基于物理的动画：刚体动力学

Physics-based Animation: Rigid Body Dynamics

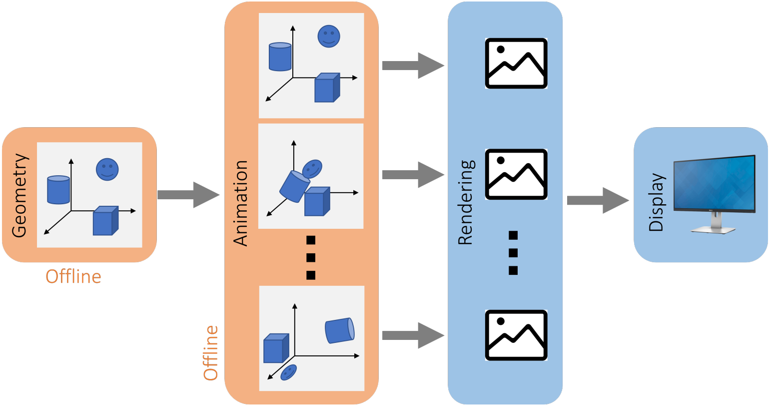
星海

光线云

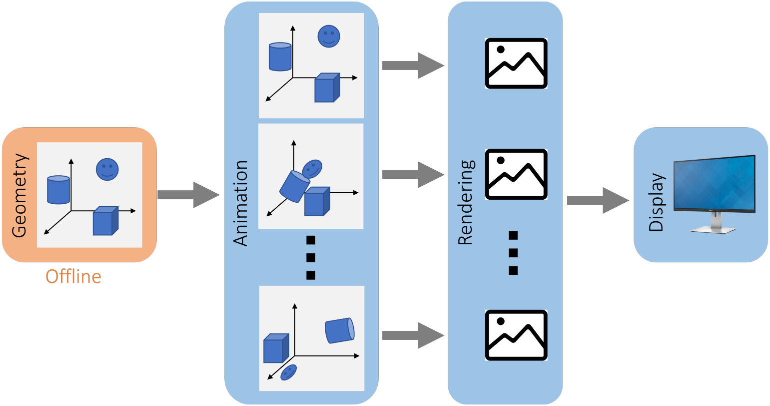
2024/10/11

目录

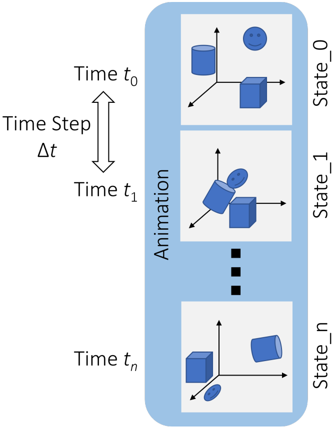
动画流水线 – 基于数据的动画



动画流水线 - 实时物理仿真



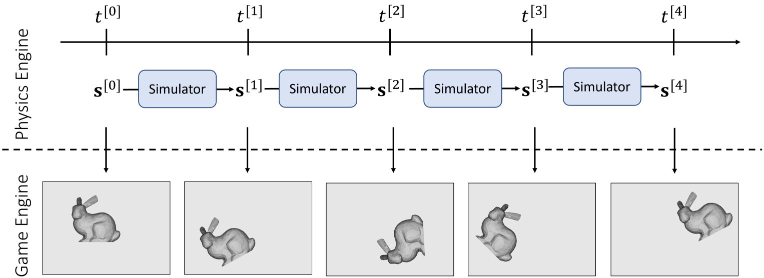
每帧的动画计算



- 动画是每帧对场景物体状态，包括视觉状态和非视觉的状态的更新
- 常见的视觉状态包括
  - 位置
  - 朝向（旋转）
  - 外观
- 常见的非视觉的物理状态包括
  - 外力
  - 速度
  - 密度
  - ...
- 离线动画应用仅需存储视觉状态，物理仿真算法则按需存储其他物理状态

物理仿真流水线

设 $t^{(k)}$ 为第 k 帧的时刻，$\boldsymbol{s}^{(k)}$ 为仿真对象在第 k 帧时的状态。仿真器在每一帧执行物理演算，并将结果提交给渲染器。



目录

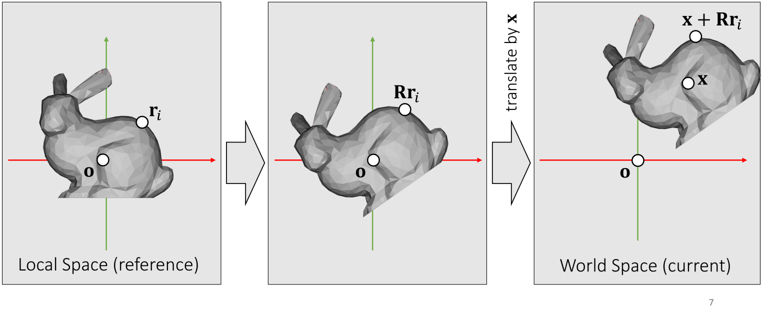
刚体的性质



- 刚体是不会产生形状、大小改变的理想物体。只有整体的平动和转动。
- 一般地，物理引擎存储的刚体仿真状态 $\boldsymbol{s} = \{\boldsymbol{v}, \boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{q}\}$ 是一个四元组，其中 $\boldsymbol{v}$ 为刚体的线速度，$\boldsymbol{x}$ 为位移，$\boldsymbol{\omega}$ 为角速度，$\boldsymbol{q}$ 为表示朝向的四元数。
- 记 n 为刚体的顶点数。如果刚体的各顶点质量不一，记顶点 i 的质量为 m_i，则 $M = (m_1, \ldots, m_n)^T$，记 $M^{-1} = (1/m_1, \ldots, 1/m_n) \in \mathbb{R}^{1\times n}$。此时我们需要让 $\boldsymbol{f} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n)^T$ 分别储存各顶点的受力情况。

