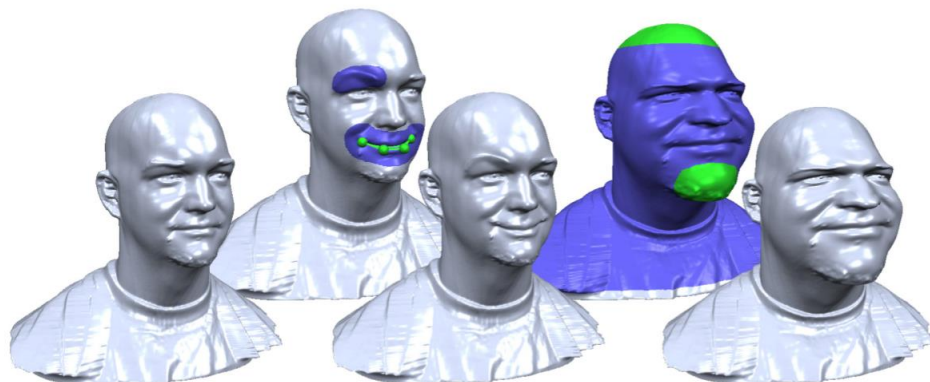
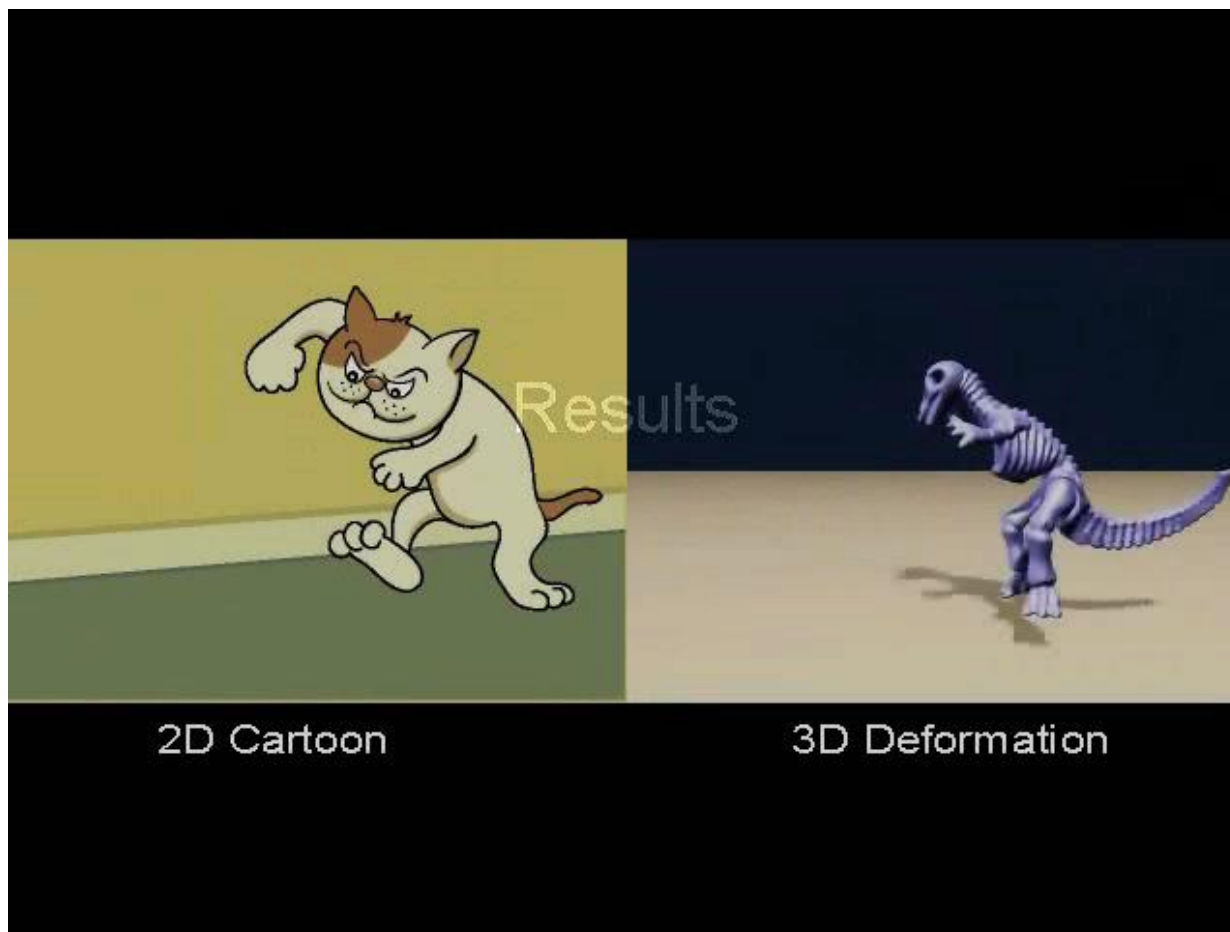


# 变形动画技术



# 变形演示



# 变形

- 变形(Deformation)是指将几何对象的形状作某种扭曲、形变，使它形成到动画师所需的形状。在这种变化中，**几何对象的拓扑关系保持不变**。
- 与Morphing不同，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**(Free Form Deformation)。
- 空间变形既可以看成是造型的范畴，也可看承是动画的范畴。确切地说，空间变形属于针对动画的造型问题，它把造型和动画有机地相结合。

# 变形和物体表示

- **与物体表示有关的变形。**是指针对物体的某种具体表示形式，如多边形网格、细分曲面、参数曲面等的变形方式。
  - **多边形网格：**如针对Polygon Mesh的editing和deformation。
  - **参数曲面：**移动控制顶点仅仅改变了基函数的系数，曲面仍然是光滑的。但是，参数曲面表示的物体也会带来**三维走样问题**，由于控制顶点的分布一般比较稀疏，物体的变形不一定是我们所期望的；对于由多个面拼接而成的物体，变形的另一个约束条件是需保持相邻曲面间的**连续性**。
- **与物体表示无关的变形。**既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。许多商用动画软件如**Maya**、**3DSMAX**、**Softimage**等都包含空间变形工具。

**核心思想：**如何用少量的点去**有效控制**更多的点？

# 整体和局部变形方法

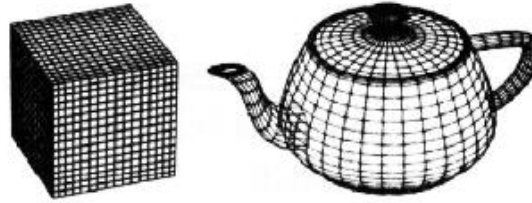
- Barr提出的整体和局部变形是空间变形中最早的方法。参考：  
**Barr A H. Global and local deformation of solid primitives. Computer Graphics, 1984, 18(3):21~30 (2019年12月24日，引用1388次)**
- 在传统的造型方法中，物体通常用CSG树来表示。通过基本物体的旋转、平移、比例缩放、求和、求交、求减等几何变换和布尔运算，CSG造型方法可以生成非常复杂的物体。

# 整体和局部变形方法

- Barr推广了传统的运算操作，他提出把整体和局部变形作为新的算子。
- 他提出的算子有：
  - Twisting(使成螺旋形)
  - Bending(弯曲)
  - Tapering(渐细)
- 这些算子的优点在于：① 推广了传统的造型运算，可以生成许多传统造型方法难以生成的形体。② 变形后物体的法向量可用原物体的法向量和变换矩阵解析求得。

# 示意图

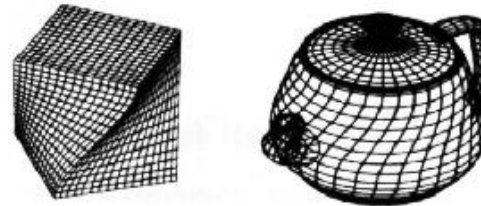
- original



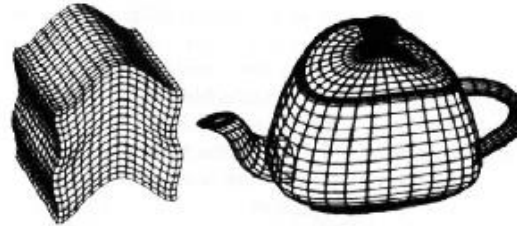
- tapering



- twisting



- bending



# 变形模型及变形后物体法向量的计算

- 在标准的三维坐标变换中，变换矩阵对被作用物体上的每个点均是不变的。
- Barr提出的非线性整体变形在于当变换作用于物体时，变换矩阵随不同的顶点变化，因而变换是物体顶点位置的函数。
- 设 $X$ 表示待变形物体上的点，其分量用 $(x_1, x_2, x_3)$  或 $(x, y, z)$ 来表示； $X'$ 表示变形后物体上的点，其分量用 $(x', y', z')$ 来表示。则整体变形可用变换 $X'=F(X)$ 来表示，其中 $F$ 为一显式地把 $X$ 变换成 $X'$ 的数学函数。



# 变形模型及变形后物体法向量的计算

- 而局部变形改变的是物体的切向量空间(较不直观，隐式改变物体的顶点位置)，该操作对物体的切向量进行旋转和扭曲，然后积分得到物体变形后的整体位置。
- 在几何造型中，切向量和法向量是两个非常重要的向量，因为切向量决定了物体的局部几何信息，法向量决定了物体的方向和光照信息。
- 变形后物体上点的切向、法向可通过原物体的切向、法向和变换的Jacobian矩阵来求得。

# 变形模型及变形后物体法向量的计算

- 变换 $\mathbf{X}' = \mathbf{F}(\mathbf{X})$ 的Jacobian 矩阵为:

$$\mathbf{J}(\mathbf{X}) = [\mathbf{J}_1(\mathbf{X}), \mathbf{J}_2(\mathbf{X}), \mathbf{J}_3(\mathbf{X})] = \left[ \frac{\partial \mathbf{F}(\mathbf{X})}{\partial x_1}, \frac{\partial \mathbf{F}(\mathbf{X})}{\partial x_2}, \frac{\partial \mathbf{F}(\mathbf{X})}{\partial x_3} \right]$$

- 设物体为 $\mathbf{X} = \mathbf{X}(u, v)$ , 物体上的某条曲线C为 $\mathbf{X} = \mathbf{X}(u(t), v(t))$ , 则物体的切向量为 $\mathbf{X}$ 对 $u, v$ 偏导的线性组合:

$$\mathbf{T} = \frac{\partial \mathbf{X}}{\partial u} \frac{\partial u}{\partial t} + \frac{\partial \mathbf{X}}{\partial v} \frac{\partial v}{\partial t}$$

- 物体上某一点的单位法向量为: 
$$\mathbf{N} = \frac{\frac{\partial \mathbf{X}}{\partial u} \times \frac{\partial \mathbf{X}}{\partial v}}{\left\| \frac{\partial \mathbf{X}}{\partial u} \times \frac{\partial \mathbf{X}}{\partial v} \right\|}$$

# 变形模型及变形后物体法向量的计算

- 变形后物体的切向量变换链为：

$$\begin{aligned}\frac{\partial \mathbf{X}'}{\partial u} &= \frac{\partial \mathbf{F}}{\partial x_1} \times \frac{\partial x_1}{\partial u} + \frac{\partial \mathbf{F}}{\partial x_2} \times \frac{\partial x_2}{\partial u} + \frac{\partial \mathbf{F}}{\partial x_3} \times \frac{\partial x_3}{\partial u} \\ &= \left[ \frac{\partial \mathbf{F}}{\partial x_1}, \frac{\partial \mathbf{F}}{\partial x_2}, \frac{\partial \mathbf{F}}{\partial x_3} \right] \left[ \frac{\partial x_1}{\partial u}, \frac{\partial x_2}{\partial u}, \frac{\partial x_3}{\partial u} \right]^T = \mathbf{J}(\mathbf{X}) \frac{\partial \mathbf{X}}{\partial u}\end{aligned}$$

可把上式改写为： $X'_{i,u} = \sum_{j=1}^3 J_{i,j} x_{j,u}$

- 换句话说，新的切向量  $\frac{\partial \mathbf{X}'}{\partial u}$  是 **Jacobian** 矩阵  **$\mathbf{J}(\mathbf{X})$**  乘以原切向量  $\frac{\partial \mathbf{X}}{\partial u}$ 。

# 变形模型及变形后物体法向量的计算

- 变形后物体的法向量变换链为：

$$\begin{aligned} \mathbf{N}' &= \frac{\partial \mathbf{Y}}{\partial u} \times \frac{\partial \mathbf{Y}}{\partial v} = \left( \mathbf{J} \frac{\partial \mathbf{X}}{\partial s} \right) \times \left( \mathbf{J} \frac{\partial \mathbf{X}}{\partial t} \right) \\ &= \left( \sum_{i=1}^3 \mathbf{J}_i x_{i,s} \right) \times \left( \sum_{j=1}^3 \mathbf{J}_j x_{j,t} \right) = \sum_{i=1}^3 \sum_{j=1}^3 (\mathbf{J}_i \times \mathbf{J}_j) x_{i,s} x_{j,t} \\ &= (\mathbf{J}_2 \times \mathbf{J}_3, \mathbf{J}_3 \times \mathbf{J}_1, \mathbf{J}_1 \times \mathbf{J}_2) \begin{pmatrix} x_{2,s} x_{3,t} - x_{3,s} x_{2,t} \\ x_{3,s} x_{1,t} - x_{1,s} x_{3,t} \\ x_{1,s} x_{2,t} - x_{2,s} x_{1,t} \end{pmatrix} \\ &= [\mathbf{J}_2 \times \mathbf{J}_3, \mathbf{J}_3 \times \mathbf{J}_1, \mathbf{J}_1 \times \mathbf{J}_2] \mathbf{N} \end{aligned}$$

# 变形模型及变形后物体法向量的计算

- 因为对任一矩阵 $\mathbf{M}$ 有：

$$\det(\mathbf{M}) = \mathbf{M}_1 \bullet (\mathbf{M}_2 \times \mathbf{M}_3)$$

$$\begin{aligned}\mathbf{M}^{-1} &= [\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3]^{-1} = \frac{[\mathbf{M}_2 \times \mathbf{M}_3, \mathbf{M}_3 \times \mathbf{M}_1, \mathbf{M}_1 \times \mathbf{M}_2]^T}{\mathbf{M}_1 \bullet (\mathbf{M}_2 \times \mathbf{M}_3)} \\ &= \frac{[\mathbf{M}_2 \times \mathbf{M}_3, \mathbf{M}_3 \times \mathbf{M}_1, \mathbf{M}_1 \times \mathbf{M}_2]^T}{\det(\mathbf{M})}\end{aligned}$$

