GRAFGAME

GRAFIKA GAME

Aditya Wikan Mahastama
mahas@ukdw.ac.id

HTML5 dan Primitif Grafika

3

Before we

begin to draw

THE CONCEPT OF CANVAS

- Sebuah bidang yang akan digunakan dalam mem-*plot* (menggambar) grafika komputer, disebut dengan **kanvas**
- Kanvas dibatasi oleh margin kiri-kanan-atas-bawah, dan batas kanvas ditentukan sebelum mulai menggambar
- Penggambaran pada koordinat di luar batas kanvas, tidak akan muncul pada kanvas, tetapi kalkulasi titik di luar kanvas masih dimungkinkan

THE CONCEPT OF CANVAS

- Setiap bahasa pemrograman berbasis visual biasanya memiliki kanvas, meskipun tertuang pada jenis komponen yang berbeda
- Komponen untuk kanvas pada umumnya berada pada posisi yang berbeda dari komponen untuk menampung citra yang dibuka dari file
- Manipulasi piksel secara bebas hanya dapat dilakukan pada komponen kanvas, tidak pada komponen penampung citra

THE CONCEPT OF CANVAS

Ilustrasi:

- Visual Basic memiliki dua komponen grafika: TPicture (untuk menampung citra dari file), dan TImage (untuk menampung kanvas)
- Delphi memiliki satu komponen grafika TImage saja, dengan tingkatan kelas yang berbeda

  TImage.TCanvas sebagai container kanvas

  TImage.TPicture sebagai container citra dari file

WHY I'M USING HTML5

Some of the most interesting new features in HTML5:

- The <canvas> element for 2D drawing
- The <video> and <audio> elements for media playback
- Mudah, tidak perlu instalasi yang rumit, cukup memiliki browser terbaru saja
- Familier, dan hanya perlu sedikit tambahan pengetahuan mengenai JavaScript

courtesy: http://www.w3schools.com/html/html5_intro.asp

KEBUTUHAN HTML5

- Anda hanya membutuhkan *web browser* yang mendukung HTML5

- Cek kompatibilitas browser anda dengan HTML5 di http://html5test.com/ dan pastikan bagian Canvas terpenuhi, untuk bisa mulai menggambar

KANVAS DI HTML5

- Elemen HTML5 Canvas adalah sebuah tag HTML yang serupa dengan , <a>, atau <table>, dengan pengecualian bahwa baris-baris di dalam tag berisi skrip JavaScript.
- In order to leverage the HTML5 Canvas, we'll need to:
  1. place the canvas tag somewhere inside the HTML document
  2. access the canvas tag with JavaScript
  3. create a context
  4. and then utilize the HTML5 Canvas API to draw visualizations.

KANVAS DI HTML5

- Baris-baris script dasar untuk kanvas HTML5

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-type" content="text/html;charset=UTF-8">
<title>Grafika Game</title>
</head>
<body>
  <canvas id="myCanvas" width="578“ height="200"></canvas>
  <script>
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');

    // tambahkan script penggambaran di sini
  </script>
</body>
</html>
```

KANVAS DI HTML5

- Menggambar garis

```
<body>
  <canvas id="myCanvas" width="578“ height="200"></canvas>
  <script>
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');

