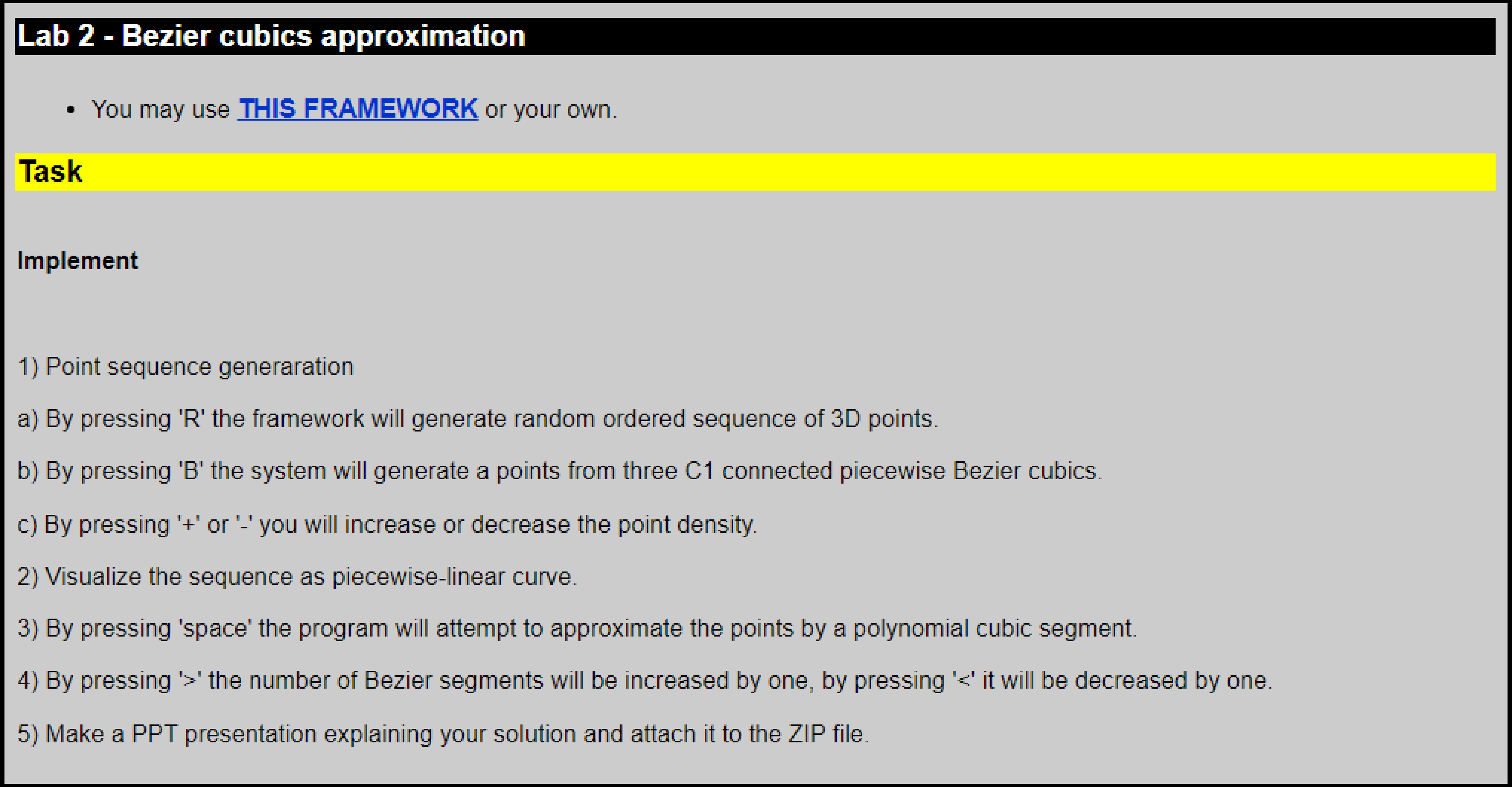
CS590

Lab 02

Devashri Vagholkar

The Problem



Lab 2 - Bezier cubics approximation

- You may use **THIS FRAMEWORK** or your own.

Task

Implement

1) Point sequence generaration

a) By pressing 'R' the framework will generate random ordered sequence of 3D points.

b) By pressing 'B' the system will generate a points from three C1 connected piecewise Bezier cubics.

c) By pressing '+' or '-' you will increase or decrease the point density.

2) Visualize the sequence as piecewise-linear curve.

3) By pressing 'space' the program will attempt to approximate the points by a polynomial cubic segment.

4) By pressing '>' the number of Bezier segments will be increased by one, by pressing '<' it will be decreased by one.

5) Make a PPT presentation explaining your solution and attach it to the ZIP file.

Navigate the program

- ‘r’ – random points
- ‘p’- Bezier curve control points
- ‘b’-Bezier curve points(3 C1 connected)
- ‘c’-Bezier curve segments
- ‘+’-Bezier curve point density increase
- ‘-’- Bezier curve point density decrease
- ‘space’- approximate curve segment
- ‘q’-approximate curve control points
- ‘>’-increase no. of segments in approximate curve
- ‘<‘-decrease no. of segments in approximate curve
- ‘a’-approximate curve points

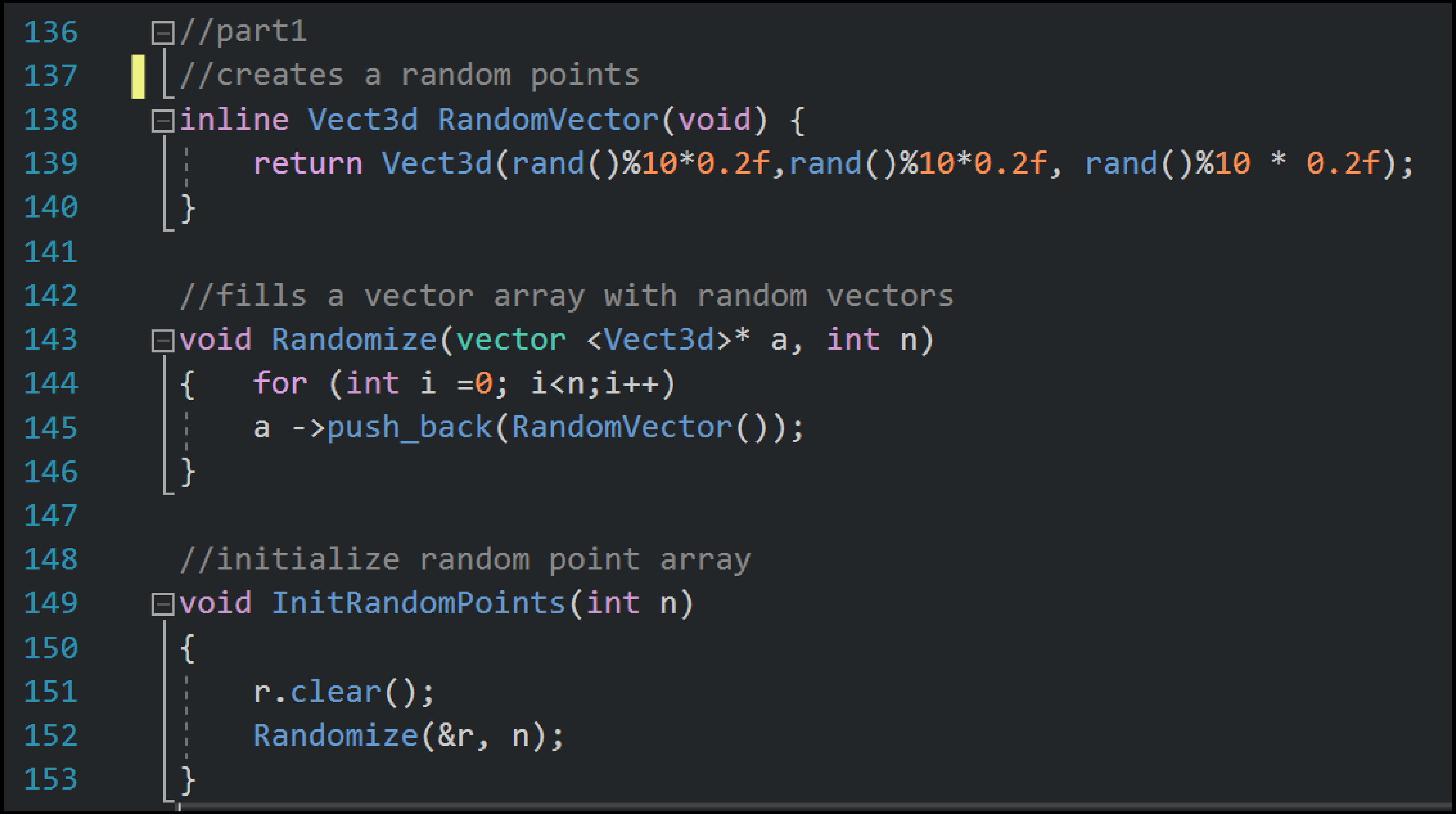
1. Point sequence generation

a. Press ‘r’ → generate random ordered sequence of 3D points

Pseudo code:

```
Global vector <Vect3d> r // to store random points
Main → InitRandomPoints(10) initialize array for random points
                              →Randomize→RandomVector→rand()


Kbd→case’r’
Render→RandomPointsFlag
```

