Curves

2008년 여름
박경신

Lines

- Line (Explicit form)
  - y = mx + b
  - m는 기울기 (the slope of the line)
  - b는 y-절편 (the y-intercept)
  - Vertical line을 제외한 모든 2D 직선에서 이 공식을 사용할 수 있다. (The slope of the vertical line is infinite and the y-intercept is either nonexistent or is all values along the y-axis.)

$$m = \frac{y\text{값의변화량}}{x\text{값의변화량}}$$

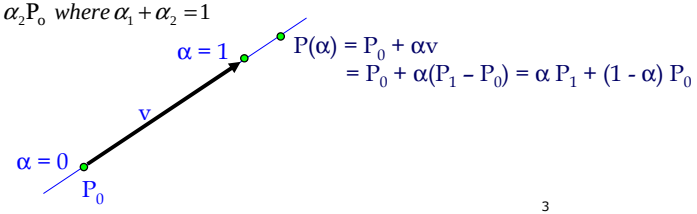
$$b = y\text{축과 만나는 점}$$

Lines

- Line (Parametric Form)
  - $P(\alpha) = P_0 + \alpha v$
  - P_0 는 임의의 점, v는 임의의 벡터, α는 임의의 점
  - 매개 변수 α를 변경함으로써 직선 위에 점들이 생성

$$P(\alpha) = P_o + \alpha v$$
$$= P_o + \alpha(P_1 - P_o)$$
$$= (1-\alpha)P_o + \alpha P_1$$

$$P = \alpha_1 P_1 + \alpha_2 P_o \quad where\ \alpha_1 + \alpha_2 = 1$$

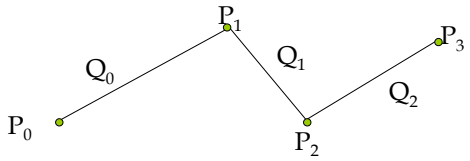


Linear Interpolation

- 두 점 간의 선형 보간 (Linear Interpolation)
  - $P(\alpha) = P_0 + \alpha (P_1 - P_0)$
    $= (1-\alpha)P_0 + \alpha P_1$
- 애니메이션에 linear interpolation을 사용할 때,
  - time t_0 와 t_1 와 시작점 P_0 와 끝점 P_1 을 사용한다.
  - 시간 t 가 $[t_0, t_1]$인 값을 parameter a 를 [0, 1]로 바꿔줘야 한다.
  - 이 때, percentage α 가 t_0 와 t_1 사이에 있는 t 값이다.

Piecewise Linear Interpolation

- 만약 두 점보다 많은 점을 선형 보간 해야 한다면, 가장 간단한 방법으로는 처음부터 두 점들 간의 구분적 선형 보간 (Piecewise linear interpolation)하는 방법으로 모든 점들을 보간하는 방법이다.
  - $t_i <= t <= t_{i+1}$에 P_i 와 P_{i+1} 간의 선형 보간을 이용한다.
  - $Q_0, Q_1, \ldots, Q_{n-1}$ where $Q_i(u) = (1 - u) P_i + uP_{i+1}$
- 하지만, 이 방법은 점들에서 급격하게 방향(direction)을 바꾸게 만든다. 보다 부드러운 보간을 원한다면 higher order polynomials 를 사용해야 한다.



Piecewise Linear Interpolation

```
osg::Vec3f EvaluatePiecewiseLinear(float t, unsigned int count,
                const osg::Vec3f *positions, const float *times) {
   // handle boundary conditions
   if (t <= times[0])
         return positions[0];
   else if (t >= times[count – 1])
         return positions[count – 1];
   // find segment and parameter
   for (unsigned int i = 0; i < count – 1; i++) {
      if (t < times[i + 1]) break;
   }
   float t0 = times[i];
   float t1 = times[i + 1];
   float u = (t – t0) / (t1 – t0);
   // evaluate
   return (1 – u) * positions[i] + u * positions[i + 1];
}
```