- 我们得到： $\mathbf{N}' = \det(\mathbf{J})\mathbf{J}^{-1T}\mathbf{N}$
- 由于法向量的大小一般并不重要，因此 $\det(\mathbf{J})$ 通常不必计算。从微积分知识可以知道，**Jacobian**矩阵的值为变换点的局部体积之比。

# 变形例子——Scaling

- 最简单的变形例子为比例缩放Scaling:

$$\mathbf{F}: \begin{cases} x' = a_1 x \\ y' = a_2 y \\ z' = a_3 z \end{cases}$$

- 其Jacobian矩阵为:

$$\mathbf{J} = \begin{pmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{pmatrix}$$

- 体积比为:  $\det(\mathbf{J}) = a_1 a_2 a_3$

# 变形例子——Scaling

- 法向量变换矩阵为:

$$\det(\mathbf{J})\mathbf{J}^{-1T} = \begin{pmatrix} a_2 a_3 & 0 & 0 \\ 0 & a_1 a_3 & 0 \\ 0 & 0 & a_1 a_2 \end{pmatrix}$$

- 由于法向量的大小并不重要，法向量变换矩阵可取为:

$$\mathbf{J}^{-1T} = \begin{pmatrix} 1/a_1 & 0 & 0 \\ 0 & 1/a_2 & 0 \\ 0 & 0 & 1/a_3 \end{pmatrix}$$

- 若变形前的曲面为中心在原点的球面，则变形后该球面变成椭球面。该变换把中心 $(x, y, z)$ 变换为 $(a_1 x, a_2 y, a_3 z)$ ，把法向 $(n_1, n_2, n_3)$ 变换为 $(n_1/a_1, n_2/a_2, n_3/a_3)$ 。

# 变形例子——Tapering



- 沿 $z$ 轴的渐细变形Tapering为:

$$\mathbf{F}: \begin{cases} x' = rx \\ y' = ry \\ z' = z \end{cases}, \quad r = f(z)$$

- 当 $f'(z) > 0$ 时, 变形物体的大小沿 $z$ 轴逐渐增大;
- 当 $f'(z) < 0$ 时, 变形物体的大小沿 $z$ 轴逐渐变小。



# 变形例子——Tapering

- 该变换的切向变换矩阵为：

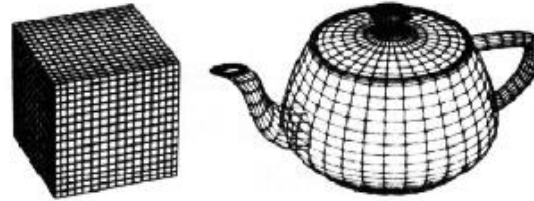
$$\mathbf{J} = \begin{pmatrix} r & 0 & f'(z)x \\ 0 & r & f'(z)y \\ 0 & 0 & 1 \end{pmatrix}$$

- 因为  $\det(\mathbf{J}) = r^2$ ，变换的局部体积之比为  $r^2$ 。  
该变换的法向变换矩阵为：

$$\det(\mathbf{J})\mathbf{J}^{-1\mathrm{T}} = \begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ -rf'(z)x & -rf'(z)y & z \end{pmatrix}$$

# 示意图

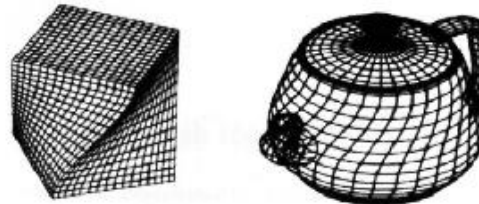
- original



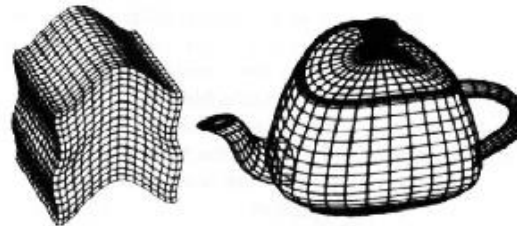
- tapering



- twisting



- bending



## 变形例子——Twisting(沿轴螺旋形变形)

- 该变换旋转其中两个坐标分量而不改变第三个坐标分量（像扭麻花）：

$$\mathbf{F}: \begin{cases} x' = xc_{\theta} - ys_{\theta} \\ y' = xs_{\theta} + yc_{\theta} \\ z = z \end{cases}$$

其中  $\theta = f(z)$ ,  $c_{\theta} = \cos \theta$ ,  $s_{\theta} = \sin \theta$ 。切向量变换矩阵为：

$$\mathbf{J} = \begin{pmatrix} c_{\theta} & -s_{\theta} & -xs_{\theta}f'(z) - yc_{\theta}f'(z) \\ s_{\theta} & c_{\theta} & xc_{\theta}f'(z) - ys_{\theta}f'(z) \\ 0 & 0 & 1 \end{pmatrix}$$

# 变形例子——Twisting(沿轴螺旋形变形)

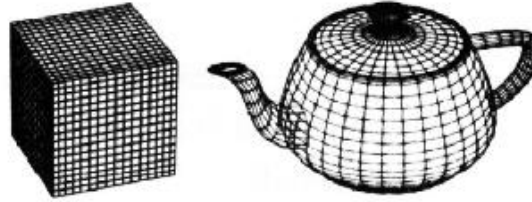
- 因为 $\det(\mathbf{J})=1$ ,所以螺旋形变形保持体积不变。  
法向变换矩阵为:

$$\det(\mathbf{J})\mathbf{J}^{-1T} = \begin{pmatrix} c_{\theta} & -s_{\theta} & 0 \\ s_{\theta} & c_{\theta} & 0 \\ yf'(z) & -xf'(z) & 1 \end{pmatrix}$$

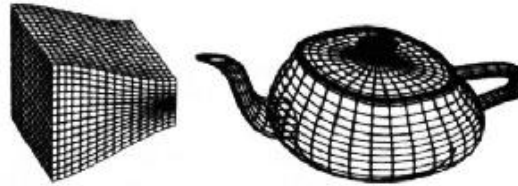


# 示意图

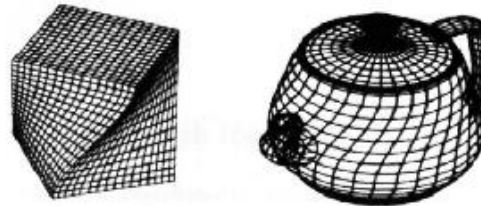
- original



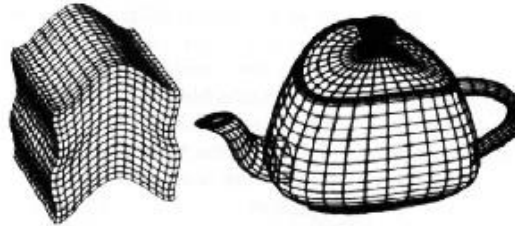
- tapering



- twisting



- bending



# 变形例子——Bending(弯曲变形)

- 沿y轴的弯曲变形(**Bending**)。
- 设变形的区域为  $y_{\min} \leq y \leq y_{\max}$ ，中心为  $y_0$ ，弯曲的曲率半径为  $1/k$ ，

$$\theta = k(\hat{y} - y_0),$$

$$\text{其中: } \hat{y} = \begin{cases} y_{\min}, & y \leq y_{\min} \\ y, & y_{\min} < y < y_{\max} \\ y_{\max}, & y \geq y_{\max} \end{cases}$$

- 即弯曲角在变形区域外为常数，在中间区域为线性变化，变形中中心线长度保持不变。

# 变形例子——Bending(弯曲变形)

- 设  $c_\theta = \cos \theta, s_\theta = \sin \theta$  , 则变形函数为:  
 $x' = x$

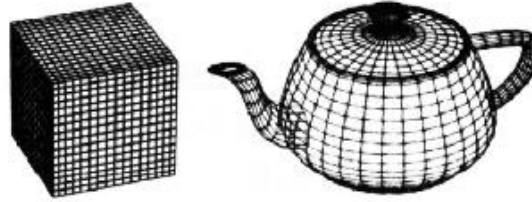
$$y' = \begin{cases} -s_\theta(z - \frac{1}{k}) + y_0, & y_{\min} \leq y \leq y_{\max} \\ -s_\theta(z - \frac{1}{k}) + y_0 + c_\theta(y - y_{\min}), & y < y_{\min} \\ -s_\theta(z - \frac{1}{k}) + y_0 + c_\theta(y - y_{\max}), & y > y_{\max} \end{cases}$$

$$z' = \begin{cases} c_\theta(z - \frac{1}{k}) + \frac{1}{k}, & y_{\min} \leq y \leq y_{\max} \\ c_\theta(z - \frac{1}{k}) + \frac{1}{k} + s_\theta(y - y_{\min}), & y < y_{\min} \\ c_\theta(z - \frac{1}{k}) + \frac{1}{k} + s_\theta(y - y_{\max}), & y > y_{\max} \end{cases}$$

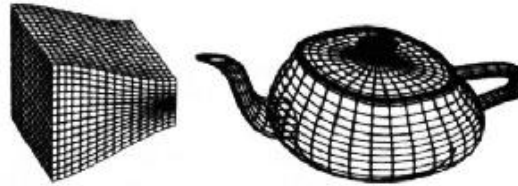
在 $y=y_0$ 时,  $x' = x, y' = y, z' = z$

# 示意图

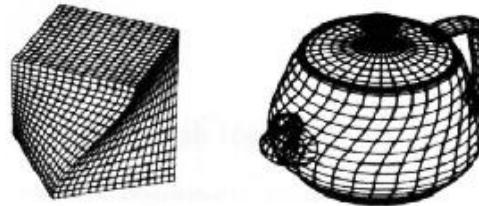
- original



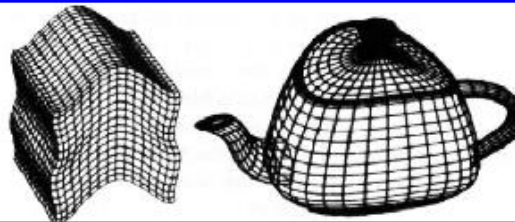
- tapering



- twisting



- bending





# 变形例子——Bending(弯曲变形)

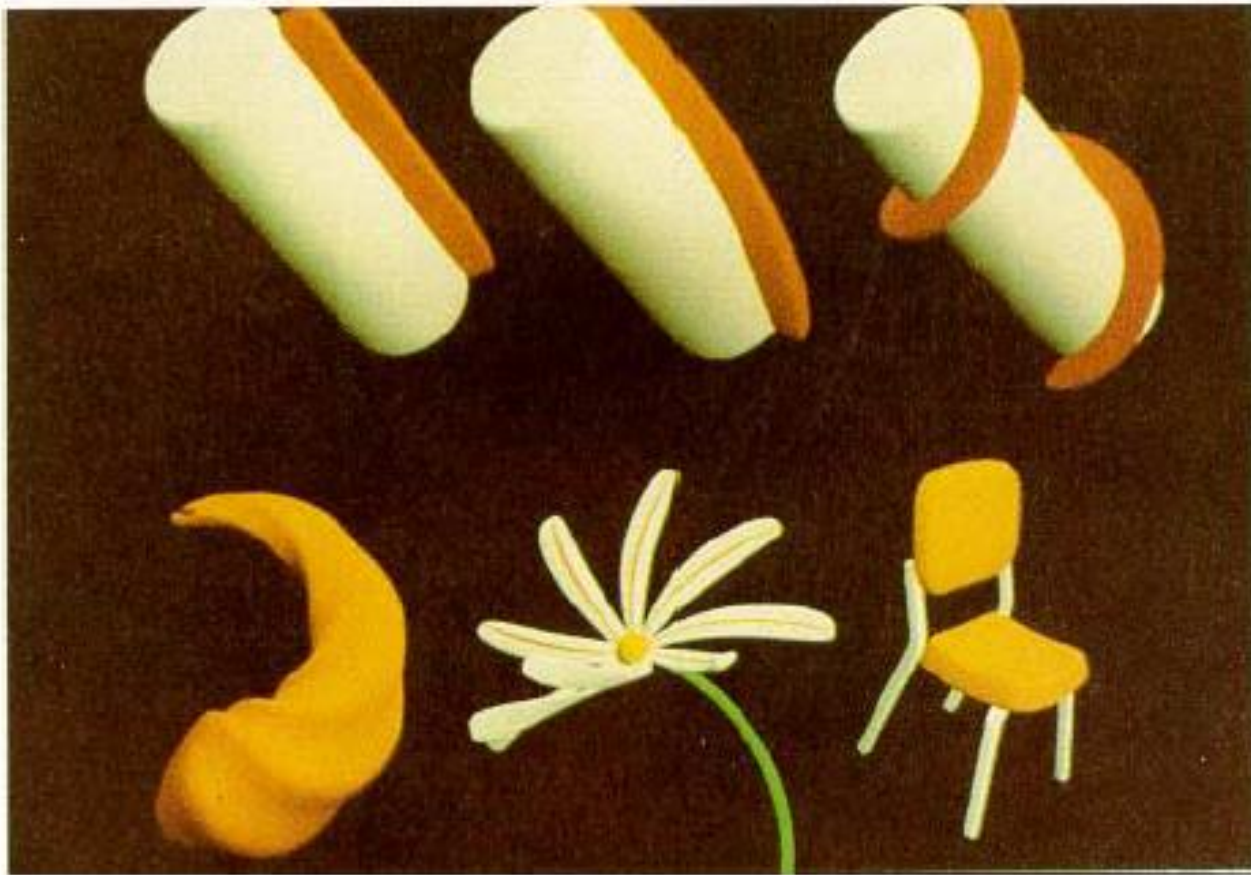
- 切向变换矩阵为:

$$\mathbf{J} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_\theta(1 - \hat{k}z) & -s_\theta \\ 0 & s_\theta(1 - \hat{k}z) & c_\theta \end{pmatrix}, \quad \text{其中 } \hat{k} = \begin{cases} k, & \hat{y} = y \\ 0, & \hat{y} \neq y \end{cases}$$

