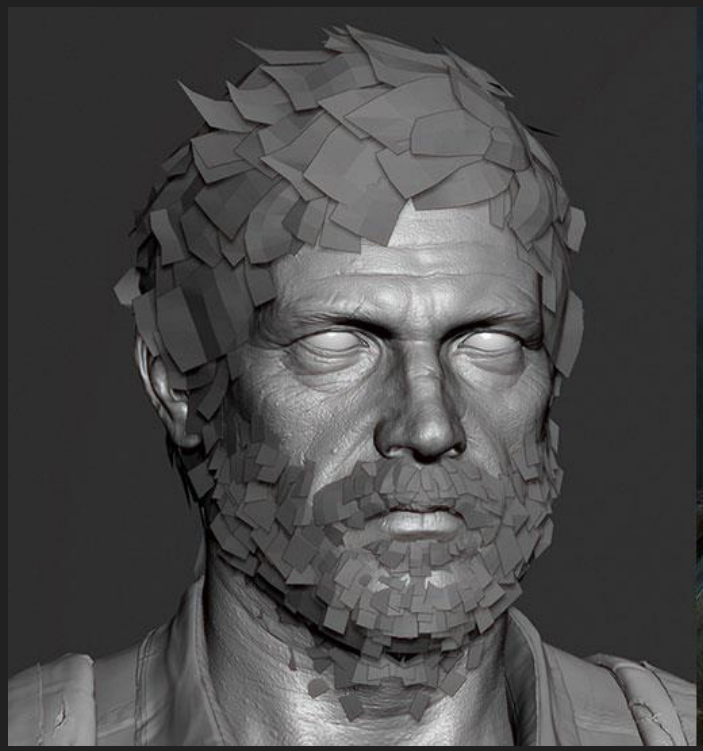
Fundamentals of Computer Graphics and Image Processing

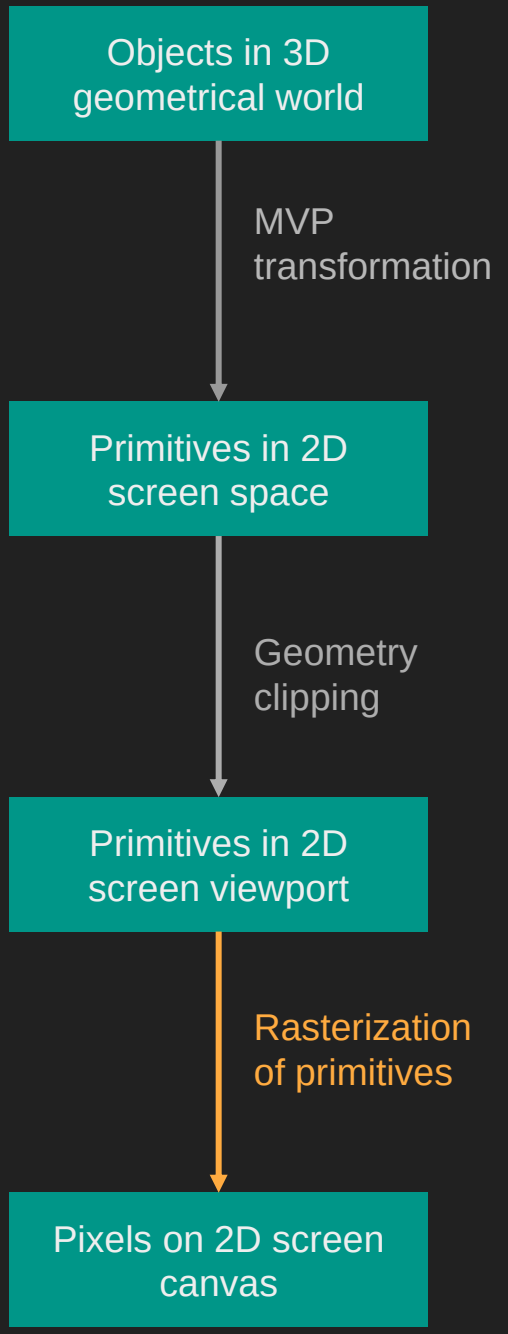
Computer Graphics - Exercise #03

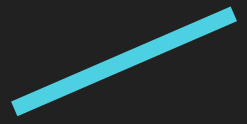
Rasterization

Comes after transformation and clipping stages

Followed by other actions (visibility testing, texturing, lighting, alpha blending) which determine final color of a pixel