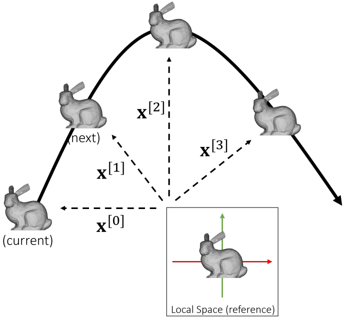
刚体的运动

刚体整体不缩放，只发生平动 $\boldsymbol{x}$（translation）和转动 $\boldsymbol{R}$（rotation）变换。



目录

平动动力学



根据牛顿运动定律：

$$\begin{cases} \boldsymbol{v}(t^{(1)}) = \boldsymbol{v}(t^{(0)}) + M^{-1} \int_{t^{(0)}}^{t^{(1)}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{v}, t)\,\mathrm{d}t \\ \boldsymbol{x}(t^{(1)}) = \boldsymbol{x}(t^{(0)}) + \int_{t^{(0)}}^{t^{(1)}} \boldsymbol{v}(t)\,\mathrm{d}t \end{cases} \tag{1}$$

进行离散半隐式（semi-implicit）积分得到

$$\begin{cases} \boldsymbol{v}^{(1)} = \boldsymbol{v}^{(0)} + \Delta t M^{-1} \boldsymbol{f}^{(0)}, \\ \boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \Delta t \boldsymbol{v}^{(1)} \end{cases} \tag{2}$$

其中 $\boldsymbol{v}$ 用上一帧的 $\boldsymbol{f}$ 数据更新，是显式积分，$\boldsymbol{x}$ 用当前帧的 $\boldsymbol{v}$ 数据更新，是隐式积分。可以证明，如此更新的 $\boldsymbol{x}$ 相对完全显式法和完全隐式法是二阶精确的。

刚体仿真-仅平动

仿真流程:

1. 输入状态 $\boldsymbol{s}^{(0)} = \{\boldsymbol{v}^{(0)}, \boldsymbol{x}^{(0)}\}$
2. $$\begin{cases} \boldsymbol{f}_i^{(0)} = \text{Force}(\boldsymbol{x}_i^{(0)}, \boldsymbol{v}_i^{(0)}, \dots), \\ \boldsymbol{f}^{(0)} = (\boldsymbol{f}_1^{(0)}, \dots, \boldsymbol{f}_n^{(0)})^T, \\ \boldsymbol{v}^{(1)} = \boldsymbol{v}^{(0)} + \Delta t M^{-1} \boldsymbol{f}^{(0)}, \\ \boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \Delta t \boldsymbol{v}^{(1)} \end{cases} \tag{3}$$
3. 输出状态 $\boldsymbol{s}^{(1)} = \{\boldsymbol{v}^{(1)}, \boldsymbol{x}^{(1)}\}$

注意:

- 各顶点质量 $M = (m_1, \dots, m_n)^T$ 为常向量，表记 $M^{-1} = (1/m_1, \dots, 1/m_n) \in \mathbb{R}^{1\times n}$
- 时间间隔 Δt 可自定义，不一定是常量，也不一定与实际物理帧计算的时间间隔相同

目录

旋转的表达 - 旋转矩阵与欧拉角

在讨论三维旋转时，我们只讨论绕某个过原点的旋转轴旋转。若旋转轴不过原点，需要先进行平移变换。

旋转矩阵：

- 三维空间中的旋转可用一个自由度为 3 的 3×3 矩阵 $\boldsymbol{R}$ 表示
- 对空间向量 $\boldsymbol{v} \in \mathbb{R}^3$，矩阵-向量乘积 $\boldsymbol{Rv}$ 可表示对该向量施加旋转变换
- 缺点：不够直观；自由度低，浪费空间；难以计算旋转相关的速度

欧拉角：

- 三维空间中的旋转亦可用直观的三元组——欧拉角（euler angle）表示
- 给定空间中三个固定正交轴（局部的或全局的），欧拉角是按固定顺序分别绕这些轴的旋转角度的记录。不同的欧拉角三元组，最终可能产生相同的旋转结果
- 缺点：可能丢失自由度（万向节死锁）；难以计算旋转相关的速度

旋转的表达 – 四元数

四元数 $\boldsymbol{q} = a + b\mathrm{i} + c\mathrm{j} + d\mathrm{k} \in \mathbb{H} \leftrightarrow \mathbb{R}^4$ 是对复数域 $\mathbb{C} \leftrightarrow \mathbb{R}^2$ 的扩展，遵从运算法则

$$\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1, \tag{4}$$

我们可以表记四元数 $\boldsymbol{q} = [s, \boldsymbol{v}]$，其中 s 为 $\boldsymbol{q}$ 的标量部分，$\boldsymbol{v}$ 为向量部分。可以推导四元数的标量积、加法和乘法运算：

$$\begin{cases} a\boldsymbol{q} = [as, a\boldsymbol{v}], \\ \boldsymbol{q}_1 + \boldsymbol{q}_2 = [s_1 + s_2, \boldsymbol{v}_1 + \boldsymbol{v}_2], \\ \boldsymbol{q}_1 \times \boldsymbol{q}_2 = \boldsymbol{q}_1\boldsymbol{q}_2 = [s_1 s_2 - \boldsymbol{v}_1 \cdot \boldsymbol{v}_2, s_1\boldsymbol{v}_2 + s_2\boldsymbol{v}_1 + \boldsymbol{v}_1 \times \boldsymbol{v}_2]. \end{cases} \tag{5}$$

