Aplikasi *Divide and Conquer* pada:
1. Grafika Komputer
2. Evaluasi *expression tree*

Oleh: Rinaldi Munir

Informatika STEI-ITB

Bezier Curve

- ***Bezier Curve*** adalah kurva yang sering digunakan dalam grafika komputer (*computer graphics*).

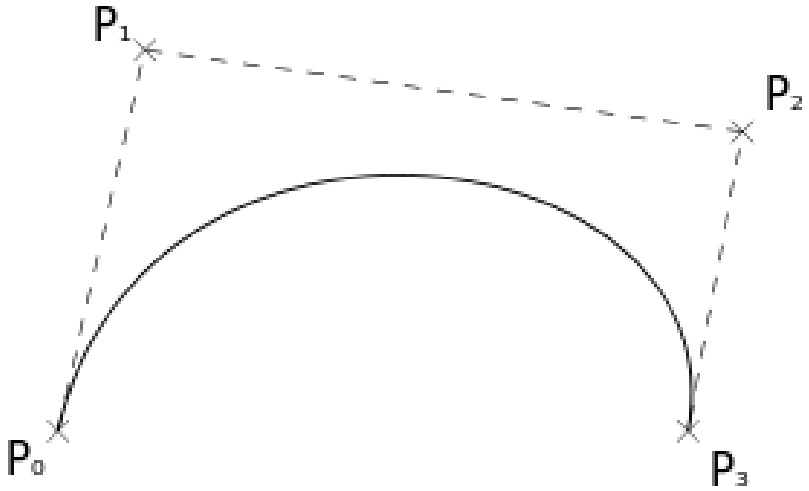
- Dalam grafik vektor, Bezier curves digunakan untuk memodelkan kurva mulus.

- Kurva Bézier dipublikasikan secara luas pada tahun 1962 oleh insinyur Pierre Bézier Perancis, yang menggunakannya untuk merancang badan mobil.

Pemodelan Kurva Bezier

- Sebuah kurva Bézier didefinisikan oleh satu set titik kontrol P_0 sampai P_n, *n* disebut order (n = 1 untuk linier, 2 kuadrat, dll).

- Titik kontrol pertama dan terakhir selalu titik akhir dari kurva, namun, titik kontrol antara (jika ada) umumnya tidak terletak pada kurva.

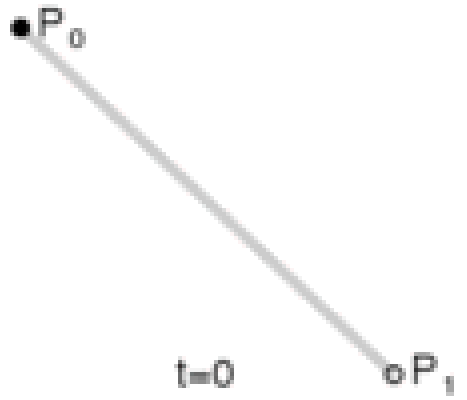


Kurva Linier

- Diberikan titik P_0 dan P_1, kurva Bézier linear adalah sebuah garis lurus antara dua titik. Kurva diberikan oleh:

$$\mathbf{B}(t) = \mathbf{P}_0 + t(\mathbf{P}_1 - \mathbf{P}_0) = (1-t)\mathbf{P}_0 + t\mathbf{P}_1 \ , \ t \in [0,1]$$

- t dalam fungsi kurva Bézier linier menggambarkan seberapa jauh B (t) dari P_0 ke P_1. Misalnya ketika *t* = 0,25, B (t) adalah seperempat dari jalan dari titik P_0 ke P_1. Seperti t bervariasi dari 0 ke 1, B (t) menggambarkan garis lurus dari P0 ke P1.



