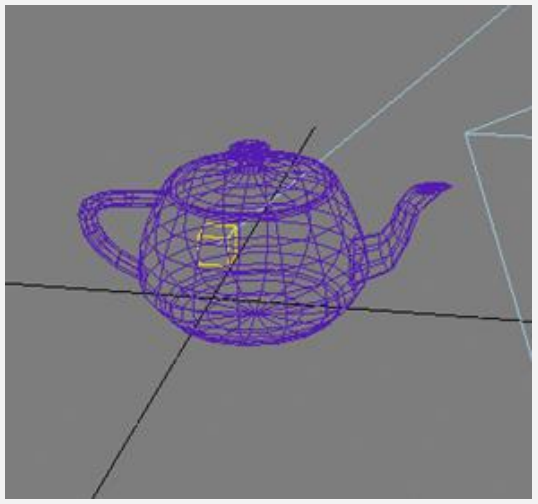
Rendering pipeline

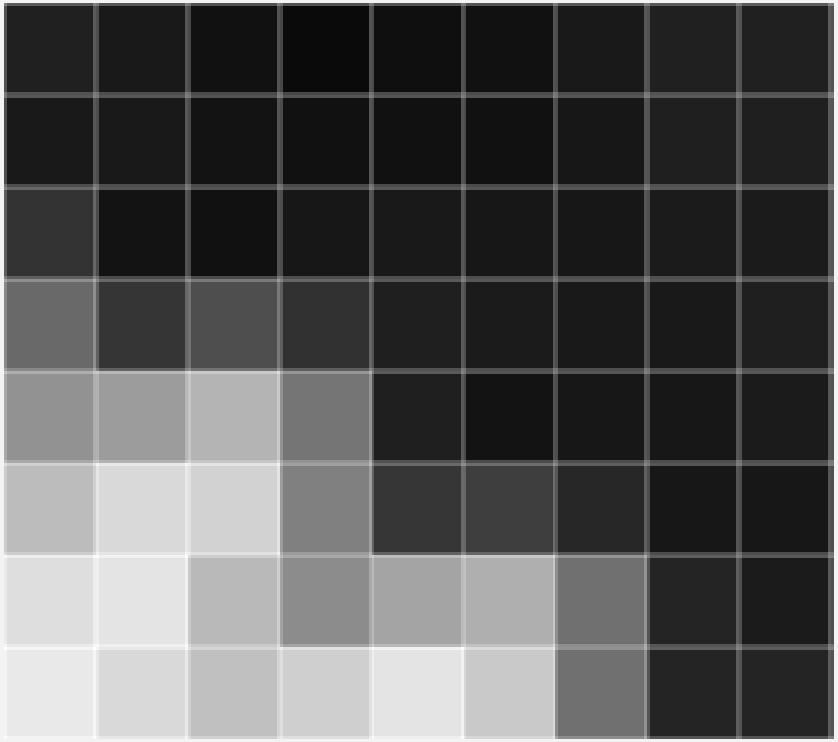
- Model transformation
  - local → global coordinates
- View transformation
  - global → camera
- Projection transformation
  - camera → screen
- Clipping, **rasterization, texturing** & Lighting
  - might take place earlier



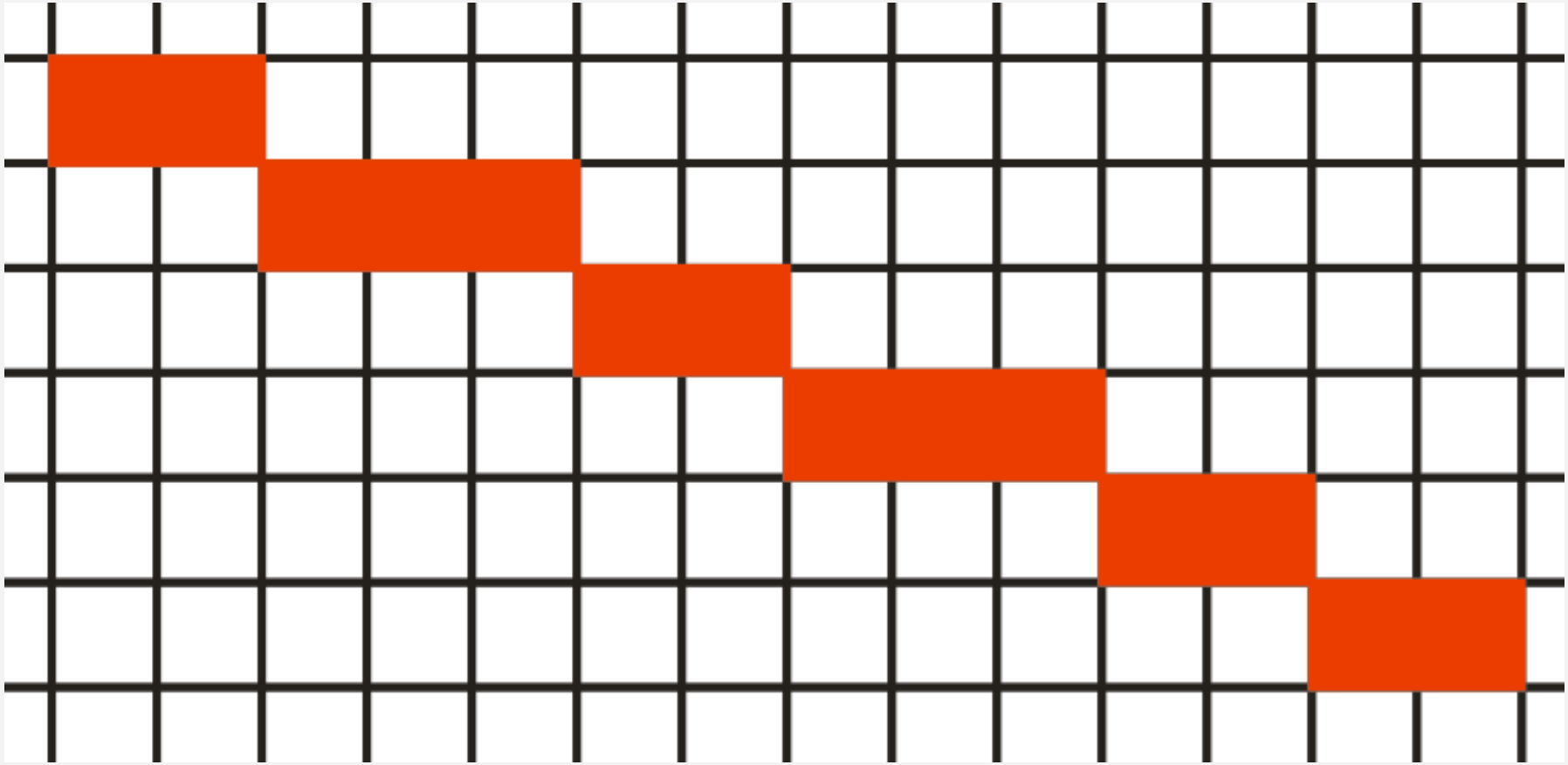
Rasterization

General problem

- Given a continuous geometric representation of an object
- Decide which pixels are occupied by the object

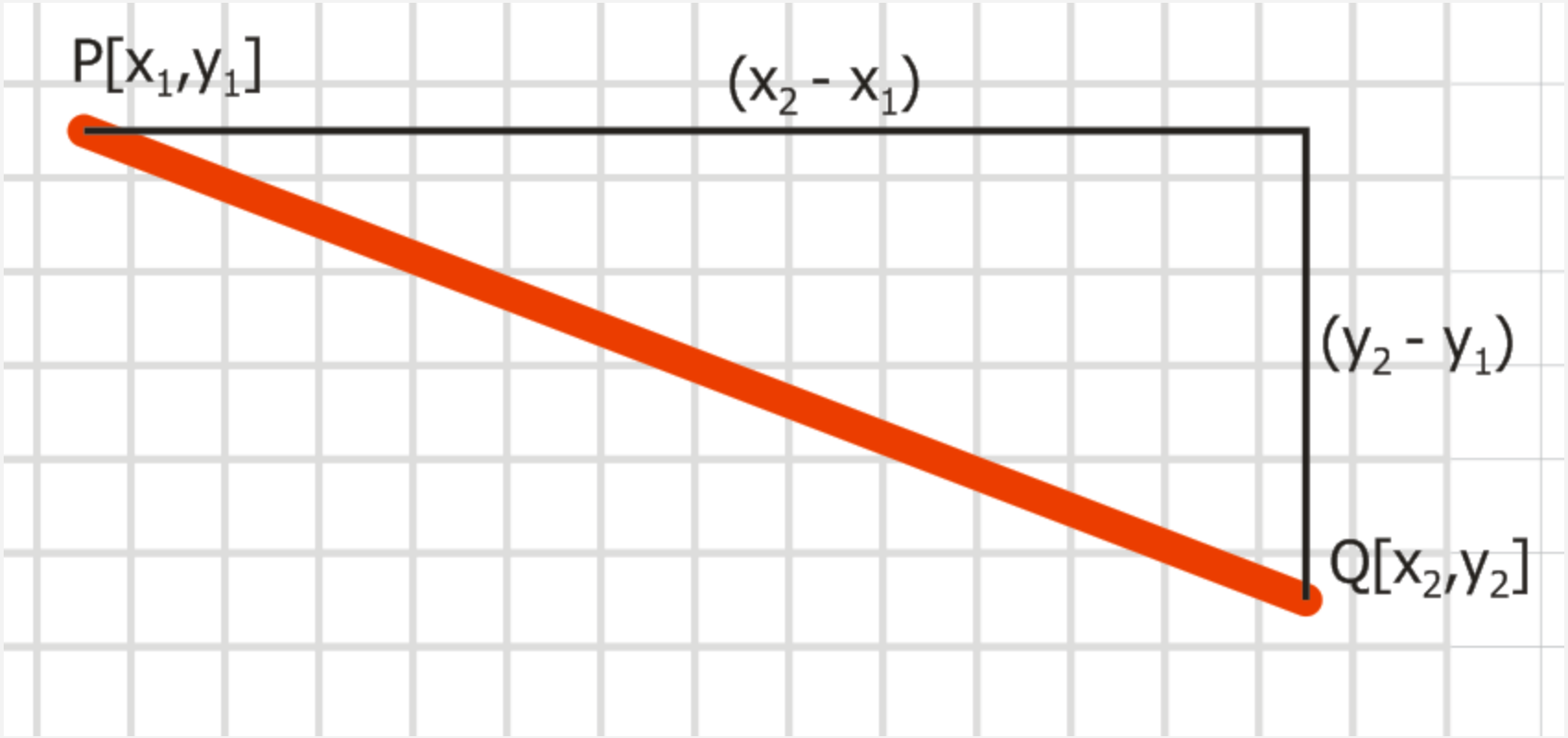


Line rasterization



Digital Differential Analyzer

- $dd = (y_2 - y_1) / (x_2 - x_1)$: float

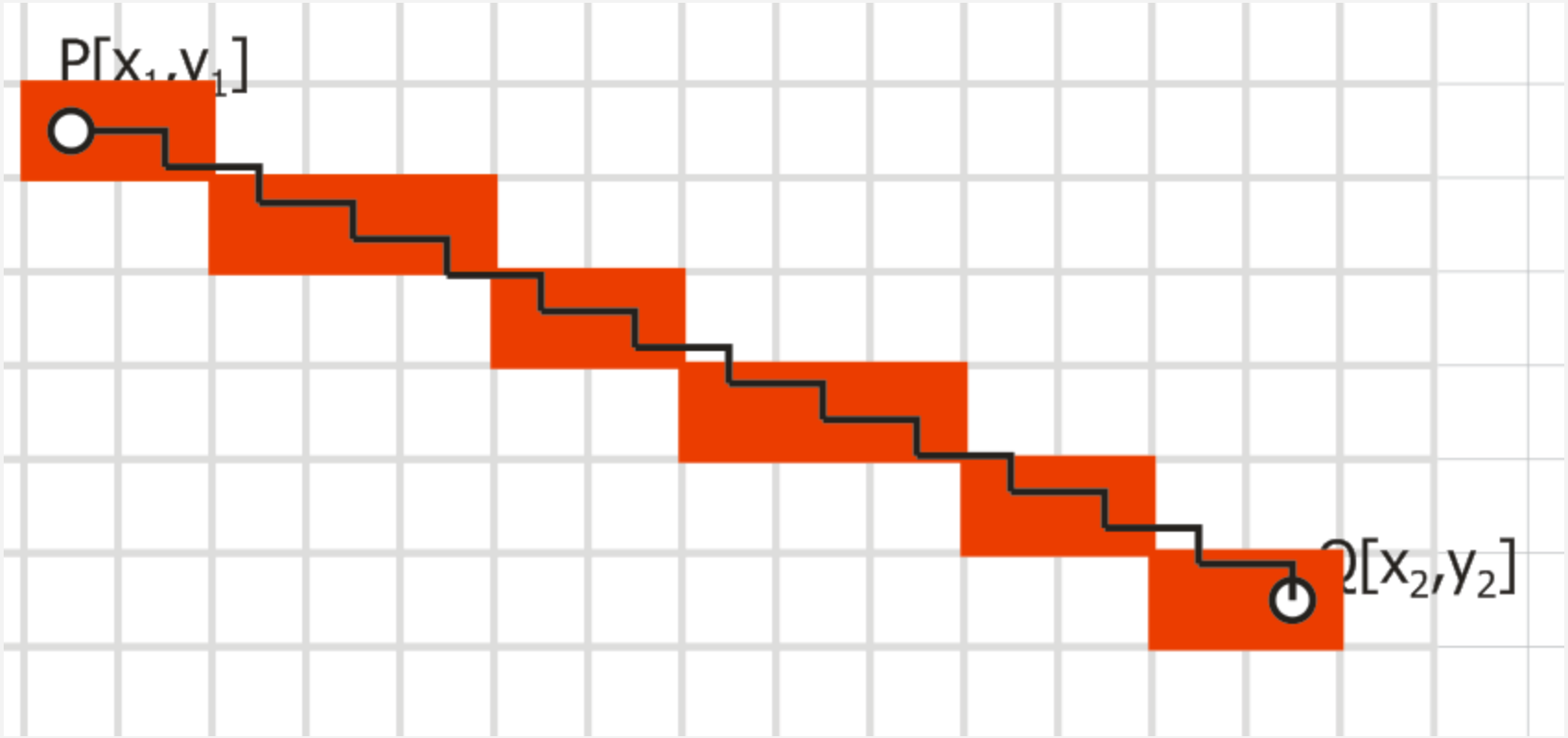


Digital Differential Analyzer

Pseudocode:

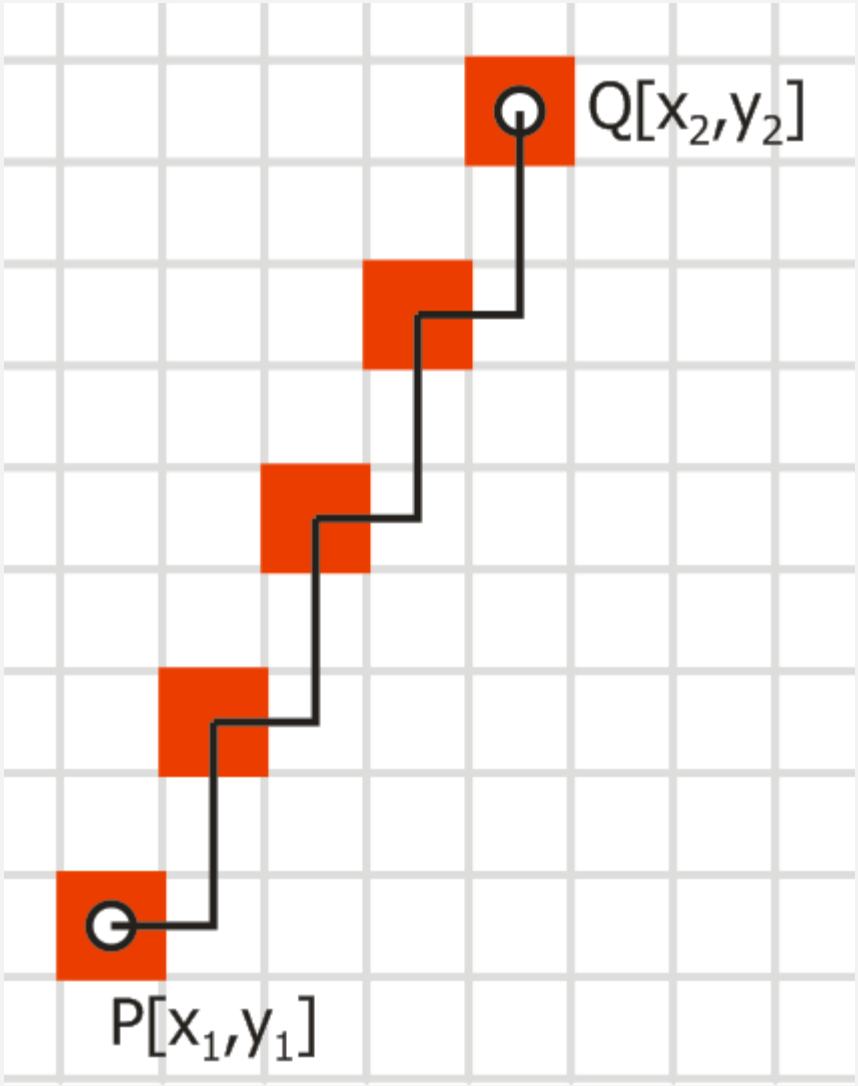
```
y = y1
for x = x1 to x2
   begin
      setpixel (x, round(y))
      y = y + dd
   end
```

Digital Differential Analyzer



Watch for line slope

- if abs(dd) > 1
- exchange x↔y in algorithm

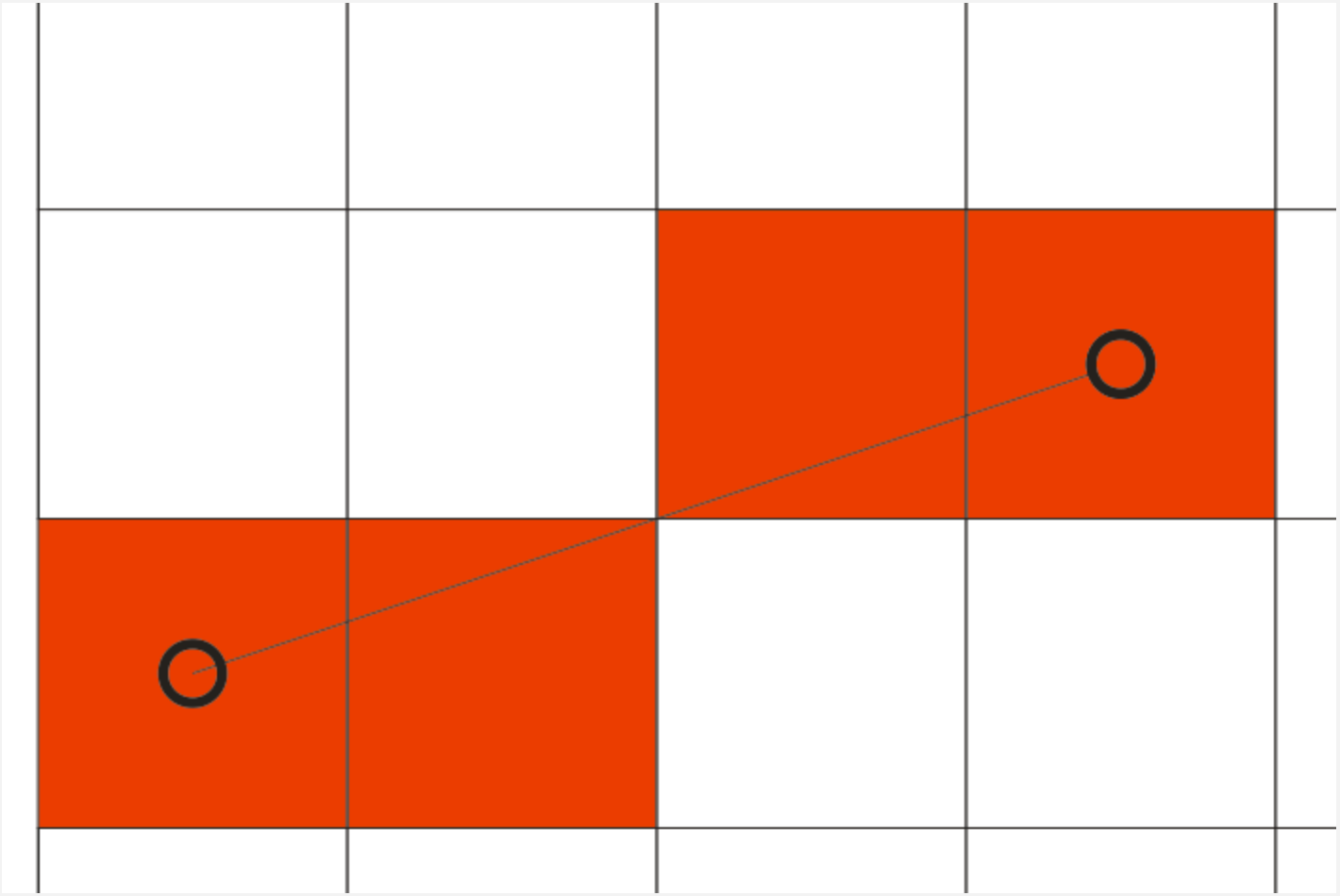


Bresenham algorithm

- DDA requires floating point
- Bresenham works with integers only
- main idea: for each x there are only 2 possible y values, pick the one with the smaller error. accumulate error over iterations.

- modify for other slopes and orientations

Bresenham algorithm



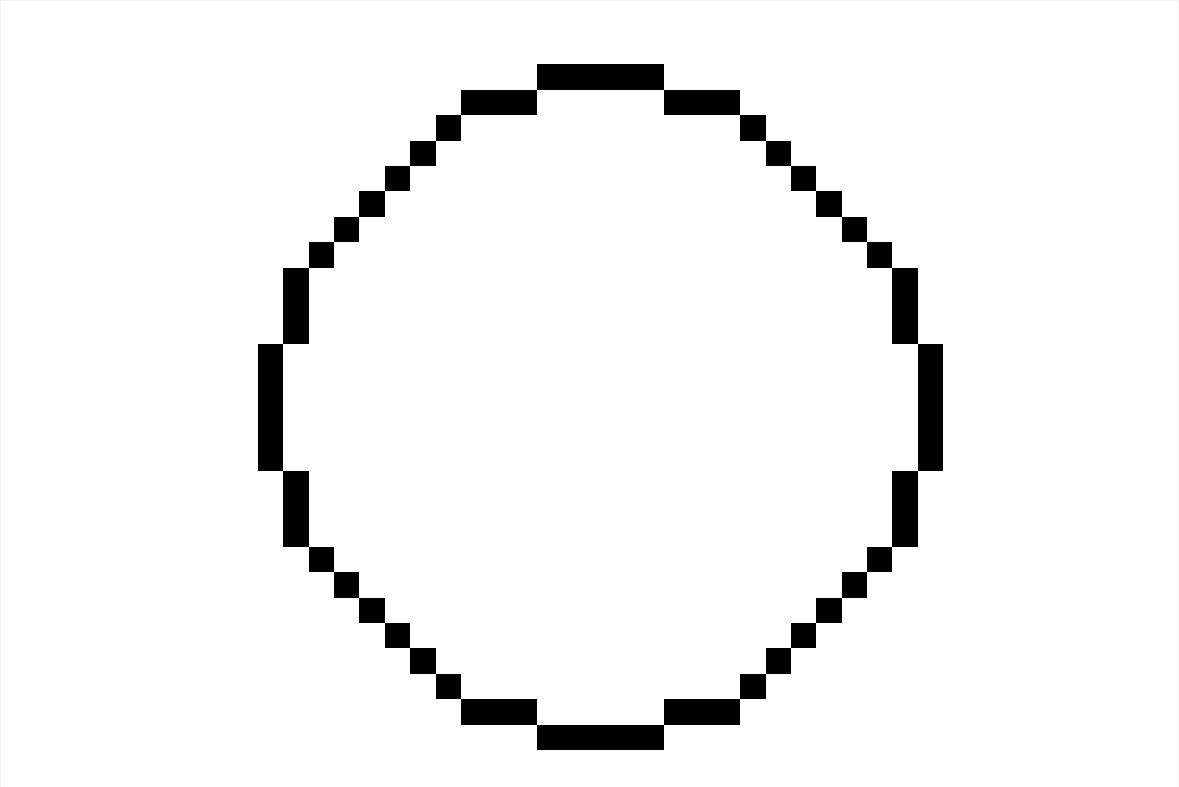
Bresenham algorithm

```
e = 0
y = y1
for x = x1 to x2
   begin
      setpixel (x,y)
      if (e + dd < 0.5)
            e = e + dd
      else
            e = e + dd – 1
            y = y + 1
   end
```

- no float rounding
- 0.5 float can be eradicated when multiplied by 2
- integers only

Circle, ellipse rasterization

- Bresenham for circles (midpoint algorithm)
- Can be modified for ellipses



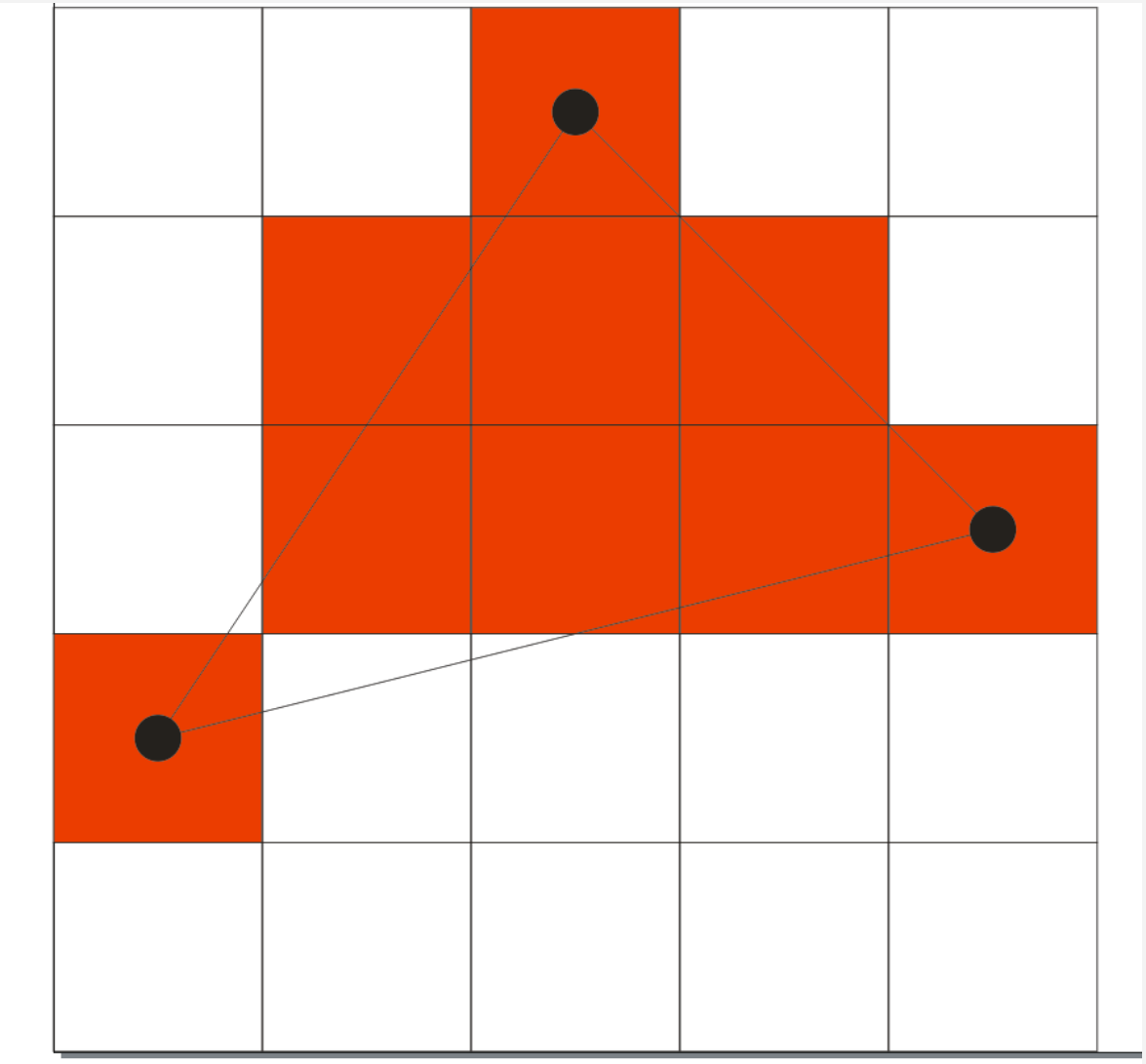
Polygon rasterization

Scanline algorithm:

For each scan line:

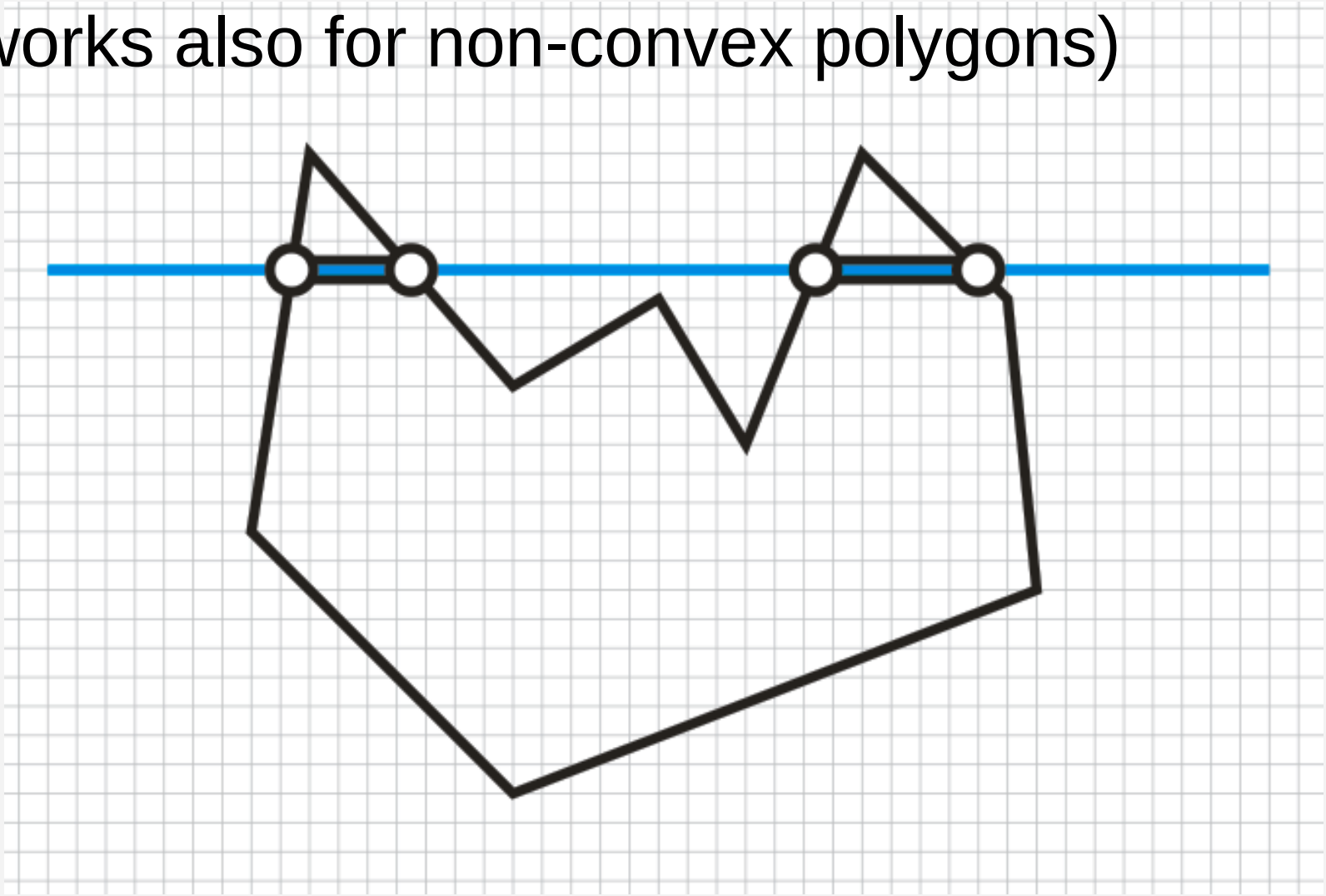
1. Find the intersections of polygon and the scan line
2. Sort the intersections by x coordinate
3. Fill the pixels between subsequent pairs of intersections

Scan-line algorithm



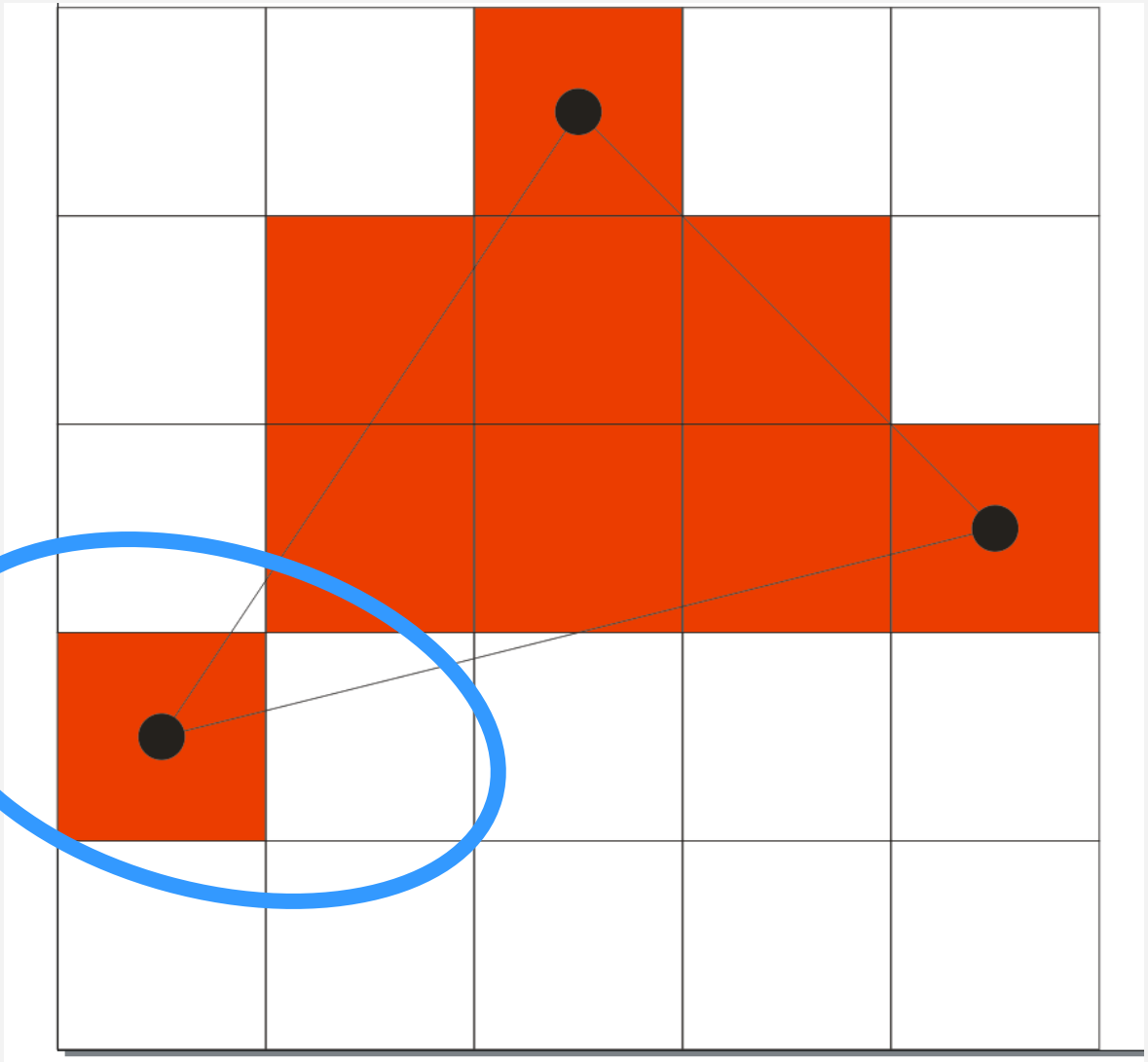
Scan-line algorithm

- (works also for non-convex polygons)



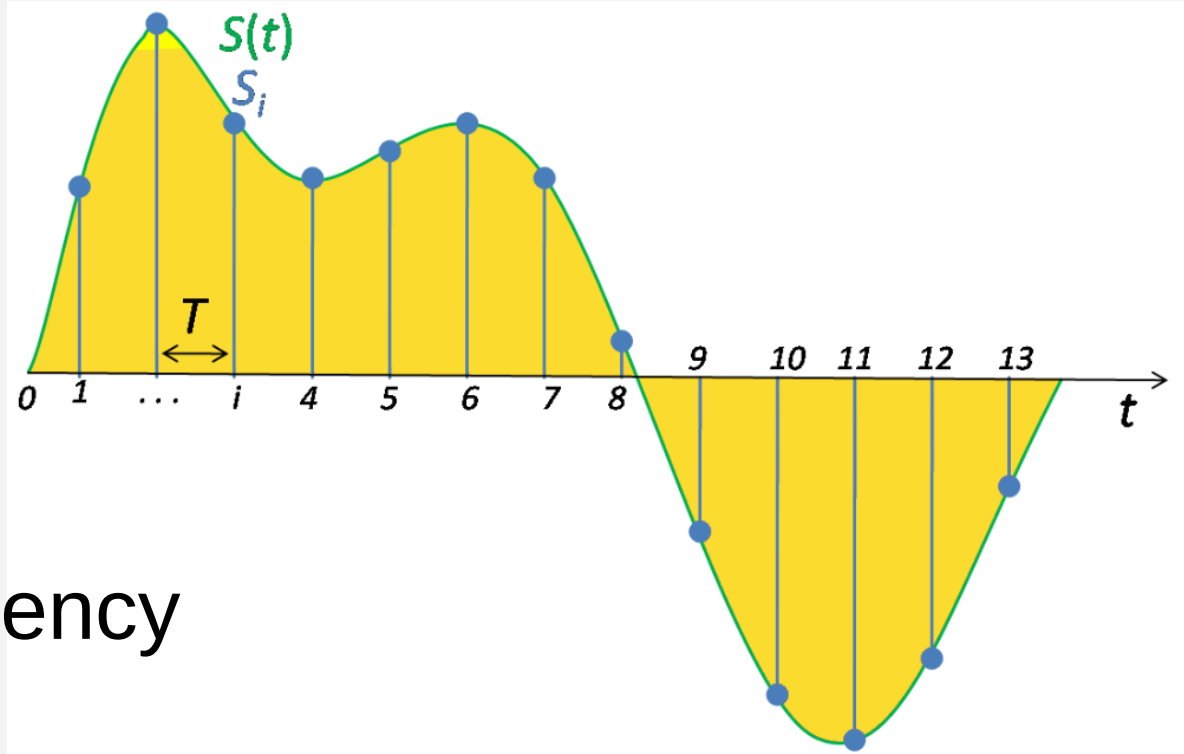
Rasterization alias

Alias!



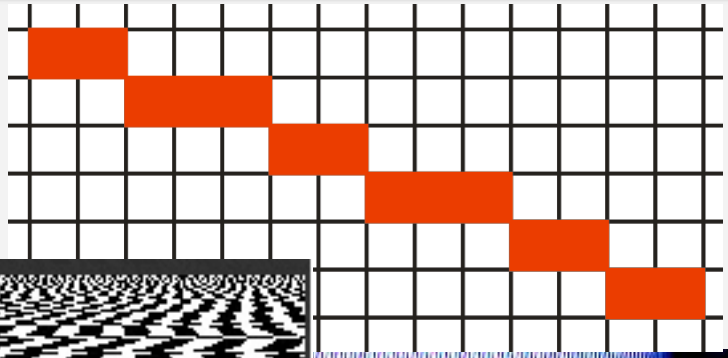
Aliasing

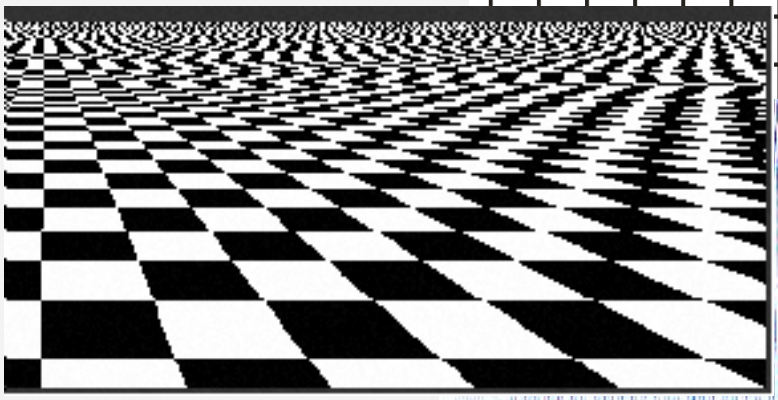
- continuous → discrete, artifacts might appear
- rasterization alias – jagged edges
- sampling
  - creating observation of continuous phenomenon in discrete intervals
- sampling frequency
  - pixel density

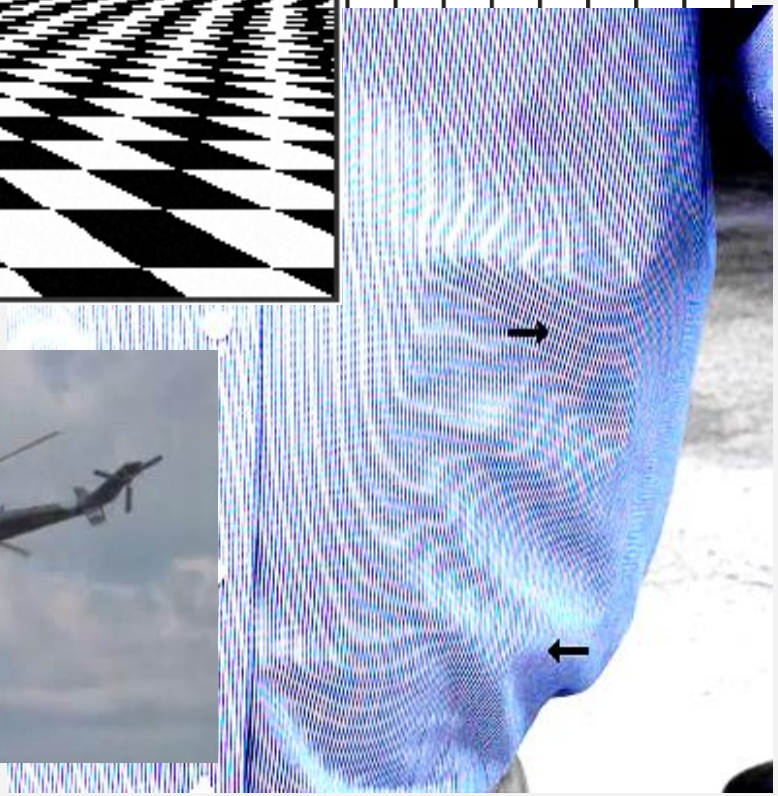


Forms of alias

- spatial alias
  - jaggy edges
  - moiré
  - texture distortion
- temporal
  - “wagon wheel”





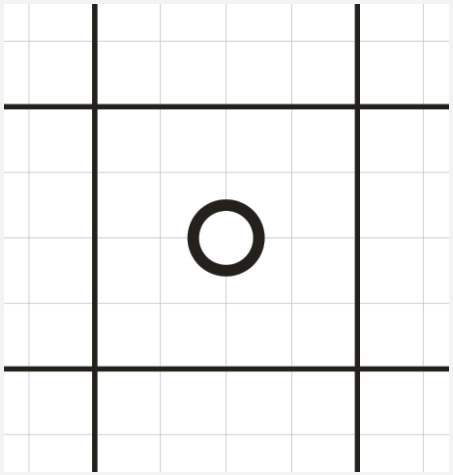


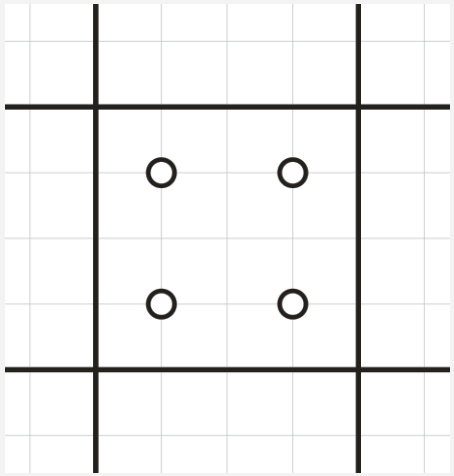
Anti-aliasing

- general (global) anti-aliasing - supersampling
  - works on all objects - positive
  - works on ALL objects - negative
- object (local) anti-aliasing
  - line anti-aliasing
  - silhouette anti-aliasing
  - texture anti-aliasing

