作业8

Rope类的构造函数
显示/半隐式欧拉法
显式Verlet

Rope类的构造函数

```
Rope::Rope(Vector2D start, Vector2D end, int num_nodes, float node_mass, float k, vector<int> pinned_nodes)
    {
        // TODO (Part 1): Create a rope starting at `start`, ending at `end`, and containing `num_nodes` nodes.

//        Comment-in this part when you implement the constructor
//        for (auto &i : pinned_nodes) {
//            masses[i]->pinned = true;
//        }
        Vector2D step = (end - start)/(num_nodes-1);
        for(int i = 0;i<num_nodes;i++)
        {
            Mass *mass = new Mass(start+i*step,node_mass,false);
            mass->velocity = Vector2D(0.f,0.f);
            masses.push_back(mass);
            if(i!=0)
                springs.push_back(new Spring(masses[i-1],masses[i],k));
        }
        for (auto &i : pinned_nodes)
            masses[i]->pinned = true;
    }
```

显示/半隐式欧拉法

```
void Rope::simulateEuler(float delta_t, Vector2D gravity)
    {
        for (auto &s : springs)
        {
            // TODO (Part 2): Use Hooke's law to calculate the force on a node

            Vector2D dis = s->m2->position-s->m1->position;
            Vector2D f = -s->k*dis.unit()*(dis.norm()-s->rest_length);
            s->m1->forces -= f;
            s->m2->forces += f;
        }

        for (auto &m : masses)
        {
            if (!m->pinned)
            {
                // TODO (Part 2): Add the force due to gravity, then compute the new velocity and position

                // TODO (Part 2): Add global damping

                m->forces += gravity*m->mass;
                float k_d = 0.01f;
                Vector2D f_d = -k_d * m->velocity;
                m->forces += f_d;
                Vector2D a = m->forces/m->mass;
                m->velocity += a * delta_t;
                m->position += m->velocity * delta_t;

            }
            // Reset all forces on each mass
            m->forces = Vector2D(0, 0);
        }
    }
```

- 笔者测试了隐式欧拉法和显式欧拉法，发现显式的绳子飞出去了，而半隐式欧拉法很正常，所以笔者采用半隐式欧拉法

显式Verlet

```
void Rope::simulateVerlet(float delta_t, Vector2D gravity)
    {
        for (auto &s : springs)
        {
            // TODO (Part 3): Simulate one timestep of the rope using explicit Verlet （solving constraints)
            Vector2D dis = s->m2->position-s->m1->position;
            Vector2D f = -s->k*dis.unit()*(dis.norm()-s->rest_length);
            s->m1->forces -= f;
            s->m2->forces += f;
        }

        for (auto &m : masses)
        {
            if (!m->pinned)
            {
                m->forces += gravity*m->mass;
                Vector2D a = m->forces/m->mass;
                Vector2D temp_position = m->position;
                // TODO (Part 3.1): Set the new position of the rope mass
                float damping_factor = 0.00005f;
                // TODO (Part 4): Add global Verlet damping

                m->position += (1-damping_factor)*
                    (m->position-m->last_position)+a*delta_t*delta_t;
                m->last_position = temp_position;
            }
            m->forces = Vector2D(0, 0);
        }
    }
```

- 按照公式完成即可

至此，作业8完成！