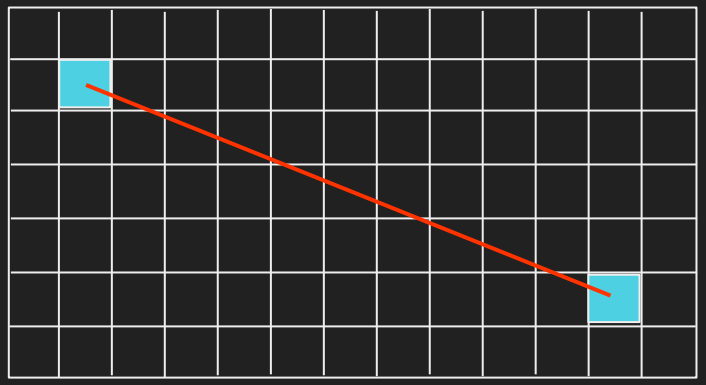




Line Rasterization

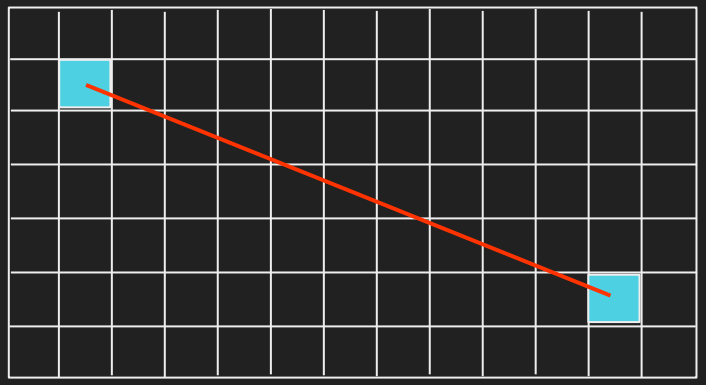
Problem definition

vector

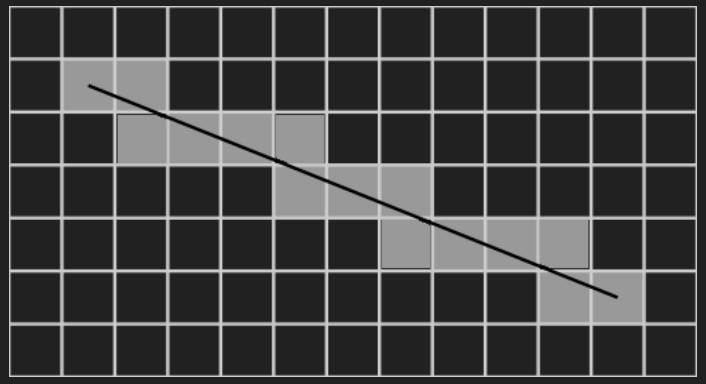


Problem definition

vector



raster



Line Rasterization Algorithm Comparison

Used for rasterization of a **line**

Digital Differential Analyzer - DDA

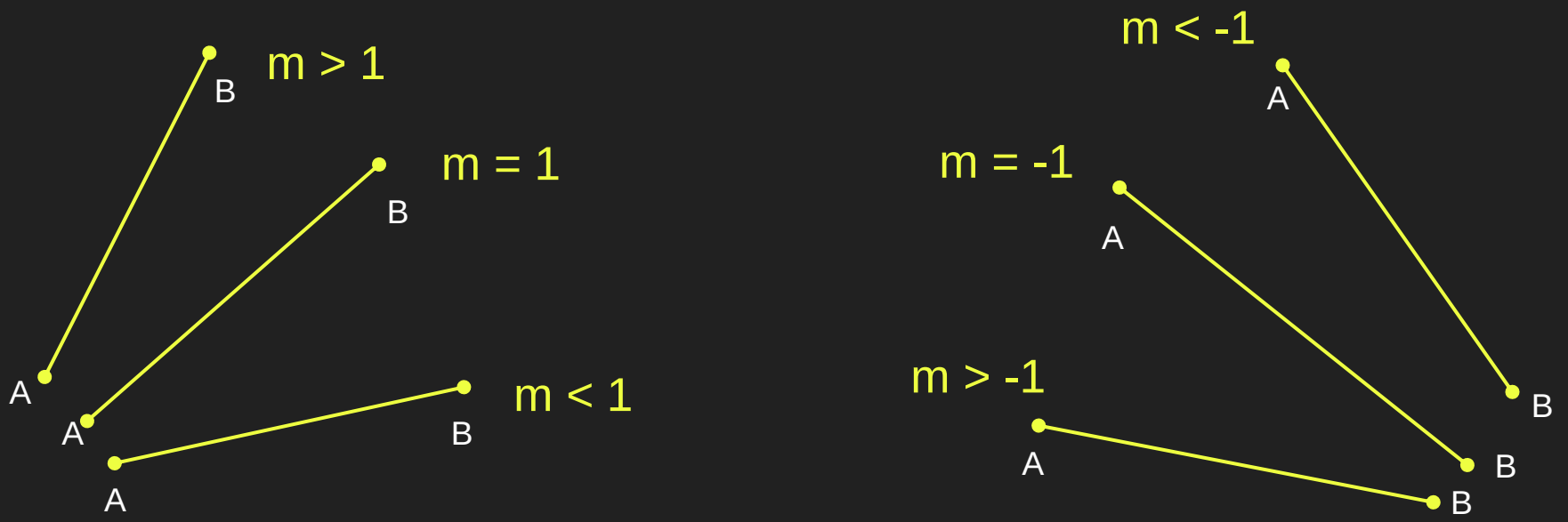
- Using floating point arithmetics and rounding (slow)
- Simplest algorithm
- Less efficient
- Less accurate

Bresenham’s algorithm

- Integer arithmetics only - avoid floating point arithmetics
- Incremental algorithm: current value uses previous value

Digital Differential Analyzer - DDA

1) From line ending points A and B calculate slope m
