VCL lab4 报告

唐一早 2100012613

Task 1: Inverse Kinematics

问题 1：如果目标位置太远，无法到达，IK 结果会怎样?

会努力伸直朝向那个点的方向，但是够不到。

问题 2：比较 CCD IK 和 FABR IK 所需要的迭代次数。

FABR IK 迭代次数 CCD IK 少，速度更快。 打印出迭代次数会发现 CCD 几乎在 maxCCDIKIteration 内都还没收敛，而 FABR 几乎在个位数迭代次数就收敛了。

问题 3:（选做，只提供大概想法即可）由于 IK 是多解问题，在个别情况下，会出现前后两帧关节 旋转抖动的情况。怎样避免或是缓解这种情况?

增加帧率，在做迭代计算的时候限制每次的旋转角度。

实现:

Sub-Task 1 (0.5'):

首先我们计算得到所有关节的全局旋转 $q0 = q^{local\ 0}$, $q1 = q0q^{local\ 1}$, $q2 = q1q^{local\ 2}$. 之后我们计算通过应用全局旋转后，关节相对于其父关节的位移 $l_{0\sim1} = q0l_{0\sim1}\, q0*$, $l'_{1\sim2} = q1l_{1\sim2}q_1 \star$, $l'_{2\sim3} = q2l_{2\sim3}q2\star$. 最后我们将位移顺次相加，得到 $P1 = P0+l'_{0\sim1}$, $P2 = P1+l'_{1\sim2}$, $P3 = P2 + l'_{2\sim3}$.

Sub-Task 2 (1'):

每个骨骼都以自身轴点到尾叶子节点的方向旋转到自身轴点到目标点方向趋近, 每个节点完成旋转后，forward 更新。

Sub-Task 3 (1'):

