

第3章 关键帧动画

3.1 关键帧动画概述

3.2 关键帧技术类型

3.3 关键帧插值计算

3.4 插值函数与参数的选择

3.5 样条驱动动画技术

3.1 关键帧动画概述

3.1.1 什么是关键帧动画

- **逐帧动画**：为每一帧都提供一张单独图片的动画。它有一个关键的缺点就是由于每一帧的图片都需要单独提供，制作起来比较麻烦，因此图片量也比较大。

为了解决这个问题

- **关键帧动画**：原理是只需要提供一张图片（或对象），然后对这张图片（或对象）进行处理产生出不同的效果。例如进行**放缩**处理产生出大小不同的效果，还有进行**透明度处理**产生出透明度不同的效果等，连续播放处理出来的效果就形成了动画。

3.1 关键帧动画概述

3.1.1 什么是关键帧动画

- **关键帧动画**：这种处理的效果有限，只能进行一些常规的处理，如改变**大小**、**透明度**、**位置**、**角度**等，所以关键帧动画**不适合**表现复杂的变形。



关键帧动画实现金鱼从小长到大变化过程

- 例如在**逐帧动画**中提到的描述一株向日葵从苗芽状态成长到绽放花朵的过程动画就不能用关键帧动画来完成，因为这个动画中的效果不是仅仅通过**变形**就能实现的，而是需要对图片中**内容**的改变，这种情况下就**只有采用逐帧动画**了。

3.1 关键帧动画概述

3.1.1 什么是关键帧动画

- **关键帧动画**：这种处理的效果有限，只能进行一些常规的处理，如位置、透明度、位置、颜色等。所以关键帧



- 例如在**逐帧动画**中提到的描述一株向日葵从苗芽状态成长到绽放花朵的过程动画就不能用关键帧动画来完成，因为这个动画中的效果不是仅仅通过**变形**就能实现的，而是需要对图片中**内容**的改变，这种情况下就**只有采用逐帧动画**了。

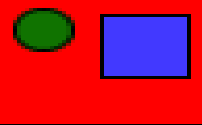

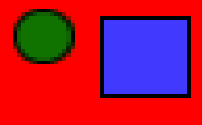

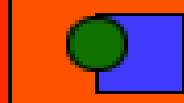


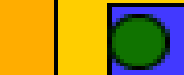
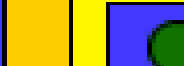
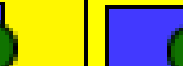
3.1 关键帧动画概述

3.1.2 关键帧动画相关概念

- 关键帧动画是对图片(或对象)的**属性值**进改变，一般可以改变的属性为**大小**、**透明度**、**位置坐标**和**旋转角度**、**方向**等，因此关键帧动画也可以称为**属性动画**。
- 关键帧动画概念及其属性值
 - **开始值**：起始帧中的的属性值；
 - **结束值**：结束帧中的的属性值；
 - **关键值**：在起始帧和结束帧之间产生的属性值，这些值都是按照一定的方式计算出来的；
 - **持续时间**：动画持续的时间；
 - **关键帧**：开始值和结束值表示的帧都称为关键帧；
 - **过渡帧**：关键值表示的帧称为过渡帧；
 - **插值器**：可以理解为关键值产生的计算方式。

3.1 关键帧动画概述

3.1.2 关键帧动画相关概念

1	2	3	4	5	6	7	8
							
1	2	3	4	5	6	7	8
							

- 关键帧动画概念及其属性值
 - **开始值**：起始帧中的属性值；
 - **结束值**：结束帧中的属性值；
 - **关键值**：在起始帧和结束帧之间产生的属性值，这些值都是按照一定的方式计算出来的；
 - **持续时间**：动画持续的时间；
 - **关键帧**：开始值和结束值表示的帧都称为关键帧；
 - **过渡帧**：关键值表示的帧称为过渡帧；
 - **插值器**：可以理解为关键值产生的计算方式。

第3章 关键帧动画

3.1 关键帧动画概述

3.2 关键帧技术类型

3.3 关键帧插值计算

3.4 插值函数与参数的选择

3.5 样条驱动动画技术

3.2 关键帧技术类型

- 关键帧技术通过关键值的设置及其插值计算，对动画效果进行直接控制，是计算机动画系统中最基本和最常用的低层控制动画技术，它不仅是关键帧动画的核心，也是很多高层动画系统的算法基础。
- 不同的关键值决定动画关键帧的不同状态，根据关键帧插值对象不同，可分为：

3.2.1. 基于图像的关键帧技术

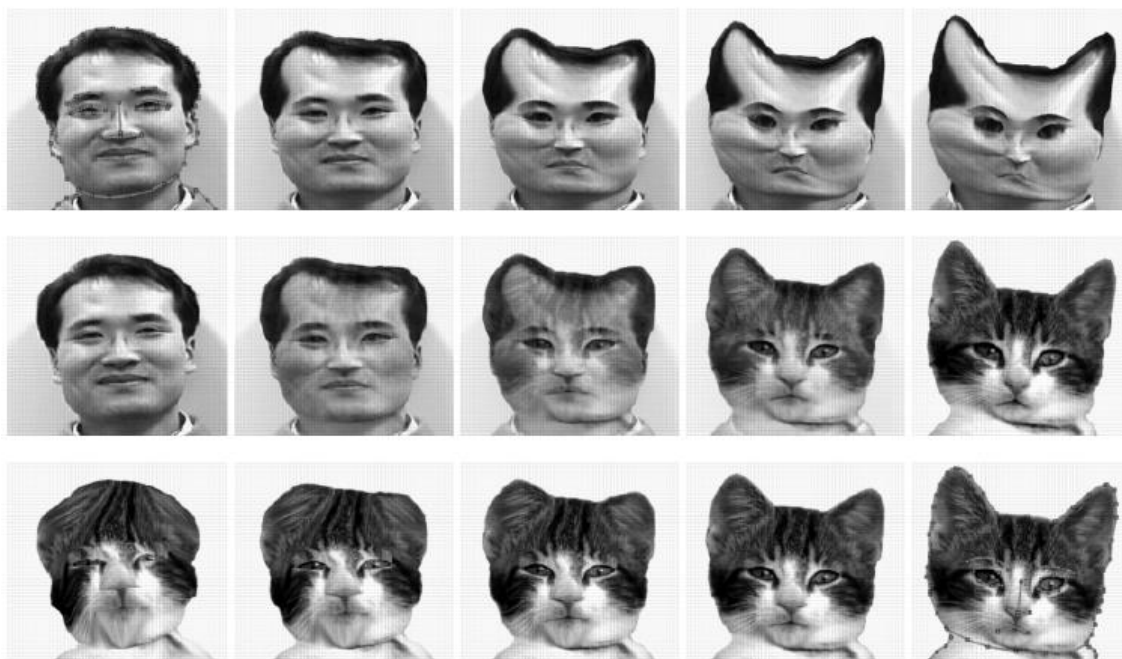
3.2.2. 基于图形的关键帧技术

3.2.3. 基于参数的关键帧技术

3.2 关键帧技术类型

3.2.1 基于图像的关键帧技术

- 基于**图像**的关键帧技术构成了图像渐变动画的算法基础。
- 图像渐变动画（Image Morphing）：把一幅数字图像以一种自然流畅的,戏剧性的,超现实主义的方式变换到另一幅数字图像。
- 由morphing生成的图像序列中,前面部分像源图像,后面部分像目标图像,中间部分既像源图像又像目标图像。



3.2 关键帧技术类型

3.2.1 基于图像的关键帧技术

- 基于图像的关键帧技术构成了图像渐变动画的算法基础。
- 图像渐变动画（Image Morphing）：把一幅数字图像以一种自然流畅的,戏剧性的,超现实主义的方式变换到另一幅数字图像。
- 由morphing生成的图像序列中,前面部分像源图像,后面部分像目标图像,中间部分既像源图像又像目标图像。



3.2 关键帧技术类型



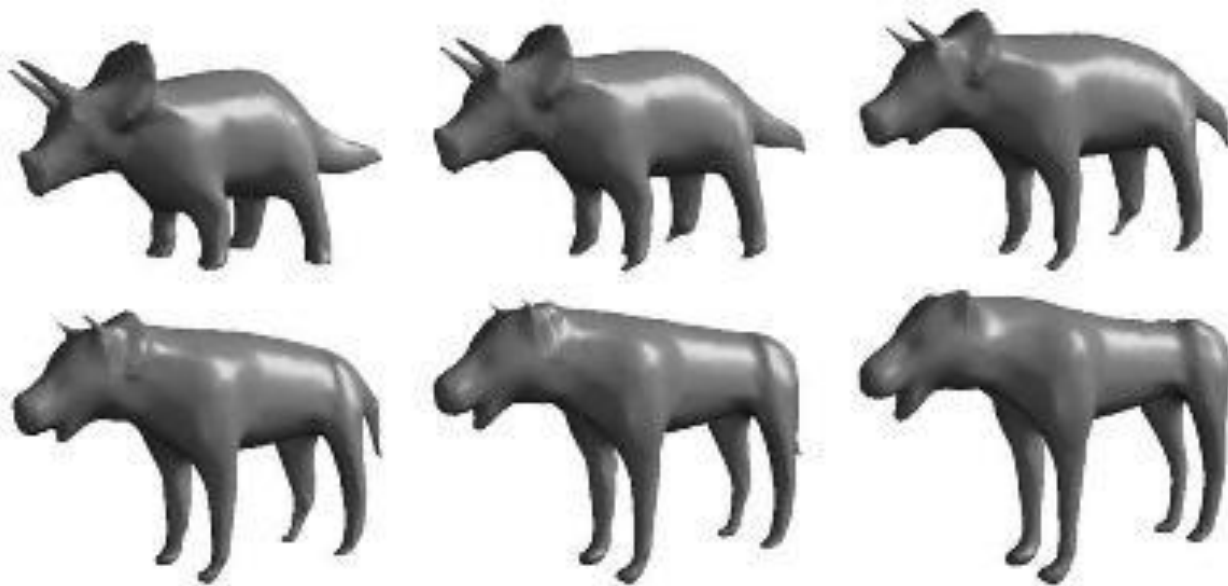
图 美国前总统小布什和现任总统奥巴马之间的相似转变过程



3.2 关键帧技术类型

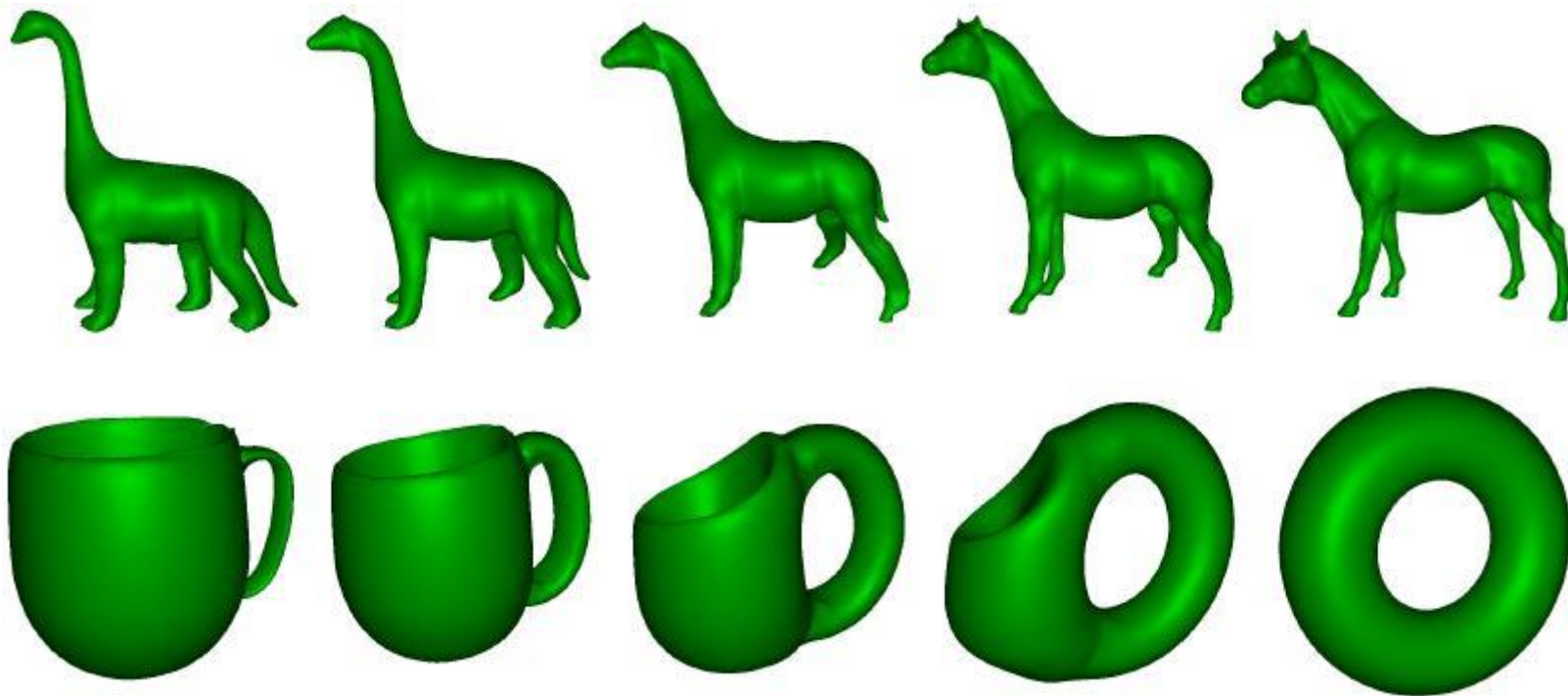
3.2.2 基于图形的关键帧技术

- 基于**图形**的关键帧技术对**几何图形**进行关键帧设置和插值处理，通过初始图形和目标图形对应几何元素（如顶点、边长和角度等）的插值来生成中间图形，形成从初始图形到目标图形连续变化的图形序列。
- 图形关键帧技术比图像关键帧技术要复杂得多,但能生成更加逼真更加生动的特技效果。



3.2 关键帧技术类型

3.2.2 基于图形的关键帧技术



- 基于图像和图形的关键帧技术是变形动画的算法基础，相应内容将在“变形动画”章节中展开。

3.2 关键帧技术类型

3.2.3 基于**参数**的关键帧技术

- **概述**

- 关键帧插值是中间插画制作的常用手段。对于给定的两帧关键帧或组成关键帧的两组图形，可以对其颜色、位置、旋转角度等**参数**进行各种插值。从而产生中间的画面图像或中间帧图形。

3.2 关键帧技术类型

3.2.3 基于参数的关键帧技术

• Key-Frame Systems

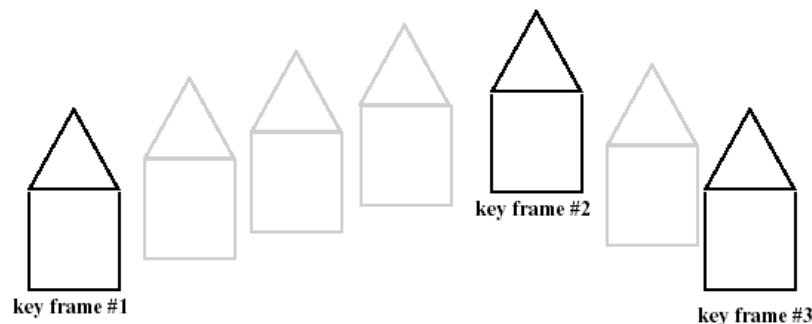
❖ 由某个参数在相邻关键帧的参数值确定中间帧的参数值

➤ 物体位置的坐标

➤ 关节角

➤ 材质属性

➤ 其他属性



❖ 特点：方法简单实用，但因不考虑物体的物理属性以及参数之间的相互关系，所以插值得到的运动**不一定**是合理的，需要动画师对运动进行仔细的调整

3.2 关键帧技术类型

3.2.3 基于参数的关键帧技术

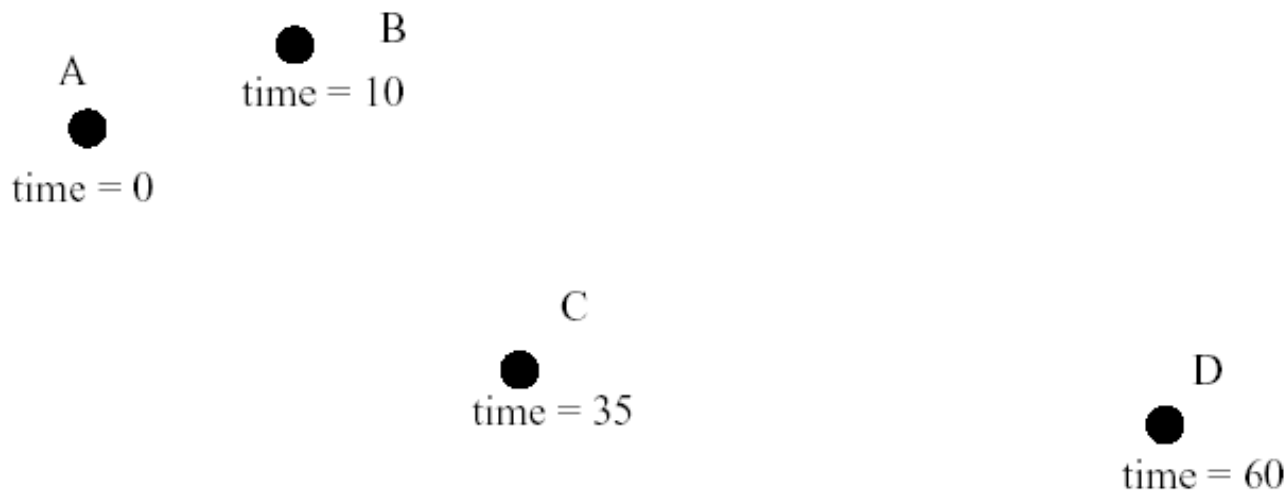
- 关键帧技术的过程（重要）

1. 确定需要控制的参数 P
2. 设置 n 个关键帧参数 (k_i, P_i) , $i=0, 1, 2, \dots, n-1$
3. 根据关键帧参数设计适当的插值函数
4. 对插值函数进行离散采样，计算中间帧的参数值

3.2 关键帧技术类型

3.2.3 基于参数的关键帧技术

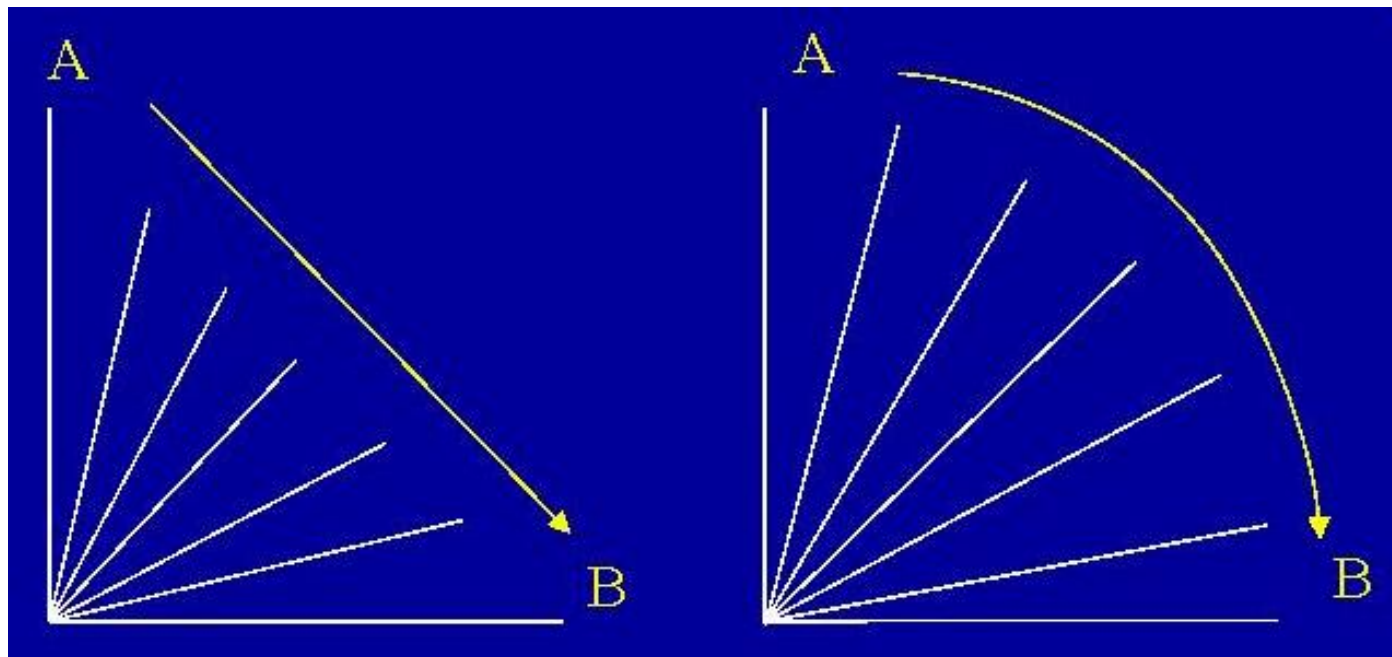
- 关键帧的设置



3.2 关键帧技术类型

3.2.3 基于参数的关键帧技术

- 参数选取



3.2 关键帧技术类型

3.2.3 基于参数的关键帧技术

- 如何最好地生成中间帧对应的参数值



- 确定适合的插值函数

- 由插值函数快速而准确地计算中间帧的参数值

第3章 关键帧动画

3.1 关键帧动画概述

3.2 关键帧技术类型

3.3 关键帧插值计算

3.4 插值函数与参数的选择

3.5 样条驱动动画技术

3.3 关键帧插值计算

- **关键帧技术**就是找到初始关键帧与目标关键帧的关键值之间的对应关系，并利用一定的**算法插值计算**出两个关键帧之间各个**过渡的参数值**，因此中间帧的插值计算对于关键帧动画的实验至关重要，**不同插值算法**生成的动画效果可能差别相当明显。

