

# OBJEK GRAFIK 2 DIMENSI

Achmad Basuki  
Nana Ramadijanti

# Materi

- Definisi Obyek Grafik 2-D
- PolyLine
- Mewarnai Area (FillPolygon)
- Membangun Obyek Grafik 2-D
- Animasi 2-D

# Definisi Obyek Grafik 2-D

- Obyek grafik 2-D adalah sekumpulan titik-titik 2-D yang dihubungkan dengan garis lurus baik berupa polyline, polygon atau kurva
- Obyek grafik 2-D didefinisikan sebagai sekumpulan titik 2-D yang secara komputasi dinyatakan sebagai array 1-D, atau linked-list.

---

Dalam tulisan ini, dibahas obyek grafik 2-D yang dinyatakan sebagai array dan antar titiknya dihubungkan dengan garis lurus (*polyline*)

# Langkah-Langkah Untuk Mendefinisikan Obyek Grafik 2-D

- Mendefinisikan struktur dari titik 2-D (Point2D\_t)
- Mendefinisikan struktur warna (Color\_t)
- Mendefinisikan struktur dari obyek grafik 2-D sebagai array dari titik 2-D (Object2D\_t)

# Mendefinisikan Titik 2-D

```
typedef struct {  
    float x;  
    float y;  
} point2D_t;
```

Definisi ini digunakan bila titik didefinisikan dalam sistem koordinat yang menggunakan bilangan pecahan (*float*)

```
typedef struct {  
    int x;  
    int y;  
} point2D_t;
```

Definisi ini digunakan bila titik didefinisikan dalam sistem koordinat yang menggunakan bilangan bulat (*integer*)

# Mendefinisikan Warna

```
typedef struct {  
    float r;  
    float g;  
    float b;  
} color_t;
```

Warna terdiri dari 3 elemen warna yaitu red (r), green (g) dan blue (b) yang nilainya antara 0 dan 1

Fungsi untuk memberi warna pada obyek grafik:

```
void setColor(color_t col)  
{  
    glColor3f(col.r, col.g, col.b);  
}
```

# Mendefinisikan Obyek Grafik 2-D

Definisi obyek ini dapat dituliskan pada *function* **userdraw** secara langsung dengan menyatakannya sebagai array dari titik 2-D. Sebagai contoh untuk menyatakan obyek shape dapat dituliskan:

```
Point2D_t shape[1000]
```

Untuk menyatakan obyek bunga dapat dituliskan:

```
Point2D_t bunga[360]
```

# PolyLine

Polyline adalah suatu fungsi yang digunakan untuk menggambar objek 2-D yang sudah didefinisikan di depan.

```
void drawPolyline(point2D_t pnt[],int n)
{
    int i;
    glBegin(GL_LINE_STRIP);
        for (i=0;i<n;i++) {
            glVertex2f(pnt[i].x, pnt[i].y);
        }
    glEnd();
}
```

# Polygon

Polygon adalah suatu fungsi yang mirip dengan polyline hanya saja hasilnya adalah kurva tertutup, sedangkan polyline hasilnya kurva terbuka

```
void drawPolygon(point2D_t pnt[],int n)
{
    int i;
    glBegin(GL_LINE_LOOP);
        for (i=0;i<n;i++) {
            glVertex2f(pnt[i].x, pnt[i].y);
        }
    glEnd();
}
```

# FillPolygon

Fungsi ini digunakan untuk mewarnai sebuah polygon dengan warna tertentu

```
void fillPolygon(point2D_t pnt[], int n,
color_t color)
{
    int i;
    setColor(color);
    glBegin(GL_POLYGON);
        for (i=0;i<n;i++) {
            glVertex2f(pnt[i].x, pnt[i].y);
        }
    glEnd();
}
```

# GradatePolygon

Fungsi ini digunakan untuk mewarnai sebuah polygon dengan warna-warna yang bergradiasi dari suatu warna ke warna lainnya

```
void GradatePolygon(point2D_t pnt[], int
n, color_t color)
{
    int i;
    glBegin(GL_POLYGON);
        for (i=0;i<n;i++) {
            setColor(color);
            glVertex2f(pnt[i].x, pnt[i].y);
        }
    glEnd();
}
```

# Membangun Obyek Grafik 2-D

- Membuat obyek grafik 2-D secara langsung.
- Membuat obyek grafik 2-D secara perhitungan matematis.