      context.beginPath();
      context.moveTo(100, 150);
      context.lineTo(450, 50);
      context.stroke();
</script>
</body>
```

KEKURANGAN KANVAS DI HTML5

- Kanvas di HTML5 menyediakan method untuk menggambar garis, tetai malah tidak bisa menggambar titik.
- Ada beberapa solusi yang ditawarkan misalnya menggambar garis yang sangat pendek.

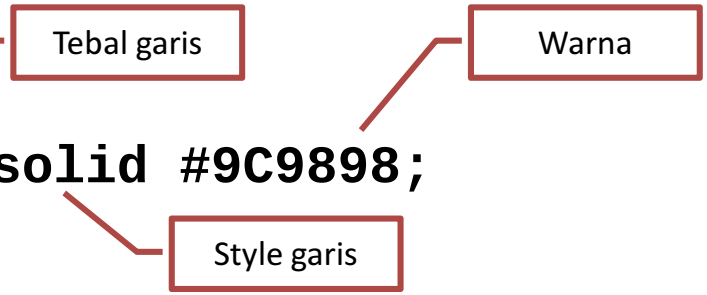
```
context.beginPath();
context.moveTo(2,1);
context.lineTo(3,2);
context.stroke();
```

- Solusi selengkapnya bisa anda lihat di http://html5tutorial.com/how-to-draw-a-point-with-the-canvas-api/

BINGKAI KANVAS

- Kanvas HTML5 dapat dibingkai melalui skrip CSS pada tag <style></style>

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-type" content="text/html;charset=UTF-8">
<title>Grafika Game</title>
</head>
<style>
   #myCanvas {
        border: 1px solid #9C9898;
   }
</style>
<body>…
```



UKURAN KANVAS

- Kanvas HTML5 dapat diganti ukurannya dengan dua cara:

```
<canvas id="myCanvas" width="578“ height="200"></canvas>
```

atau

```
<canvas id="myCanvas" style=“width:578px;
height:200px”></canvas>
```

MULAI MENGGAMBAR

- Kanvas di HTML5 menyediakan method untuk menggambar garis, tetapi malah tidak bisa menggambar titik → semua diperlakukan seperti plotting objek vektor
- Semua objek terdiri dari dua elemen:
  - Garis tepi
  - Isi

  Dengan kondisi minimal harus ada garis tepi saja (pada objek garis)
- Perintah untuk menggambarkan objek-objek dasar yang merupakan turunan primitif grafika, sudah disediakan di kanvas.

MULAI MENGGAMBAR

- Untuk mengatur tebal garis, gunakan atribut lineWidth = n dengan n dalam piksel. Jenis garisnya (tajam atau membulat) kita tentukan dengan lineCap = “butt” | “round” | “square”

```
context.lineWidth = 10;
context.lineCap = "round";
```

- Untuk mengatur warna garis, gunakan atribut strokeStyle = ‘*color’*, dengan *color* berisi warna RGB dalam heksadesimal, atau rgb(R,G,B)

```
context.strokeStyle = ‘#ff0000’;
context.strokeStyle = ‘rgb(255,0,0)’;
```

MENGGAMBAR TITIK

- Solusi titik: menggambar garis yang sangat pendek. Sebagai contoh adalah menggambar sebuah titik berwarna hijau.

```
context.beginPath();
context.moveTo(2,1);
context.lineTo(3,2);
context.lineWidth = 3;
context.lineCap = "round“;
context.strokeStyle = ‘#007700’;
context.stroke();
```

SEDIKIT PENGETAHUAN JAVASCRIPT

- Setiap statement diakhiri dengan ;
- Deklarasi variabel cukup fleksibel, di mana tipe data input pertama kali otomatis akan menentukan tipe data variabel:

```
var a = 5;
var nama = “Mahas";
```

atau

```
var a = 5, nama = “Mahas";
```

atau

```
var a = 5,
nama = “Mahas";
```

SEDIKIT PENGETAHUAN JAVASCRIPT

- Assignment dan comparison sama-sama menggunakan simbol sama dengan (=)
- Struktur perulangan dengan **for**:

```
for (var i=0;i<100;i++) {
        //statement yang akan dikerjakan
}
```

- Struktur perulangan dengan **while** (pre-test loop):

```
var i=1;
while (i<10) {
        //statement yang akan dikerjakan
        i++;
}
```

SEDIKIT PENGETAHUAN JAVASCRIPT

- Struktur perulangan dengan **do** (post-test loop):

```
var i=1;
do {
        //statement yang akan dikerjakan
        i++;
        while (i<10);
}
```

- Struktur percabangan:

```
var first=1;
var second=2;
if (second>first){
        //statement yang akan dikerjakan
}
```

Graphics Primitives

Point & Line

WHAT CONSIDERED PRIMITIVE?

