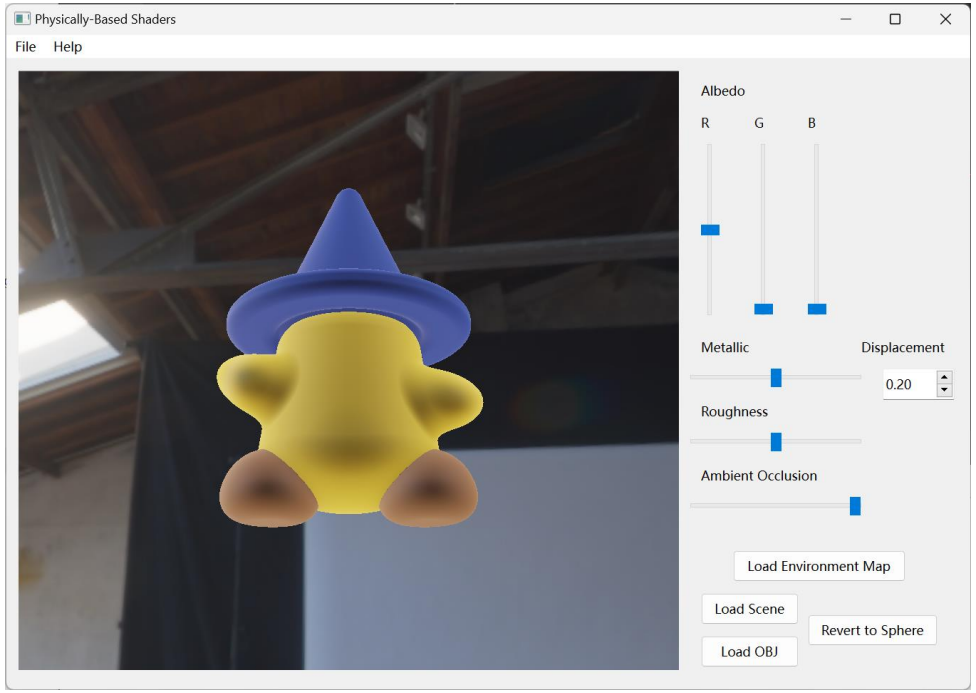
2. 光线追踪渲染器 (OpenGL+Qt)



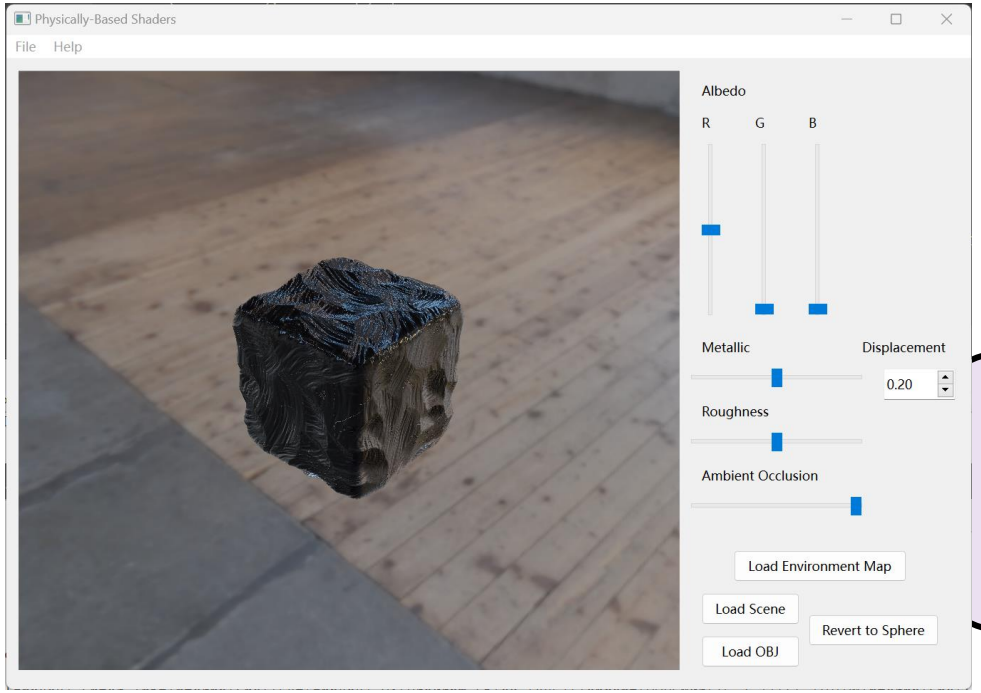




2. 光线追踪渲染器 (OpenGL+Qt)

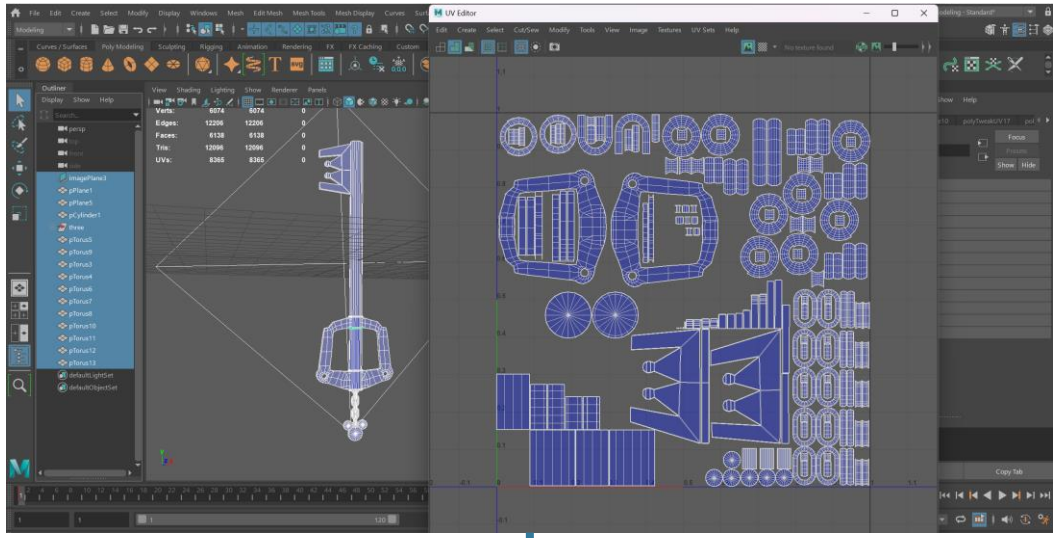


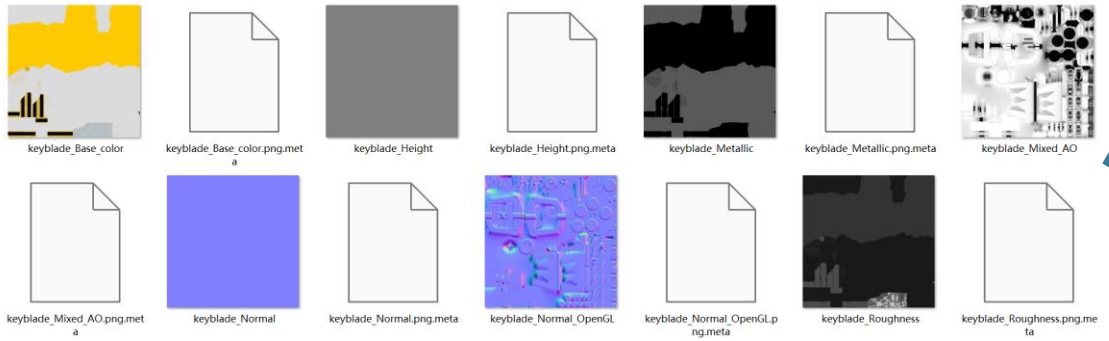
基于SDF建模并计算法线
同时计算次表面散射效果



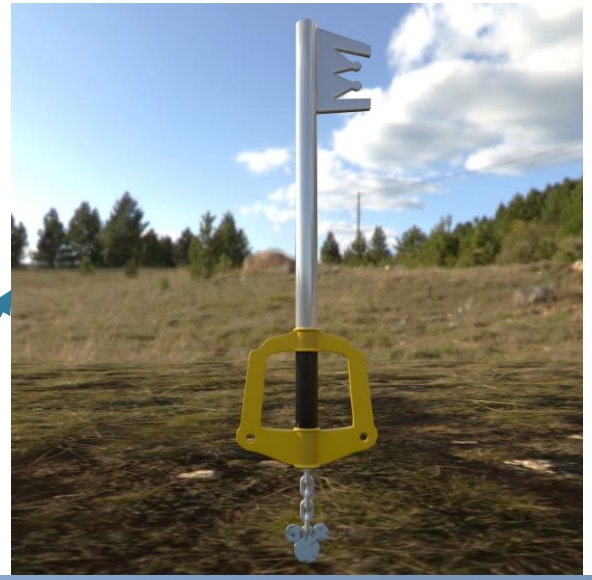
IBL+凹凸贴图

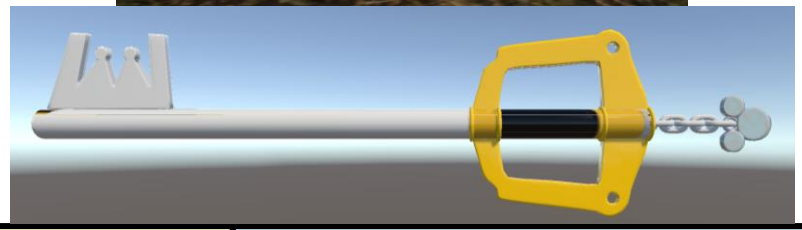
