CSCI 522 – Game Engine - Fall 2024 – M1

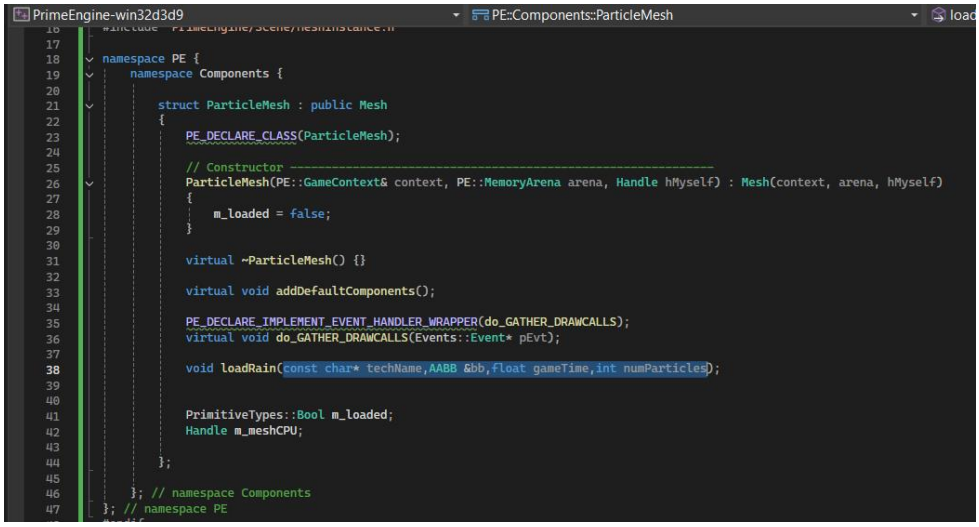
USC ID: 1154164561
NAME: Srija Madarapu
EMAIL: madarapu@usc.edu

Link to the video:

https://drive.google.com/drive/folders/1yM5kecOPfxU5k8f84o1ZXIpgKC-ZRw4Z?usp=sharing

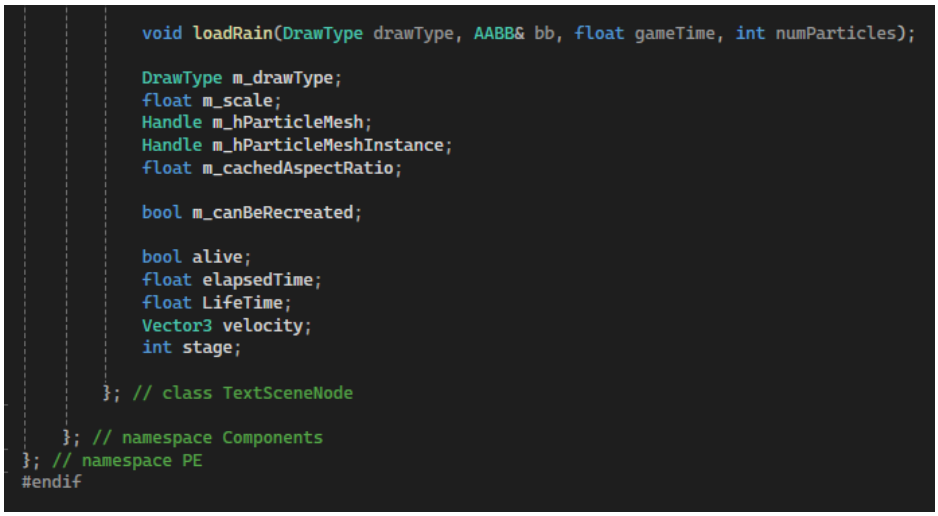
M1: Creating and managing particle effects in a game engine. Couldn't finish the rain effect but implemented particle effect.

ParticleMesh.h:



loadRain():This method takes parameters like technique name, bounding box, game time, and number of particles. It creates a billboard mesh with a specific texture and sets up manual buffer management. It then generates vertex and index buffers to represent a rain particle effect. This involves looping through a grid and creating quads at each point. Based on game time, the method generates a slightly random movement for the particles. Finally, it updates the GPU mesh based on whether it's the first time creating the effect or just updating existing particles.

ParticleSystemNode.h:



loadRain(): Creates a rain effect by calling the loadRain method of the ParticleMesh component. Configures the rendering technique and parameters for the rain effect.

MeshInstance.cpp:

```
void MeshInstance::drawParticle(AABB& bb, float gameTime)
{

    ParticleSystemNode* pSN = 0;

    if (m_hParticle.isValid())
    {
        pSN = m_hParticle.getObject<ParticleSystemNode>();
    }
    else
    {
        m_hParticle = PE::Handle("Particle_System_Node", sizeof(ParticleSystemNode));
        pSN = new(m_hParticle) ParticleSystemNode(*m_pContext, m_arena, m_hParticle);
        pSN->addDefaultComponents();
        addComponent(m_hParticle);
    }
    ParticleSystemNode::DrawType drawType = ParticleSystemNode::InWorld;

    pSN->loadRain(drawType, bb, gameTime, m_numParticles);
    pSN->m_base.setPos(bb.pos);
    pSN->m_scale = 1.0f;
}
```

Retrieving or Creating a ParticleSystemNode: If a ParticleSystemNode already exists (stored in m_hParticle), it retrieves it. Otherwise, it creates a new ParticleSystemNode, adds its default components, and attaches it to the MeshInstance.

Setting Up the ParticleSystemNode:Sets the drawType to InWorld, indicating that the particles should be drawn in the 3D world space. Calls the loadRain method on the ParticleSystemNode to initiate the rain effect, passing the bounding box, game time, and number of particles. Sets the base position of the ParticleSystemNode to the position of the bounding box. Sets the scale of the ParticleSystemNode to 1.0.

SH_DRAW.cpp:

```
bool inside = true;
if (pDrawEvent != NULL && pDrawEvent->isCornerInitialized) {
    PE::Components::AABB AABB_Box = pMeshCaller->m_AABB;
    Handle parentSceneNode = pInst->getFirstParentByType<SceneNode>();
    Matrix4x4 base = parentSceneNode.getObject<SceneNode>()->m_worldTransform;
    PE::Components::AABB aabb(base * AABB_Box.max, base * AABB_Box.min);


    float* distances = pDrawEvent->planeDistances;
    Vector3* normals = pDrawEvent->planeNormals;

    for (int i = 0; i < 6; ++i) {
        int outcount = 0;
        for (int j = 0; j < 8; ++j) {
            if (normals[i].dotProduct(aabb.corners[j]) + distances[i] < 0.0f) {outcount++;}
        }
        if (outcount == 8) {
            inside = false;
        }
    }
    if (inside)
    {
        pInst->m_culledOut = false;
        ++pMeshCaller->m_numVisibleInstances;
        DebugRenderer::Instance()->debugRenderAABB(pInst,pMeshCaller,0.2f,true);
    }
    else {pInst->m_culledOut = true;}
    if (pInst->m_numParticles != 0)
    {pInst->drawParticle(aabb, pDrawEvent->m_gameTime);PEINFO("particlesystem3");
    }
}
```

StaticMesh.lua

Added one more argument to the staticmesh to get the num of paritcles. Also added the num of particles values in standard events to read from the static mesh.

GameObjectManager.cpp

Pass number of paritcles value from event to mesh.