Kurva Kuadratik

- Diberikan poin yang diberikan P_0, P_1, P_2 , kurva Bézier kuadrat adalah lintasan yang dilalui oleh fungsi B (t),

$$\mathbf{B}(t) = (1-t)[(1-t)\mathbf{P}_0 + t\mathbf{P}_1] + t[(1-t)\mathbf{P}_1 + t\mathbf{P}_2] \ , \ t \in [0,1]$$

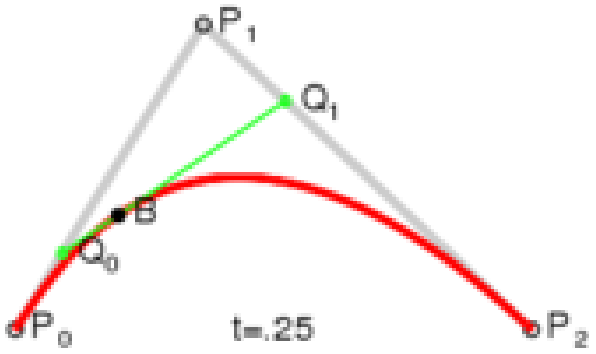
- yang dapat diartikan sebagai interpolasi linear dari titik yang sesuai pada kurva Bézier linier dari P_0 ke P_1 dan dari P_1 ke P_2.

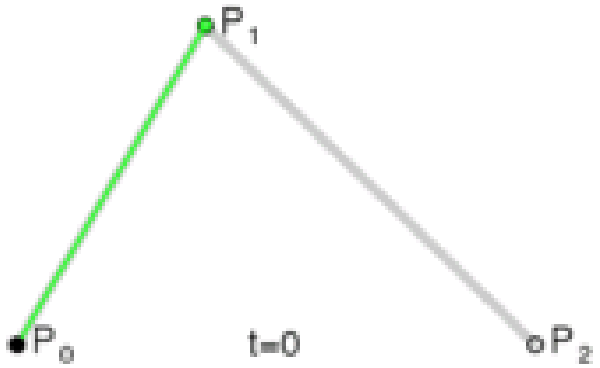
- Penyederhanaan:

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2(1-t)t\mathbf{P}_1 + t^2\mathbf{P}_2 \ , \ t \in [0,1].$$

- Untuk kurva Bézier kuadratik perlu dibangun titik antara (Q_0 dan Q_1 s bagaimana t bervariasi dari 0 sampai 1:

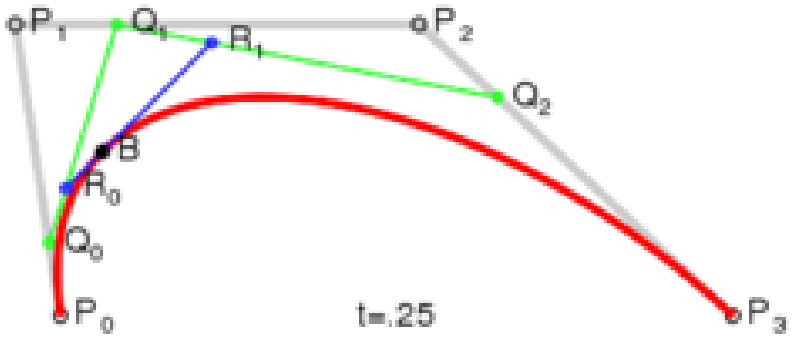
  (i) Titik Q_0 bervariasi dari P_0 ke P_1 → kurva Bezier linier.
  (ii) Titik Q_1 titik bervariasi dari P_1 ke P_2 → kurva Bézier linier.
  (iii) Titik B (t) bervariasi dari Q_0 ke Q_1 → kurva Bézier kuadrat.