- Sekarang disebut juga Primitif Geometris, merupakan objek geometrik atomis yang paling mungkin digambar dan disimpan oleh sebuah sistem
- Primitif grafika yang paling dasar:
  titik, **garis** dan **segmen garis**
- Primitif grafika berikutnya:
  bangun dua dimensi, elips (termasuk lingkaran), poligon (segitiga, segi empat, dsb), kurva lengkung (*spline curves*)
- Setiap citra yang dapat dijabarkan secara grafika, terdiri dari objek-objek atomis

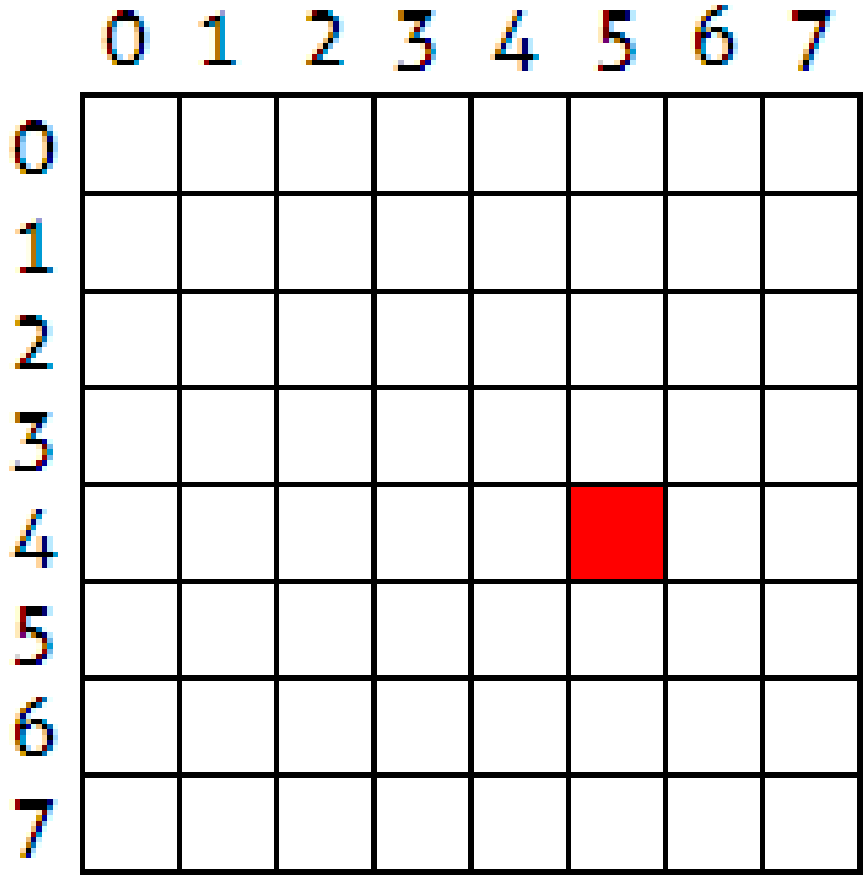
POINT / VERTEX / NODE

A single point on the Cartesian plane

Described as:

A (x,y)

e.g. A (5, 4)



