2D 게임 프로그래밍

- 반드시 카메라를 ON 하고 !
- 입장 이름은 "학번 이름”으로 설정 !
- 미리 수업 git 서버에서 자료를 Pull 해서 준비 !

KPU
한국산업기술대학교
KOREA POLYTECHNIC UNIVERSITY

Lecture #8. 곡선 이동

2D 게임 프로그래밍

이대현 교수

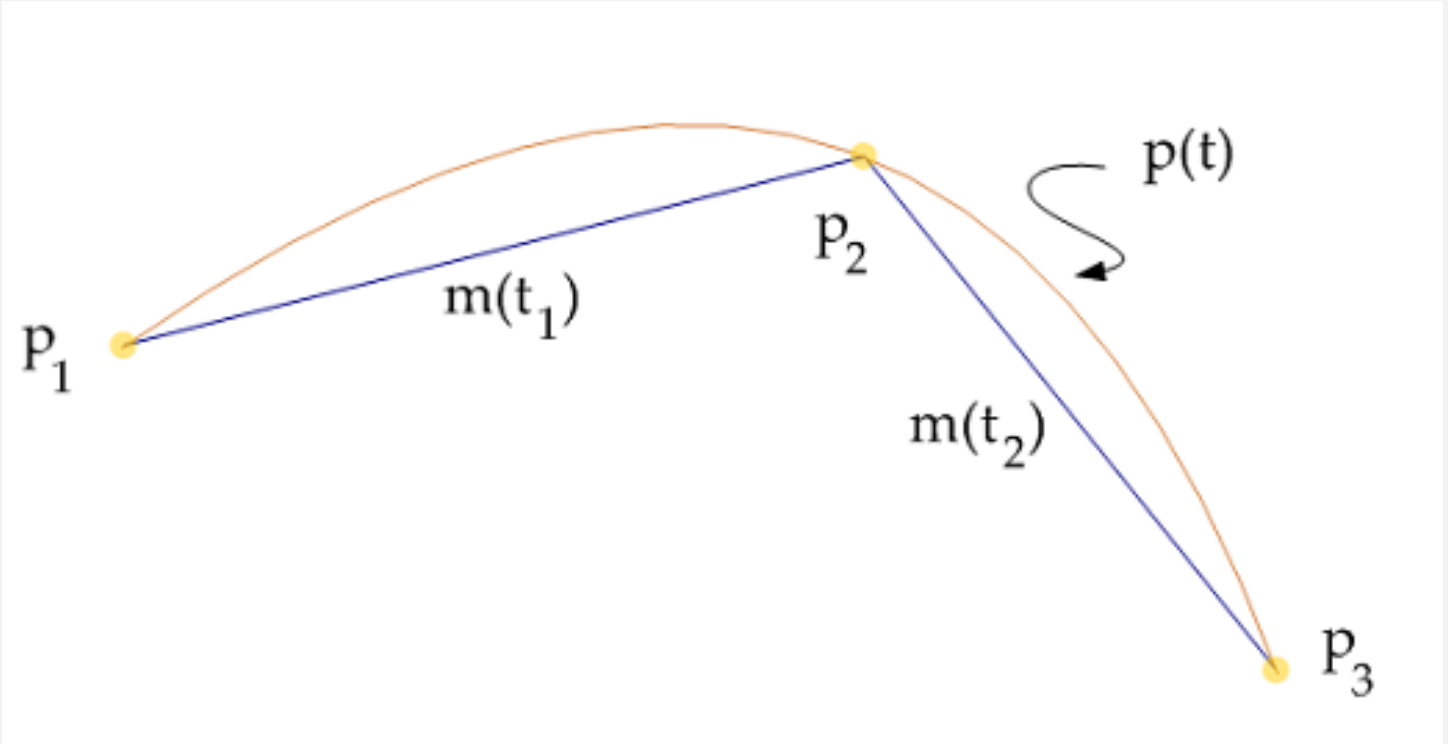
