计算题

2020年1月16日　22:55

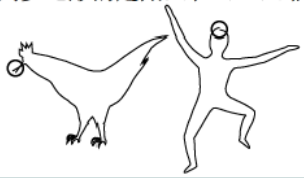
- **基于内在形状插值的多边形渐变方法**
  - **2D Shape Blending（二维形状的自然渐变）涉及哪两个子问题?**
    - 顶点的对应关系问题
    - 顶点的插值问题
  - **基于内在形状插值的多边形渐变方法的基本思想（需要掌握数学原理，在PPT03）**
    - Intrinsic Shape Interpolation的数学原理：乌龟几何+Lagrange乘数优化法
    - 乌龟几何：通过顶点处的边长和有向角来定义多边形。通过对关键帧多边形的边长和顶点角进行插值来产生比线性插值更好的效果。拉格朗日乘数法优化用于处理插值后起始点和终止点不在同一位置的问题。
    - 扩展：内在形状插值法的优缺点：
      - 优点
        - 简单
        - 能在一定程度上避免形状插值中出现的收缩和纽结现象，这在二维角色动画中尤其有用
      - 缺点
        - 不能处理曲线形状
        - 当源和目标多边形中包含短边时，由于短边的方向不稳定，中间帧多边形有可能产生较严重的畸变
  - **Edge Tweaking的思想**
    - 为了解决中间多边形不封闭的问题，一个解决方法为保持插值的顶点角不变，适当调整插值得到的边长（Edge Tweaking）

中间多边形生成

- 那么中间多边形可由插值相应的边长和顶点角得到：

$$\alpha_0 = (1-t)\alpha_{A_0} + t\alpha_{B_0}$$
$$\theta_i = (1-t)\theta_{A_i} + t\theta_{B_i} \qquad (i = 1,2,\cdots,m)$$
$$L_i = (1-t)L_{A_i} + tL_{B_i} \qquad (i = 1,2,\cdots,m)$$

- 遗憾的是，这样得到的多边形通常是不封闭的。但实验发现，得到多边形的起始点和终止点非常接近。



不封闭的多边形

中间多边形生成

- 一个解决方法为保持插值的顶点角不变，适当调整插值得到的边长(Edge Tweaking)：

$$L_i = (1-t)L_{A_i} + tL_{B_i} + S_i, \quad (i = 0,1,2,\cdots,m)$$

- 如果关键多边形的某一条对应的边具有相同的长度L，那么我们可认为中间多边形的边长也为L，因此我们可认为$|S_i| \propto |L_{A_i} - L_{B_i}|$。为了防止除以零，我们定义：

$$L_{AB_i} = \max\left\{\left|L_{A_i} - L_{B_i}\right|, L_{tol}\right\}, \qquad (i = 0,1,2,\cdots,m)$$

其中$L_{tol} = 0.0001 \times (\max_{i\in[0,m]}\left|L_{A_i} - L_{B_i}\right|)$

16

中间多边形生成

- 我们的目标是为了求得 $S_0, S_1, \cdots, S_m$，使目标函数：

$$f(S_0, S_1, \cdots, S_m) = \sum_{i=0}^{m} \frac{S_i^2}{L_{AB_i}^2}$$

最小，并且 $S_0, S_1, \cdots, S_m$ 应满足约束条件(强迫多边形封闭)：

$$\varphi_1(S_0, S_1, \cdots, S_m) = \sum_{i=0}^{m}\left[(1-t)L_{A_i} + tL_{B_i} + S_i\right]\cos\alpha_i = 0$$

$$\varphi_2(S_0, S_1, \cdots, S_m) = \sum_{i=0}^{m}\left[(1-t)L_{A_i} + tL_{B_i} + S_i\right]\sin\alpha_i = 0$$

中间多边形生成

- 这是个具有约束条件的极值问题，可用Lagrange乘数法来求解。引进Lagrange函数：

$$\Phi(\lambda_1, \lambda_2, S_1, S_2, \cdots, S_m) = f + \lambda_1\varphi_1 + \lambda_2\varphi_2$$

其中λ_1，λ_2为Lagrange乘数。对Φ求偏导得：

$$\frac{\partial\Phi}{\partial S_i} = \frac{2S_i}{L_{AB_i}^2} + \lambda_1\cos\alpha_i + \lambda_2\sin\alpha_i = 0 \qquad (i = 0,1,\cdots,m)$$

$$\sum_{i=0}^{m}\left[(1-t)L_{A_i} + tL_{B_i} + S_i\right]\cos\alpha_i = 0$$

$$\varphi_1(S_0,S_1,\cdots,S_m)=\sum_{i=0}^{m}[(1-t)L_{A_i}+tL_{B_i}+S_i]\cos\alpha_i=0$$

$$\varphi_2(S_0,S_1,\cdots,S_m)=\sum_{i=0}^{m}[(1-t)L_{A_i}+tL_{B_i}+S_i]\sin\alpha_i=0$$

其中α_i是由矢量P_iP_{i+1}和x轴构成的有向角，

$$\alpha_i=\alpha_{i-1}+\theta_i,(i=1,2,\cdots,m)$$

17

$$\frac{\partial\Phi}{\partial S_i}=\frac{2S_i}{L_{AB_i}^2}+\lambda_1\cos\alpha_i+\lambda_2\sin\alpha_i=0\qquad(i=0,1,\cdots,m)$$

$$\sum_{i=0}^{m}[(1-t)L_{A_i}+tL_{B_i}+S_i]\cos\alpha_i=0$$

$$\sum_{i=0}^{m}[(1-t)L_{A_i}+tL_{B_i}+S_i]\sin\alpha_i=0$$

18

中间多边形生成

- 由第一式求得S_i代入后二式得：

$$\begin{cases}E\lambda_1+F\lambda_2=U\\F\lambda_1+G\lambda_2=V\end{cases}$$

其中

$$E=\sum_{i=0}^{m}L_{AB_i}^2\cos^2\alpha_i,F=\sum_{i=0}^{m}L_{AB_i}^2\sin\alpha_i\cos\alpha_i,G=\sum_{i=0}^{m}L_{AB_i}^2\sin^2\alpha_i$$

$$U=2\left\{\sum_{i=0}^{m}[(1-t)L_{A_i}+tL_{B_i}]\cos\alpha_i\right\}$$

$$V=2\left\{\sum_{i=0}^{m}[(1-t)L_{A_i}+tL_{B_i}]\sin\alpha_i\right\}$$

19

中间多边形生成

- 如果 $EG-F^2\neq0$ ，则

$$\lambda_1=\begin{vmatrix}U&F\\V&G\end{vmatrix}\Big/\begin{vmatrix}E&F\\F&G\end{vmatrix}$$

$$\lambda_2=\begin{vmatrix}E&U\\F&V\end{vmatrix}\Big/\begin{vmatrix}E&F\\F&G\end{vmatrix}$$

$$S_i=-\frac{1}{2}L_{AB_i}^2(\lambda_1\cos\alpha_i+\lambda_2\sin\alpha_i),(i=0,1,2,\cdots,m)$$

- 因而可得到中间多边形顶点的坐标：

$$\begin{cases}x_i=x_{i-1}+L_{i-1}\cos\alpha_{i-1}\\y_i=y_{i-1}+L_{i-1}\sin\alpha_{i-1}\end{cases}$$

20

- **Velocity Verlet积分方法**

Velocity Verlet积分方法

- The *velocity Verlet method* gives positions, velocities, and accelerations at the same time without compromising precision:

$$\mathbf{x}(t+h)=\mathbf{x}(t)+\mathbf{v}(t)h+\frac{1}{2}\mathbf{a}(t)h^2$$

$$\mathbf{v}(t+h)=\mathbf{v}(t)+\frac{1}{2}[\mathbf{a}(t)+\mathbf{a}(t+h)]h$$

- Implemented in three stages since calculating velocities requires accelerations at both t and t+h.