# Membuat Obyek Grafik 2-D Secara Langsung

Membuat obyek grafik 2-D secara langsung bisa dilakukan pada function `userdraw()` dengan menyatakan secara langsung koordinat titik-titiknya

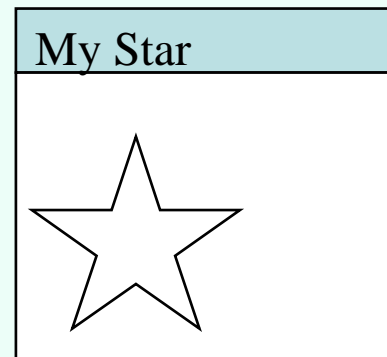
```
void userdraw( )
{
    Point2D_t kotak[4]={{100,100},{300,100},
                       {300,200},{100,200}};
    Polygon(kotak,4);
}
```

Program ini digunakan untuk membuat kotak

# Contoh Membuat Bintang

```
void userdraw( )
{
    Point2D_t bintang[10]={{80,146},{99,90},
        {157,90},{110,55},{128,1},
        {80,34},{32,1},{54,55},
        {3,90},{63,90}};
    Polygon(bintang,10);
}
```

Hasilnya adalah:



# Membuat Obyek Grafik 2-D Dengan Persamaan Matematik

Dengan persamaan matematik  $y=f(x)$  dapat digambarkan kurva dengan variasi bentuk yang menarik seperti sinus, cosinus, exponential dan logaritma, atau fungsi gabungannya. Bentuk persamaan matematik yang menarik untuk dibuat adalah persamaan matematik dengan menggunakan sistem koordinat polar.

$$r = f(\theta)$$

$$x = r \cdot \cos(\theta)$$

$$y = r \cdot \sin(\theta)$$

$\theta$  adalah sudut yang berjalan dari 0 s/d 360 yang dinyatakan dalam radian (0 s/d  $2\pi$ ). Macam-macam  $r=f(\theta)$  dapat menghasilkan gambar yang bervariasi.

# Contoh Fungsi Polar

$r = \sin(\theta)$	Lingkaran
$r = \sin(2\theta)$	Rose 4 daun
$r = \sin(3\theta)$	Rose 3 daun
$r = \sin(n\theta)$	Rose $n$ daun bila $n$ bilangan prima
$r = \theta$	Spiral

**Masih banyak variasi fungsi yang lain yang dapat dibangun dengan menggunakan koordinat polar ini**

# Program Code Membangun Obyek Grafik 2-D Dengan Menggunakan Koordinat Polar

```
void userdraw()  
{  
    Point2D_t shape[360];  
    double srad,r;  
    for(int s=0;s<360;s++)  
    {  
        srad=s*3.14/180;  
        r=sin(5*srad);  
        shape[s].x=(float)(r*cos(srad));  
        shape[s].y=(float)(r*sin(srad));  
    }  
    Polygon(shape,360);  
}
```

Fungsi  $\sin(5\theta)$  yang menghasilkan rose 5 daun.

# Animasi 2-D

- Membuat obyek grafik 2-D menjadi bergerak.
- Animasi yang dilakukan adalah memindahkan posisi gambar.
- Pada sistem koordinat kartesian animasi akan berefek gerakan linier (translasi), pada sistem koordinat polar akan berefek gerakan berputar (rotasi).

# Pembuatan Animasi 2-D

- Pada `main()` ditambahkan fungsi `glutIdleFunc(display)` sebelum fungsi `glutDisplayFunc(display)`.
- Pada awal fungsi `userdraw()` didefinisikan `static int tick`
- Pada akhir fungsi `userdraw()` ditambahkan perintah untuk menambah nilai `tick` secara terus menerus dengan `tick++`.
- Tambahkan nilai `tick` ini pada nilai variabel dasar pembuatan grafik.

# Program Code Animasi 2-D Menggunakan Koordinat Polar

```
void userdraw( )
{
    static int tick=0;
    Point2D_t shape[360];
    double srad,r;
    for(int s=0;s<360;s++)
    {
        srad=(s+tick)*3.14/180;
        r=sin(5*srad);
        shape[s].x=(float)(r*cos(srad));
        shape[s].y=(float)(r*sin(srad));
    }
    Polygon(shape,360);
    tick++;
}
```

# Animasi Lebih Lanjut

- Animasi lebih lanjut akan dipelajari pada materi Transformasi 2-D, dimana animasi dapat dilakukan dengan sangat mudah.
- Program yang dibangun dengan menggunakan Transformasi 2-D ini akan menjadi lebih user-friendly karena setiap perintahnya dapat dimengerti dengan mudah.