特别地，我们可以让三维向量 $\boldsymbol{v}$ 也参与上述计算，只需视其为四元数 $[0, \boldsymbol{v}]$ 即可。

旋转的表达 - 四元数运算

正如一个归一化的复数可以表示二维旋转一样，一个归一化的四元数也可以表示一个三维旋转，它满足

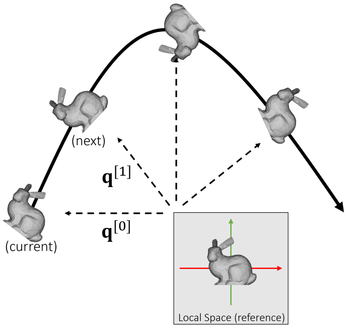
$$||\boldsymbol{q}|| = \sqrt{s^2 + \boldsymbol{v} \cdot \boldsymbol{v}} = 1. \tag{6}$$

具体地，我们让向量 $\boldsymbol{v}$ 绕某个过原点的空间轴 $\boldsymbol{u}$ 旋转 θ 角度，旋转方向使用右手螺旋判断。记

$$\begin{cases} \boldsymbol{q} = \left[\cos\frac{\theta}{2}, \boldsymbol{u}\right], \\ ||\boldsymbol{q}|| = 1, \end{cases} \tag{7}$$

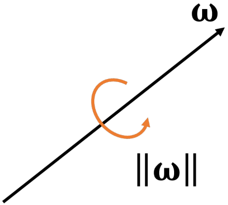
那么，可以证明，$\boldsymbol{q}\boldsymbol{v}\boldsymbol{q}^*$ 就是空间向量 $\boldsymbol{v}$ 如上旋转后的结果的四元数表示，即 $\boldsymbol{q}\boldsymbol{v}\boldsymbol{q}^*$ 的标量部分总是为 0；其中 $\boldsymbol{q}^* = \left[\cos\frac{\theta}{2}, -\boldsymbol{u}\right]$ 为 $\boldsymbol{q}$ 的共轭四元数。归一化的四元数可以与三维旋转形成一一对应。四元数也可以方便地转换为矩阵和欧拉角，因此成为了物理引擎内部旋转运算的常用数据类型。

转动动力学



记 $\boldsymbol{q}$ 为表示刚体旋转的四元数，$\boldsymbol{R}$ 表示该旋转的变换矩阵，$\boldsymbol{\omega} \in \mathbb{R}^3$ 为旋转的角速度，规定：

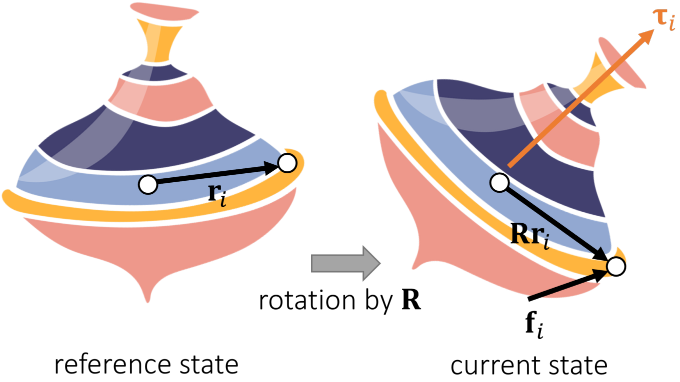
- $\boldsymbol{\omega}$ 的方向为旋转轴（与旋转方向形成右手螺旋关系），
- $\boldsymbol{\omega}$ 的大小为旋转的角速率。



力矩

转动动力学中的力矩（torque）是平动动力学中的力（force）的等效，定义式为

$$\boldsymbol{\tau}_i = (\boldsymbol{R}\boldsymbol{r}_i) \times \boldsymbol{f}_i. \tag{8}$$



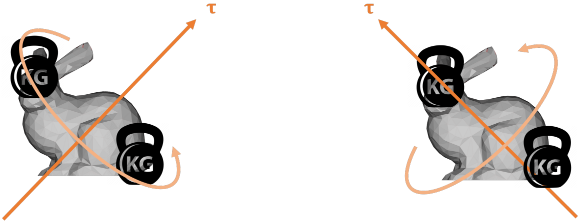
转动惯量

转动动力学中的转动惯量（inertia）是平动动力学中的质量（mass）的等效，是一个 3×3 矩阵。在初始状态下，转动惯量为一常量，定义式为

$$\boldsymbol{I}_{\text{ref}} = \sum m_i(\boldsymbol{r}_i^T \boldsymbol{r}_i \mathbf{1} - \boldsymbol{r}_i \boldsymbol{r}_i^T). \tag{9}$$

在刚体经过旋转变换 $\boldsymbol{R}$ 后，可以证明，此时转动惯量为

$$\boldsymbol{I} = \boldsymbol{R}\boldsymbol{I}_{\text{ref}}\boldsymbol{R}^T. \tag{10}$$



请注意：一个三维空间中的物体存在无数个可能的转动轴，每个转动轴都对应着一个描述其转动惯性大小的标量 I。但这些量并非完全相互独立，它们可以被整理成一个 3×3 的矩阵 $\boldsymbol{I}_{\text{ref}}$。

刚体仿真-仅转动

1. 输入状态 $\boldsymbol{s}^{(0)} = \{\boldsymbol{q}^{(0)}, \boldsymbol{\omega}^{(0)}\}$
2. 更新中间物理状态

$$\begin{cases} \boldsymbol{R}^{(0)} = \text{Matrix.Rotate}(\boldsymbol{q}^{(0)}), \\ \boldsymbol{\tau}_i^{(0)} = (\boldsymbol{R}^{(0)}\boldsymbol{r}_i) \times \boldsymbol{f}_i^{(0)}, \\ \boldsymbol{\tau}^{(0)} = \sum \boldsymbol{\tau}_i^{(0)}, \\ \boldsymbol{I}^{(0)} = \boldsymbol{R}^{(0)}\boldsymbol{I}_{\text{ref}}(\boldsymbol{R}^{(0)})^T, \end{cases} \tag{11}$$