- 本节内容主要包括：

3.3.1 线性插值

3.3.2 非线性插值

3.3.3 方向表达与插值

3.3 关键帧插值计算

- 线性插值与非线性插值

按照插值函数的类型分，插值可分为线性插值和非线性插值。

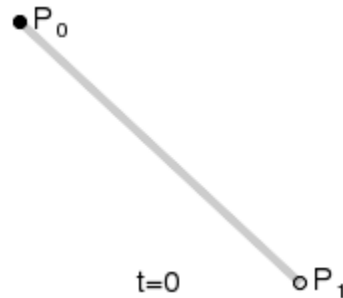
线性插值计算相对简单，计算结果也比较直观。对设定两个关键帧的情形，一般采用的线性插值算法计算中间帧。如果关键帧数目超过2，则要用到高次多项式的拟合方法。

- 方向表达与插值

空间对象有位置与方向属性，其中方向往往涉及到对象的旋转。位置与方向首选方案是4*4转换矩阵。

3.3 关键帧插值计算

3.3.1 线性插值



设：插值参数的起始值 P_0 ，终点值为 P_1 则中间值的计算可用以下**线性插值**公式：

$$P(t) = (1-t)P_0 + tP_1 \quad t \in [0, 1]$$

如果需要 n 个中间结果，则可以令

$$t = \frac{1}{n+1}, \frac{2}{n+1}, \dots, \frac{n}{n+1},$$

3.3.1 线性插值

(1) 基于颜色的线性插值

设颜色值采用“RGB”模式表示。如已知初始的颜色值为 (R_0, G_0, B_0) ，最终的颜色值为 (R_1, G_1, B_1) ，则中间结果线性插值的颜色值计算公式为

$$\begin{cases} R(t) = (1-t)R_0 + tR_1 \\ G(t) = (1-t)G_0 + tG_1 \\ B(t) = (1-t)B_0 + tB_1 \end{cases}$$

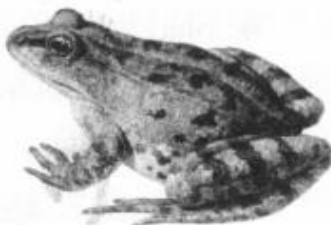
3.3.1 线性插值

例1：颜色线性插值示例 — 青蛙变白马王子

设 $n=4$, 可插出4幅不同的中间图像, 见下图。



(a) 原始图像



(b) $t=0.2$



(c) $t=0.4$



(d) $t=0.6$



(e) $t=0.8$



(f) 目标图像

3.3.1 线性插值

例2：坐标位置的线性插值

基于坐标位置的线性插值具有多种形式，如基于物体中心的坐标位置插值和基于物体顶点的坐标位置插值等。设物体在第1个关键帧中的位置为 P_0 ，在第2个关键帧中的位置为 P_1 ，则物本在中间帧中的位置的线性插值计算公式为：

$$P(t) = (1-t)P_0 + tP_1$$

$t \in [0, 1]$ ，如果需要物体在共 n 个中间帧中的位置，则将 $t=i/n+1$ ， $(i=1, 2, \dots, n)$ 分别代入上式。

3.3.1 线性插值

例2：物体坐标位置的线性插值示例 ——缆车的运动

设 $n=3$ ：



(a) 原始图像

(b) $t=0.25$

(c) $t=0.5$

(d) $t=0.75$

(e) 目标图像

3.3.1 线性插值

例3: 物体顶点坐标位置的线性插值

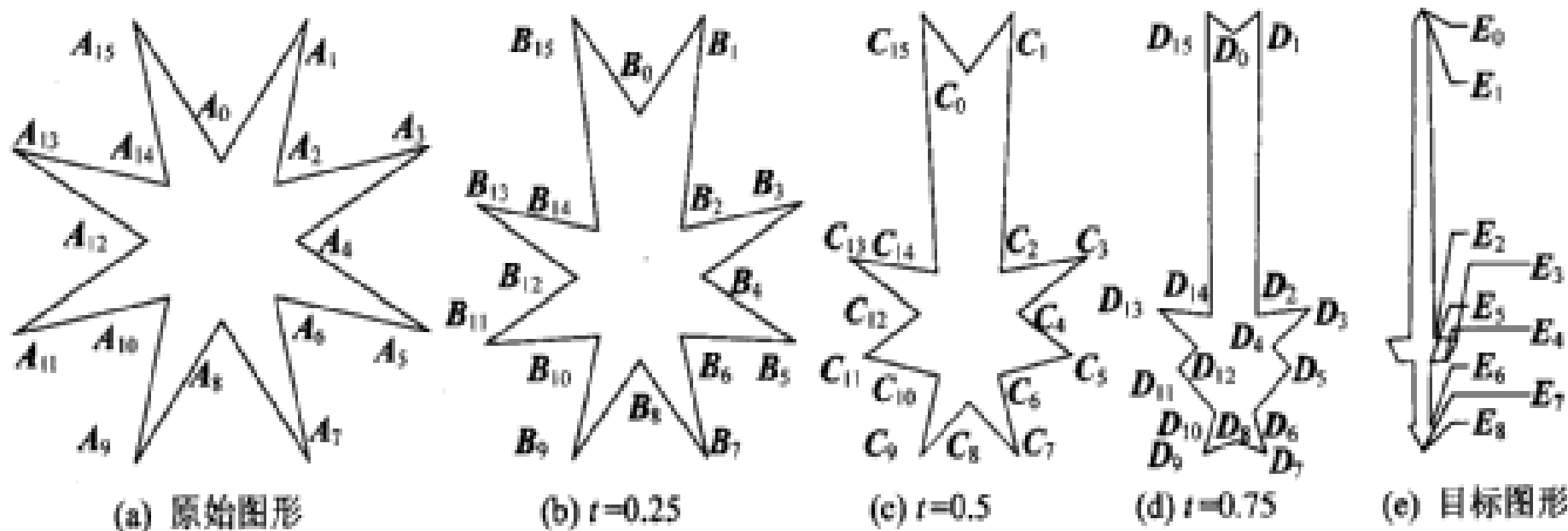
坐标位置的线性插值还可以分别作用到图形的各个顶点上。下图中，原始图形是一个星形，其顶点分别为 A_0, A_1, \dots, A_{15} ，目标图形是一把剑，其顶点分别为 E_0, E_1, \dots, E_{15} ，各中间帧中的顶点分别为 B_i, C_i, D_i ，顶点位置的线性插值计算公式为：

$$B_i = P(t) = (1-t)A_i + tE_i$$

$$t \in [0, 1], \quad (i=1, 2, \dots, 15)。$$

3.3.1 线性插值

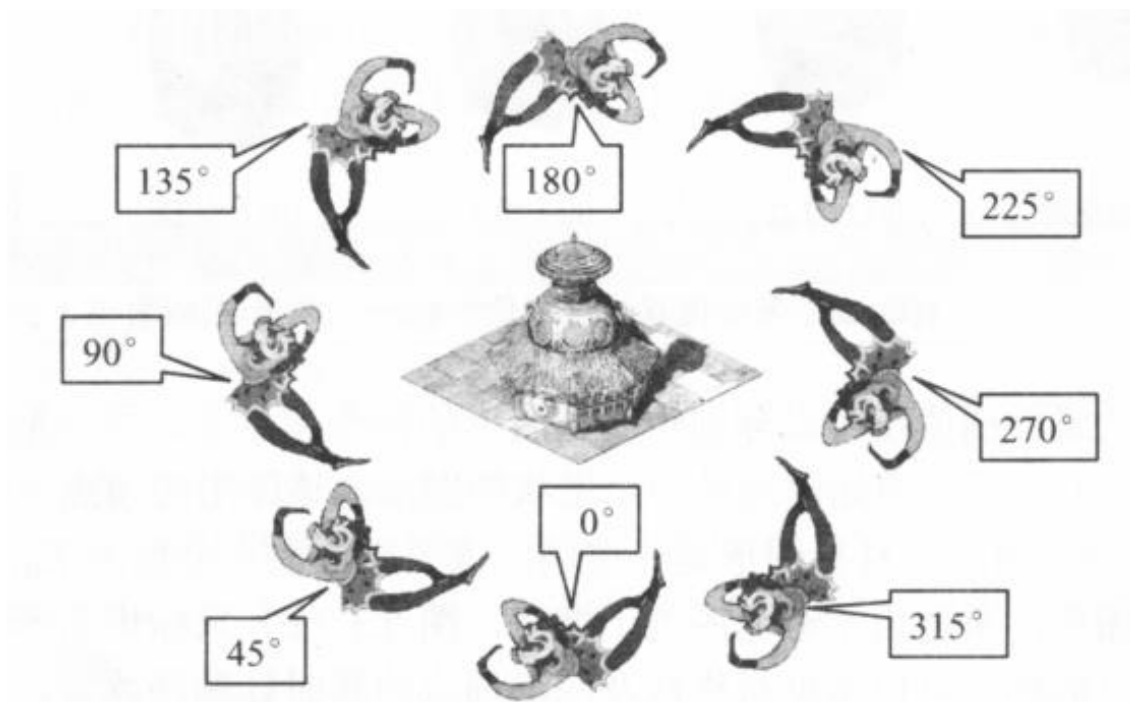
例4：物体顶点坐标线性插值示例 — 从星到剑



3.3.1 线性插值

例5：基于旋转角度的线性插值

令 $\theta_0 = 0^\circ$ ， $\theta_1 = 315^\circ$ ，可以得到孙悟空在第 i 个中间帧中的顺时针旋转角度 $\theta(t)$ 。



3.3 关键帧插值计算

3.3.2 非线性插值

为了产生更加自然流畅的动画效果，非线性插值常常是很有必要的，它是关键帧插值的重要手段之一。经常采用Bézier曲线、B样条曲线等来进行中间值的插值计算，下面以二次Bézier函数曲线为例介绍非线性关键帧插值。

3.3.2 非线性插值

例1：孙悟空抛桃和接桃
求桃子在中间帧的位置和方向



(a) 原始图像



(e) 目标图像

3.3.2 非线性插值

(1) 二次Bézier曲线进行非线性插值

设对象在起始关键帧的位置在 p_0 ，在目标关键帧的位置在 p_2 ，两端点的运动方向（切线方向） V_0 、 V_1 的交点为 p_1 ，则该对象在中间帧的位置可以采用以下二次Bézier函数计算

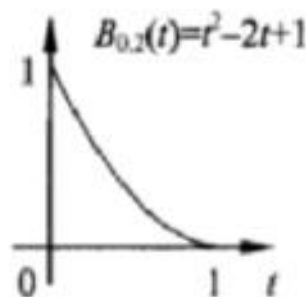
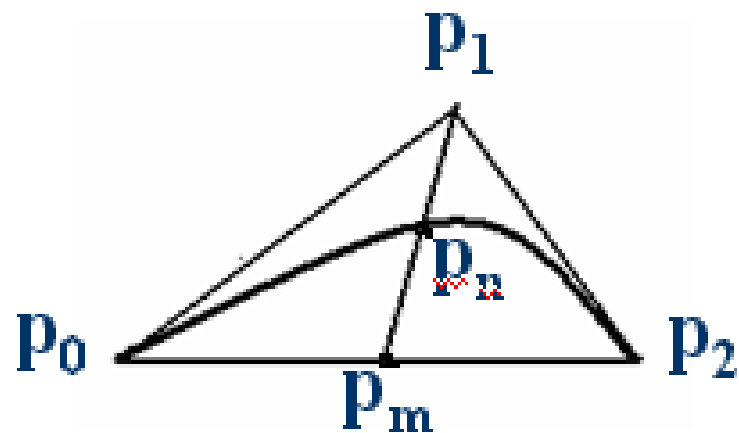


如需插入 n 个中间帧，则可以令

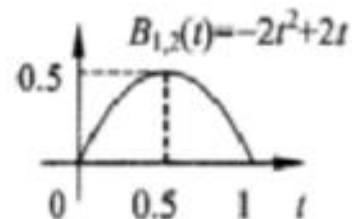
$$t = \frac{1}{n+1}, \frac{2}{n+1}, \dots, \frac{n}{n+1},$$

3.3.2 非线性插值

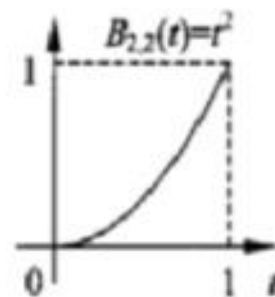
图示：二次Bézier函数插值所得为一条抛物线曲线



(c) $B_{0,2}(t)$



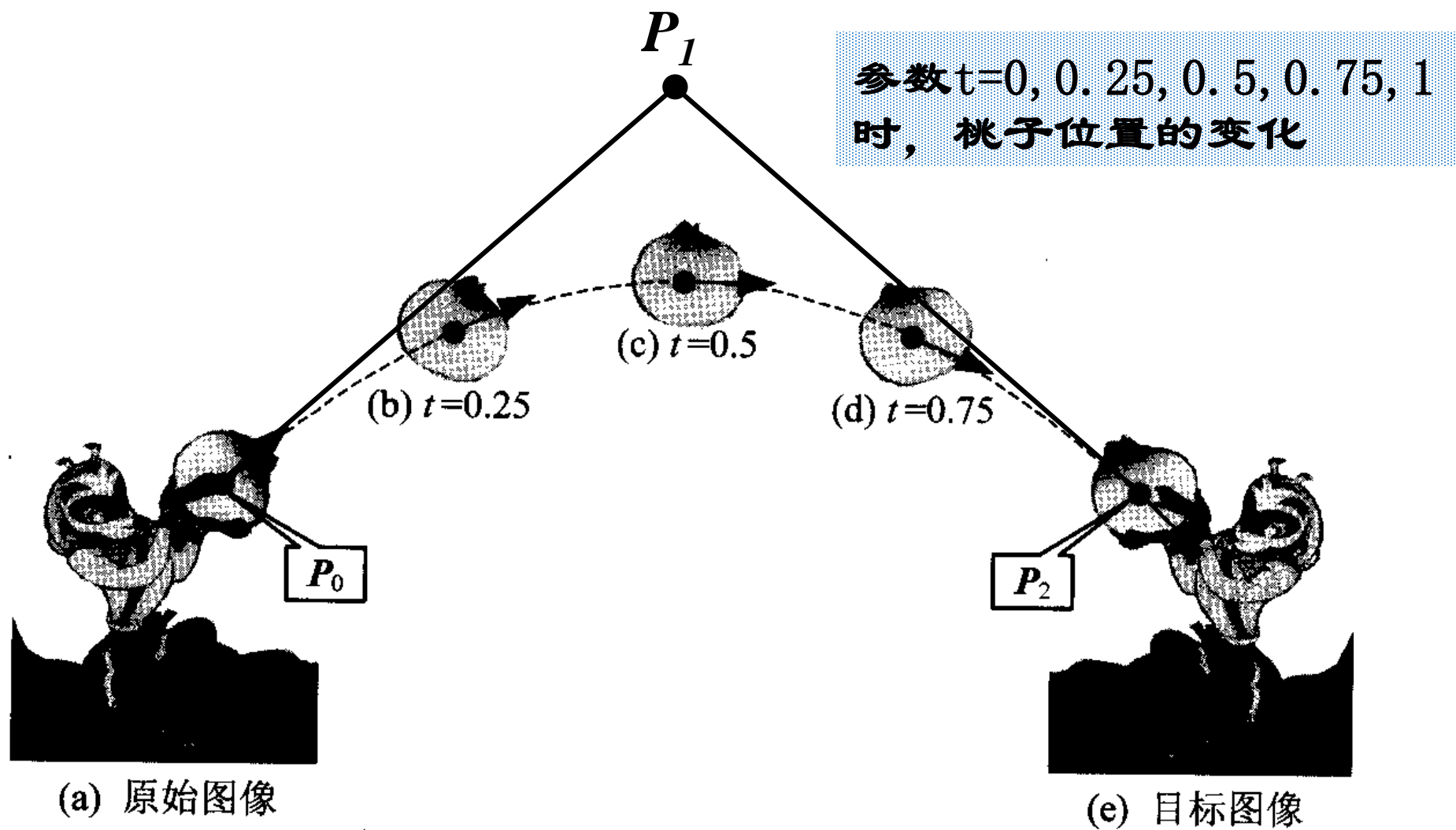
(d) $B_{1,2}(t)$



(e) $B_{2,2}(t)$

图 2 次 Bernstein 基函数图示

3.3.2 非线性插值



3.3.2 非线性插值

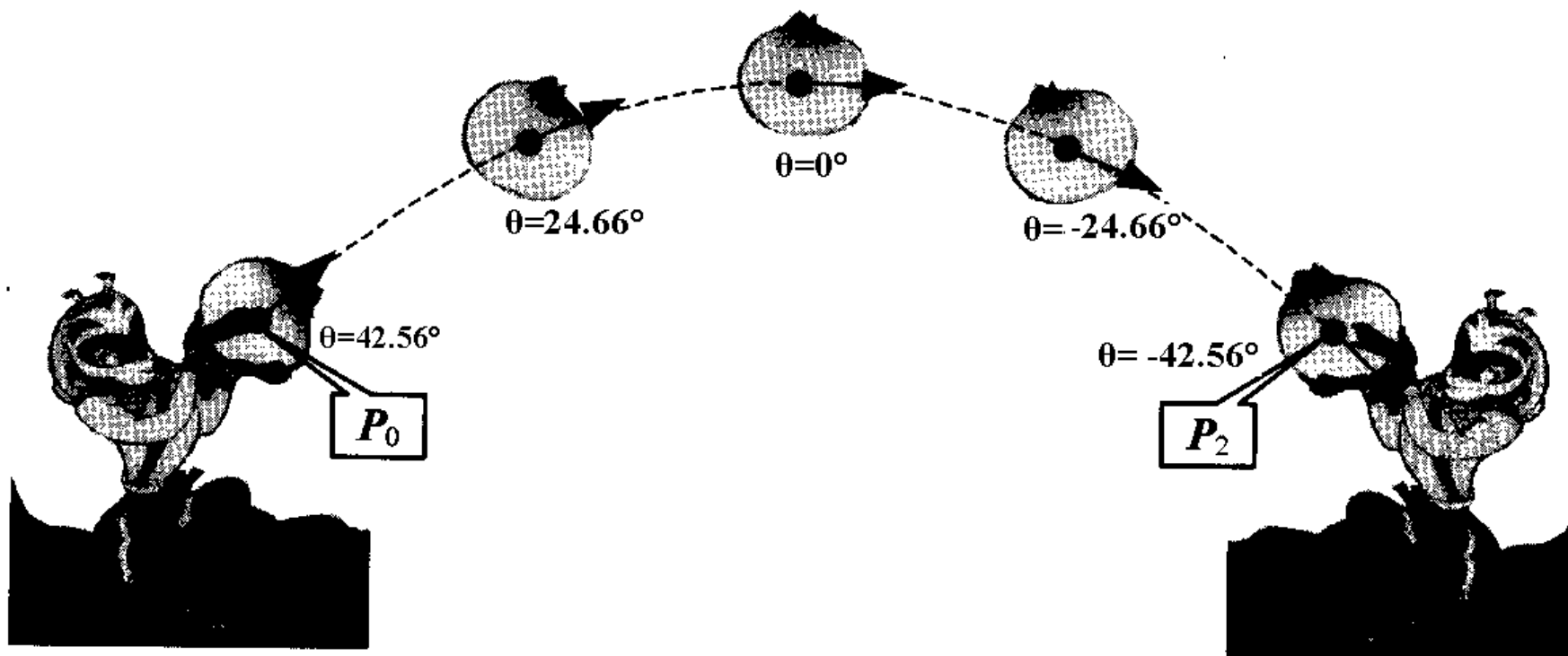
求桃子的旋转角度 θ

如果将从X轴正向到对象的运动方向之间的角度作为该对象的旋转角度 θ 。则通过计算二次Bézier函数的一阶导数，即得到对象的运动方向

$$P'(t)=(2t-2)P_0+(2-4t)P_1+2tP_2$$

3.3.2 非线性插值

参数 $t=0, 0.25, 0.5, 0.75, 1$ 时,
桃子运动方向的变化



3.3 关键帧插值计算

3.3.2 非线性插值--小结

采用基于二次Bézier曲线的关键帧插值可以**很好地**表示在空中只受到重力作用的物体的运动，这时物体的运动轨迹是**一条抛物线**，而二次Bézier曲线正好可以**精确**表示抛物线。如果采用三次Bézier曲线则可以插值更复杂的运动轨迹。