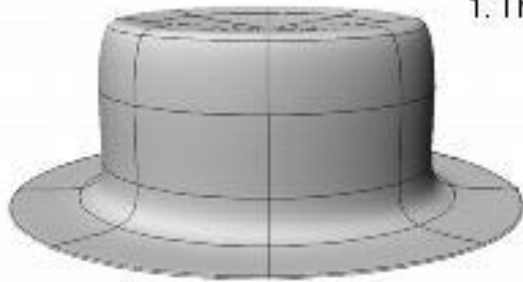
- 因为  $\det(\mathbf{J}) = 1 - \hat{k}z$ , 所以局部体积变化率为  $1 - \hat{k}z$ 。
- 法向变化矩阵为:

$$\det(\mathbf{J})\mathbf{J}^{-1T} = \begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ -rf'(z)x & -rf'(z)y & z \end{pmatrix}$$

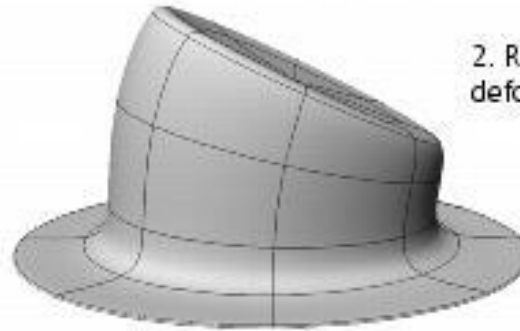
# Examples



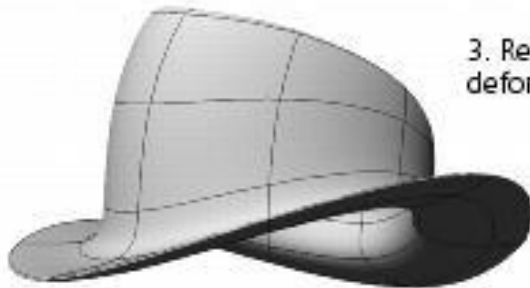
# Examples



1. The original hat



2. Result of a deformation (bend)



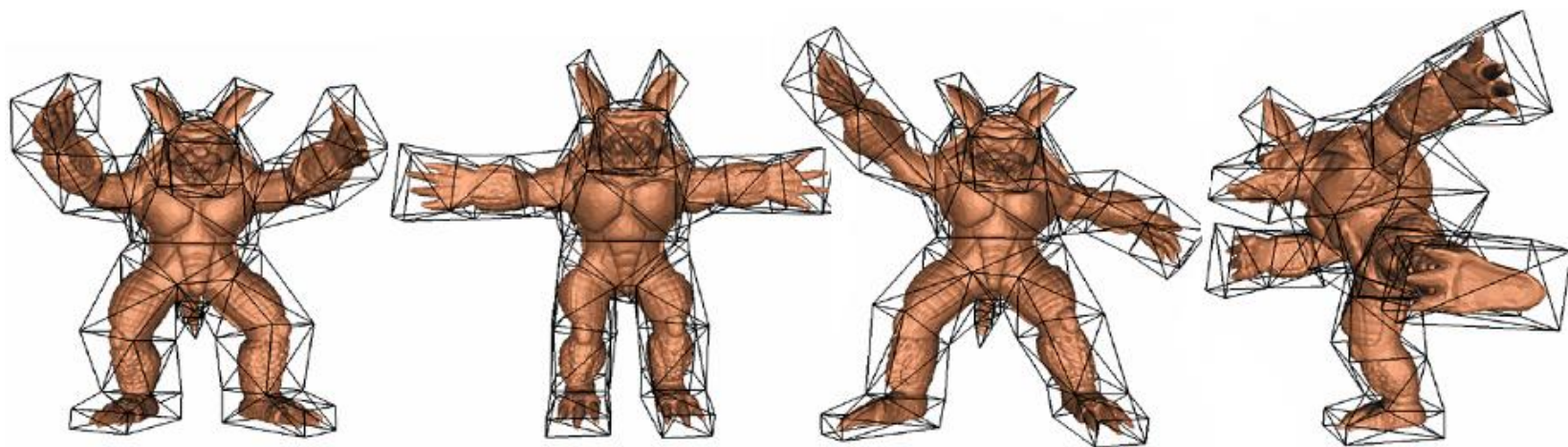
3. Result of a second deformation (twist)



4. Muting the Bend yields a twist.

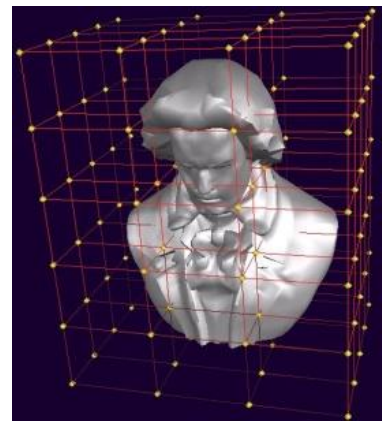
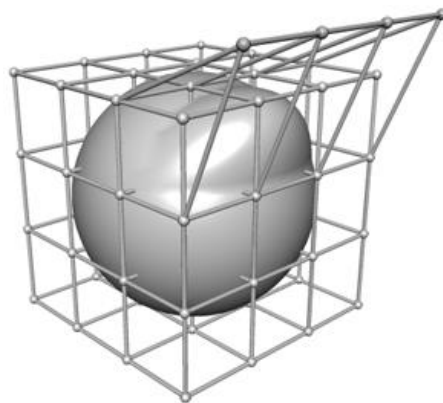
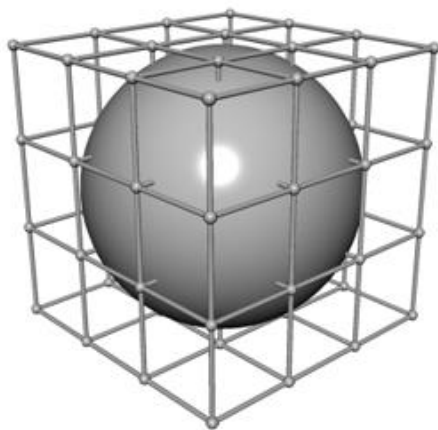
# 自由变形方法FFD及其变种

- Barr的变形方法仅仅局限于Tapering、Twisting等特定的变形，这促使人们寻找更一般的变形方法。
- 这方面的工作包括自由变形方法FFD、扩展的FFD方法EFFD、基于任意拓扑lattice的FFD方法、直接操纵的FFD等。

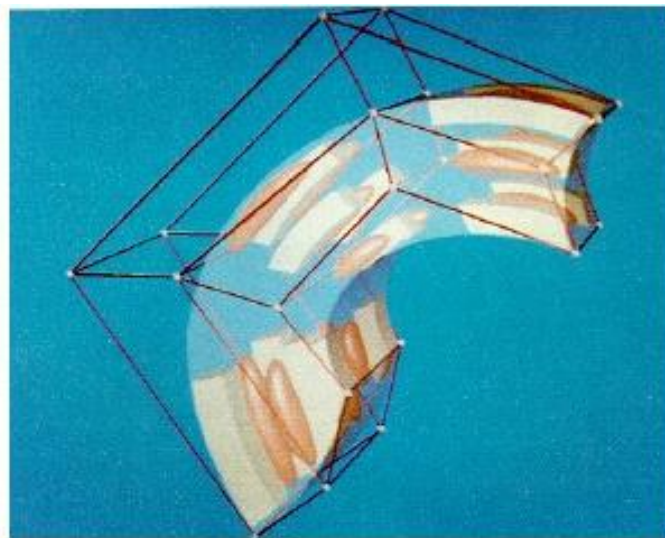
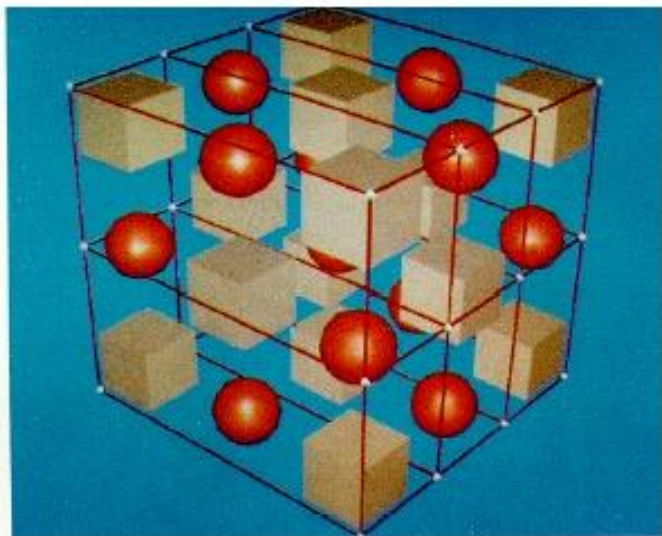


# 自由变形方法FFD

- 1986年，Sederberg等提出了一种非常适合于柔性物体动画的更为一般的方法，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。
- Sederberg T W, Parry S R. Free-form deformation of solid geometric models. Computer Graphics, 1986, 20(4):151~160 (2019年12月24日，引用3399次)







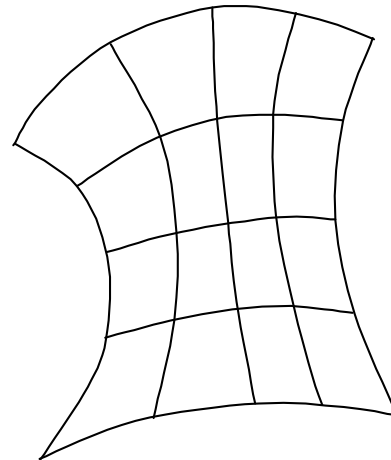
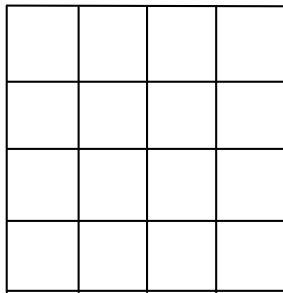
**FFD图示**

# 二维FFD

- 对于二维情形，用双三次Bezier曲面

$$\mathbf{Q}(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 \mathbf{P}_{ij} B_{i,3}(u) B_{j,3}(v)$$

可对二维空间进行变形，它将一正方形区域变换为一弯曲的曲面：



# 三维FFD

- 同样，一张三三次Bezier超曲面

$$Q(u, v, w) = \sum_{i=0}^3 \sum_{j=0}^3 \sum_{k=0}^3 \mathbf{P}_{ijk} B_i(u) B_j(v) B_k(w), \quad (u, v, w) \in [0,1] \times [0,1] \times [0,1]$$

将一正方体映射为一弯曲的物体。

其中  $B_i(u), B_j(v), B_k(w)$  为Bernstein基函数，这个Bezier体由64个控制顶点 $\mathbf{P}_{ijk}$ 来指定。

- 当物体嵌入该正方体时，物体随着 $Q(u,v,w)$ 的变形而变形。

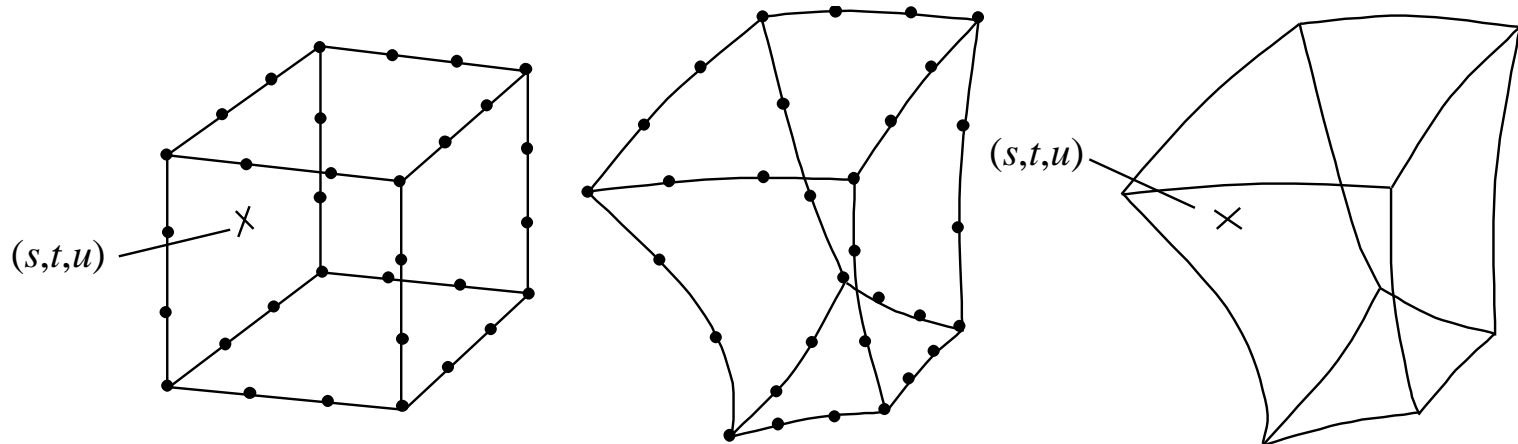


# 三维FFD

- Bezier超曲面具有与Bezier曲面类似的性质。
- 多个Bezier超曲面可拼接生成一段光滑的**Bezier体**，我们称这种复合的Bezier体为一**FFD块**。
- 我们把由三根互相垂直的坐标轴排列的长方体结构的控制顶点称为**Lattice**（晶格）。

# 三维FFD

- 设FFD块的三个坐标方向为  $(S, T, U)$ , 我们采用  $(3l+1)(3m+1)(3n+1)$  个控制顶点定义一个FFD块, 或等价地由  $l \times m \times n$  张三三次Bezier超曲面构成。
- 用FFD块对物体变形的步骤如下:



**FFD变形过程** (a) 确定物体顶点在超曲面的坐标 (b) 通过移动控制顶点变形FFD块 (c) 根据坐标确定顶点变形后的位置

# 三维FFD

1. 确定物体的顶点（或控制顶点）在lattice空间的位置。建立FFD块的局部坐标系：

$$\mathbf{X}(s, t, u) = \mathbf{X}_0 + s\mathbf{S} + t\mathbf{T} + u\mathbf{U}$$

其中 $\mathbf{X}_0$ 为局部坐标系的原点（FFD块的一个角点）， $(\mathbf{S}, \mathbf{T}, \mathbf{U})$ 为Lattice三条互相垂直的边。

- 由于这些矢量大小已反映了Lattice的体积，因而lattice体内的任意一点都有唯一的lattice空间坐标：

$$(s, t, u) \in [0, 1] \times [0, 1] \times [0, 1]$$

- 不妨设lattice为一个正方体，则其控制顶点为：

$$\mathbf{P}_{ijk} = \mathbf{X}_0 + \left(\frac{i}{3l}\right)\mathbf{S} + \left(\frac{j}{3m}\right)\mathbf{T} + \left(\frac{k}{3n}\right)\mathbf{U}$$