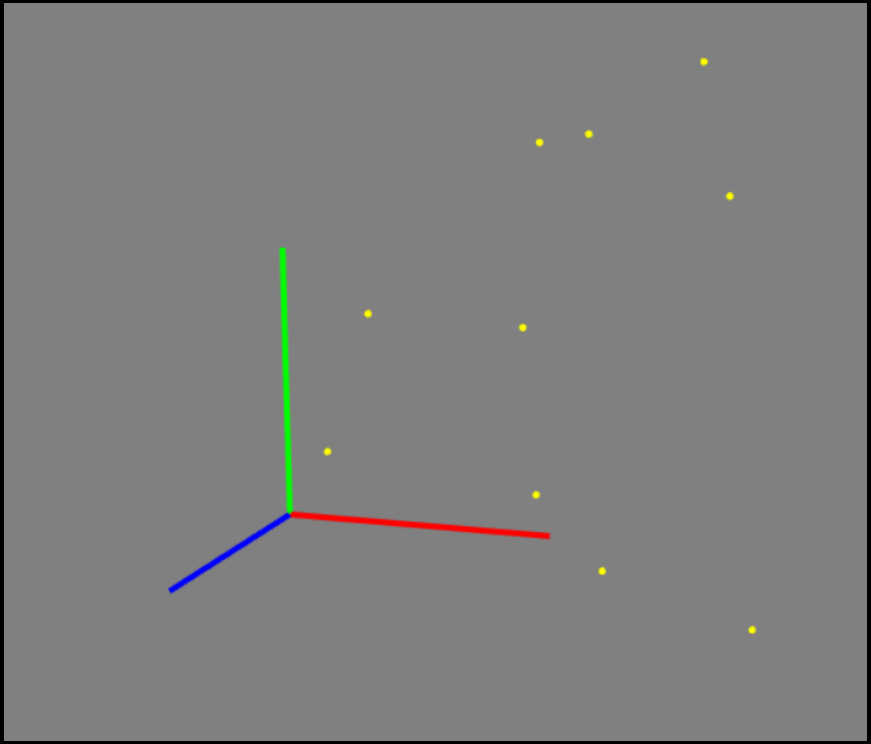
1. Point sequence generation



```
//part1
//creates a random points
inline Vect3d RandomVector(void) {
    return Vect3d(rand()%10*0.2f,rand()%10*0.2f, rand()%10 * 0.2f);
}

//fills a vector array with random vectors
void Randomize(vector <Vect3d>* a, int n)
{   for (int i =0; i<n;i++)
    a ->push_back(RandomVector());
}

//initialize random point array
void InitRandomPoints(int n)
{
    r.clear();
    Randomize(&r, n);
}
```

1. Point sequence generation

a. Press ‘r’ → generate random ordered sequence of 3D points



1. Point sequence generation

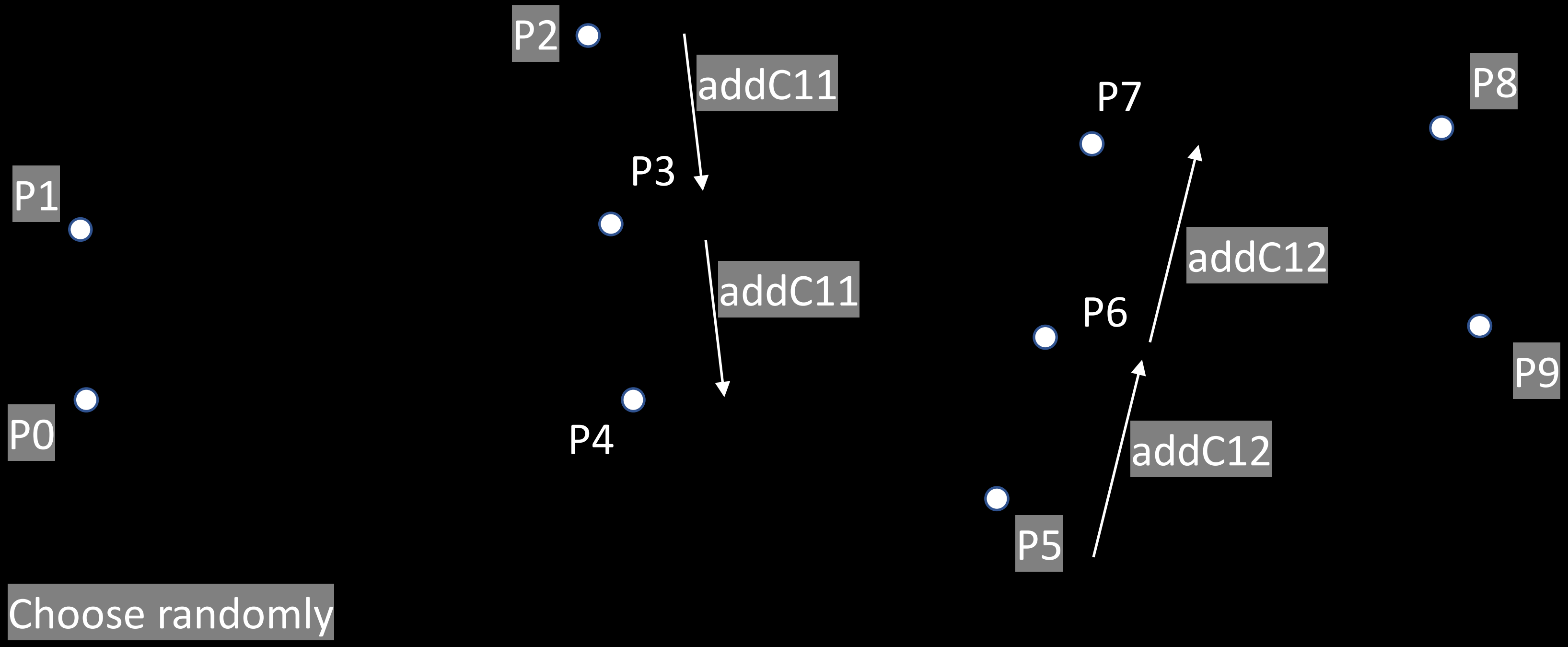
b. Press ‘b’→generate points for 3 peicewise C1 connected Bezier curves

Pseudo code:

```
Main→InitBezier()
          initializes array fpr control points
               →CreateBezierPoints()→Bezier() creates control pts for 3 C1 Bezier cubic
               →InitBezierCurve()initializes 3 arrays for each segment
                                        →CreateBezierCurve()  →C (1st segment)
                                                              →D (2nd segment)
                                                              →E (3rd segment)

Kbd→case ‘c’ show curve
     →case’b’ show Bezier segments
Render→BezierCurveFlag
         →BezierPointFlag
```

