1、Overview

GGphys is a 100% cross platform deterministic 3D rigid body physical engine, which perfectly supports the development of frame synchronous online physical games. It is written in pure C#, and the operation is very simple.

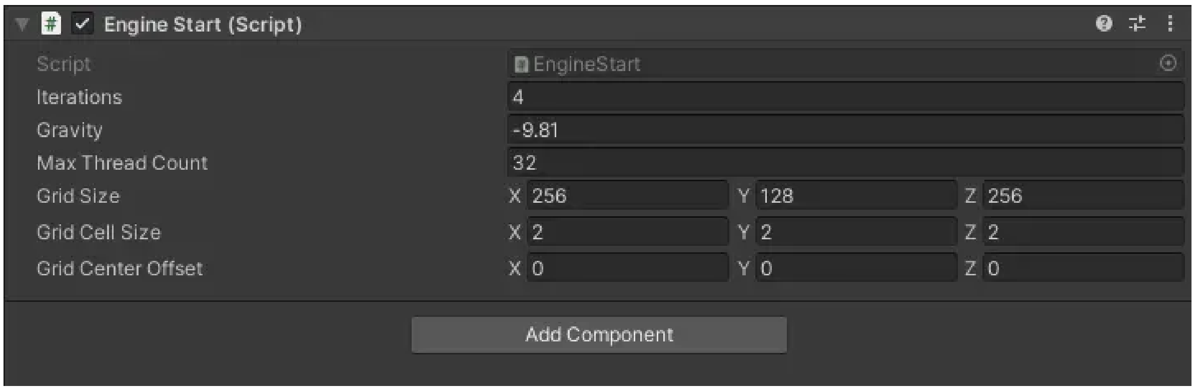
This version is used for unity。

asset store address: https://assetstore.unity.com/packages/slug/183465
e-mail: 298012895@qq.com

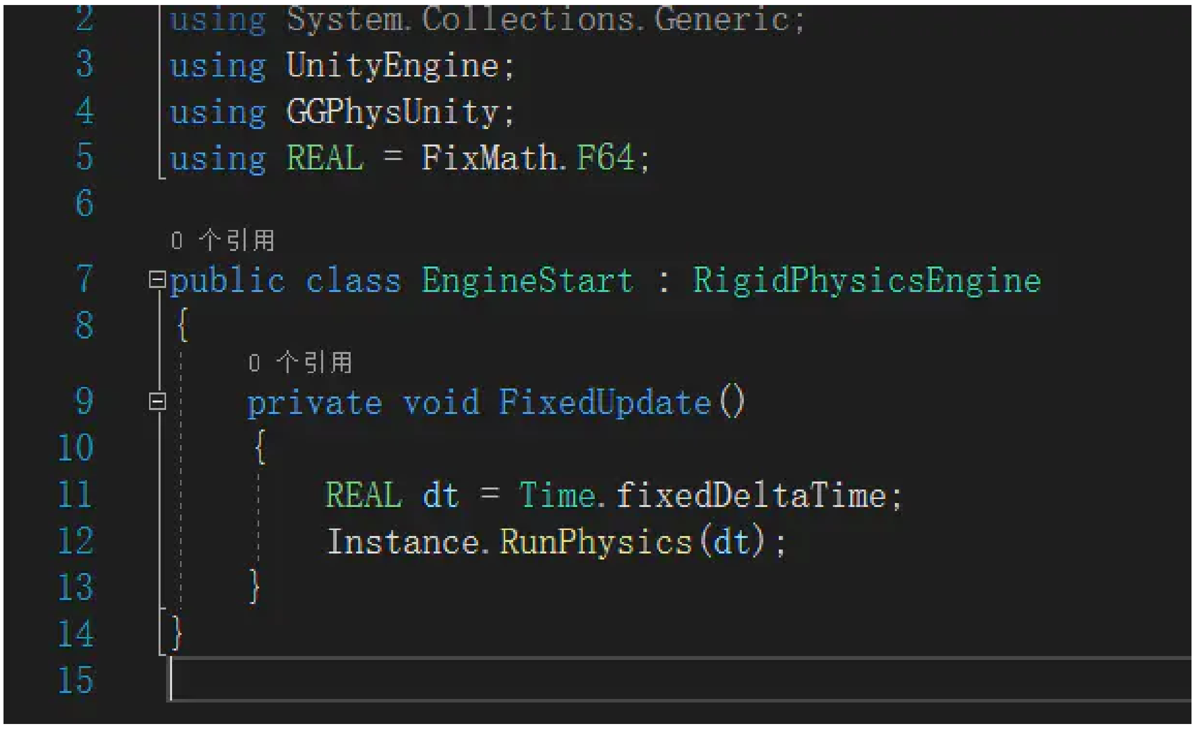
2、Features

- Pure C#
- Fixed point Math
- Spheres, capsules, boxes, and convex hulls collision detection
- Static meshes collision detection
- Collision constraint
- Triggers
- Collisions callback, triggers callback
- Layers
- Low cost sleep states for resting bodies
- Multi thread speed up
- Easy rigid body APIs
- 100% cross platform determinacy
- Support lockstep multi player game develop