3. 更新刚体仿真状态

$$\begin{cases} \boldsymbol{\omega}^{(1)} = \boldsymbol{\omega}^{(0)} + \Delta t(\boldsymbol{I}^{(0)})^{-1}\boldsymbol{\tau}^{(0)}, \\ \boldsymbol{q}^{(1)} = \boldsymbol{q}^{(0)} + \left[0, \frac{\Delta t}{2}\boldsymbol{\omega}^{(1)}\right] \times \boldsymbol{q}^{(0)} \end{cases} \tag{12}$$

4. 输出状态 $\boldsymbol{s}^{(1)} = \{\boldsymbol{q}^{(1)}, \boldsymbol{\omega}^{(1)}\}$

目录

刚体仿真流水线 I

1. 输入状态 $\boldsymbol{s}^{(0)} = \{\boldsymbol{v}^{(0)}, \boldsymbol{x}^{(0)}, \boldsymbol{\omega}^{(0)}, \boldsymbol{q}^{(0)}\}$
2. 更新中间物理状态

$$\begin{cases} \boldsymbol{f}_i^{(0)} = \mathsf{Force}(\boldsymbol{x}_i^{(0)}, \boldsymbol{v}_i^{(0)}, \dots), \\ \boldsymbol{f}^{(0)} = (\boldsymbol{f}_1^{(0)}, \dots, \boldsymbol{f}_n^{(0)})^T, \\ \boldsymbol{R}^{(0)} = \mathsf{Matrix.Rotate}(\boldsymbol{q}^{(0)}), \\ \boldsymbol{\tau}_i^{(0)} = (\boldsymbol{R}^{(0)} \boldsymbol{r}_i) \times \boldsymbol{f}_i^{(0)}, \\ \boldsymbol{\tau}^{(0)} = \sum \boldsymbol{\tau}_i^{(0)}, \\ \boldsymbol{I}^{(0)} = \boldsymbol{R}^{(0)} \boldsymbol{I}_{\mathsf{ref}} (\boldsymbol{R}^{(0)})^T, \end{cases} \tag{13}$$

刚体仿真流水线 II

3 更新刚体仿真状态

$$\begin{cases} \boldsymbol{v}^{(1)} = \boldsymbol{v}^{(0)} + \Delta t M^{-1} \boldsymbol{f}^{(0)}, \\ \boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \Delta t \boldsymbol{v}^{(1)} \\ \boldsymbol{\omega}^{(1)} = \boldsymbol{\omega}^{(0)} + \Delta t (\boldsymbol{I}^{(0)})^{-1} \boldsymbol{\tau}^{(0)}, \\ \boldsymbol{q}^{(1)} = \boldsymbol{q}^{(0)} + \left[0, \dfrac{\Delta t}{2} \boldsymbol{\omega}^{(1)}\right] \times \boldsymbol{q}^{(0)} \end{cases} \tag{14}$$

4 输出状态 $\boldsymbol{s}^{(1)} = \{\boldsymbol{v}^{(1)}, \boldsymbol{x}^{(1)}, \boldsymbol{\omega}^{(1)}, \boldsymbol{q}^{(1)}\}$

目录

预备知识：奇异值分解与极分解

- 奇异值分解：任何变换均可被依次分解成三个部分：（可能带翻转的）旋转、缩放和（可能带翻转的）旋转。即

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T, \tag{15}$$

其中 $\boldsymbol{A}$ 为任意方阵，$\boldsymbol{U}$ 和 $\boldsymbol{V}^T$ 为正交矩阵，$\boldsymbol{D}$ 为对角矩阵

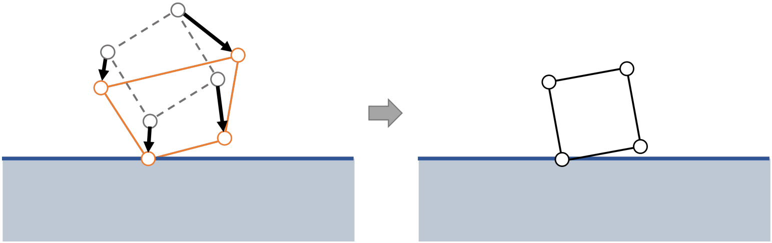
- 极分解：任何变换均可被依次分解成两个部分：（可能带翻转的）旋转和其他变形。即

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T = (\boldsymbol{U}\boldsymbol{V}^T)(\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T) = \boldsymbol{R}\boldsymbol{S}, \tag{16}$$

其中 $\boldsymbol{R}$ 为正交矩阵，$\boldsymbol{S}$ 为半正定对称矩阵

Shape Matching

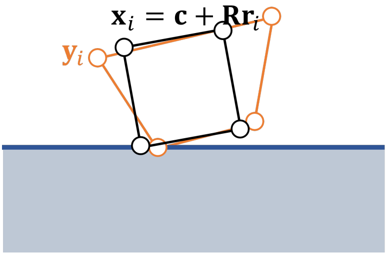
- 基本思想：先对每个顶点做自由模拟，然后再把这些顶点用某种方法变回刚体
- 变回刚体：让各顶点的位置满足刚体的位置约束，并且最小化各顶点在该过程中的移动距离。旋转变换通过极分解（polar decomposition）得到
- Shape Matching 思想与可变形体（弹性体）的仿真方案一脉相承，因此适合在弹性体上实现刚体，如衣服中的纽扣等挂件
- 由于欠缺物理正确性，Shape Matching 不适合处理带摩擦的碰撞情形