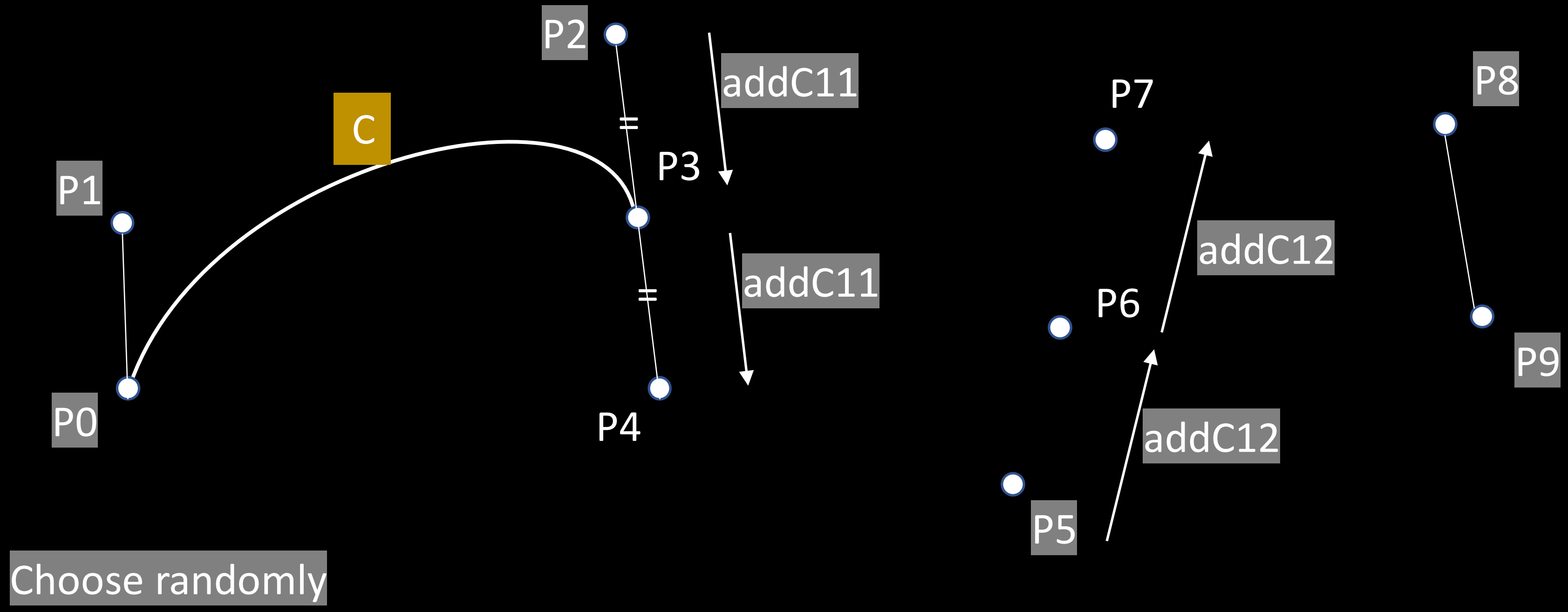
1. Point sequence generation

b. Press ‘b’→generate points for 3 peicewise C1 connected Bezier curves



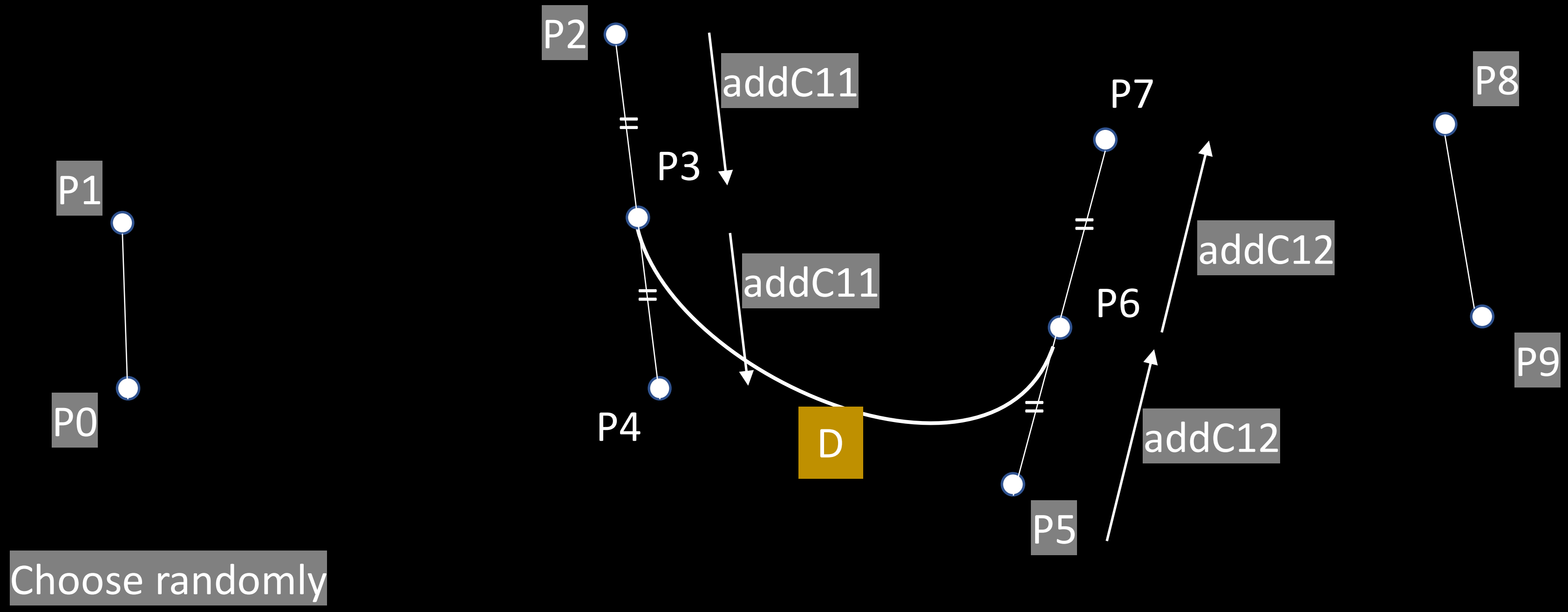
1. Point sequence generation

b. Press ‘b’→generate points for 3 peicewise C1 connected Bezier curves



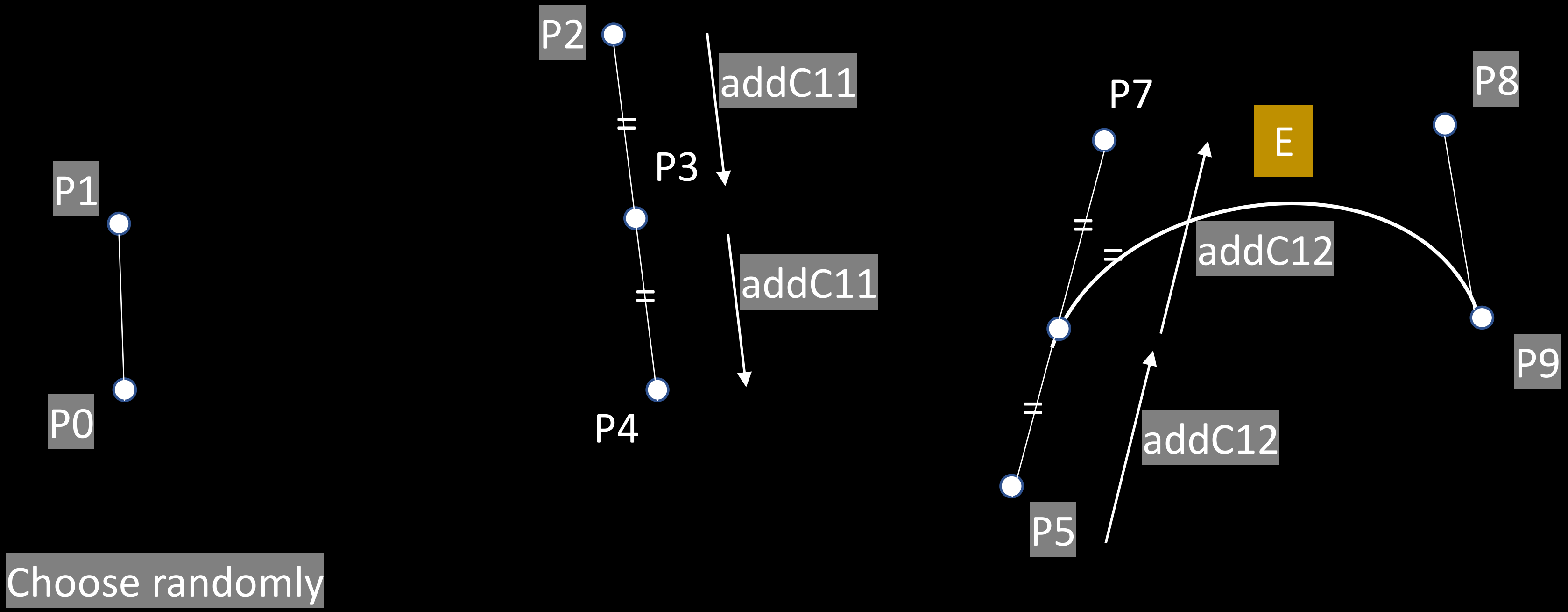
1. Point sequence generation

b. Press ‘b’→generate points for 3 peicewise C1 connected Bezier curves



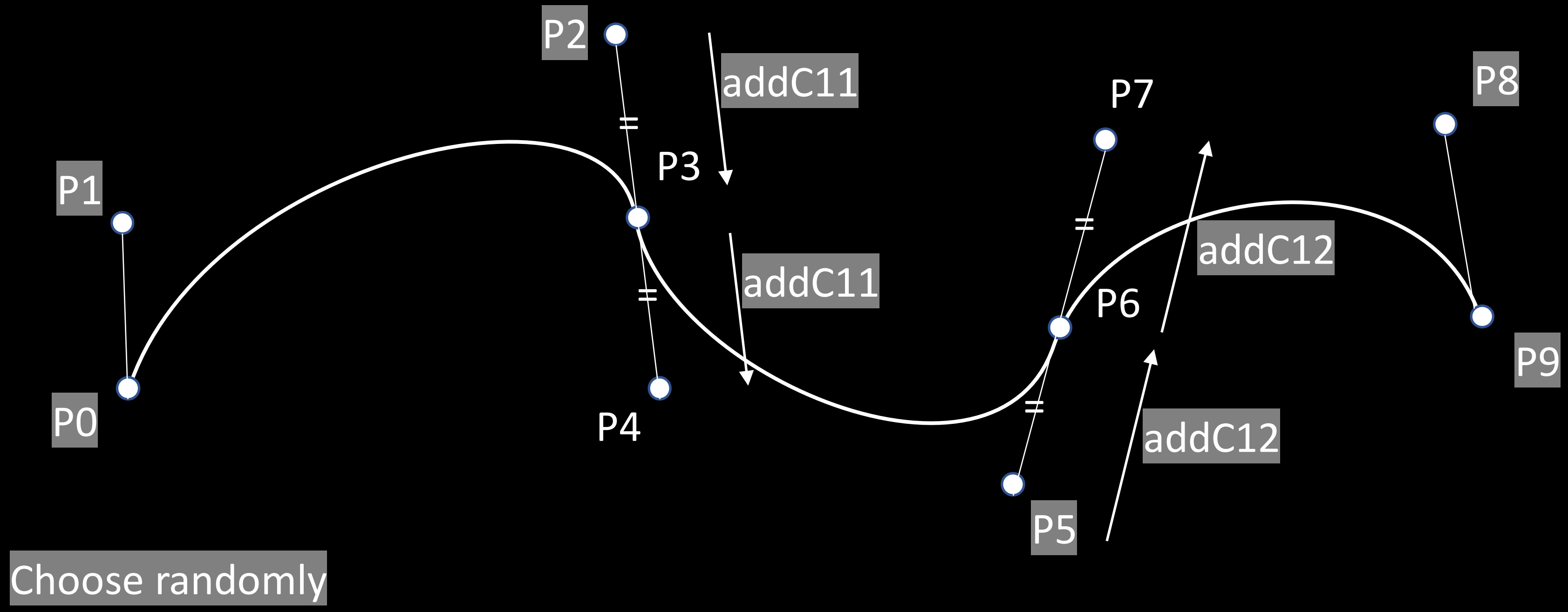
1. Point sequence generation

b. Press ‘b’→generate points for 3 peicewise C1 connected Bezier curves



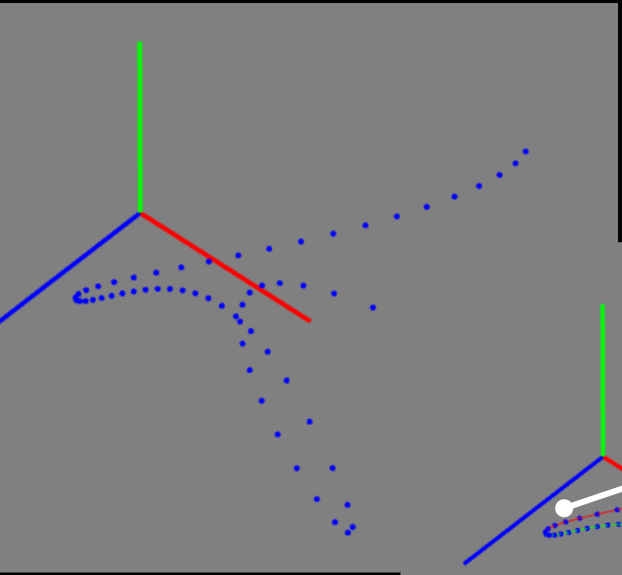