   $m = (B_y - A_y) / (B_x - A_x)$



Digital Differential Analyzer - DDA

1) From line ending points A and B calculate slope m
   $m = (B_y - A_y) / (B_x - A_x)$

2) Test line slope and modify input if necessary
   IF **abs(m) > 1**
   THEN **exchange x and y** in 5)
   m = 1 / m

3) Set **[x1, y1]** to the point with lower **x** coordinate and **[x2, y2]** to the other

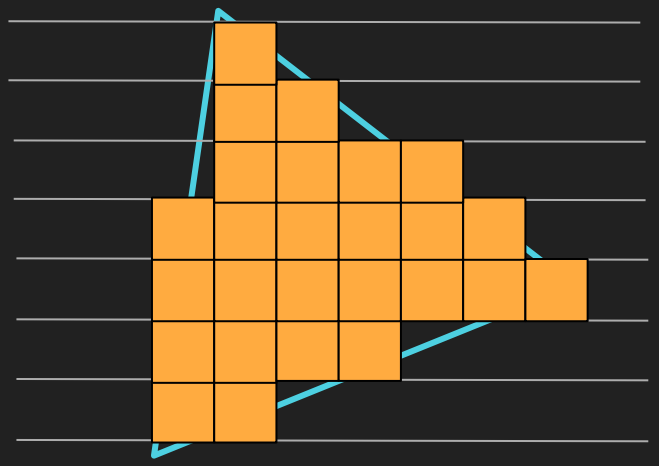
4) Initialize variables **x** and **y** as: **[x, y] = [x1, y1]**

5) WHILE **$x \leq x2$** DO
   Draw point **[x, round(y)]** (or **[round(x), y]**, depending on test in 2)
   $x = x + 1$
   $y = y + m$

Polygon Rasterization

Use line rasterization for unfilled polygons,

or scan-line algorithm.



