Rigid Body Simulation

Outline

- Introduction
- Concrete Goals
- Result and Analysis
- Conclusion

Introduction

Why we choose rigid body simulation?

- The importance and useful of rigid body

The basic goals of our project

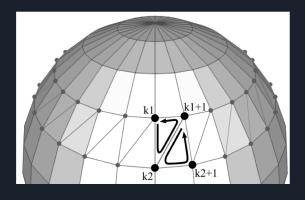
- Visualize basic rigid bodies in 3D scene.
- Simulate the rigid bodies' collision and rotation
- Implement the algorithm in "Nonconvex Rigid Bodies with Stacking"

Concrete Goals

- Geometric Representation
- Visualization
- Collision Model
- Main Algorithm

Geometric Representation

Sphere



- Generate vertices on the surface of the sphere.
- In horizontal direction, every two vertices and center generate same angle.
- Same as vertical direction
- Using indices of vertex to save the memory

Interference Detecting

Paper method: Check some sample vertices of one object in another one or not.

Pros: Can be generalize to irregular rigid bodies

Cons: Need more calculation

Our method: Applying geometric method to detecting intersect or not

Pros: Easy to compute

Cons: Cannot generalize to other object

Collision Model

- Check every pair of objects in the scene.
- Using position of next time step x' to check interference. Where $x' = x + \Delta t(v + g\Delta t)$
- Applying collision impulse to current velocity v
- Update the velocity

Result

Bouncing ball

```
j < sectorCount; ++j, ++k1, ++k2)
                                                                                                            "/Users/yixinlong/Desktop/CS230/Projec
Simulation/CS230-RigidBodySimulation/f:
                                                                                                    26 glm::mat4 projection = glm::perspective(gl
ces(k1, k2, k1+1); // k1---k2---k1+1
                                                                                                       glm::mat4 view = glm::lookAt(
                                                                                                                                 glm::vec3(20.0, 20
                                                                                                                                 glm::vec3(0.0f, 0.0
ackCount-1))
ces(k1+1, k2, k2+1); // k1+1---k2---k2+1
                                                                                                   35 void processInput(GLFWwindow *window);
                                                                                                       GLFWwindow* init();
(float x, float y, float z)
                                                                                                    41 int main(int argc, const char * argv[]) {
                                                                                                            std::vector<Object*> objects;
                                                                                                            objects.push_back(new Sphere(1, true, )
s(float x, float y, float z)
                                                                                                            objects.push_back(new Plane(50));
                                                                                                            int num_object = objects.size(); \( \triangle \) In
t(Object* object)
                                                                                                            std::cout<<"length:" <<num object<<std
```

Result

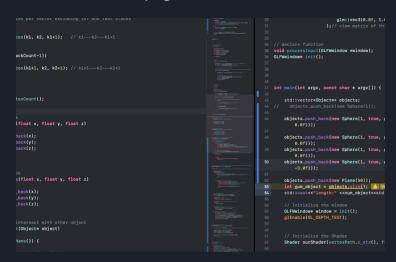
Ball Collisions

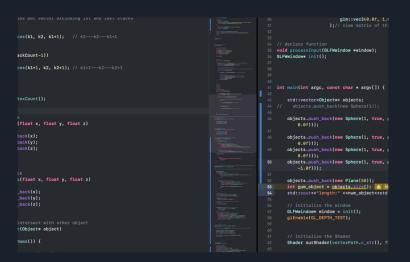
```
glm::mat4 view = glm::lookAt(
j < sectorCount; ++j, ++k1, ++k2)
                                                                                                                      glm::vec3(20.0, 20
                                                                                                                      glm::vec3(0.0f, 0.0
ces(k1, k2, k1+1); // k1---k2---k1+1
                                                                                           35 void processInput(GLFWwindow *window);
ackCount-1))
                                                                                              GLFWwindow* init();
ces(k1+1, k2, k2+1); // k1+1---k2---k2+1
                                                                                              int main(int argc, const char * argv[]) {
                                                                                                  std::vector<Object*> objects;
                                                                                                   objects.push_back(new Sphere(1, true, )
(float x, float y, float z)
                                                                                                   objects.push_back(new Sphere(1, true, )
                                                                                                   objects.push_back(new Plane(50));
                                                                                                   int num_object = objects.size(); A In
                                                                                                  std::cout<<"length:" <<num_object<<std
                                                                                                   GLFWwindow* window = init();
                                                                                                   glEnable(GL_DEPTH_TEST);
t(Object* object)
```

```
glm::vec3(0.0f, 1.
ces(k1, k2, k1+1); // k1---k2---k1+1
                                                                                            35 void processInput(GLFWwindow *window);
ackCount-1))
                                                                                            36 GLFWwindow* init();
ces(k1+1, k2, k2+1); // k1+1---k2---k2+1
                                                                                            41 int main(int argc, const char * argv[]) {
                                                                                                   std::vector<Object*> objects;
                                                                                                   objects.push_back(new Sphere(1, true,
(float x, float y, float z)
                                                                                                   objects.push_back(new Sphere(1, true,
s(float x, float y, float z)
                                                                                                   objects.push_back(new Plane(50));
                                                                                                   int num_object = objects.size(); A In
                                                                                                   std::cout<<"length:" <<num_object<<std
                                                                                                   GLFWwindow* window = init();
                                                                                                   glEnable(GL_DEPTH_TEST);
t(Object* object)
Name()) {
                                                                                                   Shader ourShader(vertexPath.c str(), f:
```

Result

Momentum Propagation





Analysis

Bugs

```
int main(int argc, const char * argv[]) {
                                                                                                   std::vector<Object*> objects;
(this->getVelocity(), ret_vec3);
(object->getVelocity(), -ret_vec3);
"<<u1<<"u2:"<<u2<<std::endl;
locity_u - 2 * u1 * ret_vec3;
= this->getMass() + object->getMass();
his->getMass() - object->getMass();
m2/mass_sum) * u1 + (2*object->getMass()/mass_sum) * u2;
his->getMass()/mass_sum) *u1 + (-m1_m2/mass_sum) * u2;
                                                                                                   objects.push_back(new Sphere(1, true,
ec3:
                                                                                                    objects.push_back(new Sphere(1, true,
eVelocity(glm::vec3 update_v)
n glm::vec3();// if this sphere not moveable, return zero vector
                                                                                                   int num_object = objects.size(); A In
                                                                                                   std::cout<<"length:" <<num_object<<std
r = glm::normalize(update_v);
```

```
void processInput(GLFWwindow *window);
3 = dynamic_cast<Plane*>(object)->getNormal();
                                                                                                 GLFWwindow* init();
(this->getVelocity(), ret vec3);
                                                                                                     std::vector<Object*> objects;
(object->getVelocity(), -ret_vec3);
                                                                                                     objects.push_back(new Sphere(1, true, )
                                                                                                     objects.push_back(new Sphere(1, true, )
                                                                                                     objects.push_back(new Sphere(1, true, )
locity_u - 2 * u1 * ret_vec3;
                                                                                                    objects.push_back(new Sphere(1, true,
= this->getMass() + object->getMass();
                                                                                                     objects.push_back(new Plane(50));
his->getMass() - object->getMass();
                                                                                                     int num_object = objects.size(); (A) In
                                                                                                    std::cout<<"length:" <<num_object<<std
n2/mass_sum) * u1 + (2*object->getMass()/mass_sum) * u2;
his->getMass()/mass_sum) *u1 + (-m1_m2/mass_sum) * u2;
locity u - u1 * ret_vec3 - v1 * ret_vec3;// update self v
                                                                                                     GLFWwindow* window = init();
                                                                                                    glEnable(GL_DEPTH_TEST);
                                                                                                     Shader ourShader(vertexPath.c_str(), f:
eVelocity(glm::vec3 update_v)
                                                                                                     unsigned int VAOs[num_object], VBOs[num_object]
```

Analysis

Bugs

```
3 = dynamic_cast<Plane*>(object)->getNormal();
                                                                                                 glm::mat4 view = glm::lookAt(
                                                                                                                          glm::vec3(20.0, 20
                                                                                                                         glm::vec3(0.0f, 0.1
                                                                                                                         glm::vec3(0.0f, 1.1
(this->getVelocity(), ret_vec3);
                                                                                              36 void processInput(GLFWwindow *window);
(object->getVelocity(), -ret_vec3);
                                                                                              37 GLFWwindow* init();
"<<u1<<"u2:"<<u2<<std::endl;
                                                                                                 int main(int argc, const char * argv[]) {
                                                                                                     std::vector<Object*> objects;
locity_u - 2 * u1 * ret_vec3;
                                                                                                     objects.push_back(new Sphere(1, true, )
= this->getMass() + object->getMass();
                                                                                                      objects.push_back(new Sphere(1, true, )
his->getMass() - object->getMass();
n2/mass_sum) * u1 + (2*object->getMass()/mass_sum) * u2;
his->getMass()/mass sum) *u1 + (-m1 m2/mass sum) * u2;
locity_u - u1 * ret_vec3 - v1 * ret_vec3;// update self v
SILON) * ret_vec3;
                                                                                                     objects.push_back(new Plane(70));
                                                                                                     int num_object = objects.size(); \( \triangle \) In
                                                                                                     std::cout<<"length:" <<num_object<<std
eVelocity(glm::vec3 update_v)
                                                                                                      GLFWwindow* window = init();
```

Object Stacking

Conclusion

• Failed in contact resolution part. We only considered elastic collision and update velocity in a naive way, fail to take force into consideration. This makes us have to rewrite most part of code to add contact part.

 Failed in rotation part. If project could start over, we would think more carefully and thoroughly before code. Questions?