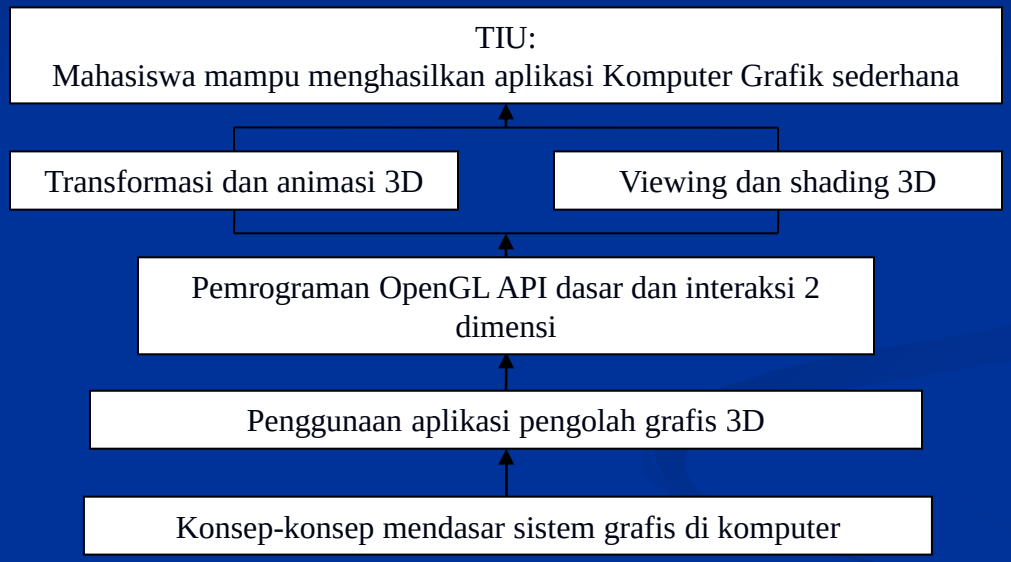
# Kurva

Pertemuan

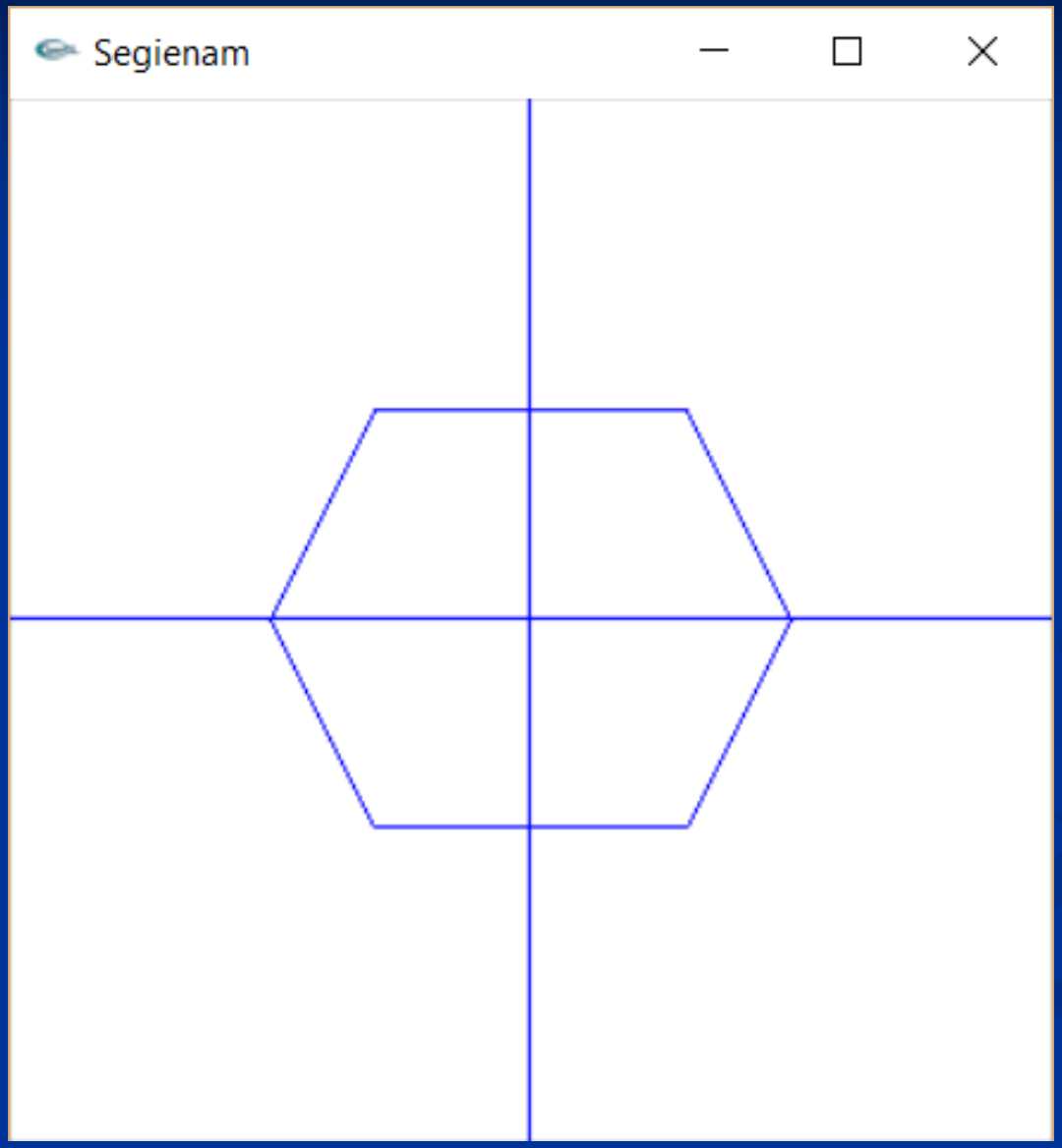
Yonathan Ferry Hendrawan
Universitas Trunojoyo Madura


TIU:
Mahasiswa mampu menghasilkan aplikasi Komputer Grafik sederhana
Transformasi dan animasi 3D
Viewing dan shading 3D
Pemrograman OpenGL API dasar dan interaksi 2 dimensi
Penggunaan aplikasi pengolah grafis 3D
Konsep-konsep mendasar sistem grafis di komputer

# Bahasan

- Pokok: Kurva
- Sub:
  - Definisi kurva
  - Pendekatan naïve
  - Pendekatan Matematis:
    - Polinomial
      - Derajat 1: Linier
      - Derajat 2: Kuadrat
      - Derajat > 2
    - Trigonometri
    - Lingkaran
    - Berbasis titik kontrol
      - Bezier
      - B-spline

# Latihan




Jarak titik-titik sudut ke pusat koordinat sama.

# Kurva

- Definisi: obyek menyerupai garis yang tidak harus lurus.
- Garis lurus adalah kurva khusus yang tidak memiliki lengkungan

# Membuat kurva

- Pendekatan naïve: menggambar titik demi titik yang saling bersebelahan.
- Pendekatan matematis: menggunakan persamaan matematika.
  - Polinomial
  - Trigonometri
  - Lingkaran
  - Control Points

# Polinomial

- Polynomial adalah persamaan matematika dalam bentuk:

$$a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_Lx^L$$

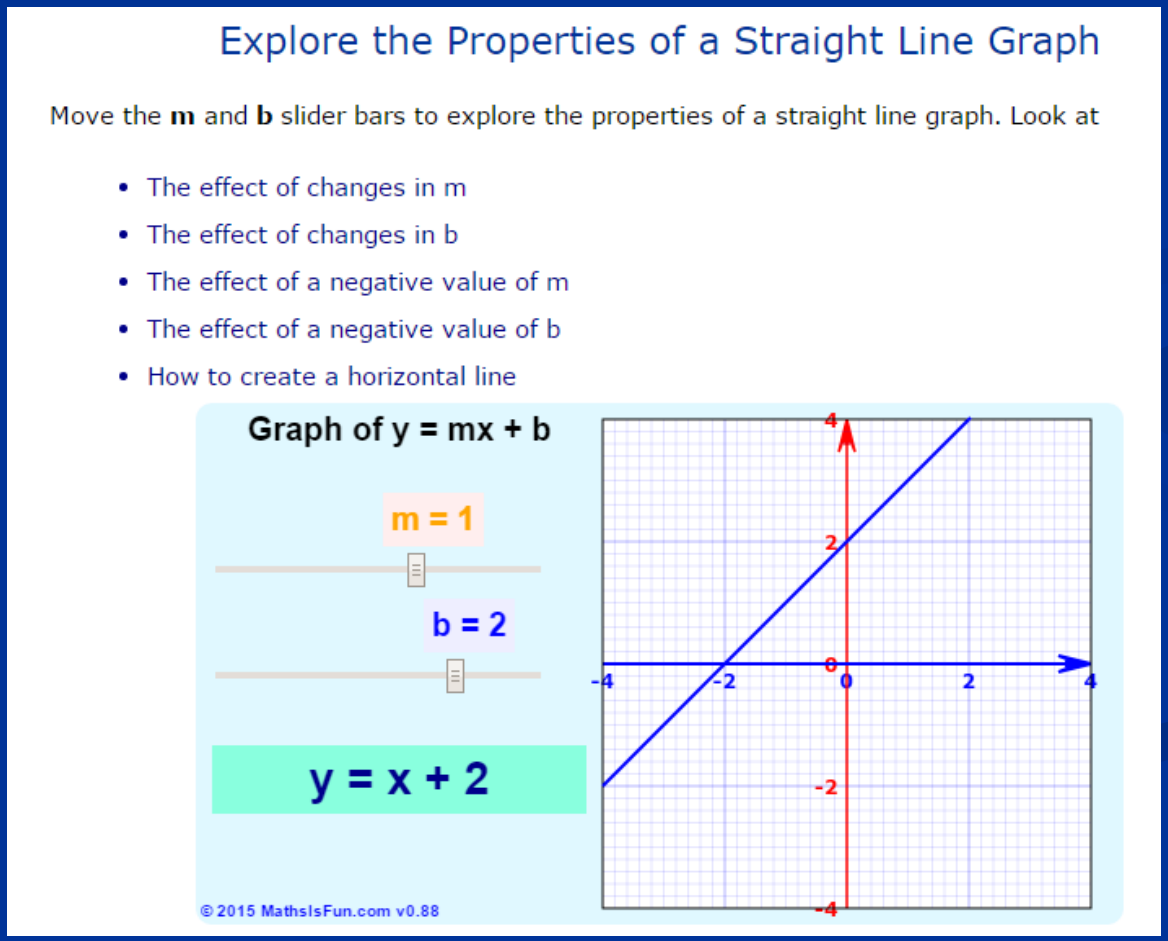
- Linier: L = 1
  - Bentuk eksplisit: $y = mx + c$
- Kuadrat: L = 2
  - Bentuk eksplisit: $y = ax^2 + bx + c$