In computer graphics, it is
Represented by a pixel

DISTANCES BETWEEN TWO POINTS

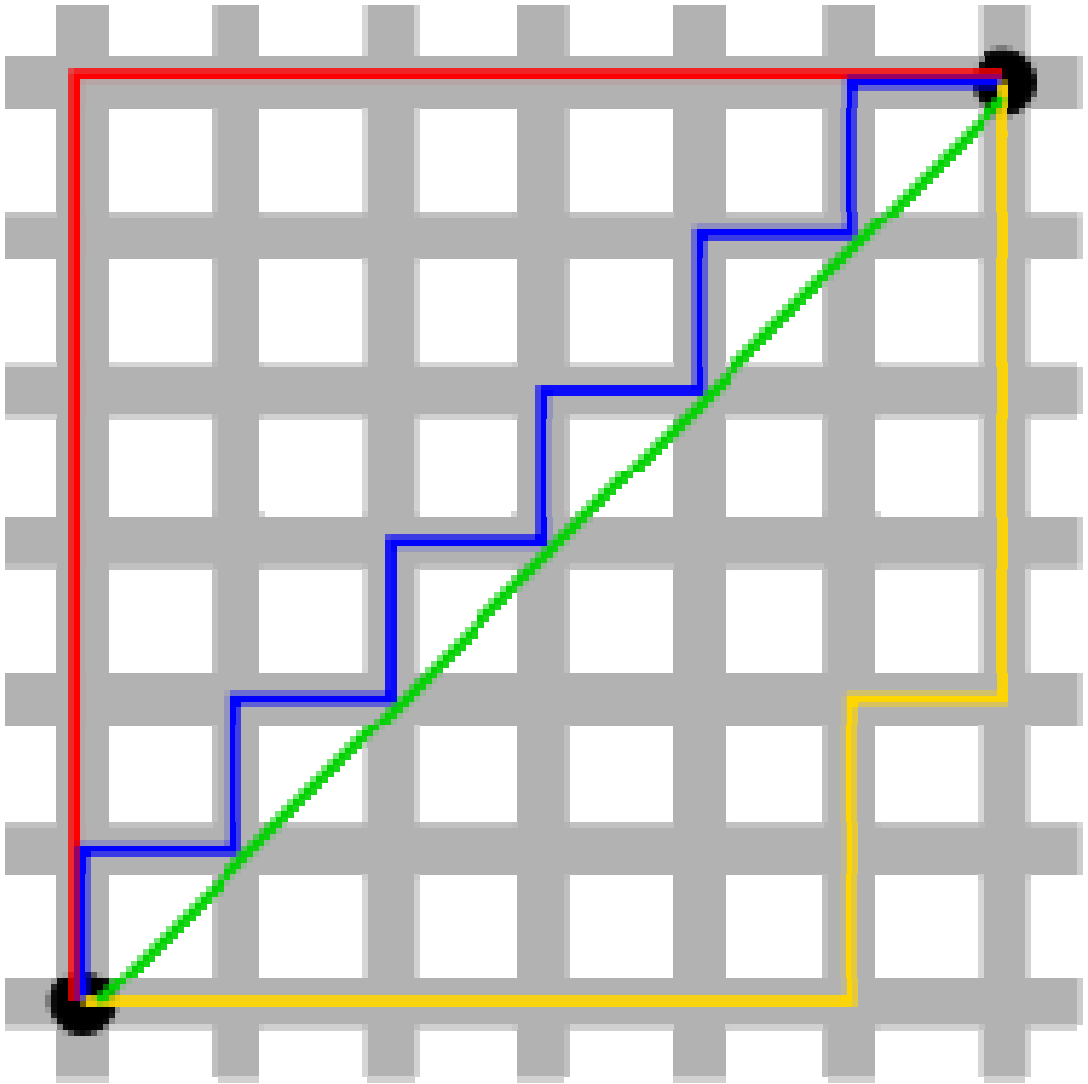
- EUCLIDEAN DISTANCE

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

- MANHATTAN DISTANCE (*CITY BLOCK DISTANCE / TAXICAB GEOMETRY*)