用作非线性插值的曲线还可以是圆弧、B样条和NURBS等曲线，这可以使插值具有更大的灵活性，对象参数的变化也可以更丰富。

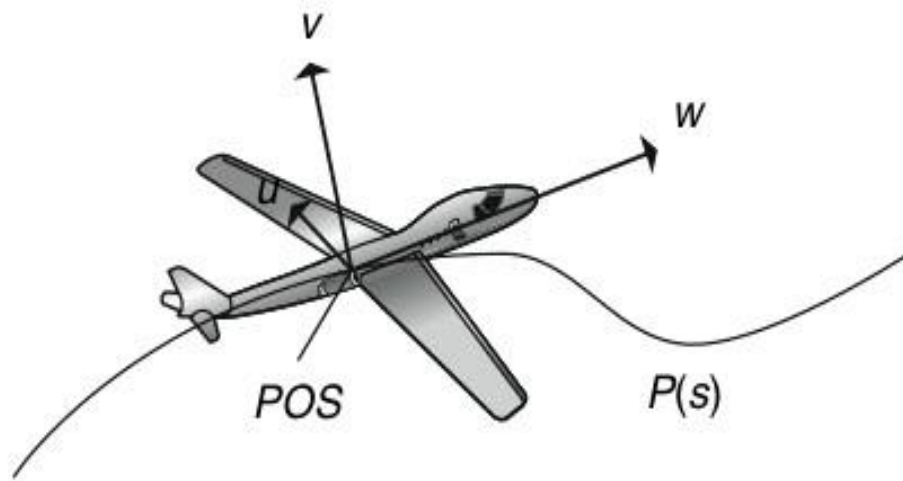
3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位

■ 关键帧插值问题

- 位置的插值；
- 给定一个坐标系，物体方位可以用旋转来表示；

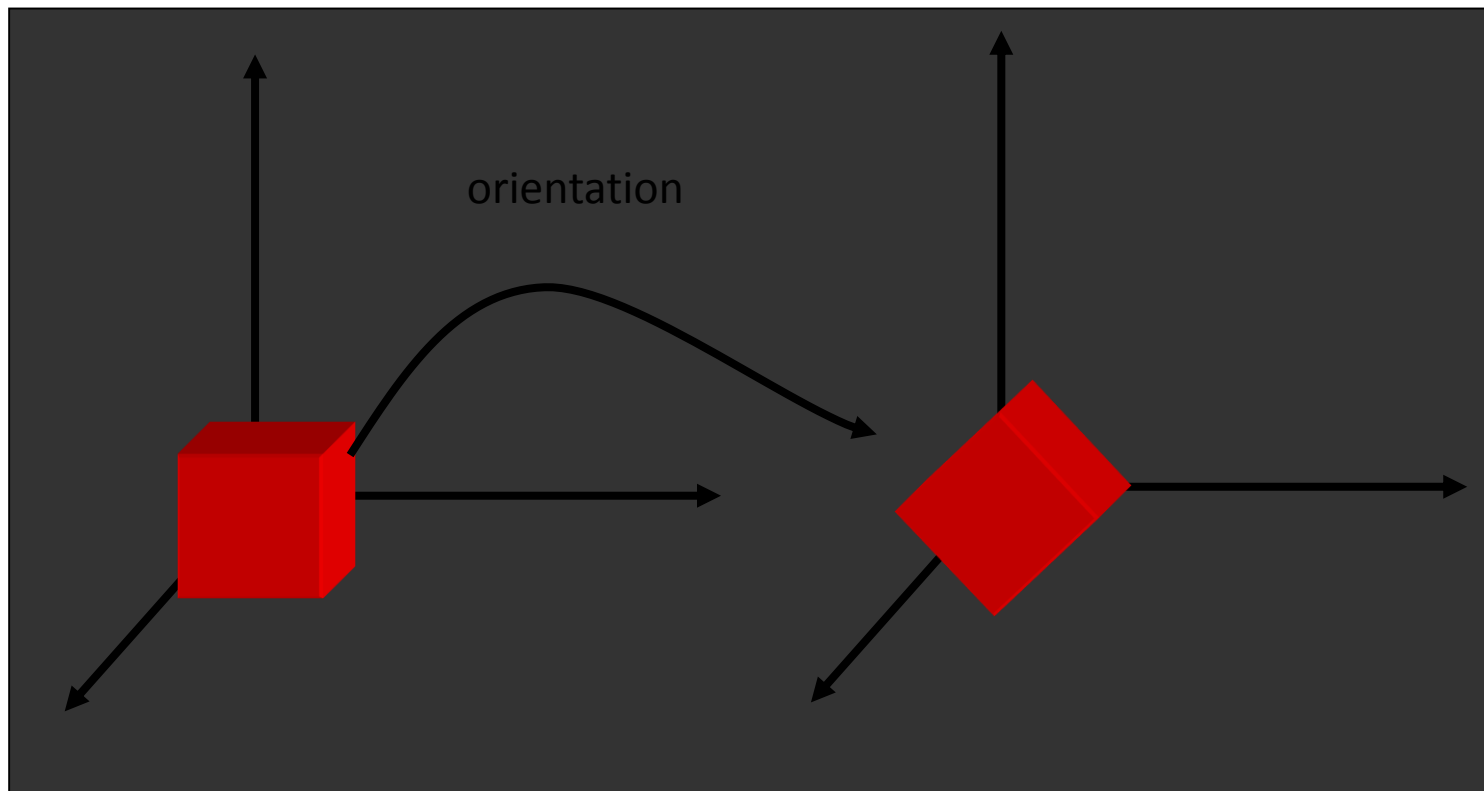


3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位

- Given a coordinate system, the orientation of an object can be represented as a rotation from a reference pose



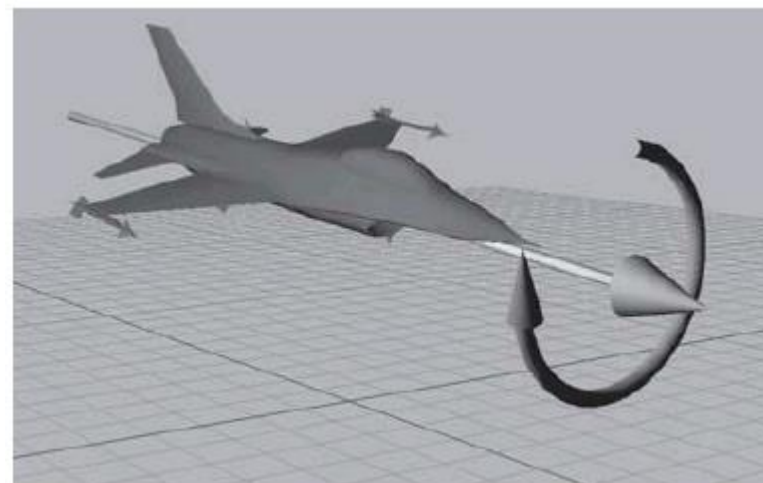
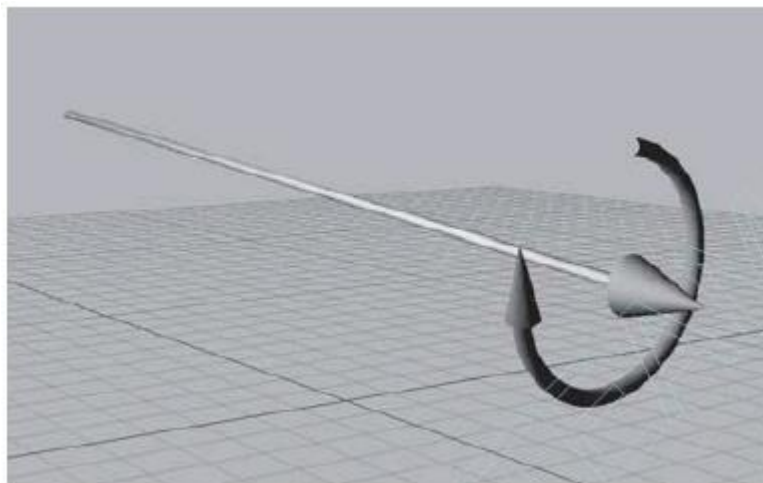
3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位

■ 方位和方向

- 方位主要描述的是物体的朝向
- 方向和方位并不完全一样
- 向量有方向但没有方位
- 当一个向量指向特定方向时，可以让向量自转，但向量（或者说它的方向）却不会有任何变化，因为向量的属性只有“大小”，而没有“厚度”和“宽度”（方向）
- 当一个物体朝向特定的方向时，让它和上面的向量一样自转，你会发现物体的方位改变了（方位）



3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位

■方位的描述

- 在3D中，只要用**两个**数字（例如：极坐标），就能用参数表示一个**方向**(direction)
- 但是，要确定一个**方位**(orientation)，却至少需要需要**三个**数字
- 描述物体**方位**时，也**不能使用绝对量**（即**相对量**）
- 与位置只是相对已知点的位移一样，方位是通过与相对已知方位(通常称为“单位”方位或“源”方位)的**旋转**来描述的
- 旋转的量称作**角位移**
- 在数学上描述**方位**就等价于描述**角位移**
- “**方位**”和“**角位移**”的区别就像“**点**”和“**向量**”的区别

3.3 关键帧插值计算

3.3.3 方向表达与插值

旋转不同于平移，旋转不直观

- Rotations are non-Euclidean (非欧拉的)
 - like travelling on a globe (球) vs. a grid (网格)
- Rotations are not commutative(非交换的)
 - x-rotate, y-rotate is not equal y-rotate, x-rotate *etc.* (旋转顺序有影响)
- Rotations are non-linear(非线性的)

3.3 关键帧插值计算

3.3.3 方向表达与插值

旋转参数化方法：

■ (1) 旋转的三维欧拉空间表示

- e.g. Euler angles (欧拉角), exponential map (指数映射)
- Pros (利)
 - three parameters for three DOFs (三个自由度)
- Cons (弊)
 - Singularities(奇异), potentially poor interpolation (潜在的差的插值)

3.3 关键帧插值计算

3.3.3 方向表达与插值

旋转参数化方法：

■ (2) 非欧拉空间表示

- *e.g.* unit quaternions (四元数) (S^3)
- Pros (利)
 - singularity free (非奇异)
- Cons (弊)
 - must take extra measures (额外措施) to stay in legal sub-space
 - four parameters (四个参数) required for three DOFs (三个自由度)

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

❖ 对3D旋转可以有多种参数化的方法，但没有一种参数化的方法适合所有的应用。

❖ 需要找到最自然最简洁的方式表示旋转和方向：

- (1) 固定角
- (2) 矩阵表示
- (3) 欧拉角
- (4) 四元数
- (5) 指数映射 (Exponential Map)

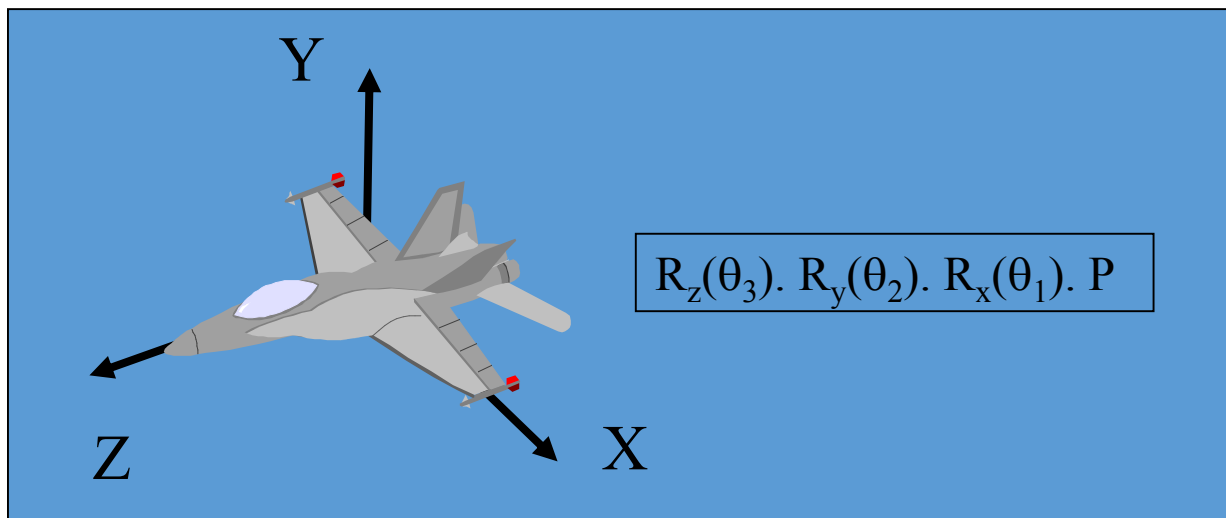
3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ (1) 固定角

- 沿世界坐标系三个轴的转角；
- 沿固定轴的转角，固定的顺序；
- $(\theta_1, \theta_2, \theta_3) = (10, 45, 90)$
 $\rightarrow R_x(10) \rightarrow R_y(45) \rightarrow R_z(90)$

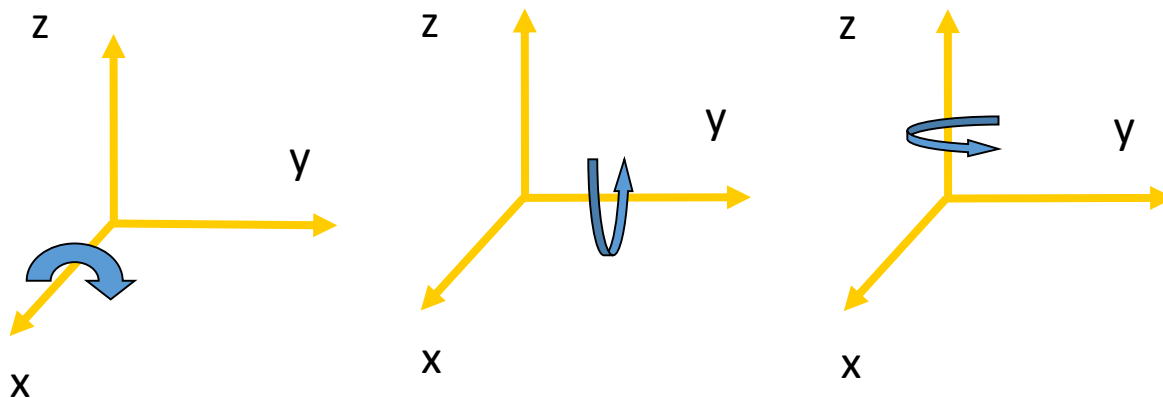


3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 固定角



物体的方向(必须)依次通过绕三个正交轴旋转的角度表示 $\rightarrow (\theta_1, \theta_2, \theta_3)$

$$V' = R * V$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 固定角

$$x\text{-roll}(\theta_1) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta_1 & \sin \theta_1 & 0 \\ 0 & -\sin \theta_1 & \cos \theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$y\text{-roll}(\theta_2) = \begin{pmatrix} \cos \theta_2 & 0 & -\sin \theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ \sin \theta_2 & 0 & \cos \theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$z\text{-roll}(\theta_3) = \begin{pmatrix} \cos \theta_3 & \sin \theta_3 & 0 & 0 \\ -\sin \theta_3 & \cos \theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\sin \theta_i = s_i, \quad \cos \theta_i = c_i$$

$$R(\theta_1, \theta_2, \theta_3) = \begin{pmatrix} c_2 c_3 & c_2 s_3 & -s_2 & 0 \\ s_1 s_2 c_3 - c_1 s_3 & s_1 s_2 s_3 + c_1 c_3 & s_1 c_2 & 0 \\ c_1 s_2 c_3 + s_1 s_3 & c_1 s_2 s_3 - s_1 c_3 & c_1 c_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

!!! 矩阵乘法不
满足交换率

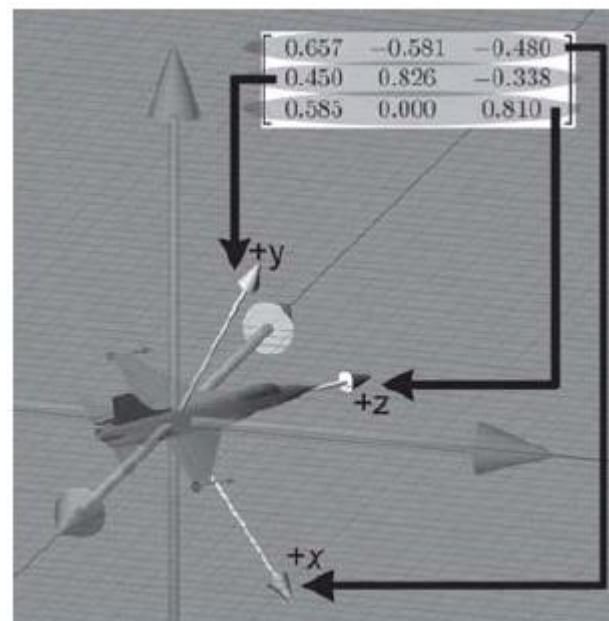
3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ (2) 矩阵形式

- 描述方位的一种方法—列出这个坐标系的**基向量**，这些基向量是用其他的坐标系来描述的；
- 用基向量构成一个**矩阵**，然后就能用矩阵形式来描述方位；
- 能用一个**旋转矩阵**来描述这**两个坐标系**之间的**相对方位**，这个旋转矩阵用于把一个坐标系中的向量转换到另外一个坐标系中；



3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

• 矩阵形式的优点

- 一种非常**直接**的描述方位的形式；
- 可以**立即**进行向量的旋转，为了旋转向量，必须将方位转换成矩阵形式；
- 矩阵形式被**图形API**所采用，如果选择了其他形式，则必须在渲染管道的某处将其转换为矩阵；
- 多个角位移**连接**，例如，如果知道A关于B的方位，又知道B关于C的方位，使用矩阵可以求得A关于C的方位；
- 矩阵的逆，用矩阵形式表达角位移时，**逆矩阵就是“反”角位移**；

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

- 矩阵形式的缺点

- 矩阵占用了更多的内存。
- 难以使用。矩阵对人类来说并不直观，有太多的数，并且它们都在-1到1之间，矩阵不是人类思考方位的直观方法。
- 矩阵可能是病态的，矩阵使用9个数，其实只有3个数是必须的，矩阵带有6阶冗余。描述方位的矩阵必须满足6个限制条件。行必须是单位向量，而且它们互相垂直。

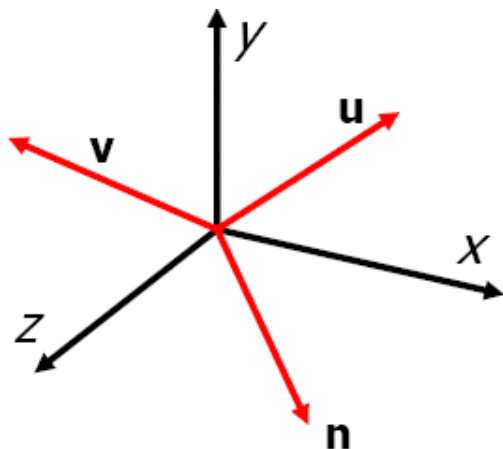
3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

- 矩阵形式插值

- 对矩阵中每个元素独立插值：
 - 例如：标准矩阵与绕x轴90度旋转矩阵的插值结果
- 结果并不是旋转矩阵（不互相垂直）；



$$M = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{pmatrix}$$

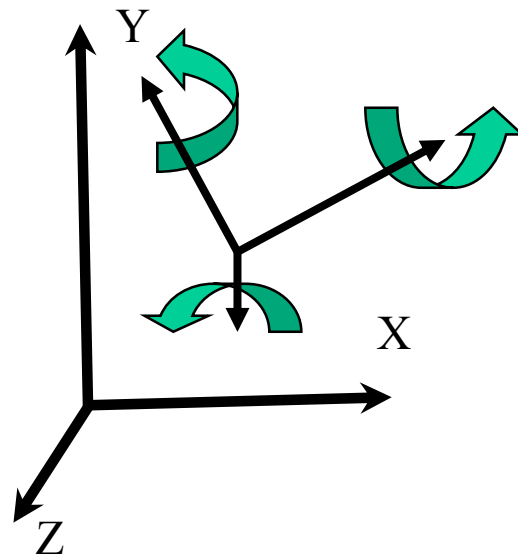
3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ (3) 欧拉角

- 以瑞士数学家[Leonhard Euler](#) (1707-1783) 的名字命名;
- 将角位移分解为绕三个互相垂直轴的三个旋转组成的序列;

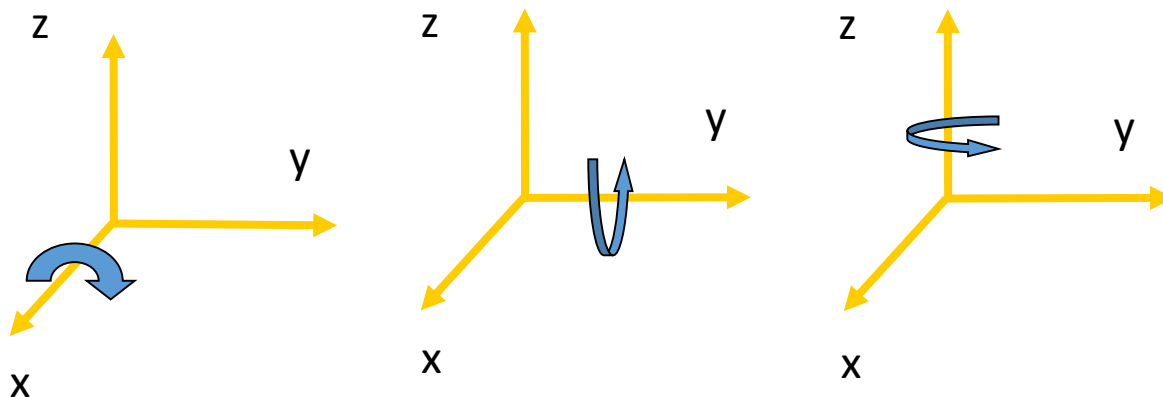


3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 欧拉角



物体的方向通过依次绕三个正交轴旋转的角度表示 $\rightarrow (\theta_1, \theta_2, \theta_3)$ (顺序是任意的)

$$V' = R * V$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 欧拉角（以 x-y-z 顺序为例）

$$x\text{-roll}(\theta_1) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta_1 & \sin \theta_1 & 0 \\ 0 & -\sin \theta_1 & \cos \theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$y\text{-roll}(\theta_2) = \begin{pmatrix} \cos \theta_2 & 0 & -\sin \theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ \sin \theta_2 & 0 & \cos \theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$z\text{-roll}(\theta_3) = \begin{pmatrix} \cos \theta_3 & \sin \theta_3 & 0 & 0 \\ -\sin \theta_3 & \cos \theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