# Persamaan Linier

- $y = mx + c$
- c: konstanta, m: gradien/kemiringan
- Rumus m:
  - $m = \Delta y/\Delta x = (y_2 - y_1) / (x_2 - x_1)$
  - $m = \tan(\theta)$

# Persamaan Linier

- Lihat:
  http://www.mathsisfun.com/data/straight_line_graph.html



# Persamaan Linier

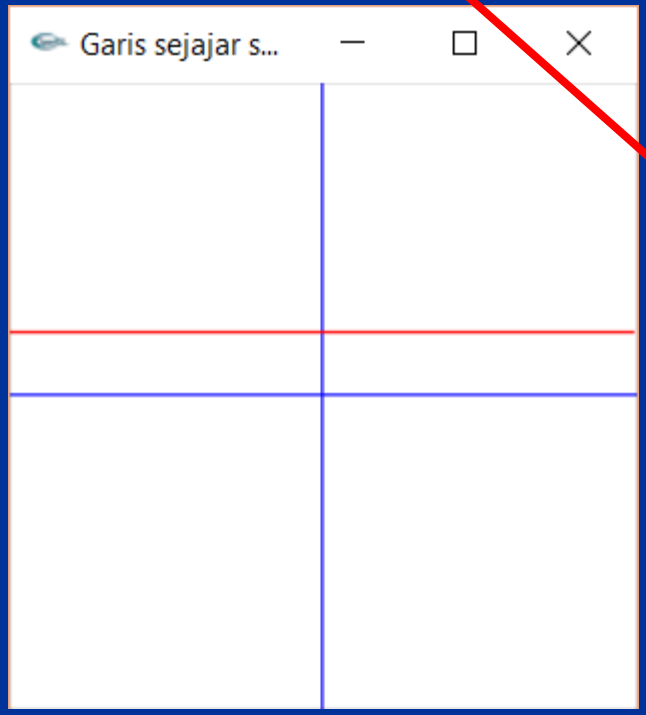
- Persamaan parametrik adalah persamaan yang mengekspresikan titik-titik koordinat sebuah kurva sebagai fungsi dari sebuah variabel/parameter.
- Contoh:
  - $S_{xt} = v_x.t + S_{0x}$
  - $S_{yt} = v_y.t + S_{0y}$

# Persamaan Linier

- Persamaan parametrik lebih mudah digambar oleh komputer.

# Persamaan Linier

- Persamaan linier y = 2
  - x = t, y = 2




```
glBegin(GL_LINES);
glVertex2i(10,0);
glVertex2i(-10,0);
glVertex2i(0,10);
glVertex2i(0,-10);
glEnd();

glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_STRIP);
for(float t =-10.0; t<=10.0; t+=0.1){
    glVertex2f(t,2);
}
glEnd();
```

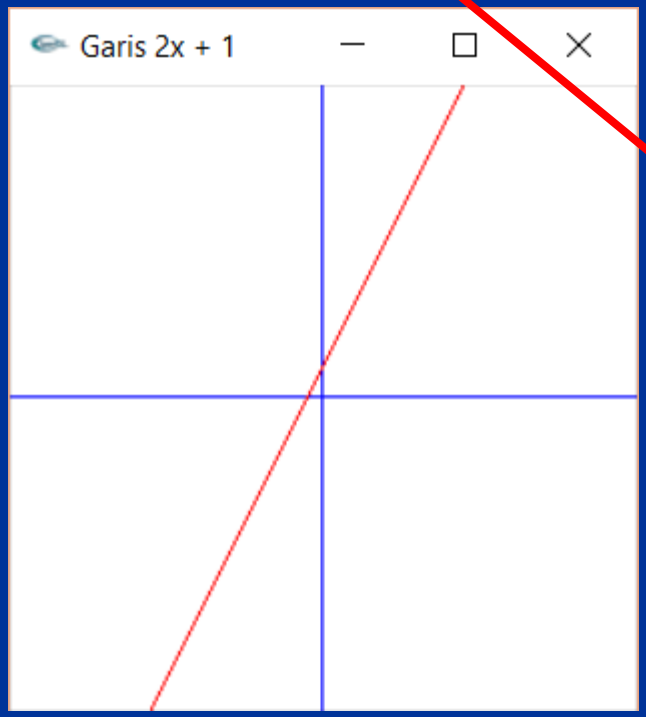
# Persamaan Linier

- Konversi:
  - `y = 2.x` → `x = t` → `y = 2.t`
  - `y = 2.x+3` → `x = t` → `y = 2.t+3`

- x menjadi fungsi dari parameter t, dan y juga menjadi fungsi dari parameter t.

# Persamaan Linier

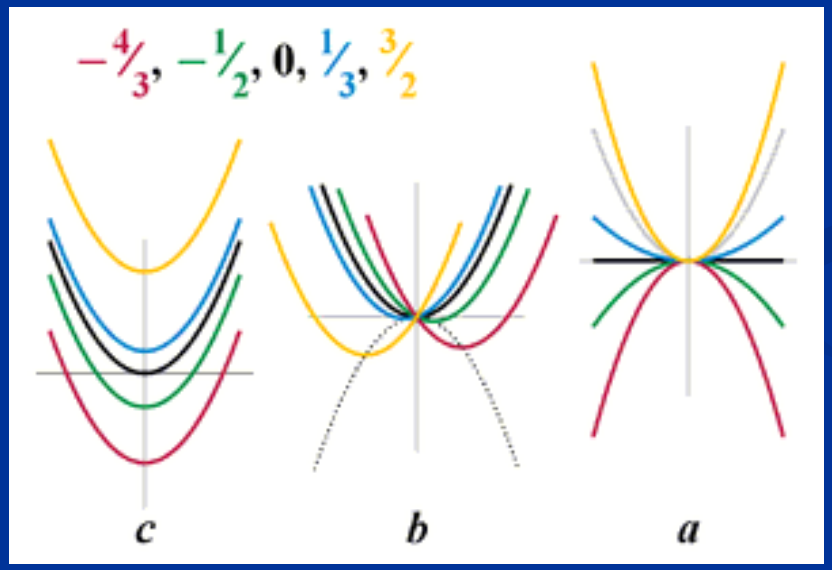
- Persamaan linier y = 2.x + 1
  - x = t, y = 2.t + 1




```
glBegin(GL_LINES);
glVertex2i(10,0);
glVertex2i(-10,0);
glVertex2i(0,10);
glVertex2i(0,-10);
glEnd();

glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_STRIP);
for(float t =-10.0; t<=10.0; t+=0.1){
    glVertex2f(t,2*t+1);
}
glEnd();
```

# Persamaan Kuadrat

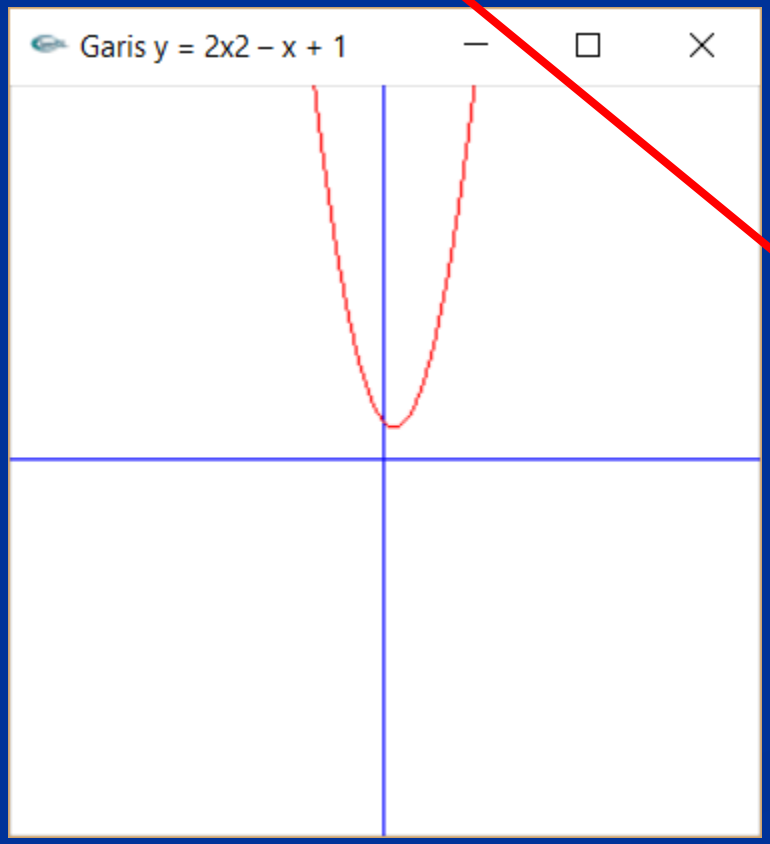
- Bentuk eksplisit: $y = ax^2 + bx + c$
- Pengaruh a, b, dan c terhadap bentuk kurva:




- Bentuk parametrik:
  - $x = t,\ y = at^2 + bt + c$

# Persamaan Kuadrat

- Persamaan kuadrat $y = 2x^2 - x + 1$
  - $x = t,\ y = 2t^2 - t + 1$



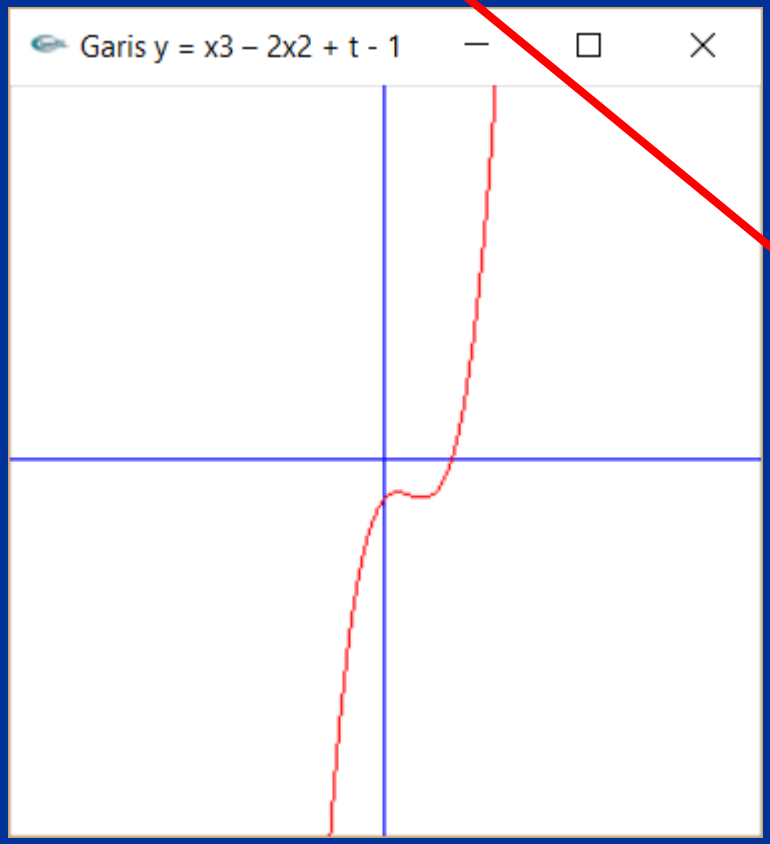

```
glBegin(GL_LINES);
glVertex2i(10,0);
glVertex2i(-10,0);
glVertex2i(0,10);
glVertex2i(0,-10);
glEnd();

glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_STRIP);
for(float t =-10.0; t<=10.0; t+=0.1){
    glVertex2f(t,2*t*t-t+1);
}
glEnd();
```

# Menggambar Kurva dengan Looping

- Untuk kurva linier, kita dapat menggunakan GL_LINE_LOOP atau GL_LINE_STRIP. Untuk kurva kuadrat dan derajat yang lebih tinggi, gunakan GL_LINE_STRIP.
- Interval untuk parameter t haruslah cukup kecil supaya kurva yang digambar terlihat halus/smooth.
- Titik awal dan titik akhir looping tidak harus sesuai gluOrtho2D

# Persamaan Polinomial derajat >2

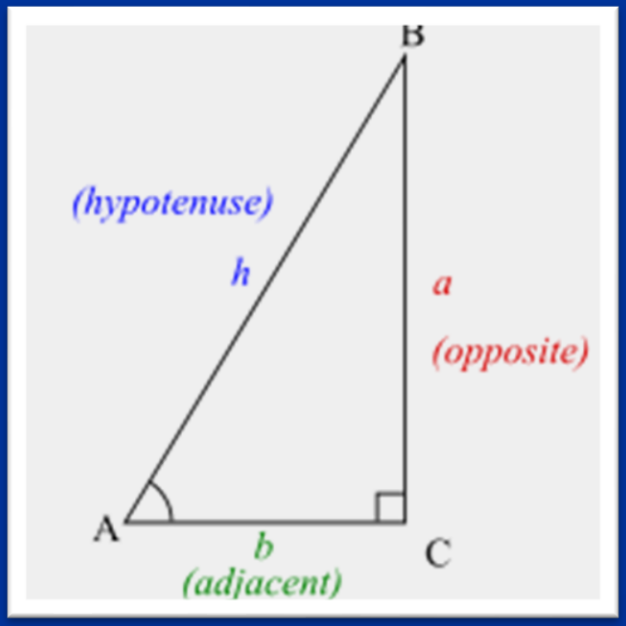
- Persamaan $y = x^3 - 2x^2 + x - 1$
  - $x = t,\ y = t^3 - 2t^2 + t - 1$

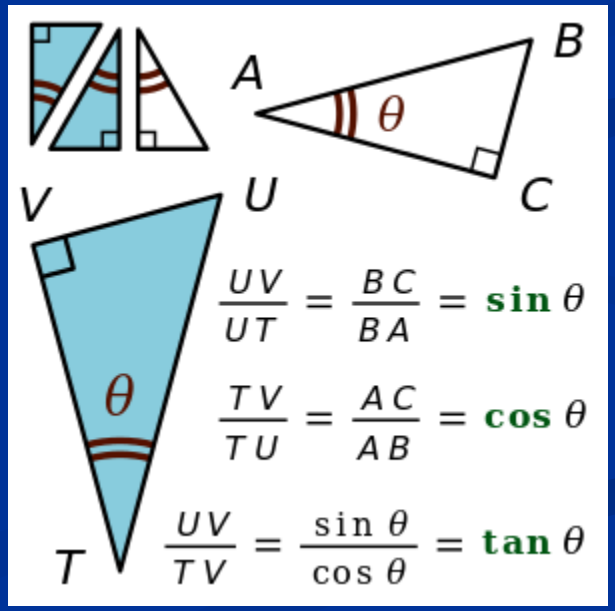



```
glBegin(GL_LINES);
glVertex2i(10,0);
glVertex2i(-10,0);
glVertex2i(0,10);
glVertex2i(0,-10);
glEnd();

glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_STRIP);
for(float t =-10.0; t<=10.0; t+=0.1){
    glVertex2f(t,t*t*t-2*t*t+t-1);
}
glEnd();
```

# Trigonometri

- Fungsi trigonometri: fungsi yang terbentuk/bergantung pada sudut sebuah segitiga

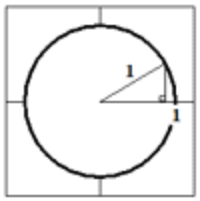



Wikipedia, Trigonometric functions, https://en.wikipedia.org/wiki/Trigonometric_functions