한국산업기술대학교
KOREA POLYTECHNIC UNIVERSITY

학습 내용

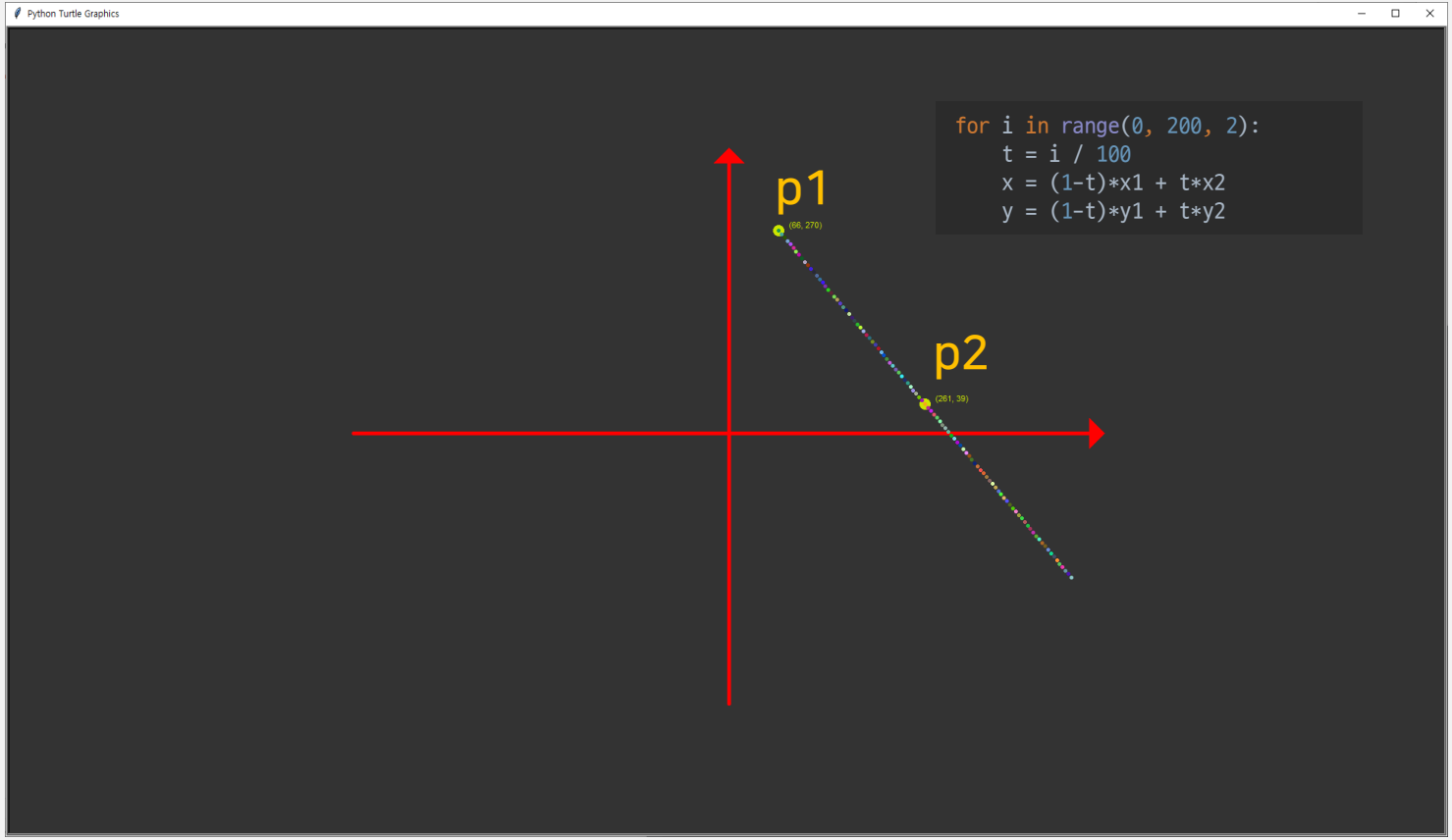
- 세 점 간의 부드러운 곡선 이동

- 네 점간의 부드러운 곡선 이동

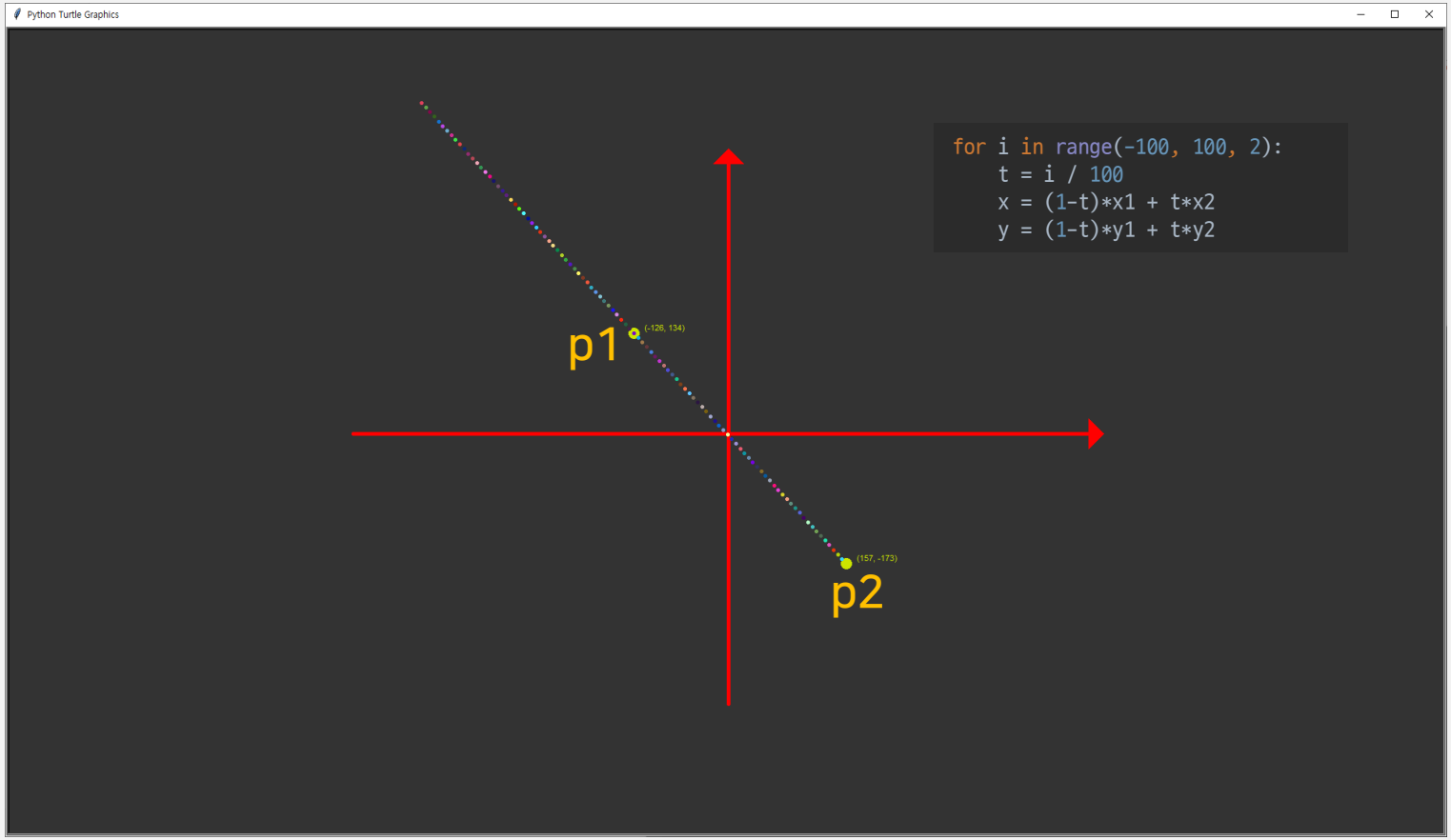
세 개의 점을 잇는 부드러운 곡선을 어떻게 그릴까?