其中  $0 \leq i \leq 3l, 0 \leq j \leq 3m, 0 \leq k \leq 3n$  。

- 一般情况下，我们所选的lattice坐标轴与物体空间坐标轴平行，这样，对于FFD块内物体的任意顶点或控制顶点，很容易找到lattice空间坐标 $(s, t, u)$ （线性对应关系！），这只需解一线性方程组：

$$\mathbf{X} = \mathbf{X}_0 + s\mathbf{S} + t\mathbf{T} + u\mathbf{U}$$

- 得到:

$$s = \frac{(\mathbf{T} \times \mathbf{U}) \bullet (\mathbf{X} - \mathbf{X}_0)}{(\mathbf{T} \times \mathbf{U}) \bullet \mathbf{S}}, t = \frac{(\mathbf{S} \times \mathbf{U}) \bullet (\mathbf{X} - \mathbf{X}_0)}{(\mathbf{S} \times \mathbf{U}) \bullet \mathbf{T}}, u = \frac{(\mathbf{S} \times \mathbf{T}) \bullet (\mathbf{X} - \mathbf{X}_0)}{(\mathbf{S} \times \mathbf{T}) \bullet \mathbf{U}}$$

- 到此为止, **Lattice**空间和物体空间仅仅是比例缩放的关系。

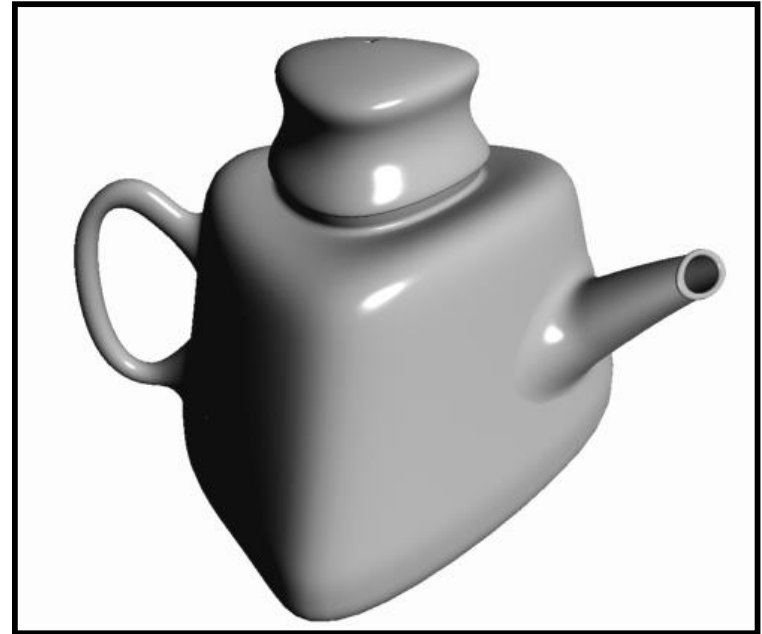
2. 变形FFD块。根据动画设计的需要, 移动控制顶点 $\mathbf{P}_{ijk}$ 。

### 3. 确定顶点变形后的位置。

- 经变形后物体空间与lattice空间完全不同，lattice空间到物体空间的转换包含了物体的变形。
- 给定lattice坐标 $(s, t, u)$ ，我们先找到它所在的Bezier超曲面，并将它转换成该超曲面的局部坐标 $(u, v, w)$ ，该超曲面对应的索引为 $(ls, mt, nu)$ 的整数部分，可写成 $(is, it, iu)$ ，得到超曲面的局部坐标为：

$$(u, v, w) = (ls - is, mt - it, nu - iu)$$

- 代入超曲面方程，便得到变形后的顶点 $Q(u, v, w)$ 。



茶壶的FFD变形 (a) 变形前 (b) 变形后

# FFD的性质

- **FFD**一个重要性质是参数曲面变形后仍然是参数曲面。设参数曲面为：

$$x = f(\alpha, \beta), y = g(\alpha, \beta), z = h(\alpha, \beta)$$

- **FFD**变换为  $\mathbf{X}_{ffd} = \mathbf{X}(x, y, z)$ ，则变形后的曲面为

$$\mathbf{X}_{ffd}(\alpha, \beta) = \mathbf{X}(f(\alpha, \beta), g(\alpha, \beta), h(\alpha, \beta))$$

- 显然，它仍然为参数曲面。



# FFD的性质

- **FFD的另一个性质使能对变换前后的体积变化进行控制。** 设FFD为

$$\mathbf{X}_{ffd}(x, y, z) = (F(x, y, z), G(x, y, z), H(x, y, z))$$

- 它的Jacobian行列式为:

$$\mathbf{J}(\mathbf{X}_{ffd}) = \begin{vmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} & \frac{\partial F}{\partial z} \\ \frac{\partial G}{\partial x} & \frac{\partial G}{\partial y} & \frac{\partial G}{\partial z} \\ \frac{\partial H}{\partial x} & \frac{\partial H}{\partial y} & \frac{\partial H}{\partial z} \end{vmatrix}$$

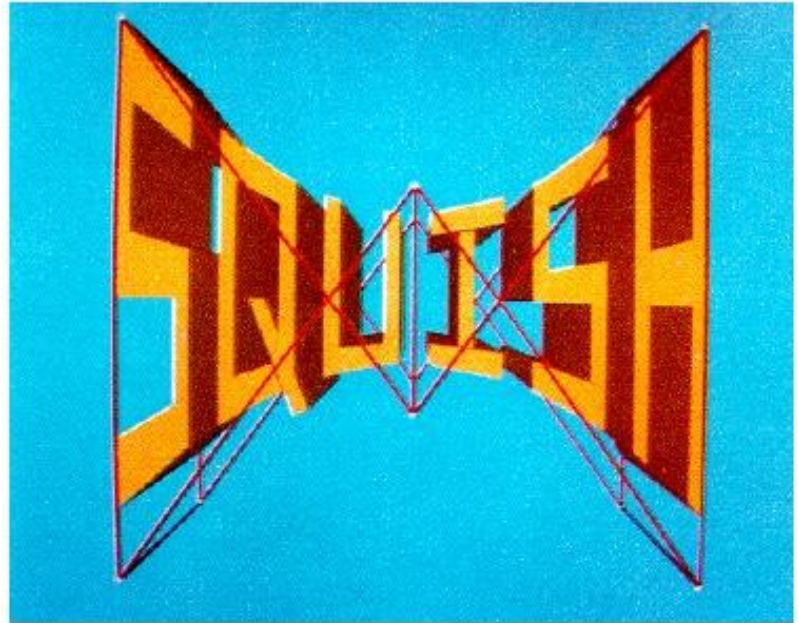
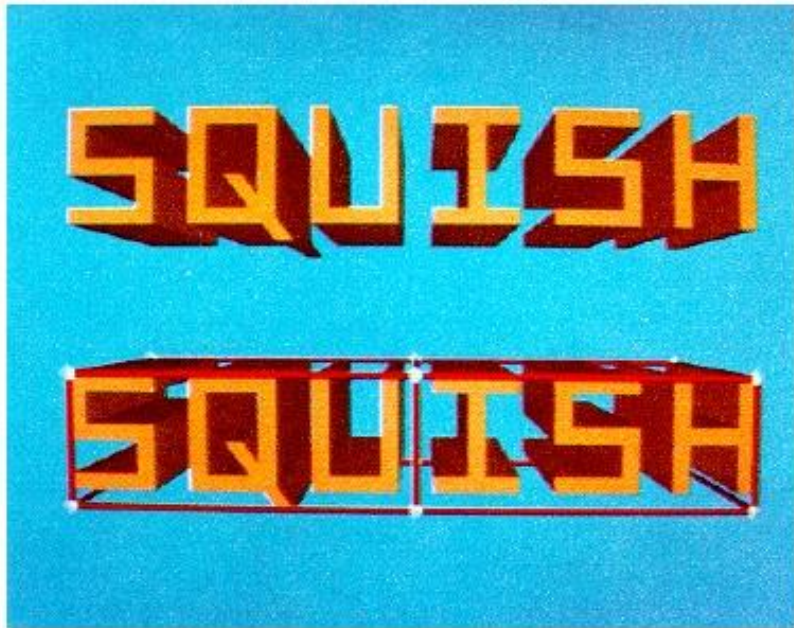
# FFD的性质

- 如果变形前物体的体积元为 $dx dy dz$ ，则变形后的体积元为  $\mathbf{J}(X_{ffd})dx dy dz$ ，变形后物体的体积为：

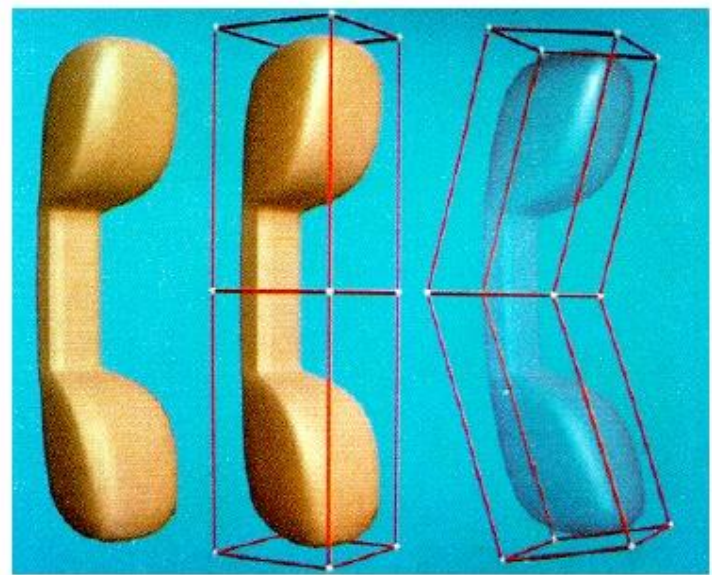
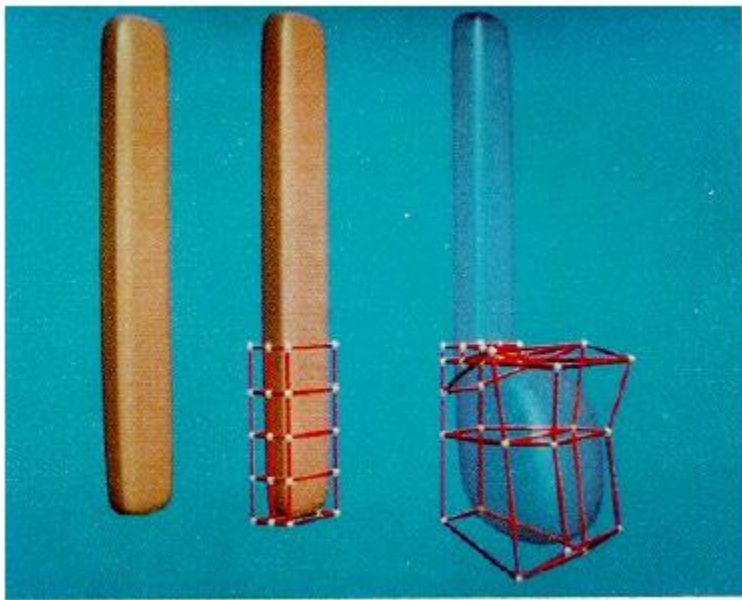
$$V = \iiint_{\Omega} \mathbf{J}(X_{ffd}) dx dy dz$$

- 若要估计体积的变化，只需估计 $\mathbf{J}(X_{ffd})$ 的上下限。如果 $\mathbf{J}(X_{ffd})$ 可表达成三变量Bernstein多项式时，变形后物体的体积可采用其系数进行估计。
- 如果  $\mathbf{J}(X_{ffd}) \equiv 1$ ，则该FFD是保体积的。