3、Usage


Engine Start (Script)

Script	EngineStart		
Iterations	4		
Gravity	-9.81		
Max Thread Count	32		
Grid Size	X 256	Y 128	Z 256
Grid Cell Size	X 2	Y 2	Z 2
Grid Center Offset	X 0	Y 0	Z 0



```
using System.Collections.Generic;
using UnityEngine;
using GGPhysUnity;
using REAL = FixMath.F64;

0 个引用
public class EngineStart : RigidPhysicsEngine
{
    0 个引用
    private void FixedUpdate()
    {
        REAL dt = Time.fixedDeltaTime;
        Instance.RunPhysics(dt);
    }
}
```

3.1、Engine Start

Use your own monobehavior, inherited from rigidphysics engine, and call it where you need it Instance.RunPhysics (F64 dt)

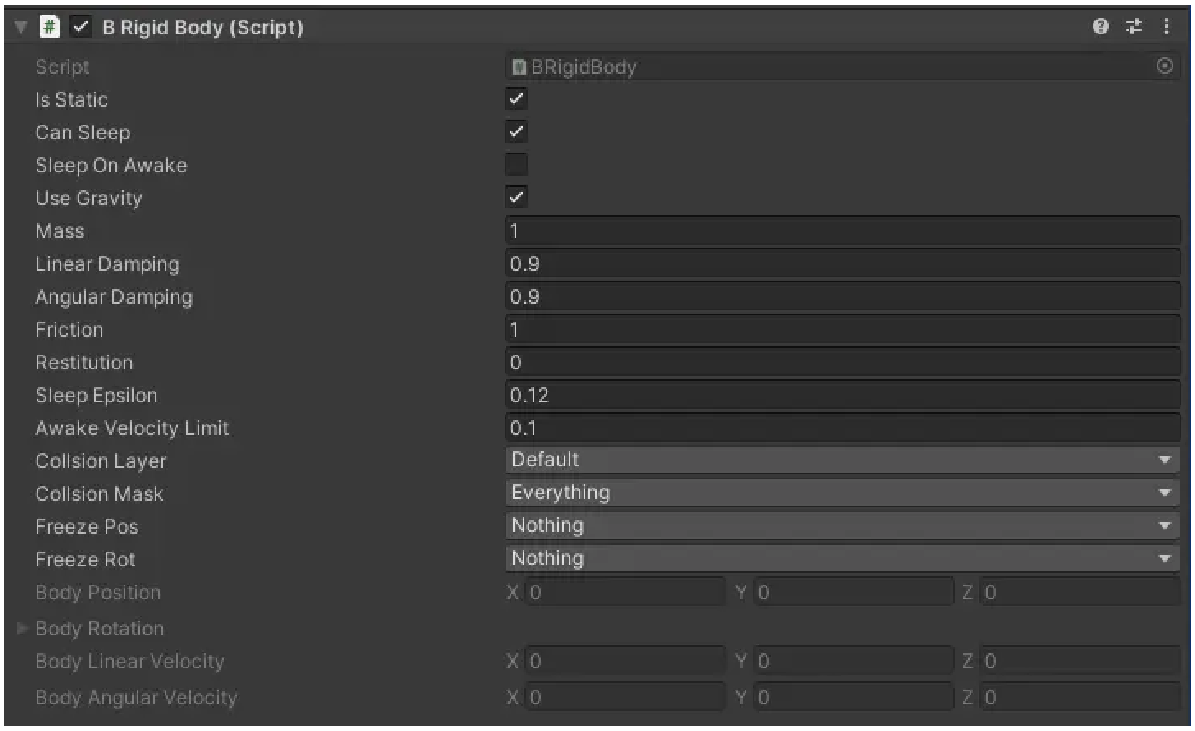
3.2、Grid

The broad phase of the collision detection system of the engine adopts grid space division, that is, the whole space is divided into several cube grids, and only the rigid bodies within the same grid range can carry out collision detection.