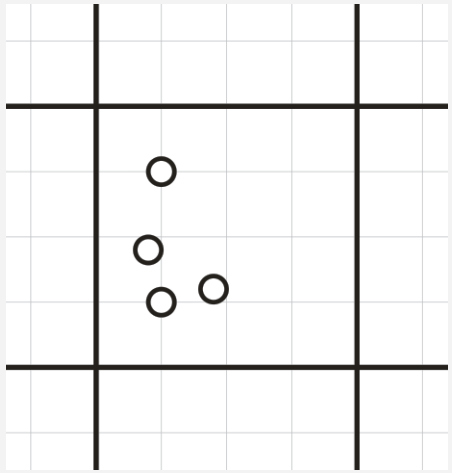
Super-sampling

- For each pixel perform multiple sub-pixel observations and combine the results

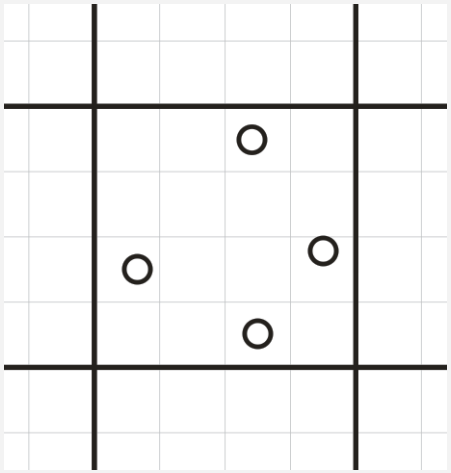




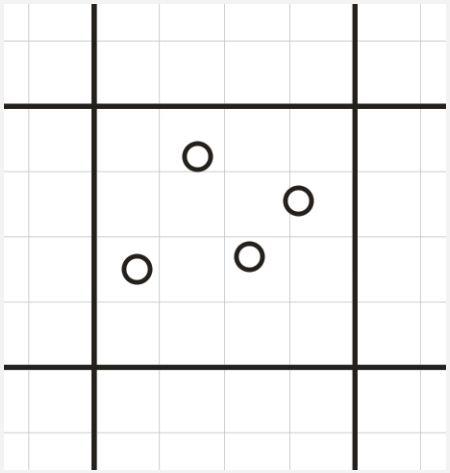
Regular (grid)



Random (stochastic)

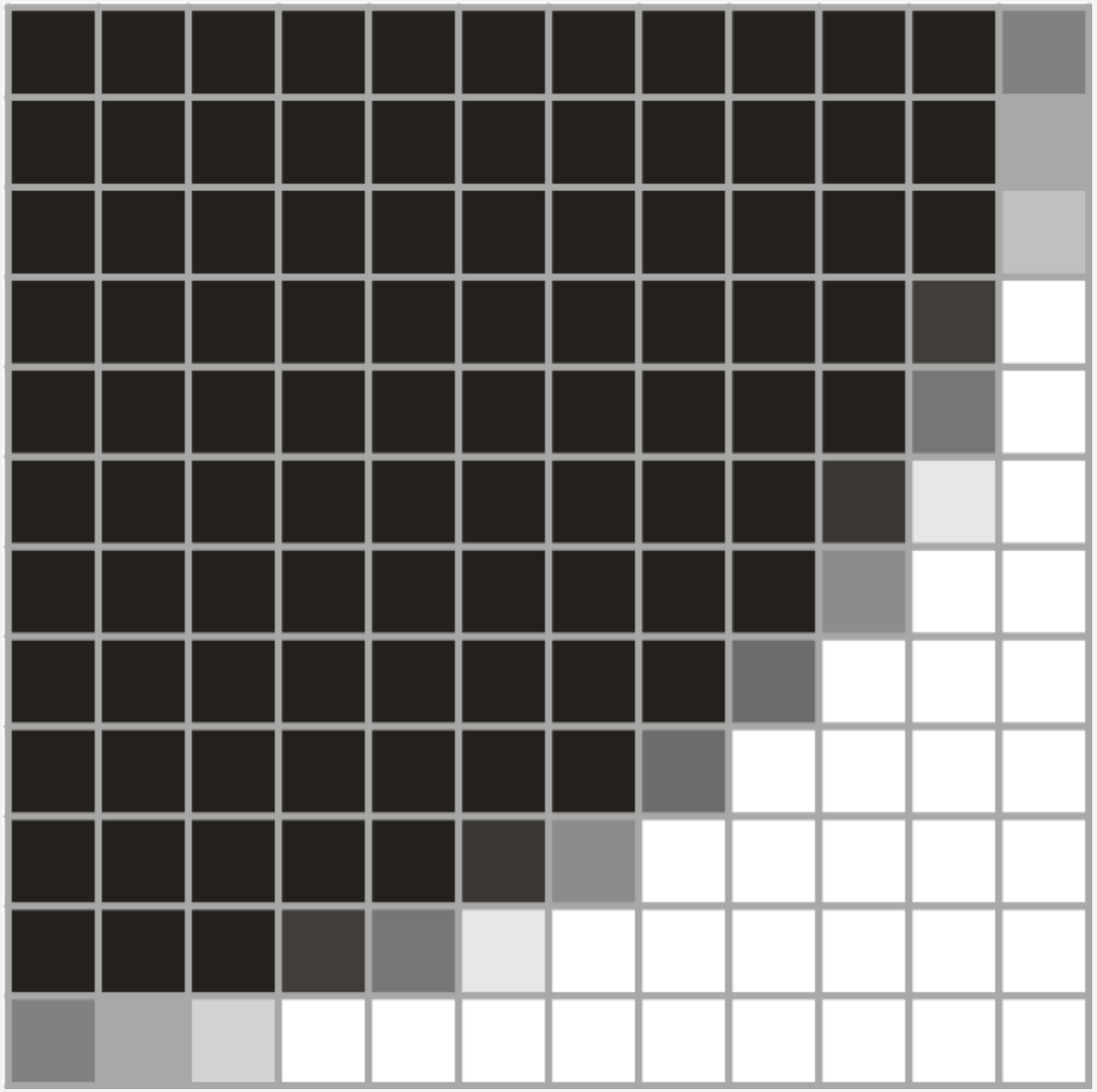


Poisson

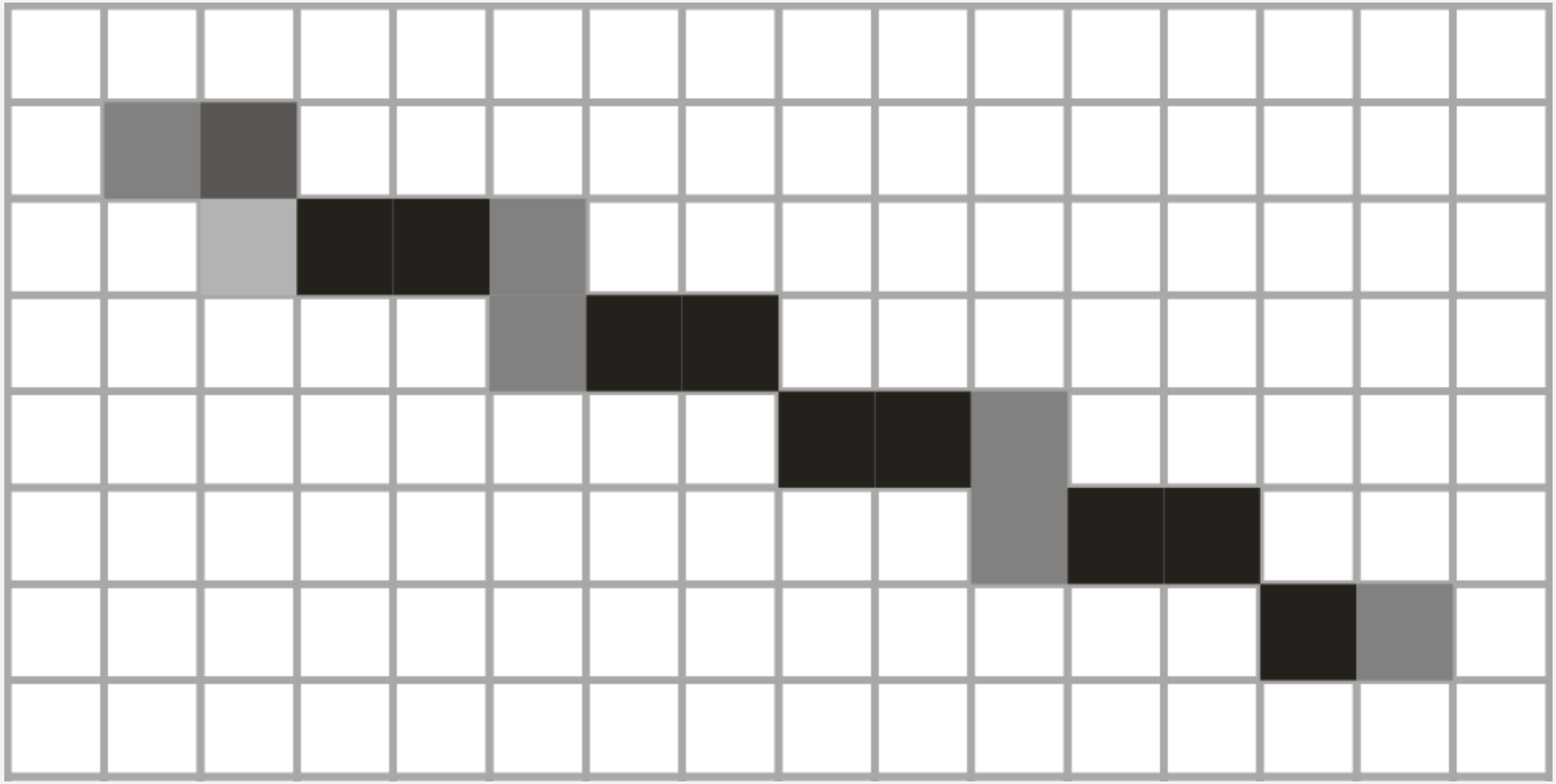


Jitter

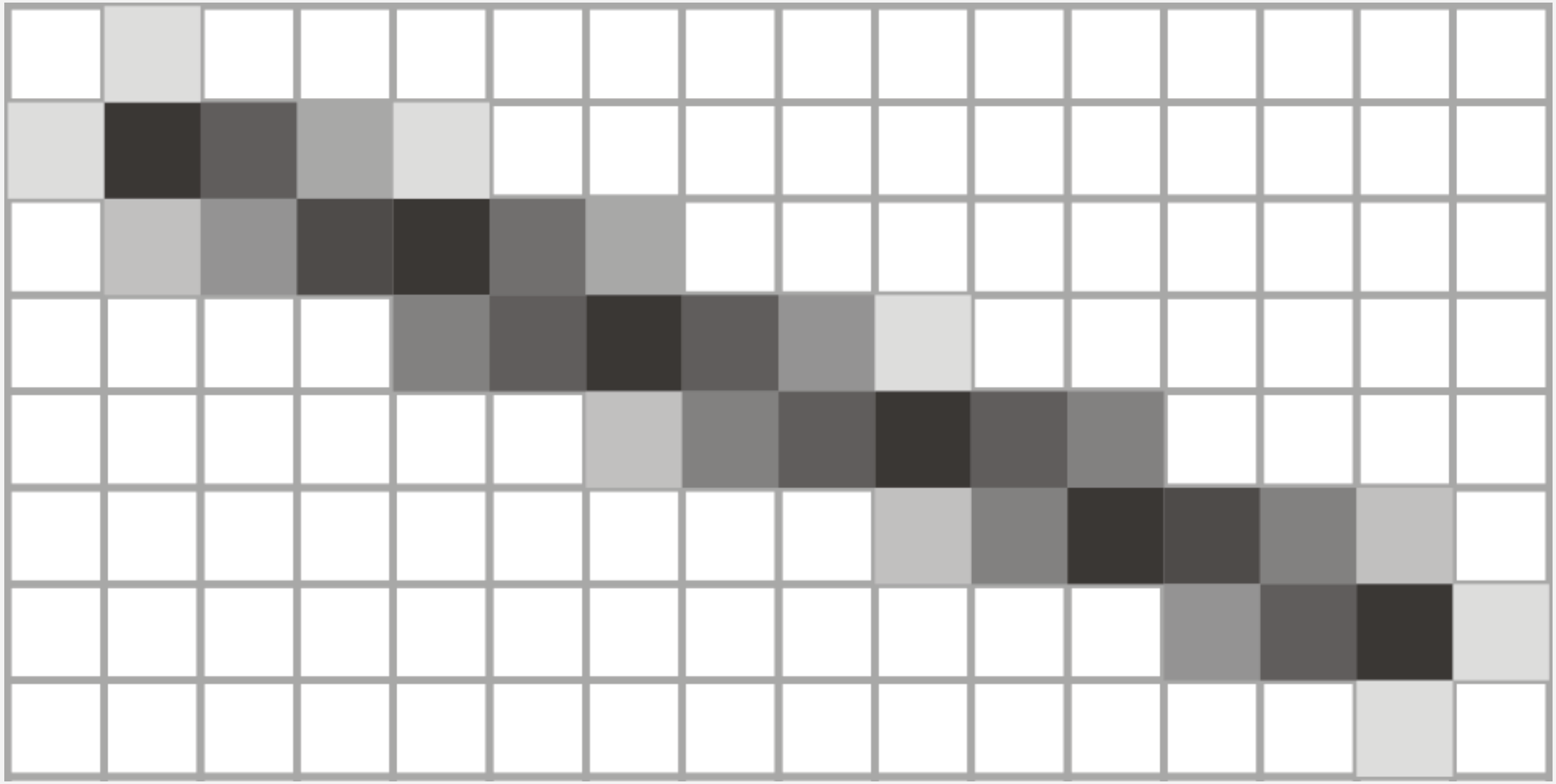
Super-sampling example



Line super-sampling



Object anti-aliasing: Line



Textures and mapping

Material

- **VISUALLY** distinguishes 2 objects with identical geometry
- For now, we focus on object's own color
- Other material properties will be discussed later

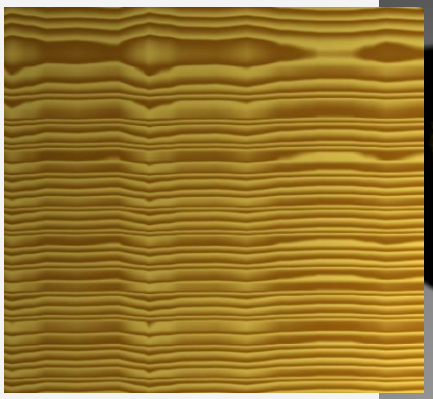
Object color

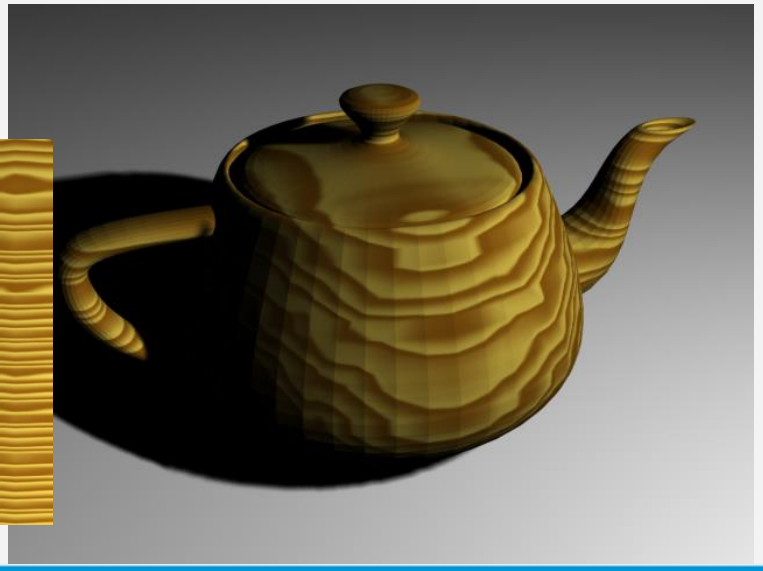
- Trivial case
  - constant color
- Usual case
  - color changes over object