# Trigonometri

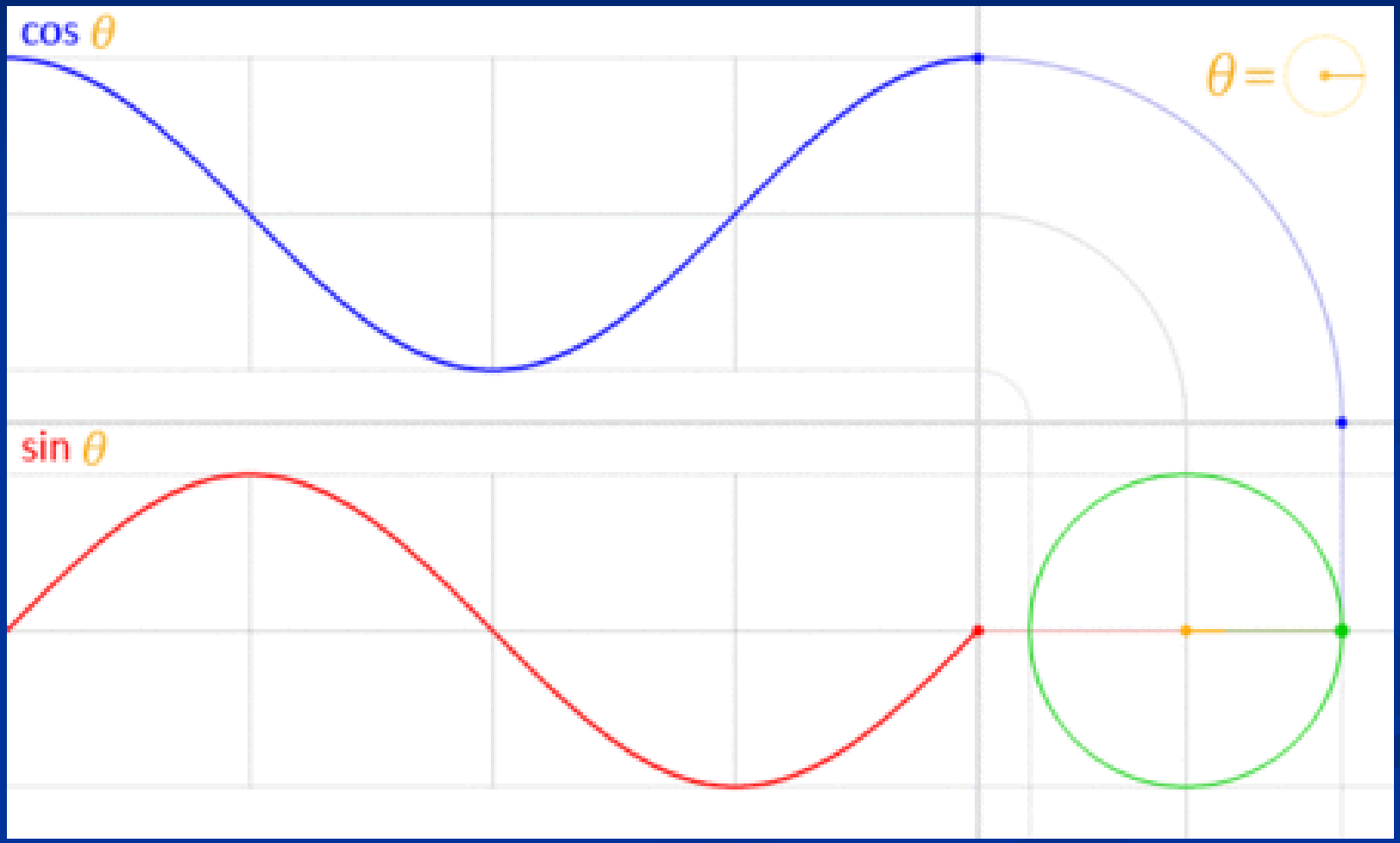
- Menggambar kurva dengan fungsi trigonometri: sudut theta jadi parameternya.




Unit Circle

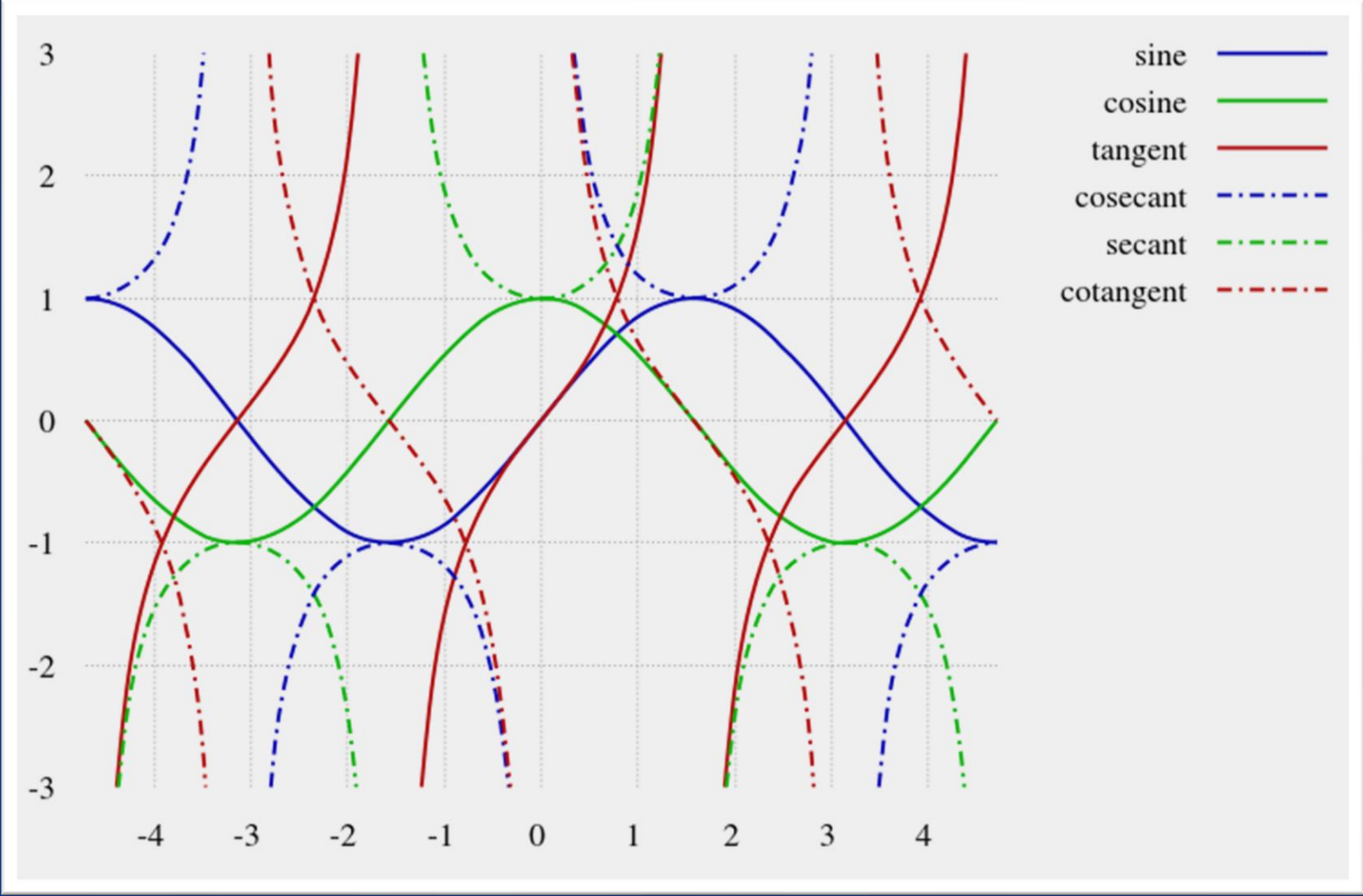
Wikipedia, Sine and Cosine fundamental relationship to Circle (and Helix)
, https://en.wikipedia.org/wiki/File:Sine_and_Cosine_fundamental_relationship_to_Circle_(and_Helix).gif

# Trigonometri




Wikipedia, Sine, https://en.wikipedia.org/wiki/Sine

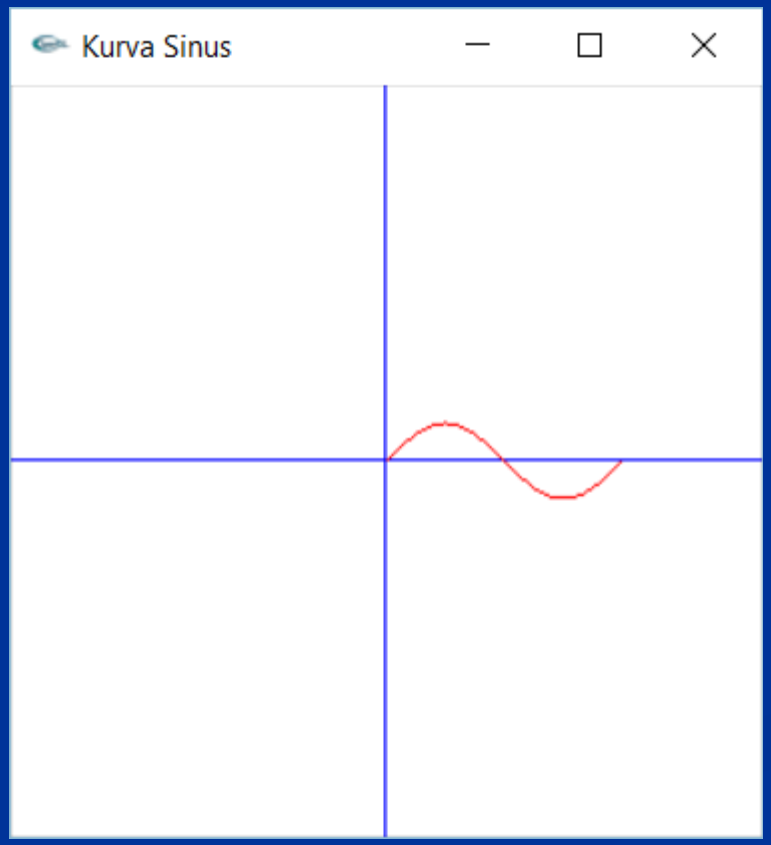
# Trigonometri




Wikipedia, Trigonometric Functions, https://en.wikipedia.org/wiki/Trigonometric_functions

# Trigonometri

- Dalam pemrograman C++, untuk memanggil fungsi trigonometri, kita perlu #include<math.h>




```
glBegin(GL_LINES);
glVertex2i(10,0);
glVertex2i(-10,0);
glVertex2i(0,10);
glVertex2i(0,-10);
glEnd();

glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_STRIP);
for(float t =0.0; t<=6.28; t+=0.01){
    glVertex2f(t,sin(t));
}
glEnd();
```