Velocity Verlet积分方法

Actual implementation trick:

1. Velocities at t+1/2h are calculated using information at t.

$$\mathbf{v}(t+\frac{1}{2}h)=\mathbf{v}(t)+\mathbf{a}(t)\frac{1}{2}h$$

2. With these terms in memory, calculate:

$$\mathbf{x}(t+h)=\mathbf{x}(t)+\mathbf{v}(t)h+\frac{1}{2}\mathbf{a}(t)h^2=\mathbf{x}(t)+\mathbf{v}\left(t+\frac{1}{2}h\right)h$$

3. Calculate **a**(t+h) from forces using positions at t+h, and from this and **v**(t), calculate **v**(t+h).

$$\mathbf{v}(t+h)=\mathbf{v}(t)+\frac{1}{2}[\mathbf{a}(t)+\mathbf{a}(t+h)]h$$

- **Wyvill六次多项式势函数的数学原理及其特点**
  - 在一个元球系统中，每个元球可以有不同的势函数，我们用f_i表示第i个元球P_i的势函数。
  - 假设第i个元球的中心为(x_i, y_i, z_i)，空间某一点(x, y, z)与P_i的距离用来r表示，

$$r = \sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2}$$

  R_i是第i个圆球的有效半径。

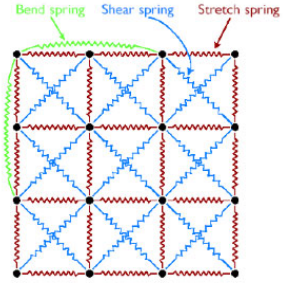
  - **Wyvill的六次多项式**

- $$f_i(r) = \begin{cases} -\frac{4}{9}(\frac{r}{R_i})^6 + \frac{17}{9}(\frac{r}{R_i})^4 - \frac{22}{9}(\frac{r}{R_i})^2 + 1, & 0 \le r \le R_i \\ 0, & r \ge R_i \end{cases}$$

Provot的衣服模型的原理

- 该模型是一个由m*n个虚拟质点组成的网络，质点之间用无质量的、自然长度不为0的弹簧连接，其连接关系有以下三种

  - 连接质点[i, j]与[i+1,j]，[i, j]与[i,j+1]的弹簧，称为“结构弹簧”；
  - 连接质点[i, j]与[i+1,j+1]，[i+1,j]与[i,j+1]的弹簧，称为“剪切弹簧”；
  - 连接质点[i, j]与[i+2,j]，[i, j]与[i, j+2]的弹簧，称为“弯曲弹簧”。



- 每个弹簧上受到的力

用$\mathbf{f}_{i,j}$表示质点i与质点j之间的弹簧作用在质点i上的力，这里的弹簧采用遵从Hooke定律的线弹性弹簧，则有：

$$\mathbf{f}_{i,j} = k_{ij}\left(\left|\mathbf{x}_j - \mathbf{x}_i\right| - l_{ij}^0\right)\frac{\left(\mathbf{x}_j - \mathbf{x}_i\right)}{\left|\mathbf{x}_j - \mathbf{x}_i\right|}$$

其中，k_{ij}表示弹簧的弹性系数，$\mathbf{x}_i$表示质点i的位置，l^0_{ij}表示弹簧的原长（松弛长度）。这样，质点i上的总作用力就可以表示为(这里E表示所有弹簧集合)：

$$\mathbf{f}_i = \sum_{(i,j)\in E}\mathbf{f}_{i,j} = \sum_{(i,j)\in E} k_{ij}\left(\left|\mathbf{x}_j - \mathbf{x}_i\right| - l_{ij}^0\right)\frac{\left(\mathbf{x}_j - \mathbf{x}_i\right)}{\left|\mathbf{x}_j - \mathbf{x}_i\right|}$$

- 显式欧拉法

显式欧拉法公式：

$$\begin{cases} \mathbf{a}^{t+h} = \mathbf{M}^{-1}\mathbf{f}^{t} \\ \mathbf{v}^{t+h} = \mathbf{v}^{t} + h\mathbf{a}^{t+h} \\ \mathbf{x}^{t+h} = \mathbf{x}^{t} + h\mathbf{v}^{t+h} \end{cases} \Rightarrow \begin{pmatrix} \mathbf{v}^{t+h} \\ \mathbf{x}^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^{t} + h\mathbf{M}^{-1}\mathbf{f}^{t} \\ \mathbf{x}^{t} + h\mathbf{v}^{t+h} \end{pmatrix}$$

其中M是质点的质量，x、v、a、f分别表示粒子的位置、速度、加速度和所受合力，*h*是时间步长。上标*t*和*t+h*是时间，分别表示当前时间和下一步的时间

优点：

- 单步计算快。
- 并行性好。在一个计算步中，每一个质点的状态都是独立计算。

- 因为采取了显式积分法，算法的稳定性成为一个明显的问题，为了保证算法稳定，必须采用很小的时间步长或者减小方程组的刚度，也即是弹簧的高度。
- 取小的计算步长意味着计算次数增加，取小的弹簧刚度则会形成超弹性的问题
- 对于超弹性问题，他采用了约束变形方法，动态修正质点的位置以保证弹簧不至于产生不现实的伸长。
- 这该方法在每个时间步的运算完成后，检测各质点的情况，如发现有变形过大的现象存在，就修正该质点的位置，使其限制在某个范围内。其实际效果是将超弹性效应在整块布料内分散。
- 该方法实际上把布料模型可以看作是半刚性的。当超弹性效应只发生在局部的少数几个点时，该方法取得了不错的效果，但当布料受力比较大超弹性现象发生比较多的时候，这个方法就无能为力了。

- **布料动画中大步长隐式方法的数学原理**
  - 在不降低刚度的条件下可以取得较大的时间步长

大步长隐式方法

$$\begin{pmatrix} \mathbf{v}^{t+h} \\ \mathbf{x}^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^{t} + h\mathbf{M}^{-1}\mathbf{f}^{t} \\ \mathbf{x}^{t} + h\mathbf{v}^{t+h} \end{pmatrix} \Rightarrow \begin{pmatrix} \mathbf{v}^{t+h} \\ \mathbf{x}^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^{t} + h\mathbf{M}^{-1}\mathbf{f}^{t+h} \\ \mathbf{x}^{t} + h\mathbf{v}^{t+h} \end{pmatrix}$$

显式欧拉法　　隐式欧拉法

$$\mathbf{f} = [\mathbf{f}_1, \ \mathbf{f}_2, \ \cdots, \ \mathbf{f}_n]^T, \quad \mathbf{v} = [\mathbf{v}_1, \ \mathbf{v}_2, \ \cdots, \ \mathbf{v}_n]^T, \quad \mathbf{x} = [\mathbf{x}_1, \ \mathbf{x}_2, \ \cdots, \ \mathbf{x}_n]^T$$

$$\mathbf{M}_i = \begin{bmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{M}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{M}_n \end{bmatrix}$$

这里 $\mathbf{f}_i$，$\mathbf{v}_i$，$\mathbf{x}_i$，m_i分别表示质点i的力、速度、位置和质量，n为质点的数目。

大步长隐式方法

$$\Delta\mathbf{v}^{t+h} = \mathbf{v}^{t+h} - \mathbf{v}^{t} = h\mathbf{M}^{-1}\mathbf{f}^{t+h}, \text{（公式1）}$$
$$\Delta\mathbf{x}^{t+h} = \mathbf{x}^{t+h} - \mathbf{x}^{t} = h\mathbf{v}^{t+h}$$

