Bullet3 Tutorial - 2

강영민

Rigid Body

- Class 이름
  - btRigidBody
- Creation
  - Collision shape이 필요
  - btRigidBodyConstructionInfo도 필요
- Rigid Body 클래스의 중요한 메소드
  - getMotion()
  - setLinearVelocity(btVector3)
  - setAngularVelocity(btVector3)
  - setRestitution(btScalar)
  - setLinearFactor(btVector3)
  - setAngularFactor(btVector3)
  - setDamping(btScalar, btScalar)

```
btCollisionShape* shape = new btStaticPlaneShape(btVector3(0, 1, 0), 1);
btDefaultMotionState* motionState = new btDefaultMotionState(btTransform(btQuaternion(0,0,0,1), btVector3(0,-1,0)));
btRigidBody::btRigidBodyConstructionInfo rigidBodyCI(
 0, // mass
 motionState, // initial position
 shape, // collision shape of body
 btVector3(0,0,0) // local inertia
);
btRigidBody *rigidBody = new btRigidBody(rigidBodyCI);
dynamicsWorld->addRigidBody(rigidBody);
```

Collision Shape

- 메시처럼 다양한 종류의 객체와 충돌하는 동작을 가짐
- 충돌 외형은 world position을 가지지 않음
  - 충돌 객체나 강체에 적용됨
  - 충돌 객체나 강체의 중심 위치와 방향(orientation)에 의해 실제 위치와 방향이 결정됨
- 종류
  - Primitives
    - btSphereShape, btBoxShape, btCylinderShape, btCapsuleShape, btConeShape, btMultiSphereShape
  - Meshes
    - btConvexHullShape, btConexTriangleMeshShape, btBvhTraiangleMeshShape, btHeightfieldTerrainShape, btStaticPlaneShape
  - Compound
    - btCompoundShape

Motion State

- 편리한 콜백 기능
  - 객체가 움직였을 때 그래픽을 갱신할 수 있게 해 줌
  - 운동중인 객체의 상태를 저장하고 있음
- 디폴트 모션 스테이트
  - btDefaultMotionState

```
btDefaultMotionState* ms = new btDefaultMotionState(btTransform(btQuaternion(0,0,0,1), btVector3(0,10,0)));
```

    - 강체를 생성할 때에 모션 스테이트를 지정하고 생성해야 함

Stepping

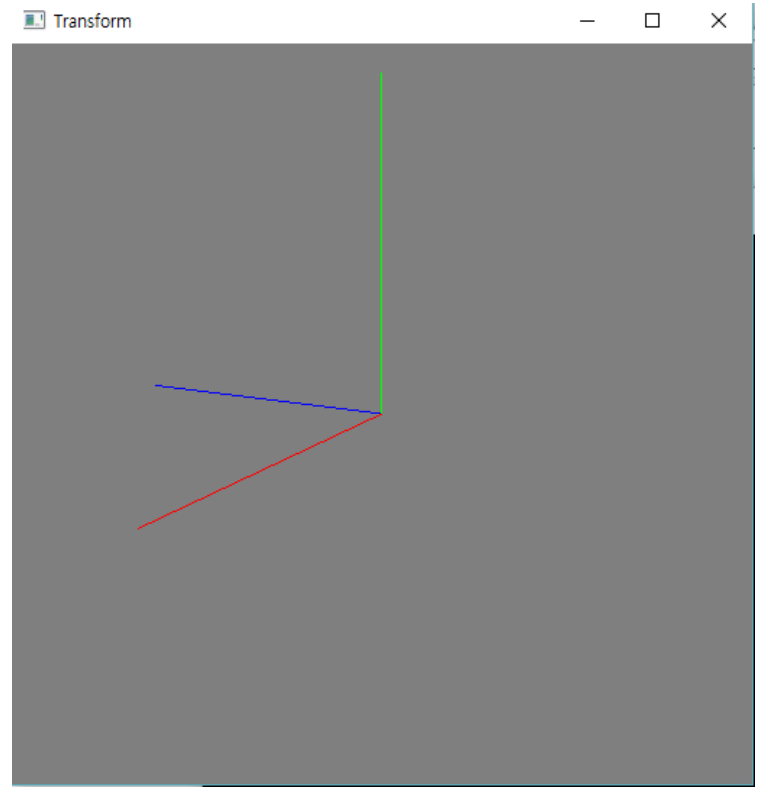
- btDynamicsWorld
  - stepSimulation

```
btDynamicsWorld::stepSimulation(
        btScalar timeStep,
        int maxSubSteps=1,
        btScalar fixedTimeStep=btScalar(1.)/btScalar(60.)
);
```