Texture

- used to define changes in object's color

- 2D bitmap
- 3D bitmap
- Texel
- procedural texture





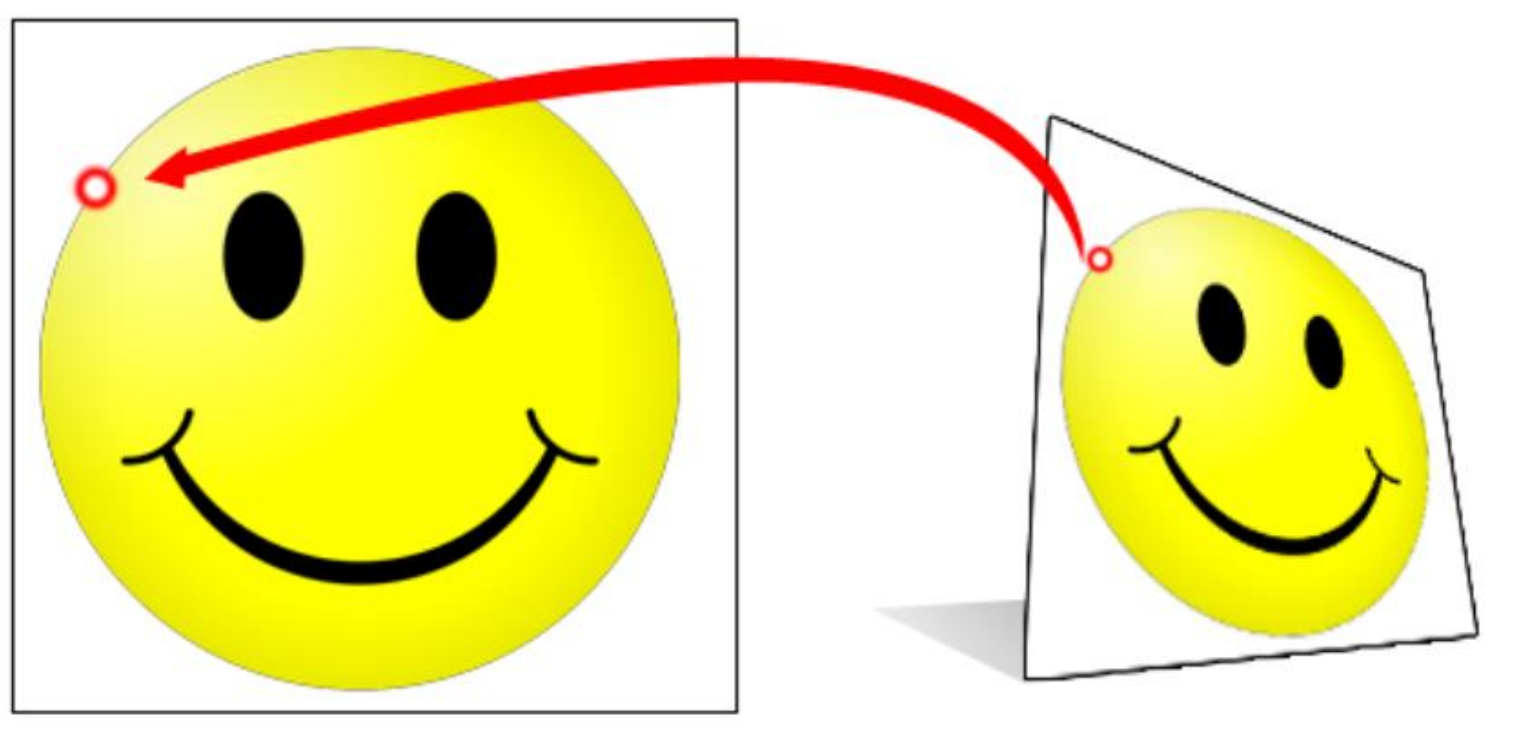
Texture mapping

- object space ↔ 2D texture space

- New coordinate system: Texture coordinates

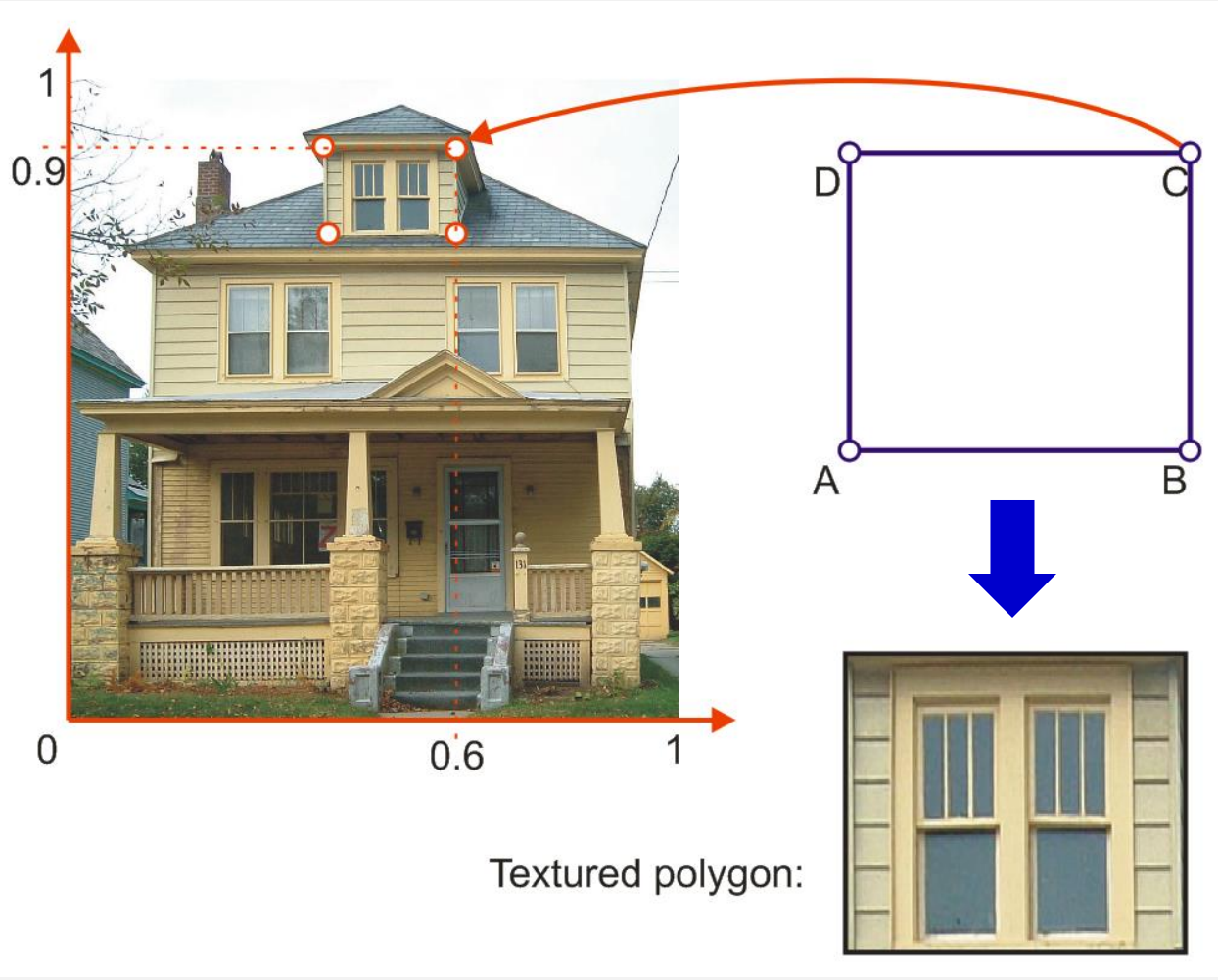
Texture mapping for polygons

- General problem:
  Which texture pixel appears under a screen pixel occupied by a textured polygon?



Texture coordinates

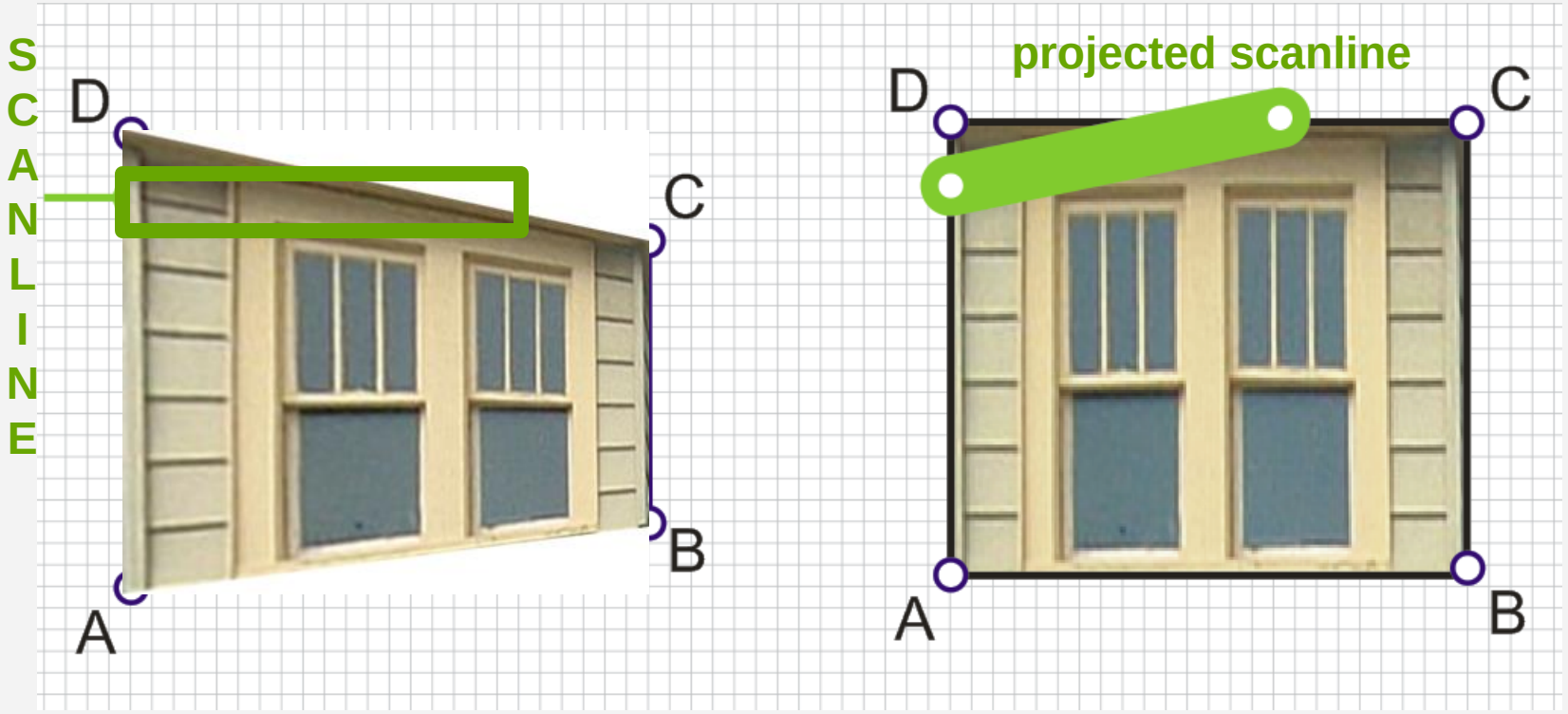
- Position of polygon's control points in texture space



- Texture coordinates for C are [0.6, 0.9]