1. Point sequence generation

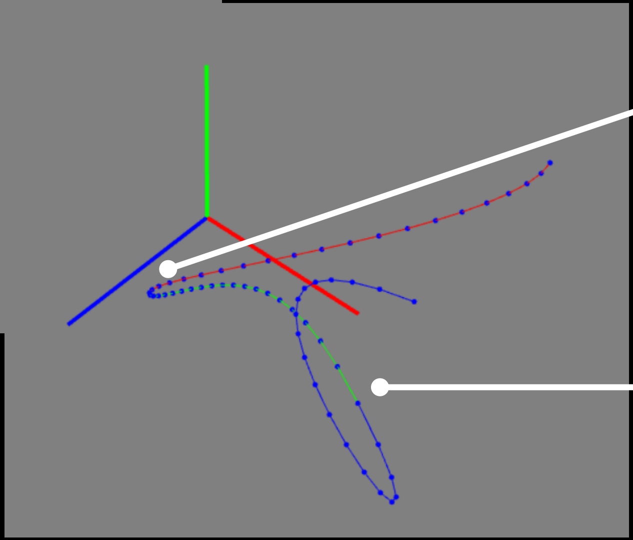
b. Press ‘b’→generate points for 3 peicewise C1 connected Bezier curves

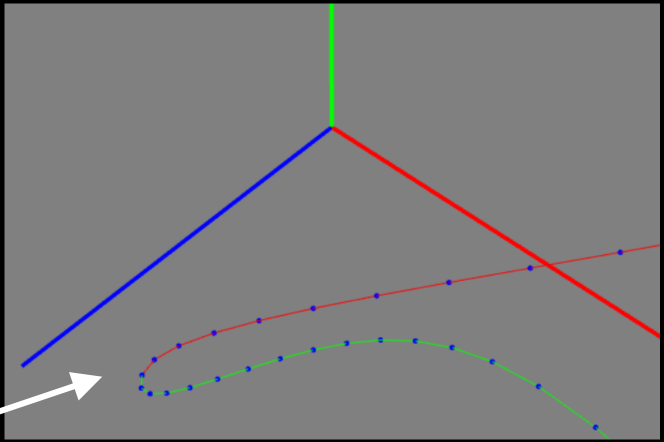


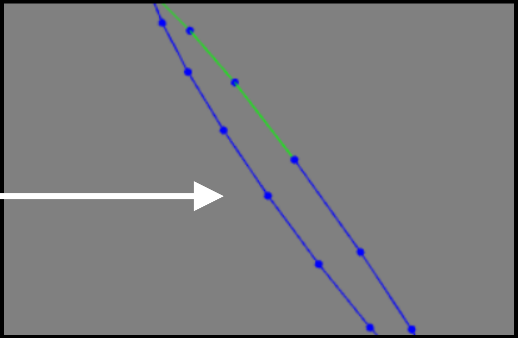
1. Point sequence generation

b. Press ‘b’→generate points for 3 peicewise C1 connected Bezier curves









1. Point sequence generation

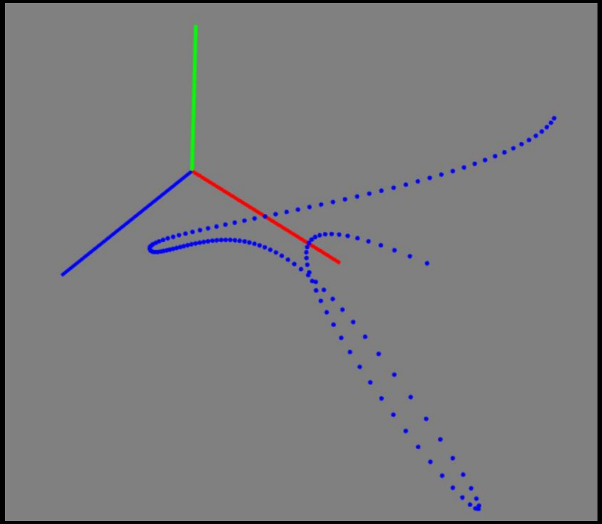
c. Press ‘+’ or ’-’ to increase or decrease point density

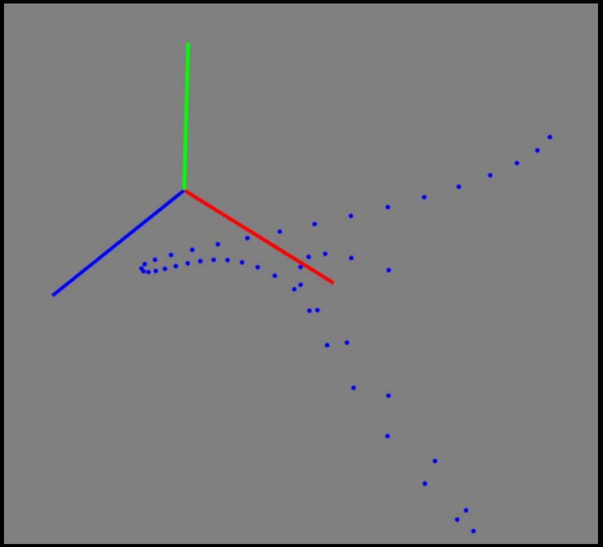
Pseudo code:

Kbd→case ‘+’

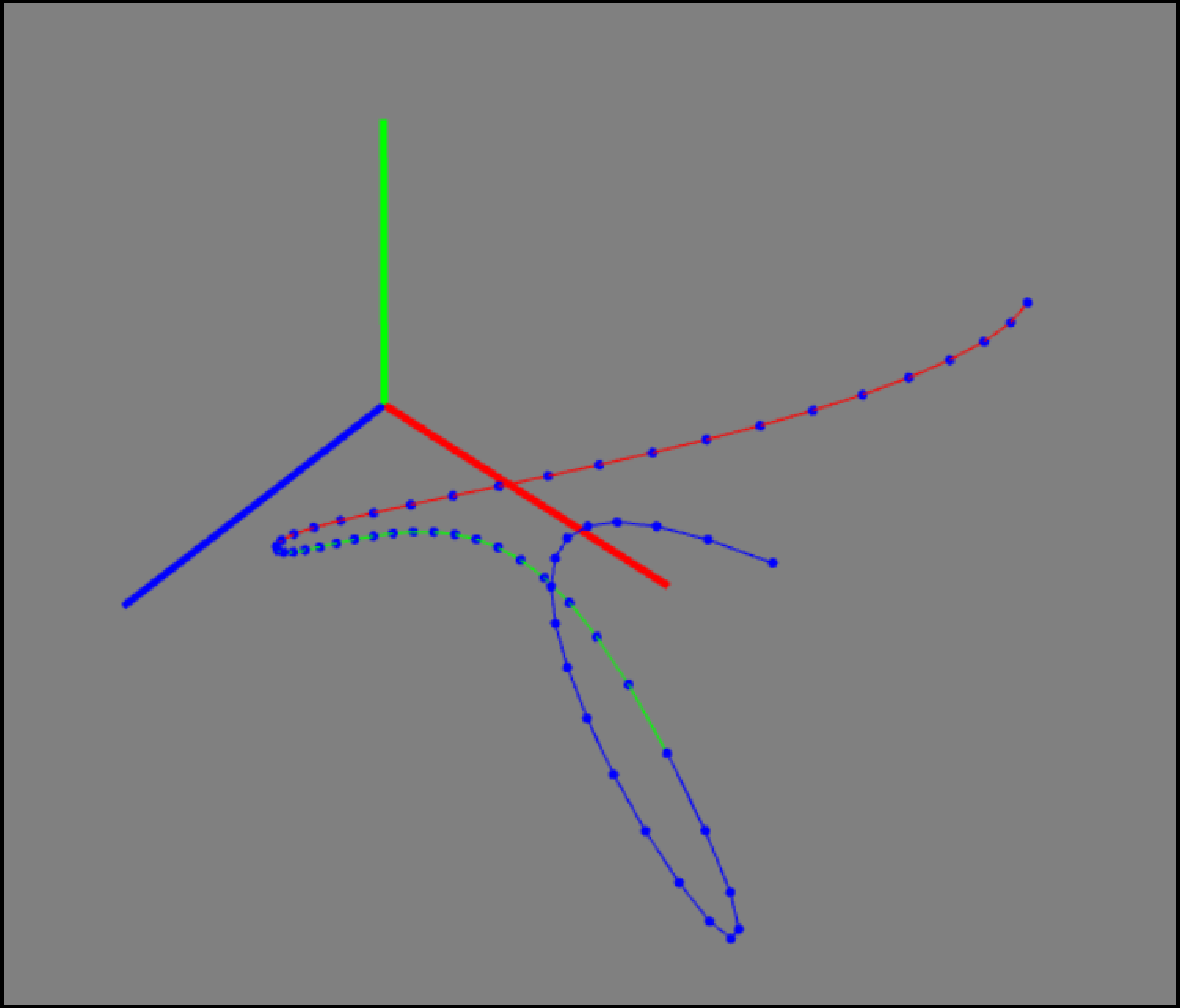
→case ‘-’