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

DISTANCES BETWEEN TWO POINTS



Green: Euclidean

Others: Manhattan

LINE / EDGE

A sequence of points plotted by a linear equation
Described as

$$y = mx + c$$

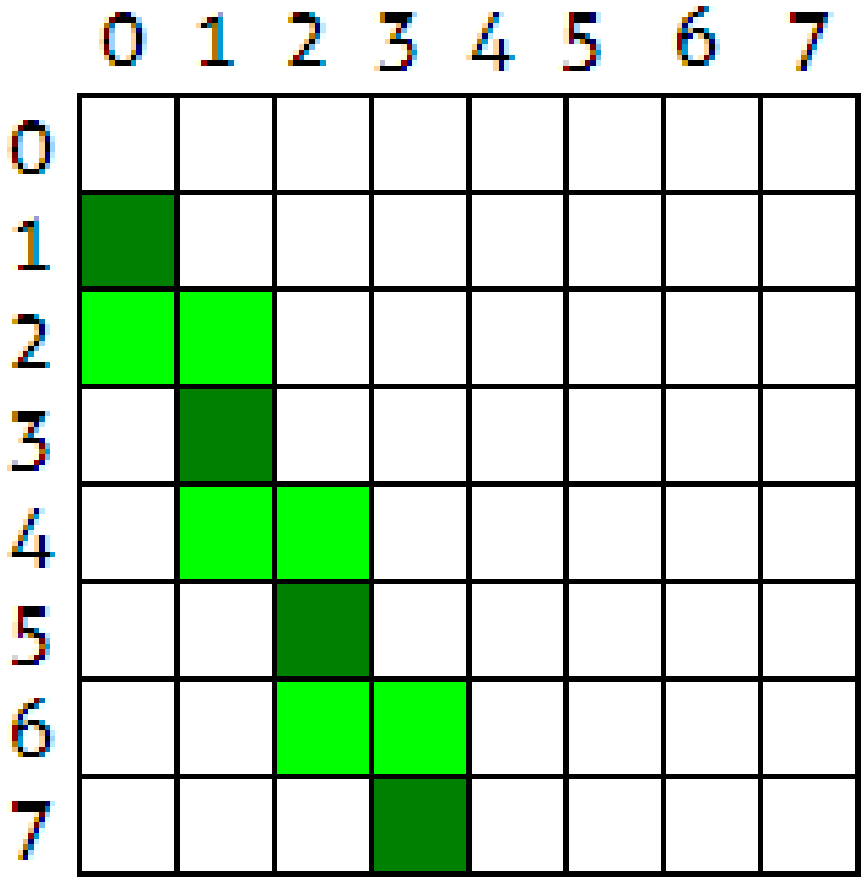
Where

m : slope (gradien)

c : constant number

and x is limited within a range

e.g. $y = 2x + 1, 0 < x < 3$



GRAFGame

LINE / EDGE

A sequence of points plotted by a linear equation Described as

y = mx + c

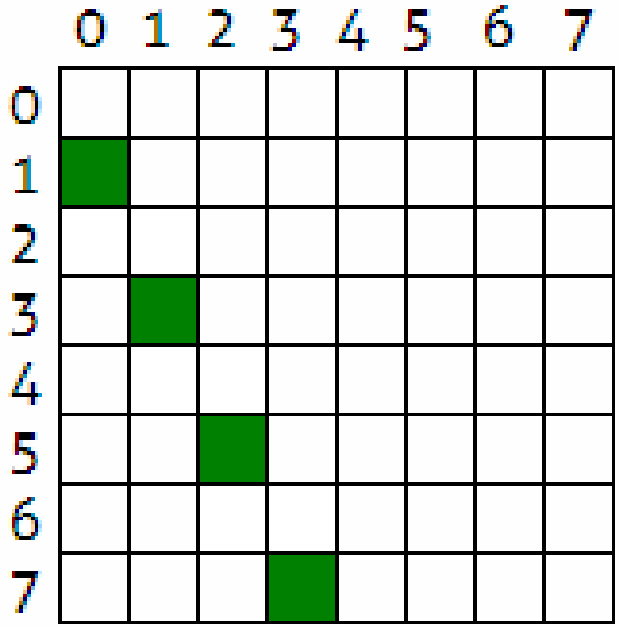
When its drawn, it will have two points at each ends of the line. Therefore we can obtain its slope degree using this equation:

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

$$m = \frac{\Delta y}{\Delta x}$$

LINE PLOTTING PROBLEM

If we plot a line using integer loop, we will have a line that is not continuous



Why? Because a line plot is a linear function, which in fact plots continuous real numbers

So, how we make it a smooth one?