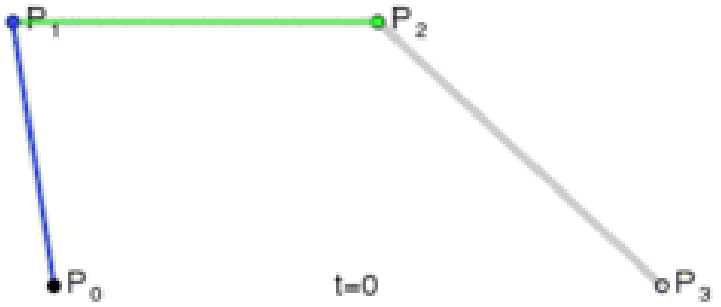




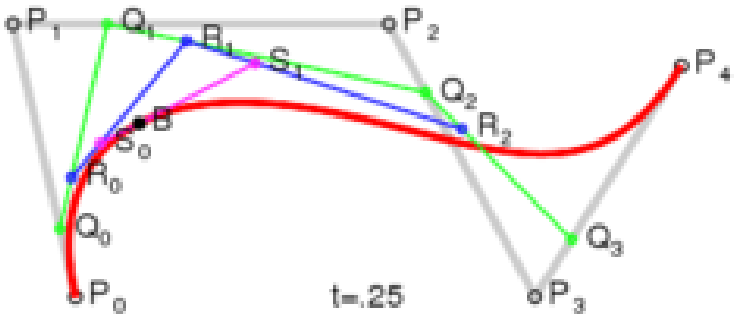
Kurva Bezier Orde Lebih Tinggi

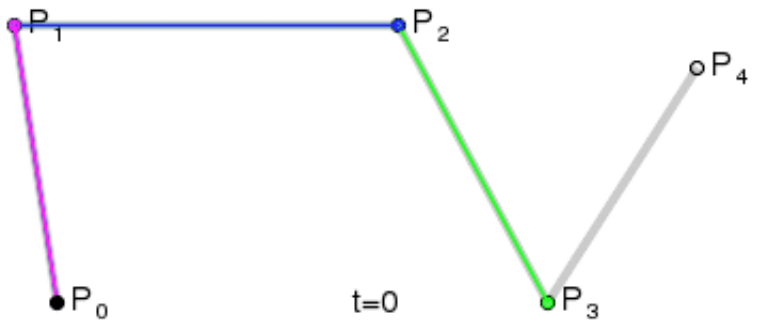
Orde 3:





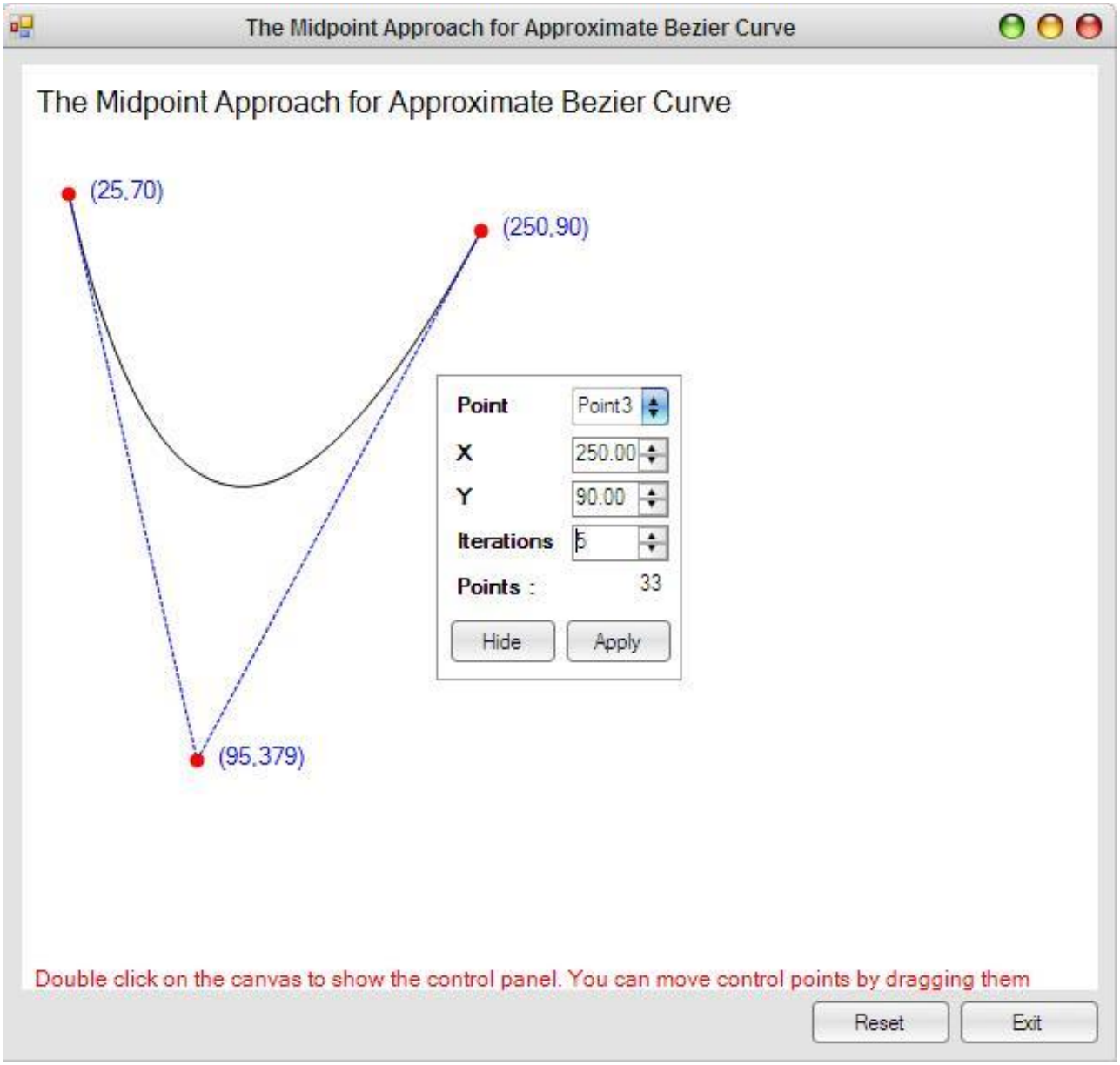
Orde 4:





Aplikasi *Divide and Conquer*

- Ada banyak cara membentuk kurva Bezier.
- Cara sederhana adalah menggunakan algoritma titik tengah yang berbasis *divide and conquer*.
- Pada contoh ini diperlihatkan cara mebentuk kurva Bezier kuadratik dengan algoritma titik tengah berbasis *divide and conquer*.


The Midpoint Approach for Approximate Bezier Curve
The Midpoint Approach for Approximate Bezier Curve
(25,70)
(250,90)
(95,379)
Point
Point3
X
250.00
Y
90.00
Iterations
5
Points :
33
Hide
Apply
Double click on the canvas to show the control panel. You can move control points by dragging them
Reset
Exit

- Kurva Bezier curve dimulai dengan tiga titik yang bisa di-set secara manual.

- Hitung titik tengah setiap garis yang terletak di antara tiga titik awal. Titik-titik pertengahan baru dihitung ditampilkan dalam warna hijau.

- Titik-titik yang mengubah warna menjadi biru akan berada di kurva Bezier akhir.

```
private void PopulateBezierPoints(PointF ctrl1, PointF ctrl2,
                    PointF ctrl3, int currentIteration)
{
 if (currentIteration < iterations) { //calculate next mid points
        PointF midPoint1 = MidPoint(ctrl1, ctrl2);
        PointF midPoint2 = MidPoint(ctrl2, ctrl3);
        PointF midPoint3 = MidPoint(midPoint1, midPoint2);   //the next
                                                              control point
        currentIteration++;
        PopulateBezierPoints(ctrl1, midPoint1, midPoint3,  currentIteration);
        //left branch
         bezierPoints.Add(midPoint3); //add the next control point
        PopulateBezierPoints(midPoint3, midPoint2, ctrl3, currentIteration);
        //right branch }
}
```

```
private PointF MidPoint(PointF controlPoint1, PointF controlPoint2)
{
  return new PointF(
        (controlPoint1.X + controlPoint2.X) / 2,
        (controlPoint1.Y + controlPoint2.Y) / 2 );
 }
```

```
private void CreateBezier(PointF ctrl1, PointF ctrl2, PointF
ctrl3)
{
        bezierPoints = new List<PointF>();
        bezierPoints.Clear();
        bezierPoints.Add(ctrl1);  // add the first control point
        PopulateBezierPoints(ctrl1, ctrl2, ctrl3, 0);
        bezierPoints.Add(ctrl3);  // add the last control point
}
```