Increase or decrease no. of steps


(+)

(-)


2. Visualize the sequence as piecewise linear curve



3. Press ‘space’ to approximate points passing through a curve

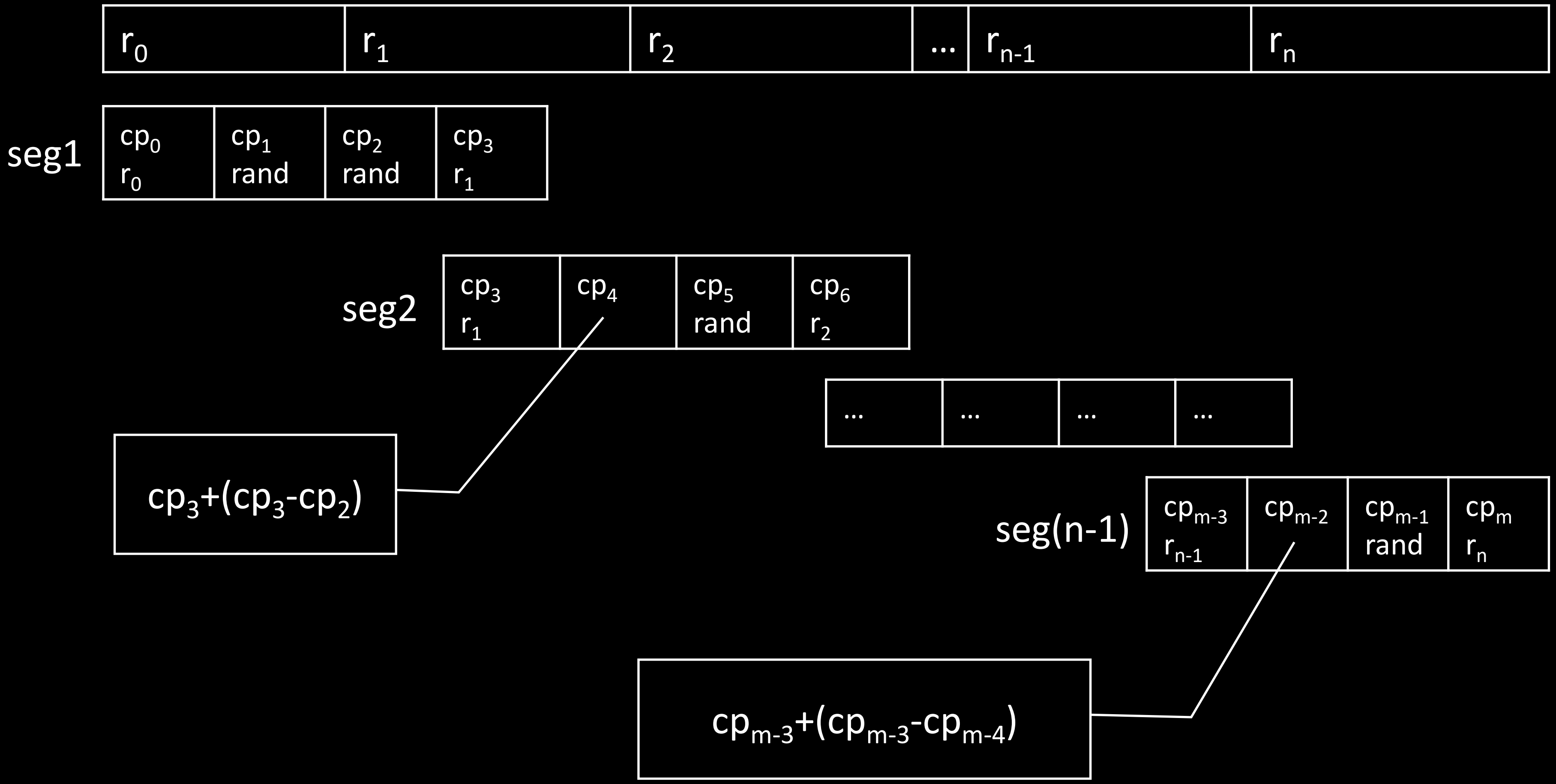
Pseudo code:

```
seg: no. of curve segments
r[i]: array of m random points
cp[j]: array of n control points for a curve passing
through every point of r[]
        seg= m-1
        n =  seg*4-seg+1
App[]: array of seg*steps points on the curve
```

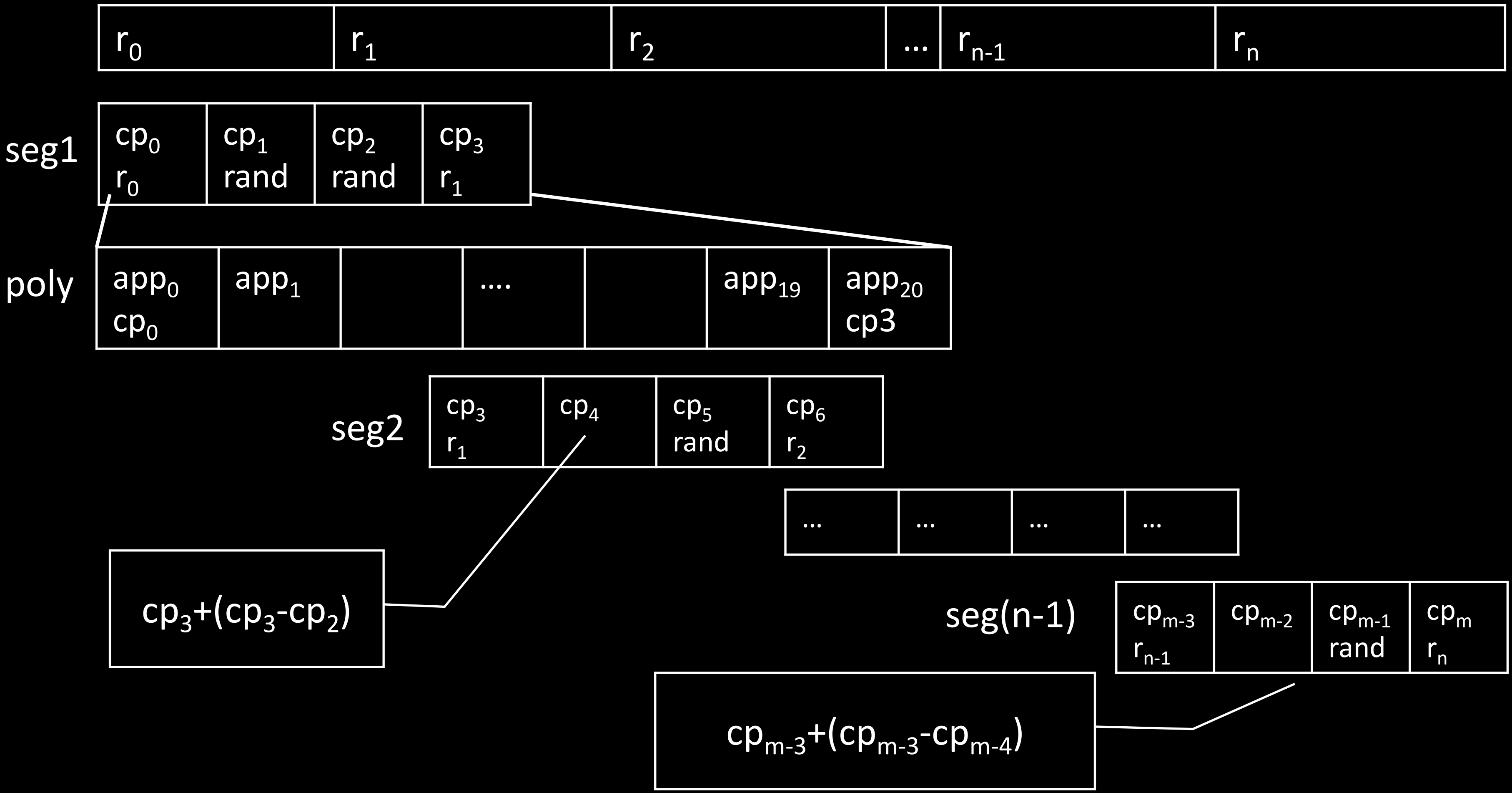
3. Press ‘space’ to approximate points passing through a curve

```
Pseudo code:
1. Create array r : random points
            r0, r1, r2, r3 …… rn

2. Create array cp : control points for polynomial/Bezier curve
seg1: cp0(r0), cp1 (random), cp2 (random), cp3(r1)
seg2: cp3(r1), cp4(cp3+(cp2-cp3)), cp5(random), cp6(r2)
seg3: cp6(r2), cp7(cp6+(cp5-cp6)), cp8(random), cp9(r3)
….
segn-1: cpn-3, cpn-2, cpn-1, cpn

3. Create app: points for approximate curve of each segment
seg1: app0=cp0, app1, app2, app2, …….app20 =cp3
seg2: app21=cp3, app22, app23,….app40=cp6
…
```

