INFOGR – Computer Graphics

Jacco Bikker - April-July 2015 - Lecture 3: “Geometry”

Welcome!

Today’s Agenda:

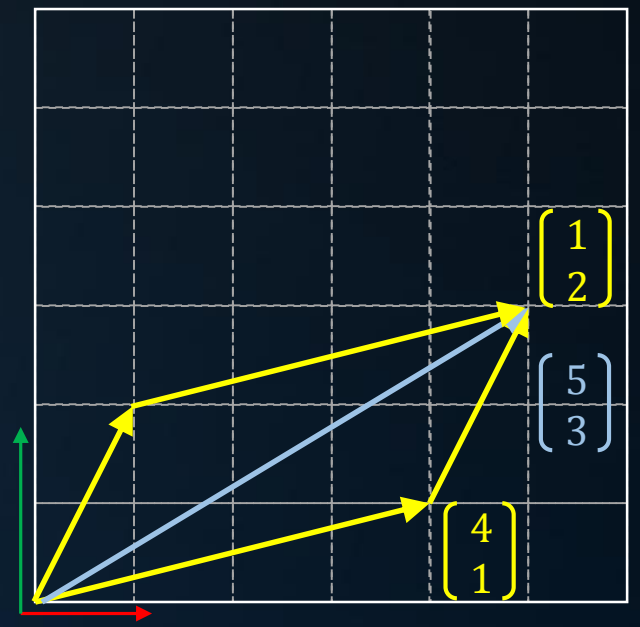
- 2D Primitives
- 3D Primitives
- Textures

2D Primitives

Recap

Last time: vectors and their properties:

- Magnitude, direction
- Scalar product
- Null vector, normal
- Parallel, linear (in)depence
- Commutative addition & subtraction
- Dot product, cross product



Concepts:

- $\mathbb{R}^d$ spaces
- (orthonormal) 2D basis, Cartesian
- Left handed, right handed

2D Primitives

Implicit representation

Implicit curve:

$f(x, y) = 0$

Function f maps two-dimensional points to a real value, i.e.

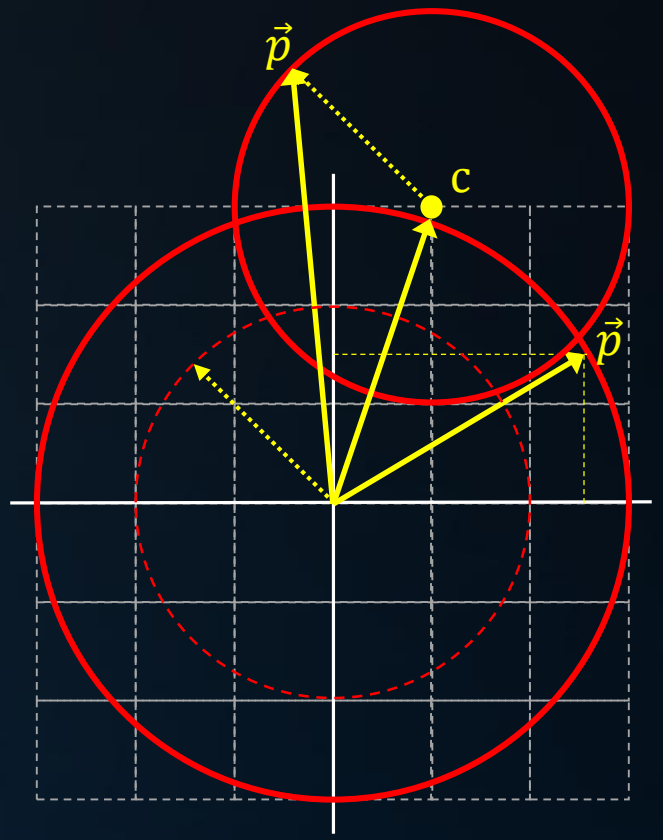
$(x, y) \rightarrow f(x, y)$

The points for which this value is 0 are on the curve.

Example: circle

$x^2 + y^2 - r^2 = 0$

If $p = (x, y)$ is a point on the circle, and
$\vec{p}$ is a vector from the origin to p, it’s length must be r, so $\|\vec{p}\| = r$.



Example: circle with center c and radius r:

$(x\text{-}c_x)^2 + (y - c_y)^2 - r^2 = 0$

2D Primitives

Implicit representation

Implicit curve:

$f(x, y) = 0$

Function f maps two-dimensional points to a real value, i.e.

$(x, y) \rightarrow f(x, y)$

The points for which this value is 0 are on the curve.

Example: line

$y = \frac{2}{3}x + 1$

Slope-intersect form:

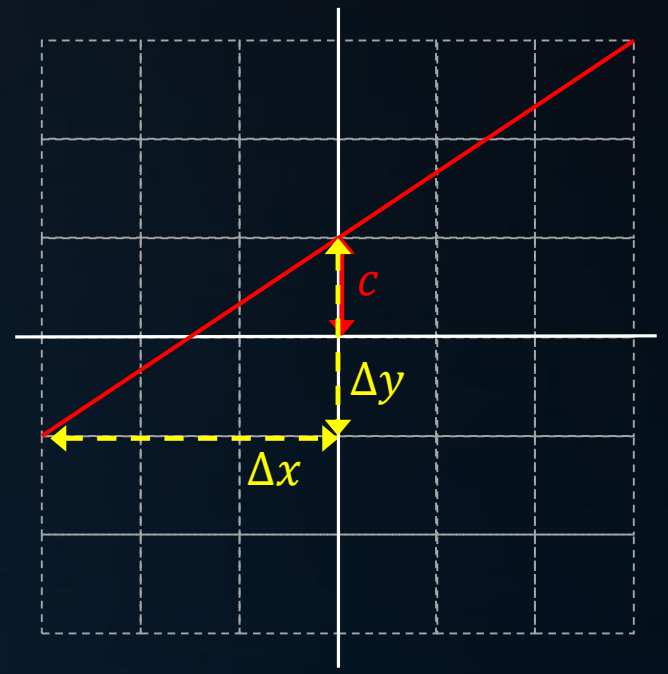
$y = ax + c$

Implicit form:

$-ax + y - c = 0$

In general:

$Ax + By + C = 0$



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

2D Primitives

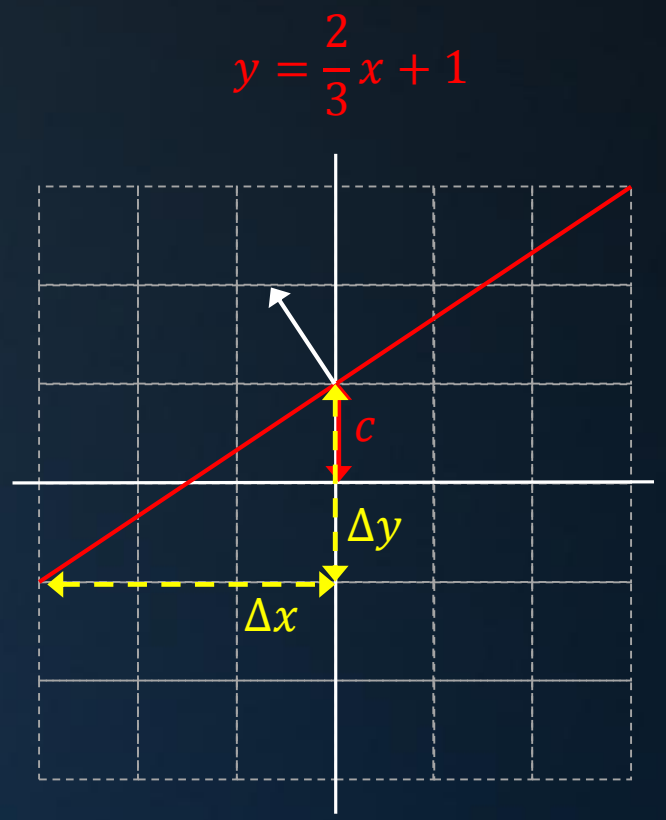
Implicit line representation

$Ax + By + C = 0$

In this case:

$\mathrm{A} = -\frac{2}{3}$, $\mathrm{B} = 1$, $\mathrm{C} = -1$

The vector (A,B) is a *normal* of the line.



Slope-intersect form:

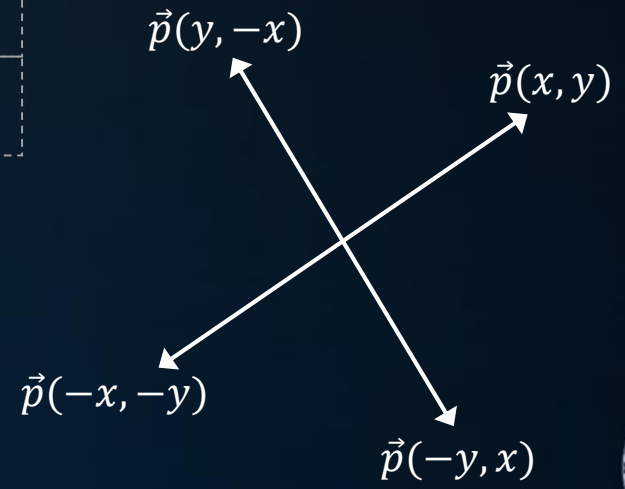
$y = ax + c$

Implicit form:

$-ax + y - c = 0$

General form:

$Ax + By + C = 0$



2D Primitives

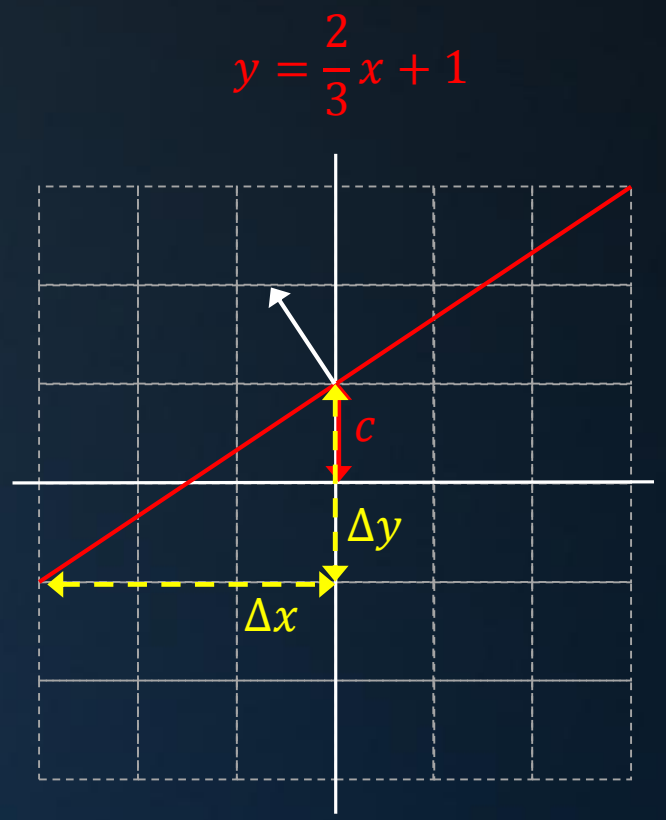
Implicit line representation

$Ax + By + C = 0$

In this case:

$\mathrm{A} = -\frac{2}{3}$, $\mathrm{B} = 1$, $\mathrm{C} = -1$

The vector (A,B) is a *normal* of the line.



We can use the normal to calculate the distance of a point to the line:

$d = \vec{N} \cdot p + C$

For $p = (3,3)$:

$$d = -\frac{2}{3} * 3 + 1 * 3 - 1$$
$$= -2 + 3 - 1 = 0$$

For $p = (0,0)$:

$$d = -\frac{2}{3} * 0 + 1 * 0 - 1$$
$$= -1\ (?)$$

2D Primitives

Implicit line representation

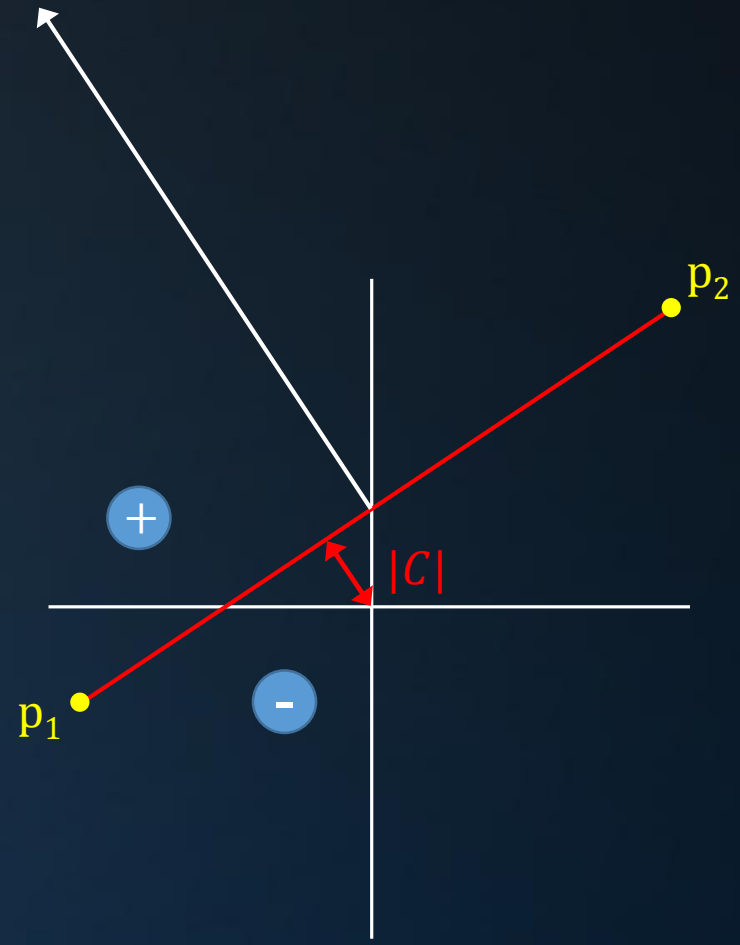
$$Ax + By + C = 0$$

Given point p_1 and p_2, we determine A, B and C as follows:

$$\vec{l} = p_2 - p_1$$

$$\vec{N} = (-l_y, l_x)$$

$$A = N_x\,, B = N_y\,, C = -(\vec{N} \cdot p_1)$$



It is convenient to normalize the normal:
Only when $\|\vec{N}\| = 1$, |C| is the distance of the line to the origin.

Test with the line from the previous slides:

$p_1 = (-3, -1)$
$p_2 = (3,3)$

$\vec{l} = (6,4)$
$\vec{N} = (-4,6)$
$A = -4, B = 6$
$C = -((-4 * -3) + (6 * -1))$
$\quad = -6$

$$-4x + 6y - 6 = 0$$

$$-\frac{2}{3}x + y - 1 = 0$$

2D Primitives

Parametric representation

Parametric curve:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} g(t) \\ h(t) \end{pmatrix}$$

Example: line

$$p_0 = (x_{p0}, y_{p0}), p1 = (x_{p1}, y_{p1})$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{p0} \\ y_{p0} \end{pmatrix} + t \begin{pmatrix} x_{p1} - x_{p0} \\ y_{p1} - y_{p0} \end{pmatrix}$$

Or

$$p(t) = p_0 + t(p_1 - p_0), t \in \mathbb{R}.$$

In this example:

p_1 is the *support vector;*
$p_1 - p_0$ is the *direction vector.*

2D Primitives

Slope-intercept:

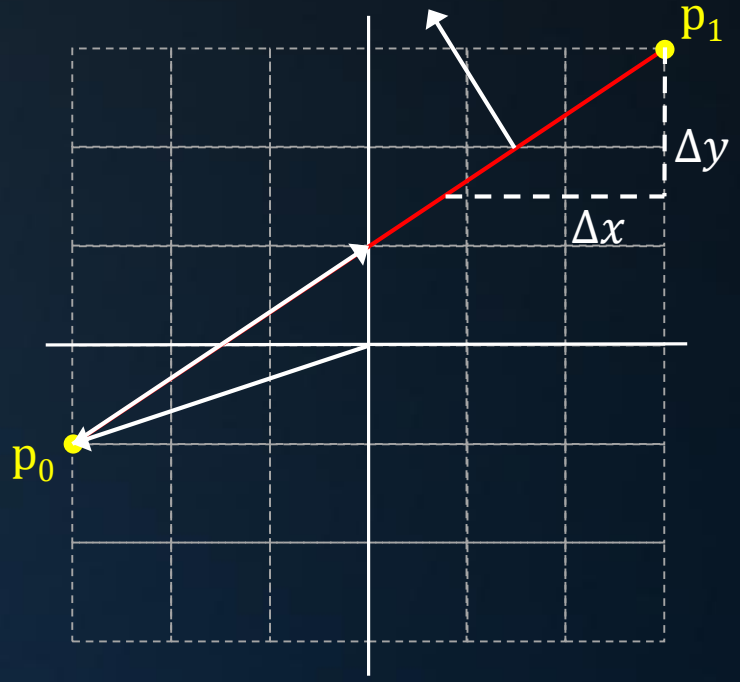
$$y = ax + c$$

Implicit representation:

$$-ax + y - c = 0$$
$$Ax + By + C = 0$$

Parametric representation:

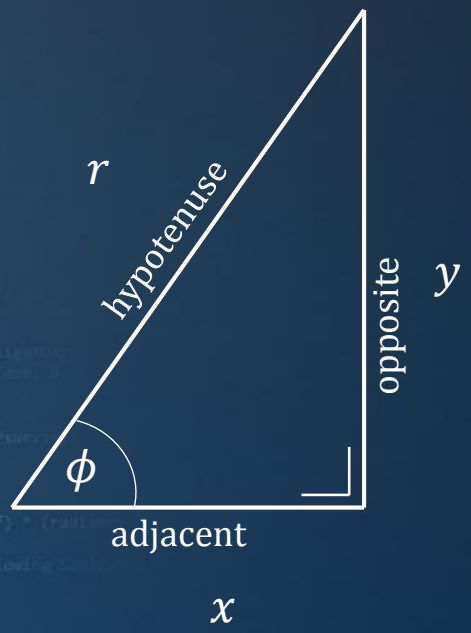
$$p(t) = p_0 + t(p_1 - p_0)$$



2D Primitives

Circle - parametric

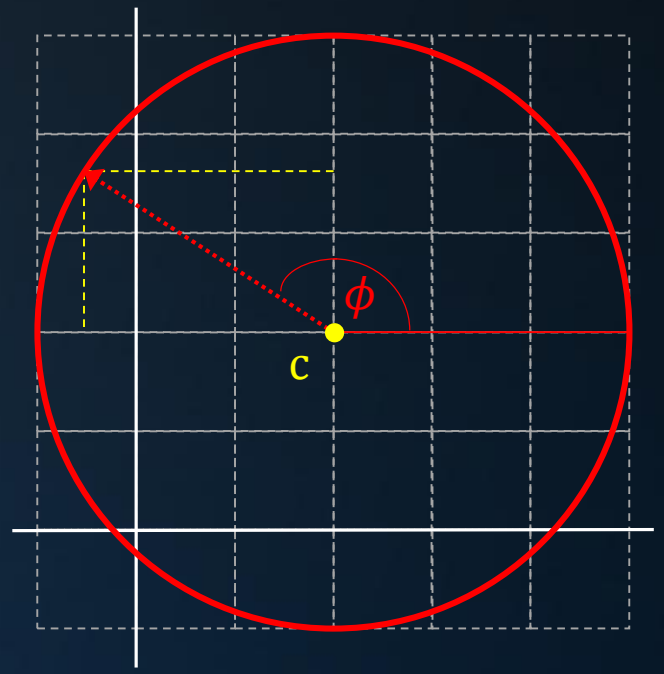
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_c + r\cos\phi \\ y_c + r\sin\phi \end{pmatrix}$$



$$\cos\phi = \frac{x}{r}$$

$$sin\,\phi = \frac{y}{r}$$

$$tan\,\phi = \frac{y}{x}$$



SOH CAH TOA

Today’s Agenda:

- 2D Primitives
- 3D Primitives
- Textures

3D Primitives

Circle – sphere (implicit)

Recall: the implicit representation for a circle with radius r and center c is:

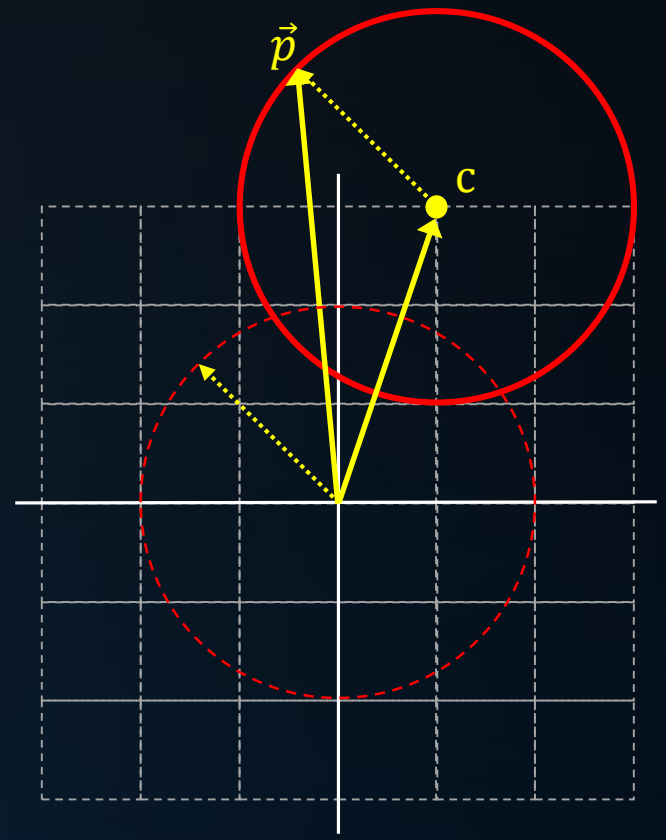
$$(x - xc)^2 + (y - yc)^2 - r^2 = 0$$

or: $\| \mathrm{p} - \mathrm{c} \|^2 - r^2 = 0$ ➔ $\| p - c \| = r$

In $\mathbb{R}^3$, we get:

$$(x - c_x)^2 + (y - c_y)^2 + (z - cz)^2 - r^2 = 0$$

or: $\| p - c \| = r$



3D Primitives

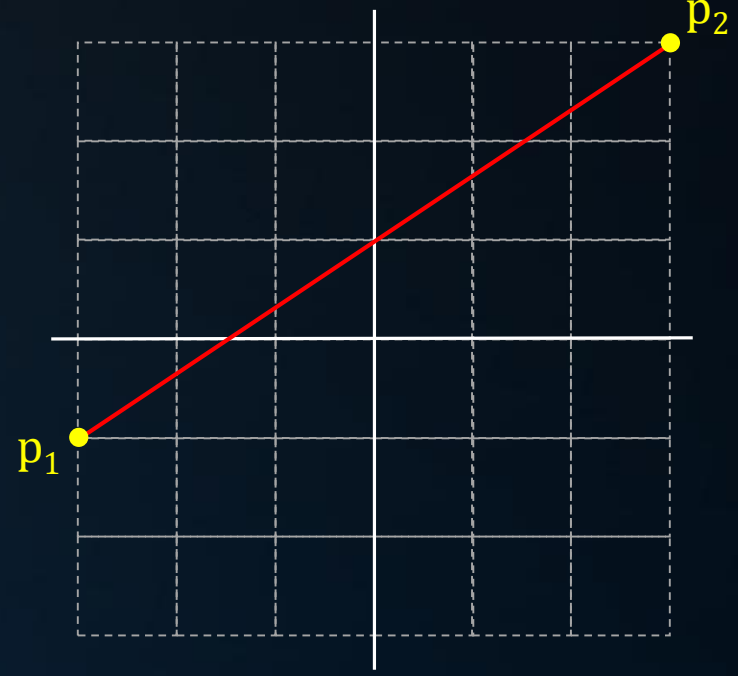
Line – plane (implicit)

Recall: the implicit representation for a line is:

$$Ax + By + C = 0$$

In $\mathbb{R}^3$, we get a plane:

$$Ax + By + Cz + D = 0$$



3D Primitives

Parametric surfaces

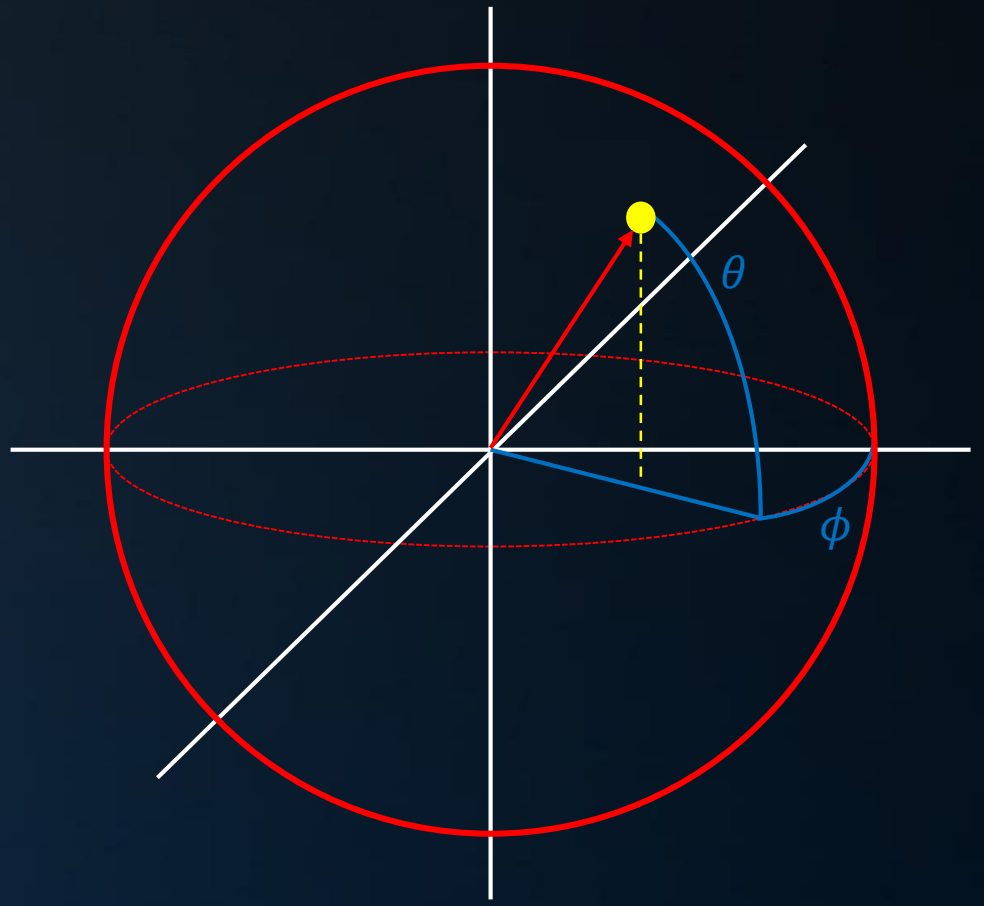
A parametric surface needs two parameters:

$$x = f(u, v),$$
$$y = g(u, v),$$
$$z = h(u, v).$$

For example, a sphere:

$$x = r \cos\phi \sin\theta,$$
$$y = r \sin\phi \sin\theta,$$
$$z = r \cos\theta.$$

Doesn’t look very convenient (compared to the implicit form), but it will prove useful for texture mapping.



3D Primitives

Parametric planes

Recall the parametric line definition:

$$p(t) = p_0 + t(p_1 - p_0)$$

For a plane, we need to parameters:

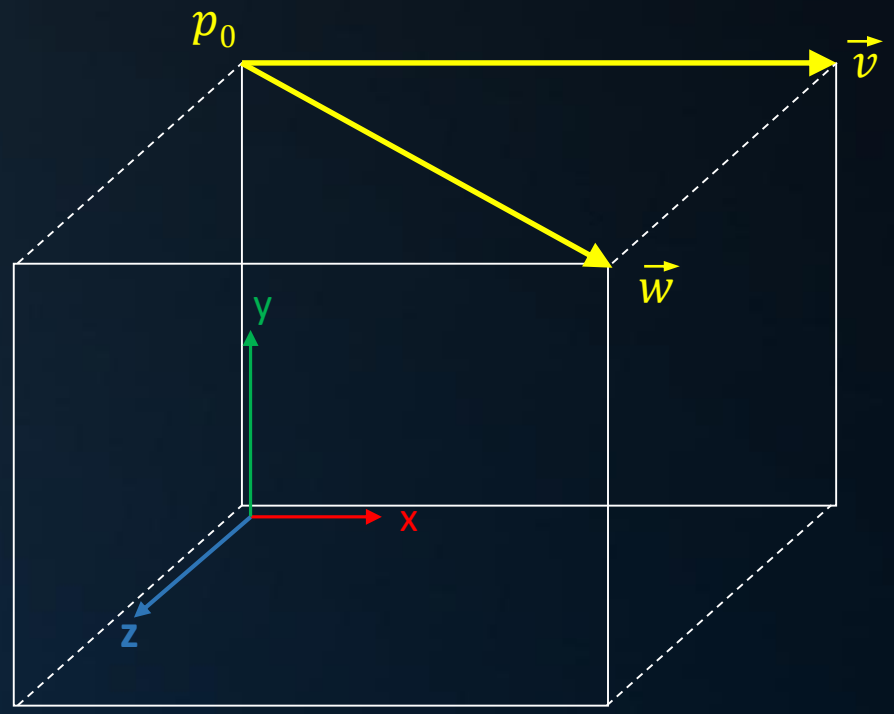
$$p(s,t) = p_0 + s(p_1 - p_0) + t(p_2 - p_0)$$

or:

$$p(s,t) = p_0 + s\vec{v} + t\vec{w}$$

where:

- p_0 is a point on the plane;
- $\vec{v}$ and $\vec{w}$ are two linearly independent vectors on the plane;
- $s, t \in \mathbb{R}$.



Today’s Agenda:

- 2D Primitives
- 3D Primitives
- Textures

Textures

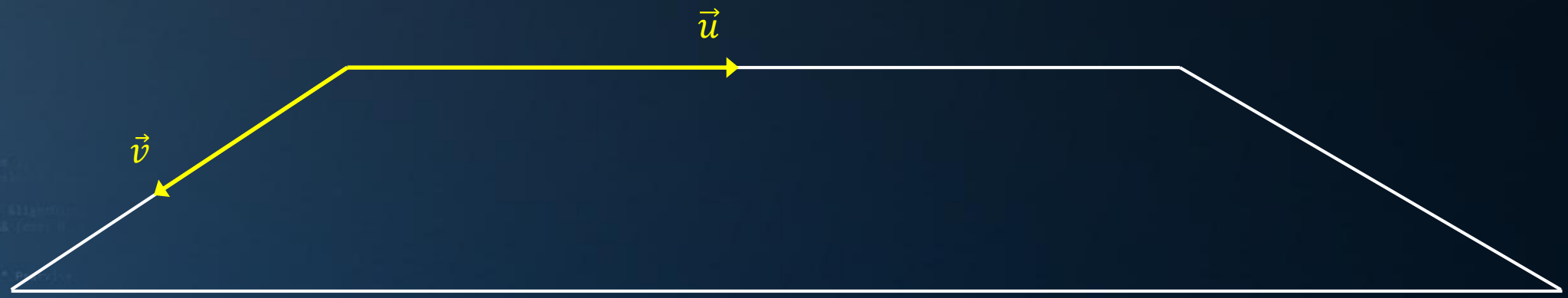
Back to the world of graphics…

Given a plane: $y = 0$ (i.e., with a normal vector $(0,1,0)$).