# FFD应用



# FFD应用

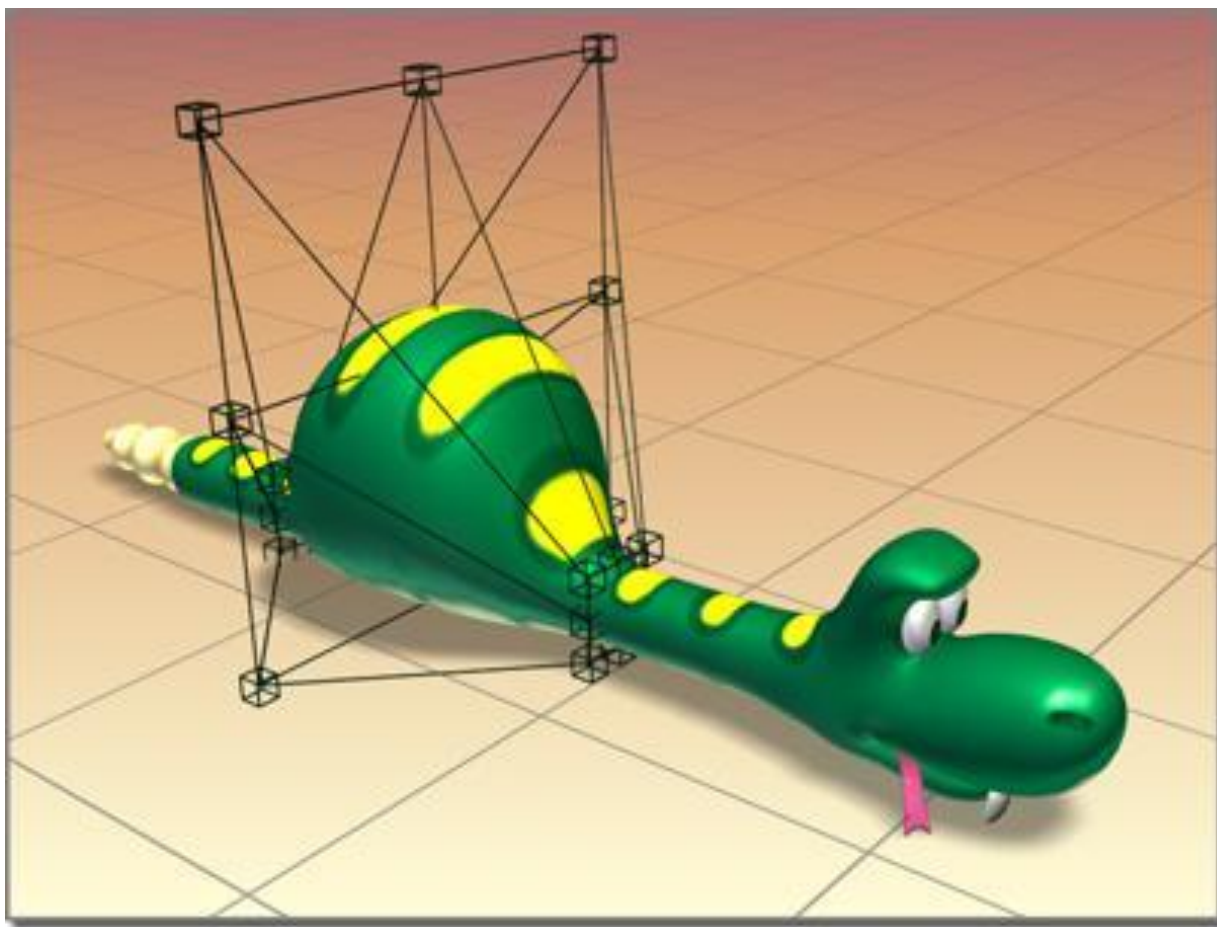


# FFD应用

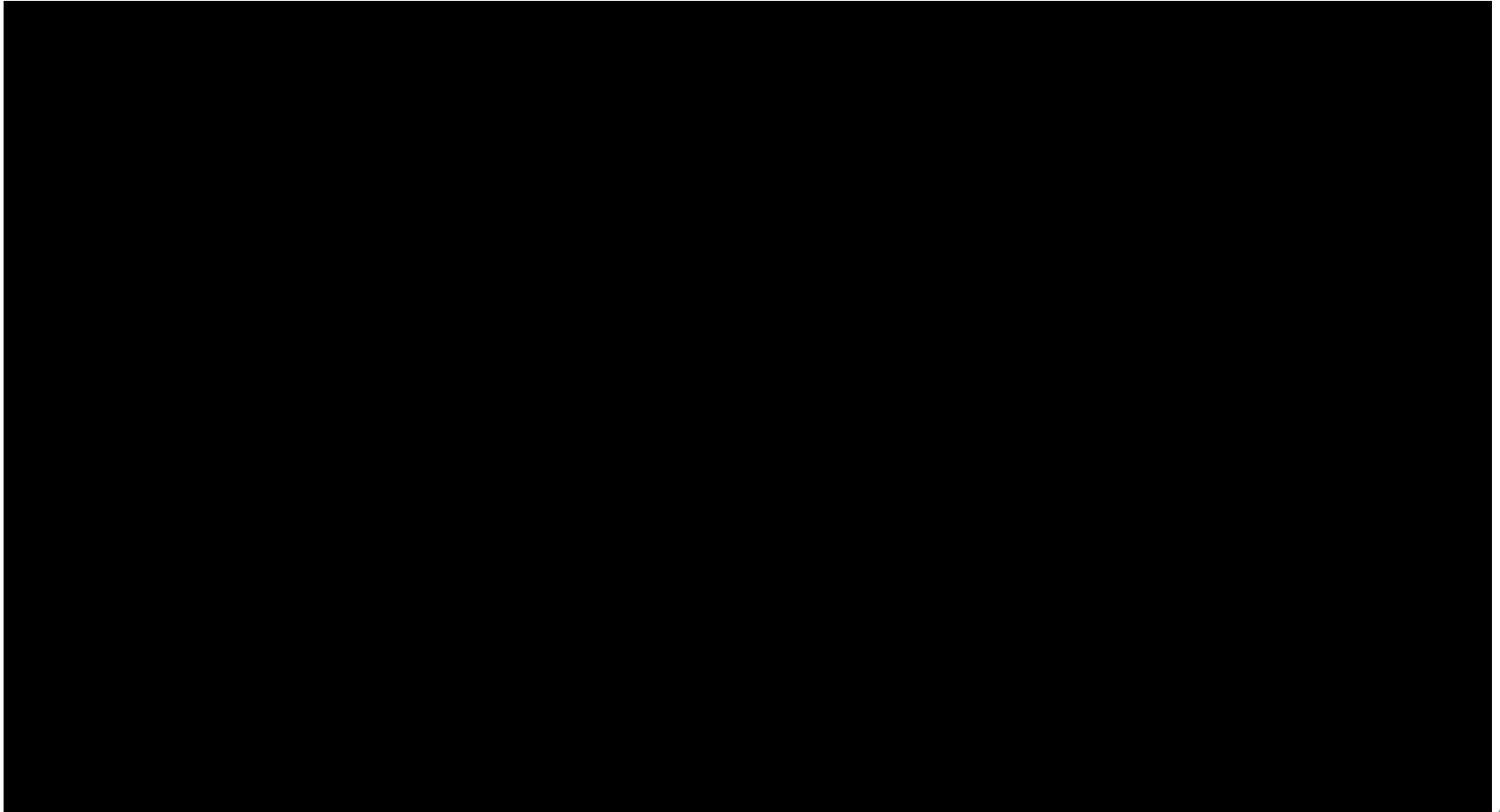




# FFD应用



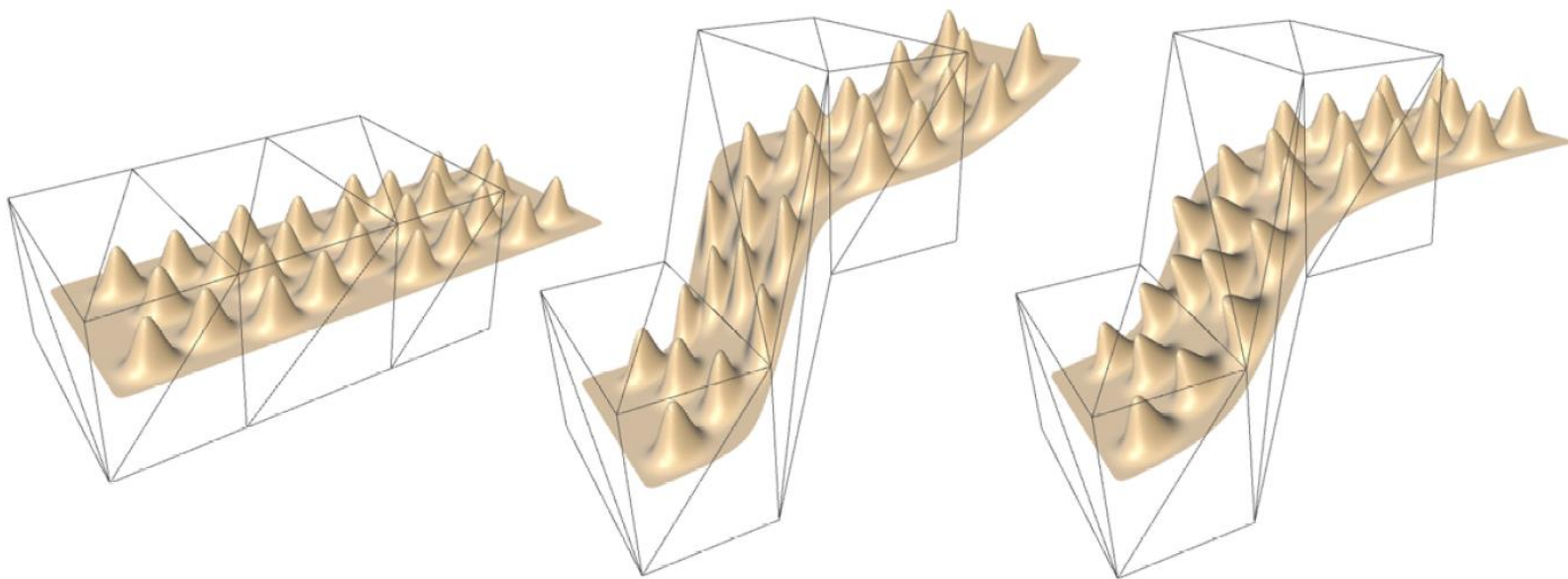
# FFD演示



# FFD总结

- 优点：
  - 与物体的几何表示无关；
  - 直观；
  - 有效；
- 缺点：
  - **Lattice**为平行六面体形状；
  - 不遵循物理规律；
  - 细节很难被保持；



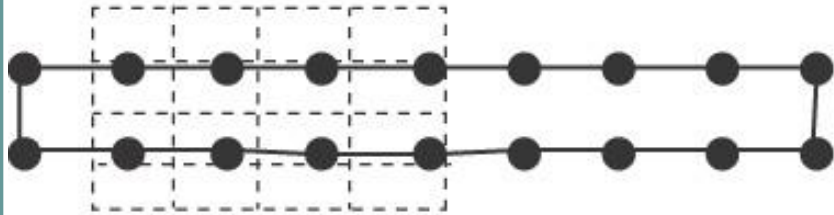


原始模型

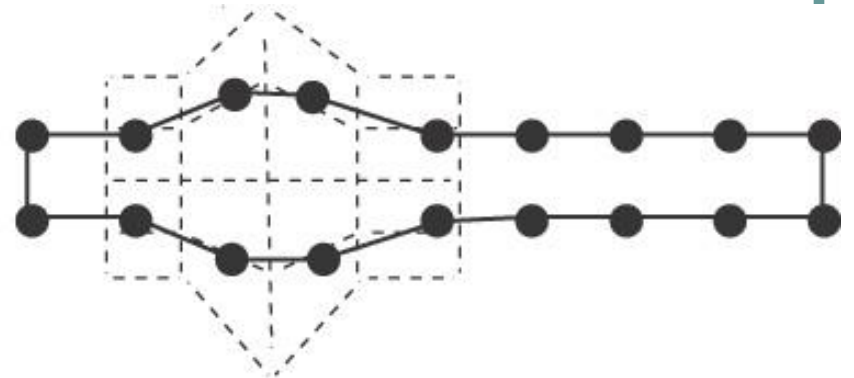
**Green Coordinate可  
保持细节**

Lipman, Y., Levin, D., & Cohen-Or, D. (2008). Green coordinates. *ACM Transactions on Graphics (TOG)*, 27(3), 78.

# Animated FFD



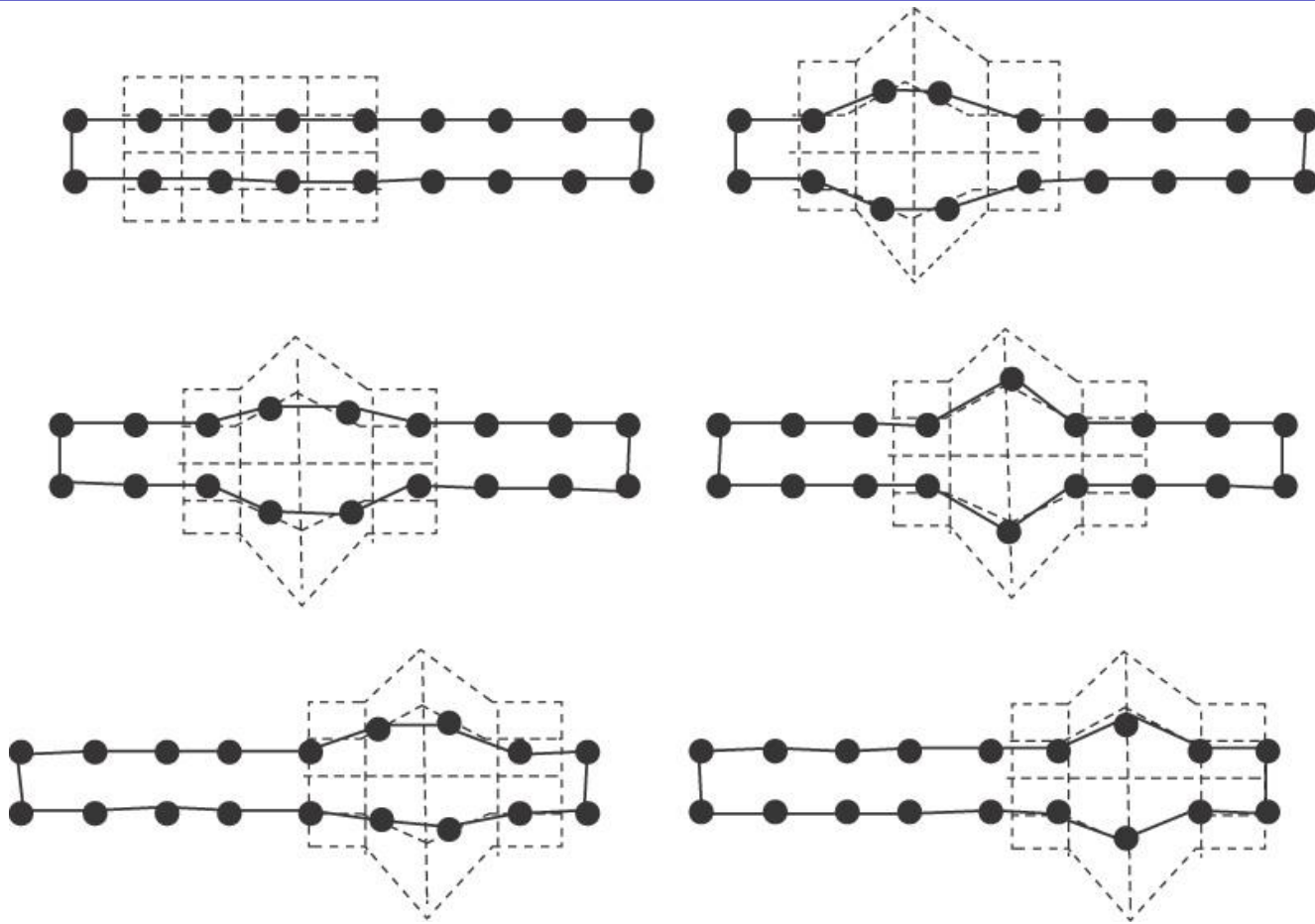
Undeformed object



Deformed object

变形工具应用到一物体

# Animated FFD

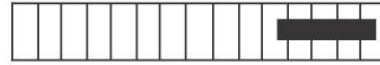


通过移动变形工具相对于物体的位置来获得变形

# Animated FFD

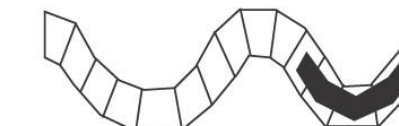
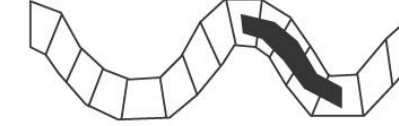
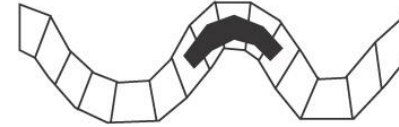
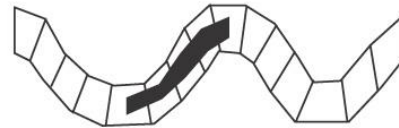
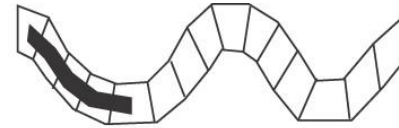
# Animated FFD