  - Parameter
    - 상태 갱신을 위한 시간 간격 (이전 프레임에서 현재 프레임까지의 시간)
    - 한 번의 프레임 계산에 Bullet이 수행할 수 있는 시뮬레이션 최대 회수
    - 시뮬레이션 계산에 사용하는 고정 크기의 간격

OpenGL 환경에서 테스트

- 프로젝트 생성
  - Win32 Console Project 생성
  - OpenGL 활용에 필요한 라이브러리 링크 설정
- OpenGLMain.cpp 생성
  - 간단한 그리기 실행



BulletWorld Header

```
#ifndef BULLETWORLD_HH_
#define BULLETWORLD_HH_
...
#include "../../src/btBulletDynamicsCommon.h"

class CBulletWorld {
		btBroadphaseInterface* broadphase;
		btDefaultCollisionConfiguration* collisionConfiguration;
		btCollisionDispatcher* dispatcher;
		btSequentialImpulseConstraintSolver* solver;
		btCollisionShape* groundShape;
		btCollisionShape* ballShape;
		btRigidBody* groundRigidBody;
		btRigidBody* ballRigidBody;
public:
		btDiscreteDynamicsWorld* mWorld;
		btDefaultMotionState* groundMotionState;
		btDefaultMotionState* ballMotionState;
public:
		void initialize();
		void draw();
		void step(float dt);
		void release();
};
#endif
```

```
void CBulletWorld::initialize() {
		broadphase = new btDbvtBroadphase();
		collisionConfiguration = new btDefaultCollisionConfiguration();
		dispatcher = new btCollisionDispatcher(collisionConfiguration);
		solver = new btSequentialImpulseConstraintSolver;

		mWorld = new btDiscreteDynamicsWorld(dispatcher, broadphase, solver, collisionConfiguration);

		mWorld->setGravity(btVector3(0, -10, 0));

		groundShape = new btStaticPlaneShape(btVector3(0, 1, 0), 1);
		ballShape = new btSphereShape(1);

		groundMotionState = new btDefaultMotionState(btTransform(btQuaternion(0, 0, 0, 1), btVector3(0, -1, 0)));
		btRigidBody::btRigidBodyConstructionInfo
			groundRigidBodyCI(0, groundMotionState, groundShape, btVector3(0, 0, 0));
		groundRigidBody = new btRigidBody(groundRigidBodyCI);
		groundRigidBody->setRestitution(1.0);
		mWorld->addRigidBody(groundRigidBody);


		ballMotionState =
			new btDefaultMotionState(btTransform(btQuaternion(0, 0, 0, 1), btVector3(0, 50, 0)));
		btScalar mass = 1;
		btVector3 ballInertia(0, 0, 0);
		ballShape->calculateLocalInertia(mass, ballInertia);
		btRigidBody::btRigidBodyConstructionInfo ballRigidBodyCI(mass, ballMotionState, ballShape, ballInertia);
		ballRigidBody = new btRigidBody(ballRigidBodyCI);
		ballRigidBody->setRestitution(1.0);
		mWorld->addRigidBody(ballRigidBody);

}
```

```
void CBulletWorld::step(float dt) {
	mWorld->stepSimulation(1/30.0, 10, 1.0/60.0);
}

void CBulletWorld::draw() {
	btTransform trans;
	ballRigidBody->getMotionState()->getWorldTransform(trans);
	glPushMatrix();
	btVector3 loc = trans.getOrigin();
	glTranslatef(loc[0], loc[1], loc[2]);
	glutWireSphere(1.0, 10, 10);
	glPopMatrix();
}
```