1. REGULAR WAY

For instance, if you want to draw a $y = 3x + 1$ limited by $0 < x < 100$, try this loop

```
var y=0;
for (var x=0; x<100; x++){
        y = (3 * x) + 1;
        context.beginPath();
        context.moveTo(x,y);
        context.lineTo(x+1,y+1);
        context.stroke();
}
```

The bigger you blow up the number, the more accurate it will plot, but it will surely makes up a huge number of redundant plot points

2. MAKE A BIGGER LOOP

In HTML5-Javascript, there’s no need to use a bigger loop, since we’re already drawing using lines, not drawing dots or pixels.

3. DDA ALGORITHM

It has a real cool name: Digital Diferential Analyzer, while what it really did is:

to draw line from known endpoints (x1, y1) and (x2, y2) considering distances and slope trend of both endpoints by:

- adding the right slope for each pairs of a (x,y) repeatedly from (x1,y1) for tstep times
- tstep is either dx or dy, according to the trend

3. DDA ALGORITHM

```
var x1 = 0,      y1 = (3 * x1) + 1,
x2 = 100,        y2 = (3 * x2) + 1;

var dx = x2-x1;          //manhattan distance of x
var dy = y2-y1;          //manhattan distance of y

if (Math.abs(dx)>Math.abs(dy)){ var tstep=Math.abs(dx);
} else { var tstep=Math.abs(dy); }
var add_x = dx / tstep;
var add_y = dy / tstep;
var x = x1, y = y1;              //draw beginning at x1,y1
context.beginPath();
context.moveTo(x,y);
context.lineTo(x+1,y+1);
context.stroke();
for(var i=1; i<tstep; i++){
        x= x + add_x;
        y= y + add_y;
        context.beginPath();
        context.moveTo(Math.floor(x), Math.floor(y));
        context.lineTo(Math.floor(x)+1, Math.floor(y)+1);
        context.stroke();
}
```

4. BRESENHAM ALGORITHM

It is used to draw a line with a positive slope, when the slope value is less than 1

For such a slope, Bresenham said that the equation is

$$y = m(x_k + 1) + c$$

and comes those really buzzing equations (read by yourself in the book!), so that a decisive point P can be obtained, which is started with

$$P_0 = 2\Delta y - \Delta x$$

4. BRESENHAM ALGORITHM

```
x1 := 0;      y1 := round(0.5 * x1) + 1;
x2 := 10;     y2 := round(0.5 * x2) + 1;

dx := x2-x1;        //manhattan distance of x
dy := y2-y1;        //manhattan distance of y

p := (2*dy) - dx;

if x1 > x2 then
begin
      x := x2;      y := y2;
      xend := x1;
end else
begin
      x := x1;      y := y1;
      xend := x2;
end;
```

4. BRESENHAM ALGORITHM

```
//draw beginning at x,y
image1.canvas.pixels[x,y] := clRed;
for i:= x to xend do
begin
        x:= x + 1;
        if p < 0 then p := p + (2*dy)
        else
        begin
                y := y + 1;
                p := p + (2*(dy-dx));
        end;
        image1.canvas.pixels[x,y] := clRed;
end;
```

5. NATIVE HTML5 KEYWORD

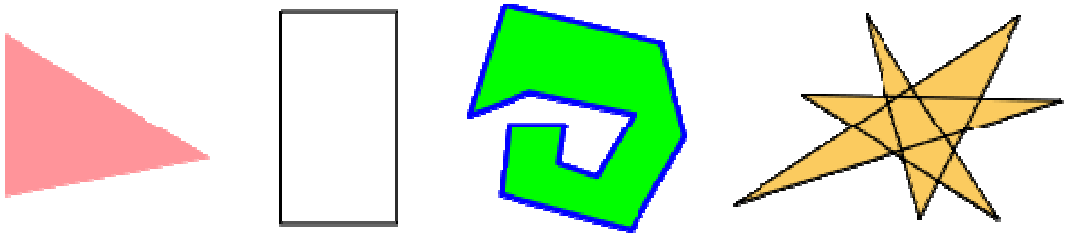
It is already known.

Primitives Descendant

Polygon & Polyline

WHAT IS POLYGON?