- 但是我们只能求得当前时刻的力$\mathbf{f}^t$，为了计算$\mathbf{f}^{t+h}$，我们用一阶泰勒展开对其进行近似 展开：

$$\mathbf{f}^{t+h} = \mathbf{f}^{t} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Delta\mathbf{x}^{t+h} + \frac{\partial \mathbf{f}}{\partial \mathbf{v}}\Delta\mathbf{v}^{t+h}$$

- 其中，$\Delta\mathbf{x}^{t+h}$表示质点在下一时刻$t+h$与当前时刻t的位置变化，可由如下公式计算：$\Delta\mathbf{x}^{t+h} = (\mathbf{v}^{t}+\Delta\mathbf{v}^{t+h})h$

$$\Rightarrow \quad \mathbf{f}^{t+h} = \mathbf{f}^{t} + h\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{v}^{t}+\Delta\mathbf{v}^{t+h}) + \frac{\partial \mathbf{f}}{\partial \mathbf{v}}\Delta\mathbf{v}^{t+h}$$

用公式1

$$\Rightarrow \quad \Delta\mathbf{v}^{t+h} = h\mathbf{M}^{-1}\mathbf{f}^{t} + h^2\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{v}^{t}+\Delta\mathbf{v}^{t+h}) + h\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{v}}\Delta\mathbf{v}^{t+h}$$

整理得

$$\Rightarrow \quad \left(\mathbf{I} - h^2\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{x}} - h\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{v}}\right)\Delta\mathbf{v}^{t+h} = h\mathbf{M}^{-1}\mathbf{f}^{t} + h^2\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{v}^{t}$$

大步长隐式方法

两边乘M

$$\Rightarrow \quad \left(\mathbf{M} - h^2\frac{\partial \mathbf{f}}{\partial \mathbf{x}} - h\frac{\partial \mathbf{f}}{\partial \mathbf{v}}\right)\Delta\mathbf{v}^{t+h} = h\mathbf{f}^{t} + h^2\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{v}^{t}$$

- 这是一个关于 $\Delta\mathbf{v}^{t+h}$ 的线性方程组，其中的质量矩阵M是以3×3对角矩阵为子矩阵的对角阵。
- 两个Jacobian矩阵 $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ 和 $\frac{\partial \mathbf{f}}{\partial \mathbf{v}}$ 都是子矩阵为 3×3对称矩阵的稀疏矩阵，只有当质点i和质点j之间有弹簧连接时非零（对角线上也非零）。
- 因此，总的系数矩阵 $\left(\mathbf{M} - h^2\frac{\partial \mathbf{f}}{\partial \mathbf{x}} - h\frac{\partial \mathbf{f}}{\partial \mathbf{v}}\right)$是一个稀疏矩阵，可以采用*共轭梯度法迭代*求解。
- 计算开销大！

- 因为f一般由其一阶泰勒展开近似得到，因此稳定性仍然有限制
- 是

背诵题

2019年9月27日 10:11

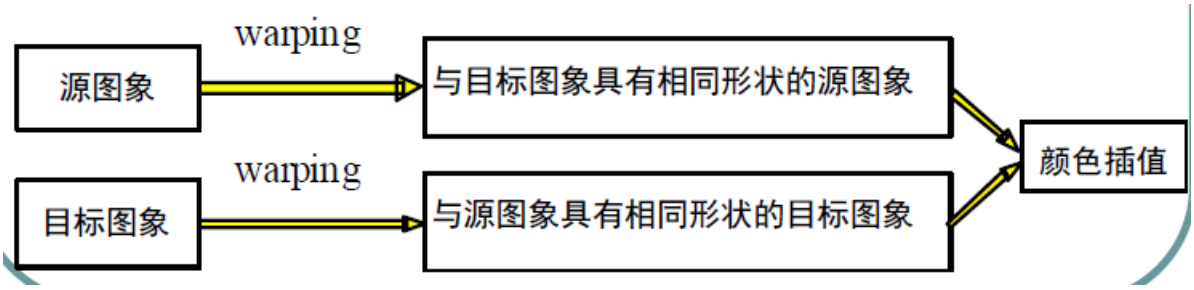
- **01-Introduction**
  - **动画形成的视觉原理**
    - 所谓动画，就是指通过以每秒若干帧的速度顺序地播放静止图像帧以产生运动错觉的艺术。动画利用了人的视觉残留这一特点，即上个画面的残留还未消失，下一个画面又进入视觉，这样循环往复，在人的眼中形成动态的画面
- **02-12Rules**
  - **传统动画应用于三维计算机动画的基本原则**
    - Arcs 弧形动作
    - Anticipation 预期性
    - Appeal 吸引力
    - Balance & Weight 平衡及重量感-衍生原则
    - **Depth of Field 景深-衍生原则**
    - Exaggeration 夸张
    - Follow-Through and Overlapping Action 跟随动作与重叠动作
    - Secondary Action 附属动作
    - Solid drawing 手绘技巧
    - Slow In and Slow Out 慢入和慢出
    - **Staging 布局**
    - Straight-Ahead Action and Pose-to-Pose Action 连贯动作与关键动作法
    - Squash and Stretch 挤压和伸展
    - Timing 掌握时序
- **03-2D blending**
  - **2D Shape Blending（二维形状的自然渐变）涉及哪两个子问题?**
    - 顶点的对应关系问题
    - 顶点的插值问题
  - **基于内在形状插值的多边形渐变方法的基本思想（需要掌握数学原理，在PPT03）**
    - Intrinsic Shape Interpolation的数学原理：乌龟几何+Lagrange乘数优化法
    - 乌龟几何：通过顶点处的边长和有向角来定义多边形。通过对关键帧多边形的边长和顶点角进行插值来产生比线性插值更好的效果。拉格朗日乘数法优化用于处理插值后起始点和终止点不在同一位置的问题。
    - 扩展：内在形状插值法的优缺点：
      - 优点

        - 简单
        - 能在一定程度上避免形状插值中出现的收缩和纽结现象，这在二维角色动画中尤其有用
    - 缺点
        - 不能处理曲线形状
        - 当源和目标多边形中包含短边时，由于短边的方向不稳定，中间帧多边形有可能产生较严重的畸变
- **Edge Tweaking的思想**
    - 为了解决中间多边形不封闭的问题，一个解决方法为保持插值的顶点角不变，适当调整插值得到的边长（Edge Tweaking）

- **04-2Dmorphing**
    - **图像Morphing的原理（几何元->图像特征对应关系->几何变换->几何对应关系->C0C1->颜色插值）**
        - 为了实现两幅二维图像I_S和I_D的morphing过程，动画师首先通过简单的几何元建立图像特征之间的对应关系
        - 由这些特征对应关系计算出Morphing所需的几何变换，几何变换定义了两幅图像上点之间的几何对应关系。
        - 满射C0：I_S->I_D把第一幅图像的几何形状映射为第二幅图像的几何形状，满射C_1：I_D->I_S把第二幅图像的几何形状映射为第一幅图像的几何形状，需要两个映射的原因是图像点和点之间的对应关系不一定是一一对应
        - 当两幅图像变形对齐后，我们可采用简单的颜色插值得到中间帧图像