**Deformation by moving  
the object through FFD  
space**



Object traversing the logical FFD coordinate space

**In logical FFD space**

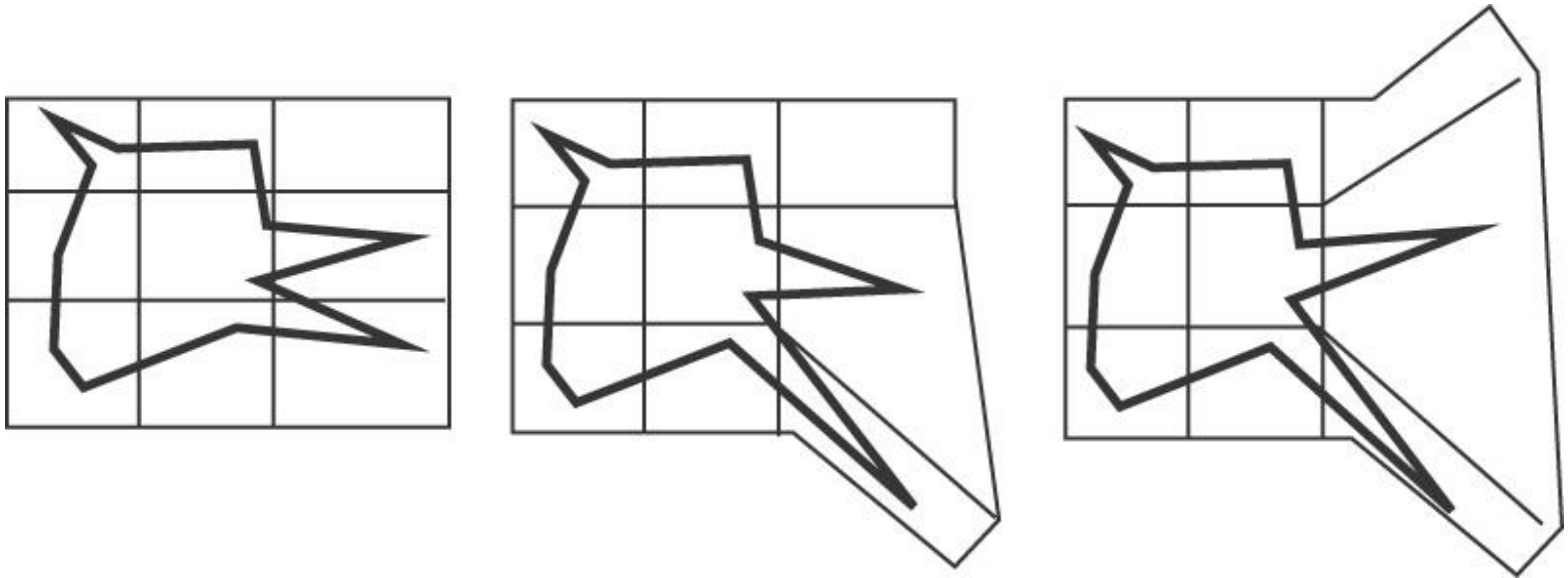


Object traversing the distorted space

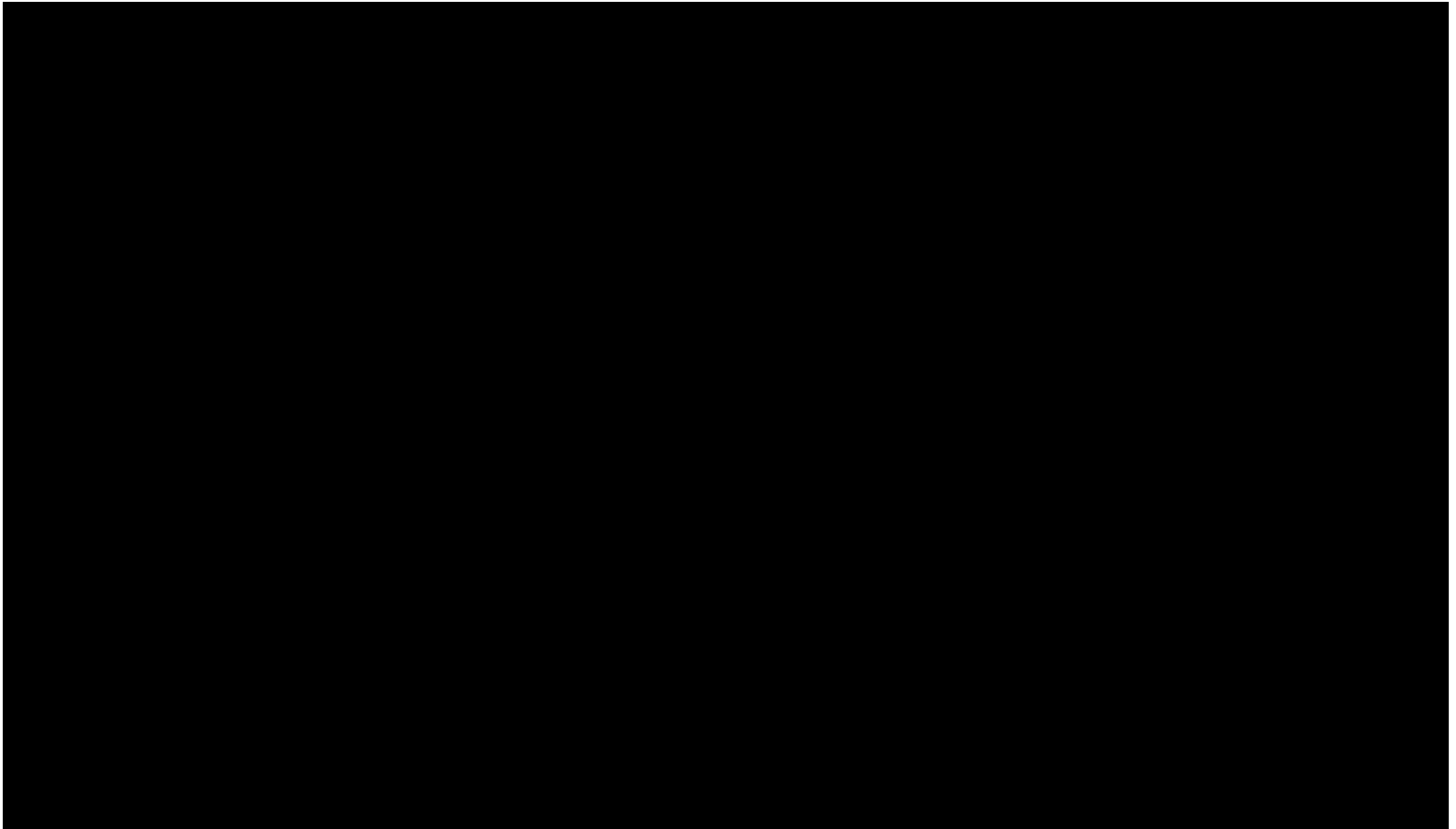
**In distorted FFD space**

# Animated FFD

- Animating the FFD control points using, e.g., key-frame animation or by the result of **physically based simulation**.



# Animated FFD



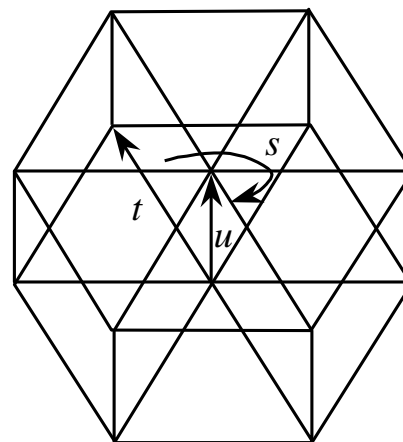
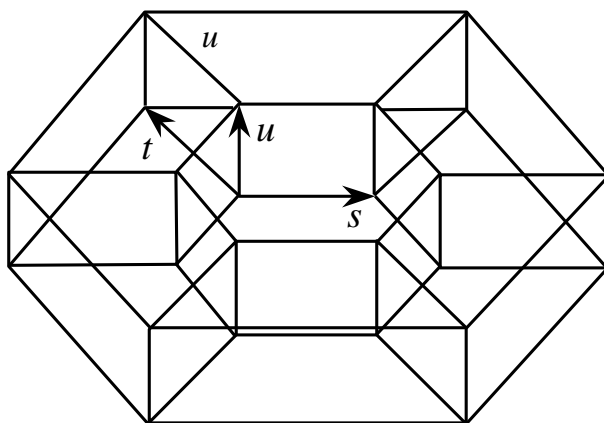
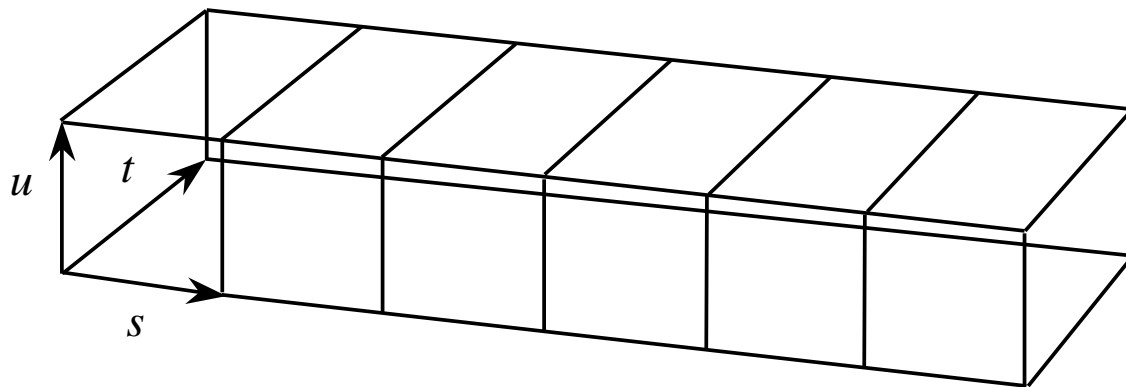
# 扩展的FFD方法EFFD

- **FFD**是一个非常直观的变形工具，但它只适合于平行六面体的lattice形状。
- 1990年，Coquillart提出了一种称为扩展的FFD方法，该方法允许非平行六面体的lattice形状，从而能实现更任意的变形。
- Ref: Coquillart S. Extended free-form deformation: a sculpturing tool for 3D geometric modeling. Computer Graphics, 1990, 24(4):187~196



# 扩展的FFD方法EFFD

- EFFD型lattice和FFD型lattice的区别在于：它允许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。
- 例如，对于下图所示的棱柱形lattice，可以用下面的方法来构造。首先，在Lattice空间，构造由六个超曲面组成的FFD块，见图(a)。然后，通过合并平面 $s=0$ 和 $s=1$ 的控制顶点，把lattice两端的两块超平面融合在一起，如图(b)所示。最后，合并所有沿轴的顶点，即使所有满足且和且的点重合，如图(c)所示。这样，我们得到一个基本的EFFD块。
- 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。在合并超曲面块的控制顶点时，**必须注意合并后的块之间的切向连续性问题。**



## 棱柱形EFFD的构造

(a) Lattice空间的超曲面 (b) 融合两端平面 (c) 合并轴的控制顶点

# 扩展的FFD方法EFFD

- **EFFD**块构造好以后，我们可以采用与**FFD**方法类似的方法来使物体变形。
- 在**FFD**方法中，由于**FFD**块的三条轴与景物空间的坐标轴重合，景物空间的点在lattice空间的局部坐标很容易求得（**线性关系！**）。
- 但在**EFFD**中，两个空间之间不再有这种简单的对应关系，**通常需要迭代求解**。采用**EFFD**块对物体变形的步骤如下：

# 扩展的FFD方法EFFD

1. 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

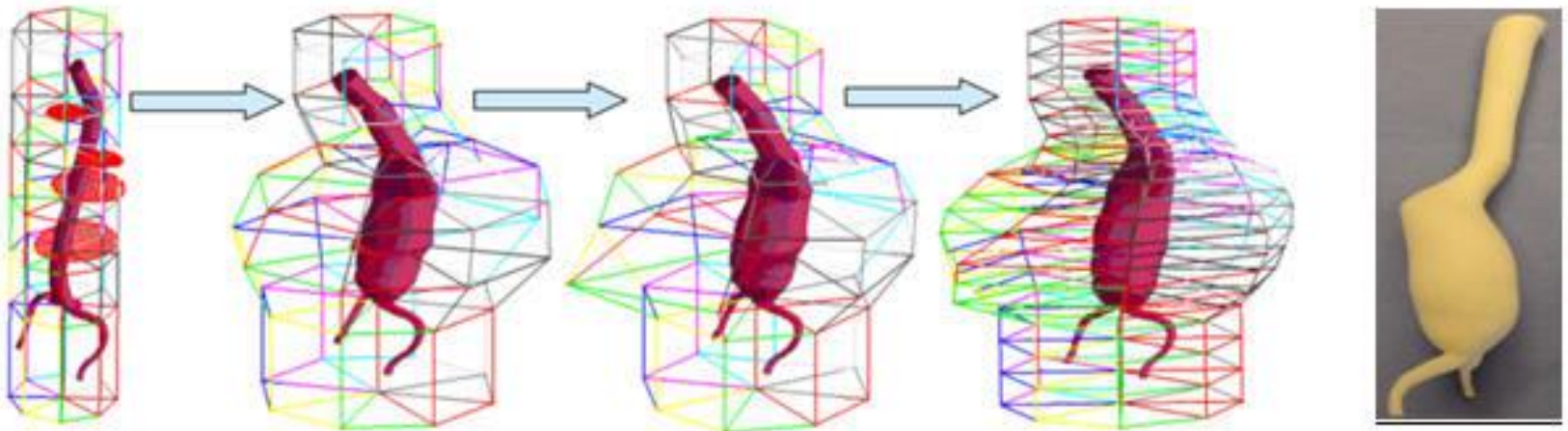
$$Q(u, v, w) = \sum_{i=0}^3 \sum_{j=0}^3 \sum_{k=0}^3 P_{ijk} B_i(u) B_j(v) B_k(w),$$