- Sebuah bangun datar yang dibatasi oleh sirkuit (jalur tertutup) berupa sejumlah segmen garis lurus yang saling bersambung
- Segmen-segmen garisnya disebut dengan sisi (*sides*), dan pertemuan dua segmen disebut dengan pojok (*corners*)
- Bahasa Indonesia: POLIGON

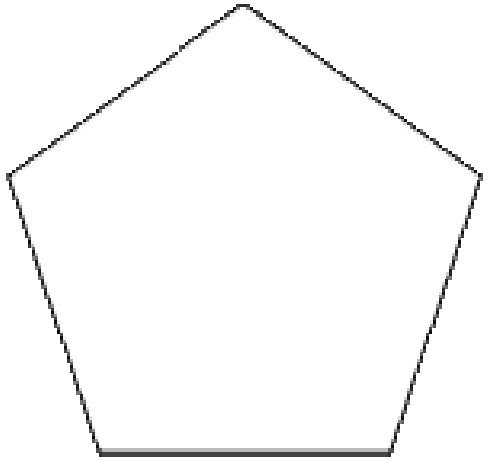


WHAT IS POLYLINE?

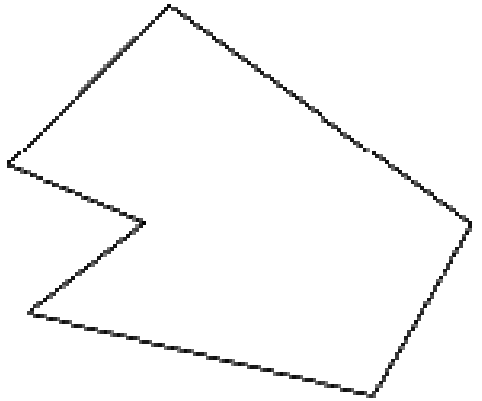
- Sejumlah segmen garis lurus yang saling bersambung yang tidak membentuk sirkuit tertutup
- Istilah ini tidak resmi digunakan
- Bahasa Indonesia: tidak ada

JENIS – JENIS POLIGON

- **KONVEKS (*CONVEX*)**
  Jika besar setiap sudut internalnya kurang dari 180 derajat dan letak sisi-sisinya berada di dalam atau sama dengan titik terluar poligon

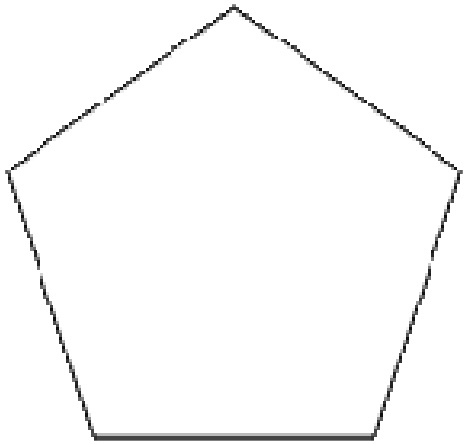
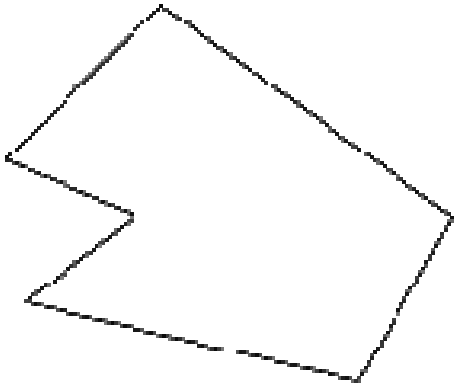
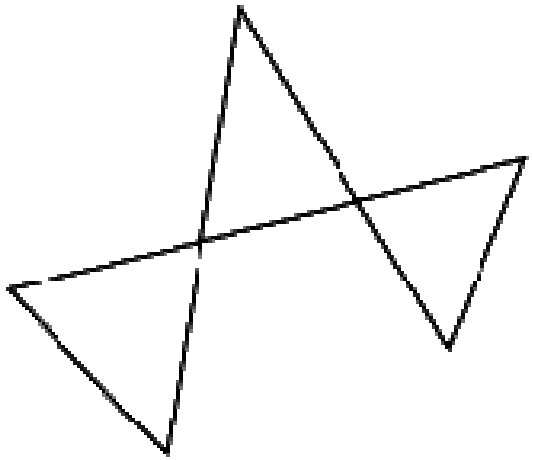