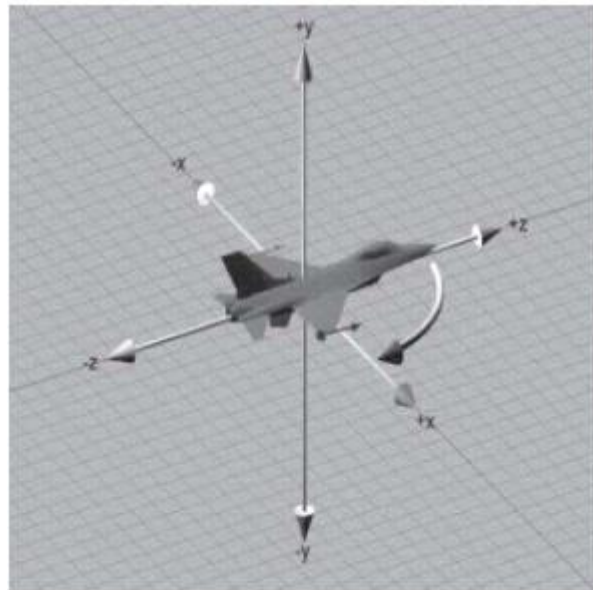
$$\sin \theta_i = s_i, \quad \cos \theta_i = c_i$$

$$R(\theta_1, \theta_2, \theta_3) = \begin{pmatrix} c_2 c_3 & c_2 s_3 & -s_2 & 0 \\ s_1 s_2 c_3 - c_1 s_3 & s_1 s_2 s_3 + c_1 c_3 & s_1 c_2 & 0 \\ c_1 s_2 c_3 + s_1 s_3 & c_1 s_2 s_3 - s_1 c_3 & c_1 c_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

!!! 不同顺序的
旋转矩阵不同

3.3 关键帧插值计算

3.3.3 方向表达与插值



物体方位的参数化

■ 欧拉角

- 欧拉角将方位分解为绕三个互相垂直轴的旋转。
- 那么是哪三个轴？按什么顺序？
 - 任意三个轴和任意顺序都可以，最有意义的是使用笛卡尔坐标系并按一定顺序所组成的旋转序列。
- 最常用的约定是所谓的“heading - pitch - bank”约定，一个方位被定义为一个heading角，一个pitch角，一个bank角。
- 基本思想是让物体开始于“标准”方位 —— 就是物体坐标轴和惯性坐标轴对齐。
- 在标准方位上，让物体作heading、pitch、bank旋转，最后物体到达我们想要描述的方位。
- 此时物体坐标系和惯性坐标系重合，heading为绕y轴的旋转量，向右旋转为正（如果从上面看，旋转正方向就是顺时针方向）。

3.3 关键帧插值计算

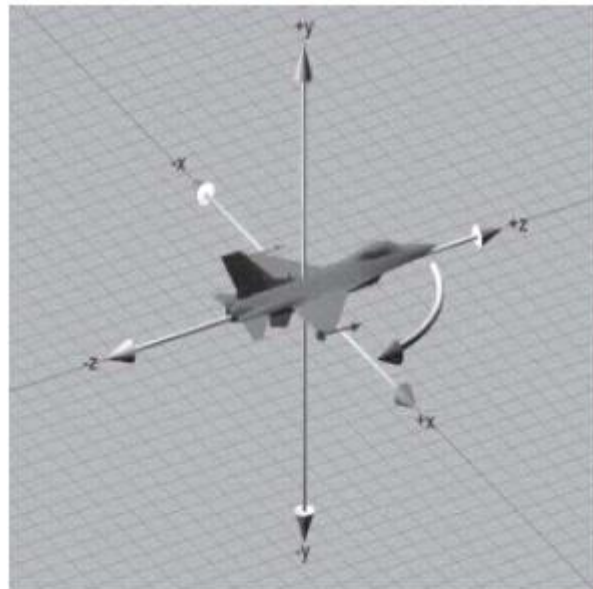
3.3.3 方向表达与插值

物体方位的参数化

■ 欧拉角

- heading为绕y轴的旋转量，向右旋转为正。
- 经过heading旋转后，pitch为绕x轴的旋转量，向下旋转为正。
- 经过了heading和pitch，bank为绕z轴的旋转量。

依据左手法则，从原点向+z看，逆时针方向为正。



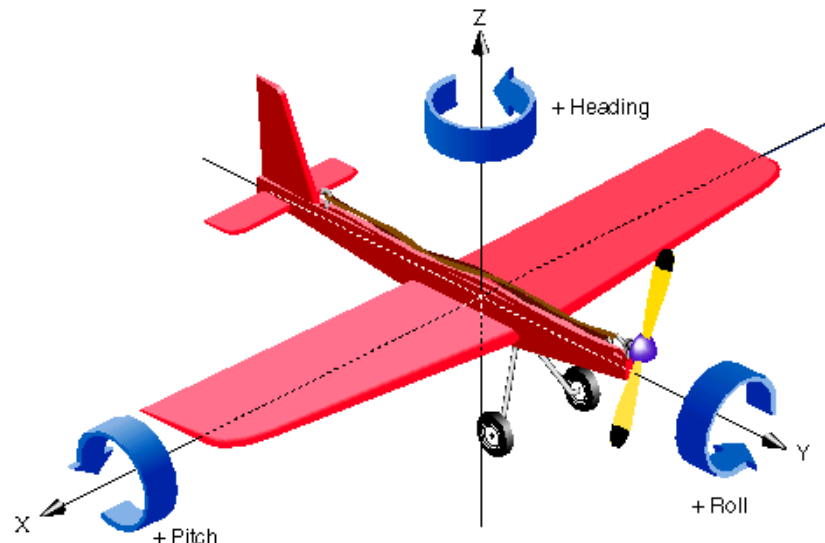
3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 欧拉角说明

- heading-pitch-bank系统**不是唯一**的欧拉角系统，欧拉角约定是多样的。
- **任意**三个轴都能作为旋转轴，不一定必须是笛卡尔轴，但使用笛卡尔轴最有意义。
- 决定每个旋转的**正方向**时不一定必须遵守左手或右手法则。
- 旋转可以以**不同的顺序**进行。



3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 欧拉角的各种形式

XYZ	XZY	XYX	XZX
YXZ	YZX	YXY	YZY
ZXY	ZYX	ZXZ	ZYZ

!!! XXZ YZZ 没有意义

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 局部坐标系与全局坐标系

`glRotatef(angleX, 1, 0, 0)` `//M1`

`glRotatef(angleY, 0, 1, 0)` `//M2`

`glRotatef(angleZ, 0, 0, 1)` `//M3`

`// translate`

`//Some Object` `// V`

$$V' = M3 * M2 * M1 * V$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 局部坐标系与全局坐标系---reverse order

局部坐标系下的欧拉角 (alpha, beta, gamma)

$$R_x(\alpha) R_y(\beta) R_z(\gamma) = R_z(\gamma) R_y(\beta) R_x(\alpha)$$

全局坐标系下的欧拉角 (gamma, beta, alpha)

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 局部坐标系与全局坐标系---reverse order

- 欧拉系统等价于前述的固定角系统，且相关操作顺序以逆序加以呈现，该结论对于任意欧拉系统均成立。
 - 例如，z-y-x欧拉系统等价于x-y-z固定角系统。
- 因此，欧拉系统与固定角系统具有相同的优缺点。

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 欧拉角优点

- 欧拉角**仅用三个数**来表达方位，并且这三个数都是**角度**；
- 欧拉角很容易使用，因为欧拉角中的数都是角度，符合人们**思考方位**的方式，能直接描述出最重要的角度；
- 最**简洁**的表达方式，欧拉角用三个数来表达方位。在3D中，表达方位不能少于三个数，如果要考虑内存的因素，欧拉角是最合适的描述方位的方法；
- **任意**三个数都是**合法**的，取任意三个数，它们都能构成合法的欧拉角，而且可以把它看成一个对方位的描述；

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 欧拉角缺点

- 给定方位的表达方式**不唯一**；
- **别名问题**—对于一个给定方位，存在多个欧拉角可以描述它；
 - 将heading和bank限制在+180度到-180度之间，pitch 限制在+90度到-90度之间。
- **万向锁（万向节死锁）现象**：
 - 先heading45度再pitch90度 = 先pitch90度再bank45度是等价的。
 - 角度为+(-)90度的**第二次**旋转使得**第一次**和**第三次**旋转的旋转轴相同，称作**万向锁**。
 - 规定万向锁情况下由heading完成绕竖直轴的全部旋转。在限制欧拉角中，如果pitch为+(-)90度，则bank为0。

3.3 关键帧插值计算

3.3.3 方向表达与插值

Gimbal Lock-数学推导及其示意图

$$R(\theta_1, \pi/2, \theta_3) = \begin{bmatrix} 0 & 0 & -1 & 0 \\ \sin(\theta_1 - \theta_3) & \cos(\theta_1 - \theta_3) & 0 & 0 \\ \cos(\theta_1 - \theta_3) & -\sin(\theta_1 - \theta_3) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

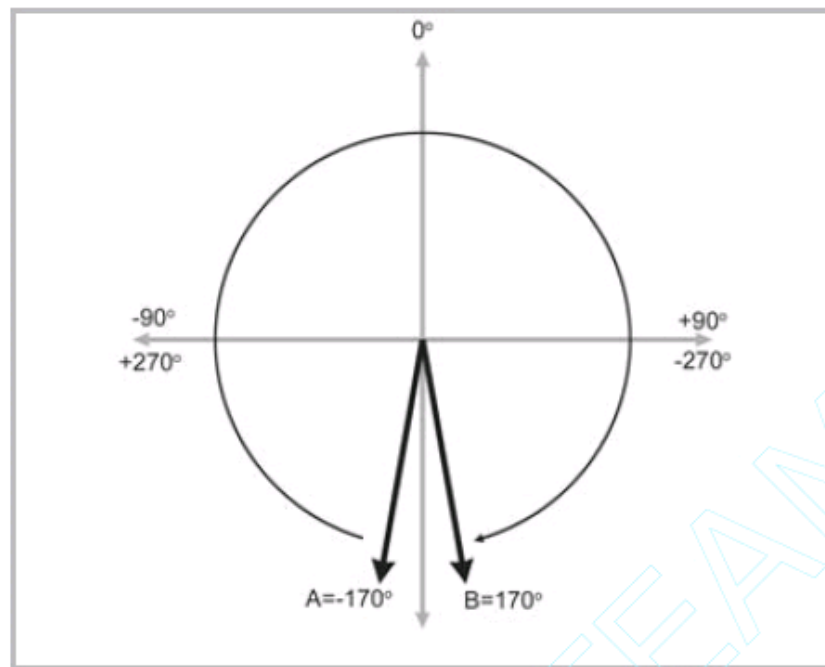
3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 欧拉角缺点

- 两个角度间求插值非常困难；
- 1. 如果没有使用限制欧拉角，将得到很大的角度差；
- 2. 旋转角度的周期性引起的。



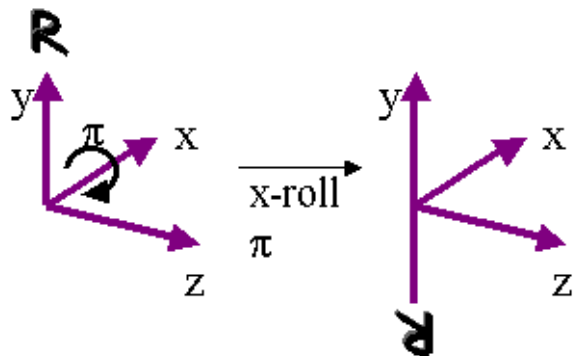
欧拉角插值仍然可能碰到万向锁的问题，它在大多数情况下会产生抖动、路径错误等现象，物体会突然飘起来像是"挂"在某个地方。根本问题是插值过程中角速度不是恒定的

3.3 关键帧插值计算

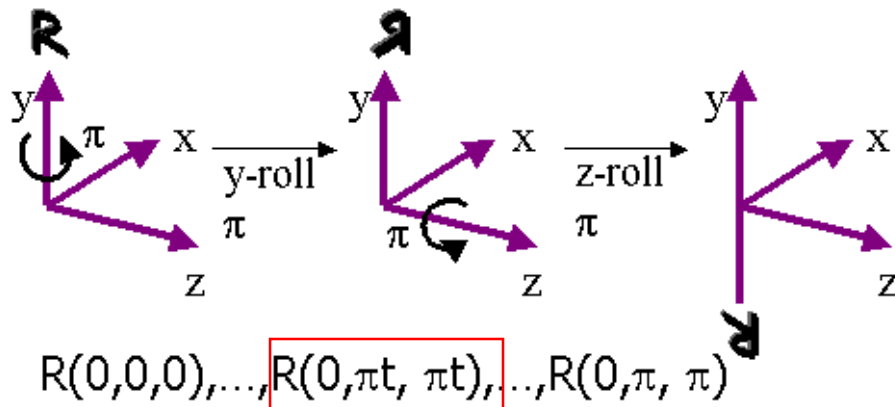
3.3.3 方向表达与插值

物体方位的参数化

■ 欧拉角的插值



$$R(0,0,0), \dots, R(\pi t, 0, 0), \dots, R(\pi, 0, 0) \\ t \in [0, 1]$$



$$R(0,0,0), \dots, R(0, \pi t, \pi t), \dots, R(0, \pi, \pi)$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 欧拉角的插值

用欧拉角对**两个方向**进行插值，得到的结果是**不唯一**的。这是因为欧拉角的三个自由度**不是相互独立的**，对三个欧拉角单独插值完全忽略了它们之间的相互关系。

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ (4) 四元数表示及其插值技术

➤ 1830年，英国数学家William Hamilton爵士研究复数，将其扩展复数到更高的维次：

➤ $i^2 = j^2 = k^2 = i j k = -1$

➤ 1843年，哈密顿立刻将此方程刻在都柏林Broome桥

➤ 80年代以后开始广泛应用

➤ 计算机动画、机器人路径规划、三维分形显示

➤ 四元数可以方便地表示矢量和物体的旋转

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 四元数

- 传统求解: 用单位四元数表示旋转(**rotations**)
 - S^3 has same **topology** as rotation space (a sphere), so no singularities
 - A member of **unit sphere** in R^4

$$q = (q_w, q_x, q_y, q_z)$$

- 对于 v 轴的旋转 $q = (\cos \frac{\theta}{2}, v \cdot \sin \frac{\theta}{2})$

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 四元数定义

Q是实数域R上的四维矢量空间

$$(1, 0, 0, 0) \rightarrow e \quad (0, 1, 0, 0) \rightarrow i$$

$$(0, 0, 1, 0) \rightarrow j \quad (0, 0, 0, 1) \rightarrow k$$

$$q = [S, V] = a_0 * e + a_1 * i + a_2 * j + a_3 * k$$

$$S = a_0$$



标量部分

$$V = (a_1, a_2, a_3)$$



矢量部分

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 四元数旋转

- 绕矢量 v 旋转

- 空间中一点 v （即原点到该顶点的矢量）表示为 $[0, v]$
- 旋转四元数 q 表示为：

$$q = Rot_{\theta, (x, y, z)} = [\cos(\theta/2), \sin(\theta/2) \bullet (x, y, z)]$$
$$v' = Rot_q(v) = q \bullet v \bullet q^{-1}$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 四元数旋转

- 旋转链接— 先按 q_1 旋转，然后按 q_2 旋转等价于按 $q_2 * q_1$ 旋转，即

$$\begin{aligned} q_2 * (q_1 * P * q_1^{-1}) * q_2^{-1} &= (q_2 * q_1) * P * (q_1^{-1} * q_2^{-1}) \\ &= (q_2 * q_1) * P * (q_2 * q_1)^{-1} \end{aligned}$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 四元数加法和数乘

加法：

$$q_1 + q_2 = [S_1, V_1] + [S_2, V_2] = [S_1 + S_2, V_1 + V_2]$$

数乘：

$$kq = k[S, V] = [kS, kV]$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 四元数的乘法

*	e	i	j	k
e	e	i	j	k
i	i	-e	k	-j
j	j	-k	-e	i
k	k	j	-i	-e

e是乘法的单位元，i, j, k按照 $i \rightarrow j \rightarrow k \rightarrow i$ 的顺序，相邻两单位元按照箭头顺序方向相乘等于第三单元，与箭头相反方向相乘则等于第三个单位元的负元

3.3 关键帧插值计算

3.3.3 方向表达与插值

物体方位的参数化

■ 四元数乘法

$$q1 = (w1, x1, y1, z1);$$

$$q2 = (w2, x2, y2, z2);$$

$$q1 * q2 = (w1.w2 - v1.v2, \quad w1.v2 + w2.v1 + v1 \times v2)$$

$$\text{其中, } v1 = (x1, y1, z1) \quad v2 = (x2, y2, z2)$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 四元数运算的属性

□ 四元数乘法满足结合律

$$(q_1 * q_2) * q_3 = q_1 * (q_2 * q_3)$$

□ 四元数乘法对加法满足分配律

$$(q_1 + q_2) * q = q_1 * q + q_2 * q$$

$$q * (q_1 + q_2) = q * q_1 + q * q_2$$

□ 四元数乘法不满足交换律

$$q_1 * q_2 \neq q_2 * q_1$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 四元数的属性

- 四元数的共轭和范数与复数相似

$$\bar{q} = (s, -v)$$

$$\|q\| = \sqrt{q^* \bar{q}} = \sqrt{s^2 + v_x^2 + v_y^2 + v_z^2}$$

- 四元数的逆

$$q^{-1} = \frac{\bar{q}}{q^* \bar{q}}$$

- 单位四元数

$$\|q\| = 1 \Rightarrow q^{-1} = \bar{q}$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 四元数的其他性质

❖ 四元数的乘积是保范的

$$||q_1 * q_2|| = ||q_1|| * ||q_2||$$

❖ 四元数的共轭满足

$$(q_1 q_2)^* = q_2^* q_1^*$$

$$(q_1 + q_2)^* = q_1^* + q_2^*$$

❖ 四元数的逆满足

$$(q_1 q_2)^{-1} = q_2^{-1} q_1^{-1}$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 四元数的其他性质

若把单位四元数表示成 $q=[\cos(\theta/2), n\sin(\theta/2)]$ 的形式，则该四元数可以表示绕单位轴 n 进行 θ 角的旋转

对于单位四元数 $q=[\cos\theta, n\sin\theta]$ ，其中 n 为单位旋转轴，有：

$$q=\exp(n\theta) \quad q^u=(\cos u\theta, n\sin u\theta)$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 四元数的其他性质

对于单位四元数 $q = [\cos\theta, n\sin\theta]$, 有:

$$\ln(q) = (0, n\theta) ;$$

对于四元数 $q = (0, n\theta)$, $\exp(q) = [\cos\theta, n\sin\theta]$

设 $q_1q_2\dots, q_n$ 表示一系列连续作用的旋转。
当 q_i 在全局坐标系中度量时, $q_nq_{n-1}\dots q_1$ 表示净旋转；
当 q_i 在局部坐标系 q_{i-1} 中度量时, $q_1q_2\dots q_n$ 表示净旋转