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 $P_{ijk}$ 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。

# 扩展的FFD方法EFFD

2. 根据动画设计的需要，移动EFFD块的控制顶点。
  3. 根据超曲面方程，求得变形后的位置。
- EFFD方法的优点是允许更加复杂的变形空间，但同时也丧失了FFD方法的一些优点。
  - EFFD方法的缺点是：(1) 在移动内部控制顶点时必须保持块与块之间的连续性。(2) 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。

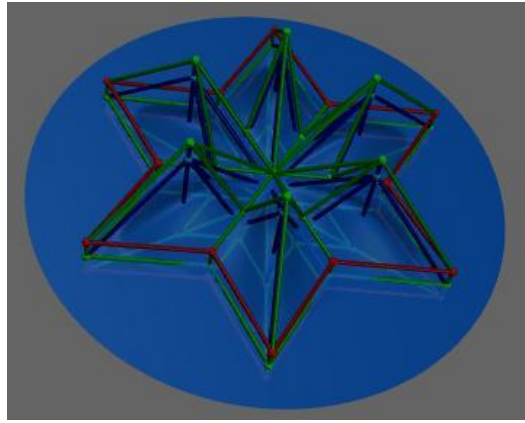
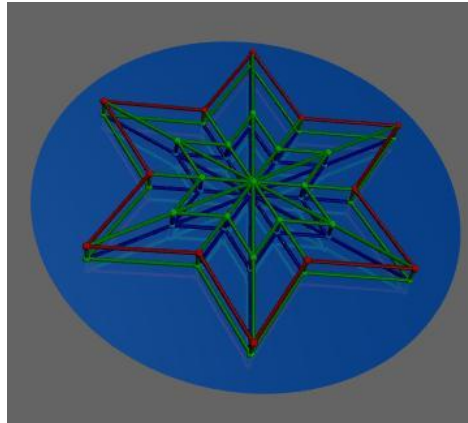
# Example



# 基于任意拓扑Lattice的FFD方法

- Sederberg的FFD方法要求lattice为平行六面体，Coquillart的EFFD允许多个lattice组合成任意形状lattice的空间。
- 1996年，MacCracken提出了一种允许lattice为任意拓扑形状的更一般的FFD方法。在该方法中，变形空间由细分(subdivision)方法生成的体来定义。
- 在细分过程中，lattice一步步加细，生成一系列收敛于一三维区域的lattice。加细过程定义了一个嵌入点的伪参数化方法，当lattice的控制顶点改变时，嵌入的物体产生变形。

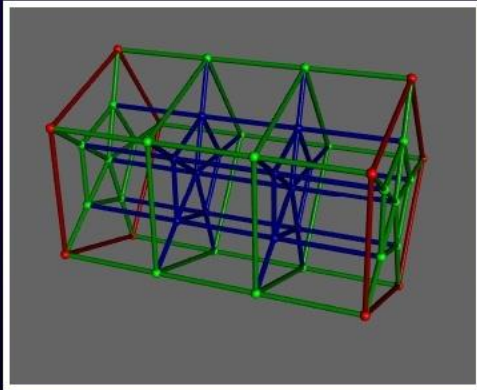
# Example



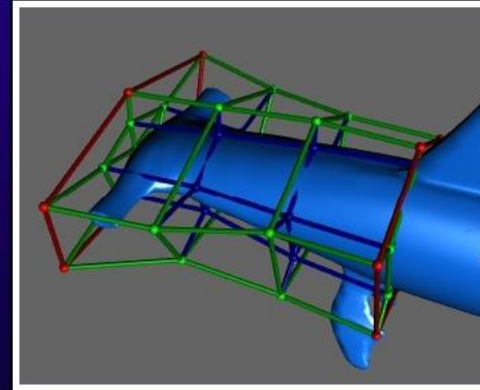


# Example

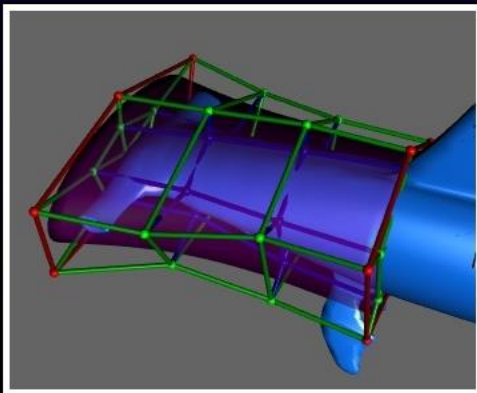
**Design the Lattice**



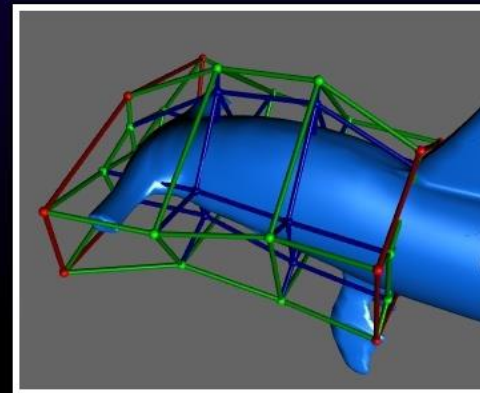
**Orient about the Object**



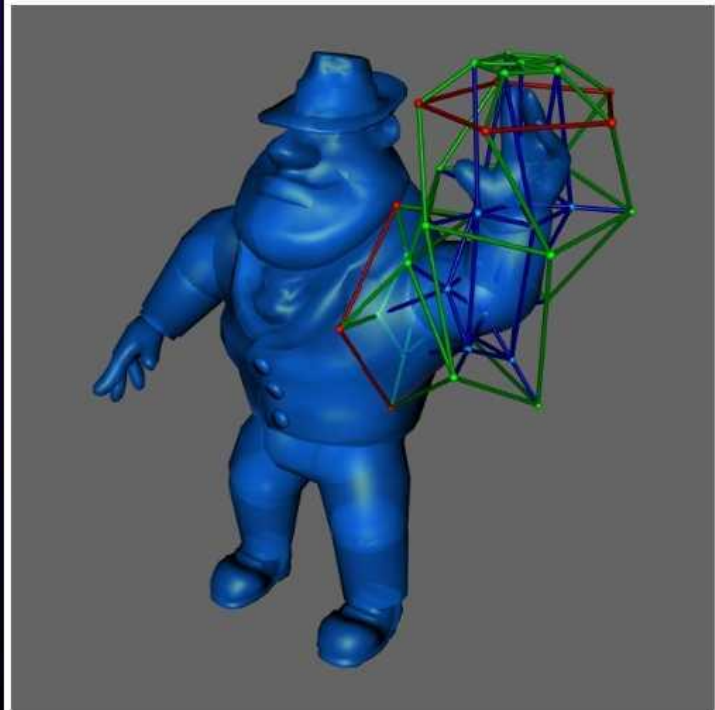
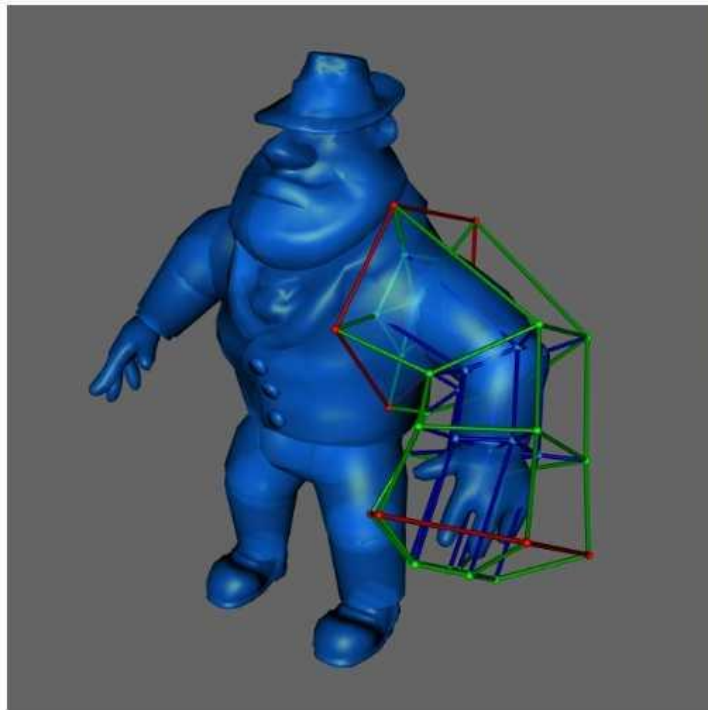
**Freeze the Lattice**



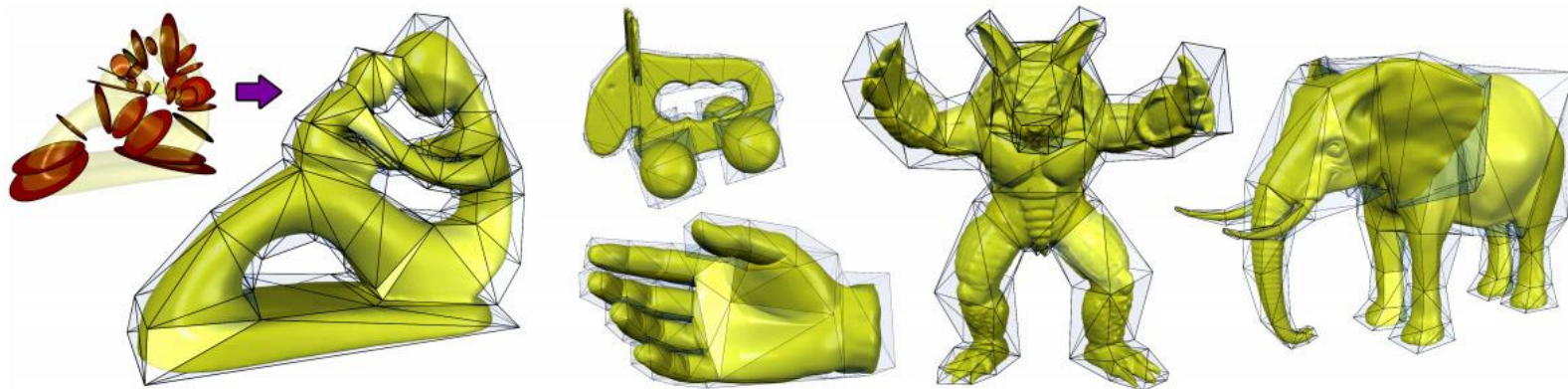
**Deform the Lattice**



# Example



# 基于Cage的变形原理



- **Cage:** 一个包裹高分辨率模型的**低分辨率控制网格**。
- 基于**cage**的变形方法具有**直观性**、**简单性**和**高效性**，在最近十多年得到了越来越多的关注。
- **变形原理:** (1). 构建模型的**Cage**; (2). 计算模型的**Cage**坐标; (3). 编辑**Cage**模型; (4). 把**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。

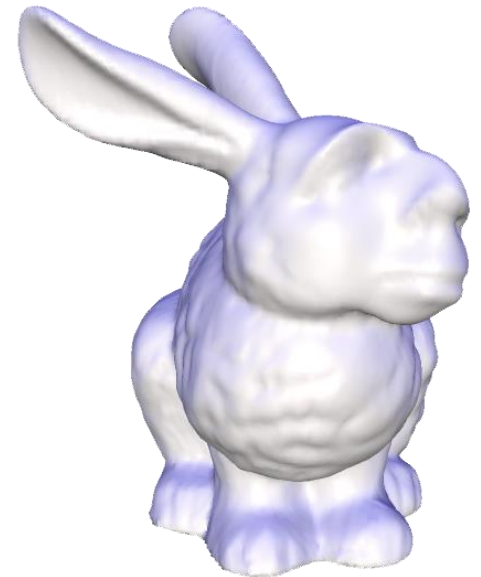
# 与物体表示有关的变形

- **Mesh Editing**
- **Mesh Deformation**

**Digital Geometry Processing !**

# 基于Laplacian坐标的Mesh Deformation

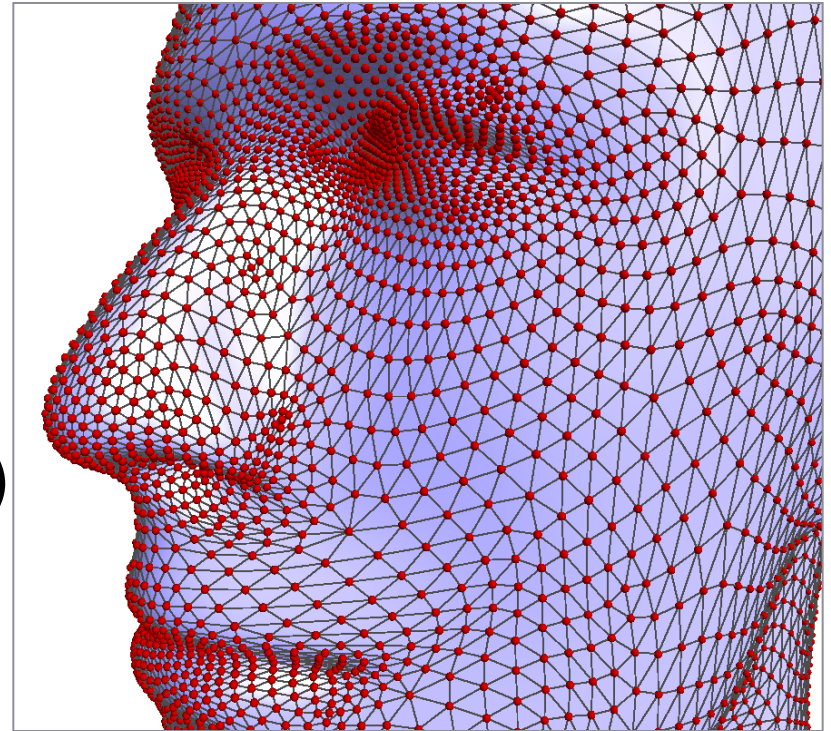
- Differential surface representation
- Ideas and applications
  - Compact shape representation
  - Mesh editing and manipulation



Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, Hans-Peter Seidel: Laplacian Surface Editing. Symposium on Geometry Processing 2004: 175-184