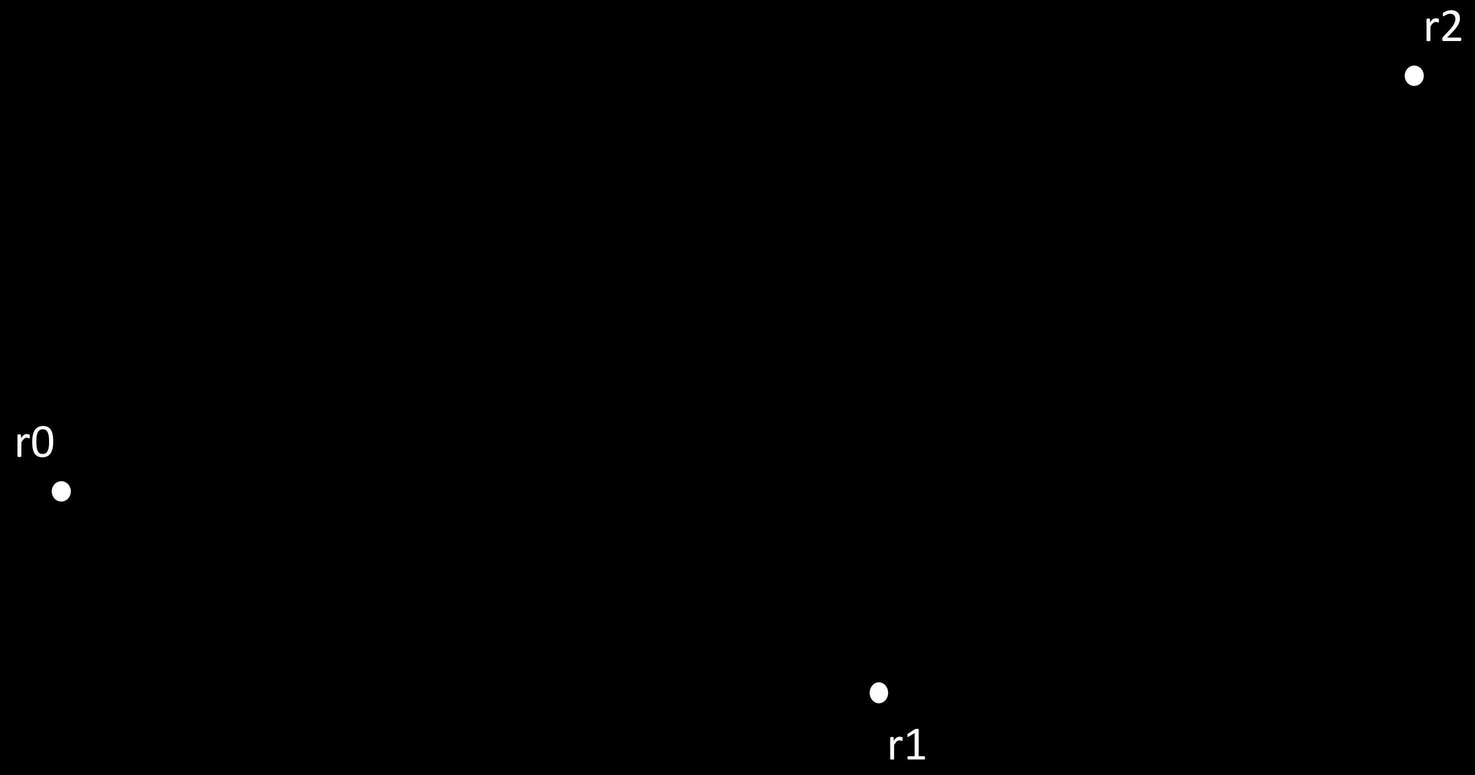
3. Press ‘space’ to approximate points passing through a curve



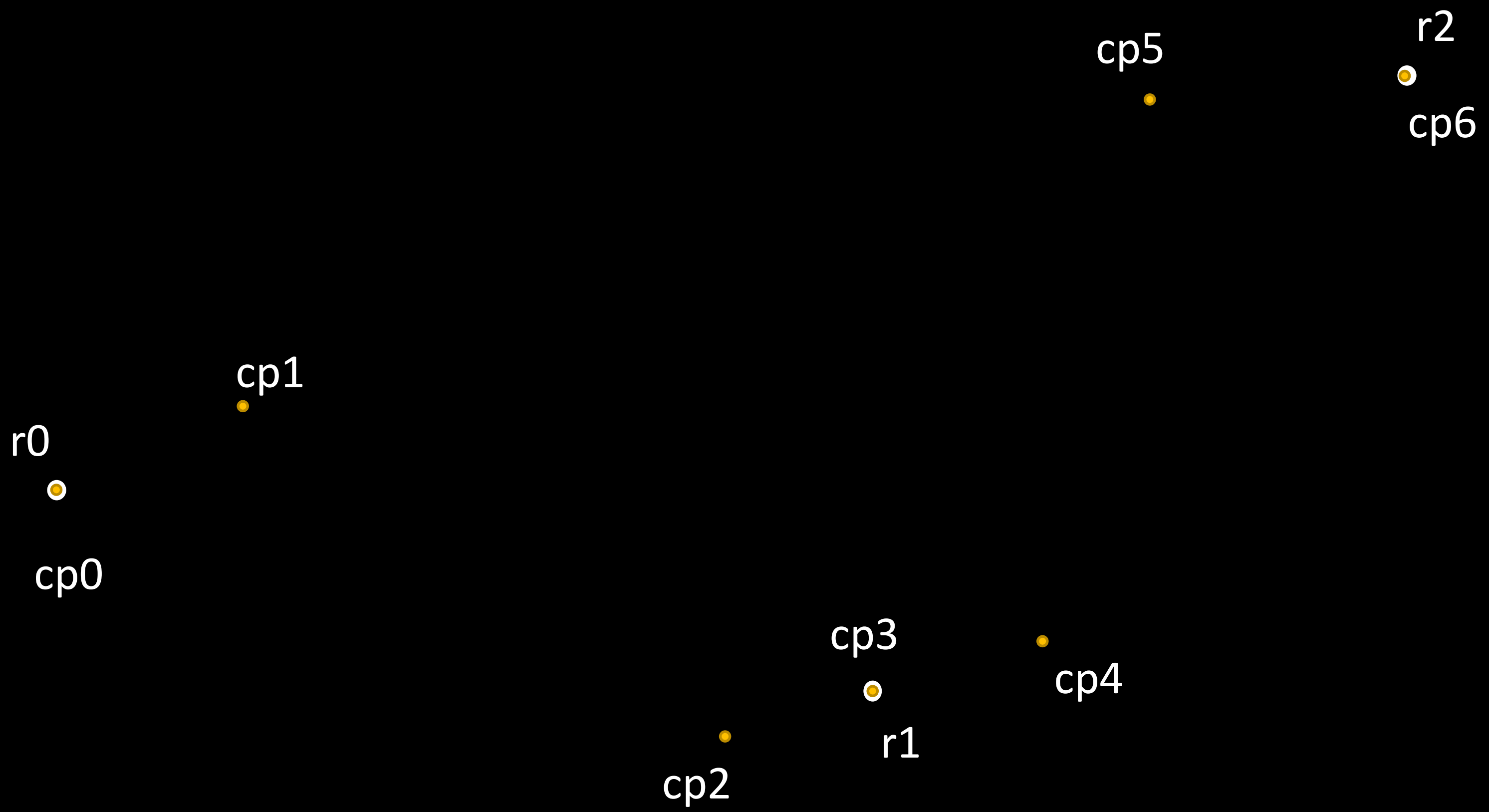
3. Press ‘space’ to approximate points passing through a curve


r_0 | r_1 | r_2 | … | r_{n-1} | r_n
seg1
cp_0 r_0 | cp_1 rand | cp_2 rand | cp_3 r_1
poly
app_0 cp_0 | app_1 | | …. | | app_{19} | app_{20} cp3
seg2
cp_3 r_1 | cp_4 | cp_5 rand | cp_6 r_2
… | … | … | …
$cp_3+(cp_3-cp_2)$
seg(n-1)
cp_{m-3} r_{n-1} | cp_{m-2} | cp_{m-1} rand | cp_m r_n
$cp_{m-3}+(cp_{m-3}-cp_{m-4})$