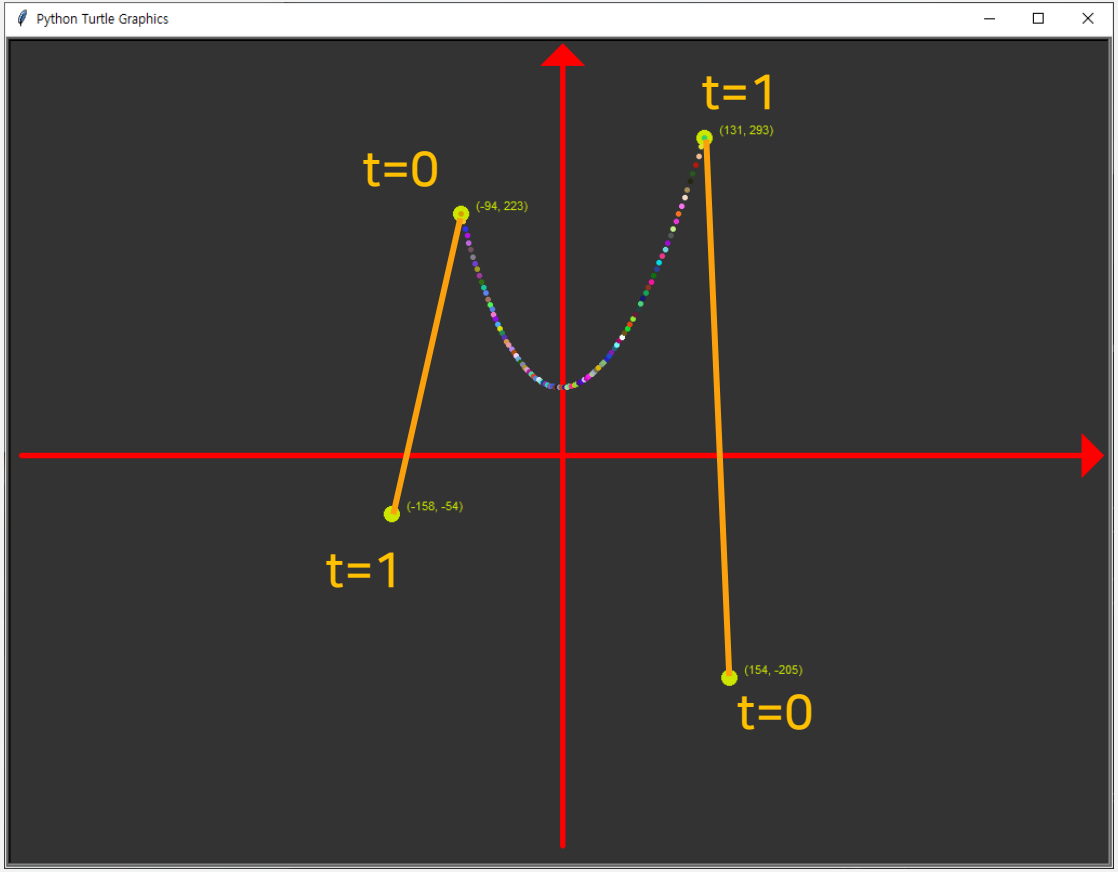
직선 그리기 (t = 0 ~ 2)



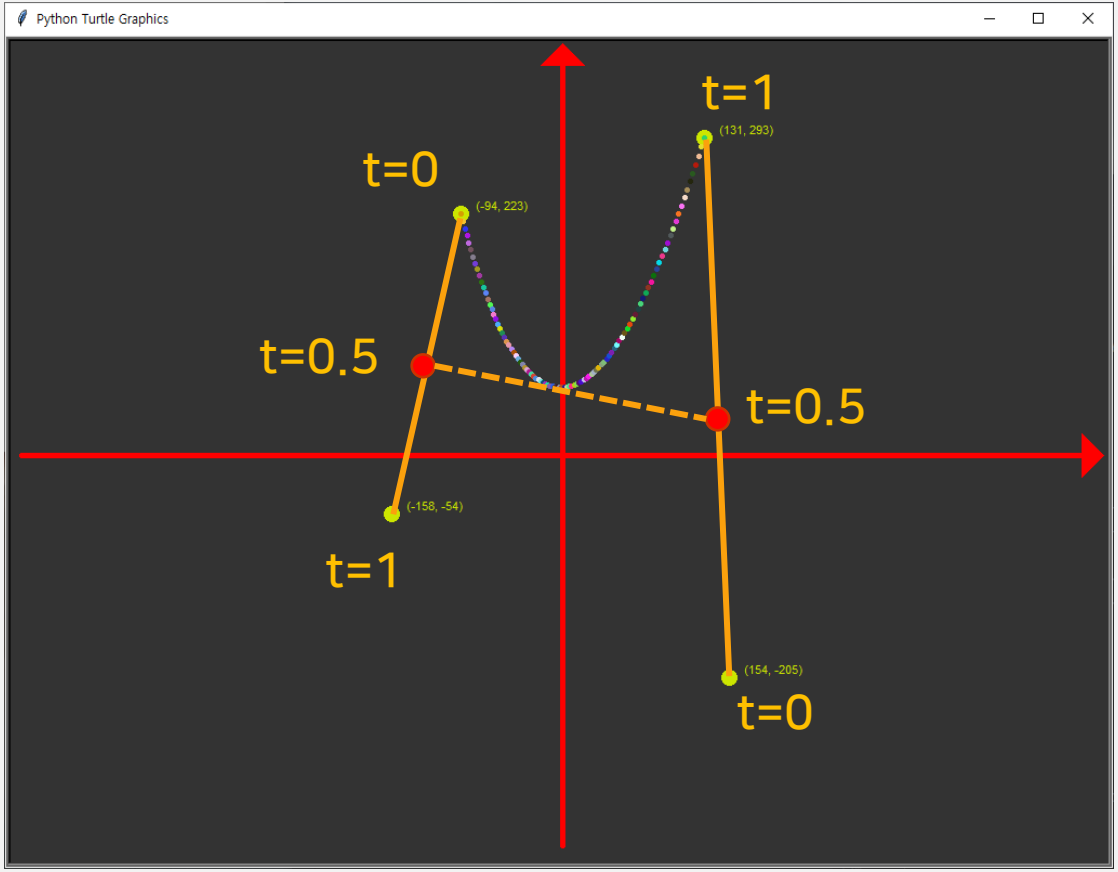
직선 그리기 (t = -1 ~ 1)



선분과 선분을 섞기

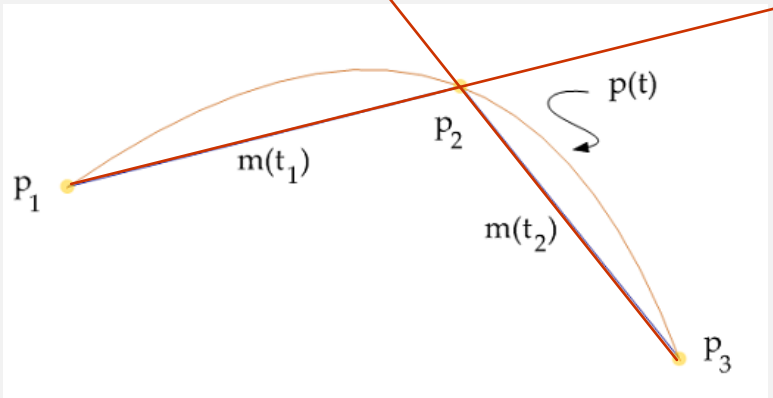


선분과 선분을 섞기



t=0
t2=-1

t=1
t1=2



t=0
t1=0

t=1
t2=1

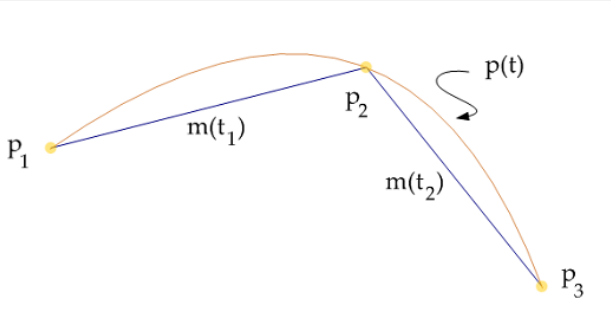
아이디어: 두개의 선분 m(t1)과 m(t2)를 1-t:t의 비율로 섞음.

