3D Collision System

Ruijia Huang

[Download the project file here]

WHAT THE PROJECT HAS

Three types of Colliders:

1. Sphere Collider
2. AABB Collider (Axis Aligned Bounding Box)
   The size of the AABB collider will be adjusted automatically when you rotate the game object using **sRigidBodyState**.
3. OBB Collider (Oriented Bounding Box)

Collision Detection and Response:

function<void(const ColliderBase*)> OnColliderEnter;
function<void(const ColliderBase*)> OnColliderStay;
function<void(const ColliderBase*)> OnColliderExit;

Collision System Octree Optimization(Only applicable to AABB colliders)

INSTRUCTIONS TO USE

Set the project

1. Copy the **include** and **library** folder to wherever you like in the project.
2. Add Additional Include Directories (Property ->C/C++ -> General -> Additional Include Directories).
3. Add Additional Library Directories (Property ->Linker -> General -> Additional Library Directories).
4. Add Additional Dependencies (Property ->Linker -> General -> Additional Dependencies).
5. Include **Collision.h** in the MyGame.cpp file.

The variables and functions that you might use

Collison System

```
cResult AddCollider(eae6320::Collision::ColliderBase* s_Collider);
cResult Update(const float i_secondCountToIntegrate); //Collision is detected here.
cResult Initialize();
cResult CleanUp();
```

ColliderBase

```
bool isActive;
ColliderType m_type;
const Physics::sRigidBodyState* gameObject;

ColliderBase();
void SetActive(bool active);

std::function<void(const ColliderBase*)> OnColliderEnter;
std::function<void(const ColliderBase*)> OnColliderStay;
std::function<void(const ColliderBase*)> OnColliderExit;
```

Sphere Collider

```
SphereCollider(const Physics::sRigidBodyState* rigidBodyState, float r = 0.5f);
void SetSize(float r);
float GetSize() const;
```

AABB Collider & OBB Collider

```
AABBCollider(const Physics::sRigidBodyState * rigidBodyState, float hX = 0.5f, float hY = 0.5f, float hZ = 0.5f);
void SetSize(float hX, float hY, float hZ);
Math::sVector GetSize();
```

Add a Collider

```
eae6320::cResult eae6320::cMyGame::Initialize()
{
    Collision::Initialize();

    /*
    * Initialize your own gameobject.
    */

    //Set the size or use the default size
    Collision::AABBCollider* collider = new Collision::AABBCollider(gameObject->sRigidBodyState, 0.1f, 0.1f, 0.1f);

    collider->OnColliderEnter = [this](const Collision::ColliderBase* collider) {
        //TODO
    };
    collider->OnColliderStay = [this](const Collision::ColliderBase* collider) {
        //TODO
    };
    collider->OnColliderExit = [this](const Collision::ColliderBase* collider) {
        //TODO
    };

    Collision::AddCollider(collider);
}
```

Update the Collision System

```
void eae6320::cMyGame::UpdateBasedOnTime(const float i_elapsedSecondCount_sinceLastUpdate) {
    /*
    * Other systmes update.
    */

    Collision::Update(i_elapsedSecondCount_sinceLastUpdate);

}
```

There is an example of using this system in **demo.h** file. (DO NOT just copy the code into your project!)

Use the Octree Optimization
Add Preprocessor Definition **OCTREE** (Property-> C/C++ -> General -> Preprocessor -> Preprocessor Definitions).

WHAT I HAVE LEARNED THIS SEMESTER

The most helpful knowledge I learned this semester is the project setup. It helps me nicely organize the new collision system and make it easy for other students to use.

THINKING

Octree Optimization

When I put a thousand cubes in the game scene, the movement of cubes starts stuttering. That's because the time complexity of the collision detection algorithm without optimization is n*n and it requires a lot of calculations to determine whether two objects are intersecting or not. So, I'm thinking about doing a little optimization for my collision system.

OCTREE is a good choice. It can divide objects in the game scene so I only need to check neighboring cubes while doing collision detection.

If all the colliders are OBB colliders, the total time to update the system once is about FOUR times as long as when the colliders are all AABB colliders. And if I use the octree to accelerate, the time will be reduced to only a QUATRE.

ECS architectural pattern

At the beginning of this project, I was confused about where to add my collision system in the game engine, since I need a place to store all the colliders in the game scene and I don't want to add coupling. But it seems that we don't have a component manager in the game engine now. Finally, I make the collision system a component similar to the physics system and make all colliders contain a reference to sRigidBodyState, but if I can make the engine an ECS engine, it would be better to manage all the systems.