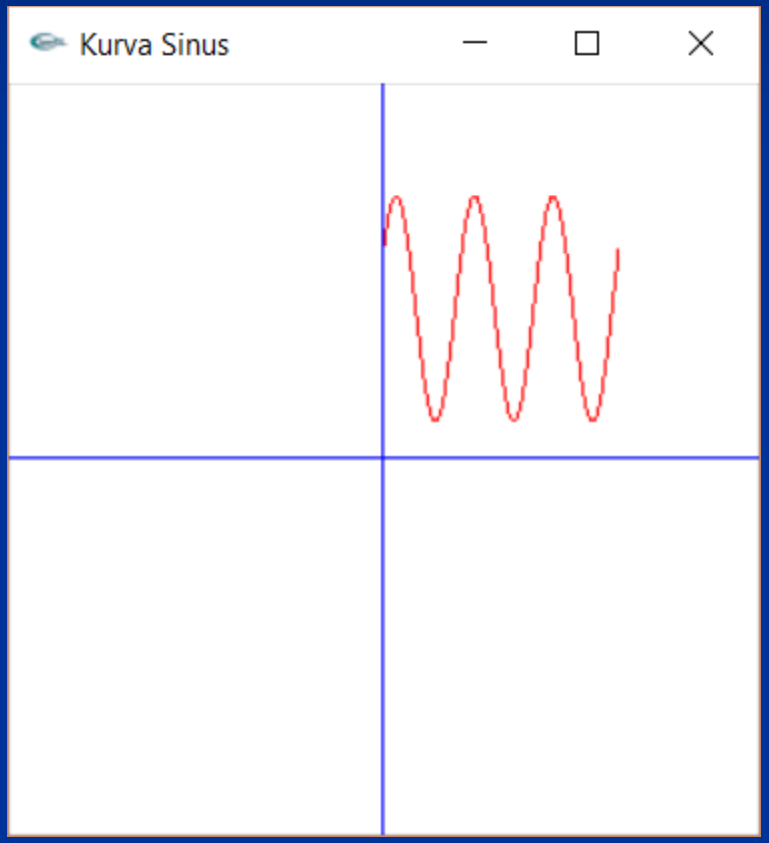
# Trigonometri

- Bentuk umum fungsi Sin:

$$y = A.\sin(B.\alpha + C) + D$$

- Dimana:
  - A: besar kecilnya amplitudo
  - B: banyaknya gelombang dalam satu periode
  - C: pergeseran gelombang pada sumbu x
  - D: pergeseran gelombang pada sumbu y

# Trigonometri

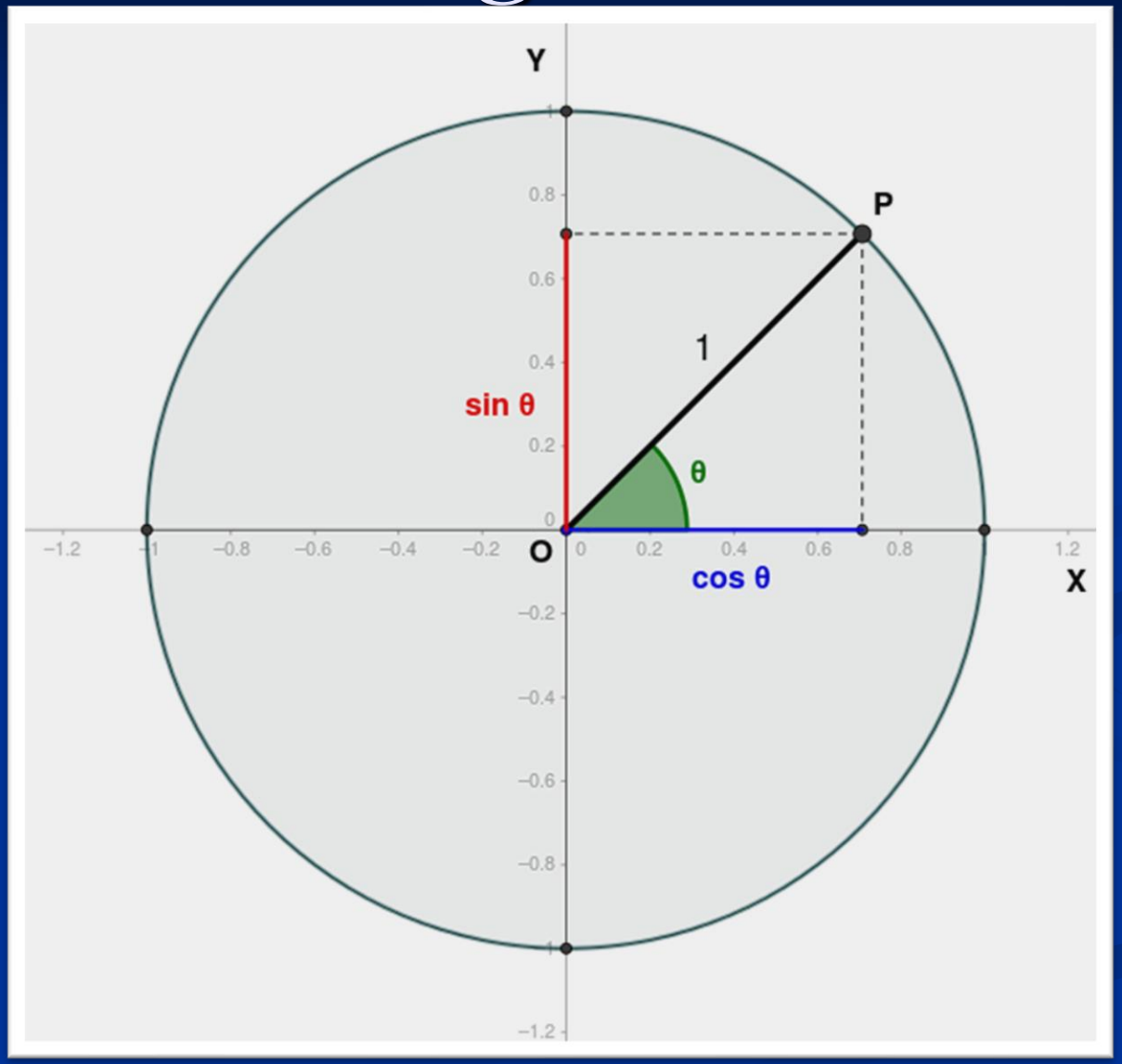



```
glBegin(GL_LINES);
glVertex2i(10,0);
glVertex2i(-10,0);
glVertex2i(0,10);
glVertex2i(0,-10);
glEnd();

glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_STRIP);
for(float t =0.0; t<=6.28; t+=0.01){
    glVertex2f(t,3*sin(3*t+0.6)+4);
}
glEnd();
```

# Lingkaran

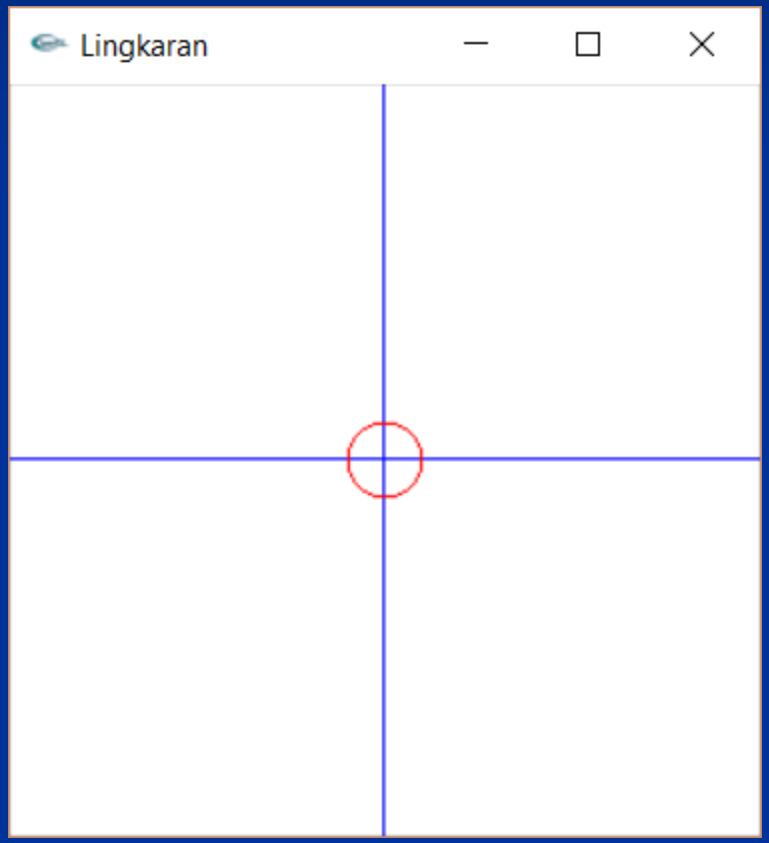
- Pada 3ds Max, untuk membuat sebuah silinder, kita membaginya menjadi beberapa segmen.
- Cara yang sama kita gunakan untuk membuat sebuah lingkaran: membuat titik-titik yang cukup rapat satu sama lain sepanjang jalur lingkar.
- Fungsi parametriknya?