$$m(t_1) = (1 - t_1)\ p_1 + t_1\ p_2$$

$$m(t_2) = (1 - t_2)\ p_2 + t_2\ p_3$$

$$p(t) = (1 - t)\ m(t_1) + t\ m(t_2)$$

$$= (1 - t)((1 - t_1)\ p_1 + t_1\ p_2)) + t\ ((1 - t_2)\ p_2 + t_2\ p_3)$$



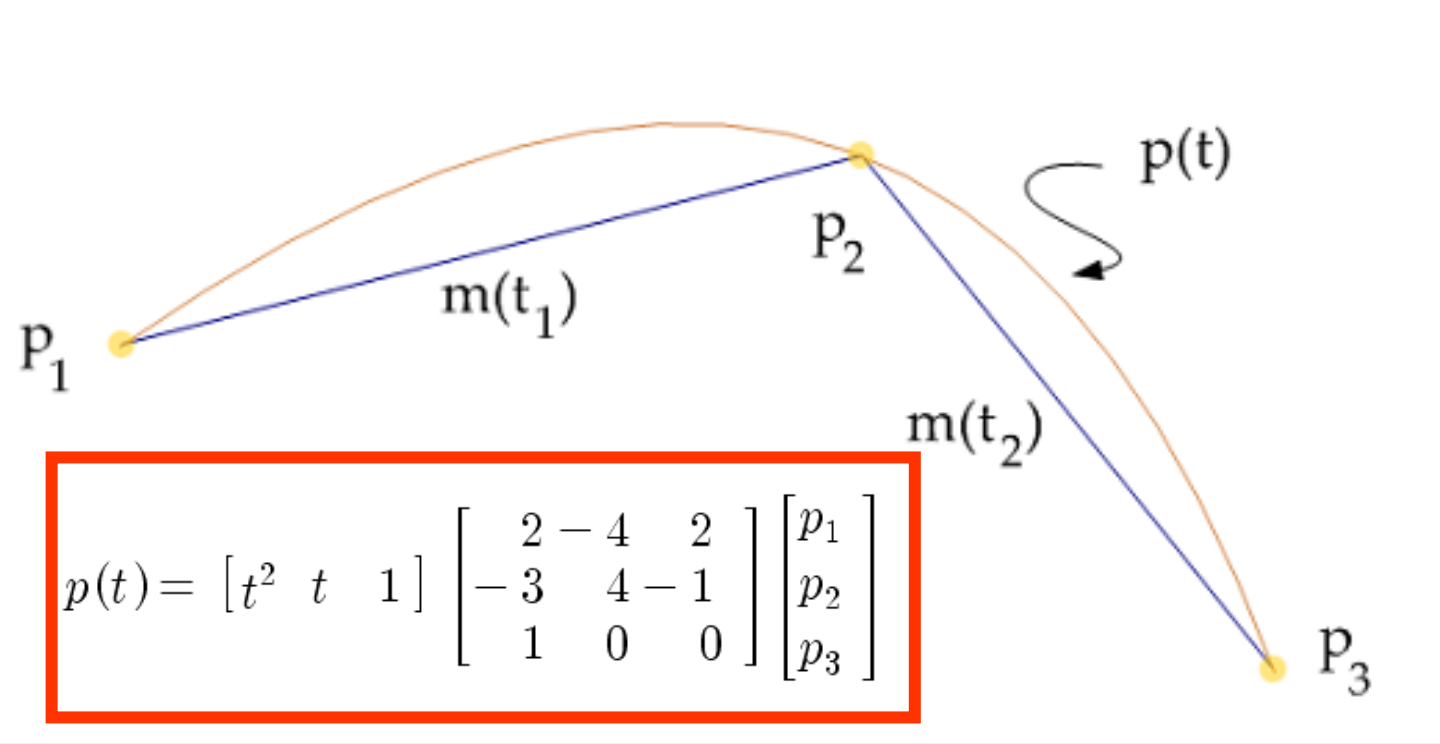
m(t1)에서 t1 = 0 인 점 p1은 p(t)로 볼 때 t = 0인 점

m(t1)에서 t1 = 1 인 점 p2는 p(t)에서 t = 1/2 인 중간점으로 간주

m(t2)에서 t2 = 0 인 점 p2는 t = 1/2

따라서 t1 = 2t, t2 = 2t -1.

카디날 스플라인(Cardinal Spline)