设P_0为源图象上的点，P_1为目标图象上的点，则源图象和目标图象的warping函数W_0和W_1分别定义为：

$$W_0(P_0,t)=(1-t)P_0+tC_0(P_0)$$
$$W_1(P_1,t)=(1-t)C_1(P_1)+tP_1$$
$$t\in[0,1]$$



        - 拓展：尽管图像morphing在二维图像空间处理问题，但可以让人产生神奇的三维形状改变的错觉。可以避免复杂的三维造型过程。
    - **基于网格的图像morphing的原理（MsMd->M->MSM+MDM->I0+I1->I）**
        - 在源图像中放置曲面网格Ms，Ms指定了控制顶点的坐标，在目标图像中放置网格MD，M_D指定了Ms在目标图像的对应点。曲面Ms和M_D用来定义把源图像的所有点映射到目标图像的空间变换，Ms和M_D的拓扑结

构相同。（为了简单起见，网格的边界通常与图像的边界重合，而且冻结不动）

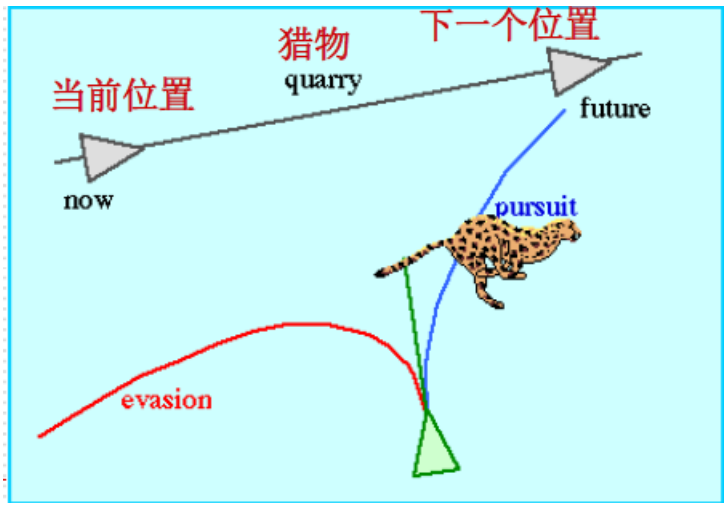
- 中间帧图像的生成过程
  - 线性插值网格Ms和M_D，得到网格M
  - 应用由网格Ms和M定义的变换，使源图像Is扭曲变形到I_0
  - 应用由网格MD和M定义的变换，使目标图像I_D扭曲变形到I_1
  - 对图像I0和I1进行线性插值，得到中间帧图像
- 拓展：基于网格(Grid)的图象morphing是图象morphing中最早的方法。采用的网格通常为双三次样条曲面，如Bezier样条曲面、Catmull-Rom样条曲面等。

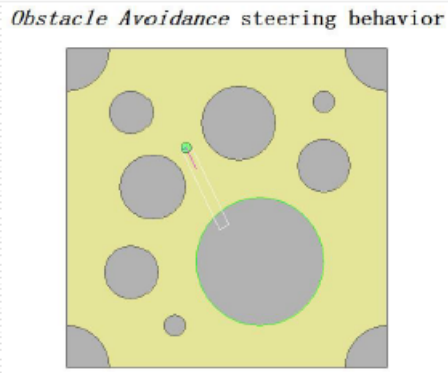
- **基于线对的图像morphing方法（基本思路：逆向映射->I1I2->交溶得I）**
  - 其采用的映射方法为逆向映射，即逐个扫描目标图像的像素，根据其位置采样源图像。（正向：正向映射为逐个扫描源图象素，然后把它拷入目标图象的适当位置。）
  - 在源图像中定义一条有向线段，再在目标图像中定义一条有向线段，那么这一对直线段定义了一个从一幅图像到另一副图像的映射，该映射把一条直线映射为另一条直线。
  - 设I_S为原图像，I_D为目标图像，先在两幅图像中定义控制变形的对应直线对，通过插值得到中间图像I的直线。
    - 插值方法有两种
      - 对直线端点进行线性插值，该方法的缺点是对于旋转的直线对可能得到的缩短的插值直线
      - 对直线的中点、朝向和长度进行插值，通常这一方法效果较好
  - 由I_S->I的直线对变换可以得到一副变形图I1，由I_D->I的直线对变换可以得到I2。对交溶参数作动画，把I_1和I_2进行交溶处理可以得到中间的变形图像I。
  - 扩展：基于线对特征法的优点是直观，缺点是可能生成一些意料之外的图像。
- **二、三维形状渐变（morphing）各有什么优缺点**
  - 二维形状渐变
    - 优点
      - 它是一种达到特殊视觉效果的有效方法
      - 可以让人产生神奇的三维形状改变的错觉
      - 可以避免复杂的三维造型过程
    - 缺点
      - 有可能生成一些意料之外的图像（基于线对特征法）
      - 物体没有三维几何信息，无法进行几何变换，使摄像机的动画收到了很大的限制
  - 三维形状渐变

        - 优点
            - 能生成跟逼真和生动的特技效果
            - 三维morphing的结果与视点和光照参数无关，并能够生成精确的光照和阴影效果
            - 得到的中间帧是物体序列而不是图像，可以用不同的摄像机角度和光照条件来对他们进行重新绘制，也可以把它们与其它的三维场景相结合进行绘制（运用范围更广）
        - 缺点
            - 物体之间的对应关系很难建立
            - 对于物体的几何表示相当苛刻，比如要求多边形的个数相等，物体的拓扑结构相同等等

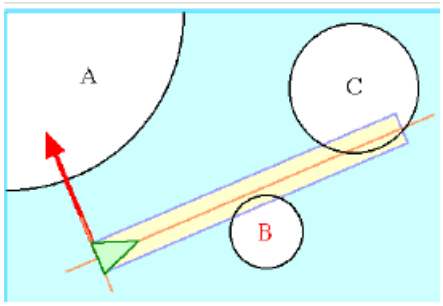
- **05-3Dmorphing**
    - **基于星形物体的多面体morphing方法（基本思路：对应->插值）**
        - 该方法通过合并一对三维多面体物体模型的拓扑结构，使得它们具有相同的顶点/边/面网络结构，然后对相应的顶点进行插值
        - 通过合并拓扑结构将一个物体的形状变换为另外一个物体的形状通常可分为以下两步（其中第一步称为对应问题，第二步称为插值问题）：
            - 建立源物体表面的点与目标物体上的点对应关系
            - 插值对应的点
        - 对于亏格为0的两个多面体的形状渐变：
            - 首先，它们都同构于球，可以采取下面三个步骤来建立亏格为0的多面体之间的对应关系
                - 把两个多面体投影到单位球面上
                - 将投影在单位球面上的两个拓扑结构合并在一起构成一个新的拓扑结构
                - 将新的拓扑结构映射回原来的两个多面体
            - 这样就建立好了相应的顶点对应关系
        - 一旦顶点对应关系建立以后，通常采用线性插值或Hermite插值来插值相应的顶点
    - **基于体表示的三维morphing方法（对应关系->空间变换->扭曲变形->几何对齐+混合）**
        - 给定源体S和目标体T，（与二维图像morphing的思想相似，该方法）
        - 首先依据指定的对应特征生成一空间变换，该变换使给定的两个体扭曲变形（warp）成S'和T'，达到几何对齐的目的
        - 然后对得到的两个扭曲变形体S'和T'进行混合
            - 先对S和T进行体绘制，然后把绘制好的图像进行交融处理。（缺点：生成结果的光照和遮挡效果不正确，缺乏真实的三维morphing效果）
            - 对S'和T‘体素的颜色和不透明度值进行交溶处理，然后对混合后的体素进行体绘制。