# Lingkaran




Wikipedia, Trigonometric Functions, https://en.wikipedia.org/wiki/Trigonometric_functions

# Lingkaran




```
glBegin(GL_LINES);
glVertex2i(10,0);
glVertex2i(-10,0);
glVertex2i(0,10);
glVertex2i(0,-10);
glEnd();

glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_STRIP);
for(float t =0.0; t<=6.28; t+=0.01){
    glVertex2f(cos(t),sin(t));
}
glEnd();
```

# Lingkaran

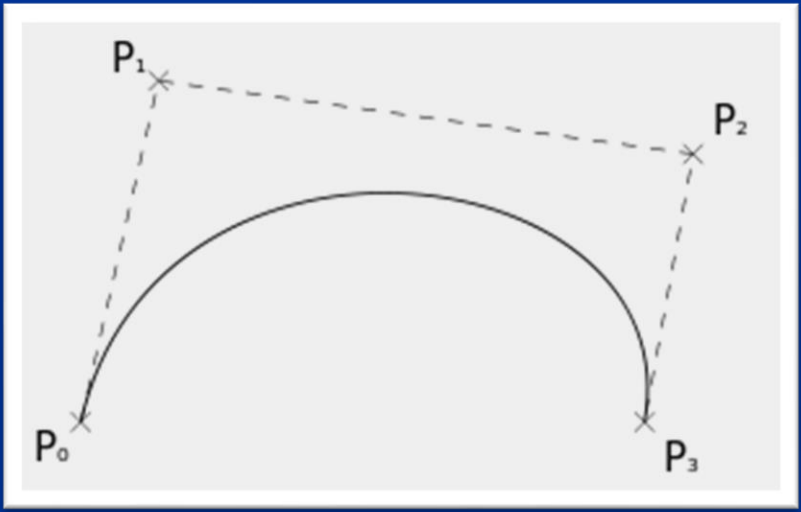
- Untuk memperbesar radius lingkaran, cukup kalikan fungsi cos dan sin dengan angka tertentu.
- Untuk membuat elips, angka pengali fungsi cos dan sin dibuat berbeda.

# Berbasis Titik Kontrol

- Mencari fungsi yang pas dengan kebutuhan kadang kala sukar dilakukan.
- Lebih mudah jika fungsinya fix, kita hanya perlu memberikan titik-titik pengontrolnya:
  - Bezier curve
  - B-Spline

# Bezier Curve

- Punya 4 titik kontrol
- Sifat:
  - Menginterpolasi $P_0$ dan $P_3$
  - Mengaproksimasi $P_1$ dan $P_2$




Wikipedia, Bezier Curve, https://en.wikipedia.org/wiki/B%C3%A9zier_curve

# Bezier Curve

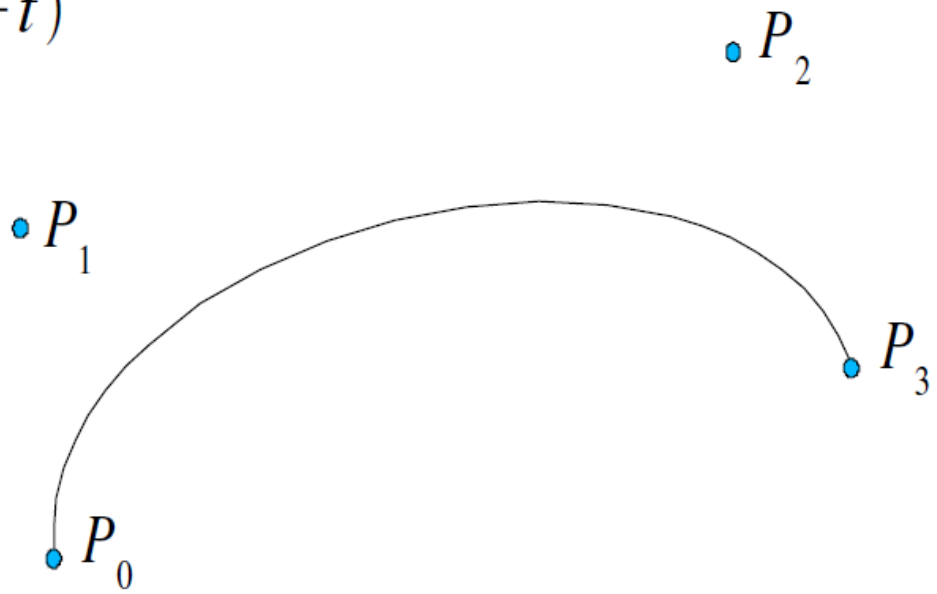
$$B(t)=b_0(t)P_0+b_1(t)P_1+b_2(t)P_2+b_3(t)P_3,\ 0\le t\le 1$$