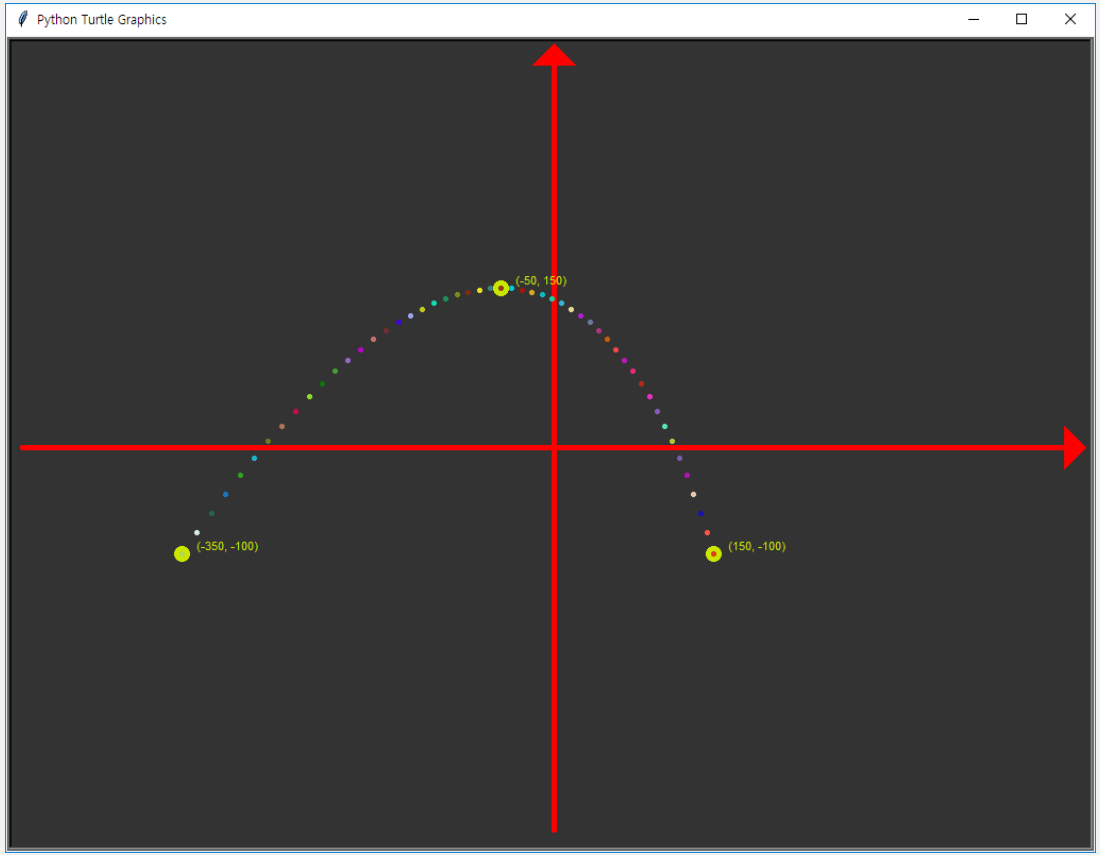


$$p(t) = \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

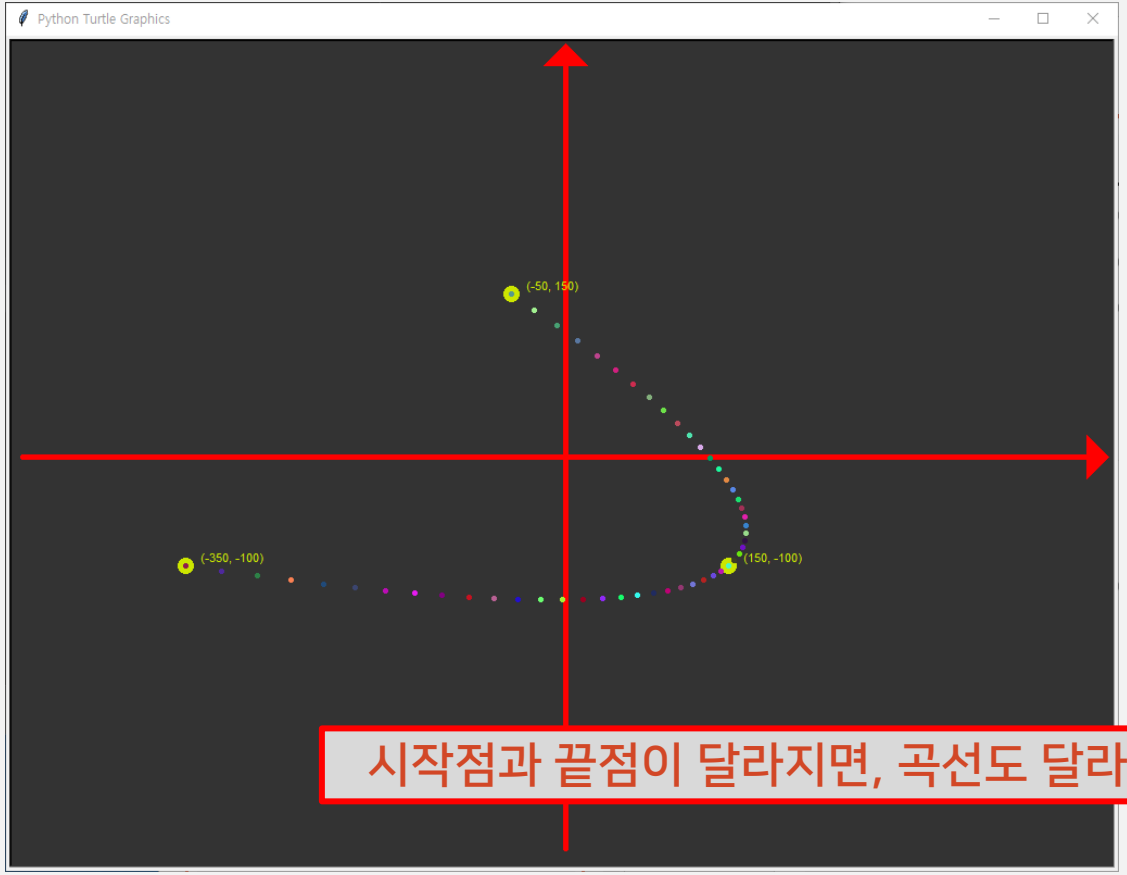
curve.py


```
def draw_curve_3_points(p1, p2, p3):
    draw_big_point(p1)
    draw_big_point(p2)
    draw_big_point(p3)

    x1, y1 = p1; x2, y2 = p2; x3, y3 = p3

    for i in range(0, 100, 2):
        t = i / 100
        x = (2 * t ** 2 - 3 * t + 1) * x1 + (-4 * t ** 2 + 4 * t) * x2 + (2 * t ** 2 - t) * x3
        y = (2 * t ** 2 - 3 * t + 1) * y1 + (-4 * t ** 2 + 4 * t) * y2 + (2 * t ** 2 - t) * y3
        draw_point((x, y))

    draw_point(p3)
```