Problem of Texturing

A point [x,y] on a screen have also texture coordinate [u,v]

The point should be rendered using a color from the texture

Determine texture color which corresponds to [u,v]

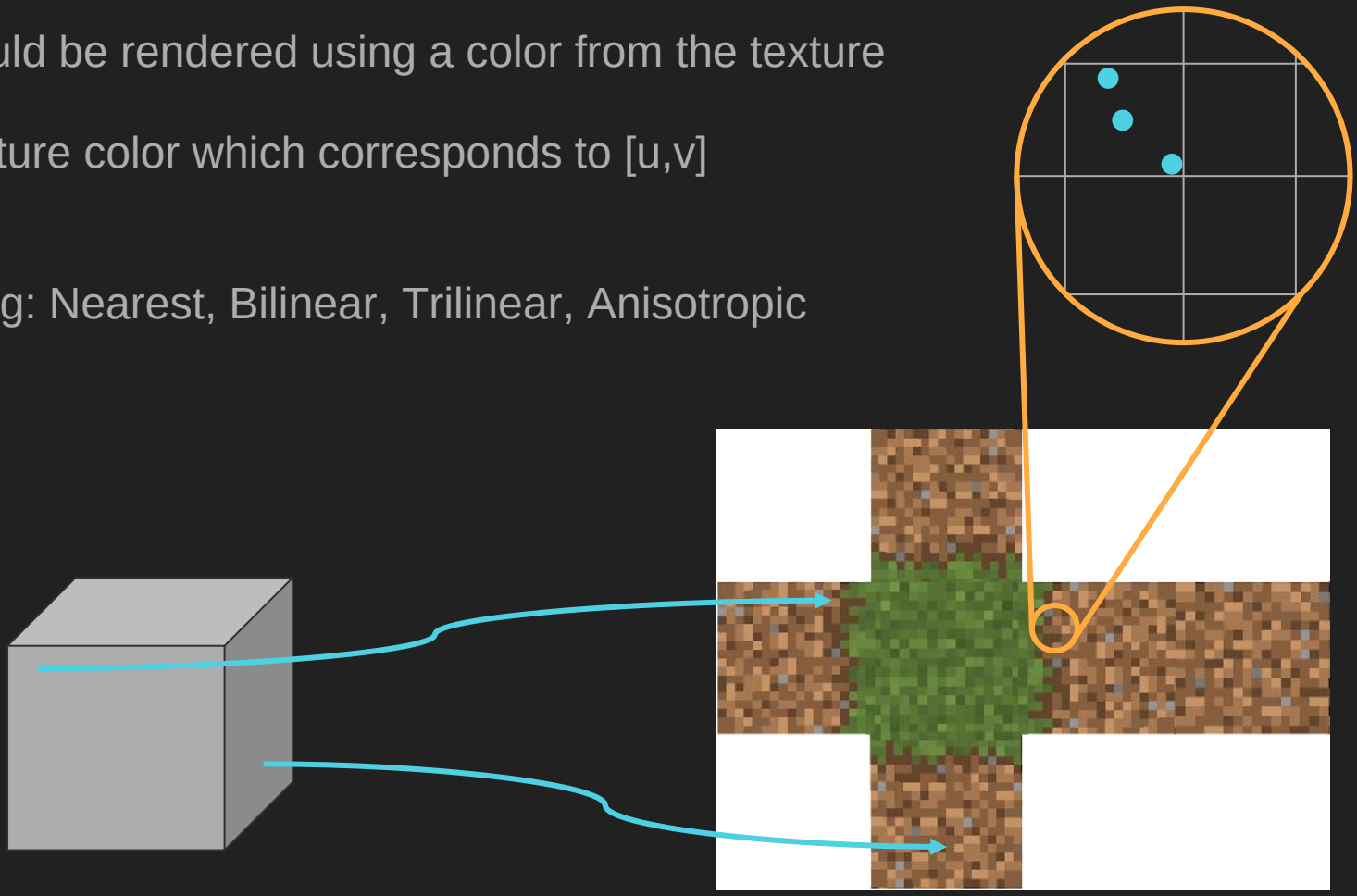
Problem of Texturing

A point [x,y] on a screen have also texture coordinate [u,v]