3. PBR材质制作



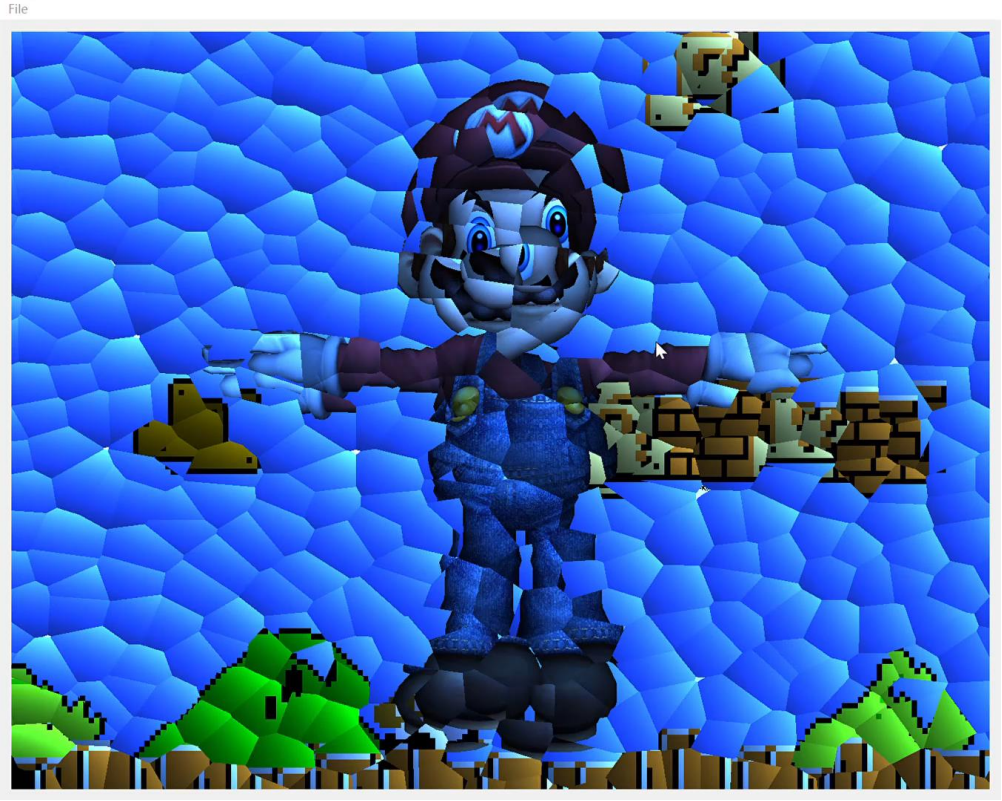


1. Maya建模+展UV
2. SP制作PBR贴图
3. SP渲染/引擎内渲染

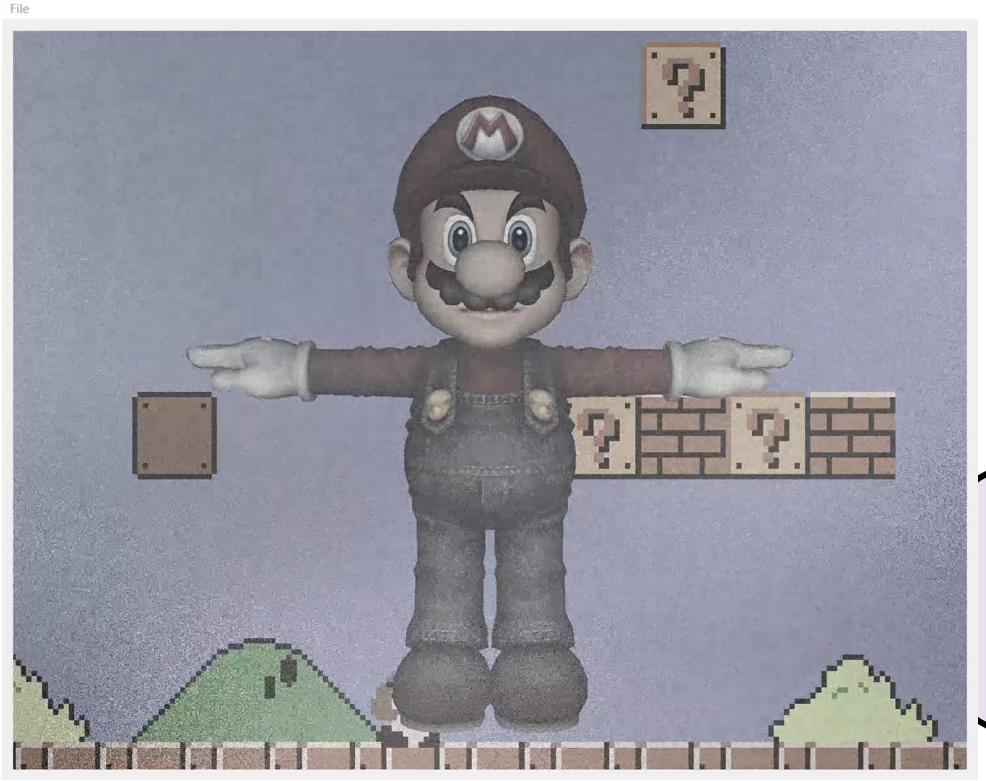




2. OpenGL后处理效果



水面碎镜



迷雾

感谢您的翻阅！
作品集会持续不断更新！