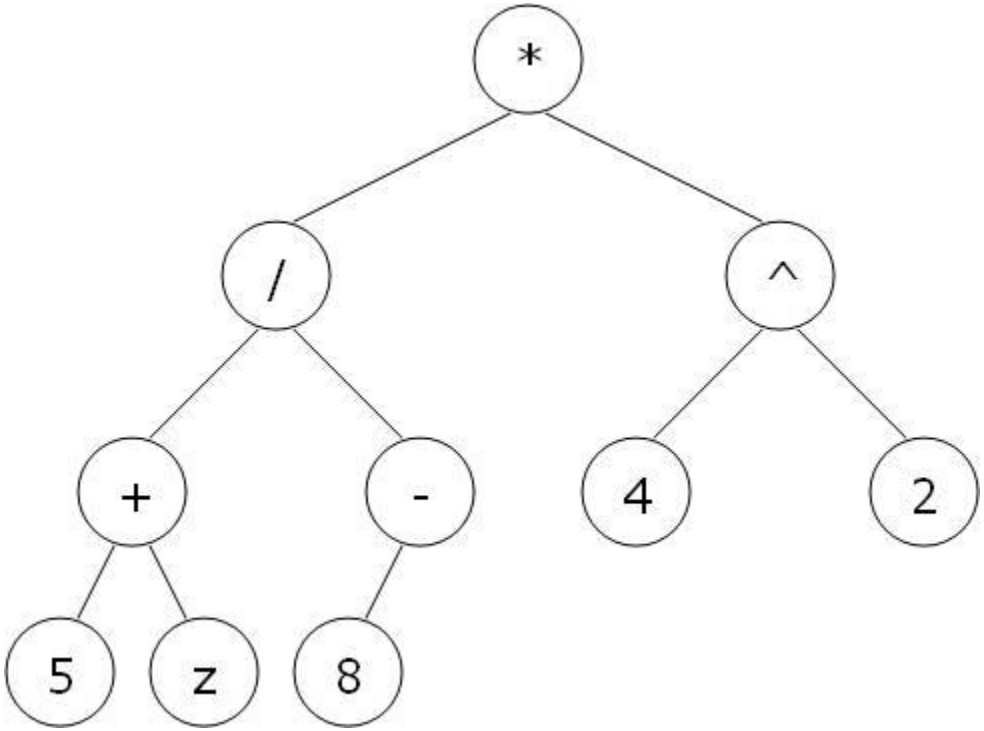
Sumber:

1. http://www.codeproject.com/Articles/223159/Midpoint-Algorithm-Divide-and-Conquer-Method-for-D

2. http://en.wikipedia.org/wiki/Bezier_curve

Expression Tree

- Di dalam *compiler* bahasa pemrograman, ekspresi aritmetika direpresentasikan dalam pohon biner yaitu *expression tree*
- Contoh: (5 + z) / -8) * (4 ^ 2)

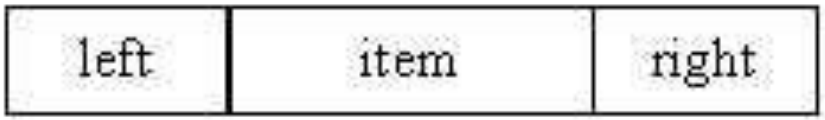


Sumber gambar: Wikipedia.org

Mengevaluasi *Expression Tree*

- Simpul daun → operand
- Simpul dalam → operator (+, -, *, /)

- Struktur data pohon:



- Pada simpul daun → left = NIL dan right = NIL

- Algoritma *divide and conquer*:

If node adalah simpul daun

return nilainya

else

secara rekursif evaluasi upa-pohon kiri dan return nilainya

secara rekursif evaluasi upa-pohon kanan dan return nilainya

lakukan operasi yang bersesuaian dengan operator dan return nilainya

```
procedure Evaluasi(input T : Pohon, output nilai : integer)
Algoritma:
    if left(T) = NIL and right(T) = NIL  { simpul daun}
        nilai ← item(T)
    else   { simpul dalam }
            Evaluasi(left(T), nilai1);
            Evaluasi(right(T), nilai2);
            case item(T) of
                "+" : nilai ← nilai1 + nilai2
                "-" : nilai ← nilai1 - nilai2
                "*" : nilai ← nilai1 * nilai2
                "/" : nilai ← nilai  / nilai2
     end
end
```

function Evaluasi(T : Pohon) → **integer**

Algoritma:

```
    if left(T) = NIL and right(T) = NIL  { simpul daun}
        return item(T)
    else   { simpul dalam }
      case item(T) of
          "+" : return Evaluasi(left(T)) + Evaluasi(right(T))
          "-" : return Evaluasi(left(T)) - Evaluasi(right(T))
          "*" : return Evaluasi(left(T)) * Evaluasi(right(T))
          "/" : return Evaluasi(left(T)) / Evaluasi(right(T))
     end
    end
```