The point should be rendered using a color from the texture

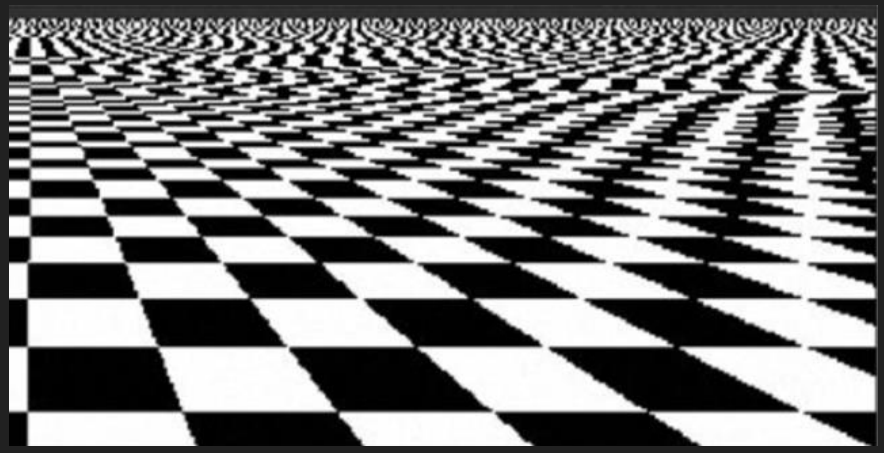
Determine texture color which corresponds to [u,v]

Texture filtering: Nearest, Bilinear, Trilinear, Anisotropic

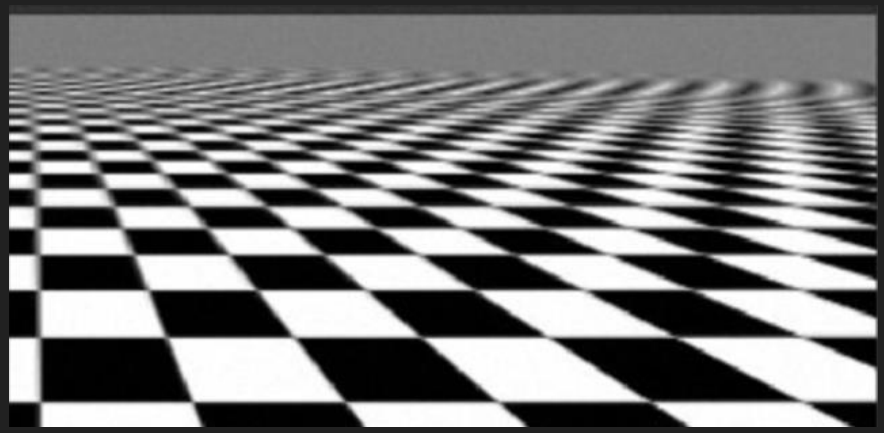


Texture filtering

Nearest neighbor



Bilinear



Texture filtering: Nearest, Bilinear, Trilinear, Anisotropic

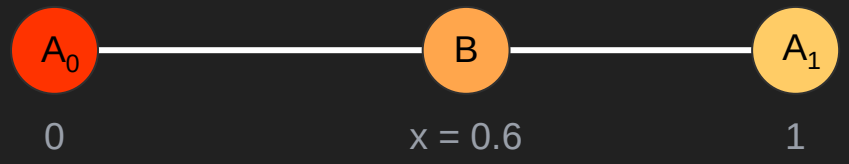
Linear Interpolation

Knowing the values of two points on one axis, how to determine a value of a point located somewhere in between?

$$B = A_0 + (x - x_0)\frac{(A_1 - A_0)}{(x_1 - x_0)}$$

In case that $x_0=0$ and $x_1=1$:

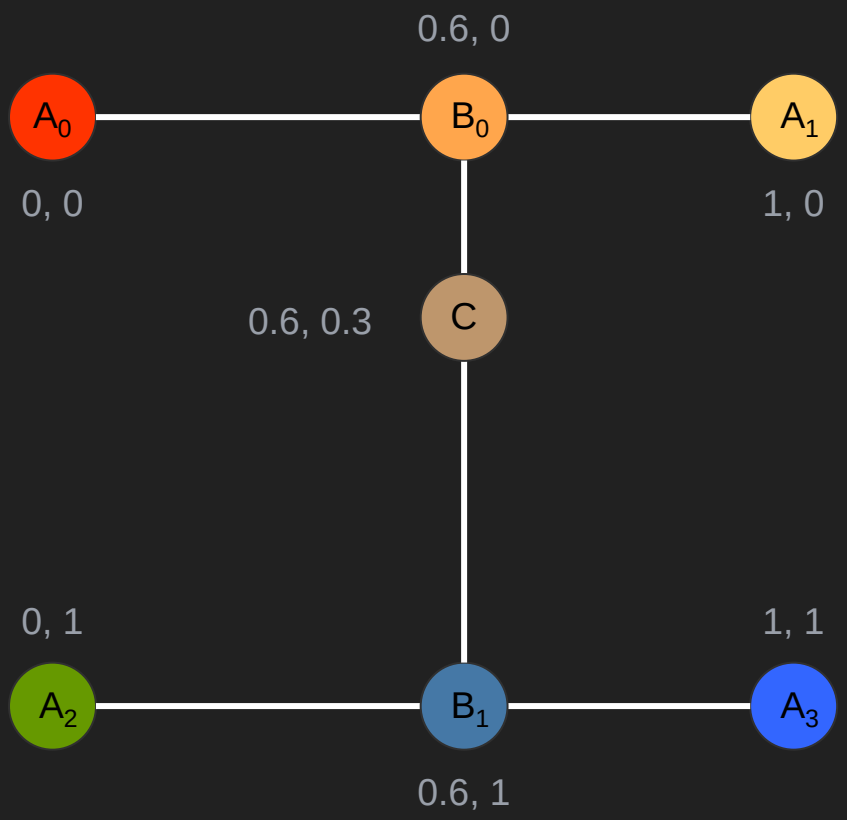
$$B = (1 - x)A_0 + xA_1$$

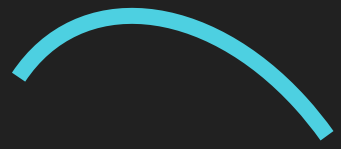


Bilinear Interpolation

Extension of linear interpolation for interpolating values over two axes

Interpolates the values over one axis, then interpolate results over other

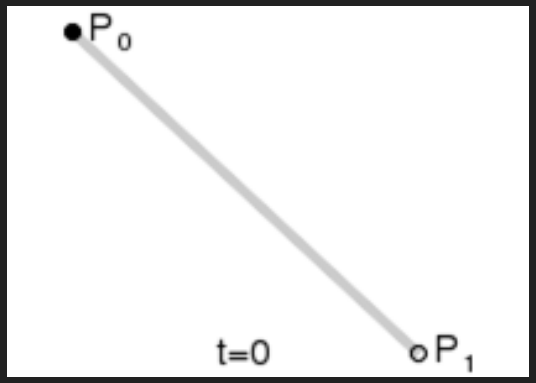




Curve Rasterization

Bézier Curves

Parameter t = <0,1>

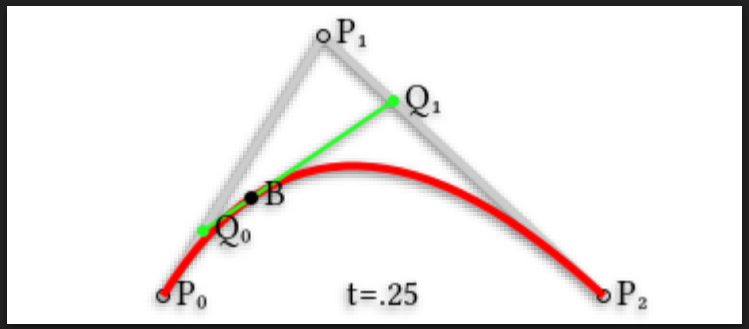


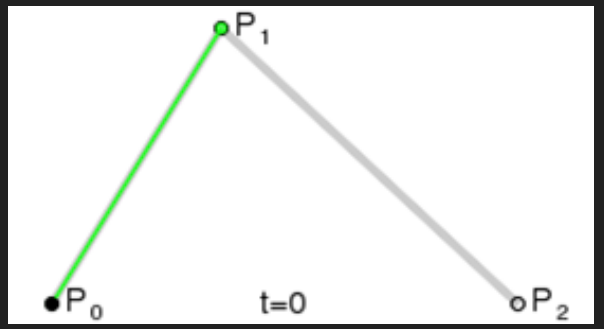
Linear Bézier curve

$$B(t) = (1-t)P_0 + tP_1$$

Quadratic Bézier curve

$$B(t) = (1-t)^2P_0 + 2(1-t)tP_1 + t^2P_2$$

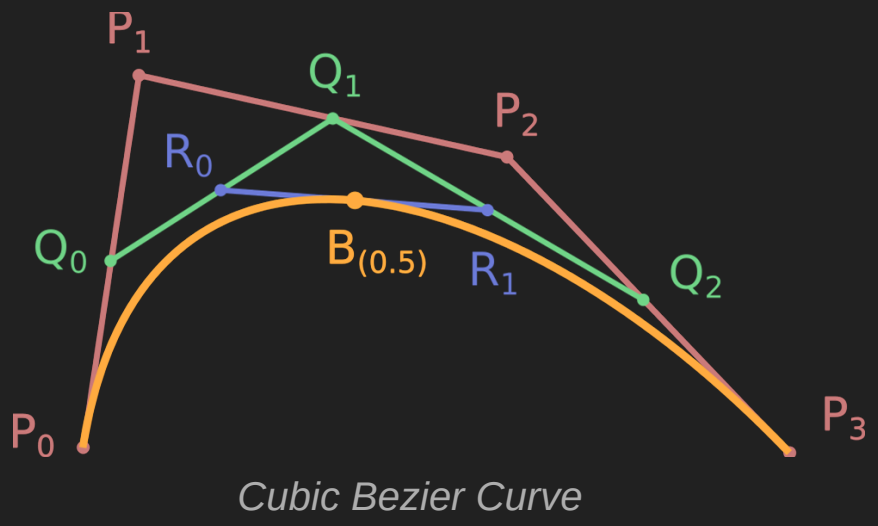




Animated Bézier Curves

Bézier Curves

Parameter t = <0,1>



Cubic Bezier Curve

Linear Bézier curve

$$B(t) = (1-t)P_0 + tP_1$$

Quadratic Bézier curve

