Assignment 7: 弹簧质点系统

Computer Graphics Teaching Stuff, Sun Yat-Sen University

- Due Date：12月13号晚上12点之前，提交到zhangzk3@mail2.sysu.edu.cn邮箱。

作业总览

1.1 连接绳子的约束

在 rope.cpp 中, 实现 Rope 类的构造函数。这个构造函数应该可以创建一个新的绳子 (Rope) 对象，该对象从 start 开始， end 结束，包含 num_nodes 个节点。也就是如下图所示：

```
Rope(start, end, 3, ...)

  0------------0-------------0
  ^                         ^
start                      end
```

每个结点都有质量，称为质点；质点之间的线段是一个弹簧。通过创建一系列的质点和弹簧，你就可以创建一个像弹簧一样运动的物体。

Rope::Rope()的参数pinned_nodes 保存着结点的索引，这些索引对应结点的固定属性 (pinned attribute) 应该设置为真（他们是静止的，如绳子的末端节点）。对于每一个结点，你应该构造一个Mass对象（质点对象），并在 Mass 对象的构造函数里设置质量和固定属性。（请仔细阅读代码，确定传递给构造函数的参数）。你应该在连续的两个结点之间创建一个弹簧，设置弹簧两端的结点索引和弹簧系数 k，请检查构造函数的签名以确定传入的参数。

运行./ropesim。你应该可以看到屏幕上画出绳子，但它不发生运动。

1.2 显式/半隐式欧拉法

胡克定律表示弹簧连接的两个质点之间的力和他们之间的距离成比例。也就是：

$$\mathbf{f}_{\mathbf{b}\to\mathbf{a}} = -k_s \frac{\mathbf{b}-\mathbf{a}}{||\mathbf{b}-\mathbf{a}||}(||\mathbf{b}-\mathbf{a}|| - l)$$

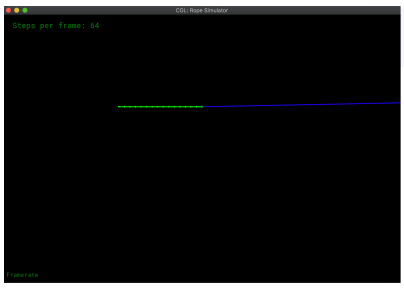
在 Rope::simulateEuler 中, 首先实现胡克定律。遍历所有的弹簧，对弹簧两端的质点施加正确的弹簧力。保证力的方向是正确的！对每个质点，累加所有的弹簧力。

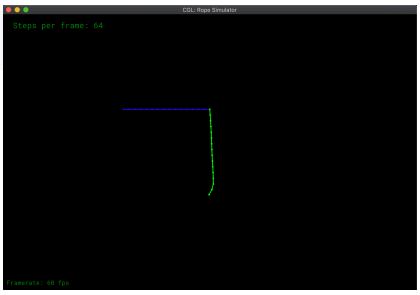
一旦计算出所有的弹簧力，对每个质点应用物理定律：

```
F=ma
v(t+1) = v(t) + a(t) * dt
x(t+1) = x(t) + v(t) * dt   // For explicit method
x(t+1) = x(t) + v(t+1) * dt // For semi-implicit method
```

运行./ropesim。仿真应该就开始运行了，但是只有 3 个结点，看起来不够多。在 application.cpp 文件的最上方，你应该可以看到欧拉绳子和 Verlet 绳子的定义。改变两个绳子结点个数（默认为 3 个），比如 16 或者更多。

