Real-Time Animation Assignment 2, Particle System

Jinwei Yuan 17306137

Tools used:
Opengl, glfw,glew, assimp, mesh.h&model.h(used for model loading ,from modern opengl tutorials https://github.com/SonarSystems/Modern-OpenGL-Tutorials/tree/master/%5BMODEL%20LOADING%5D/%5B16%5D%20Model%20Class & https://github.com/SonarSystems/Modern-OpenGL-Tutorials/tree/master/%5BMODEL%20LOADING%5D/%5B16%5D%20Model%20Class)

Summary:
In this assignment, I build a snow particle system program which applies a drag (from left to right) force, a resistance force (opposite direction to Gravity) and Gravity to snow particles.

In addition, I also build a particle system with plane collision, which applies a drag (from left to right) force, and Gravity to the particles, there are two planes one is the floor(z=0), the other one is the right edge of the window.

Demo Link:
https://youtu.be/zUeKOsCjaEU

Some Codes for update:

```
glm::vec3 Particle::update(){
    glm::vec3 nForce = glm::vec3(0.0f);

    // particle velosity
    float currentGameTime = float(glfwGetTime());
    float time_delta =  float( currentGameTime- time_old);
    // sum up forces get nForce
    std::for_each(forces.begin(), forces.end(), [&] (glm::vec3 f){
        nForce += f;
    });
```

```
    if(planeDetect()==true){
        glm::vec3 normal = glm::vec3(0.0f,1.0f,0.0f);

        nForce+= -0.7f*glm::dot(nForce,normal) * normal;
        // vel = vel_tangential - k*vel_normal_direction
//        vel = glm::vec3(vel.x,0.0f,vel.z) - 0.80000f*glm::vec3(0.0f,vel.y,vel.z);
        vel = -0.0f*vel;

    }
```

or

```
    if(collisionDetect()==true){
        glm::vec3 normal = glm::vec3(0.0f,1.0f,0.0f);

        // vel = vel_tangential - k*vel_normal_direction
        vel = glm::vec3(vel.x,0.0f,vel.z) - 0.80000f*glm::vec3(0.0f,vel.y,vel.z);
//        vel = -1.0f*vel;

    }
    if(collisionDetectRight()==true){
        glm::vec3 normal = glm::vec3(-1.0f,0.0f,0.0f);

        // vel = vel_tangential - k*vel_normal_direction
        vel = glm::vec3(0.0f,vel.y,vel.z) - 0.80000f*glm::vec3(vel.x,0.0f,vel.z);
//        vel = -1.0f*vel;

    }
```

for snow

for plane collision

```
    vel += time_delta*(nForce /mass);

    //  update position Pos += vel + a
    position += time_delta*vel ;
    // update old position
    time_old = currentGameTime;
    //nForce = glm::vec3(0.0);
    return position;
}
```

Example screenshots from results of a single particle with plane collision: