컴퓨터애니메이션실습(CAL)

HW10_Uniform Cubic B-Splines

Self-scoring table

	P1	P2	P3	E1	E2	Total
Score	**1**	**1**	**1**	**1**	**1**	**5**

2018707068 김경환

KwangWoon University

Code Analysis:

먼저 이번에는 knot spacing(매듭 간격)을 Uniform하게 갖고 모든 curve segment를 cubic polynomial으로 표현하는 Basis Spline인 Uniform Cubic B-Spline을 수행한다.

해당 과제에서는 이를 위해서 먼저 8개의 Data Point를 미리 전역변수로 선언해 두었다.
그리고 Uniform Cubic B-Spline을 수행할 controlPoints를 설정해준다.
이는 Endpoint Interpolation을 위해 comtrolPoint의 개수인 nControlPoints를 Data point의 개수 + endpoint에서의 repetition 횟수 * 2로 설정한다.
또한 controlPoints를 저장할 때는 endpoint가 아닌 지점부터 Data point를 저장해야 하므로 0에서부터 시작하는 배열에 repetition의 index에서 repetition + N - 1까지 Data point를 저장하고 0 ~ repetition, repetition * 2 + N - 1까지 endpoint 값을 반복해서 저장해준다.

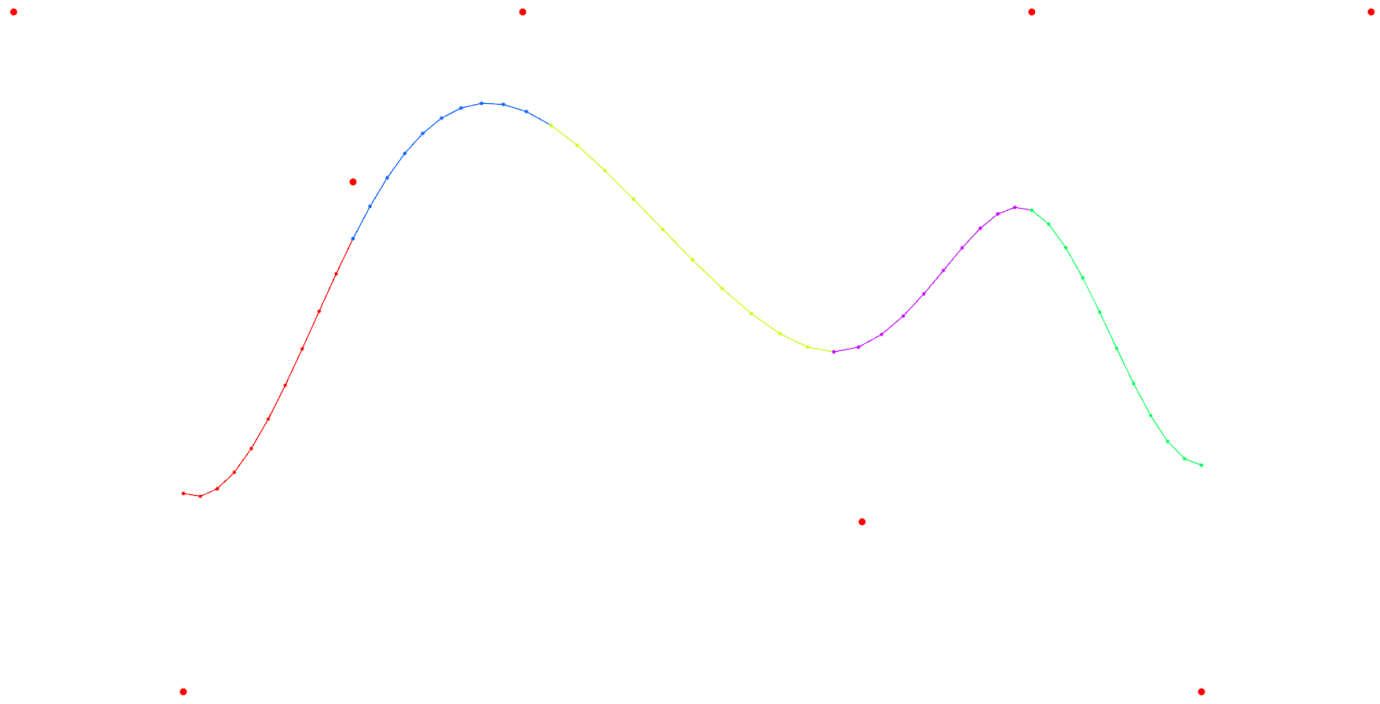
해당 controlPoints를 기반으로 Curve Segment를 그릴 때는 인접한 Curve Segment는 가시성을 높이기 위해 보색으로 지정해주었다.
이는 그려야할 Curve Segment의 수인 nControlPoints - 3으로 180도를 분할한 값을 가장 왼쪽 Curve Segment부터 hsv의 angle로 누적합을 부여하고 짝수, 홀수에 따라 180도씩 차이나게 하여 구현하였다.
이렇게 부여한 각도에 따른 HSV값을 RGB값으로 변환하고 해당 RGB로 Curve Segment를 그린다.

위에서 지정한 controlPoints 중 B-Spline을 통해 Curve Segment를 그릴 Control Point 4개를 순차적으로 지정하고, 해당 Control Point 4개와 0에서 1까지 균일하게 나눠진 t값을 순차적으로 B-Spline의 point 계산을 사용한다.
이러한 B-Spline의 point를 계산하는 방법은 t값을 Bell-shaped basis function에 대입하고 Control Point 4개와 곱하여 값을 얻어낸다.

Curve Segment를 그릴 때는 이렇게 얻어낸 B-Spline 위의 Point값을 GL_LINE_STRIP으로 그려낸다.
또한 Draw Control Polygon을 수행할 때는 B-Spline 위의 Point를 계산할 때 사용한 Control Point 4개를 GL_LINE_STIPPLE과 GL_LINE_STRIP을 사용하여 그린다.

Thick curve segment을 수행할 때는 bool type의 ctrlPolygonDrawingEnabled flag를 사용하여 해당 flag가 true이고 현재 그리고자 하는 Curve Segment와 Control Polygon의 index가 동일할 때 glLIneWidth를 키우는 방식으로 구현하였다.

Practice 01. Computing/drawing a uniform cubic B-spline curve:

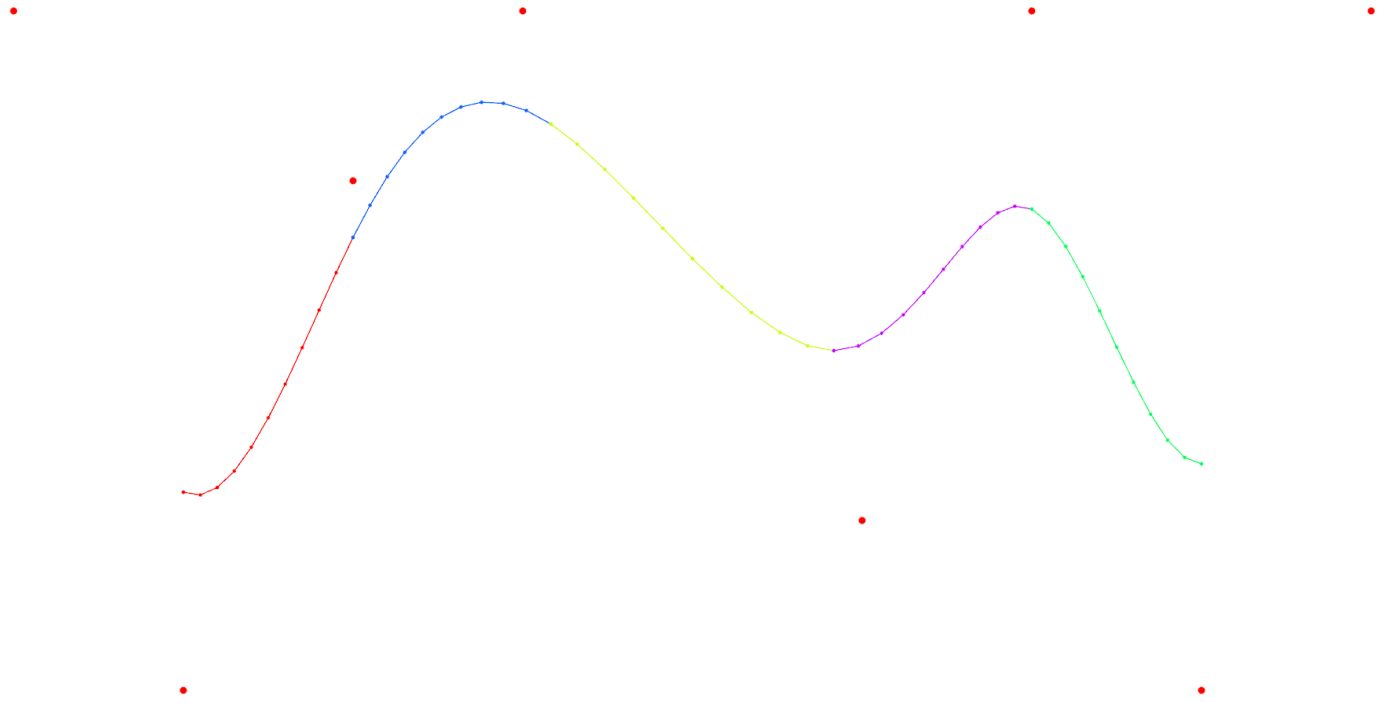


Unifrom Cubic B-Spline Curve을 계산하고 그린 모습이다.

8개의 Data Point에 대해 5(8-3)개의 Curve Segment가 그려진 것을 볼 수 있다.

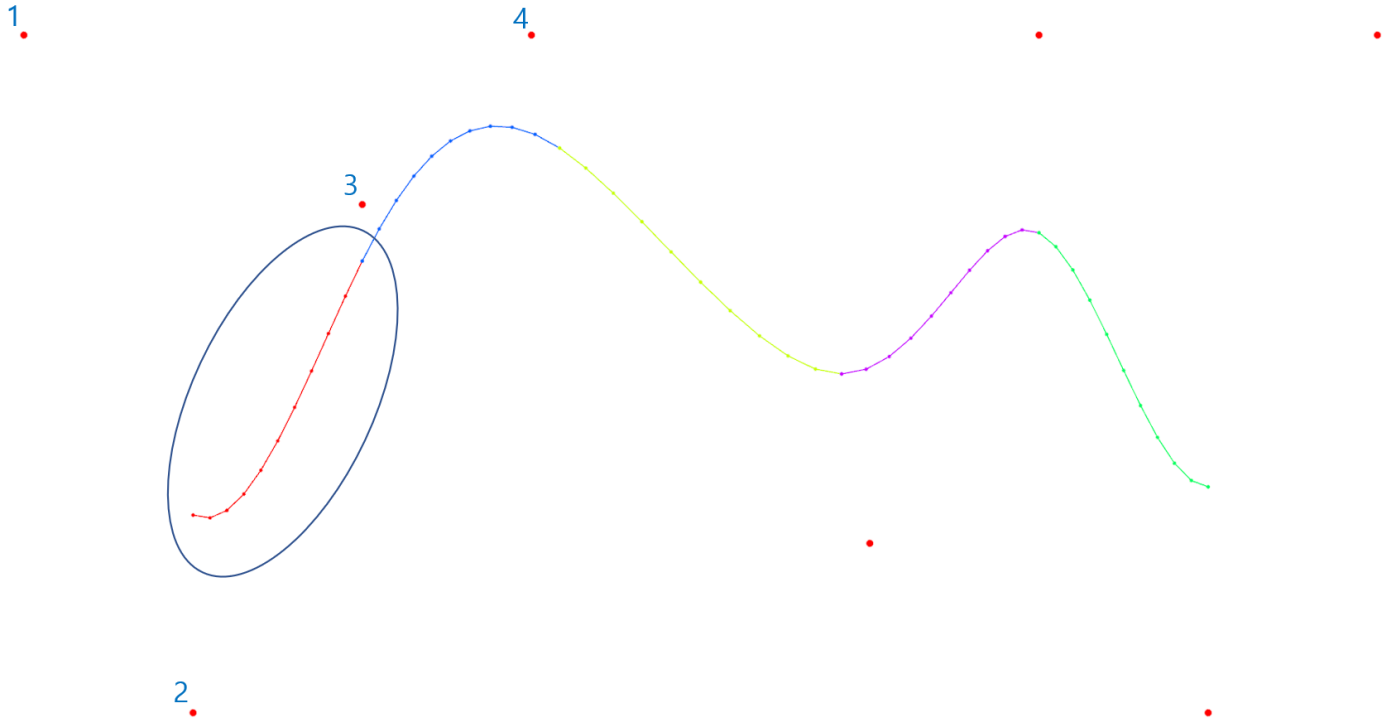
모든 Curve Segment의 Sample Point는 0에서 1 사이의 값으로 1을 Sub Segment의 수로 나눈 동일한 값으로 Sampling 되어있다.

Practice 02. Employing a unique color for each curve segment using the HSV color space:



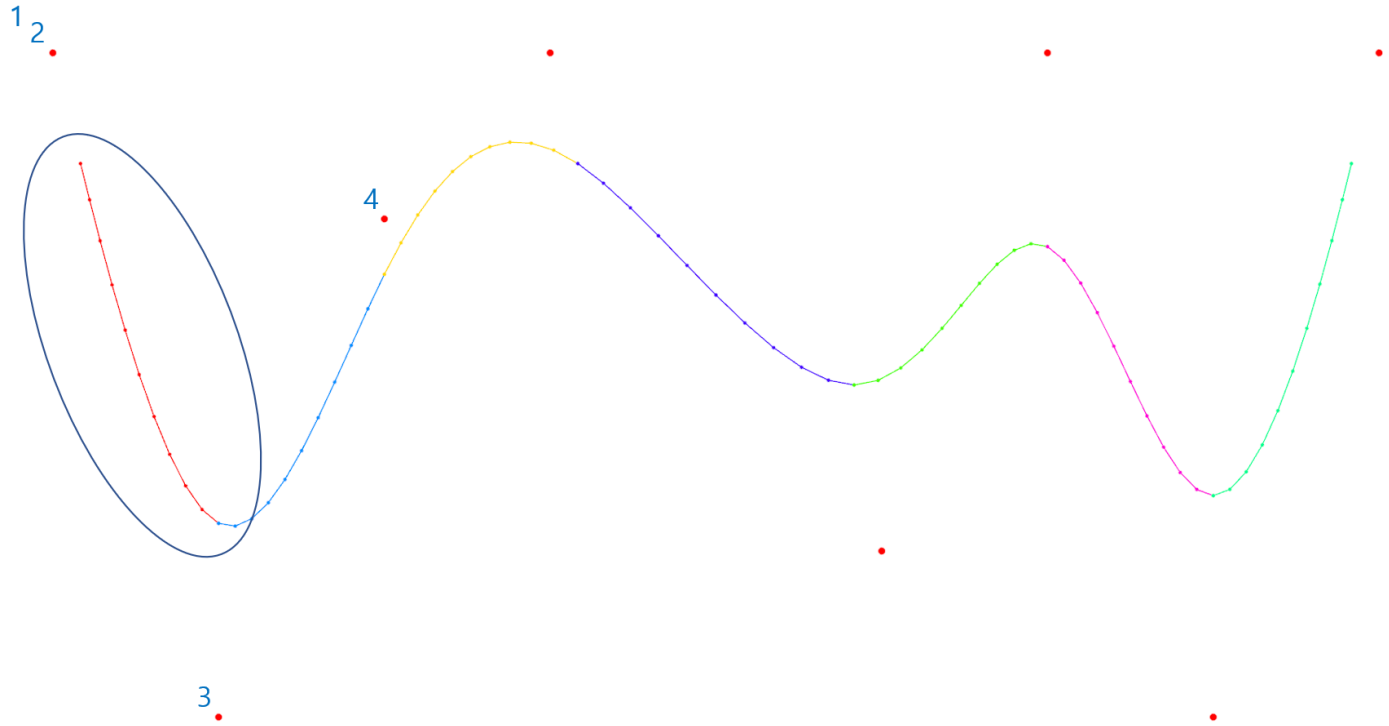
인접한 Curve Segment를 서로 보색으로 지정하여 구분하기 편하게 한 것을 확인할 수 있다.

Practice 03. Demonstrating the endpoint interpolation:



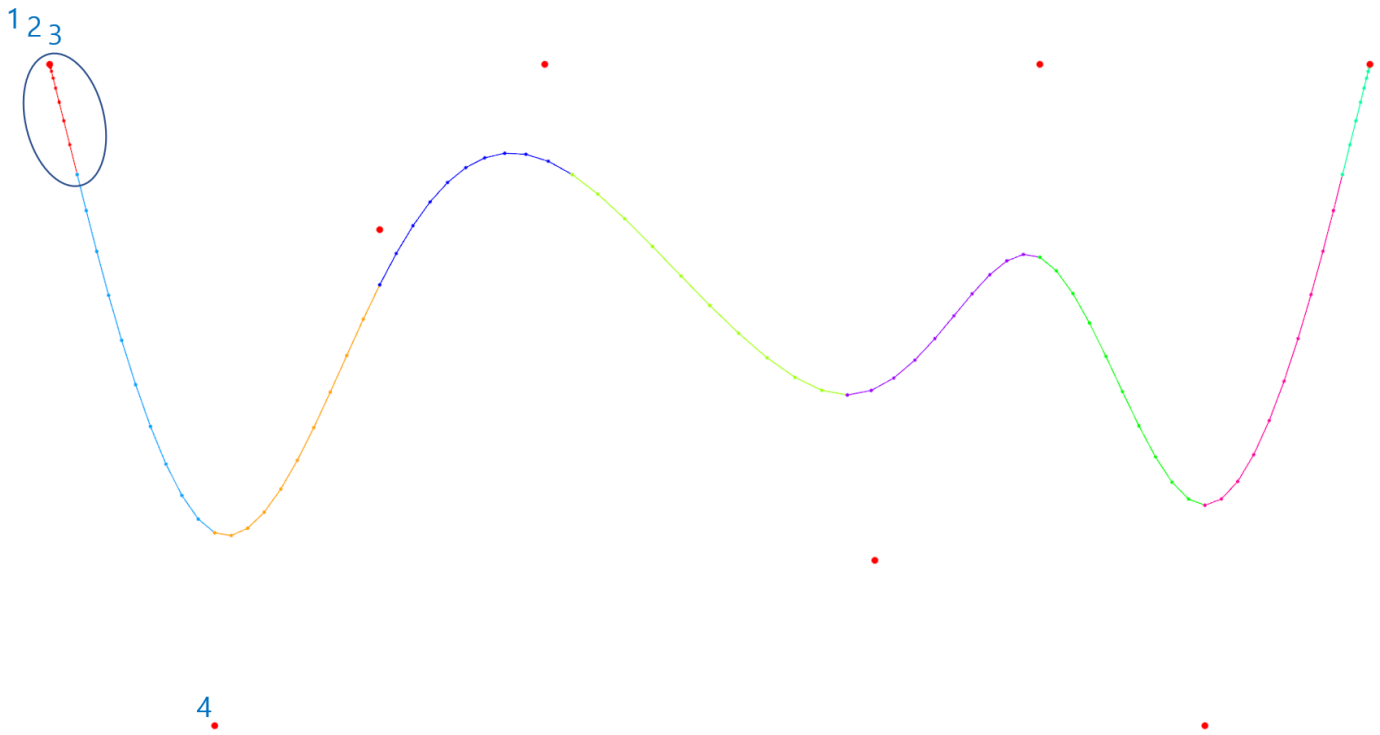
Repition을 0으로 전혀 End Point을 전혀 반복하지 않은 모습이다.

1~ 4로 표시된 4개의 Data Point를 Control Point로 사용해 파란색 타원안의 빨간색의 Curve를 그렸다.



Repition을 1으로 하여 End Point의 값을 controlPoints에 가장 앞과 뒤에 한 번씩 더 저장한 모습이다.

1~ 4로 표시된 4개의 Data Point를 Control Point로 사용해 파란색 타원안의 빨간색의 Curve를 그렸다.



Repition을 2로 하여 End Point의 값을 controlPoints에 가장 앞과 뒤에 두 번씩 더 저장한 모습이다.

1~ 4로 표시된 4개의 Data Point를 Control Point로 사용해 파란색 타원안의 빨간색의 Curve를 그렸다.

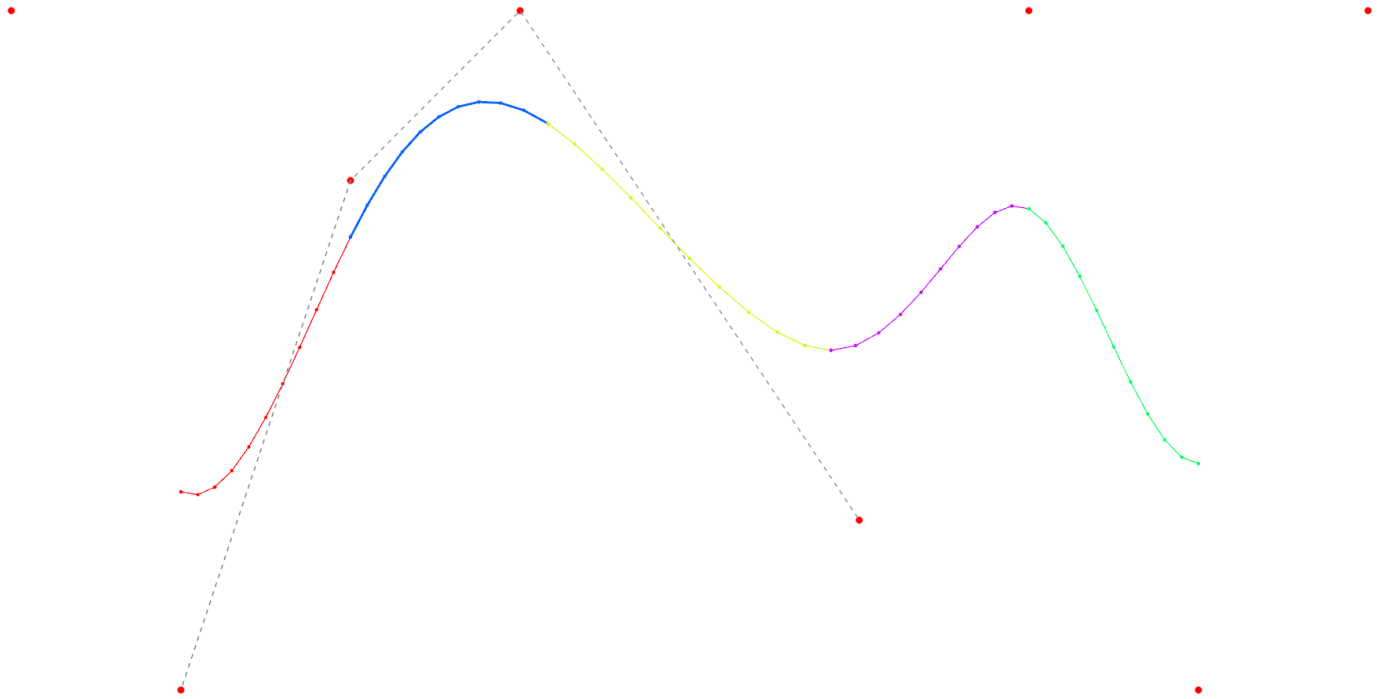
End Point를 2번 Repition하여 Curve를 그렸을 때 Data Point의 End Point를 Interpolation하듯 해당 Point를 지나가는 Curve가 그려진 것을 볼 수 있다.

이는 Bell-shpaed basis function이 Curve의 End Point에서는 4개의 Control Point 중 3개 많으로 그려지기 때문이다. 즉, 위 사진에서 End Point를 그릴 때는 Curve의 4번째 Control Point를 사용하지 않는다.

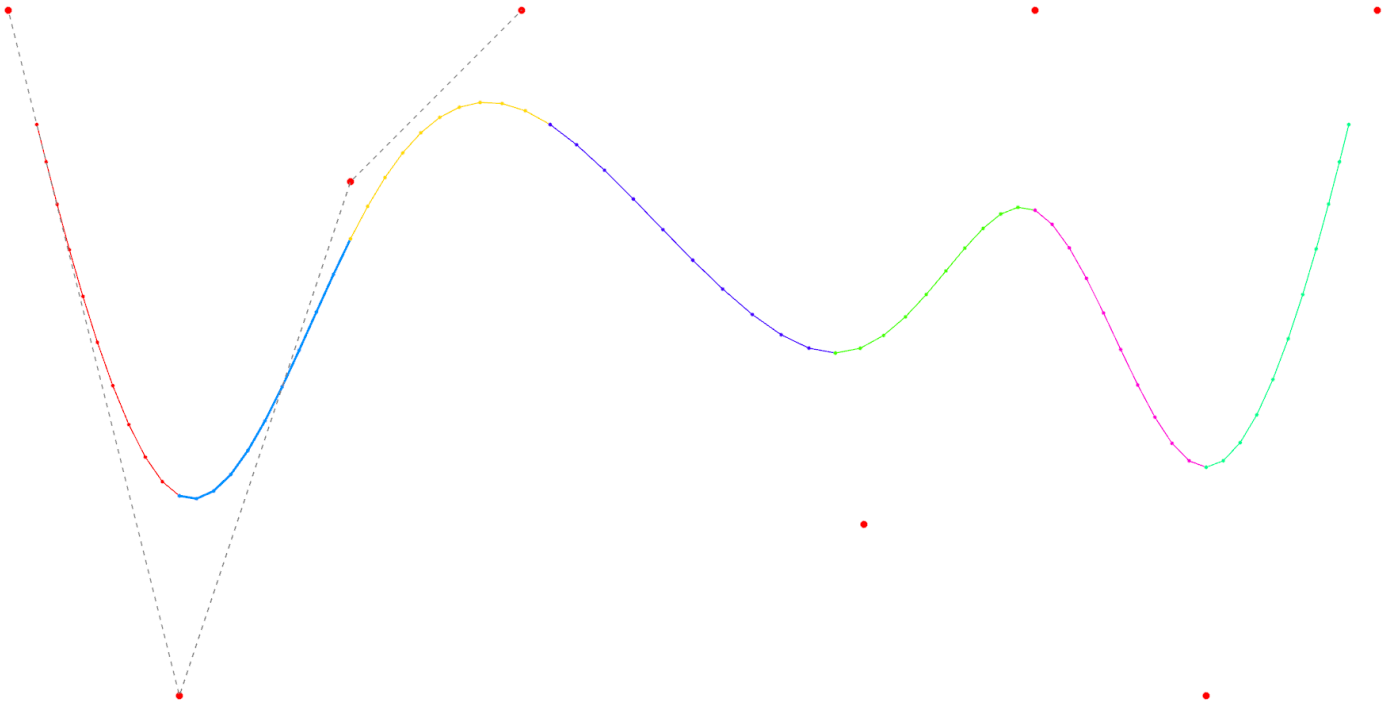
이는 2번을 Repition하면서 End Point가 2번 겹쳐지면서 총 3개가 한 곳에 위치하기 때문이다.

Exercise 01. Draw the polyline of the 4 control points for a specific curve segment:

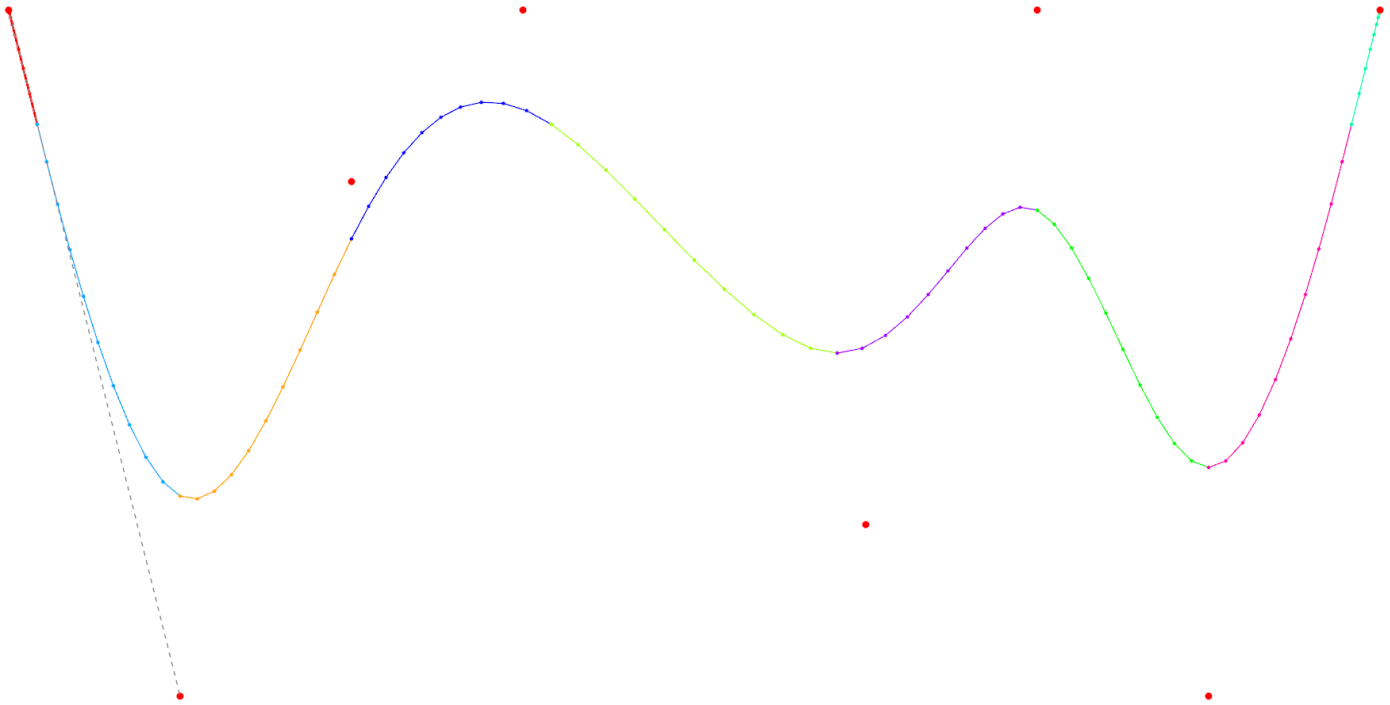
Exercise 02. Draw the corresponding curve segment with thick line segments:



8개의 Data Point를 1부터 8까지 번호를 매겼을 때 2, 3, 4, 5번째 Data Point를 Control Point로 하여 Control Polygon이 그려진 것을 볼 수 있다. 또한 그에 해당하는 파란색의 Curve Segment가 다른 Curve Segment에 비해 더 두껍게 그려진 것도 확인할 수 있다.



위 사진은 1번의 End Point Repition을 진행하여 총 8+2*1개의 Data Point에서 1번부터 번호를 매겼을 때 2, 3, 4, 5번째 Data Point를 Control Point로 하여 Control Polygon이 그려진 것을 볼 수 있다. 또한 그에 해당하는 하늘색의 Curve Segment가 다른 Curve Segment에 비해 더 두껍게 그려진 것도 확인할 수 있다.



위 사진은 2번의 End Point Repition을 진행하여 총 8+2*2개의 Data Point에서 1번부터 번호를 매겼을 때 1, 2, 3, 4번째 Data Point를 Control Point로 하여 Control Polygon이 그려진 것을 볼 수 있다. 또한 그에 해당하는 빨간색의 Curve Segment가 다른 Curve Segment에 비해 더 두껍게 그려진 것도 확인할 수 있다.