Two vectors on the plane define a basis: $\vec{u} = (1,0,0)$ and $\vec{v} = (0,0,1)$.

Using these vectors, any point on the plane can be reached: $P = \lambda_1\vec{u} + \lambda_2\vec{v}$.

We can now use λ_1, λ_2 to define a color at P: $F(\lambda_1, \lambda_2) = \cdots$.



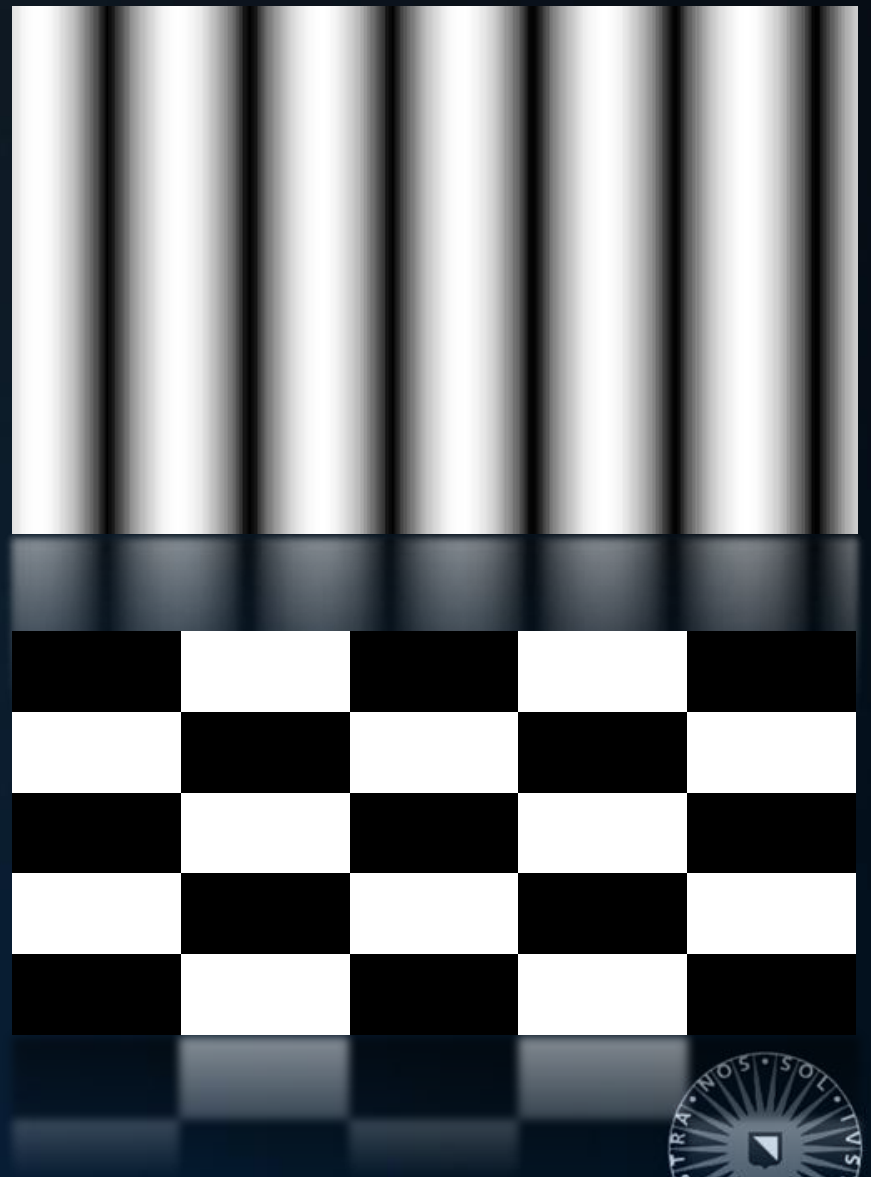
Textures

Example:

$$F(\lambda_1, \lambda_2) = \sin(\lambda_1)$$

Another example:

$$F(\lambda_1, \lambda_2) = ((\text{int})(2 * \lambda_1) + (\text{int})\lambda_2) \,\&\, 1$$

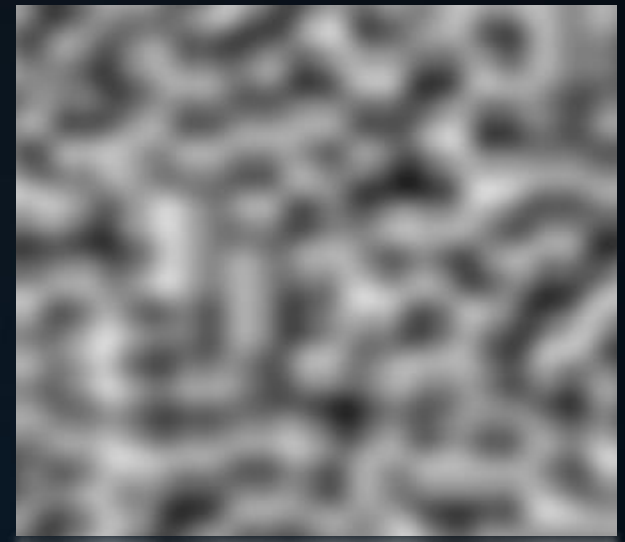


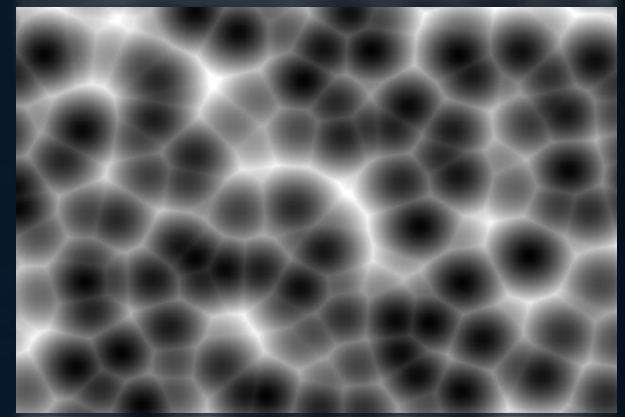
Textures

Other examples (not explained here):

Perlin noise
Details: http://www.noisemachine.com/talk1

Voronoi / Worley noise
Details: “A cellular texture basis function”, S. Worley, 1996.