draw_curve_3_points((-350, -100), (-50, 150), (150, -100))

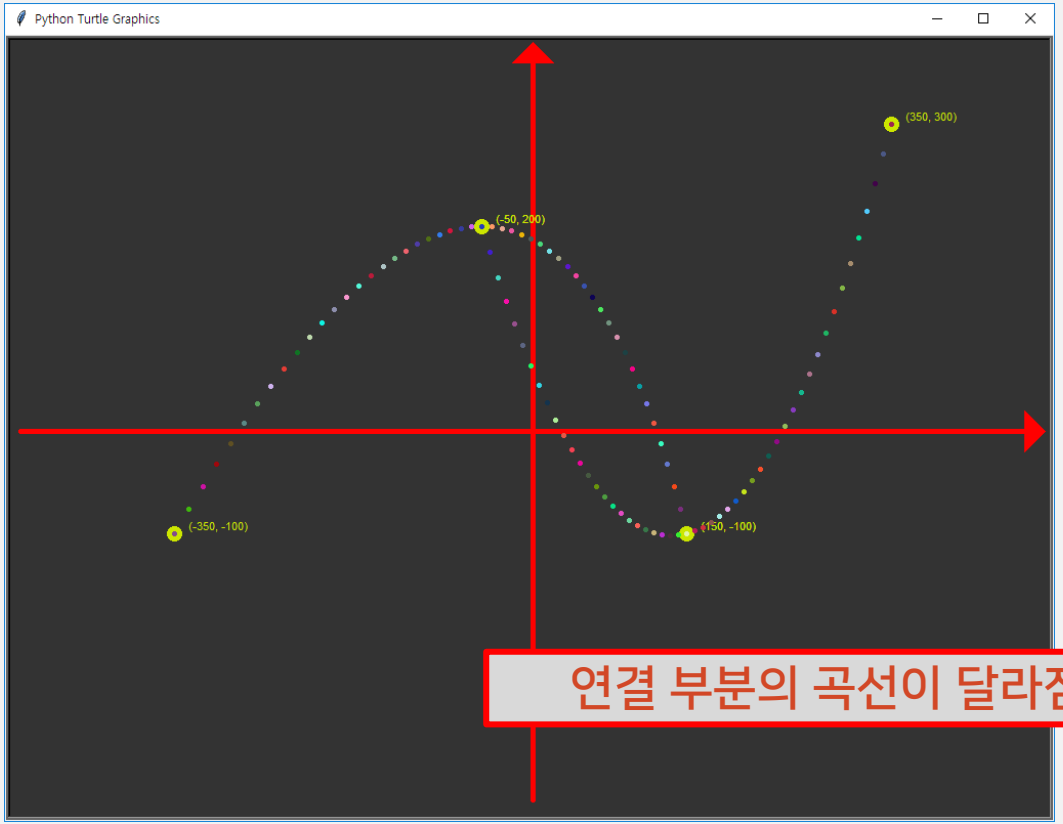


draw_curve_3_points((-50, 150), (150, -100), (-350, -100))