3.3、Engine Parameters

Iterations: contact resolve Iteration count
Gravity:gravity magnitude
Max Thread Count: used for speed up
Grid Size:the size of grid
Grid Cell Sizethe size of every grid cell
Grid Center Offset:the center of all grids

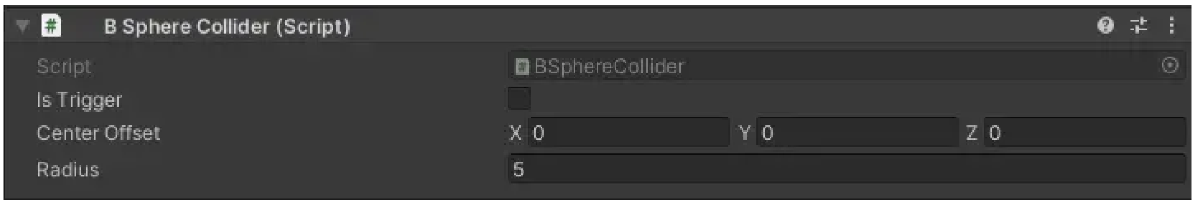
3.4、Rigid Body


B Rigid Body (Script)

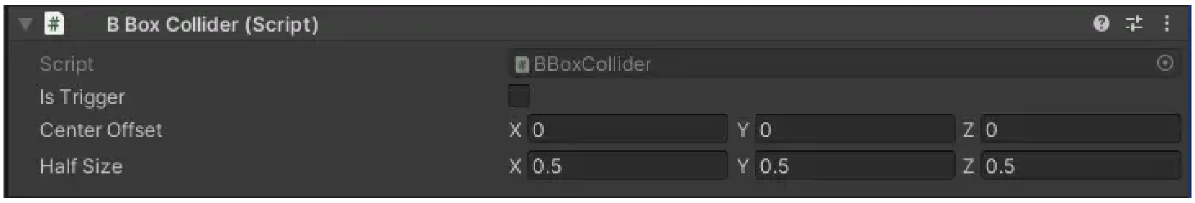
Script	BRigidBody
Is Static	✓
Can Sleep	✓
Sleep On Awake	
Use Gravity	✓
Mass	1
Linear Damping	0.9
Angular Damping	0.9
Friction	1
Restitution	0
Sleep Epsilon	0.12
Awake Velocity Limit	0.1
Collsion Layer	Default
Collsion Mask	Everything
Freeze Pos	Nothing
Freeze Rot	Nothing
Body Position	X 0 Y 0 Z 0
Body Rotation	
Body Linear Velocity	X 0 Y 0 Z 0
Body Angular Velocity	X 0 Y 0 Z 0

Is Static:Whether it is a static rigid body, the static rigid body can save performance. The static rigid body will not be affected by collision and force, but it will participate in collision detection, and two static objects will not collide with each other
Can Sleep:Whether to enable the sleep mechanism or not, when the speed of the enabled rigid body is low enough, it will enter into sleep, which can save performance
Sleep On Awake:Whether to enable the sleep on awake
Use Gravity:Is it affected by gravity
Mass:mass magnitude
Linear Damping:[0,1]
Angular Damping:[0,1]
Friction:[0,1]
Restitution:[0,1]
Sleep Epsilon:If the overall speed is lower than this value, it will enter into sleep
Awake Velocity Limit:If the collision speed is higher than this value, Automatic wake-up will be carried out
Collision Layer:layer of rigid body
Collision Mask:collision with witch layers
Freeze Pos:the direction of position freezed
Freeze Rot:the direction of rotation freezed

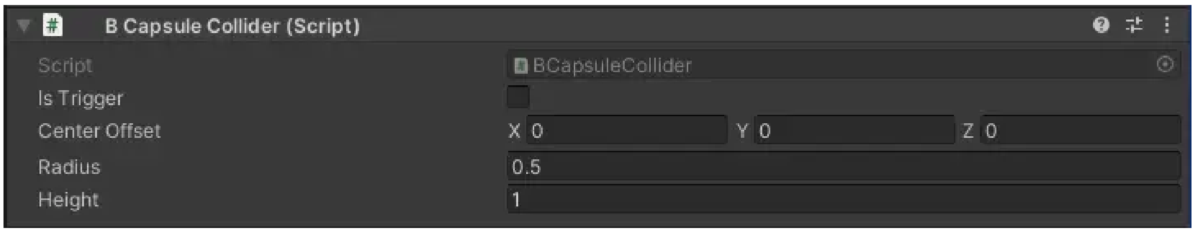
3.5、Collider


B Sphere Collider (Script)