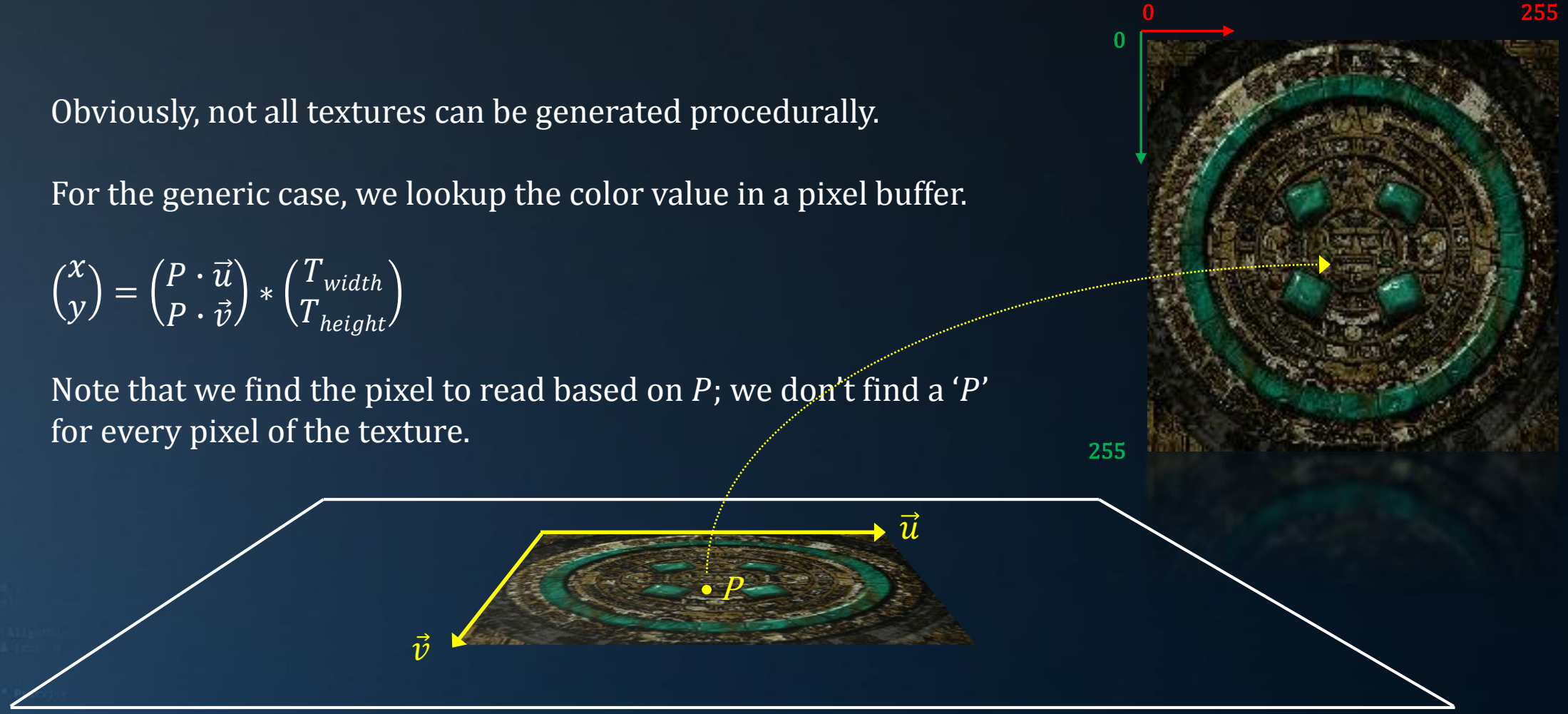
Textures

Obviously, not all textures can be generated procedurally.

For the generic case, we lookup the color value in a pixel buffer.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} P \cdot \vec{u} \\ P \cdot \vec{v} \end{pmatrix} * \begin{pmatrix} T_{width} \\ T_{height} \end{pmatrix}$$

Note that we find the pixel to read based on P; we don’t find a ‘P’ for every pixel of the texture.



Textures

Retrieving a pixel from a texture:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} P \cdot \vec{u} \\ P \cdot \vec{v} \end{pmatrix} * \begin{pmatrix} T_{width} \\ T_{height} \end{pmatrix}$$

We don’t want to read outside the texture. To prevent this, we have two options:

1. Clamping

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} clamp(P \cdot \vec{u}, 0, 1) \\ clamp(P \cdot \vec{v}, 0,1) \end{pmatrix} * \begin{pmatrix} T_{width} \\ T_{height} \end{pmatrix}$$

2. Tiling

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} frac(P \cdot \vec{u}) \\ frac(P \cdot \vec{v}) \end{pmatrix} * \begin{pmatrix} T_{width} \\ T_{height} \end{pmatrix}$$

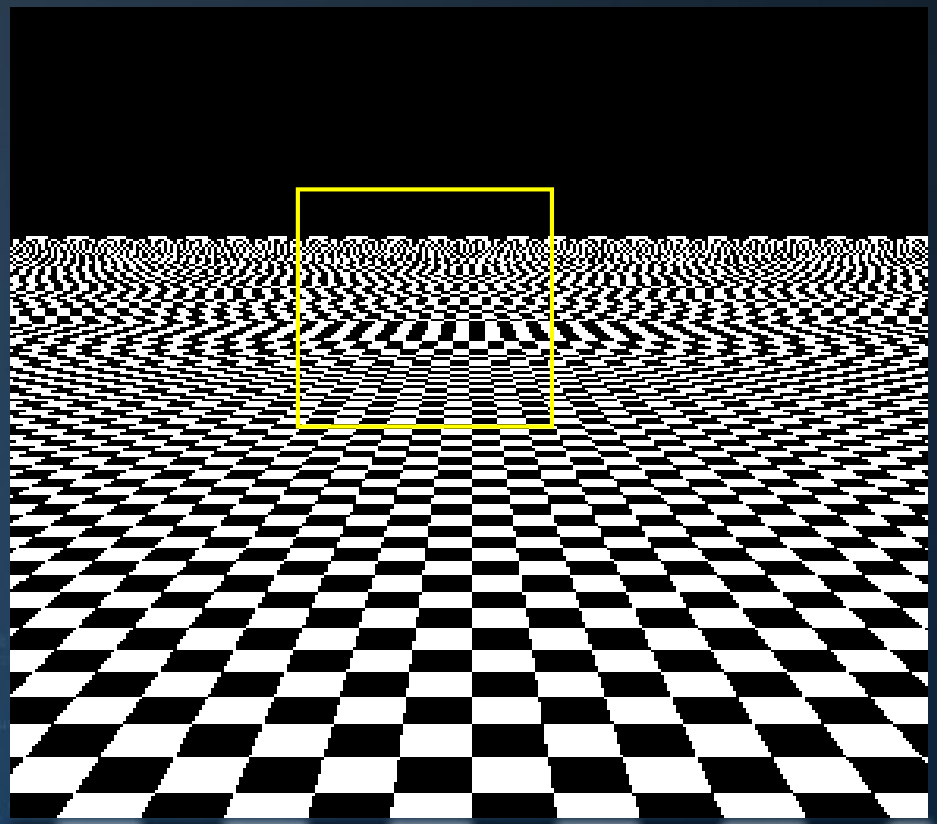
Tiling is efficiently achieved using a bitmask. This requires texture dimensions that are a power of 2.