- **KONKAF (*CONCAVE*)**
  Jika besar sudut internalnya ada yang melebihi 180 derajat



JENIS – JENIS POLIGON

- **SIMPEL (*SIMPLE*)**
  Jika tidak ada segmen garis yang berpotongan dalam poligon. Disebut juga **poligon Jordan**
- **NON-SIMPEL (*NON-SIMPLE*)**
  Jika ada segmen garis yang saling berpotongan

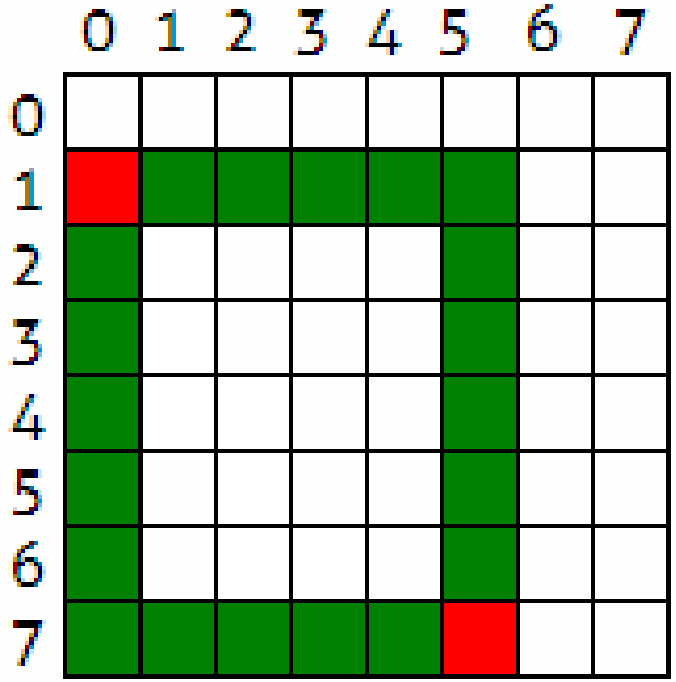
JENIS – JENIS POLIGON

- **EQUIANGULAR**
  Jika besar tiap-tiap sudut pada pojoknya sama besar
- **CYCLIC**
  Jika semua pojoknya terletak dalam sebuah lingkaran
- **EQUILATERAL**
  Jika semua sisi-sisinya sama panjang

WELL, THEN, HOW TO DRAW A BOX?

Given two endpoints as one and the other corner, separated diagonally

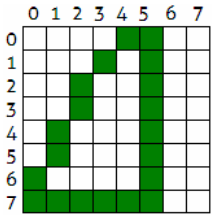
(x1, y1) = (0, 1)
(x2, y2) = (5, 7)



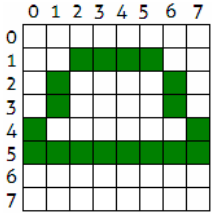

```
context.beginPath();
context.moveTo(x1,y1);
context.lineTo(x2,y1);
context.lineTo(x2,y2);
context.lineTo(x1,y2);
context.lineTo(x1,y1);
context.stroke();
```

YOU'RE SMART!

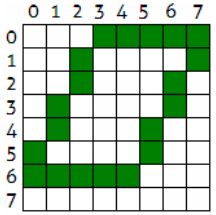
Well, I suppose now you can figure out the way to draw these objects too:



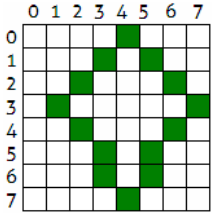
Triangle – with three known points



Trapesium – with four known points



Parallelepipedum – with four known points



Kite – with four known points