Shape Matching - 公式推导

- 记仿真得到的刚体的预期质心为 $\boldsymbol{c}$，顶点 i 的位置为 $\boldsymbol{r}_i$，依质心性质有 $\sum_i \boldsymbol{r}_i = \mathbf{0}$
- 设 $\boldsymbol{y}_i$ 为自由仿真后的顶点位置，$\boldsymbol{x}_i$ 为满足刚体约束的顶点位置，则有 $\boldsymbol{x}_i = \boldsymbol{c} + \boldsymbol{R}\boldsymbol{r}_i$，其中 $\boldsymbol{R}$ 为某旋转矩阵
- 优化目标为

$$\{\boldsymbol{c}, \boldsymbol{R}\} = \operatorname{argmin} \sum_i ||\boldsymbol{c} + \boldsymbol{R}\boldsymbol{r}_i - \boldsymbol{y}_i||^2 \tag{17}$$



Shape Matching - 公式推导

$$\{\boldsymbol{c}, \boldsymbol{R}\} = \text{argmin} \sum_i ||\boldsymbol{c} + \boldsymbol{R}\boldsymbol{r}_i - \boldsymbol{y}_i||^2$$

考虑 $\boldsymbol{A}$ 为任意矩阵，其中的旋转分量为 $\boldsymbol{R}$，那么我们可以优化

$$\{\boldsymbol{c}, \boldsymbol{A}\} = \text{argmin} \sum_i ||\boldsymbol{c} + \boldsymbol{A}\boldsymbol{r}_i - \boldsymbol{y}_i||^2 \tag{18}$$

求导求极值

$$\frac{\partial E}{\partial \boldsymbol{c}} = 0 \leftrightarrow \sum_i \boldsymbol{c} + \boldsymbol{A}\boldsymbol{r}_i - \boldsymbol{y}_i = 0 \tag{19}$$

得到

$$\boldsymbol{c} = \frac{1}{n} \sum_i \boldsymbol{y}_i \tag{20}$$

即变回刚体前后的刚体质心位置不变。

Shape Matching - 公式推导

$$\{\boldsymbol{c}, \boldsymbol{R}\} = \text{argmin} \sum_i ||\boldsymbol{c} + \boldsymbol{R}\boldsymbol{r}_i - \boldsymbol{y}_i||^2$$

另一方面

$$\frac{\partial E}{\partial \boldsymbol{A}} = 0 \leftrightarrow \sum_i (\boldsymbol{c} + \boldsymbol{A}\boldsymbol{r}_i - \boldsymbol{y}_i)\boldsymbol{r}_i^T = 0 \tag{21}$$

得到

$$\boldsymbol{A} = \left(\sum_i (\boldsymbol{y}_i - \boldsymbol{c})\boldsymbol{r}_i^T\right)\left(\sum_i \boldsymbol{r}_i\boldsymbol{r}_i^T\right)^{-1} \tag{22}$$

最后，我们将 $\boldsymbol{A}$ 做极分解

$$\boldsymbol{A} = \boldsymbol{R}\boldsymbol{S} \tag{23}$$

使用其中的旋转部分 $\boldsymbol{R}$ 即可。

Shape Matching 流水线 I

在 Shape Matching 刚体仿真流程中，不再以统一的 $\mathbb{R}^3$ 向量更新刚体的位置、速度等仿真状态，而是单独仿真每个顶点；也不再考虑转动动力学。即此时 $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$，$\boldsymbol{v} = (\boldsymbol{v}_1 \ldots, \boldsymbol{v}_n)^T$。

1. 输入状态 $\boldsymbol{s}^{(0)} = \{\boldsymbol{x}^{(0)}, \boldsymbol{v}^{(0)}\}$
2. 对每个顶点单独仿真

$$\begin{cases} \boldsymbol{f}_i = \mathsf{Force}(\boldsymbol{x}_i^{(0)}, \boldsymbol{v}_i^{(0)}, \ldots), \\ \boldsymbol{v}_i = \boldsymbol{v}_i^{(0)} + \Delta t m_i^{-1} \boldsymbol{f}_i, \\ \boldsymbol{y}_i = \boldsymbol{x}_i^{(0)} + \Delta t \boldsymbol{v}_i, \end{cases} \tag{24}$$

Shape Matching 流水线 II

3 计算刚体位置约束

$$\begin{cases} \boldsymbol{c} = \dfrac{1}{n}\displaystyle\sum_i \boldsymbol{y}_i, \\ \boldsymbol{A} = \left(\displaystyle\sum_i (\boldsymbol{y}_i - \boldsymbol{c})\boldsymbol{r}_i^T\right)\left(\displaystyle\sum_i \boldsymbol{r}_i\boldsymbol{r}_i^T\right)^{-1}, \\ \boldsymbol{R} = \mathrm{Polar}(\boldsymbol{A}), \end{cases} \tag{25}$$

4 更新各顶点仿真状态

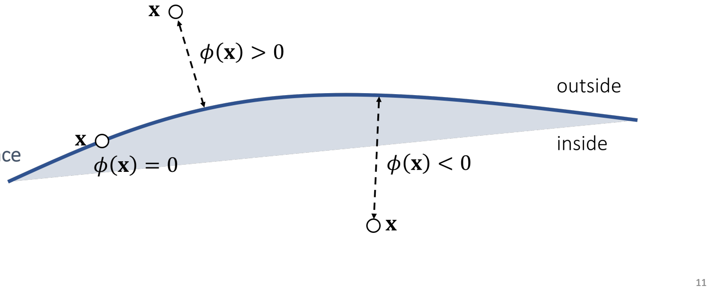
$$\begin{cases} \boldsymbol{x}_i^{(1)} = \boldsymbol{c} + \boldsymbol{R}\boldsymbol{r}_i, \\ \boldsymbol{v}_i^{(1)} = (\boldsymbol{x}_i^{(1)} - \boldsymbol{x}_i^{(0)})/\Delta t, \end{cases} \tag{26}$$