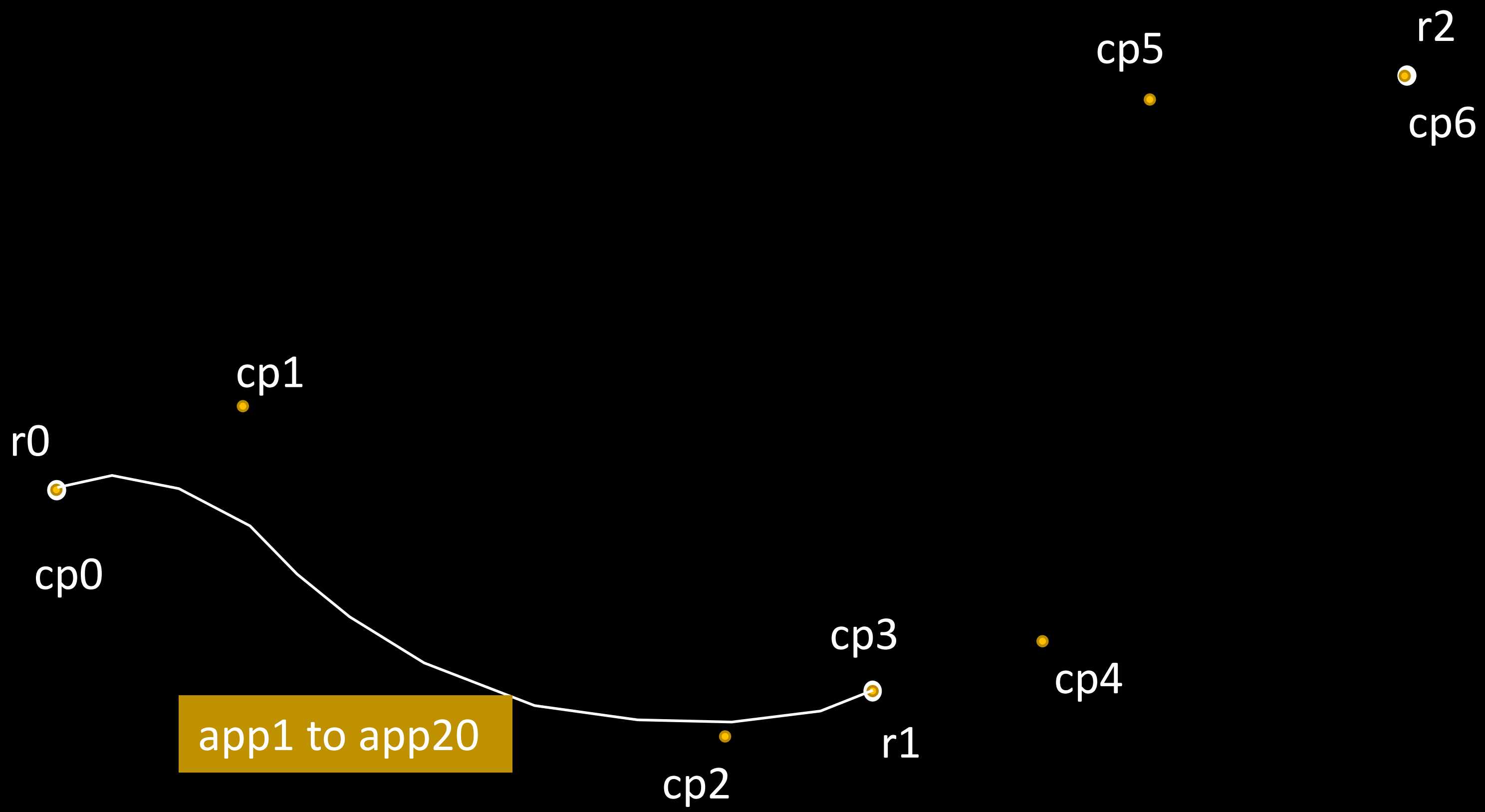
3. Press ‘space’ to approximate points passing through a curve



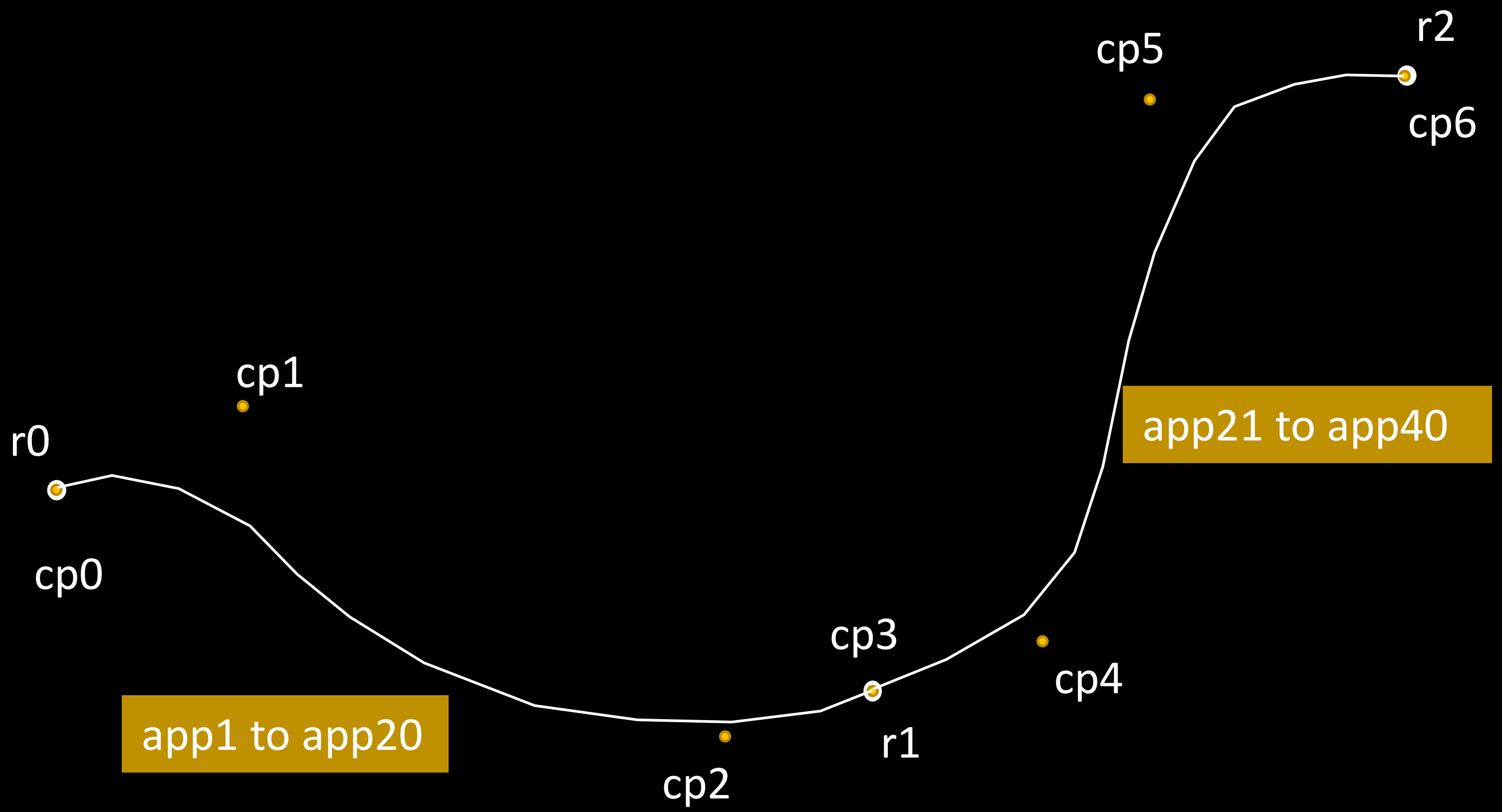
3. Press ‘space’ to approximate points passing through a curve