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 四元数、欧拉角和旋转矩阵之间的相互转换

尽管四元数能有效地表示旋转，但它并不易于交互给出。从用户角度看，用欧拉角定义物体的方向比较直观和方便，因此**需要把欧拉角转换为四元数**。

四元数所表示的旋转作用于物体时，需要四元数对应的齐次旋转矩阵，因此**需要把四元数转化为旋转矩阵**。

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 欧拉角向四元数的转换

设绕x, y, z轴旋转的欧拉角分别为 θ_1 , θ_2 , θ_3

对应的四元数分别为 q_{θ_1} , q_{θ_2} , q_{θ_3}

由公式 $R_q(p) = qpq^{-1}$ 可知, 对应的合成四元数为

$$q = q_{\theta_3}q_{\theta_2}q_{\theta_1}$$

$$= [\cos \frac{\theta_3}{2}, (0, 0, \sin \frac{\theta_3}{2})][\cos \frac{\theta_2}{2}, (0, \sin \frac{\theta_2}{2}, 0)][\cos \frac{\theta_1}{2}, (\sin \frac{\theta_1}{2}, 0, 0)]$$

$$w = \cos \frac{\theta_1}{2} \cos \frac{\theta_2}{2} \cos \frac{\theta_3}{2} - \sin \frac{\theta_1}{2} \sin \frac{\theta_2}{2} \sin \frac{\theta_3}{2} \quad q = [w, (x, y, z)]$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 欧拉角向四元数的转换

$$x = \sin \frac{\theta_1}{2} \sin \frac{\theta_2}{2} \cos \frac{\theta_3}{2} + \cos \frac{\theta_1}{2} \cos \frac{\theta_2}{2} \sin \frac{\theta_3}{2}$$

$$y = \cos \frac{\theta_1}{2} \sin \frac{\theta_2}{2} \cos \frac{\theta_3}{2} - \sin \frac{\theta_1}{2} \cos \frac{\theta_2}{2} \sin \frac{\theta_3}{2}$$

$$z = \sin \frac{\theta_1}{2} \cos \frac{\theta_2}{2} \cos \frac{\theta_3}{2} + \cos \frac{\theta_1}{2} \sin \frac{\theta_2}{2} \sin \frac{\theta_3}{2}$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 四元数向旋转矩阵的转换

$$q = (w, x, y, z) = [\cos(\theta/2), \mathbf{n} \sin(\theta/2)]$$

$$M = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2zw & 2xz + 2yw \\ 2xy + 2zw & 1 - 2x^2 - 2z^2 & 2yz - 2xw \\ 2xz - 2yw & 2yz + 2xw & 1 - 2x^2 - 2y^2 \end{bmatrix}$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 旋转矩阵向四元数的转换

$$M = \begin{bmatrix} M_{00} & M_{01} & M_{02} \\ M_{10} & M_{11} & M_{12} \\ M_{20} & M_{21} & M_{22} \end{bmatrix}$$

$$q = [w, (x, y, z)]$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 旋转矩阵向四元数的转换 ($\text{tr}(M) > 0$)

$$w = \frac{1}{2} \sqrt{\text{tr}(M) + 1}$$

$$x = (M_{21} - M_{12}) / 4w$$

$$y = (M_{02} - M_{20}) / 4w$$

$$z = (M_{10} - M_{01}) / 4w$$

$\text{tr}(A)$ = the trace of the matrix A 矩阵A的迹。

定义： 设 $A = (a_{ij})$ 是一个 n 阶方阵， A 的对角线元素之和称为 A 的迹，记为 $\text{tr}A$ ，即
 $\text{tr}A = a_{11} + a_{22} + \dots + a_{nn}$

它有两个重要的**性质**：

性质 1： $b_1 + b_2 + \dots + b_n = \text{tr}A$

性质 2： $b_1 * b_2 * \dots * b_n = \det A$

其中 b_1, b_2, \dots, b_n 为矩阵A的特征值， $\det A$ 表示 A 的行列式。

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 旋转矩阵向四元数的转换 ($\text{tr}(M) < 0$)

$\text{tr}(M) < 0$ 时, 先从 x, y, z 分量中选取最大的分量。如果 $M_{00} > M_{11}$, 且 $M_{00} > M_{22}$, 则分量 $x > y$, 且分量 $x > z$

$$x = \frac{1}{2} \sqrt{M_{00} - M_{11} - M_{22} + 1}$$

$$y = (M_{01} + M_{10}) / 4x$$

$$z = (M_{02} + M_{20}) / 4x$$

$$w = (M_{21} - M_{12}) / 4x$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 单位四元数空间的球面线性插值

一个旋转变换 $\langle - \rangle$ 一个单位四元数

所有旋转 $\langle - \rangle$ 四元数空间的一个单位四维球面

方向的插值曲线应位于该球面上



球面线性插值

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 单位四元数空间的球面线性插值

$$SLERP(q_1, q_2, t) = \frac{q_1 \sin((1-t)\theta) + q_2 \sin(t\theta)}{\sin(\theta)}$$

$$\cos \theta = q_1 \bullet q_2$$

当 q_1 和 q_2 非常接近时,

$$Slerp(q_1, q_2, t) \approx (1-t)*q_1 + t*q_2$$

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 四元数超球面上圆弧的选择

四维球面的拓扑并不是三维欧式球面的直接推广

在单位四元数球面上径向相反的点表示相同的旋转

$$R_q(p) = qpq^{-1} = (-q)p(-q)^{-1}$$

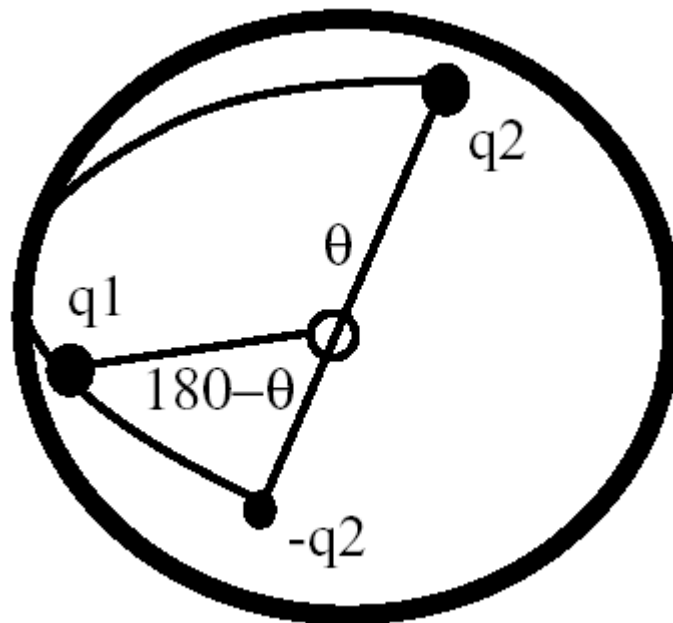
解决策略：从插值 (q_1, q_2) 和插值 $(q_1, -q_2)$ 中选择一个

$(q_1 - q_2) \cdot (q_1 - q_2) < (q_1 + q_2) \cdot (q_1 + q_2) \rightarrow$ 选择 (q_1, q_2)

$(q_1 - q_2) \cdot (q_1 - q_2) > (q_1 + q_2) \cdot (q_1 + q_2) \rightarrow$ 选择 $(q_1, -q_2)$

3.3 关键帧插值计算

3.3.3 方向表达与插值



The closer of the two representations of orientation is the better choice to use in interpolation.

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 四元数插值关键帧方向参数的过程

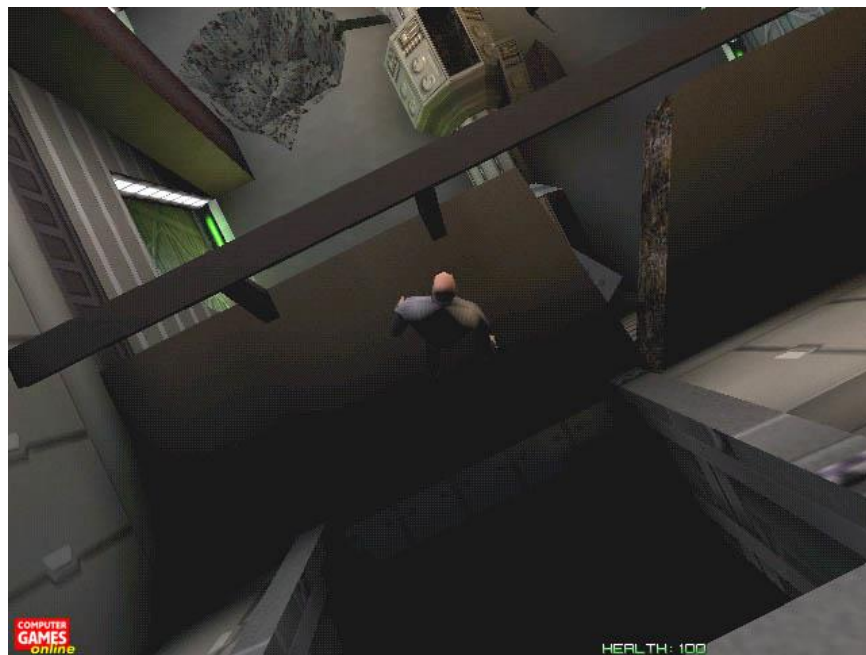
1. 用欧拉角方法建立物体的方向，并转化为单位四元数 q_i
2. 依据关键帧四元数 q_i , $i=0, 1, 2, \dots, n-1$, 生成插值四元数曲线 $q(t)$
3. 计算 t 时刻的四元数，把它转化为旋转矩阵

3.3 关键帧插值计算

3.3.3 方向表达与插值

■ 四元数的实际应用

相机旋转 in third-person-perspective games



3.3 关键帧插值计算

3.3.3 方向表达与插值

旋转表示方法的总结—欧拉角

Euler angles are quite suitable for integrating **ODEs**, but since inverse kinematics, dynamics, and spacetime optimization require (at least) freedom from **gimbal lock**, Euler angles are unsuitable for these applications.

3.3 关键帧插值计算

3.3.3 方向表达与插值

旋转表示方法的总结—四元数

In summary, use of quaternions to parameterize rotations leads to numerically well-conditioned systems in the applications under consideration, but incurs an overhead in **efficiency** and/or **code complexity** whenever **derivatives** are used for control or optimization.

Especially in light of recent developments, however, they may be the **best choice for interpolation** of three DOF rotations.

3.3 关键帧插值计算

参数关键帧技术小结

- 关键帧技术已发展成为可以用来插值影响运动的任何参数；
- 主要步骤；
- 插值参数和插值类型的选择；
- 方向表达与插值。

第3章 关键帧动画

3.1 关键帧动画概述

3.2 关键帧技术类型

3.3 关键帧插值计算

3.4 插值函数与参数的选择

3.5 样条驱动动画技术

3.4 插值函数与参数的选择

3.2.3 基于参数的关键帧技术中提出：

- 如何最好地生成中间帧对应的参数值



- 确定适合的插值函数

- 由插值函数快速而准确地计算中间帧的参数值

- 本节内容

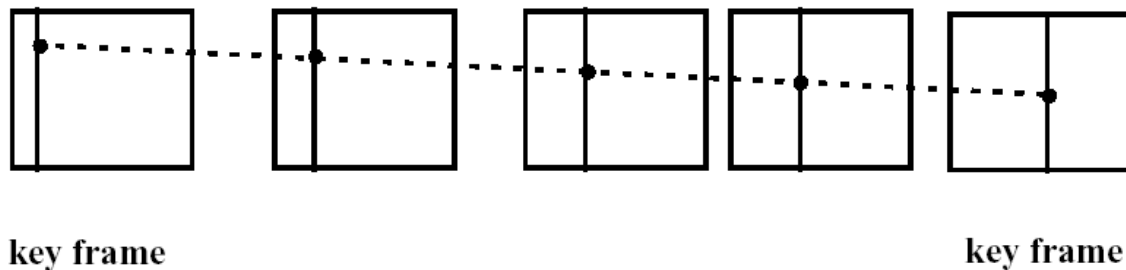
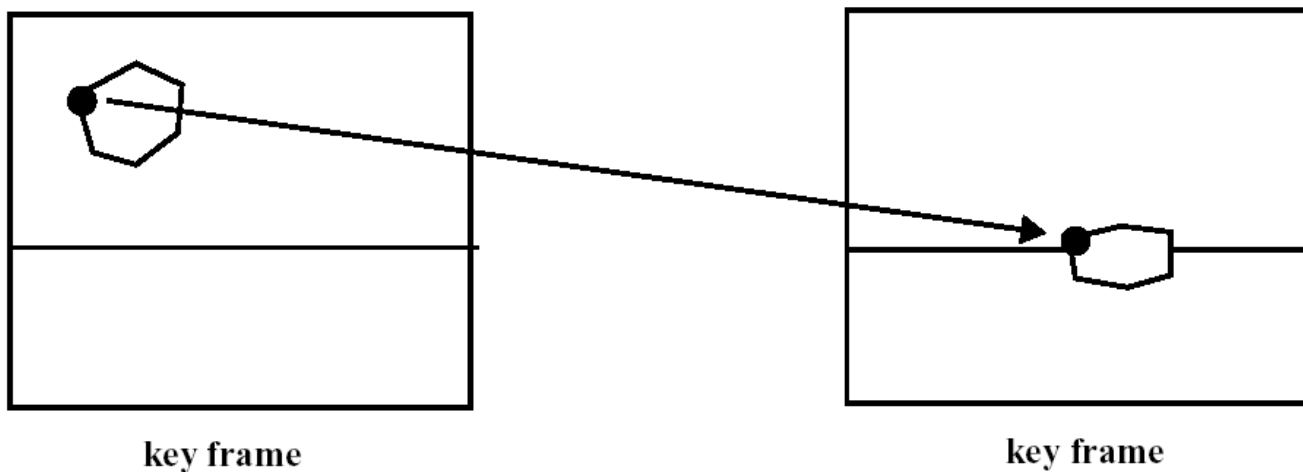
- 3.4.1 插值函数选择

- 3.4.2 插值参数选择

3.4 插值函数与参数的选择

3.4.1 插值函数选择

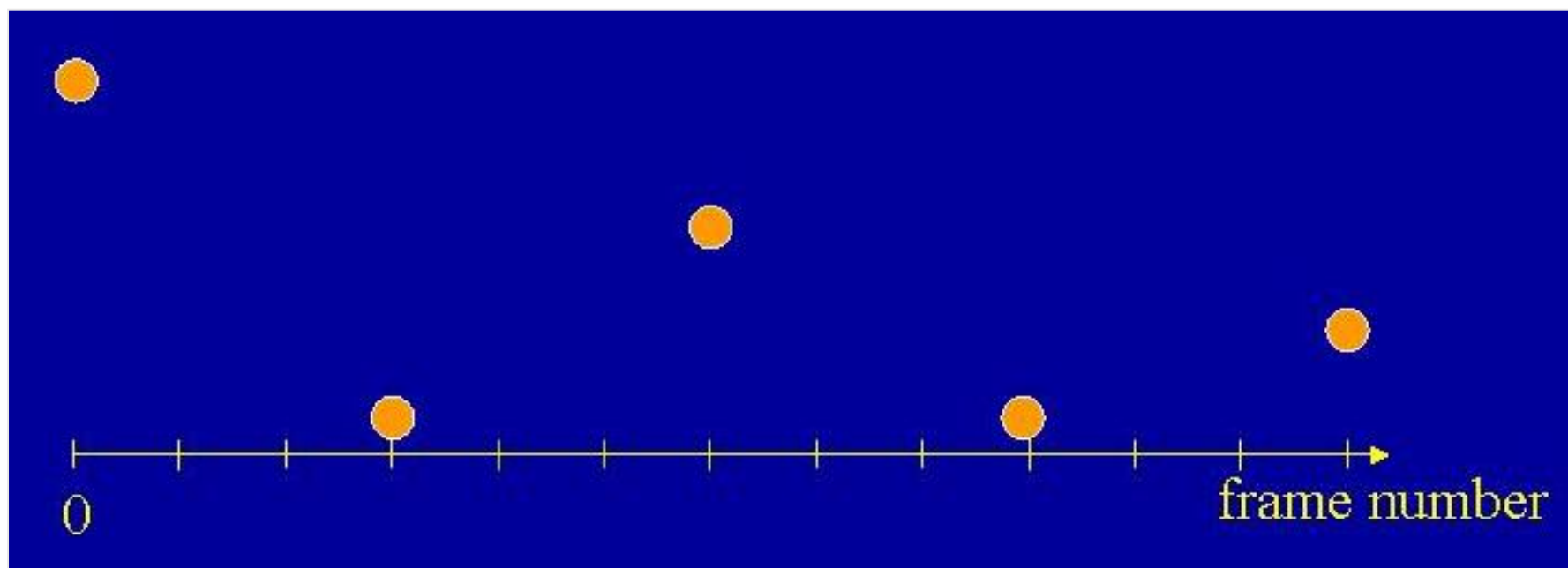
关键帧系统



3.4 插值函数与参数的选择

3.4.1 插值函数选择

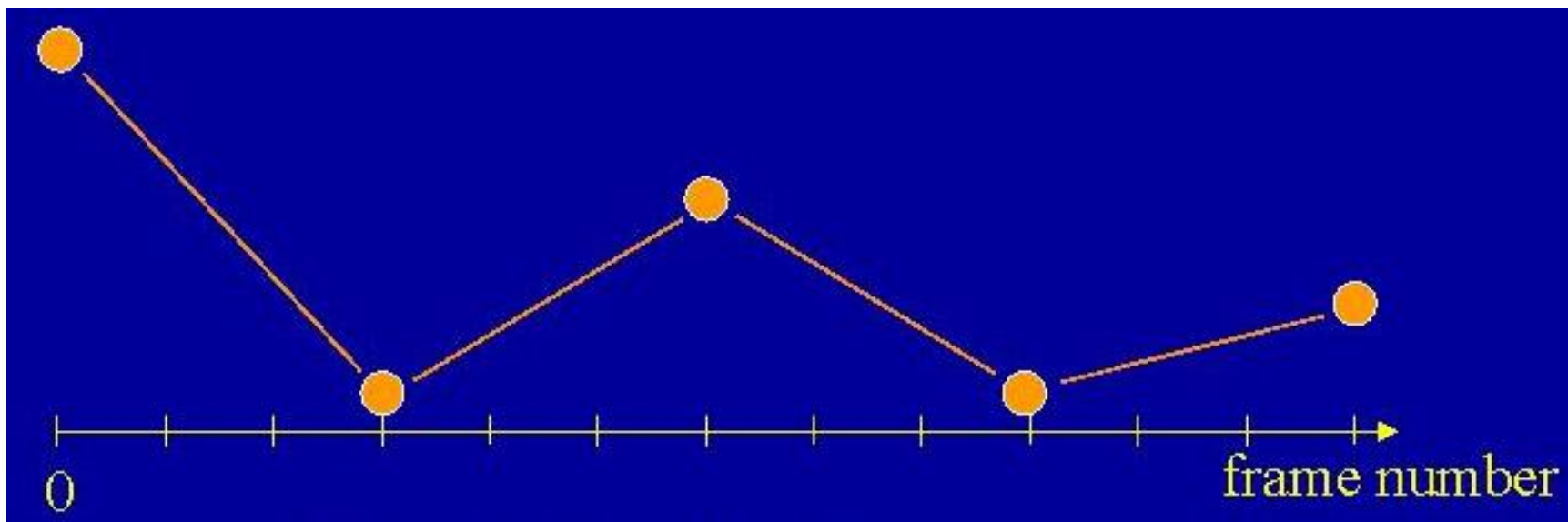
关键帧---弹起的小球



3.4 插值函数与参数的选择

3.4.1 插值函数选择

线性插值

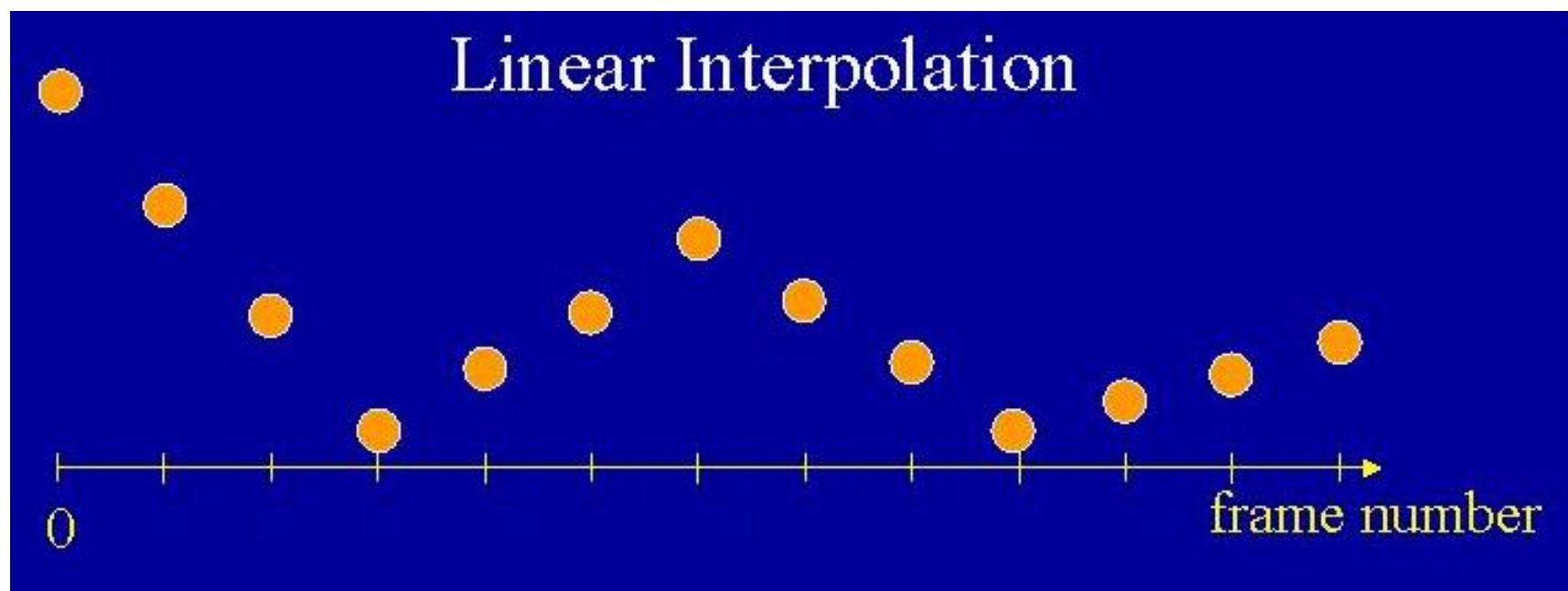


$$pos(frame_i) = pos(key_{prior}) + \left(\frac{pos(key_{next}) - pos(key_{prior})}{(key_{next} - key_{prior})} \right) (frame_i - key_{prior})$$