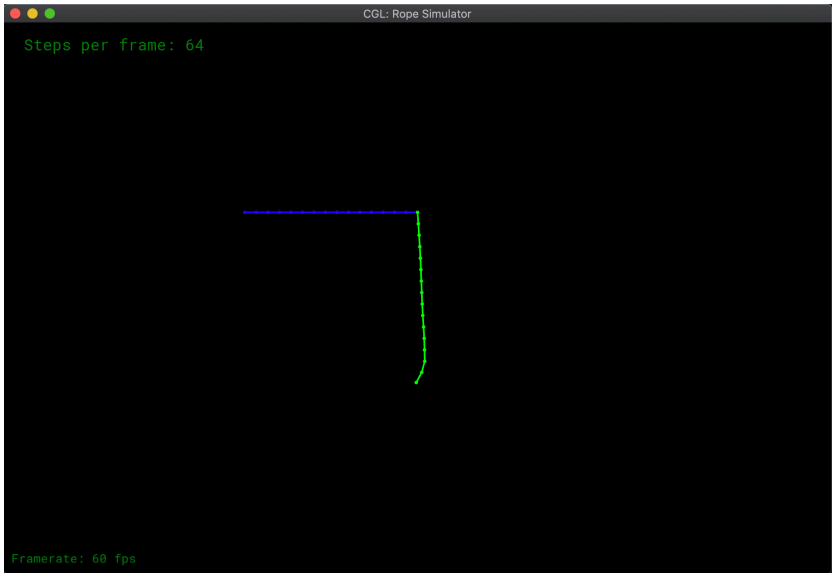
运行 ./ropesim -s 32 来设置仿真中每帧不同的仿真步数。尝试设置较小的值和较大的值（默认值为 64）。





1.3 显式 Verlet

Verlet 是另一种精确求解所有约束的方法。这种方法的优点是只处理仿真中顶点的位置并且保证四阶精度。和欧拉法不同， Verlet 积分按如下的方式来更新下一步位置：



利用这种方法，我们可以仿真弹簧系数无限大的弹簧。

在本次作业中，上述公式的$a(t)$特指重力加速度，不包含弹簧力加速度。具体实现时，首先利用上述公式计算重力影响下，质点的位置变化。然后用解约束的方法来进一步更新质点位置：简单地移动每个质点的位置使得弹簧的长度保持原长。假设弹簧当前长度和稳定状态长度的差值L，每个质点的修正向量长度应为$L/2$，方向为一个质点指向另一质点。

只要对每个弹簧执行这样的操作，我们就可以得到稳定的仿真。为了使运动更加平滑，每一帧可能需要更多的仿真次数。

1.4 阻尼

向显式 Verlet 方法积分的胡克定律中加入阻尼。现实中的弹簧不会永远跳动-因为动能会因摩擦而减小。阻尼系数设置为 0.00005, 加入阻尼之后质点位置更新如下：

```
x(t+1) = x(t) + (1 - damping_factor) * [x(t) - x(t-1)] + a(t) * dt * dt
```

1.5 说明

你应该修改的函数是:

- rope.cpp 中的 Rope::rope(...)
- rope.cpp 中的 void Rope::simulateEuler(...)
- rope.cpp 中的 void Rope::simulateVerlet(...)

程序实现

2.1 安装依赖

本次作业需要预先安装 OpenGL, Freetype 还有 RandR 这三个库。可以通过以下命令进行安装:

```
$sudo apt install libglu1-mesa-dev freeglut3-dev mesa-common-dev
$sudo apt install xorg-dev
```

2.1 编译、运行程序

请下载工程的代码框架并通过下面的命令创建工程:

```
$ mkdir build
$ cd build
$ cmake ..
$ make
```

之后，你应该可以使用命令./ropesim 来运行仿真。

作业描述与提交

3.1 作业提交

将PDF报告文件和代码压缩打包提交到zhangzk3@mail2.sysu.edu.cn邮箱，邮件及压缩包命名格式为：hw7_姓名_学号。

3.2 作业描述

Task1：完善rope.cpp 中的 Rope::rope(...)。 （20分）

完善该构造函数，运行程序并截图。

Task2：完善rope.cpp 中的 Rope::simulateEuler(...)（40分）

- 在application.cpp文件内将欧拉绳子的质点数量修改为16。
- 分别实现显式欧拉法、半隐式欧拉法，附上代码，运行程序并截图。
- 比较在不同的步数下（如16,64,256,1024），欧拉绳子的摆动情况。
- 比较阻尼系数为0, 0.001,0.01,0.1下，欧拉绳子的摆动情况。

注意：此处，阻尼力使用简化的做法，k_d为阻尼系数：

```
f_damping = - k_d * velocity
```

Task3：完善rope.cpp 中的 Rope::simulateVerlet(...)（40分）

- 在application.cpp文件内将Verlet绳子的质点数量修改为16。
- 实现显式Verlet法，附上代码，运行程序并截图。
- 比较在不同的步数下（如16,64,256,1024），Verlet绳子的摆动情况。
- 比较阻尼系数为0, 0.00005,0.0005,0.005下，Verlet绳子的摆动情况。