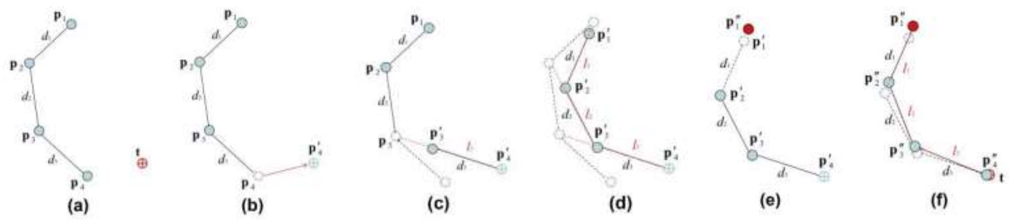
1.先从末端骨骼开始计算，先将最末端的骨骼移到目标位置处。

2.将其关节点和结尾'连成一条直线，通过原有的关节长度，将现在的关节点拉到与末端同样的距离处。

3.以此类推，一直处理到根骨骼。

4.再从根骨骼开始处理。由于根骨骼再整个迭代过程中是默认为不动的，因此再把根骨骼移到原来的位置处，接着使用同样的距离约束，一直处理到尾骨。

5.重复 2~5 的迭代过程，直到最终的尾骨骼位置与到达目标位置（或与目标位置距离小于某个预定值）停止。此时整个算法结束。

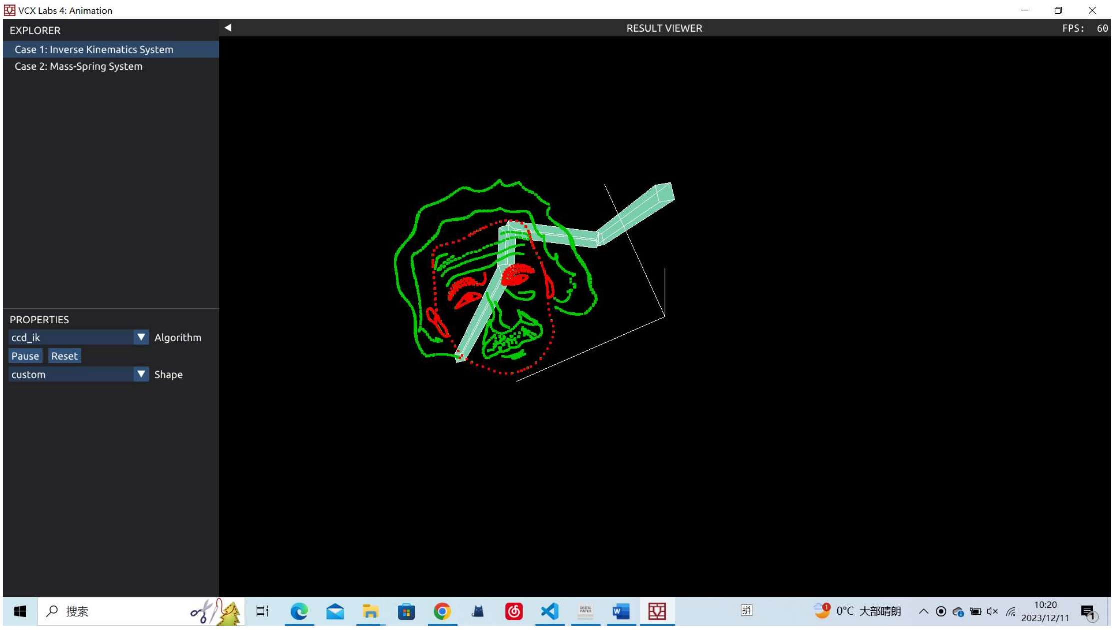


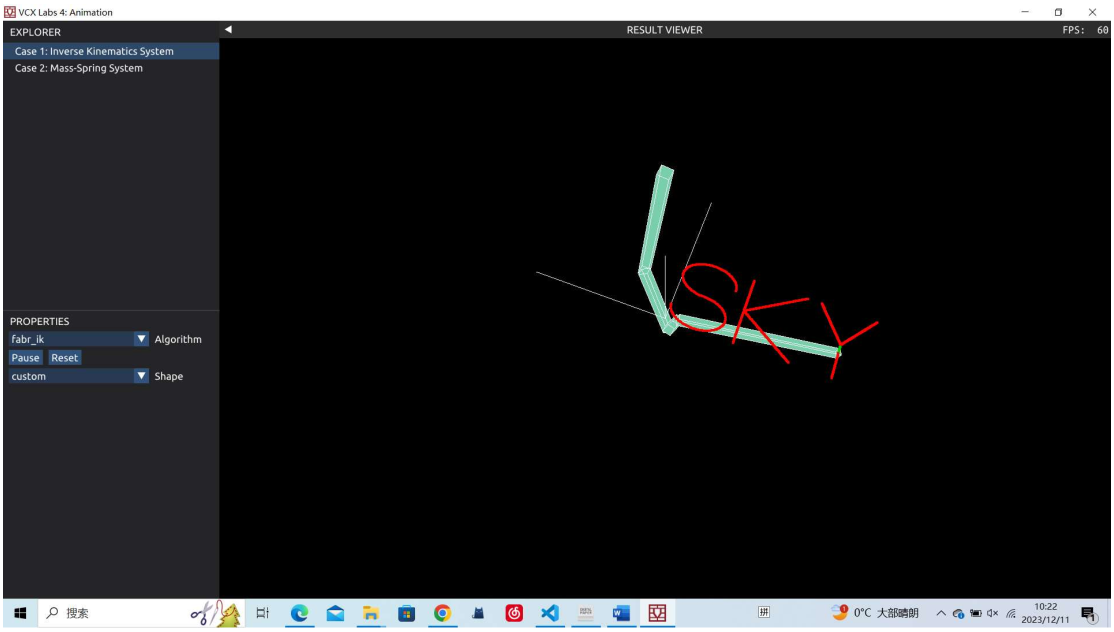
Sub-Task 4 (0.5'):