3.4 插值函数与参数的选择

3.4.1 插值函数选择

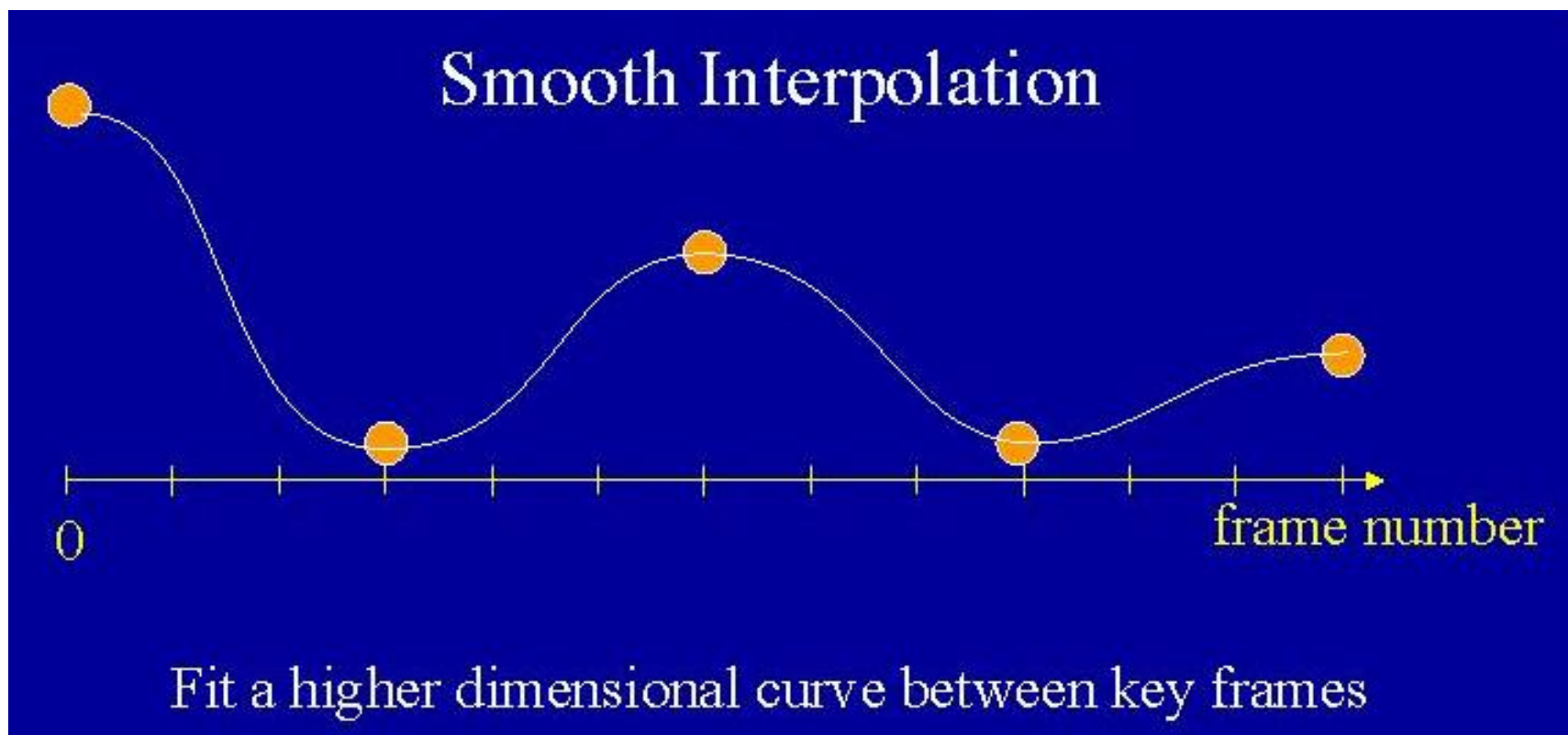
关键帧----弹起的小球



3.4 插值函数与参数的选择

3.4.1 插值函数选择

关键帧----弹起的小球



3.4 插值函数与参数的选择

3.4.1 插值函数选择

插值函数的选择 (1)

® 如何选择最适当的插值技术

✧ 首先判断给定的值是关键帧的实际值(插值), 还是仅仅用于控制插值函数的, 不需要很准确(近似)



3.4 插值函数与参数的选择

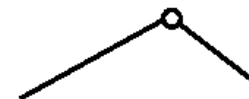
3.4.1 插值函数选择

插值函数的选择 (2)

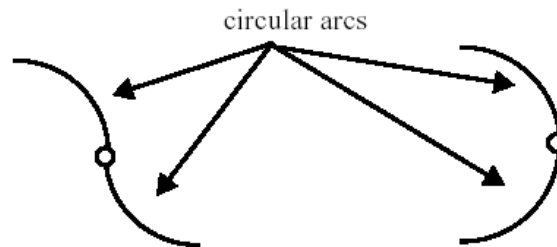
® 另一个影响插值技术使用的的问题是得到结果的光滑程度



a) positional discontinuity at the point



b) positional continuity but not tangential continuity at the point



c) positional and tangential continuity but not curvature continuity at the point

d) positional, tangential, and curvature continuity at the point

3.4 插值函数与参数的选择

3.4.1 插值函数选择

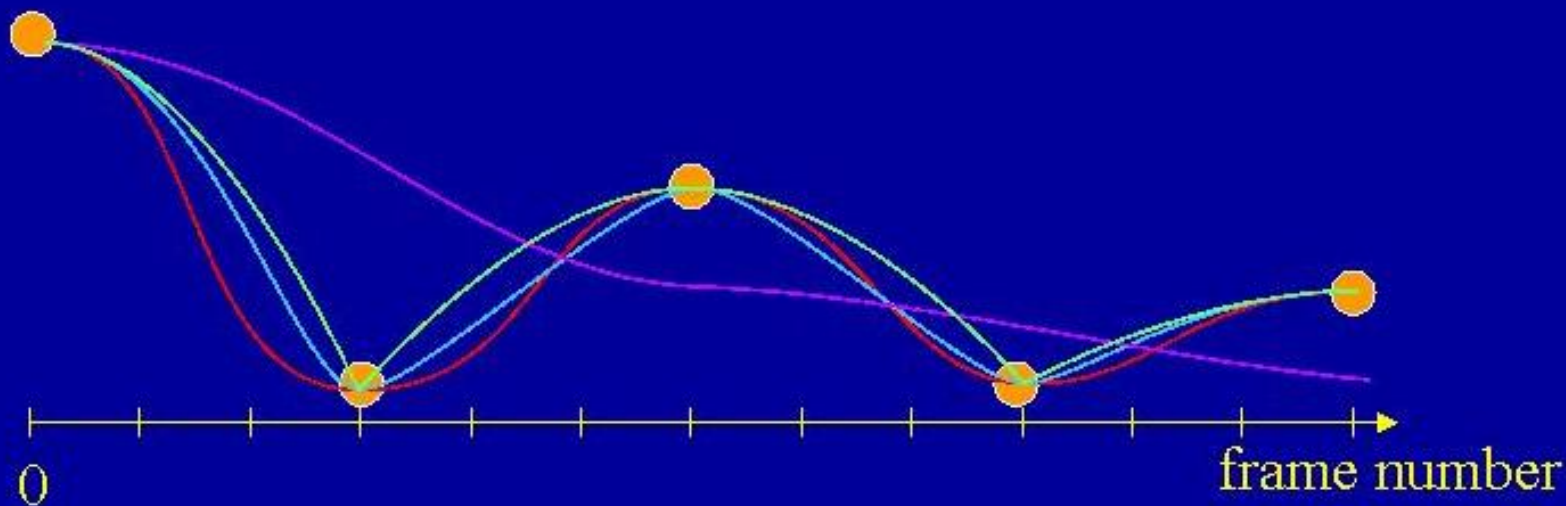
不同的插值函数

Decisions:

Do spline endpoints go through key frames exactly?

Must adjoining splines preserve continuous derivatives?

Allow time to reparameterize?

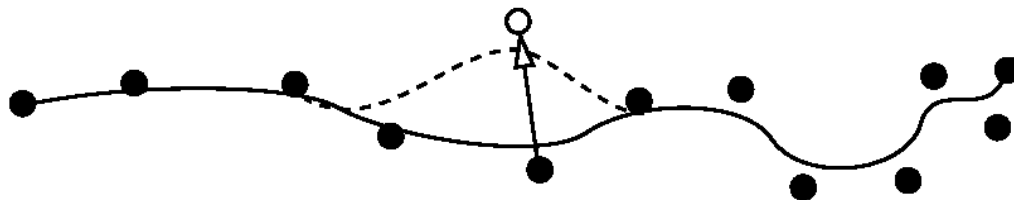


3.4 插值函数与参数的选择

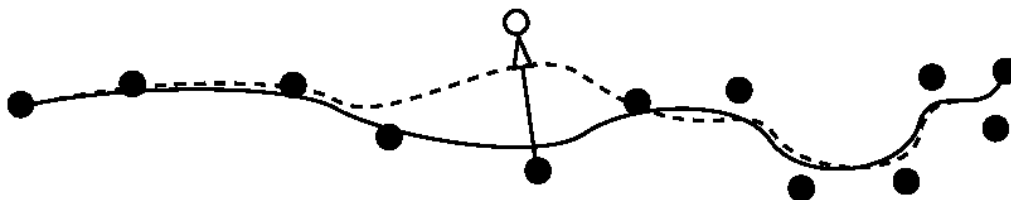
3.4.1 插值函数选择

插值函数的选择 (3)

1. 插值多项式的阶数
2. 插值函数需要局部控制还是全局控制



a) local control: moving one control point only changes the curve over a finite bounded region

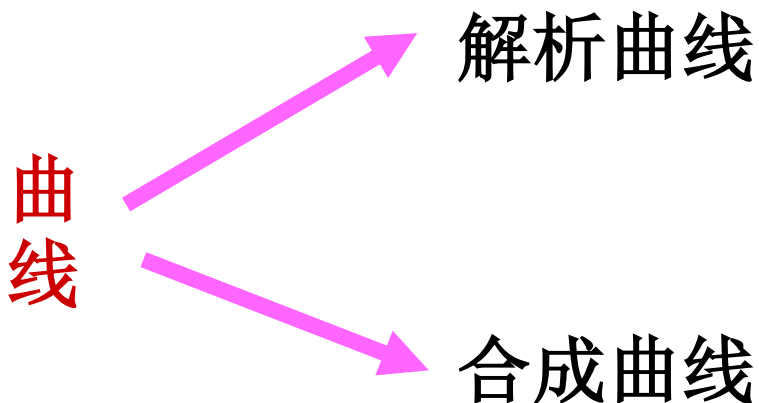


b) global control: moving one control point changes the entire curve; distant sections may change only slightly.

3.4 插值函数与参数的选择

3.4.1 插值函数选择

- 曲线是构建几何模型的最基本元素之一



3.4 插值函数与参数的选择

3.4.1 插值函数选择

解析曲线

- 解析曲线通常是先有**曲线方程**，然后才能把曲线画出来，这种方式对于曲线的构造者来说是非常**复杂**且**不直观**的，而且改变曲线的一些参数，设计者也无法立刻了解曲线形状会做怎样的变化

3.4 插值函数与参数的选择

3.4.1 插值函数选择

合成曲线

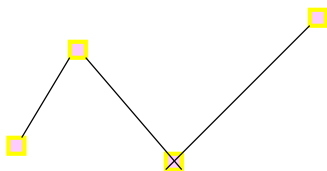
- 在作曲线设计时，设计者通常希望能先将大致形状用很直观的方式描绘出来，并能够很容易的依照所需要的形状作修改，因此合成曲线是比较合适的方式
- 合成曲线通常以参数的形式来表现，是由设计者根据其设计的需求和几何信息，去“合成”出这条曲线。这样由设计者输入的几何数据，就是曲线的控制点

3.4 插值函数与参数的选择

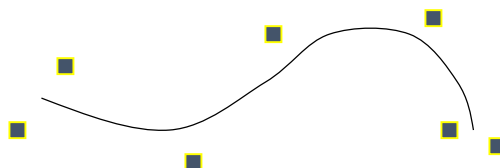
3.4.1 插值函数选择

样条 (Spline) 曲线&控制点

- 根据一组给定的点（控制点），利用数学方法拟和一条曲线
- 艺术家设置一组控制点，程序根据这些控制点生成样条曲线



Polyline



B样条

3.4.1 插值函数选择

Bezier Curve

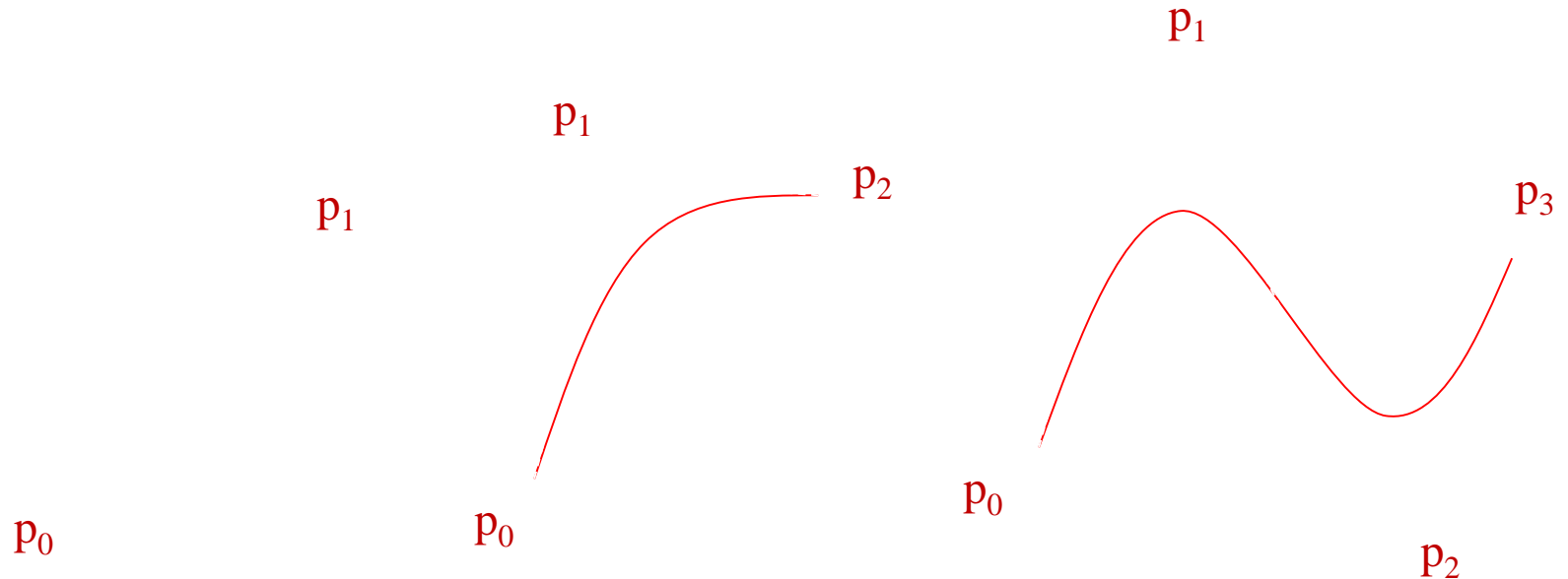
$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} (t)^i$$

$$\mathbf{v}(t) = \sum_{i=0}^n B_i^n(t) \mathbf{p}_i$$

3.4.1 插值函数选择

Bezier Curve

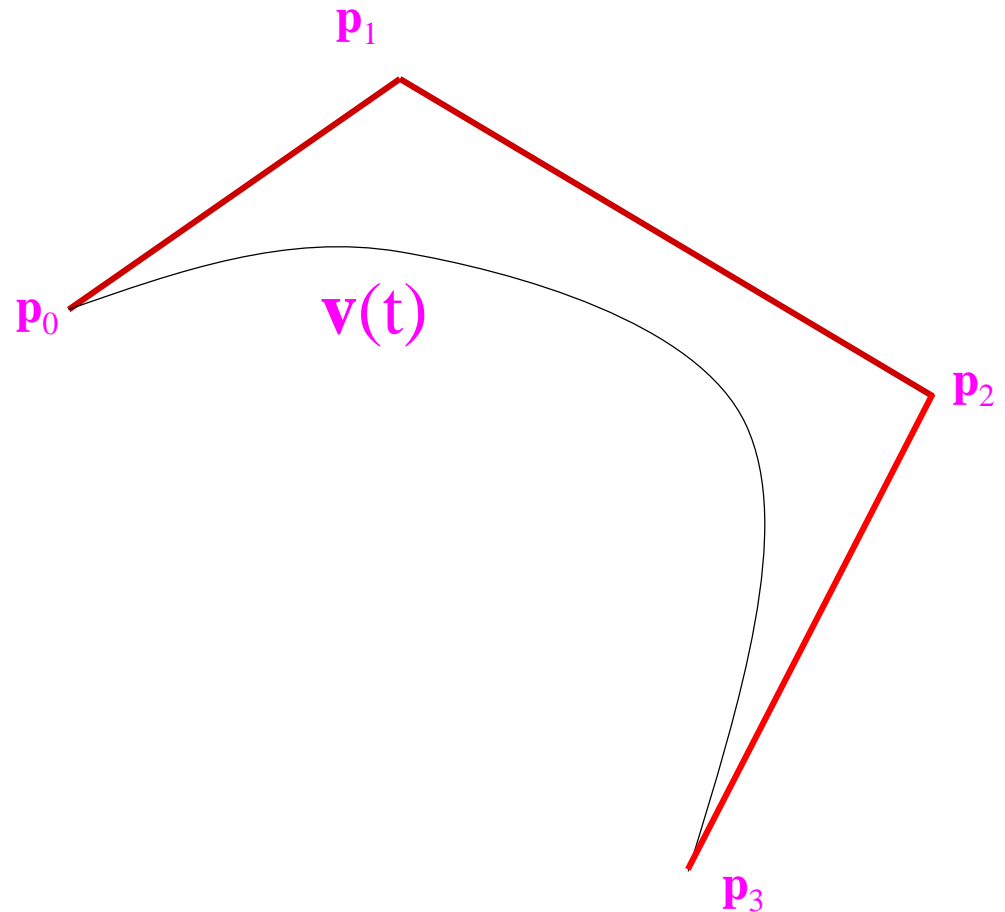
- Bezier curves can be thought of as a higher order extension of linear interpolation



3.4.1 插值函数选择

Bezier Curve

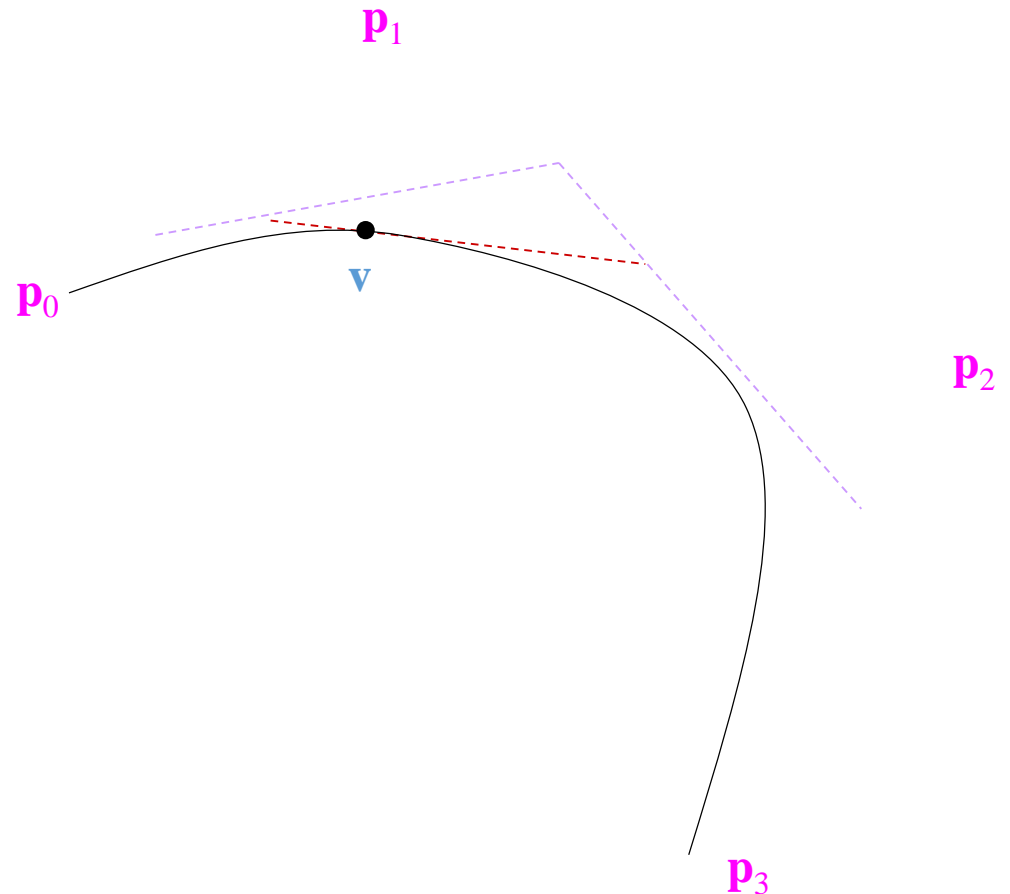
- Find the point v on the curve as a function of parameter u :