$$B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2$$

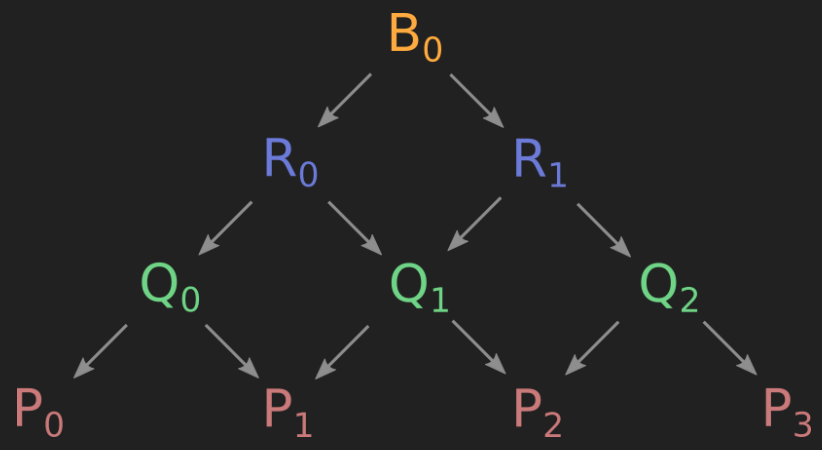
Bézier curve of nth grade

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i$$

[Animated Bézier Curves]

De Casteljau's algorithm

Finds a curve point B for a specified parameter t




```
INPUT
    t
    P0, P1, P2 …, Pn

FUNCTION DeCasteljau(k,i)
BEGIN
    IF k = 0 THEN
        RETURN Pi
    ELSE
        RETURN (1-t)* DeCasteljau(k-1,i) + t* DeCasteljau(k-1,i+1)
END

B(t) = DeCasteljau(n, 0)
```

Rasterizing Bézier Curves

1) Sample interval <0,1> to get curve parameters **(t_0, t_1, t_2..., t_n)**
   - More samples will result in finer curve

2) Use De Casteljau's algorithm to compute points for all parameters **($B_k = B(t_k)$)**

3) Rasterize lines connecting each pair of points B_k and B_{k+1}