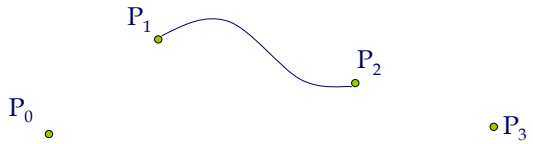

Cubic Interpolation

- 선형 보간 (Linear Interpolation)

  $P(t) = P_0 + t (P_1 - P_0) = (1 – t)P_0 + t P_1$

- 삼차 보간 (Cubic Interpolation)

  $P(t) = C_3 t^3 + C_2 t^2 + C_1 t + C_0$

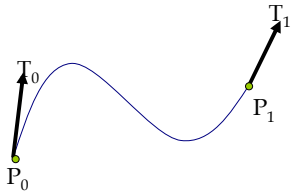


Three Major Types of Curves

- Hermite
  - Defined by two end points & two end point tangent vectors
- Bezier
  - Defined by two end points and two other points that control the end point tangent vectors
- Spline
  - Defined by four control points
    - uniform B-spline
    - non-uniform B-spline
    - β-spline

Hermite Curves

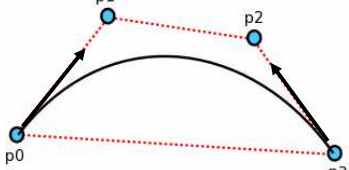
- Hermite 곡선은 두 점과 그 점의 기울기를 만족하는 삼차 곡선에 관한 해법이다.
  - P_0: 곡선의 시작점
  - T_0: 시작점에서의 tangent vector
  - P_1: 곡선의 끝점
  - T_1: 끝점에서의 tangent vector



- $P(t) = (1 - 3t^2 + 2t^3)P_0 + (t - 2t^2 + t^3)T_0 + (3t^2 - 2t^3)P_1 + (-t^2 + t^3)T_1$
  - $H1(t) = 1 - 3t^2 + 2t^3$
  - $H2(t) = 3t^2 - 2t^3$
  - $H3(t) = t - 2t^2 + t^3$
  - $H4(t) = -t^2 + t^3$
- Piecewise Hermite Curves를 사용하려면 곡선의 끝점에서 tangent가 다음 곡선의 시작점 tangent와 일치해야 한다.

Bezier Curve

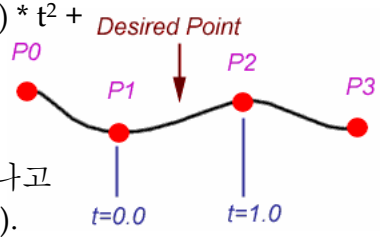
- Bezier 곡선은 모든 컨트롤 포인트들을 통과하지만 각각의 컨트롤 포인트마다 tangent vector를 가진다.
- Bezier 곡선은 첫 점과 끝 점이 반드시 곡선 안에 있고, 두 개의 중간의 컨트롤 포인트 (control points)는 이 곡선을 매끄럽게 해주는 효과를 제공한다.
- $P(t) = (1 - t)^3 P_0 + 3t(1 - t)^2 P_1 + 3t^2(1 - t) P_{12} + t^3 P_3$
  - $H1(t) = (1 - t)^3$
  - $H2(t) = 3t(1 - t)^2$
  - $H3(t) = 3t^2(1 - t)$
  - $H4(t) = t^3$



- Piecewise Bezier Curves를 사용하려면 곡선의 마지막 두 점과 다음 곡선의 시작 두 점이 일치해야 한다.

Catmull-Rom Splines

- Catmull-Rom Spline 곡선은 네 개의 컨트롤 포인트를 사용하여 중간의 두 포인트만 곡선이 지난다.
- $P(t) = 0.5 * ((-P_0 + 3P_1 - 3P_2 + P_3) * t^3 + (2P_0 - 5P_1 + 4P_2 - P_3) * t^2 + (-P_0 + P_2)t + 2P_1)$



- 이 곡선은 t = 0 일 때 점 P_1 를 지나고 t = 1일 때 점 P_2 를 지난다 (t [0,1]).
- Piecewise Catmull-Rom Spline Curves를 사용하려면, 현 지점에서 앞의 점과 현 지점의 점과 다음 두 점을 사용하여 그 네 점을 이용하여 spline 보간을 계산한다.