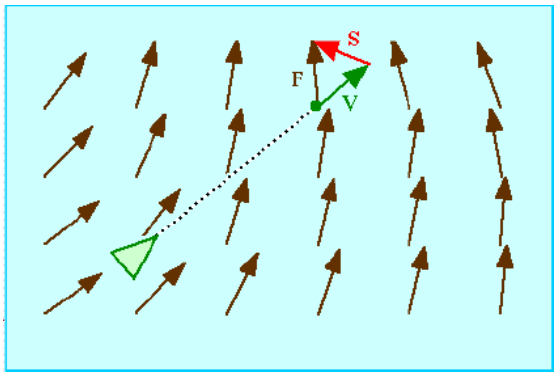
- **06-ParticleSystem**
  - **粒子系统的基本原理**
    - 粒子系统的基本思想是将许多简单形状的微小粒子作为基本元素聚集起来形成一个不规则模糊物体。每个粒子均经历出生，成长，衰老和死亡的过程。生成粒子系统的某一帧画面的基本步骤是：
      - 生成新的粒子并加入系统中
        - 对于每一帧，根据一个控制的随机过程生成粒子
          - 用户可控制每帧的平均粒子数和其概率分布
          - 粒子数可以是时间的函数
      - 赋予新粒子一定的属性
      - 删除已经超过生命周期的粒子
      - 根据粒子的动态属性对粒子进行移动和变换
      - 绘制并显示由有生命的粒子组成的图形
    - 粒子系统将造型和动画巧妙地连成一体，景物被定义为成千上万个不规则的、随机分布的粒子所组成，而每个粒子均有一定的生命周期，它们不断改变形状、不断运动。
    - 粒子系统的假设：（不碰撞+不向其他粒子投射阴影+只向其它环境投射阴影+不反射光+有有限的生命周期）
      - 粒子一般与其他粒子不碰撞
      - 除非处于聚集状态，粒子不向其它粒子投射阴影
      - 粒子只向其它环境投射阴影
      - 粒子不反射光
      - 粒子通常有有限的生命周期
    - 粒子的属性
      - 位置
      - 速度
      - 大小
      - 质量
      - 力累加器（force accumulator）
      - 生命周期
      - 绘制属性（形状参数、颜色、透明度）
    - 粒子系统属性
      - article List：指向一个粒子系统的指针
      - Position：粒子系统的位置
      - Emission Rate：决定生成新粒子的创建方法
      - Forces ：加入力可极大地增强物理真实程度
      - Current State：用来记录系统的状态，例如在模拟爆炸时，粒子系统会很快停止发射
      - Blending：用来设置粒子相互混合的方式
      - Representation：例如，粒子的纹理

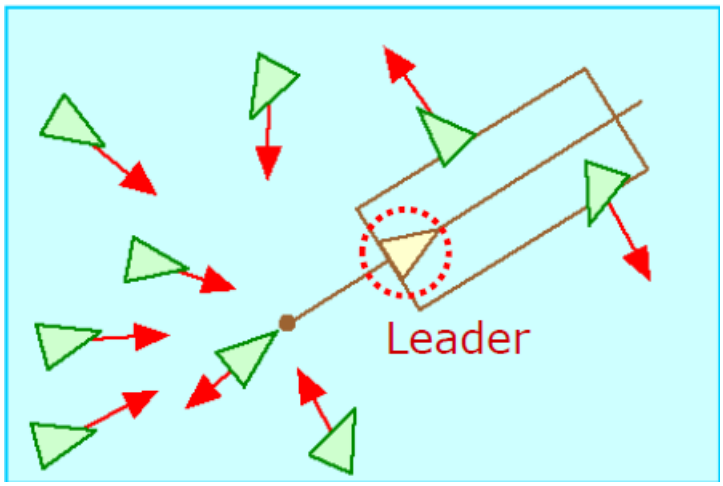
- **07-Crowd1**
  - Boid是一个模拟的类似鸟一样的物体
  - **Boids模型的三条原则（优先级递减的群体模拟三大原则）**
    - 碰撞避免原则 Collision Avoidance
      - 避免与相邻的群体成员相碰
    - 速度匹配原则 Velocity Matching
      - 尽量保持与周围邻居群体成员的速度匹配
    - 群体合群原则 Flock Centering
      - 群体成员尽量靠近
- **08-Crowd2**
  - **Reynolds追逐和躲避（Peruit and Evasion）**
    - 追逐（Pursuit）与寻找（Seek）非常类似，其区别在于目标为一移动的角色（猎物）
      - 假设猎物在预测区间T内不会转向
      - 猎物在将来的位置可通过把它的当前速度乘以T，并将该偏移量与当前位置相加来得到
      - 

  - **障碍避免（Obstacle Avoidance）**
    - 该行为的目标是：在该角色前面，保证有一个假想圆柱的自由空间
      - 

    - 障碍避免返回的值：
      - 返回避免最有威胁障碍物的导航值（返回的是行进路线导航，不是障碍物位置）
      - 如果没有碰撞是迫在眉睫的，返回一个特殊值（空值，或零向量），表示在这一时刻不需要纠正量



- **路径跟随（Path following）**
  - 在路径跟随中，路径意味着行驶方向
  - 投影距离（Projection distance）：从预测位置到最近路径点的位置
  - 沿着一条路径移动角色，并同时保持在脊柱线的指定半径内
  - 投影距离：小于路径半径时，不需要进行导航校正
  - 否则，把预测的位置投影到路径上，把该点作为目标点，并进行寻找（Seeking）行为
- **流畅跟随行为（Flow Field Following）**
  - 假设运动区域已经有一个速度流场
  - 估计角色将来的位置并计算在这点的流场
  - 得到的速度（F）即为我们期望的速度，导航方向（S）为期望速度和当前速度的差
  - 

- **跟随领导（Leader Following）**
  - 如果一个跟随成员发现自己处于领导前面的一个矩形区域，它会横向远离领导者的路径
  - 否则，到达（arrival）目标为目标后面的一个偏移点
  - 跟随成员采用分离行为来避免相互拥挤
  - 