5 输出状态 $\boldsymbol{s}^{(1)} = \{\boldsymbol{x}^{(1)}, \boldsymbol{v}^{(1)}\}$

目录

有向距离场

有向距离场（signed distance field, SDF）$\phi(\boldsymbol{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}$ 针对空间中的位置输出其到某个物体表面的最短距离。若该位置在物体内部，则输出距离为负，否则距离为正。



$\phi(\boldsymbol{x})$ 的梯度 $\nabla\phi(\boldsymbol{x})$ 的方向给出了最快朝外远离物体表面的方向，可以类比法线方向理解。

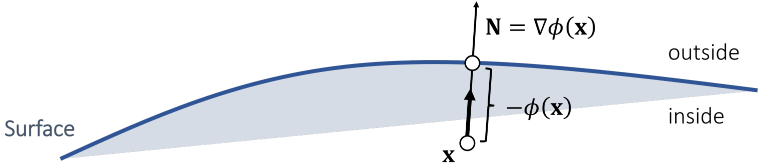
目录

朴素线性罚函数

- 质点碰撞检测的一般方法：检测其位置 $\boldsymbol{x}$ 是否满足 $\phi(\boldsymbol{x}) \geq 0$，若否，则意味着该质点位于物体内部，需进行碰撞处理
- 质点碰撞处理的一般方法：将其沿 $\nabla\phi(\boldsymbol{x})$ 的方向，移动到物体外
- 罚函数法（penalty method）是一类简单的碰撞处理方法，但在宏观物理上不一定正确。它的一般思路是在质点位于物体内部，或接近物体时，给其一个推力，使其远离物体。推力的方向 $\boldsymbol{N}$ 一般是归一化的 $\nabla\phi(\boldsymbol{x})$
- 朴素的罚函数法使用一个正比于 $\phi(\boldsymbol{x})$ 的惩罚推力

$$\boldsymbol{f} = -k\phi(\boldsymbol{x})\boldsymbol{N} \tag{27}$$

- 该方法只有顶点位于碰撞体内部时才能将其推开，存在穿透瑕疵

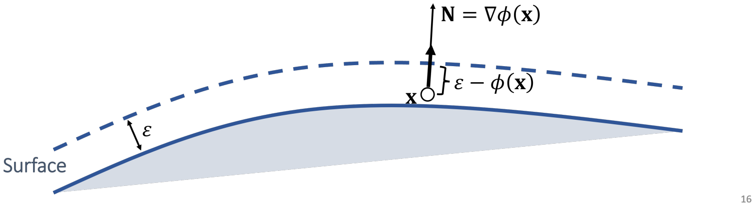


有缓冲区的线性罚函数

- 加入一个宽度为 ε 的缓冲区，可一定程度上缓解穿透问题。此时若检测到 $\phi(\boldsymbol{x}) < \varepsilon$，就要启动碰撞处理，施加惩罚推力

$$\boldsymbol{f} = -k(\phi(\boldsymbol{x}) - \varepsilon)\boldsymbol{N} \tag{28}$$

- 此方法可以让质点在穿透之前就有远离物体的倾向，且计算简单。但若质点速率很高，直接穿进了碰撞体，将导致推力过大，产生视觉不正确的“飞天”现象。

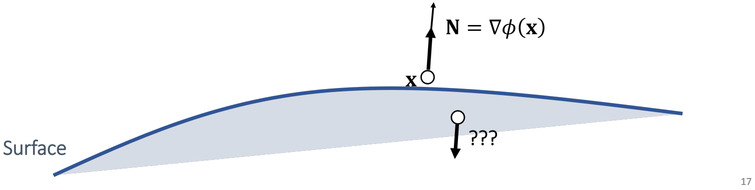


使用 log-barrier 的罚函数

- 同样地，考虑在一定距离上就施加惩罚推力，可以设计使用 log-barrier 的罚函数

$$\boldsymbol{f} = \rho \frac{1}{\phi(\boldsymbol{x})} \boldsymbol{N} \tag{29}$$

- 问题仍然存在：穿透后，将产生错误的推力方向



总结：

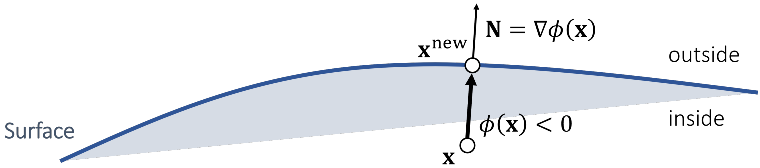
- 如今的物理引擎一般都实装一套简单的罚函数法，罚函数一般为有缓冲区的线性罚函数或 log-barrier 罚函数。它们实现简单，但都可能存在穿透瑕疵和推力可能过大的问题
- 罚函数法不适合处理摩擦碰撞的情况

目录

冲量法

- 冲量法（impulse method）是一类假设质点的碰撞将导致位置和速度的突变的碰撞处理方法，它基于库仑摩擦定律，在宏观物理上表现得更正确。它的一般思路是在质点位于物体内部时，将质点的位置直接移动到最近的物体表面，并改变其速度方向使其不要再接近物体
- 对碰撞位置 $\boldsymbol{x}$，新的位置 $\boldsymbol{x}^{\text{new}}$ 的更新方法是