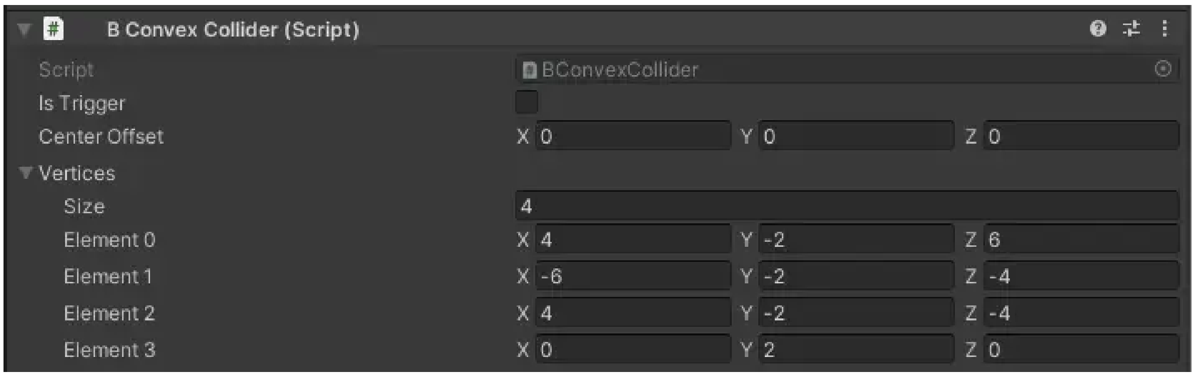
Script	BSphereCollider
Is Trigger	
Center Offset	X 0 Y 0 Z 0
Radius	5


B Box Collider (Script)

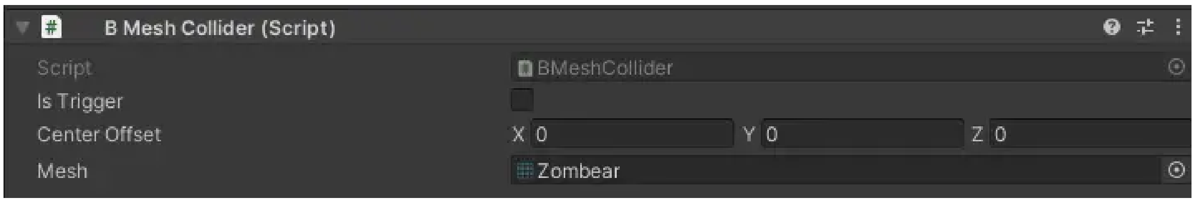
Script	BBoxCollider
Is Trigger	
Center Offset	X 0 Y 0 Z 0
Half Size	X 0.5 Y 0.5 Z 0.5


B Capsule Collider (Script)

Script	BCapsuleCollider
Is Trigger	
Center Offset	X 0 Y 0 Z 0
Radius	0.5
Height	1


B Convex Collider (Script)

Script	BConvexCollider
Is Trigger	
Center Offset	X 0 Y 0 Z 0
Vertices	
Size	4
Element 0	X 4 Y -2 Z 6
Element 1	X -6 Y -2 Z -4
Element 2	X 4 Y -2 Z -4
Element 3	X 0 Y 2 Z 0


B Mesh Collider (Script)