$$b_0(t)=1-3t+3t^2-t^3=(1-t)^3$$

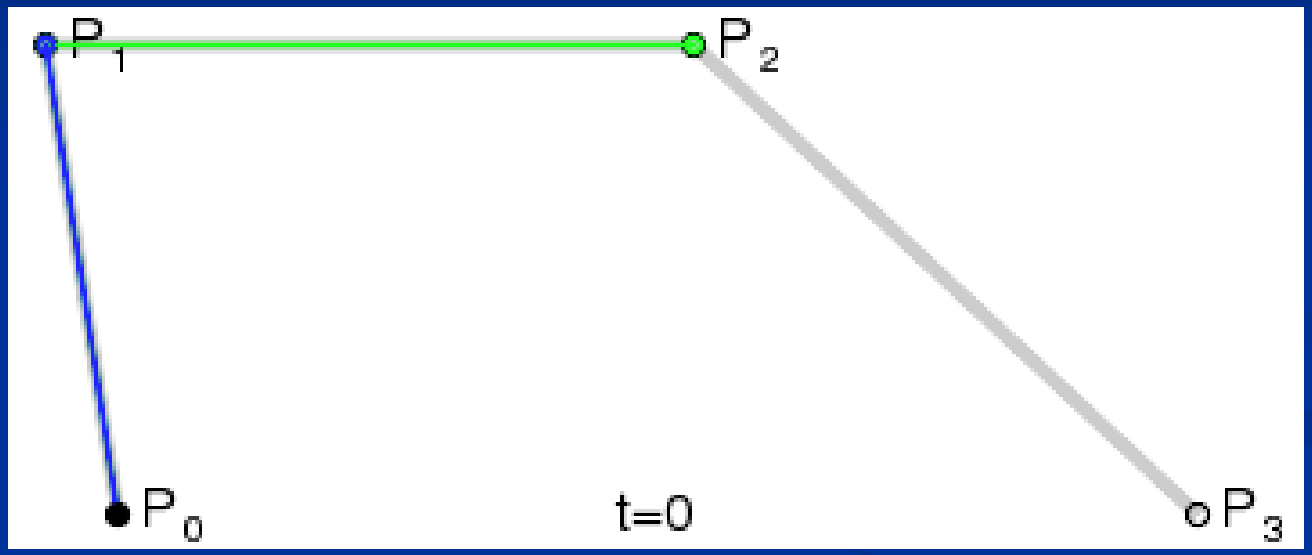
$$b_1(t)=3t-6t^2+3t^3=3t(1-t)^2$$

$$b_2(t)=3t^2-3t^3=3t^2(1-t)$$

$$b_3(t)=t^3$$

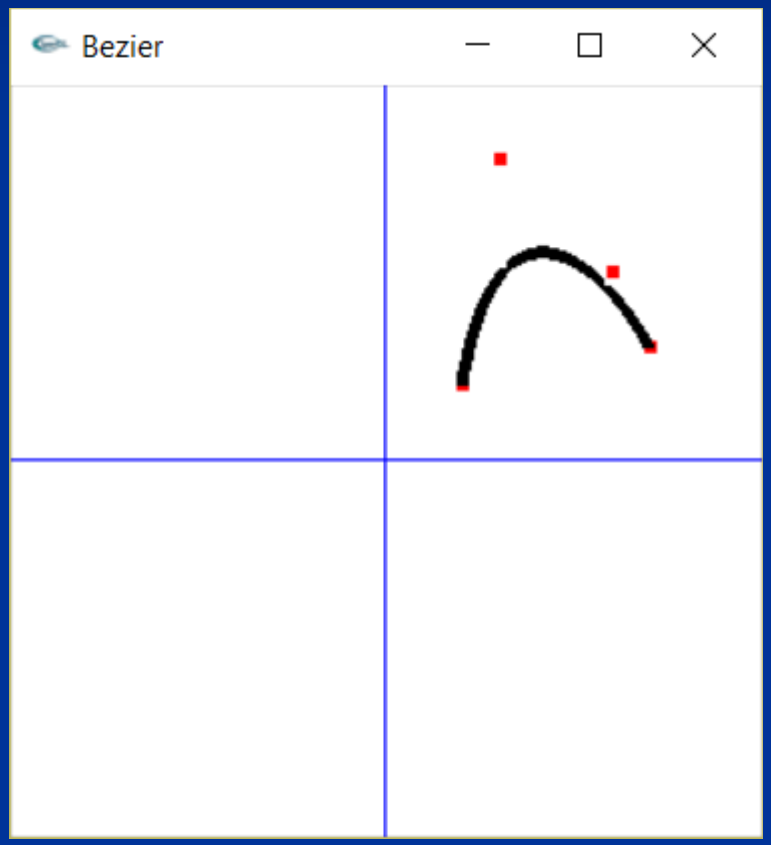


# Bezier Curve



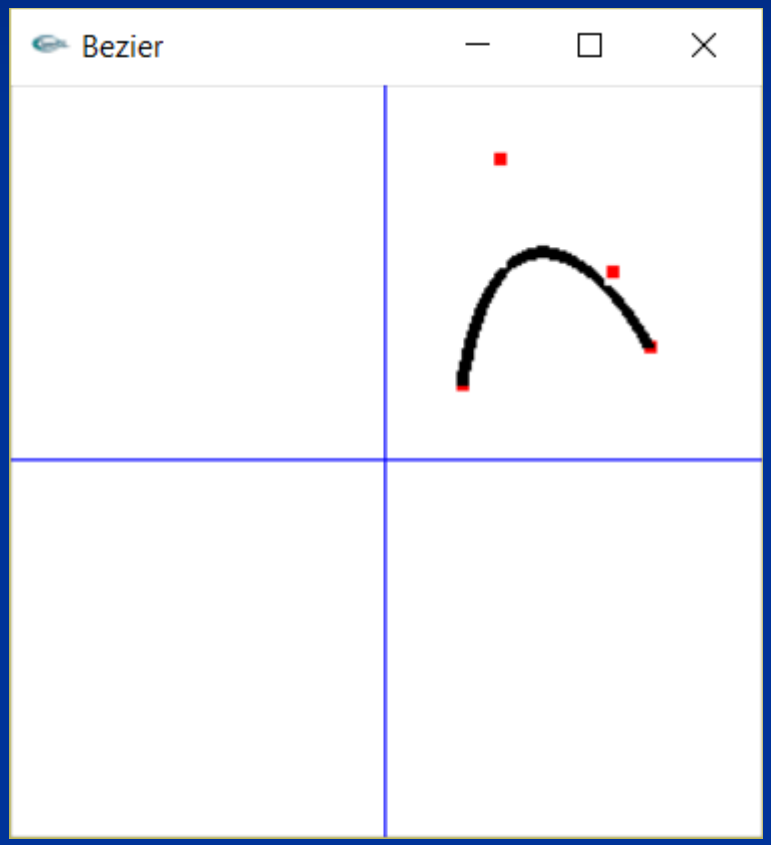

Wikipedia, Bezier Curve, https://en.wikipedia.org/wiki/B%C3%A9zier_curve

# Bezier Curve




```
float b(int i, float t) {
      switch (i) {
      case 0:
        return (1-t)*(1-t)*(1-t);
      case 1:
        return 3*t*(1-t)*(1-t);
      case 2:
        return 3*t*t*(1-t);
      case 3:
        return t*t*t;
      }
      return 0; //mengembalikan nilai
                  0 ketika i tidak valid
    }

float x0=2.0,x1=3.0,x2=6.0,x3=7.0;
float y0=2.0,y1=8.0,y2=5.0,y3=3.0;

float x4=7.0,x5=8.0,x6=9.0;
float y4=2.0,y5=2.0,y6=5.0;
```

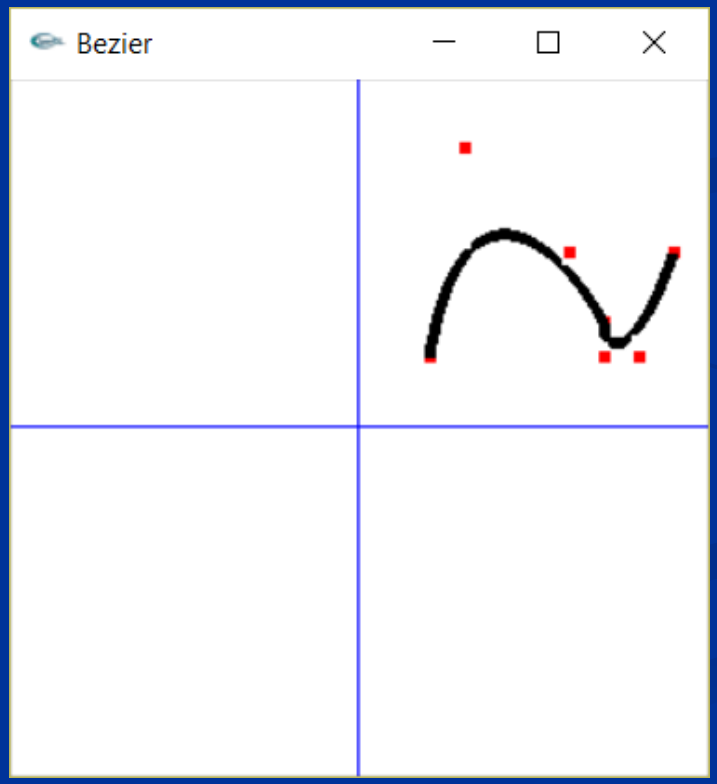
# Bezier Curve




```
glBegin(GL_LINES);
glVertex2i(10,0);
glVertex2i(-10,0);
glVertex2i(0,10);
glVertex2i(0,-10);
glEnd();

glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_STRIP);
for(float t = 0.0; t <=1.0; t+=0.01){
    glVertex2f(b(0,t)*x0+b(1,t)
     *x1+b(2,t)*x2+b(3,t)*x3,
     b(0,t)*y0+b(1,t)*y1+b(2,t)
     *y2+b(3,t)*y3);
}
glEnd();
```

# B-Spline

- Kita bisa menggabung 2 atau lebih kurva Bezier dengan menambahkan 3 titik kontrol berikutnya.
- Namun hasilnya kadang terlihat terputus/tersendat.



# B-Spline

- Punya 4 titik kontrol
- Sifat:
  - Mengaproksimasi semua titik kontrol ($P_0$ , $P_1$ , $P_2$ , dan $P_3$)

# B-Spline

$$B_0(t) = b_0(t)P_0 + b_1(t)P_1 + b_2(t)P_2 + b_3(t)P_3$$

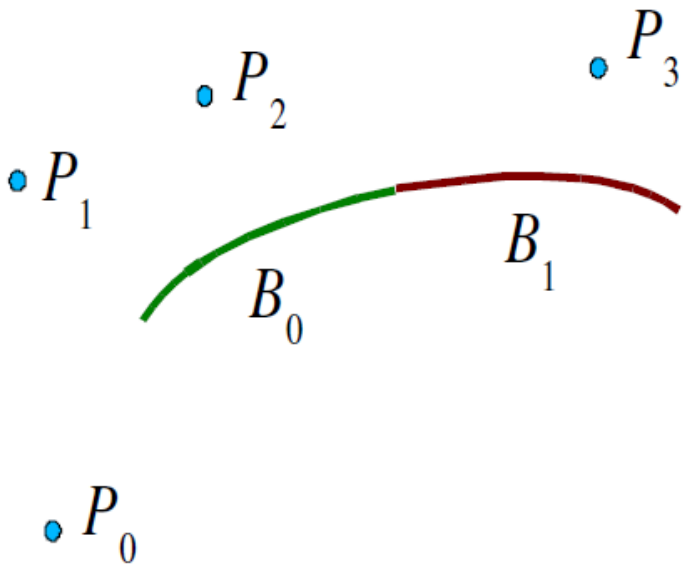
$$B_1(t) = b_0(t)P_1 + b_1(t)P_2 + b_2(t)P_3 + b_3(t)P_4$$