画了我的英文名大写的 SKY

Sub-Task 4.1 (bonus=0.5')

计算两个采样点之间距离，如果大于某个阈值，就在两个点之间按照阈值重采样，这样可以让图像更加均匀。





Task 2: Mass-Spring System (3')

实现：

Eigen::VectorXf grav(MassSpringSystem & system) 计算重力加速度

Eigen::VectorXf damp_force(MassSpringSystem & system, Eigen::VectorXf const & x, Eigen::VectorXf const & v)计算阻尼，并把阻尼视为外力，与位置无关量。

Eigen::VectorXf grad_E(MassSpringSystem & system, Eigen::VectorXf const & x)计算$\nabla E(x)$

$$\mathbf{f}_{ij} = -\nabla_i E_{ij} = k_{ij}\left(\|\mathbf{x}_j - \mathbf{x}_i\| - l_{ij}\right)\frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} = k_{ij}\left(\|\mathbf{x}_j - \mathbf{x}_i\| - l_{ij}\right)\mathbf{n}_{ij}$$

Eigen::VectorXf y(MassSpringSystem & system, Eigen::VectorXf const & x, Eigen::VectorXf const & v, float const ddt)计算 y，其中 fext 包括重力和阻尼。

$$\mathbf{y}^k = \mathbf{x}^k + h\mathbf{v}^k + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$$

static Eigen::VectorXf grad_g(MassSpringSystem & system, Eigen::VectorXf const & x, Eigen::VectorXf const & v, float const ddt)计算$\nabla g(x)$。

$$\nabla g(\mathbf{x}^k) = \frac{1}{h^2}\mathbf{M}(\mathbf{x}^k - \mathbf{y}^k) + \nabla E(\mathbf{x}^k)$$

Eigen::SparseMatrix<float> Hessian(MassSpringSystem & system, Eigen::VectorXf const & x, float const ddt)计算海森矩阵，其中对于每个弹簧连接，把 ij 项和 ji 项直接加入稀疏矩阵，ii 项和 jj 项累加进入另一个存有对角块的 diagonal 中，最后将 diagonal 加入稀疏矩阵，得到完整的海森矩阵。

$$\mathbf{H}_g(\mathbf{x}^k) = \frac{1}{h^2}\mathbf{M} + \mathbf{H}(\mathbf{x}^k)$$

$$\mathbf{H}_e := \frac{\partial^2 E_{ij}(\mathbf{x}^k)}{\partial \mathbf{x}_i^2} = k_{ij}\frac{(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top}{\|\mathbf{x}_i - \mathbf{x}_j\|^2} + k_{ij}\left(1 - \frac{l_{ij}}{\|\mathbf{x}_i - \mathbf{x}_j\|}\right)\left(\mathbf{I} - \frac{(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top}{\|\mathbf{x}_i - \mathbf{x}_j\|^2}\right)$$

$$\mathbf{H}_{ij}(\mathbf{x}^k) = \begin{bmatrix} & \vdots & & \vdots & \\ \cdots & \mathbf{H}_e & \cdots & -\mathbf{H}_e & \cdots \\ & \vdots & & \vdots & \\ \cdots & -\mathbf{H}_e & \cdots & \mathbf{H}_e & \cdots \\ & \vdots & & \vdots & \end{bmatrix}$$

$$\mathbf{H}(\mathbf{x}^k) = \sum_{(i,j)} \mathbf{H}_{ij}(\mathbf{x}^k)$$

得到海森矩阵，和$\nabla g(x)$后，求解得到更新的 x，并用更新的 x 计算 v

$$\mathbf{H}_g(\mathbf{x}^k)(\mathbf{x}^{k+1} - \mathbf{x}^k) = -\nabla g(\mathbf{x}^k)$$

$$\mathbf{v}^{k+1} = \mathbf{v}^k + h\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^{k+1})$$

最终结果：