- **09-Crowd3**

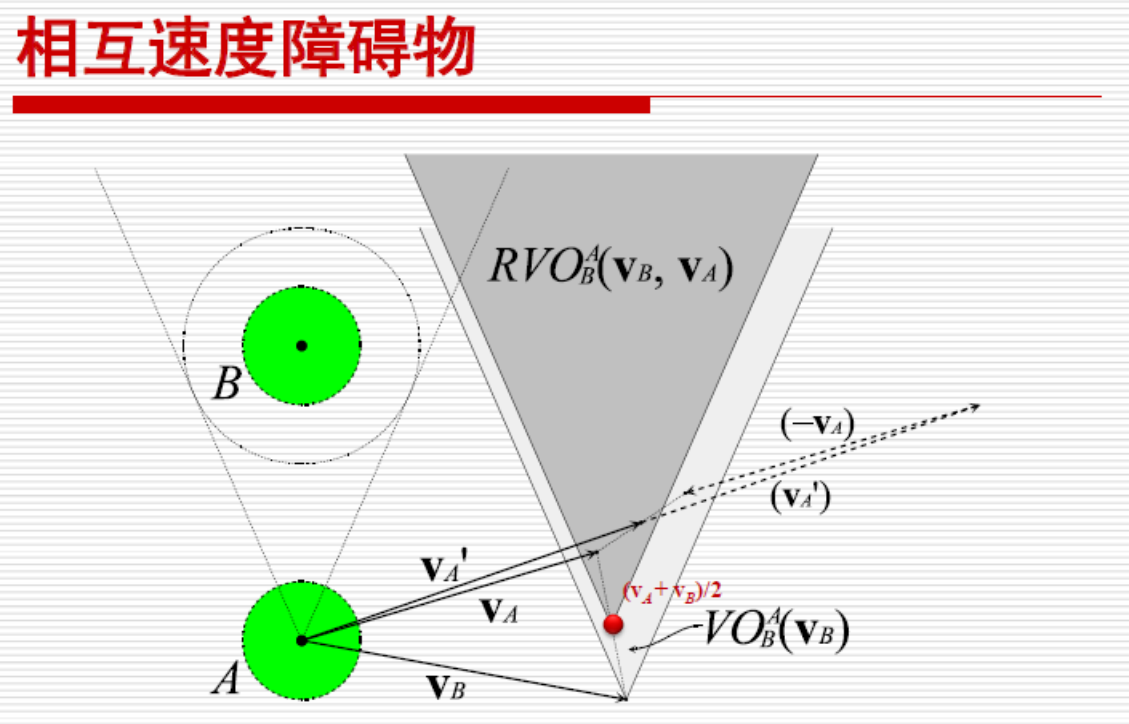
- **Helbing基于社会力模型的群体行为模拟方法的基本原理**
  - 社会力模型以牛顿动力学为基础，由各个力的表达式来体现行人不同的动机和影响。在社会力模型中，由于对影响个体的因素考虑得比较全面，对个体行为的建模比较合理，该模型可以逼真地模拟人群的疏散过程。
  - 个体的实际行为受主观意识，其它个体及障碍物三方面因素的影响，均可等效为力在个体上的作用。
- **相互速度障碍物（RVO）原理（匀速->相对运动+计算可能碰撞->相关个体->相互协调）**
  - 假设个体以匀速前进
  - 每个个体在保持与周围个体相对运动的同时，在速度域中计算出可能导致碰撞的速度集合，并对自身速度进行必要的调整。在调整过程中，碰撞避免的任务同时分配给相关的个体，使它们相互协调完成碰撞避免任务
  - 扩展：想方法避免振荡
    - 原因：智能体之间没有通讯或整体的协调
    - 简单的想法：选择当前速度和一个位于速度障碍物外部的速度的平均
    - 形成相互速度障碍物（RVO，Reciprocal Velocity Obstacle）
  -

相互速度障碍物

- B对A的相互速度障碍物包含A的所有以下速度的集合：该集合的成员为A的当前速度$\mathbf{v}_A$和B的速度障碍物内的一个速度的平均；

$$RVO_B^A(\mathbf{v}_B,\mathbf{v}_A)=\left\{\mathbf{v}_A' \mid 2\mathbf{v}_A'-\mathbf{v}_A\in VO_B^A(\mathbf{v}_B)\right\}$$

- 几何上，可解释为把B的速度障碍物$VO_B^A(\mathbf{v}_B)$平移到顶点$(\mathbf{v}_A+\mathbf{v}_B)/2$;



- **10-IK1**
  - **正向运动学**
    - 动画师通过直接指定关节处的关节运动参数来控制物体的运动
    - 从关节空间映射到笛卡尔空间
    - 计算整棵树：从根节点到叶节点进行深度优先遍历
  - **逆向运动学**
    - 动画师指定目标位置，系统求解满足要求的关节角
    - 从笛卡尔空间映射到关节空间
    - 给定初始姿态向量和目标姿态向量，计算关节向量的值，使得物体满足所需的姿势
  - **逆向雅克比方法求解IK**
    - 逆向雅克比矩阵把笛卡尔空间的速度映射到关节空间的速度
    - 给定初始姿势和所需要的姿势，迭代变化关节角，使得末端影响器朝目标位置和方向前进
      - 对于中间帧，插值得到所需的姿态向量
      - $P(t) = f(\theta(t))$　　$P \in R^n$ （通常$n=6$）
        $\theta \in R^m$（m=自由度）
        - 雅克比矩阵为$n \times m$的矩阵，它把θ的微分$(d\theta)$与P的微分相关联(dP)
        - $\frac{dP}{dt} = J(\theta)\frac{d\theta}{dt}$　　其中$J_{ij} = \frac{\partial f_i}{\partial \theta_j}$
        - 所以雅克比矩阵把关节空间的速度映射到笛卡尔空间的速度　$V = J(\theta)\dot{\theta}$

逆向雅克比问题为：

$$\theta = f^{-1}(P)$$

- f是一个高度非线性函数

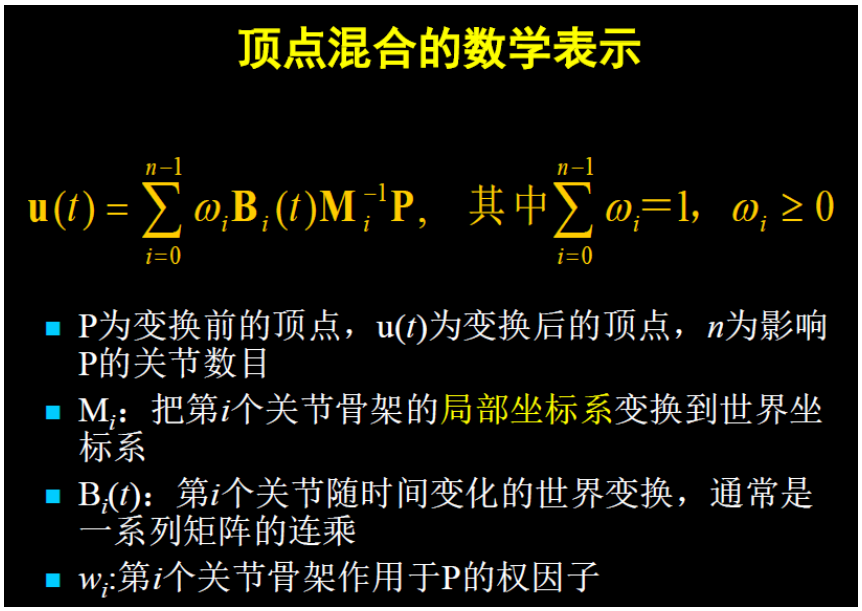
通过把雅克比矩阵求逆，把该问题在当前位置局部线性化 $\dot{\theta} = J^{-1}V$

- 通过一系列增量步骤，迭代到所需要的位置

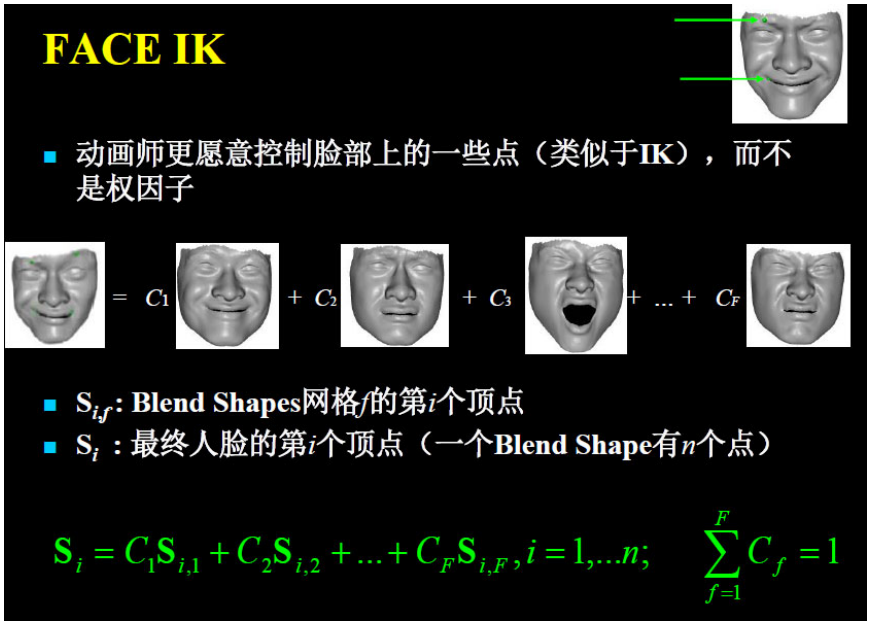
对于每一步k, 通过上述公式得到关节的角速度 $\dot{\theta}$ 然后执行