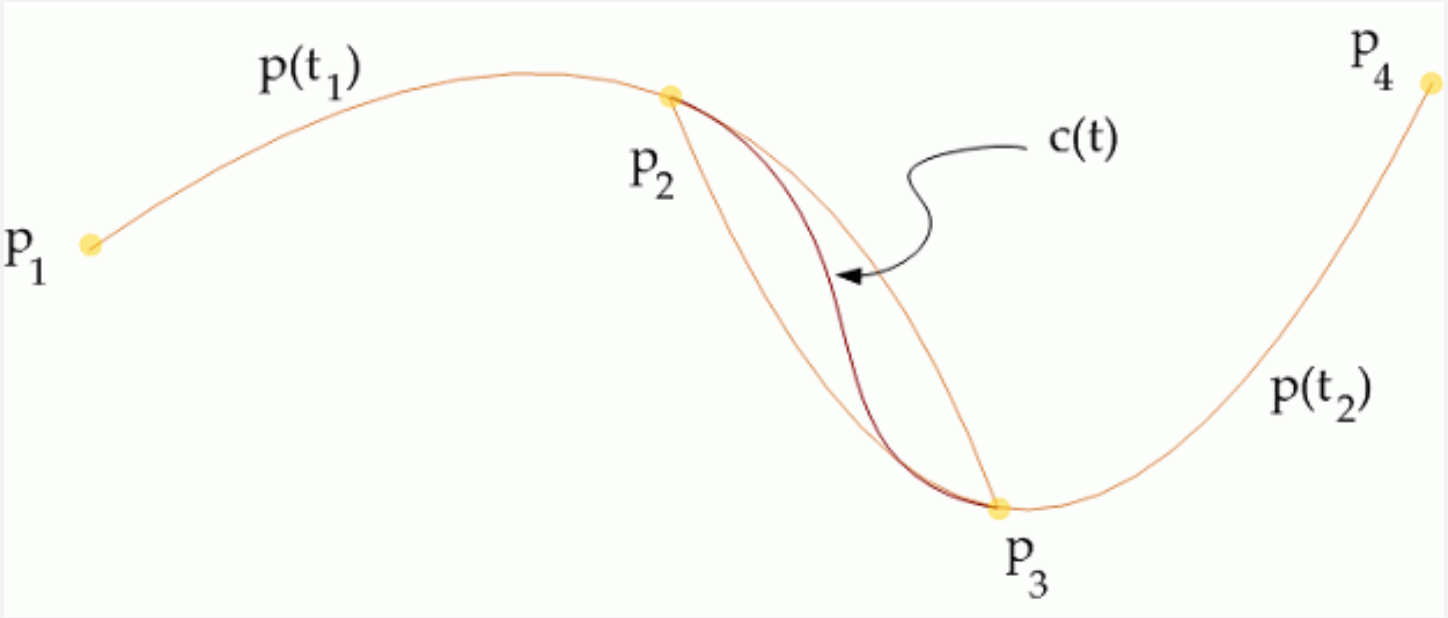
연속되는 점들 사이의 이동

draw_curve_3_points((-350, -100), (-50, 200), (150, -100))
draw_curve_3_points((-50, 200), (150, -100), (350, 300))



네 점을 연결하는 부드러운 곡선은?

아이디어: 두개의 곡선 p(t1)과 p(t2)를 1-t:t의 비율로 섞음.

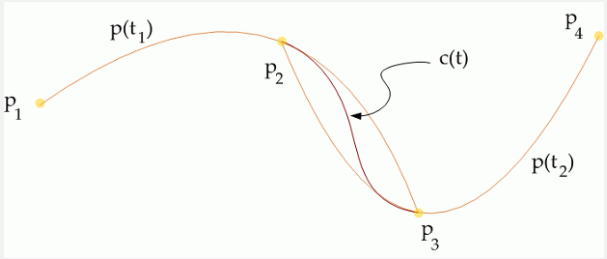


$$c(t) = (1-t)p(t_1) + tp(t_2)$$

$$= \frac{1}{2}\left((-t^3+2t^2-t)p_1 + (3t^3-5t^2+2)p_2 + (-3t^3+4t^2+t)p_3 + (t^3-t^2)p_4\right)$$

$$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \qquad (0 \le t \le 1)$$

$$= T \; M_c \; P$$



곡선 c(t)는 p2-p3 구간에서만 정의됨.

점 네개가 주어질 때, 가운데 두점 사이에서만 정의됨.

curve.py


```
def draw_curve_4_points(p1, p2, p3, p4):
    draw_big_point(p1); draw_big_point(p2); draw_big_point(p3); draw_big_point(p4)

    # draw p1-p2
    for i in range(0, 50, 2):
        t = i / 100
        x = (2*t**2-3*t+1)*p1[0]+(-4*t**2+4*t)*p2[0]+(2*t**2-t)*p3[0]
        y = (2*t**2-3*t+1)*p1[1]+(-4*t**2+4*t)*p2[1]+(2*t**2-t)*p3[1]
        draw_point((x, y))
    draw_point(p2)

    # draw p2-p3
    for i in range(0, 100, 2):
        t = i / 100
        x = ((-t**3 + 2*t**2 - t)*p1[0] + (3*t**3 - 5*t**2 + 2)*p2[0] + (-3*t**3 + 4*t**2 + t)*p3[0] + (t**3 - t**2)*p4[0])/2
        y = ((-t**3 + 2*t**2 - t)*p1[1] + (3*t**3 - 5*t**2 + 2)*p2[1] + (-3*t**3 + 4*t**2 + t)*p3[1] + (t**3 - t**2)*p4[1])/2
        draw_point((x, y))
    draw_point(p3)

    # draw p3-p4
    for i in range(50, 100, 2):
        t = i / 100
        x = (2*t**2-3*t+1)*p2[0]+(-4*t**2+4*t)*p3[0]+(2*t**2-t)*p4[0]
        y = (2*t**2-3*t+1)*p2[1]+(-4*t**2+4*t)*p3[1]+(2*t**2-t)*p4[1]
        draw_point((x, y))
    draw_point(p4)
```

p1-p2구간은 p1,p2,p3로부터 앞의 50%를 계산

P2-p3구간은 p1, p2, p3, p4로부터 계산

p3-p4구간은 p2, p3, p4로부터 뒤의 50%를 계산

draw_curve_4_points((-350, -100), (-50, 200), (150, -100), (350, 300))