Intermediate pixels

- Remember polygon rasterization

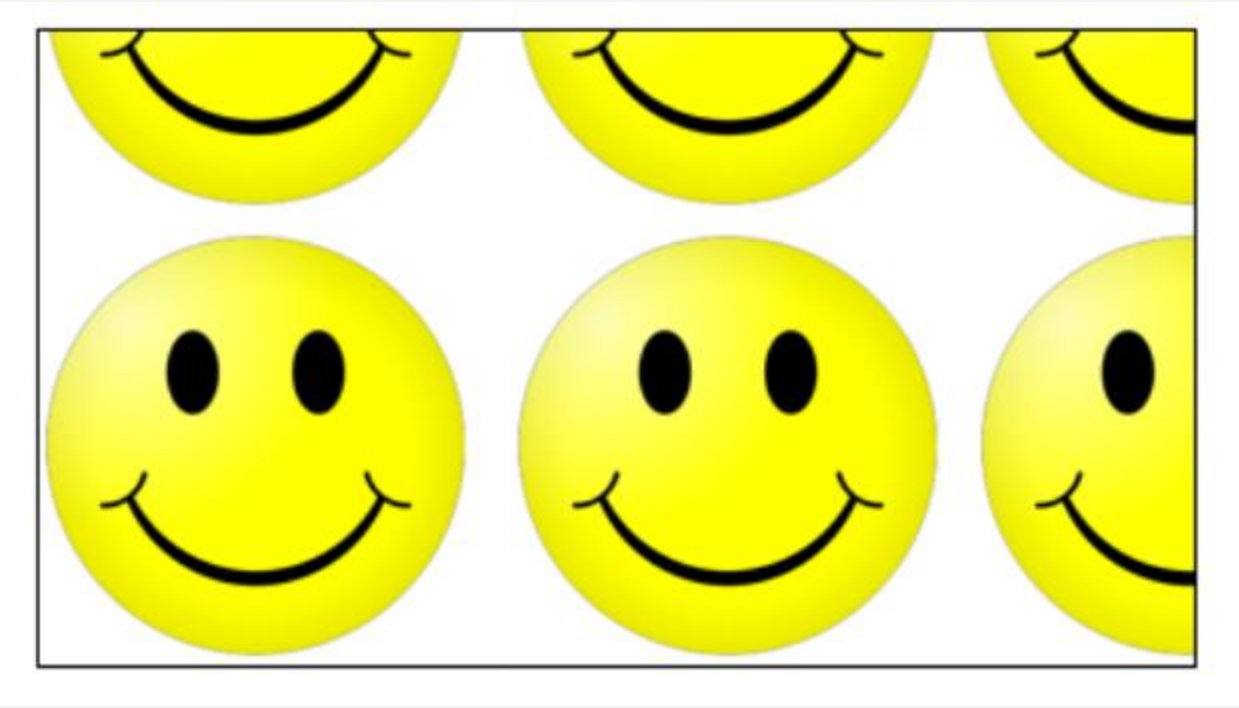


Screen space

Texture space

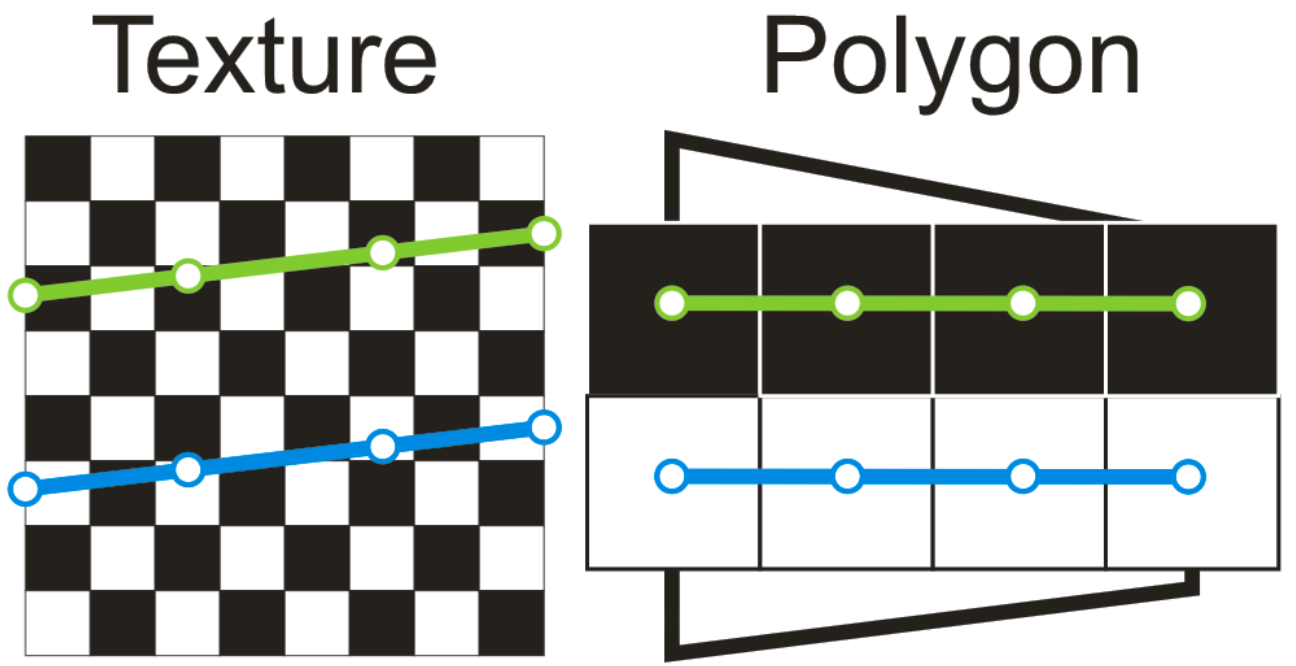
Texture tiling

- What happens if texture coordinates > 1 ?
- e.g. [0, 0] [2.5 , 0] [2.5, 1.5], [0, 1.5]

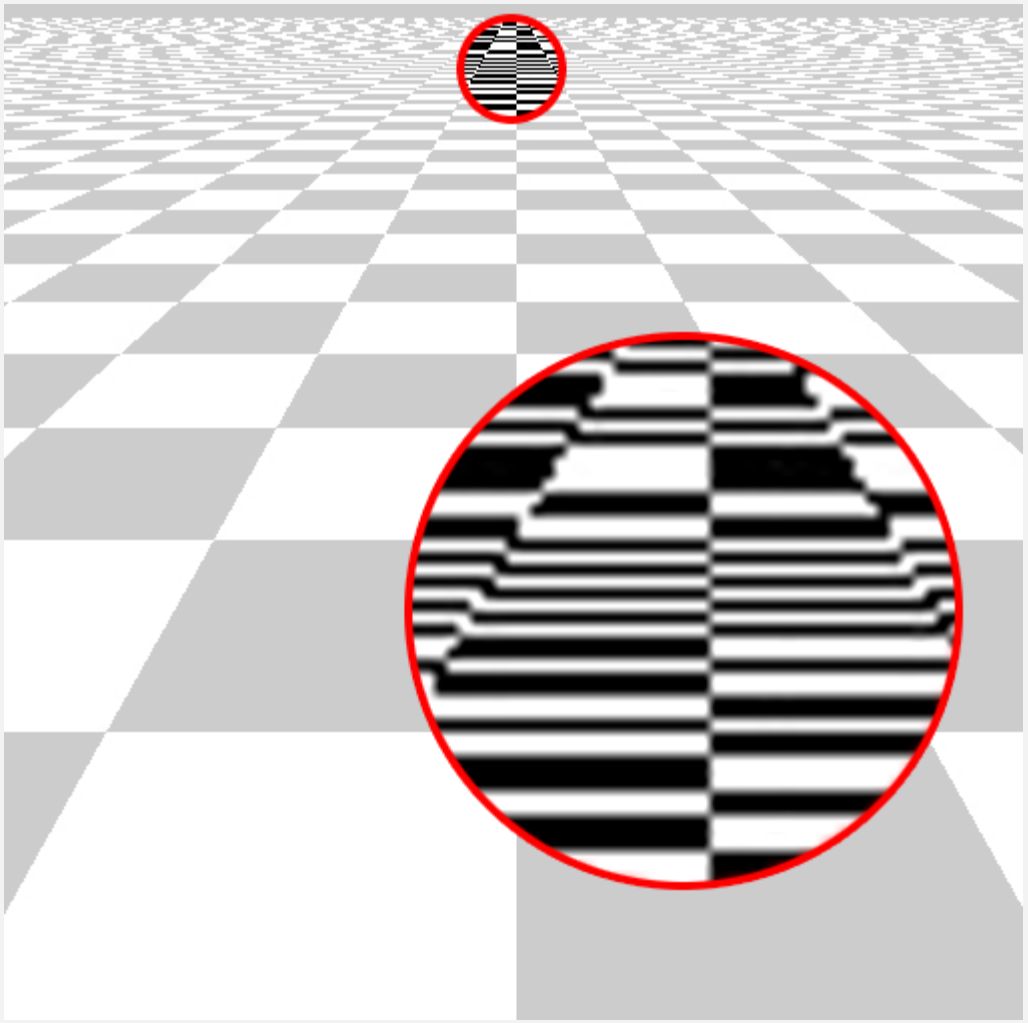


Texturing issues

- Texture size <> polygon size
  - 1-pixel step in screen <> 1-pixel step in texture
- Reading pixels from texture is a case of sampling
- Artifacts
- Alias



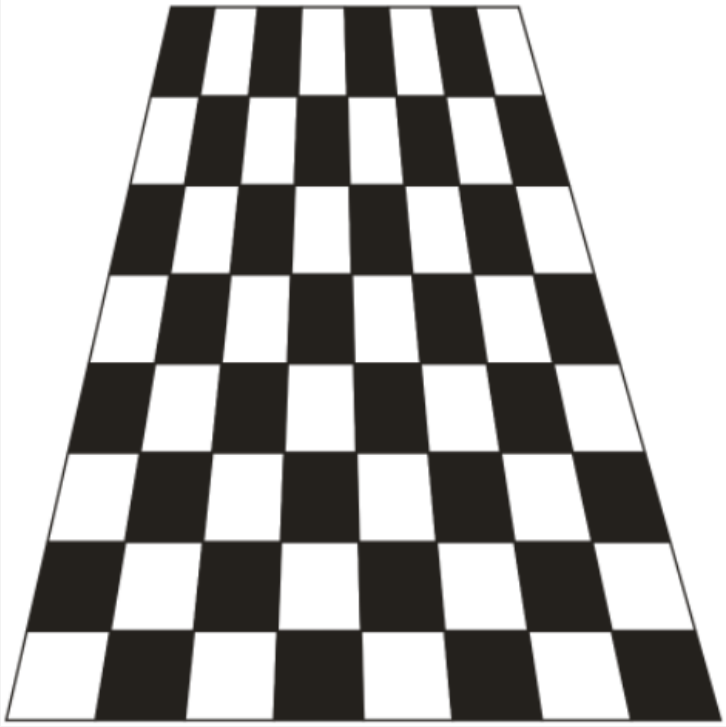
Example



3D – perspective correct!

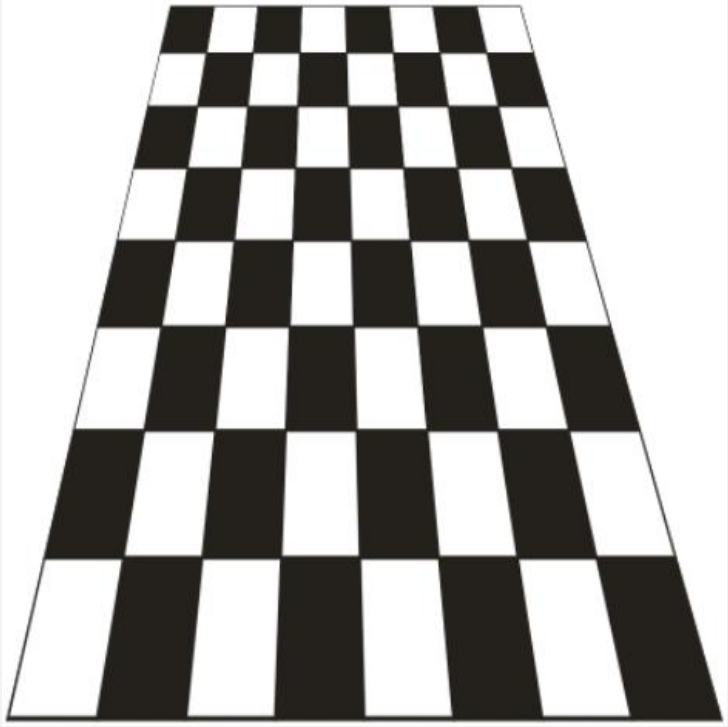
INCORRECT

$$T = (1-t)P + tQ$$



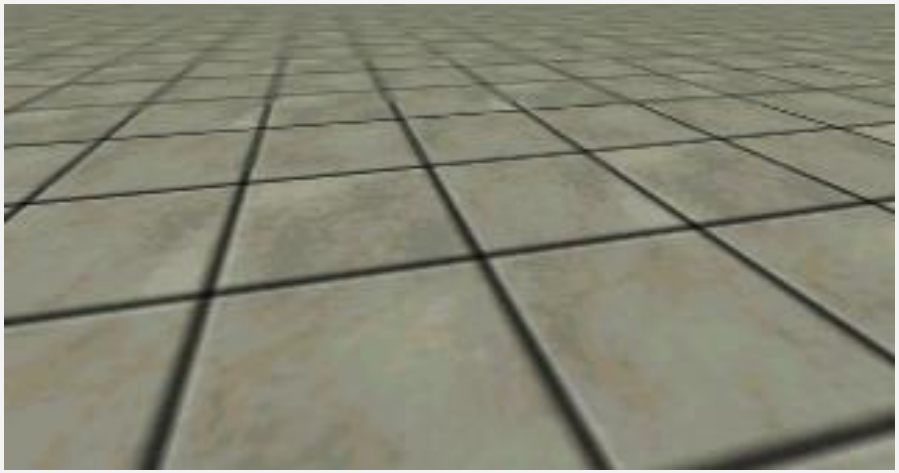
CORRECT

$$T = (1-t)\frac{P}{P_z} + t\frac{Q}{Q_z}$$



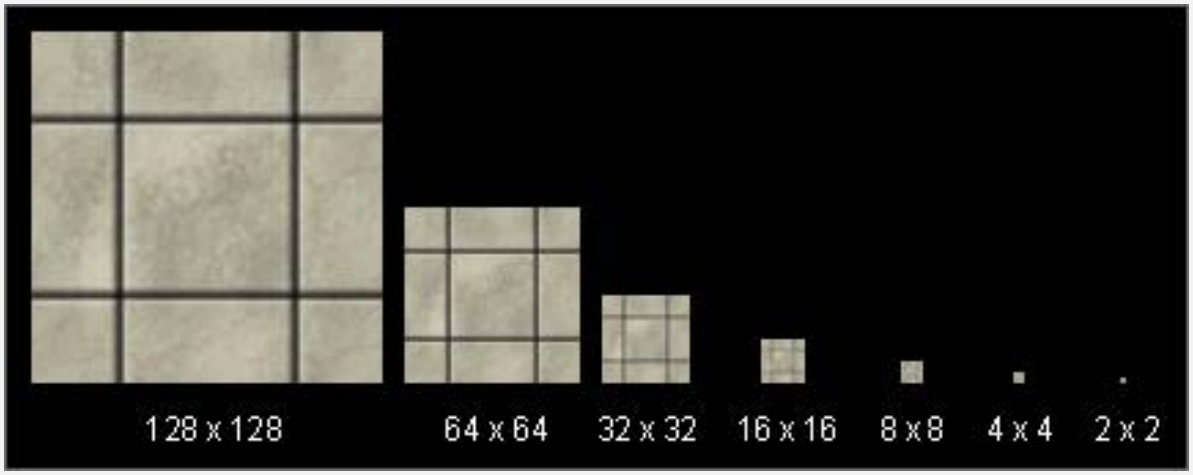
Improvements – MIP-maps

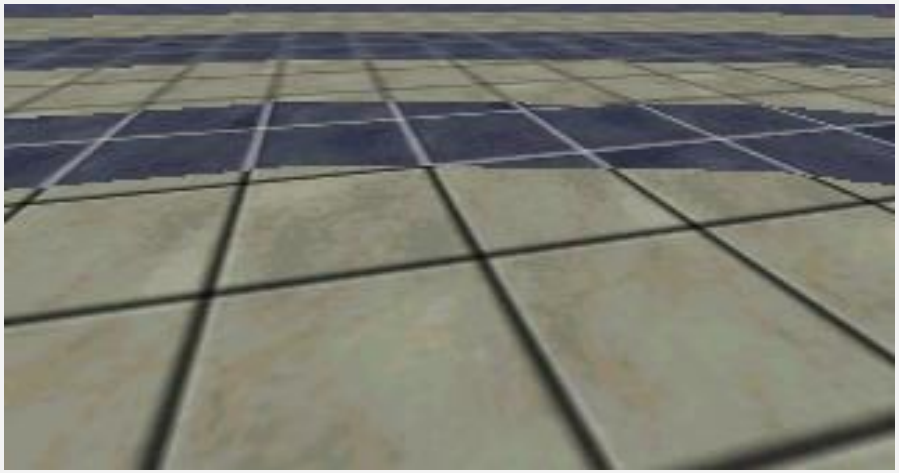
- Distant polygons use a reduced texture
- MIP-maps



- Drawbacks – “waves”
- http://www.gamedev.net/reference/articles/article1233.asp