Script	BMeshCollider
Is Trigger	
Center Offset	X 0 Y 0 Z 0
Mesh	Zombear

4、API

```
        /// <summary>
        /// position of gameObject
        /// </summary>
        public Vector3 Position;

        /// <summary>
        /// rotation of gameObject
        /// </summary>
        public Quaternion Rotation;

        /// <summary>
        /// linear velocity
        /// </summary>
        public Vector3 Velocity;

        /// <summary>
        /// angular velocity
        /// </summary>
        public Vector3 AngularVelocity;

        /// <summary>
        /// set isStatic
        /// </summary>
        public void SetStatic(bool isStatic);

        /// <summary>
        /// set mass
        /// </summary>
        /// <param name="mass"></param>
        public void SetMass(float mass);
        /// <summary>
        /// set position frozen
        /// </summary>
        /// <param name="freeze"></param>
        public void SetFreezePosition(byte freeze);

        /// <summary>
        /// set rotation frozen
        /// </summary>
        /// <param name="freeze"></param>
        public void SetFreezeRotation(byte freeze);

        /// <summary>
        /// set friction
        /// </summary>
        /// <param name="friction"></param>
        public void SetFriction(REAL friction);

        /// <summary>
        /// set restitution
        /// </summary>
        /// <param name="restitution"></param>
        public void SetRestitution(REAL restitution);

        /// <summary>
        /// enable gravity
        /// </summary>
        /// <param name="useGravity"></param>
        public void SetUseGravity(bool useGravity);

        /// <summary>
        /// set linear damping
        /// </summary>
        /// <param name="damping"></param>
        public void SetLinearDamping(float damping);

        /// <summary>
        /// set angular damping
        /// </summary>
        /// <param name="damping"></param>
        public void SetAngularDamping(float damping);

        /// <summary>
        /// set sleep episilon
        /// </summary>
        /// <param name="epsilon">If the overall speed is lower than this value,
it will enter into
        /// sleep</param>
        public void SetSleepEpsilon(float epsilon);

        /// <summary>
        /// set awakeVelocityLimit
        /// </summary>
        /// <param name="limit"></param>
        public void SetAwakeVelocityLimit(float limit);

        /// <summary>
        /// awake
        /// </summary>
        /// <param name="awake">If the collision speed is higher than this value, 
Automatic wake-
        /// up will be carried out</param>
        public void SetAwake(bool awake);

        /// <summary>
        /// addforce to body
        /// </summary>
        /// <param name="force"></param>
        /// <param name="awakeBody">whether awake body</param>
        public void AddForce(Vector3 force, bool awakeBody= false);

        /// <summary>
        /// addforce to body
        /// </summary>
        /// <param name="force"></param>
        /// <param name="awakeBody">whether awake body</param>
        public void AddForce(Vector3d force, bool awakeBody = false);

        /// <summary>
        /// AddForceAtPoint in world space
        /// </summary>
        public void AddForceAtPoint(Vector3 force, Vector3 point);

        /// <summary>
        /// AddForceAtPoint in world space
        /// </summary>
        public void AddForceAtPoint(Vector3d force, Vector3 point);

        /// <summary>
        /// ApplyLinearImpulse
        /// </summary>
        /// <param name="impulse"></param>
        public void ApplyLinearImpulse(Vector3 impulse);

        /// <summary>
        /// ApplyLinearImpulse
        /// </summary>
        /// <param name="impulse"></param>
        public void ApplyLinearImpulse(Vector3d impulse);

        /// <summary>
        /// ApplyAngularImpulse
        /// </summary>
        /// <param name="impulse"></param>
        public void ApplyAngularImpulse(Vector3 impulse);

        /// <summary>
        /// ApplyAngularImpulse
        /// </summary>
        /// <param name="impulse"></param>
        public void ApplyAngularImpulse(Vector3d impulse);

        /// <summary>
        /// set body position
        /// </summary>
        /// <param name="position"></param>
        public void SetPosition(Vector3 position);

        /// <summary>
        /// set body position
        /// </summary>
        /// <param name="position"></param>
        public void SetPosition(Vector3d position);

        /// <summary>
        /// SetOrientation
        /// </summary>
        /// <param name="orientation"></param>
        public void SetOrientation(UnityEngine.Quaternion orientation);

        /// <summary>
        /// SetOrientation
        /// </summary>
        /// <param name="orientation"></param>
        public void SetOrientation(GGPhys.Core.Quaternion orientation);

        /// <summary>
        /// SetPositionAndOrientation
        /// </summary>
        public void SetPositionAndOrientation(Vector3 position,
UnityEngine.Quaternion orientation);

        /// <summary>
        /// SetPositionAndOrientation
        /// </summary>
        public void SetPositionAndOrientation(Vector3d position,
GGPhys.Core.Quaternion orientation);

        /// <summary>
        /// move
        /// </summary>
        /// <param name="delta"></param>
        public void Move(Vector3 delta);

        public void Move(Vector3d delta);

        /// <summary>
        /// rotate
        /// </summary>
        /// <param name="delta"></param>
        public void Rotate(Vector3 delta);

        public void Rotate(Vector3d delta);

        /// <summary>
        /// set body linear velocity
        /// </summary>
        /// <param name="velocity"></param>
        public void SetLinearVelocity(Vector3 velocity);

        public void SetLinearVelocity(Vector3d velocity);

        /// <summary>
        /// set body angular velocity
        /// </summary>
        /// <param name="velocity"></param>
        public void SetAnguarVolocity(Vector3 velocity);

        /// <summary>
        /// set body angular velocity
        /// </summary>
        /// <param name="velocity"></param>
        public void SetAnguarVolocity(Vector3d velocity);
```