$$\theta_{k+1} = \theta_k + \Delta t \dot{\theta}$$

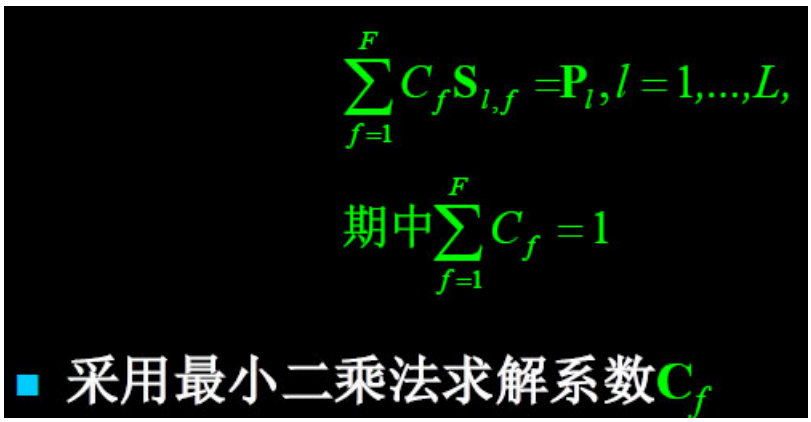
- **11-IK2**
  - **运动捕获系统（MOCAP）流水线**
    - 标定（Calibration）
    - 捕获（Capture）
    - 三维位置重建（3D Position Reconstruction）
    - 拟合到骨架（Fitting to the Skeleton）
    - 后处理（Post Processing）
  - **循环坐标下降法原理**
    - 对所有受IK影响的骨骼，按从最远侧子骨骼到父骨骼的顺序执行迭代操作：旋转当前骨骼，使当前骨骼位置到目标骨骼的连线指向IK目标位置
    - 由于所有骨骼是从一个特定状态出发开始IK计算，所得到的结果也会比较稳定。通常5~10次跌倒之后就能得到很好的结果
    - 拓展阅读：
      - 如果是人体骨骼的话，不是所有的关节都可以向任意方向旋转，所以我们必须对骨骼的旋转加以约束，比如肘关节实际上只有一个轴的自由度，而且不能向后弯曲。
      - 奇异情况：当所有需要IK控制的骨骼正好在一条直线上，而IK目标位置正好在也落在这条直线上时（如下图），算法就会失败，因为不论迭代多少次，每一个骨骼都会认为自己不需要旋转。
      - 解决技巧：如果发现骨骼链“很直”，就向骨骼允许的任意方向加一些细微的旋转；或者干脆在骨骼的限制角度数据中就禁止完全“伸直”。
  - **基于运动捕获的关节动画制作的优缺点**
    - 优点
      - 只要能被捕获，可记录人体运动的所有细节，运动真实
    - 缺点

        - 不容易进行控制和编辑
        - 较昂贵
    - **骨架与角色模型的绑定原理**
        - 在骨架绑定中，皮肤的运动函数定义为对应控制骨架的函数
        - 角色的表面（外皮）必须随着骨架的运动而运动（变形）
        - 很多骨架绑定系统采用一个称为中性姿势或者静止姿势（Rest Pose）的几何信息
        - 骨架与三维角色模型的自动绑定
            - 首先在未知的三维模型中嵌入骨架
            - 然后计算骨骼对表面网格上每个顶点的影响权值，并将表面皮肤依附在骨骼上
    - **顶点混合**
        - 假设一个数字角色的手臂用前臂和后臂来模拟，为了防止肘关节处不像真实手臂，关节处应是柔性的。
        - 最简单的方法就是前臂和后臂依然设置动画，关节处用一柔软的“Skin”来连接。
        - 柔软部分的一部分矩阵的顶点由前臂的矩阵来变换，另一部分由后臂的矩阵来变换。即：一个三角形的顶点可以由不同的矩阵来变换，而不是一个矩阵。这种基本技术有时也称为“Stitching”
        - 进一步推广：一个顶点可以由几个不同的矩阵进行加权变换
        - 实施方法：在物体上放置关节骨架，每个骨架按用户给定的权因子影响顶点
    - **顶点混合的数学表达**
        - 

- **12-IK3**
    - **基于Blend Shapes表情动画的原理**
        - 根据插值这些Blend Shapes来得到任意的表情
            - 最终表情=C_1*表情$_1$+C_2*表情$_2$+C_3*表情$_3$+…+C_F*表情$_F$
                - 其中$C_1+C_2+C_3+\ldots C_F=1$



- 我们希望约束L个点（动画师控制），使得顶点l的位置为P



- **13-Deformation**
  - **Deformation与Morphing的区别**
    - 变形（Deformation）是指几何对象的形状作某种扭曲、形变，使它形成动画师所需的形状。在这种变化中，几何对象的拓扑关系保持不变。
    - 与Morphing不同的是，空间变形更具某种随意性，所以空间变形也常称为自由变形
  - **与物体表示无关的变形的原理**
    - **即可作用于多边形表示的物体，又可作用于参数曲面表示的物体？啥啊这？**
  - **扩展的FFD方法EFFD的原理（FFD->EFFD->迭代求解）**
    - FFD：该方法不直接操作物体，而是将物体嵌入一空间，当所嵌的空间变形时，物体也随之变形。只适合于平行六面体的lattice形状
    - EFFD允许非平行六面体的lattice形状，从而能实现更任意的变形。EFFD型Lattice允许FFD型Lattice作为它结构的一部分，许多个FFD型Lattice可合并构成EFFD的lattice。（通过把多个基本EFFD块融合，我们可以得到更复杂的复合EFFD块。）EFFD块构造好后，我们可以采用与FFD方法类似的方法来使物体变形，通常需要迭代求解。
    - 采用EFFD方法对物体进行变形的步骤如下：

        - 给定点Q->所在的超曲面->通过对方程迭代得到Q的局部坐标（控制顶点，初始值）

给定物体上的一点Q，我们首先利用超曲面的凸包性质找到Q所在的超曲面，然后对方程

$$\mathbf{Q}(u,v,w)=\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}\mathbf{P}_{ijk}B_i(u)B_j(v)B_k(w),$$

进行牛顿迭代，求得Q在相应超曲面的局部坐标(u, v, w)，其中$\mathbf{P}_{ijk}$为该超曲面变形前的控制顶点。在迭代时，只需把初始值简单地设为$(u, v, w)=(0.5, 0.5, 0.5)$，便可得到较好的收敛结果。

        - 根据动画设计的需要，移动EFFD块的控制顶点
        - 根据超曲面方程，求得变形后的位置
    - 扩展：EFFD的优缺点
        - 优点
            - 允许更加复杂的变形空间
        - 缺点
            - 在移动内部控制顶点时必须保持块与块之间的连续性
            - 计算景物点在lattice空间的局部坐标需要数值求解方法，导致计算速度变慢
- **基于Cage的变形原理**
    - Cage：一个包裹高分辨率模型的低分辨率网格
    - 基于Cage的变形方法具有直观性、简单性、高效性
    - 变形原理：
        - 构建模型的Cage
        - 计算模型的Cage坐标
        - 编辑Cage模型
        - 将Cage的变形通过预先计算的Cage坐标光滑传播到其包裹的模型