Improvements – MIP-maps

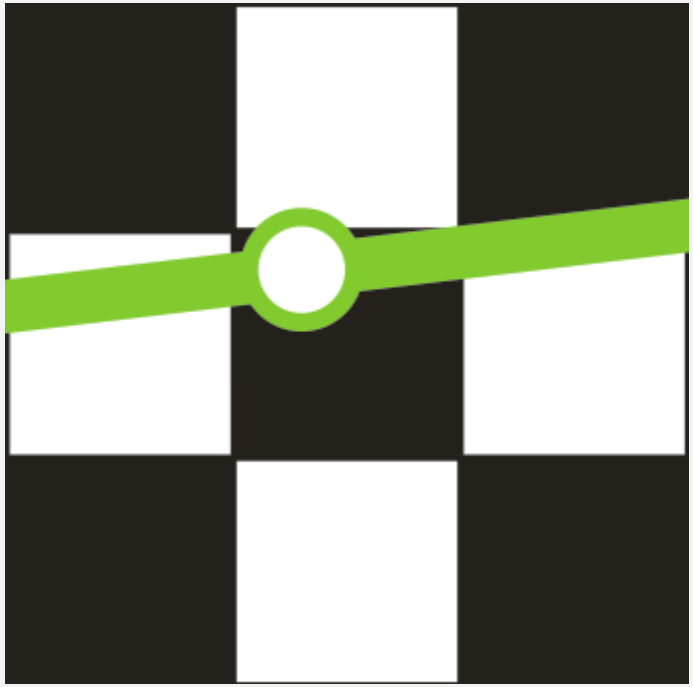


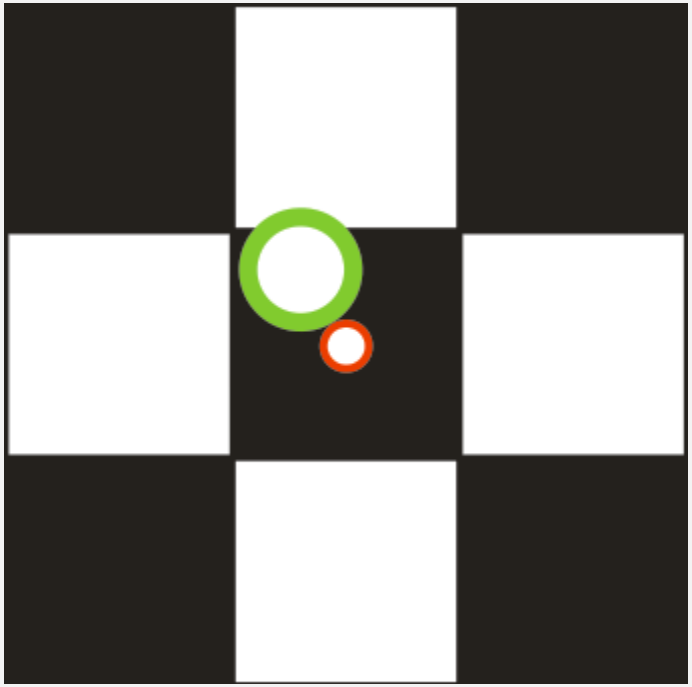


- http://www.gamedev.net/reference/articles/article1233.asp

Improvements – Filtering

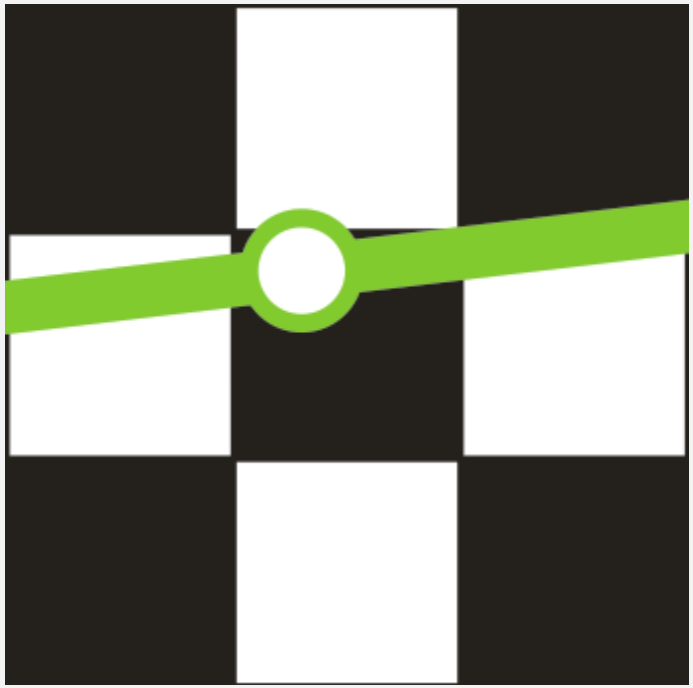
- Nearest neighbor – not filtering

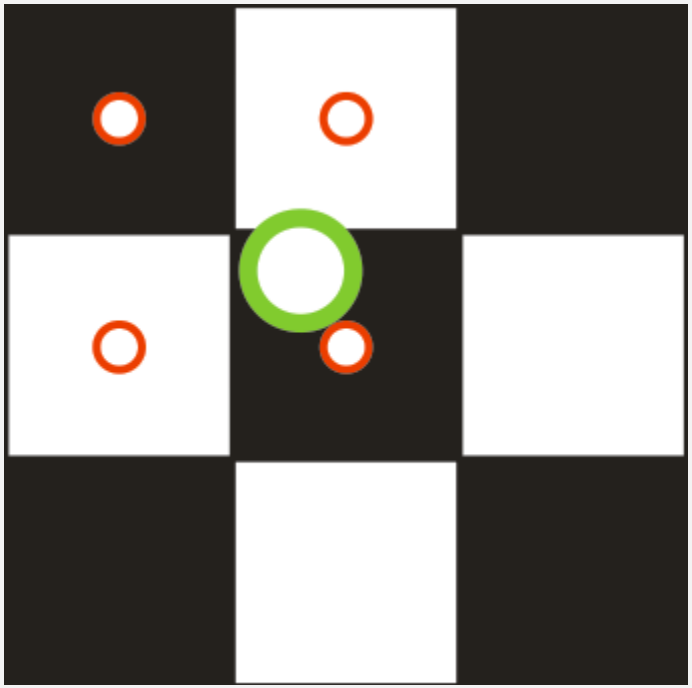


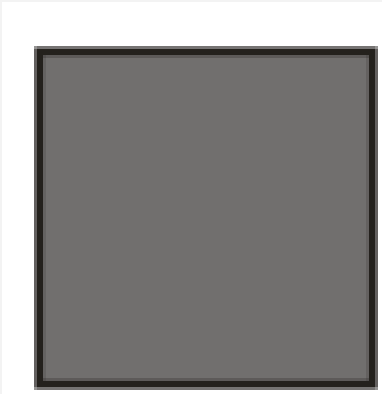


Improvements – Filtering

- Bilinear filtering

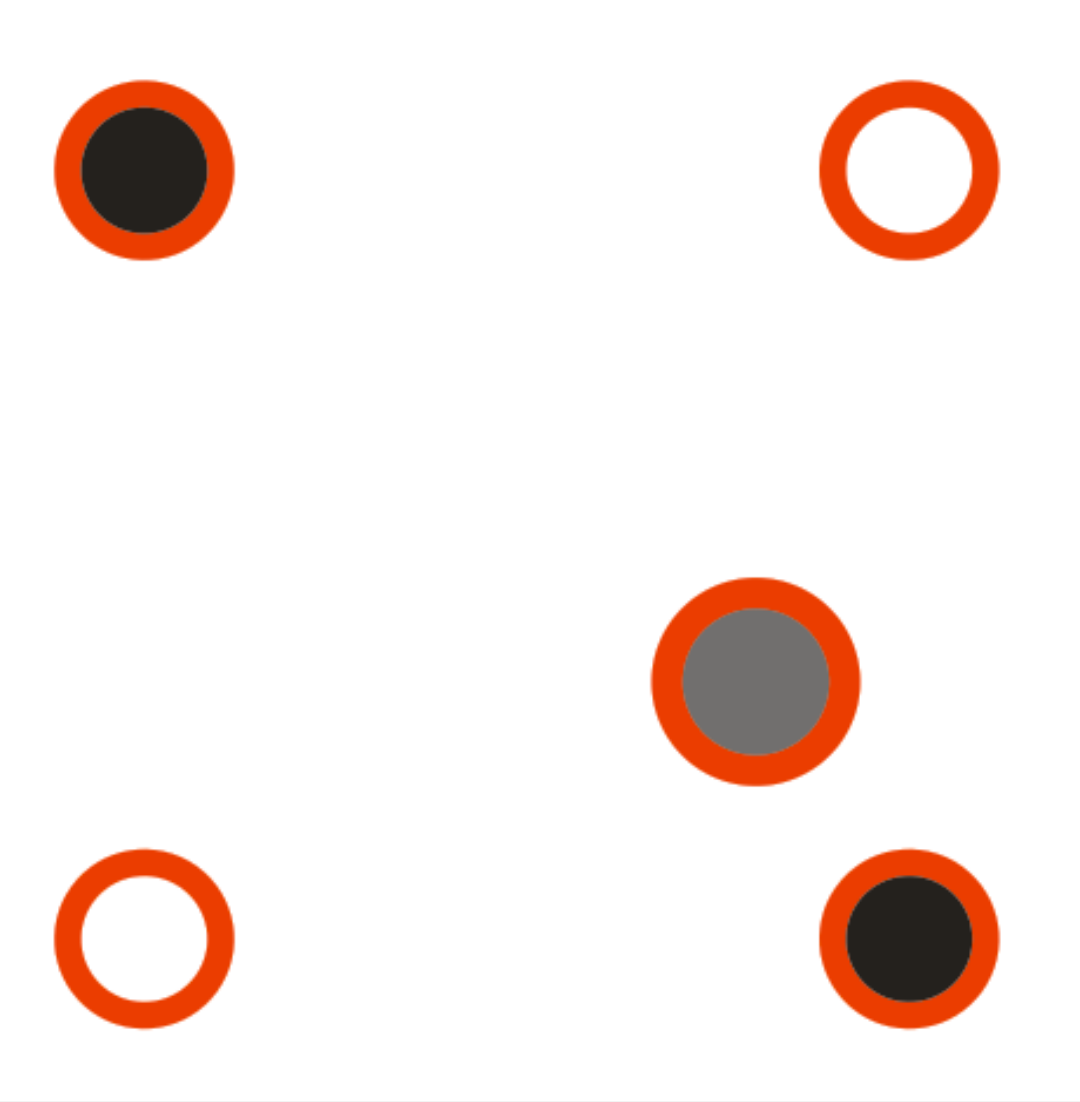






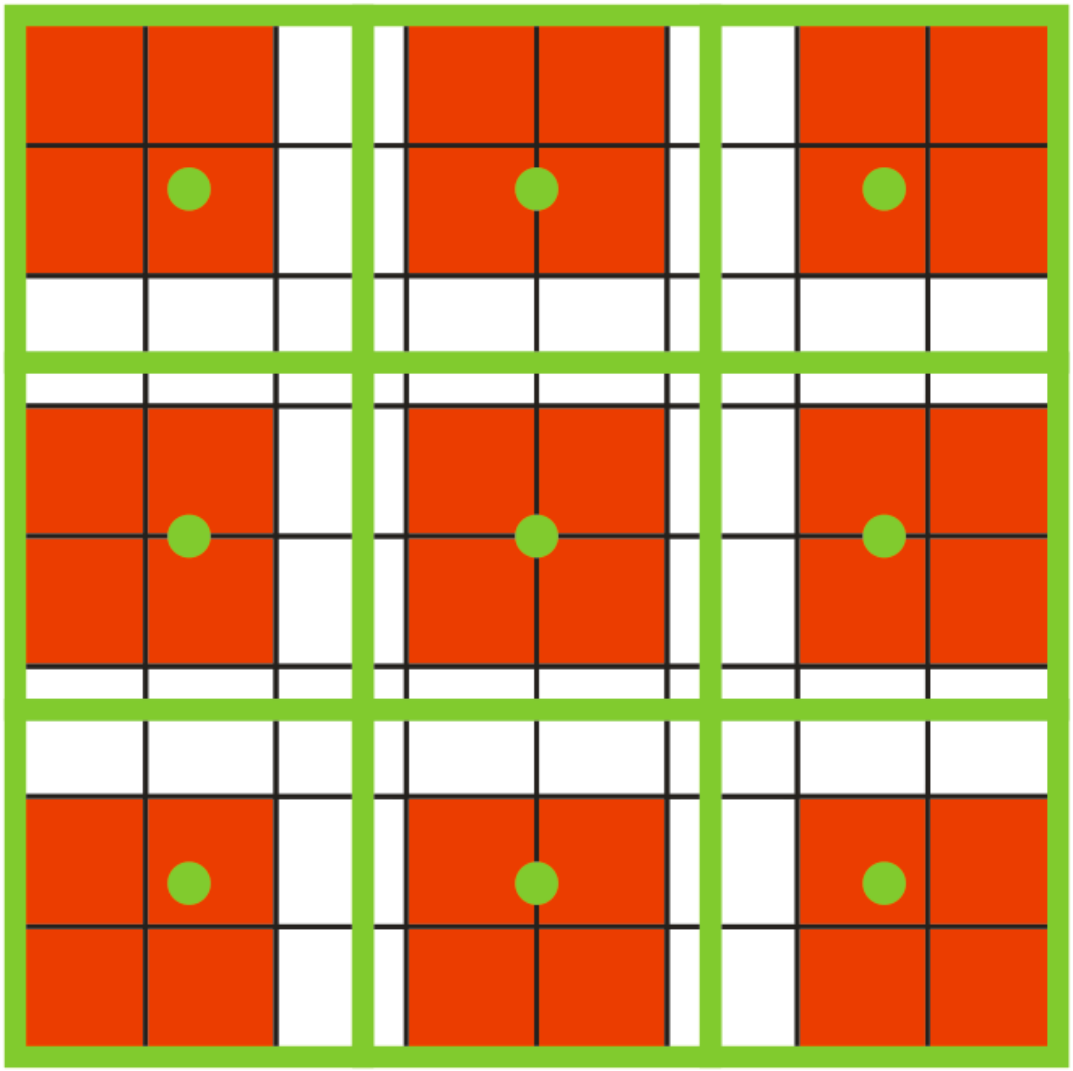
- Drawbacks – artifacts still occur

Bilinear interpolation



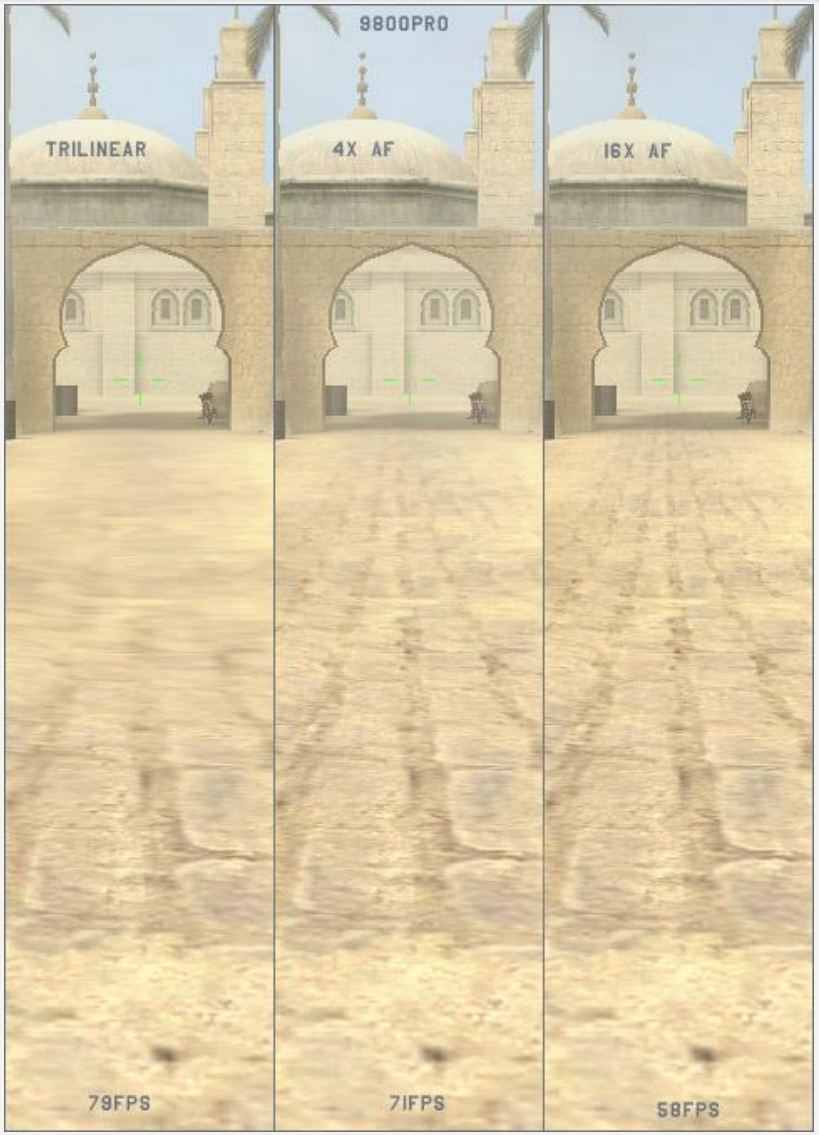
Bilinear filtering issues

- Ignored texels when minimizing below 50%



Further improvements

- Trilinear filtering
  - bilinear interpolation in mipmaps + lin.interpol.
- Anisotropic filtering
  - different texture samples in different directions
- Procedural textures
  - Parametric
  - suffer less from aliasing