$$b_0(t) = (1 - 3t + 3t^2 - t^3)/6$$

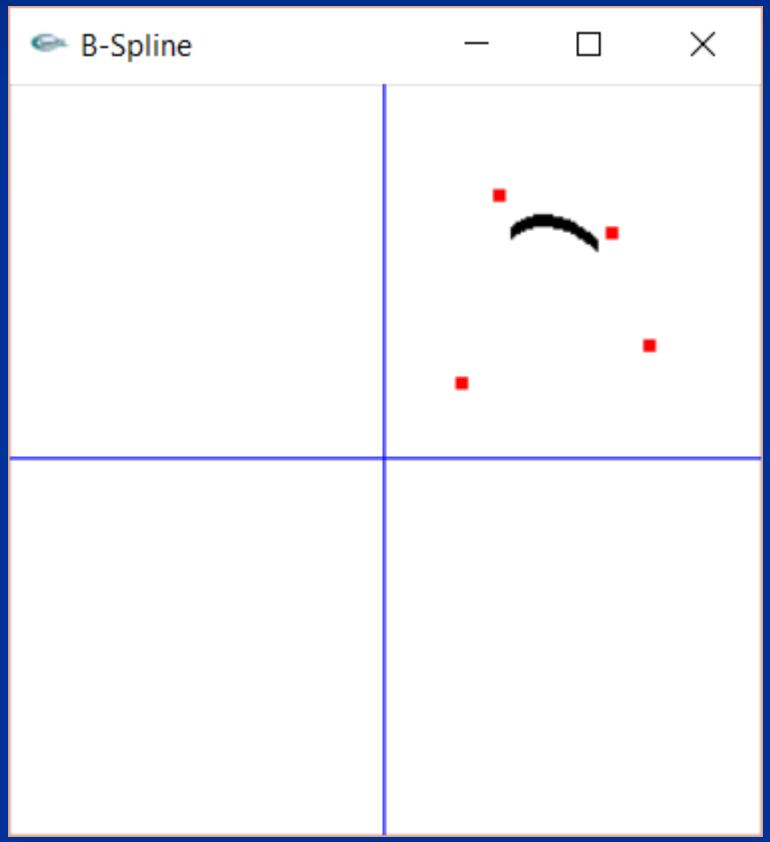
$$b_1(t) = (4 - 6t^2 + 3t^3)/6$$

$$b_2(t) = (1 + 3t + 3t^2 - 3t^3)/6$$

$$b_3(t) = t^3/6$$

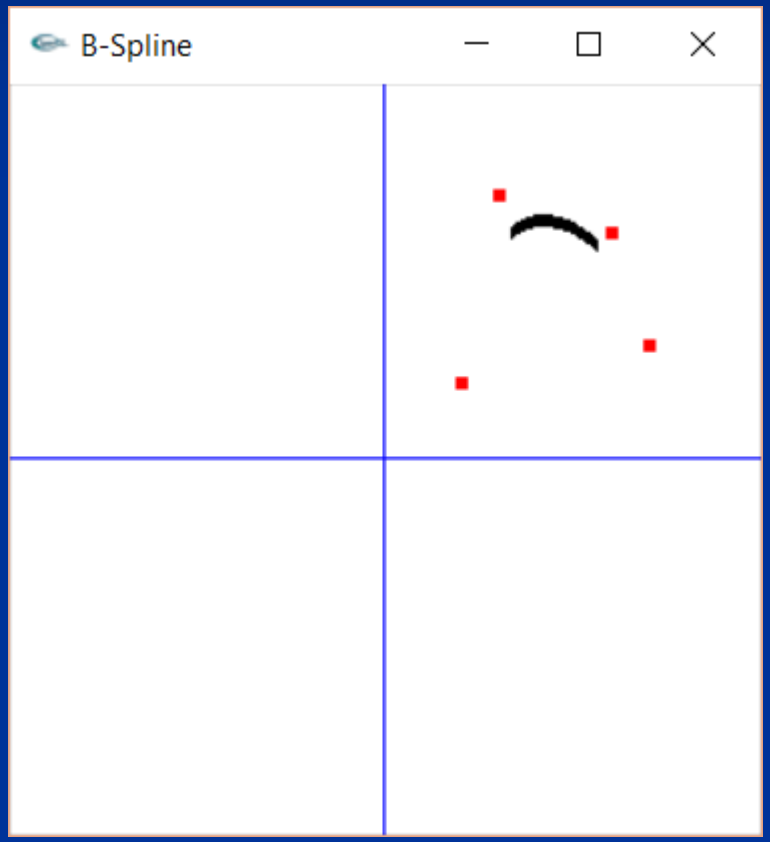


# B-Spline




```
float b(int i, float t) {
      switch (i) {
      case 0:
        return (1.0-3*t+3*t*t-
t*t*t)/6;
      case 1:
        return (4-6*t*t+3*t*t*t)/6;
      case 2:
        return (1+3*t+3*t*t-
3*t*t*t)/6;
      case 3:
        return t*t*t/6;
      }
      return 0; //mengembalikan nilai
0 ketika i tidak valid
}

float x0=2.0,x1=3.0,x2=6.0,x3=7.0;
float y0=2.0,y1=7.0,y2=6.0,y3=3.0;
```

# B-Spline



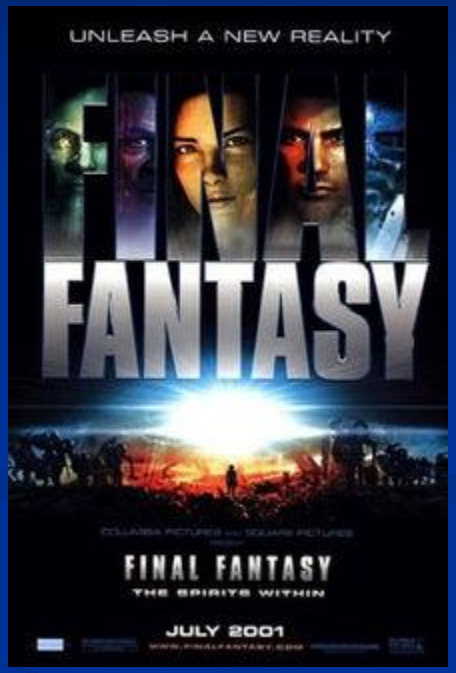

```
glBegin(GL_LINE_STRIP);
for(float t = 0.0; t <=1.0; t+=0.01){
    glVertex2f(b(0,t)*x0+b(1,t)*
      x1+b(2,t)*x2+b(3,t)*x3,
      b(0,t)*y0+b(1,t)*y1+b(2,t)*
      y2+b(3,t)*y3);
}
glEnd();
```

# Animasi Rambut Karakter

1995


1998

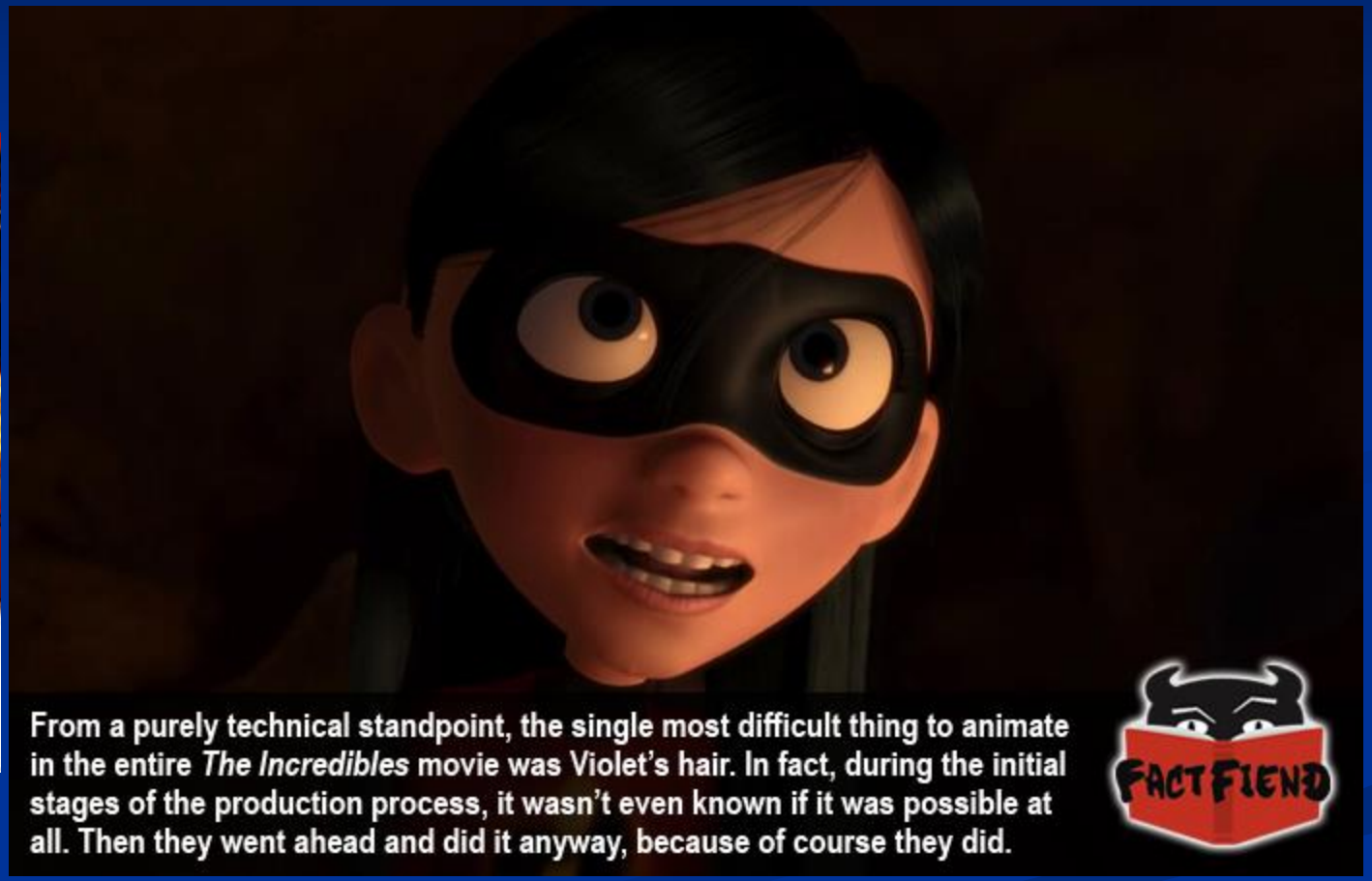

2001




1997

# Animasi Rambut Karakter

2004

# Animasi Rambut Karakter

2010


"TANGLED" Progression sequence illustrating animation process from concept to final frame. Step 5 Final Color 

# Animasi Rambut Karakter

2012


# Relevansi dengan demo 2

# Rangkuman

- Kurva adalah obyek geometri menyerupai garis yang tidak harus lurus.
- Menggambar kurva pada komputer biasanya dilakukan dengan menggunakan pendekatan matematis berbasis persamaan parametrik.

# Referensi


- Wikipedia, Curve, https://en.wikipedia.org/wiki/Curve
- Wikipedia, Parametric Equation, https://en.wikipedia.org/wiki/Parametric_equation
- Wikipedia, Trigonometric Functions, https://en.wikipedia.org/wiki/Trigonometric_functions
- Wikipedia, Bezier Curve, https://en.wikipedia.org/wiki/B%C3%A9zier_curve