# Triangle mesh

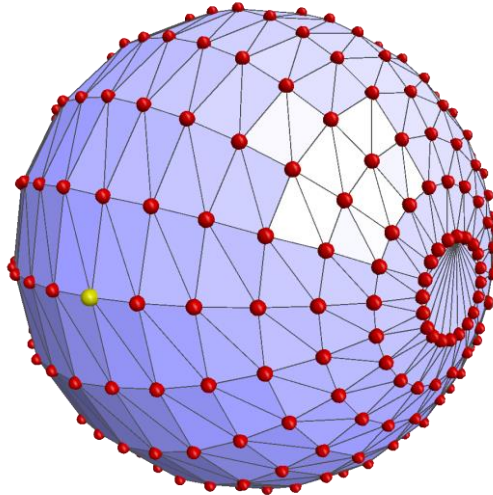
- Geometry:
  - Vertex coordinates
$$\begin{aligned} &(x_1, y_1, z_1) \\ &(x_2, y_2, z_2) \\ &\dots \\ &(x_n, y_n, z_n) \end{aligned}$$
- Connectivity (the graph)
  - List of triangles
$$\begin{aligned} &(i_1, j_1, k_1) \\ &(i_2, j_2, k_2) \\ &\dots \\ &(i_m, j_m, k_m) \end{aligned}$$





# Motivation

- Meshes are great, but:
  - Topology is explicit, thus hard to handle
  - Geometry is represented in a *global* coordinate system
    - Single Cartesian coordinate of a vertex doesn't say much



# Differential coordinates

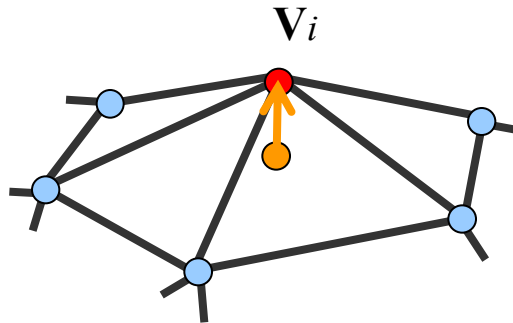
- Represent *local detail* at each surface point
  - better describe the shape
- Linear transition from global to differential
- Useful for operations on surfaces where surface details are important





# Differential coordinates (Laplacian坐标)

- Detail = surface – *smooth*(surface)
- Smoothing = averaging



$$\delta_i = \mathbf{v}_i - \frac{1}{d_i} \sum_{j \in N(i)} \mathbf{v}_j$$

$$\delta_i = \sum_{j \in N(i)} \frac{1}{d_i} (\mathbf{v}_i - \mathbf{v}_j)$$

# Laplacian matrix

- The transition between the  $\delta$  and  $xyz$  is linear:

$$\begin{pmatrix} \begin{matrix} & & & & \\ & & & & \\ & & \mathbf{L} & & \\ & & & & \\ & & & & \end{matrix} \\ \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \delta_1^{(x)} \\ \delta_2^{(x)} \\ \vdots \\ \vdots \\ \delta_n^{(x)} \end{pmatrix}$$

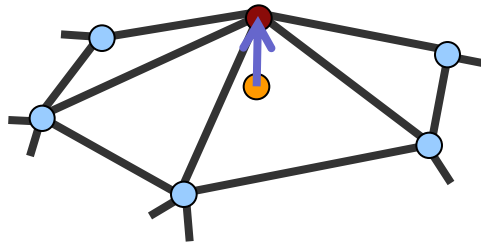
$$A_{ij} = \begin{cases} 1 & i \in N(j) \\ 0 & \text{otherwise} \end{cases}$$

$$D_{ij} = \begin{cases} d_i & i = j \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1} \mathbf{A}$$

# Laplacian matrix

- The transition between the  $\delta$  and  $xyz$  is linear:



$$\delta_i = \sum_{j \in N(i)} w_{ij} (\mathbf{v}_i - \mathbf{v}_j)$$

$$\begin{array}{l} \mathbf{L} \mathbf{v}_x = \delta_x \\ \mathbf{L} \mathbf{v}_y = \delta_y \\ \mathbf{L} \mathbf{v}_z = \delta_z \end{array}$$

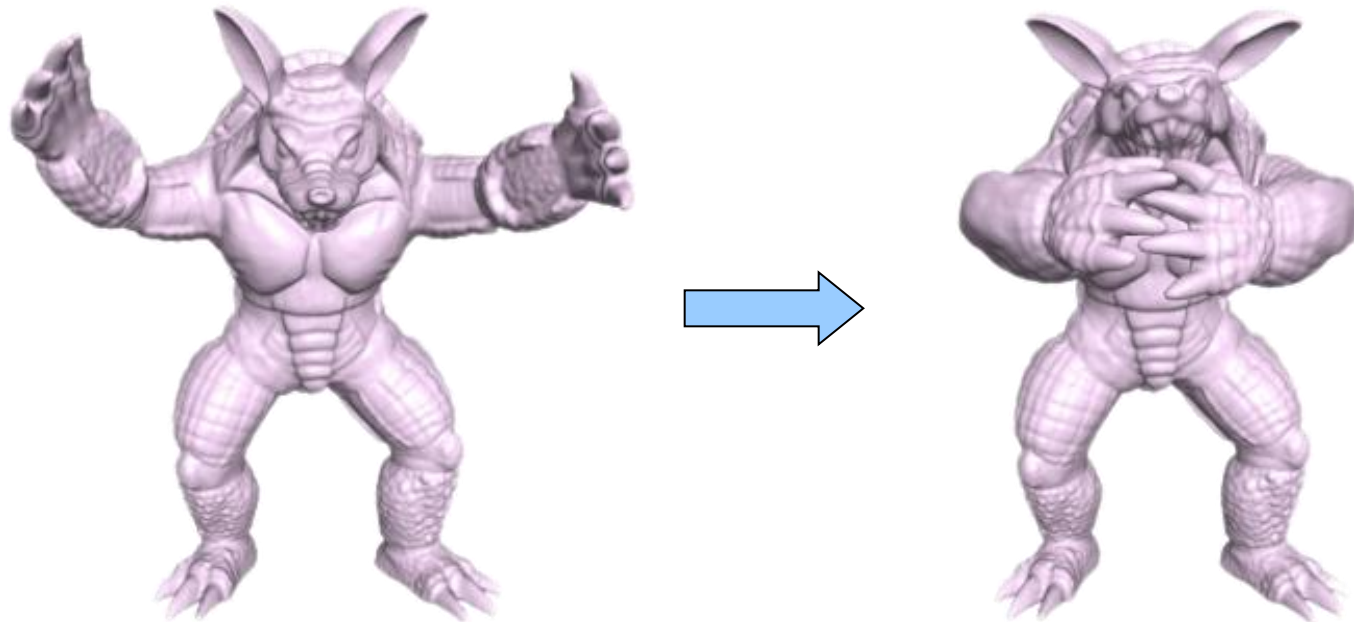
# Basic properties

- $\text{Rank}(L) = n - c$  ( $n - 1$  for connected meshes)
- We can reconstruct the xyz geometry from delta up to translation

$$L\mathbf{x} = \delta$$

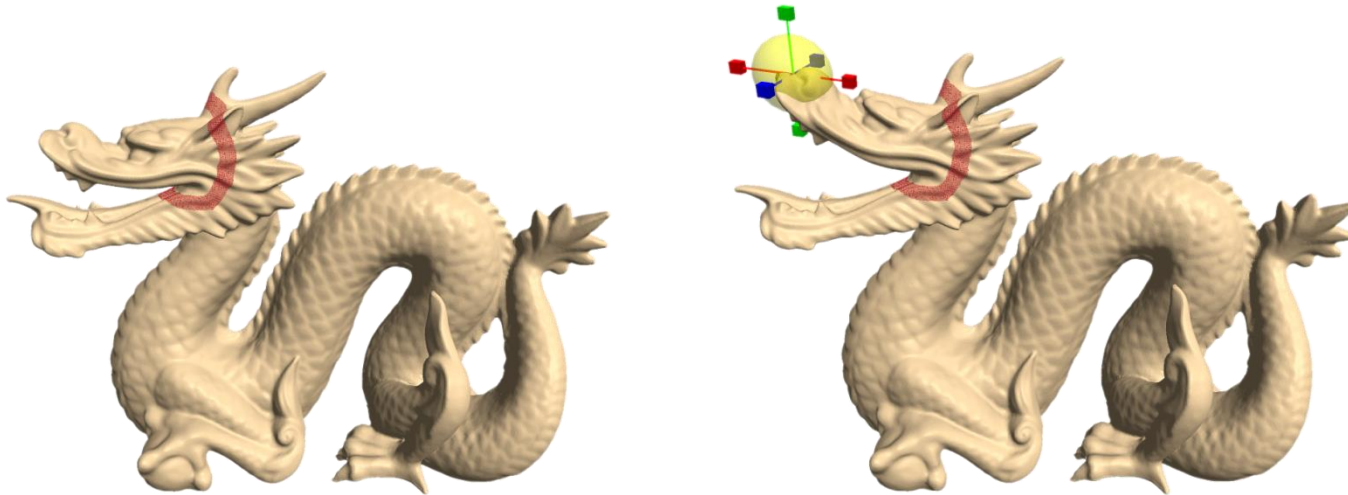
# Differential coordinates for editing

- Intrinsic surface representation
- Allows various surface editing operations that preserve local surface details



# Why differential coordinates?

- Local detail representation – enables **detail preservation** through various modeling tasks
- Representation with **sparse** matrices
- Efficient **linear** surface reconstruction



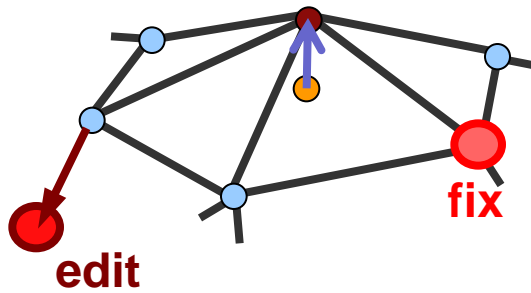
# Editing framework

- The spatial constraints will serve as modeling constraints
- Reconstruct the surface every time the modeling constraints are changed

Detail constraints:  $L\mathbf{x} = \delta$

Modeling constraints:  $x_j = c_j, \quad j \in \{j_1, j_2, \dots, j_k\}$

# Reconstruction



$$\begin{bmatrix} L \\ 1 \\ 1 \end{bmatrix} \mathbf{v}_x = \begin{bmatrix} \delta_x \\ c_x \\ e_x \end{bmatrix}$$

$$\begin{bmatrix} L \\ 1 \\ 1 \end{bmatrix} \mathbf{v}_y = \begin{bmatrix} \delta_y \\ c_y \\ e_y \end{bmatrix}$$

$$\begin{bmatrix} L \\ 1 \\ 1 \end{bmatrix} \mathbf{v}_z = \begin{bmatrix} \delta_z \\ c_z \\ e_z \end{bmatrix}$$



# Reconstruction

$$\begin{array}{c} \text{L} \\ 1 \\ 1 \end{array} \begin{array}{c} \mathbf{v}_x \end{array} = \begin{array}{c} \delta_x \\ \mathbf{c}_x \\ \mathbf{e}_x \end{array}$$

$$\tilde{\mathbf{x}} = \arg \min_{\mathbf{x}} \left( \left\| \textcolor{green}{L} \textcolor{blue}{\mathbf{x}} - \textcolor{red}{\boldsymbol{\delta}}_x \right\|^2 + \sum_{s=1}^k \left| \textcolor{blue}{x}_k - \textcolor{red}{c}_k \right|^2 \right)$$

# Reconstruction

$$\begin{bmatrix} \mathbf{L} \\ 1 \\ 1 \end{bmatrix} \mathbf{v}_x = \begin{bmatrix} \delta_x \\ \mathbf{c}_x \\ \mathbf{e}_x \end{bmatrix}$$

$$\mathbf{A} \mathbf{x} = \mathbf{b}$$

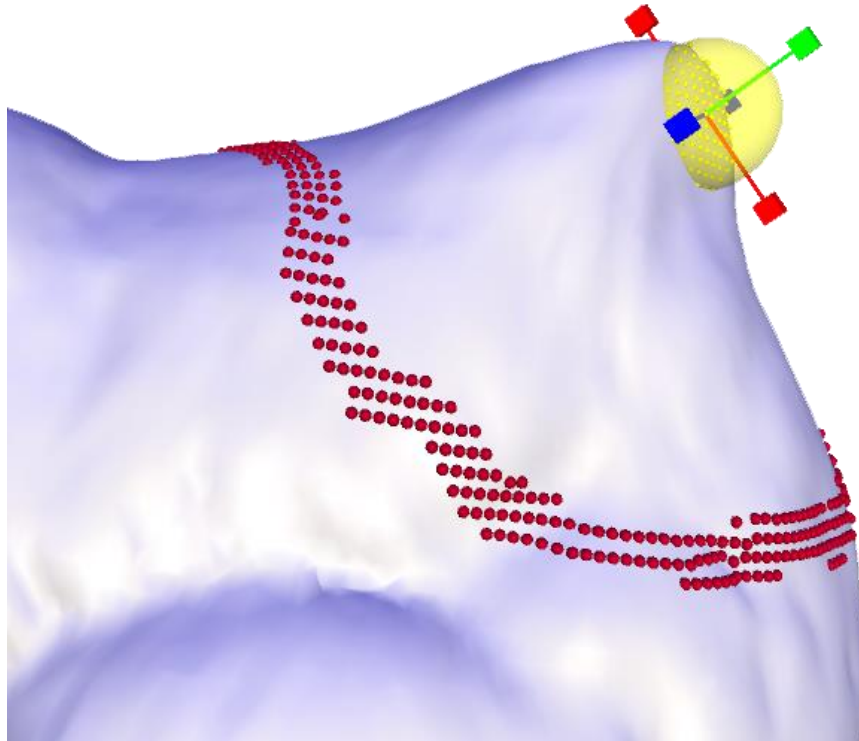
Normal Equations:

$$\mathbf{A}^T \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{b}$$

$$\mathbf{x} = \underbrace{(\mathbf{A}^T \mathbf{A})^{-1}}_{\text{compute once}} \mathbf{A}^T \mathbf{b}$$

# Editing framework

- ROI is bounded by a belt (static anchors)
- Manipulation through handle(s)



# Demo

## Laplacian Mesh Editing

A short editing session  
with the *Octopus*

# Demo



**The End**