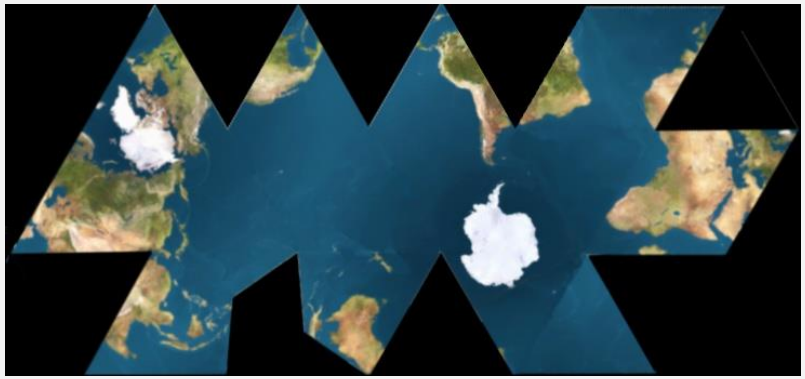
Texturing objects

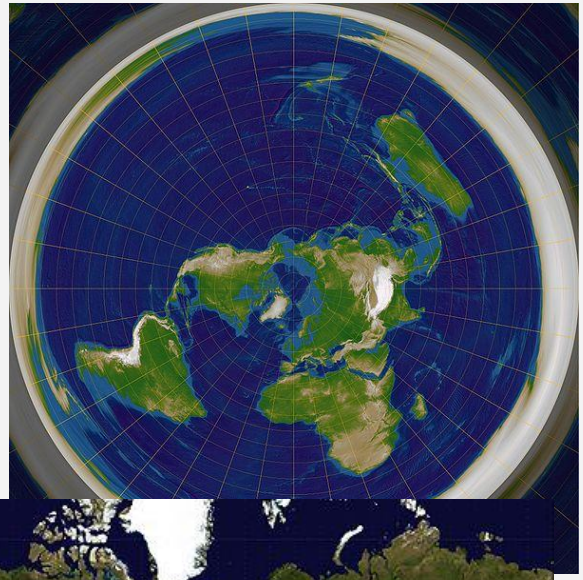
Texturing objects

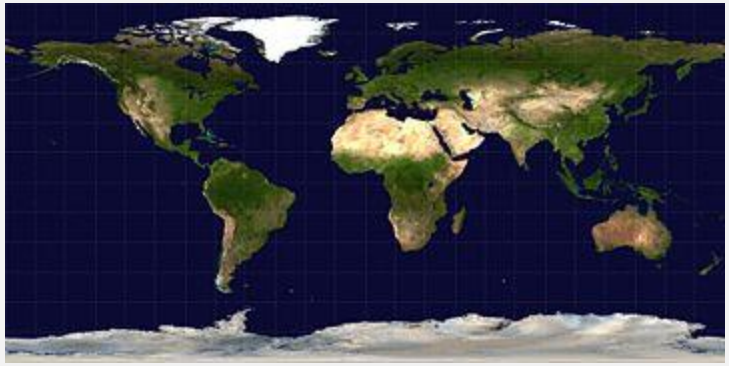
- Polygon objects
- We know how to texture polygons using the texture coordinates of their vertices
- But how do we set the texture coordinates?
  - except for manually of course
- Unwrap object into plane

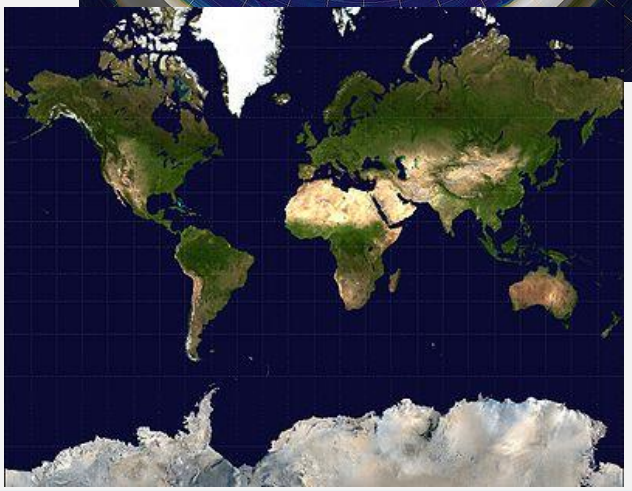
Example – cartography

- Unwrapping earth into a plane







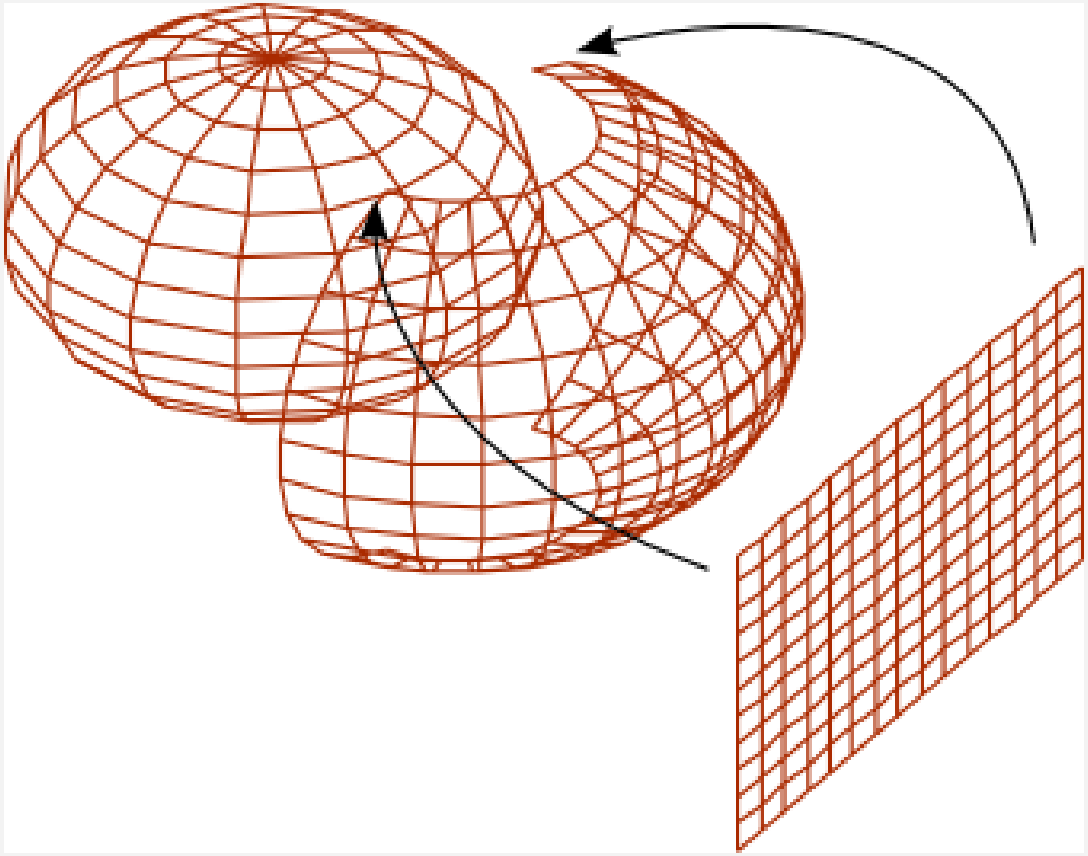


UV mapping

- XYZ ↔ UV
- sphere:

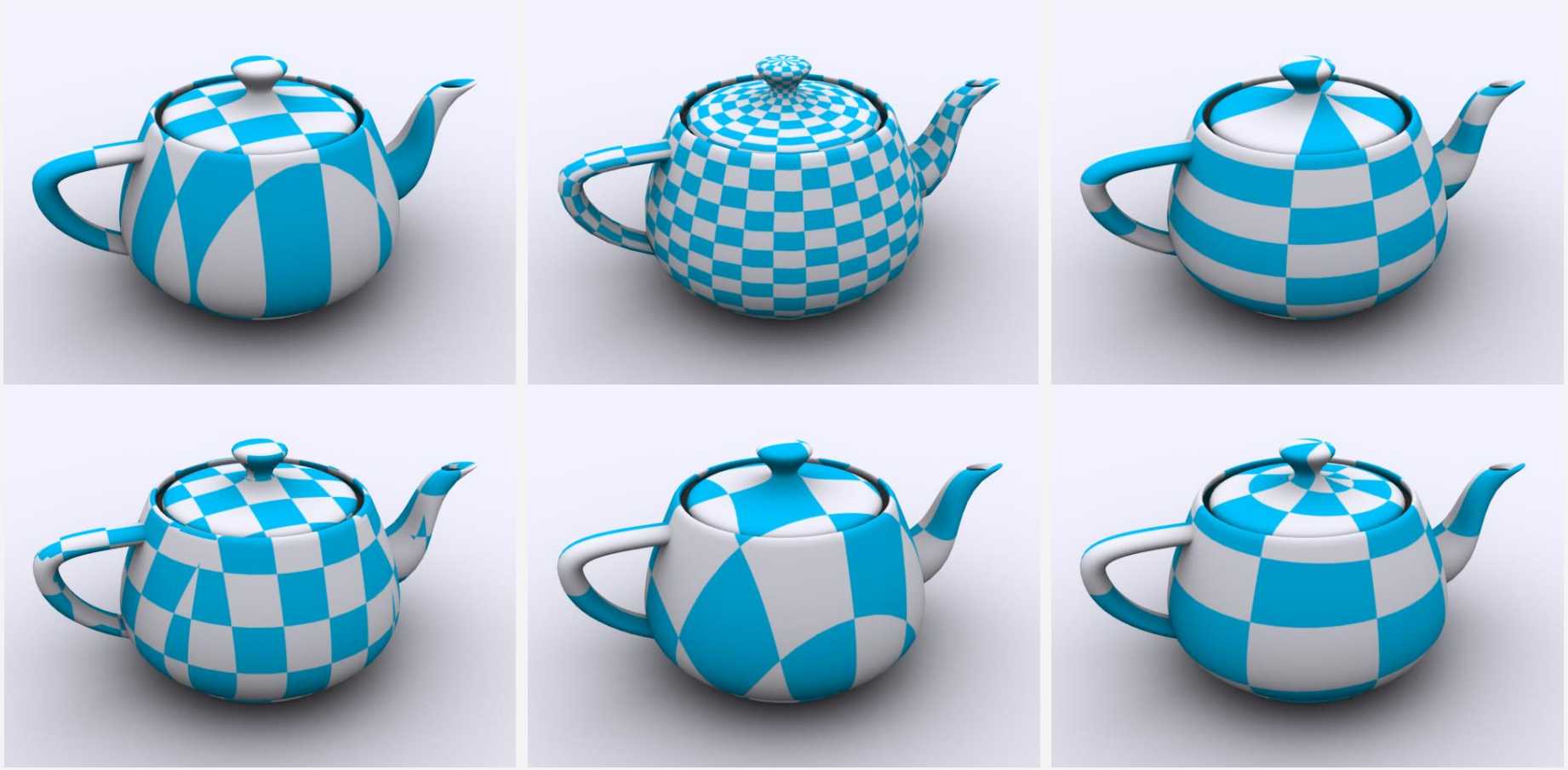
$$u = \frac{x}{\sqrt{x^2 + y^2 + z^2}}$$

$$v = \frac{y}{\sqrt{x^2 + y^2 + z^2}}$$



http://tobias.preclik.de/codeblog/?p=9

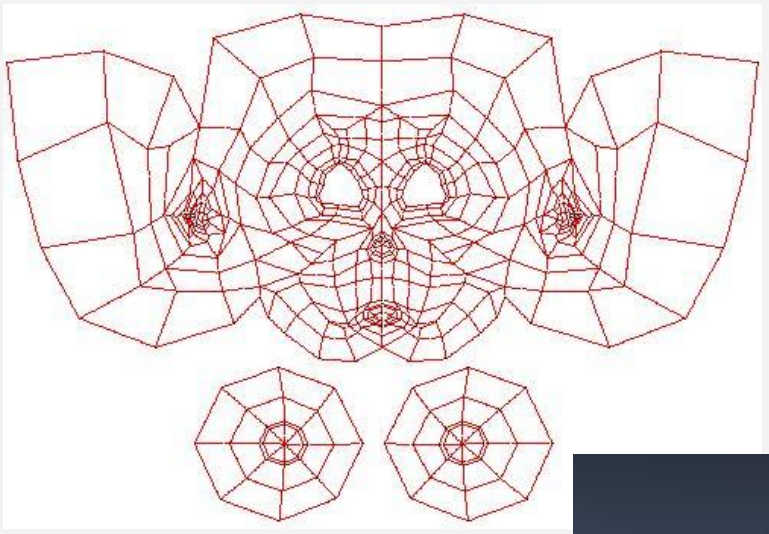
Different UV mappings

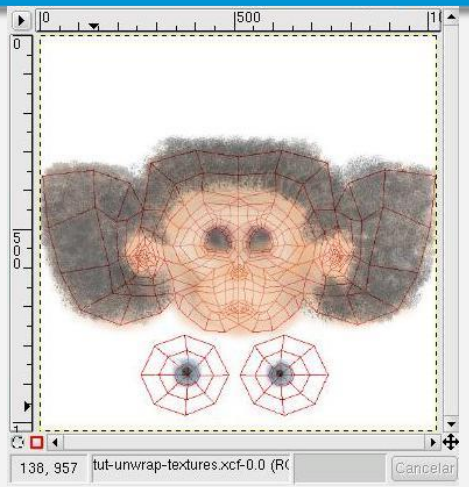


UV mappings

3DS MAX:

- Planar
- Face
- Cylindrical
- Box
- Shrink wrap
- Spherical





or... Unwrap

- http://www.ru.is/kennarar/hannes/useful/BlenderManual/htmlI/