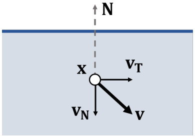
$$\boldsymbol{x}^{\text{new}} = \boldsymbol{x} - \nabla\phi(\boldsymbol{x})\phi(\boldsymbol{x}) \tag{30}$$



冲量法 - 速度更新 I

- 仅仅更新位置是不够的，为了避免接下来仍然发生穿透，我们还需要更新质点的速度
- 对碰撞时的速度 $\boldsymbol{v}$ 正交分解得到

$$\begin{cases} \boldsymbol{v}_N = (\boldsymbol{v} \cdot \boldsymbol{N})\boldsymbol{N}, \\ \boldsymbol{v}_T = \boldsymbol{v} - \boldsymbol{v}_N \end{cases} \tag{31}$$



- 考虑库仑摩擦，分别修改法向速度和切向速度

$$\begin{cases} \boldsymbol{v}_N^{\text{new}} = -\mu_N \boldsymbol{v}_N, \\ \boldsymbol{v}_T^{\text{new}} = a\boldsymbol{v}_T \end{cases} \tag{32}$$

冲量法 - 速度更新 II

- 最后合成得到

$$\boldsymbol{v}^{\text{new}} = \boldsymbol{v}_N^{\text{new}} + \boldsymbol{v}_T^{\text{new}} \tag{33}$$

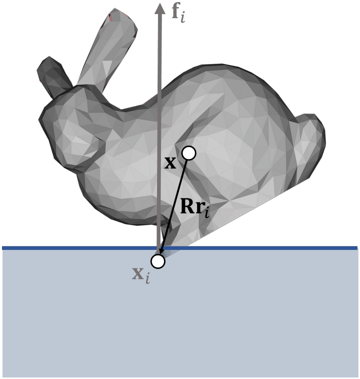
其中 μ_N, μ_T 是摩擦因数，参数 a 应当满足质点摩擦的库仑定律：

$$a = \max\left(1 - \mu_T(1+\mu_N)\frac{||\boldsymbol{v}_N||}{||\boldsymbol{v}_T||}, 0\right) \tag{34}$$

前项是动摩擦的结果，后项是静摩擦的结果。

目录

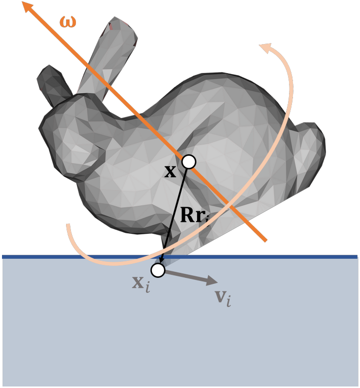
刚体碰撞检测与惩罚法的碰撞处理



- 朴素的刚体碰撞检测方法：对刚体的所有顶点 i，逐一检测 $\phi(\boldsymbol{x}_i)$ 是否为负值
- 聪明一点的碰撞检测方法：对碰撞不频繁的刚体，预先建立 BVH，进行 $O(\log n)$ 级别的检测
- 惩罚法的碰撞处理：对产生碰撞的顶点 i，累加相应的惩罚力 $\boldsymbol{f}_i$

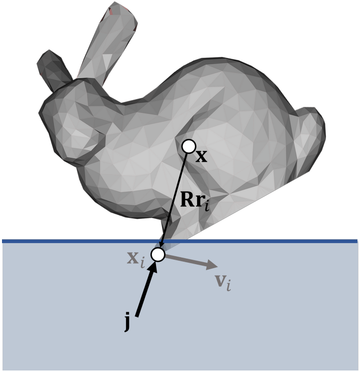
目录

冲量法的碰撞处理



- 使用冲量法进行刚体碰撞处理时，我们需要修改碰撞顶点的状态包括位置 $\boldsymbol{x}_i$、速度 $\boldsymbol{v}_i$ 和角速度 $\boldsymbol{\omega}_i$。但刚体为一整体，单独修改顶点的状态是不可行的
- 为此，我们需要从局部状态的修改，推导出整体状态的修改，思路类似逆向运动学（IK）

冲量法的碰撞处理-从局部到整体



- 简单起见，我们先只讨论碰撞点为单个顶点的情况
- 初始时，$\boldsymbol{v}_i = \boldsymbol{v} + \boldsymbol{\omega} \times \boldsymbol{R}\boldsymbol{r}_i$
- 顶点 i 被施加冲量 $\boldsymbol{j}_i = \boldsymbol{f}_i \Delta t$ 后，有

$$\begin{cases} \boldsymbol{v}^{\text{new}} = \boldsymbol{v} + M^{-1}\boldsymbol{j}_i, \\ \boldsymbol{\omega}^{\text{new}} = \boldsymbol{\omega} + \boldsymbol{I}^{-1}(\boldsymbol{R}\boldsymbol{r}_i \times \boldsymbol{j}_i) \end{cases} \tag{35}$$

- 此时，$\boldsymbol{v}_i^{\text{new}}$ 与整体 $\boldsymbol{v}^{\text{new}}, \boldsymbol{\omega}^{\text{new}}$ 的关系为

$$\boldsymbol{v}_i^{\text{new}} = \boldsymbol{v}^{\text{new}} + \boldsymbol{\omega}^{\text{new}} \times \boldsymbol{R}\boldsymbol{r}_i \tag{36}$$