3.4.1 插值函数选择

de Casteljau Algorithm

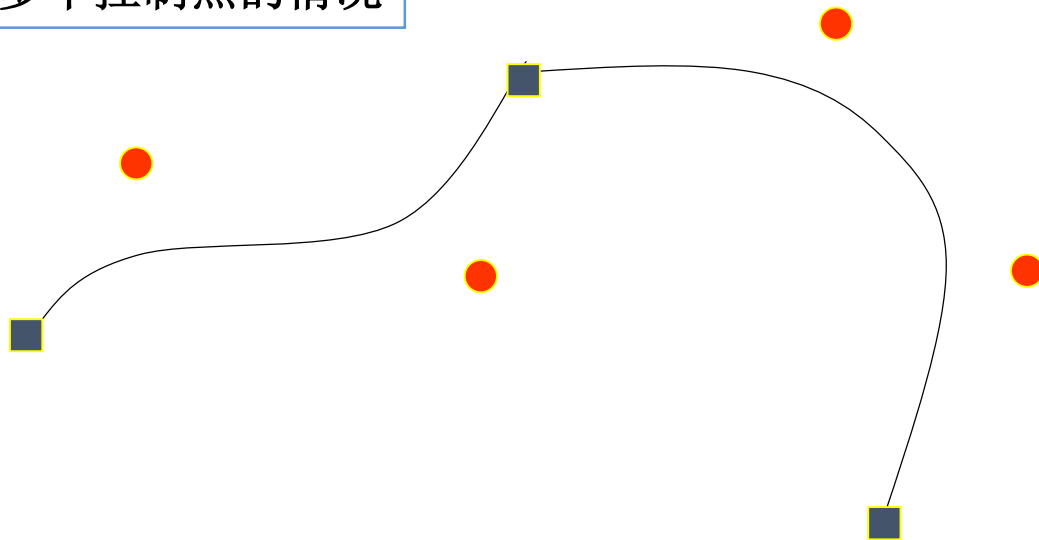
- The de Casteljau algorithm describes the curve as a recursive series of linear interpolations
- This form is useful for providing an intuitive understanding of the geometry involved, but it is not the most efficient form



3.4.1 插值函数选择

Bezier Spline Curve

多个控制点的情况



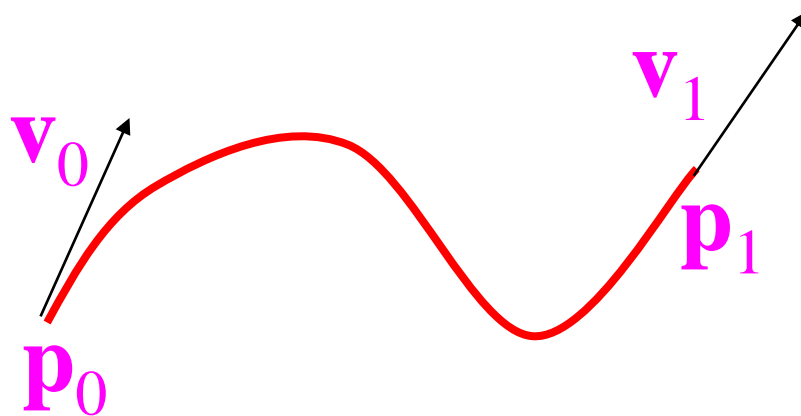
3.4.1 插值函数选择

埃尔米特（Hermite）曲线

- Let's look at an alternative way to describe a cubic curve
- Instead of defining it with the 4 control points as a Bezier curve, we will define it with a **position** and a **tangent** (velocity) at both the start and end of the curve (p_0, p_1, v_0, v_1)

3.4.1 插值函数选择

埃尔米特 (Hermite) 曲线



Hermite

3.4.1 插值函数选择

假设 $p = a*t^3 + b*t^2 + c*t + d$

- We want the value of the curve at $t=0$ to be p_0 , and p_1 at $t=1$
- We want the derivative of the curve at $t=0$ to be v_0 , and v_1 at $t=1$

$$p_0 = a0^3 + b0^2 + c0 + d = d$$

$$p_1 = a1^3 + b1^2 + c1 + d = a + b + c + d$$

$$v_0 = 3a0^2 + 2b0 + c = c$$

$$v_1 = 3a1^2 + 2b1 + c = 3a + 2b + c$$

3.4.1 插值函数选择

$$\mathbf{p}_0 = \mathbf{d}$$

$$\mathbf{p}_1 = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d}$$

$$\mathbf{v}_0 = \mathbf{c}$$

$$\mathbf{v}_1 = 3\mathbf{a} + 2\mathbf{b} + \mathbf{c}$$

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

3.4.1 插值函数选择

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix}$$

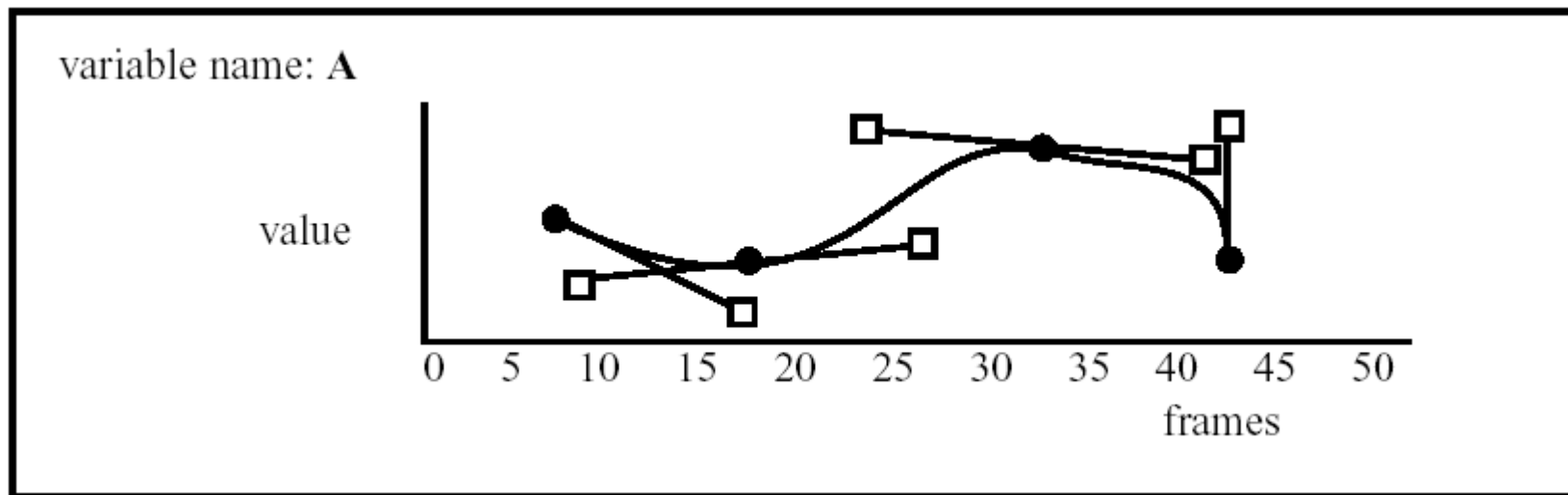
3.4.1 插值函数选择

埃尔米特 (Hermite) 曲线

$$\mathbf{v} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_{0x} & p_{0y} & p_{0z} \\ p_{1x} & p_{1y} & p_{1z} \\ v_{0x} & v_{0y} & v_{0z} \\ v_{1x} & v_{1y} & v_{1z} \end{bmatrix}$$

3.4.1 插值函数选择

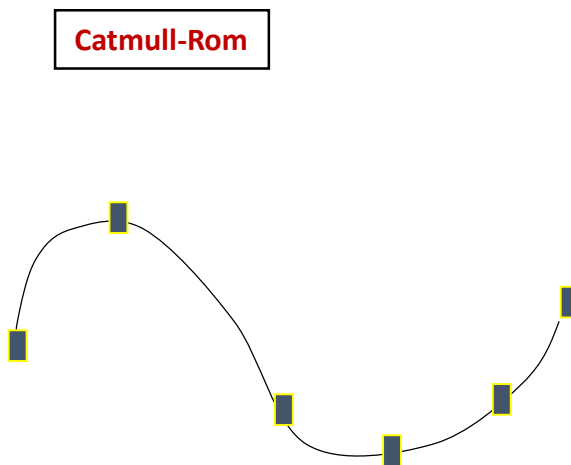
埃尔米特样条曲线



3.4.1 插值函数选择

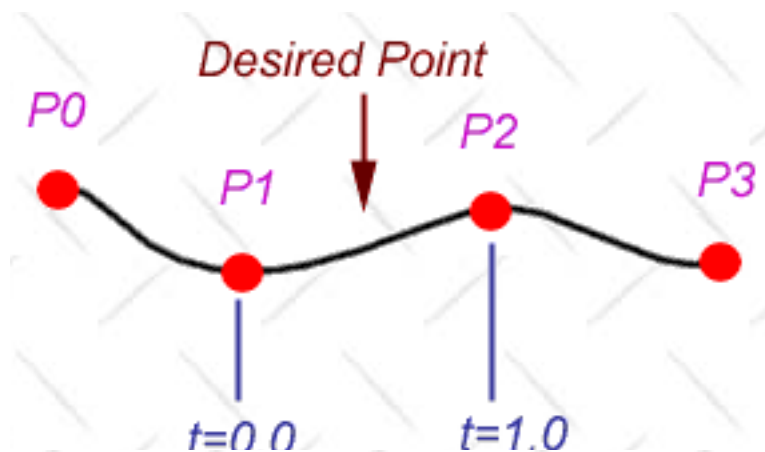
Catmull-Rom Spline

- 属于插值样条
- 但所有控制点落在拟和曲线上
- 曲线的显示和编辑更直观
- 该曲线是 C^1 连续



3.4.1 插值函数选择

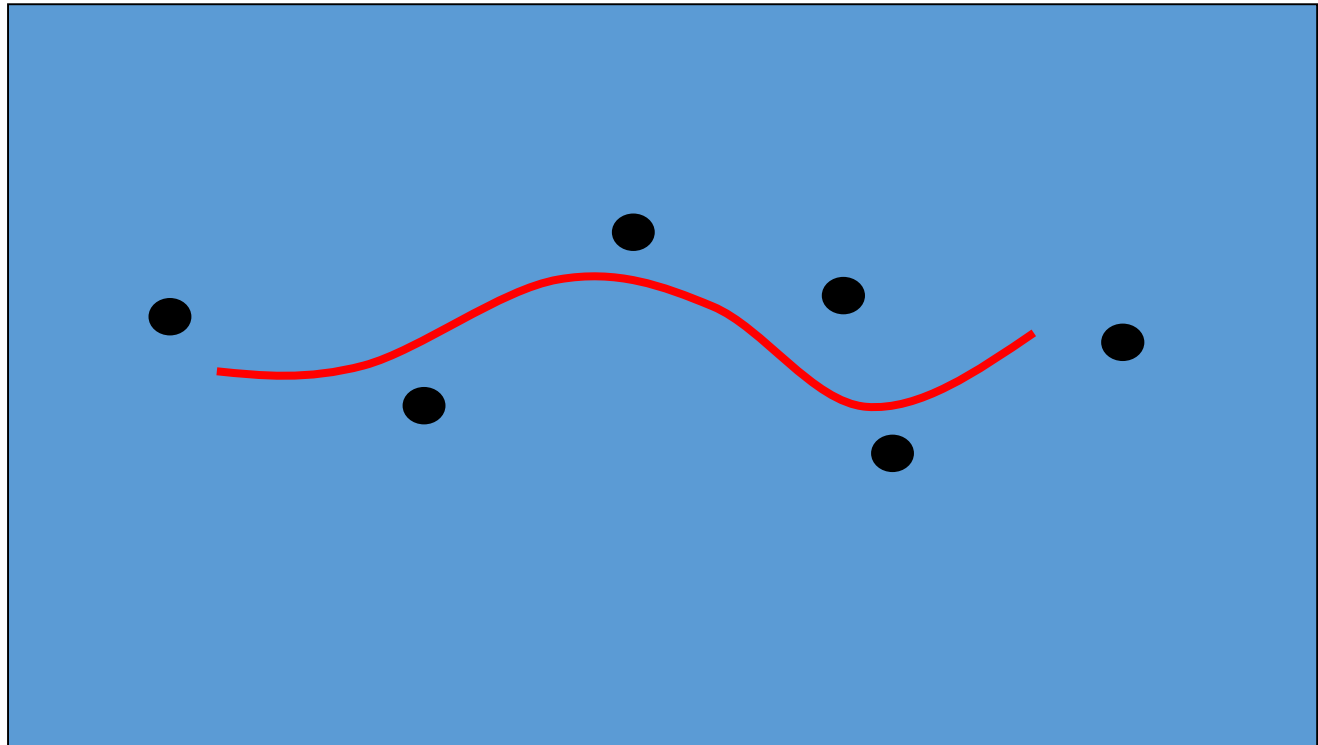
Catmull-Rom Spline



$$q(t) = 0.5 * (1.0f, t, t^2, t^3) * \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

3.4.1 插值函数选择

B样条



3.4.1 插值函数选择

对插值函数的计算

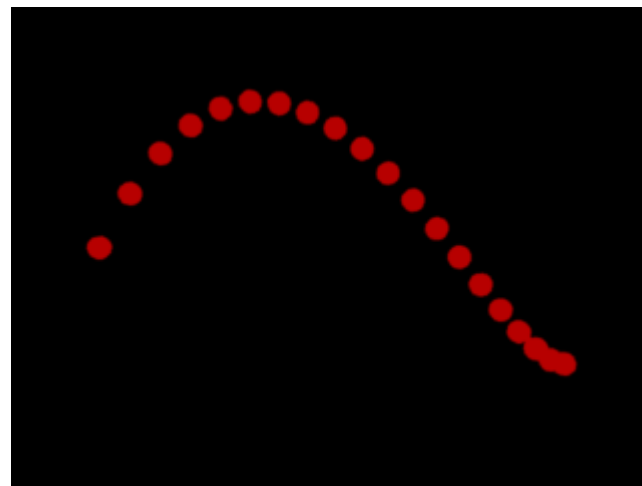
- 关键帧对应于特定时刻的参数值
- X, Y, Z 坐标独立处理
- 例如将 (X, t) 用作插值曲线的控制点,
- $X = P_x(t)$
- $Y = P_y(t), Z = P_z(t) \rightarrow (X, Y, Z)$

3.4.1 插值函数选择

插值参数与插值结果的关系

❖ 将插值参数(例如 u)
改变一个常量，得到的
值（如位置）未必
会改变一个常量

❖ 这意味着自变量的
变化和因变量的变化
一般是不同的



3.4.1 插值函数选择

插值—参数化运动

- 这个动画显示一个沿着三次参数曲线运动的球
- $P(t) = a*t^3 + b*t^2 + c*t + d$
球的位置是通过均匀改变参数值得到的 (如 $t = 0.0, 0.1, 0.2, \dots$).
- 参数空间等距离的值对应于非等距的函数值点集