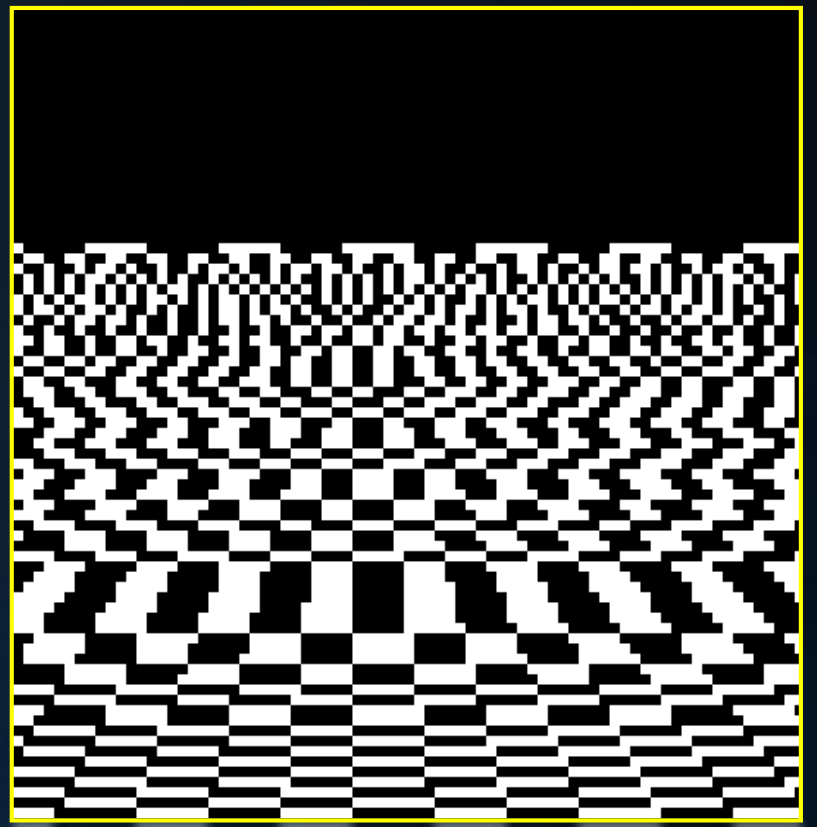
Textures

Texture mapping: oversampling

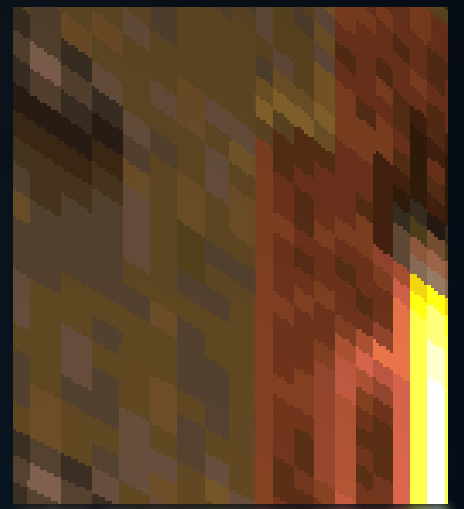
Textures

Texture mapping: undersampling





Textures

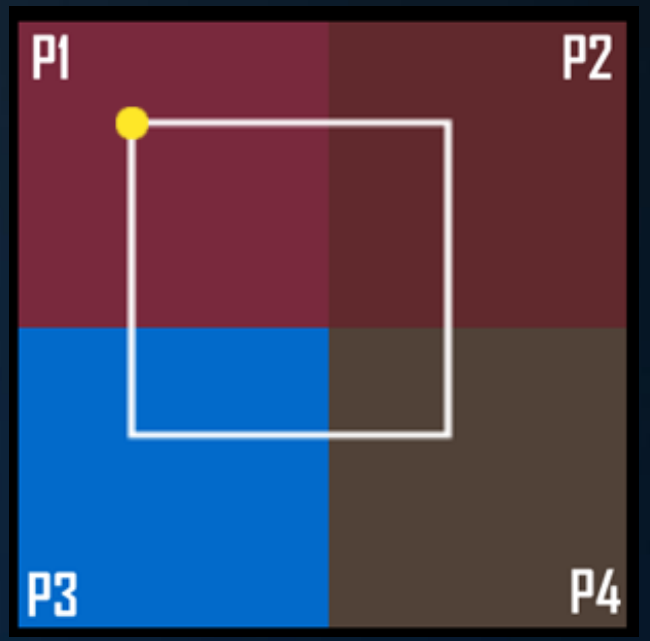


Fixing oversampling

Oversampling: reading the same pixel from a texture multiple times.
Symptoms: blocky textures.

Remedy: bilinear interpolation:
Instead of clamping the pixel location to the nearest pixel, we read from four pixels.

$$w_{p1}: (1 - frac(x)) * (1 - frac(y))$$
$$w_{p2}: frac(x) * (1 - frac(y))$$
$$w_{p3}: \big(1 - frac(x)\big) * frac(y)$$
$$w_{p4}: 1 - (wP_1 + wP_2 + wP_3)$$



Textures

Fixing oversampling

Textures

Fixing undersampling

Undersampling: skipping pixels while reading from a texture.
Symptoms: Moiré, flickering, noise.

Remedy: MIP-mapping.

The texture is reduced to 25% by averaging 2x2 pixels. This is repeated until a 1x1 image remains.

When undersampling occurs, we switch to the next MIP level.


NO MIPMAPS


WITH MIPMAPS

Textures

Trilinear interpolation: blending between MIP levels.

Today’s Agenda:

- 2D Primitives
- 3D Primitives
- Textures

INFOGR – Computer Graphics

Jacco Bikker - April-July 2015 - Lecture 3: “Geometry”

END of “Geometry”

next lecture: “3D Engine Fundamentals”