冲量法的碰撞处理 – 推导 I

代入计算得到

$$\begin{aligned}
\boldsymbol{v}_i^{\text{new}} &= \boldsymbol{v}^{\text{new}} + \boldsymbol{\omega}^{\text{new}} \times \boldsymbol{R}\boldsymbol{r}_i \\
&= \boldsymbol{v} + M^{-1}\boldsymbol{j}_i + (\boldsymbol{\omega} + \boldsymbol{I}^{-1}(\boldsymbol{R}\boldsymbol{r}_i \times \boldsymbol{j}_i)) \times \boldsymbol{R}\boldsymbol{r}_i \\
&= \boldsymbol{v}_i + M^{-1}\boldsymbol{j}_i + (\boldsymbol{I}^{-1}(\boldsymbol{R}\boldsymbol{r}_i \times \boldsymbol{j}_i)) \times \boldsymbol{R}\boldsymbol{r}_i \\
&= \boldsymbol{v}_i + M^{-1}\boldsymbol{j}_i - \boldsymbol{F}_i\boldsymbol{j}_i
\end{aligned} \tag{37}$$

其中 $\boldsymbol{F}_i$ 是一个与 $\boldsymbol{R}, \boldsymbol{r}_i, \boldsymbol{I}$ 有关的线性算子。于是

$$\boldsymbol{j}_i = \boldsymbol{K}_i^{-1}(\boldsymbol{v}_i^{\text{new}} - \boldsymbol{v}_i) \tag{38}$$

其中

$$\boldsymbol{K}_i = M^{-1}\mathbf{1} - \boldsymbol{F}_i. \tag{39}$$

冲量法的碰撞处理 - 流水线 I

冲量法碰撞处理的一般步骤：

1. 输入状态 $\boldsymbol{s} = \{\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{\omega}\}$
2. 使用 SDF 检测碰撞，记碰撞点集合为 U。若 U 为空集，则无需碰撞处理
3. 否则，计算每个碰撞点更新后的位置和速度

$$\begin{cases} \boldsymbol{v}_i = \boldsymbol{v} + \boldsymbol{\omega} \times \boldsymbol{R}\boldsymbol{r}_i, \\ \boldsymbol{v}_{N,i} = (\boldsymbol{v}_i \cdot \boldsymbol{N})\boldsymbol{N}, \\ \boldsymbol{v}_{T,i} = \boldsymbol{v}_i - \boldsymbol{v}_{N,i}, \\ a = \max\left(1 - \mu_T(1+\mu_N)\dfrac{||\boldsymbol{v}_{N,i}||}{||\boldsymbol{v}_{T,i}||}, 0\right), \\ \boldsymbol{v}_{N,i}^{\text{new}} = -\mu_N \boldsymbol{v}_{N,i}, \\ \boldsymbol{v}_{T,i}^{\text{new}} = a\boldsymbol{v}_{T,i}, \\ \boldsymbol{v}_i^{\text{new}} = \boldsymbol{v}_{N,i}^{\text{new}} + \boldsymbol{v}_{T,i}^{\text{new}} \end{cases} \tag{40}$$

冲量法的碰撞处理 – 流水线 II

4 对每个碰撞点，计算使得其速度发生如上变化的冲量 $\boldsymbol{j}_i$

$$\begin{cases} \boldsymbol{K}_i = M^{-1}\mathbf{1} - \boldsymbol{F}_i, \\ \boldsymbol{j}_i = \boldsymbol{K}_i^{-1}(\boldsymbol{v}_i^{\text{new}} - \boldsymbol{v}_i), \end{cases} \tag{41}$$

5 若有多个碰撞点，求该冲量的平均值；取陷入物体最深的那个碰撞点，作为位移的参考

$$\begin{cases} \boldsymbol{j} = \dfrac{\sum \boldsymbol{j}_i}{|U|}, \\ \boldsymbol{x} = \min\limits_{U}(\boldsymbol{x}_i) \end{cases} \tag{42}$$

6 最后，利用该冲量更新刚体整体的速度和角速度，并更新位置以移除碰撞点

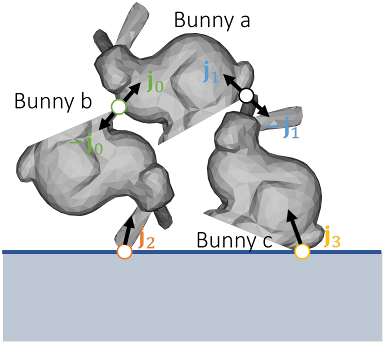
$$\begin{cases} \boldsymbol{c}^{\text{new}} = \boldsymbol{c} + \boldsymbol{R}(\boldsymbol{x} - \nabla\phi(\boldsymbol{x})\phi(\boldsymbol{x})), \\ \boldsymbol{v}^{\text{new}} = \boldsymbol{v} + M^{-1}\boldsymbol{j}, \\ \boldsymbol{\omega}^{\text{new}} = \boldsymbol{\omega} + \boldsymbol{I}^{-1}(\boldsymbol{R}\boldsymbol{r}_i \times \boldsymbol{j}) \end{cases} \tag{43}$$

冲量法的碰撞处理 – 流水线 III

7 输出状态 $\boldsymbol{s}^{\text{new}} = \{\boldsymbol{c}^{\text{new}}, \boldsymbol{v}^{\text{new}}, \boldsymbol{\omega}^{\text{new}}\}$

并行碰撞处理

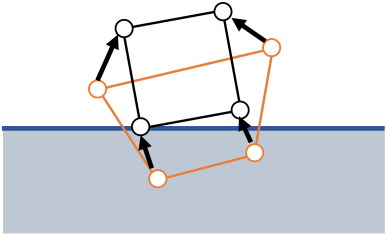
当多点同时接触时，并行碰撞处理问题，会变成维度更高的线性系统。实际处理时，常进行简单的串行处理。



目录

Shape Matching 的碰撞处理

- Shape Matching 仿真流程中，只需对每个顶点单独进行碰撞处理即可。这一处理方式与可变形体相同
- 若使用冲量法进行碰撞处理，则为了保证无穿透，需要在刚体约束之前、约束之后均再执行一遍碰撞检测和处理



Thanks for your watching!