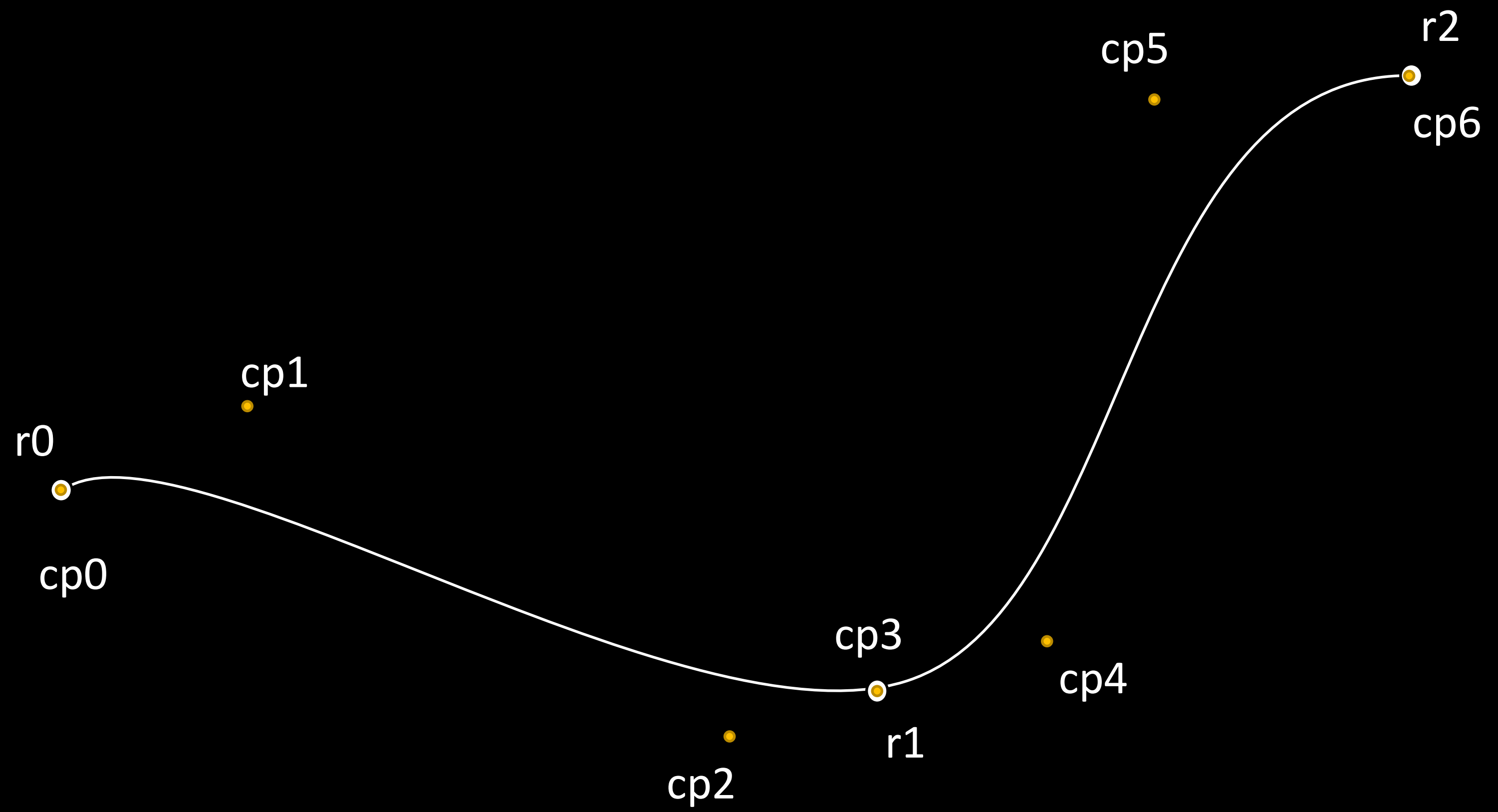
3. Press 'space' to approximate points passing through a curve



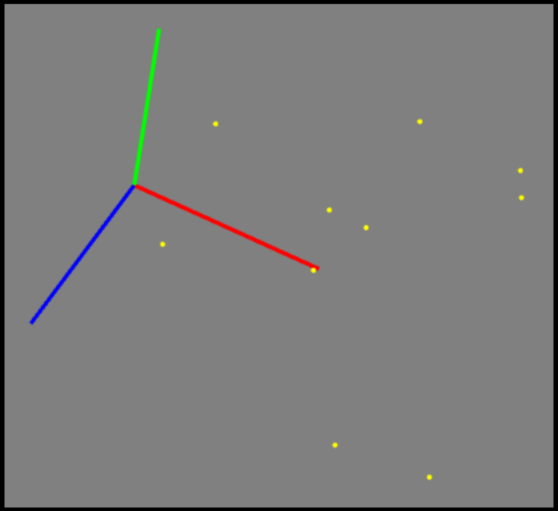
3. Press 'space' to approximate points passing through a curve

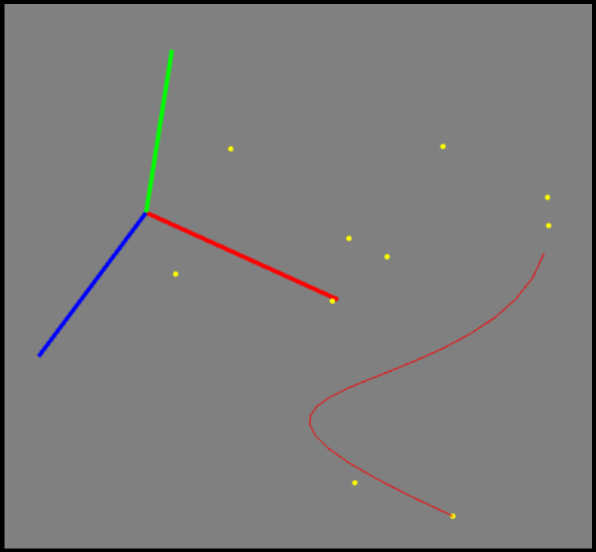


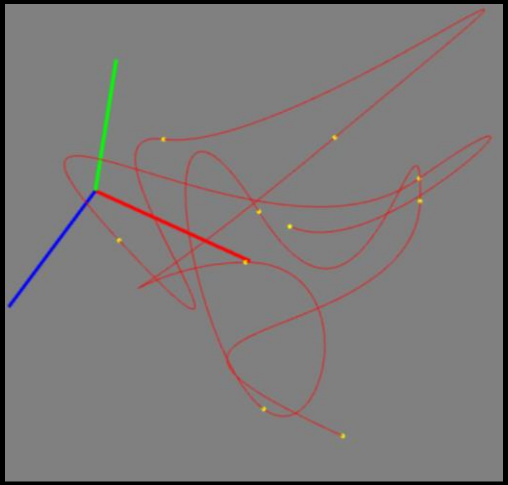
3. Press ‘space’ to approximate points passing through a curve



3. Press ‘space’ to approximate points passing through a curve



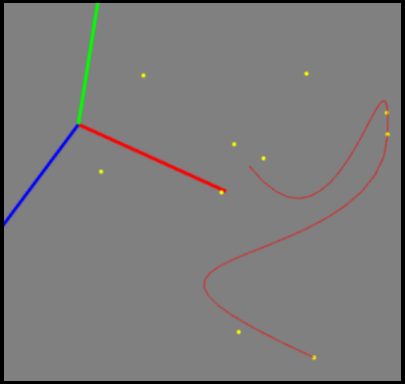




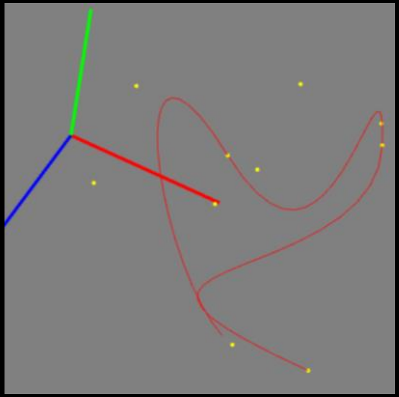
4. Press ‘<’ or ’>’ to change no. of segments of approximate curve

```
spaceCount=1
Kbd→case ‘>’
        spaceCount++
        case ‘<‘
        spaceCount--
```

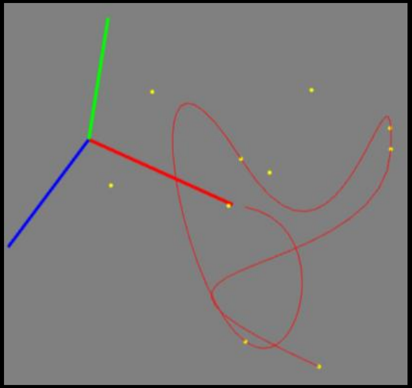
4. Press '<' or '>' to change no. of segments of approximate curve

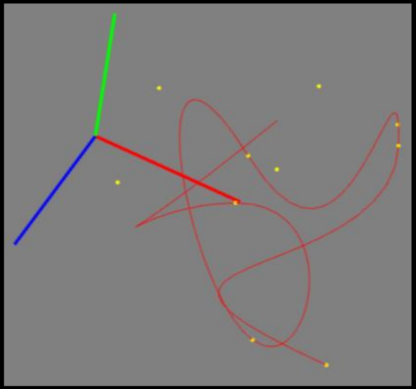


>

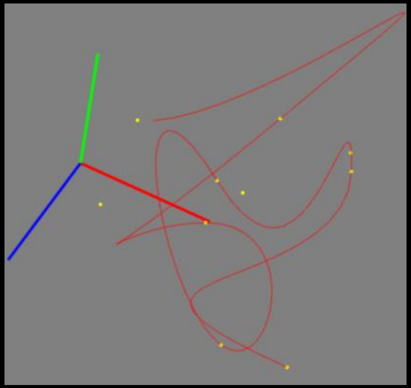


>

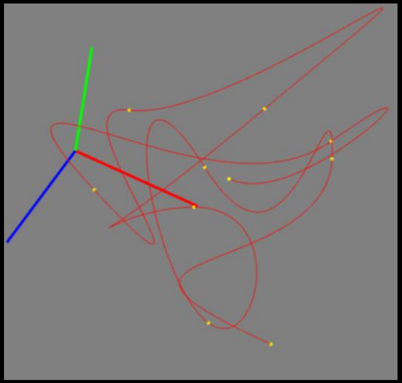




>



>



Thank You