14-cloth

- **布料动画的核心问题**
    - 建立布料的物理（力学）模型
        - 物理参数的设置
    - 物理模型的求解（偏微分方程数值求解）
        - 速度
        - 稳定性
    - 碰撞检测和响应
        - 衣服与人体的碰撞，衣服与衣服的碰撞
        - 多层衣服
        - 胳肢窝等特殊位置的处理
        - 纽扣、装饰物与衣服的碰撞

        - 从材料角度看：织物的力学性能的预测（结构问题）
    - **布料的物理机械性能**
        - In general, cloth resists motion in four directions:
            - In-plane stretch 拉伸力
            - In-plane compression 压缩力
            - In-plane shear(trellising) 剪切力
            - Out-of-plane bending 弯曲力
- **15-implicit**
    - 隐式曲面的表示
        - 把曲面表示为定义在空间的函数f（x,y,z）
        - 在空间对点进行分类
        - 曲面通过隐式定义
            - 在内部f>0
            - 在外部f<0
            - 在曲面上f=0
        - 等价地
            - 在内部f<0
            - 在外部f>0
    - 隐式曲面的性质
        - 可有效判断一个点是否位于曲面的内部
        - 有效的曲面求交计算
        - 有效的布尔操作
    - **隐式曲面的优缺点**
        - **优点**
            - 很容易判断一个点是否在曲面上
            - 很容易计算曲面的交/并/差
            - 很容易处理拓扑变化
        - **缺点**
            - 曲面通过间接指定
            - 很难描述尖锐特征
            - 很难对曲面上的点进行枚举
            - 绘制慢
    - **隐式曲面核函数的选择**
        - 通过骨架和核函数的卷积来形成标量场

- 通过骨架和核函数的卷积来生成标量场
  - 标量场

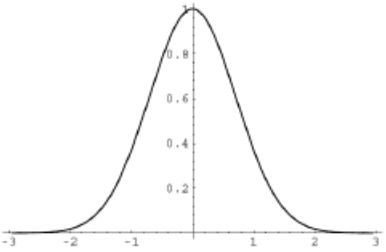
$$f(\mathbf{P}) = \int_V g(\mathbf{Q})h(\mathbf{P}-\mathbf{Q})dV = (f \otimes h)(\mathbf{P})$$

  - 骨架

$$g(\mathbf{P}) = \begin{cases} 1, & \mathbf{P} \in \text{skeleton } V; \\ 0, & \text{otherwise.} \end{cases}$$

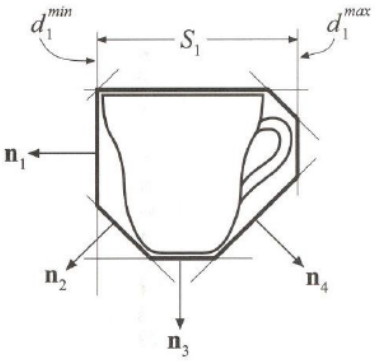
  - 核函数(低通滤波函数)
    - 例如：高斯函数

$$h(r) = h(\mathbf{P}-\mathbf{Q}) = \exp(-a^2 r^2)$$



- 16-Collision
  - **如何选择包围体**
    - 有很多种选择，每一种选择都是一种折中
    - 更好的包裹是更好的选择
      - 可以剔除一些假的相交
    - 物体的形状越简单，效果越好
      - 简单的形状容易计算
    - 具有旋转不变性的包围体最好
  - **离散有向多面体k-DOP**
    - AABB：轴对齐包围盒，是一个表面法向量与坐标轴方向一致的长方体（横平竖直的长方体）
    - OBB：有向包围盒，是一个表面法向两两垂直的长方体，是一个可以任意旋转的AABB
    - k-DOP：与AABB类似的思想，但用更多的轴
    - 包裹性比AABB好，但计算量更大

「离散有向多面体k-DOP的定义」



一个茶杯的二维8-DOP示例。

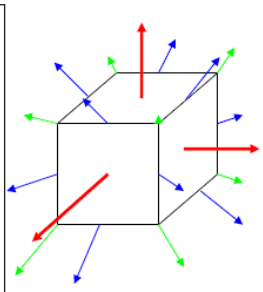
- 由$k/2$(k是偶数)个归一化法向$\mathbf{n}_i$来定义($1 \le i \le k/2$)，每个$\mathbf{n}_i$有两个相关标量值d_i^{min}和d_i^{max}，其中$d_i^{min}<d_i^{max}$。每个三元组($\mathbf{n}_i$, d_i^{min},d_i^{max})描述一个平板层S_i，表示两个平面之间的空间。这两个平面是$\mathbf{n}_i \cdot \mathbf{x}+d_i^{min}=0$和$\mathbf{n}_i \cdot \mathbf{x}+d_i^{max}=0$。所有平板层的交集为*k*-DOP的实际空间。

「k-dops轴的选择」

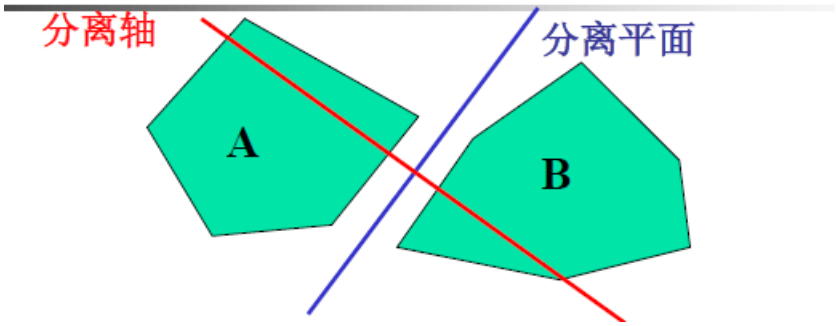
常用的轴: 从一个立方体中心出来的轴:

- Through faces: 6-dop
  - 与AABB一样
- Faces和vertices: 14-dop (6+8)
- Faces和edge centers: 18-dop (6+12)
- Faces, vertices, and edge centers; 26-dop (6+8+12)

- 比26-dop更大并没有多大帮助
- 实验结果表明， 14 或18-dop具有最好的性能



- **分离轴定理**
  - 对于任意两个互相分离的凸多面体（这里指的广义上的凸多面体，包括线段和三角形等）A和B，存在一条分离轴，其中这两个多面体在这条轴上具有一定间隔而且在轴上的投影也是互相分开的，这条轴正交于下列其一
    - A的一个面
    - B的一个面
    - 多面体的一条边



- **求交测试的重要规则**
  - 这些规则可以使得相交测试更快速、稳定、精确
    - 尽量使用简单的比较和计算来排除和确认各种相交类型，避免进一步的运算
    - 尽量搁置开销量大的计算，如三角函数，平方根，除法等
    - 尽量找出在相交测试前可以预先完成的计算
    - 如果使用了多次排除或确认测试，试着改变它们之间的内部顺序，这有可能产生一个更有效的测试结果
    - 降低维度
    - 尽量充分利用上次的测试结果
    - 如果相交测试比较复杂，尽量先利用物体的包围球进行初步排除
    - 上机实践进行计时比较，计时要用真实的数据和测试环境
    - 尽量使程序具有较好的鲁棒性，能够处理各种特殊情况，对浮点精度误差不敏感