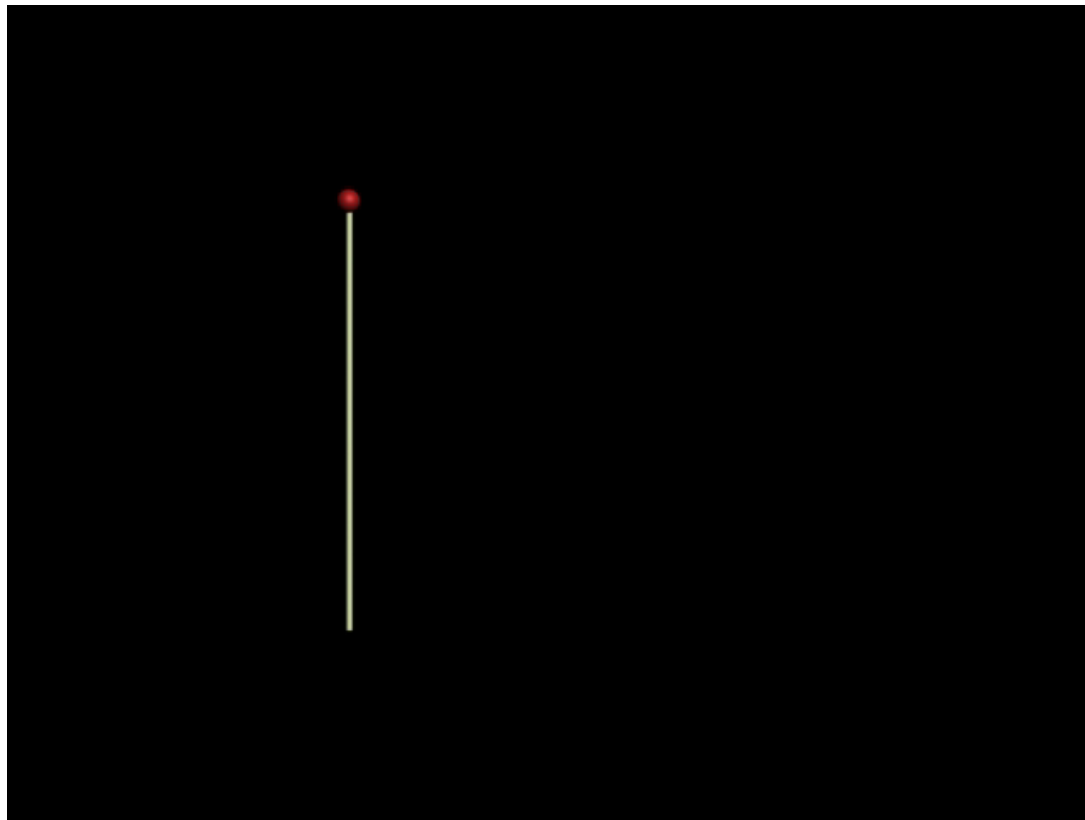
3.4 插值函数与参数的选择

3.4.2 插值参数选择

- 在应用参数关键帧技术时，应注意所插值的参数，否则会产生不恰当的运动。以下两例说明了这一点。

3.4.2 插值参数选择

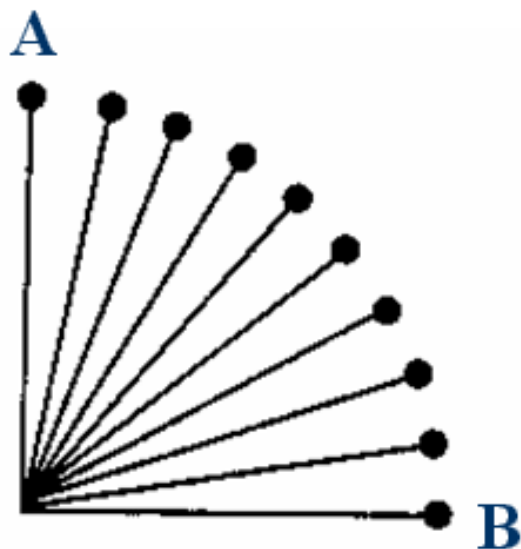
例1： 直线由A点变换至B



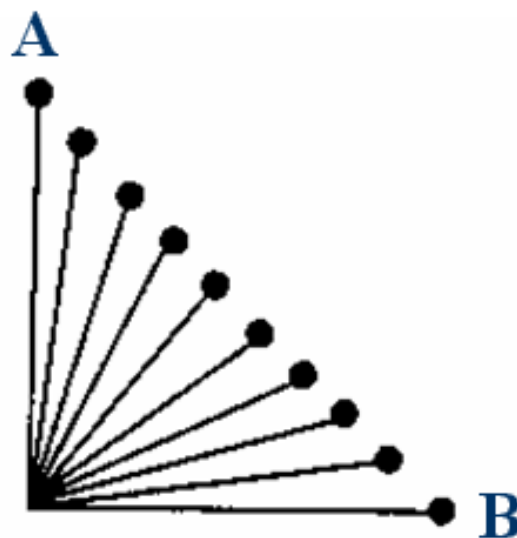
3.4.2 插值参数选择

例1：直线由A点变换至B

插值参数的不同，可有(a)(b)两种动画效果。



(a) 插值线旋转角



(b) 插值端点

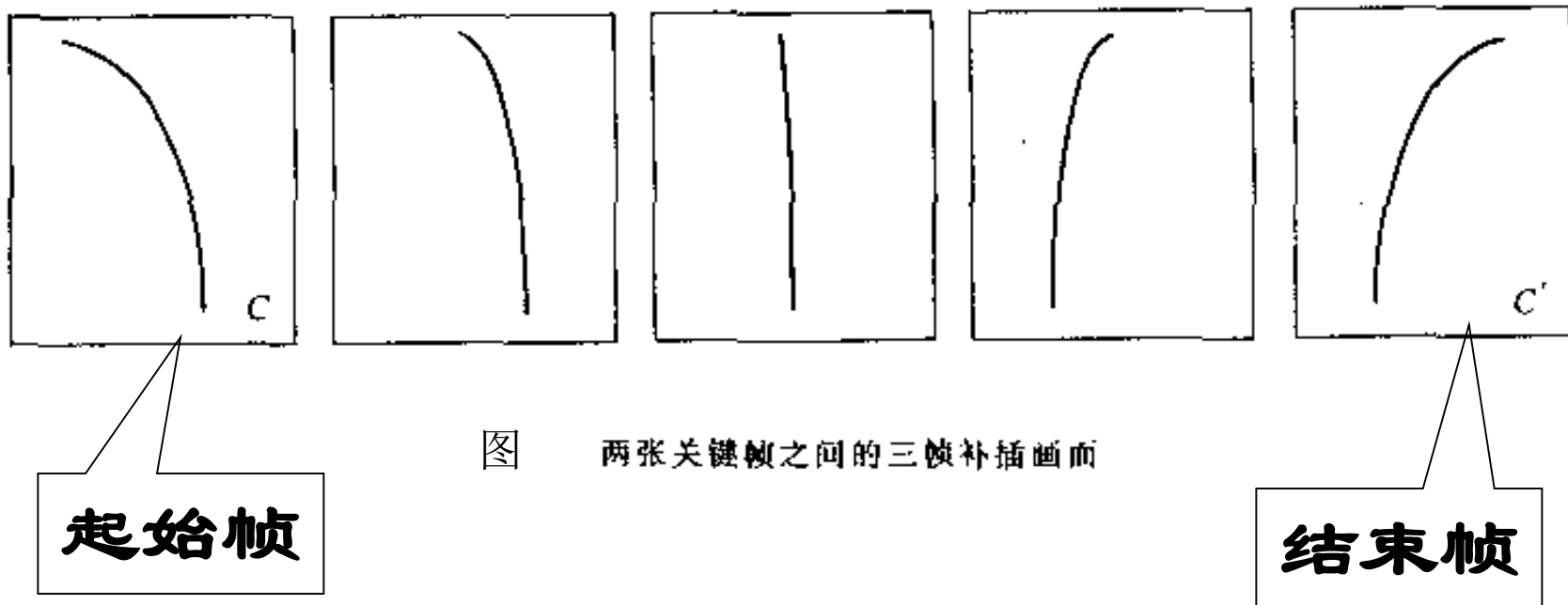
3.4.2 插值参数选择

例2：曲线运动



3.4.2 插值参数选择

例2：曲线运动 — 在起始和结束帧间插补三帧



3.4.2 插值参数选择

已知：曲线C和C'的拟合函数

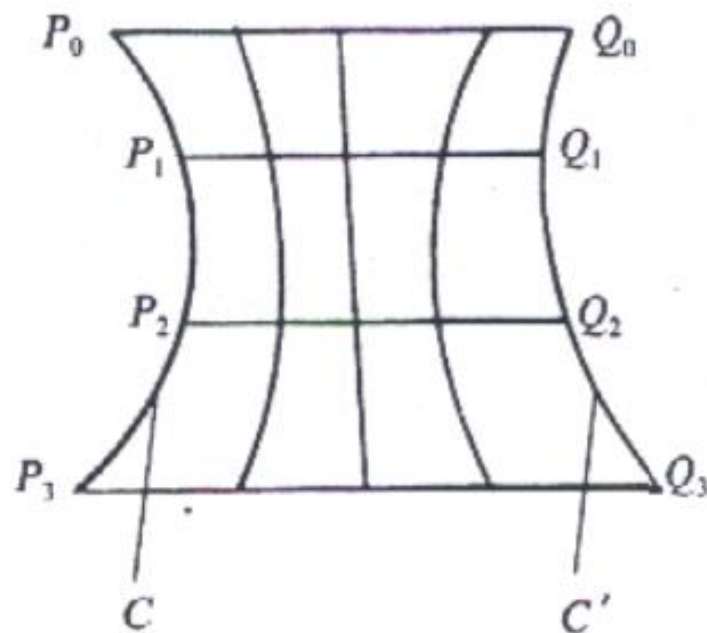
设想这五帧画面重叠在一起，并且曲线C和曲线C'，各自用四个点经过拟合或者插值得到。采用调配函数方法，写成

$$C(s) = \sum_{j=0}^3 P_j \varphi_j(s), \quad C'(s) = \sum_{j=0}^3 Q_j \varphi_j(s)$$

3.4.2 插值参数选择

确定插值两关键帧间对应的控制点

规定两组点的对应关系为： $P_0 \leftrightarrow Q_0, P_1 \leftrightarrow Q_1, P_2 \leftrightarrow Q_2, P_3 \leftrightarrow Q_3$.



3.4.2 插值参数选择

确定线性插值两端点关键帧

因为 P_i, Q_i ($i=0, 1, 2, 3$) 位置已知, 只要令:

$$R_i = (1-t)P_i + tQ_i$$

当 $t=0$ 时, $R_i = P_i$ (初始帧曲线的型值点)

当 $t=1$ 时, $R_i = Q_i$ (终止帧曲线的型值点)

$$t \in [0, 1]$$

3.4.2 插值参数选择

★ 插值求中间控制点

取 $t=0.25, 0.5, 0.75$

$$P_0=(2.5,8), P_1=(3.5,6), P_2=(3.5,3.5), P_3=(2,1)$$

$$Q_0=(8.8,8), Q_1=(8.5,6), Q_2=(8.8,3.5), Q_3=(10,1)$$

分别代入下式：

$$R_i^{(1)} = \frac{3}{4}P_i + \frac{1}{4}Q_i, \quad R_i^{(2)} = \frac{1}{2}P_i + \frac{1}{2}Q_i, \quad R_i^{(3)} = \frac{1}{4}P_i + \frac{3}{4}Q_i$$

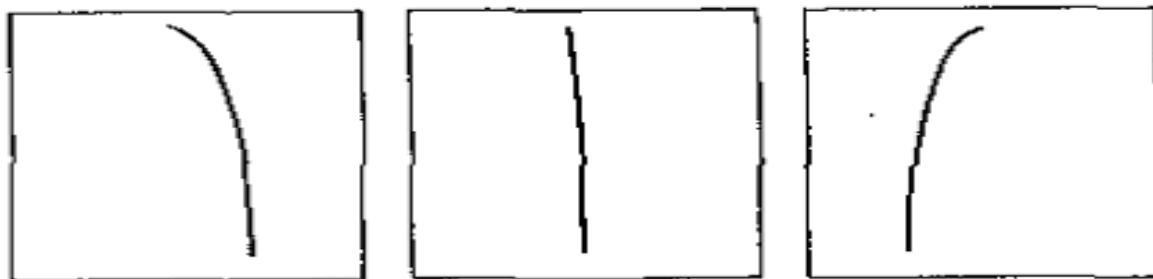
3.4.2 插值参数选择

✧ 得中间帧曲线拟合函数

然后,由 $R_i^k (i=0,1,2,3; k=1,2,3)$ 用拟合 C 及 C' 同样的方法得到

$$C_1(s) = \sum_{j=0}^3 k_j^{(1)} \varphi_j(s), \quad C_2(s) = \sum_{j=0}^3 k_j^{(2)} \varphi_j(s), \quad C_3(s) = \sum_{j=0}^3 k_j^{(3)} \varphi_j(s)$$

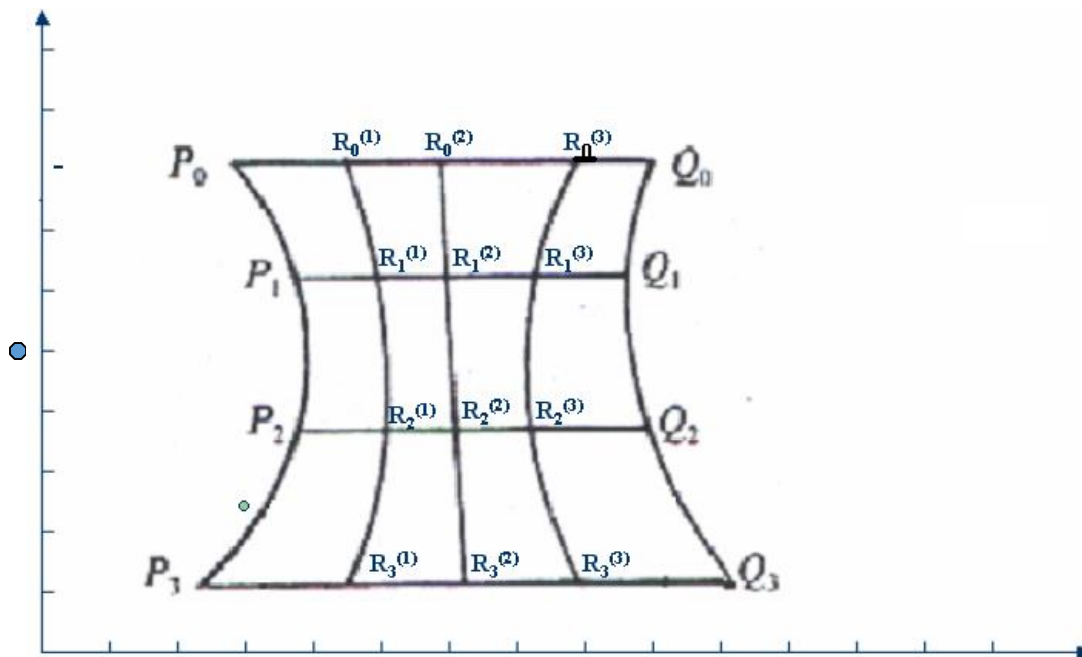
于是, $C_1(s), C_2(s), C_3(s)$ 便是插补出来的中间画面(曲线).



3.4.2 插值参数选择

★ 绘制曲线

- $P_0=(2.5,8)$, $P_1=(3.5,6)$, $P_2=(3.5,3.5)$, $P_3=(2,1)$
- $Q_0=(8.8,8)$, $Q_1=(8.5,6)$, $Q_2=(8.8,3.5)$, $Q_3=(10,1)$



第3章 关键帧动画

3.1 关键帧动画概述

3.2 关键帧技术类型

3.3 关键帧插值计算

3.4 插值函数与参数的选择

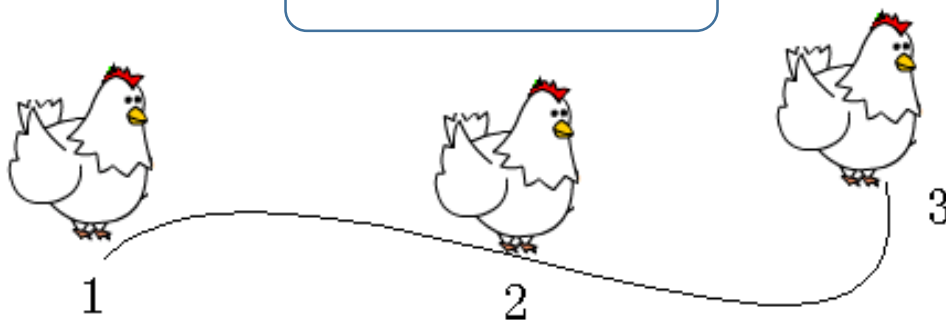
3.5 样条驱动动画技术

3.5 样条驱动动画技术(路径动画)

■样条驱动动画：

- 先设计好物体的**运动轨迹**，然后指定物体**沿该轨迹运动**；
- 通常运动轨迹为**三次样条曲线**，由用户**交互**给出；
- 假设物体的运动轨迹为一空间**参数曲线** $Q(u)$ ；
- 需要对 $Q(u)$ **等间隔采样**，以求得物体在每一帧的位置；

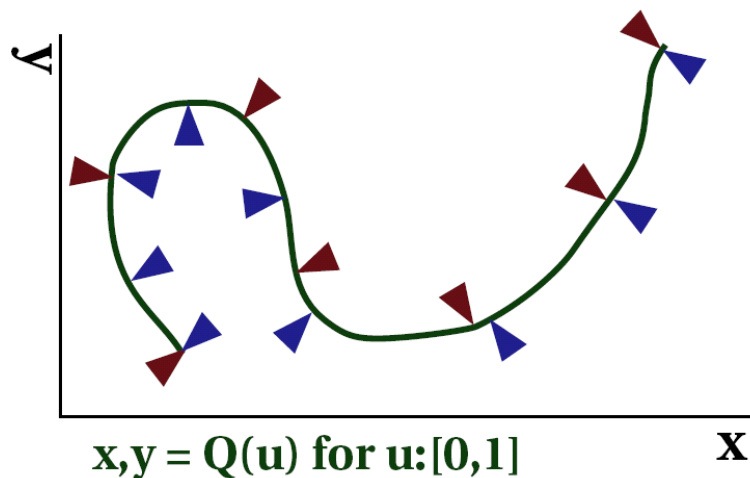
有几种间隔？



3.5 样条驱动动画技术

■ 等参数采样与等弧长采样：

- 在给定的帧数内使物体**匀速率**沿样条曲线 $Q(u)$ 运动；
- 以参数 u 等间距采样？ **X**
- **等间距的参数**不一定对应**等间距的弧长**；
- 对曲线以弧长为参数**重新参数化**；
- 便于对运动做**细微的控制**；



3.5 样条驱动动画技术

■弧长参数化:

- 弧长函数 $s=A(u)$ 是单调递增函数;
- 重新参数化: $Q(u) \rightarrow Q(A^{-1}(s))$;
- 关键如何求得: $u = A^{-1}(s)$

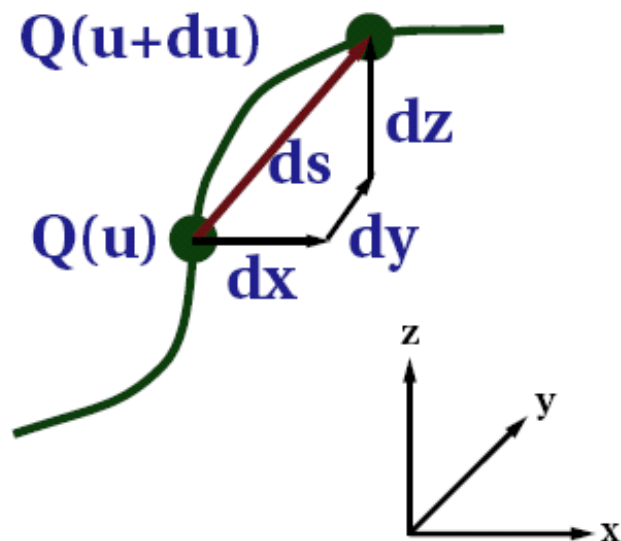
步骤1: 利用二分查找法计算给定弧长 s_a 所对应的参数值 u_a ;

步骤2: 利用积分公式计算弧长:

$$Q(u) = (x(u), y(u), z(u)) = au^3 + bu^2 + cu + d$$

$$ds = \sqrt{x'(u)^2 + y'(u)^2 + z'(u)^2} du$$

$$s = A(u) = \int_{u_0}^u \sqrt{Au^4 + Bu^3 + Cu^2 + Du + E} du$$



3.5 样条驱动动画技术

■弧长计算的加速方法：

- 动画制作中要经常调整物体的轨迹曲线并实时观看调整后物体的运动效果；
- Watt提出一种基于向前差分的近似算法：
 - 首先用向前差分计算曲线上的点 $Q(i \Delta u)$ ，建立一张关于 u 的累加弧长查找表；
 - 在查找表中对应于 $u_i = i \Delta u$ 的第 i 个元素为：

$$\sum_{j=0}^i d_j, \quad d_j = \|Q(u_j) - Q(u_{j-1})\|, \quad d_0 = 0$$

- 对于给定弧长 s ，先查找满足 $\sum_{j=0}^i d_j < s < \sum_{j=0}^{i+1} d_j$ 的 i ，然后采用线性插值近似求得弧长所对应的参数 u ：

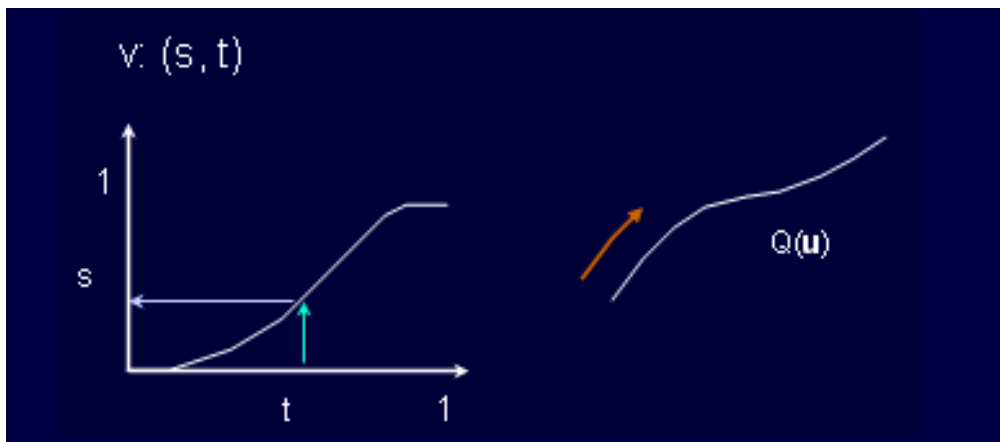
$$u = \frac{\sum_{j=0}^{i+1} d_j - s}{d_{i+1}} u_i + \frac{s - \sum_{j=0}^i d_j}{d_{i+1}} u_{i+1}$$

3.5 样条驱动动画技术

■速度曲线：

- 物体的运动可以由速度曲线来控制，**速度曲线**是弧长对时间的函数；
- 速度曲线可以表示为一二维样条曲线 $V(u)$ ，是弧长对时间的函数：

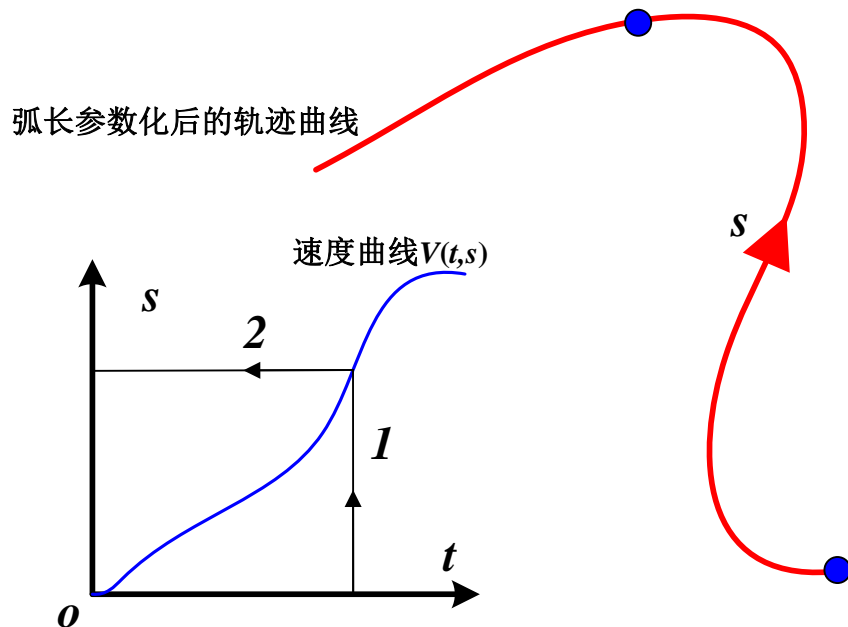
$$V(u) = (t, s) = (T(u), s(u))$$



3.5 样条驱动动画技术

■速度曲线：

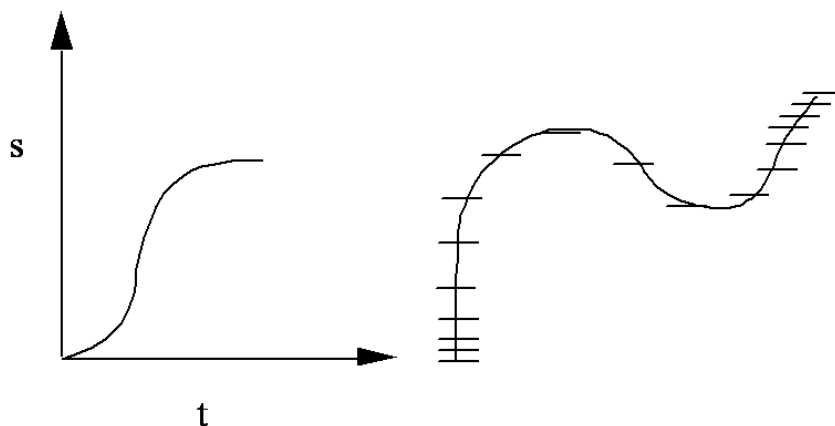
- 给定一条空间轨迹曲线和一条速度曲线，在某一时刻物体在曲线上的位置计算；
 1. 从速度曲线上找到给定时间对应的弧长；
 2. 使物体沿轨迹曲线运动该弧长距离；



3.5 样条驱动动画技术

■样条驱动控制运动的优点：

- 直接指定物体的运动轨迹，避免了关键帧方法中的样条插值问题；
- 空间轨迹曲线和速度曲线独立指定：
 - 用不同的速度曲线应用于相同的空间曲线，生成不同的运动效果；
- 速度曲线的控制非常直观，可以预先存储有用的曲线



思考题：

1. 什么是关键帧技术，如何分类？
2. 参数关键帧动画与路径动画的区别、优缺点？
3. 四元数、旋转矩阵和欧拉角的特点分别是什么？三者之间的转化关系如何？
4. 参数关键帧动画的主要步骤是什么？如何更好的生成中间帧的参数？

项目建议：[树叶飘落](#)、[花瓣飘落](#)(1.基于视频的位置插值；2.方位插值)

結束