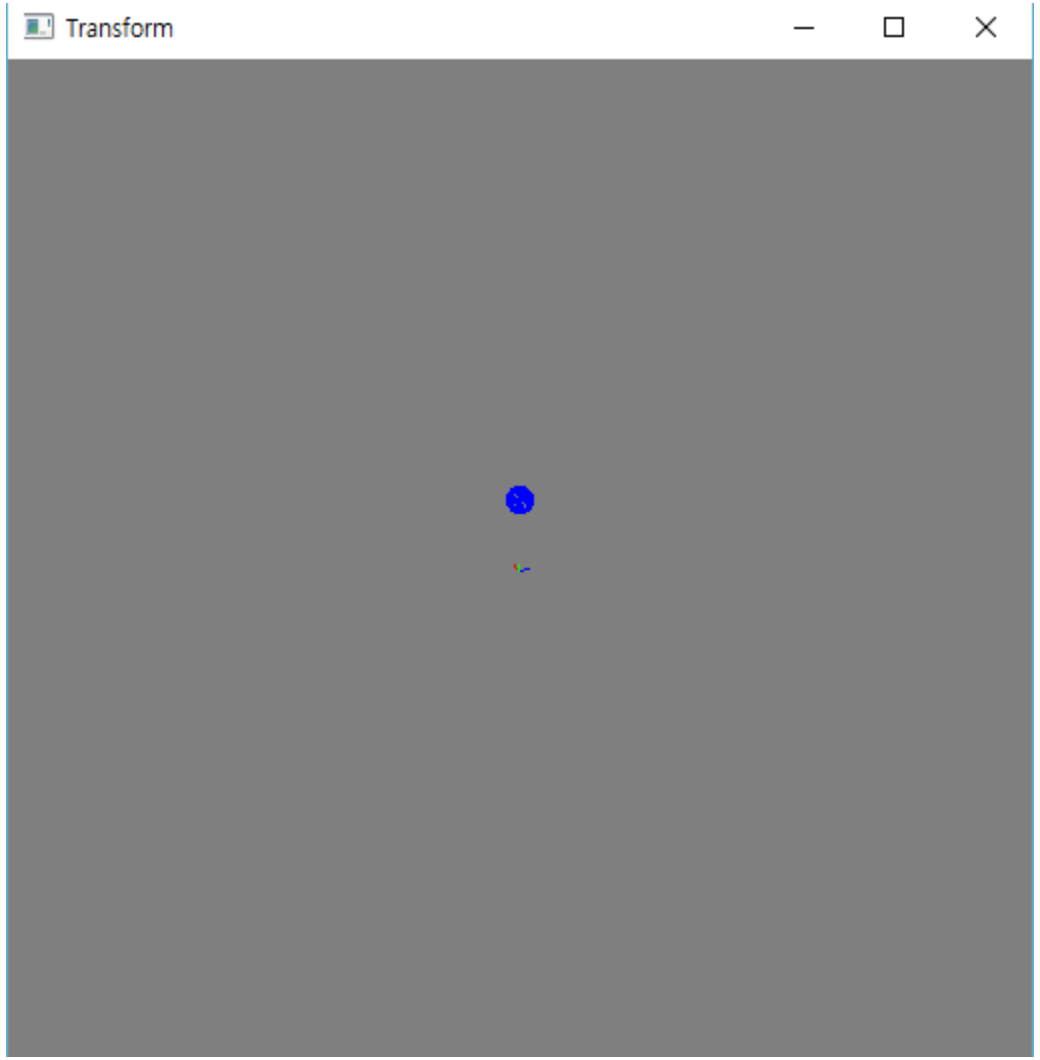
```
void CBulletWorld::release() {
	mWorld->removeRigidBody(ballRigidBody);
	delete ballRigidBody->getMotionState();
	delete ballRigidBody;

	mWorld->removeRigidBody(groundRigidBody);
	delete groundRigidBody->getMotionState();
	delete groundRigidBody;


	delete ballShape;

	delete groundShape;


	delete mWorld;
	delete solver;
	delete collisionConfiguration;
	delete dispatcher;
	delete broadphase;
}
```


Transform