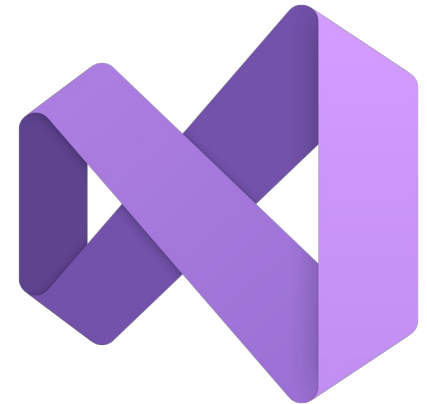
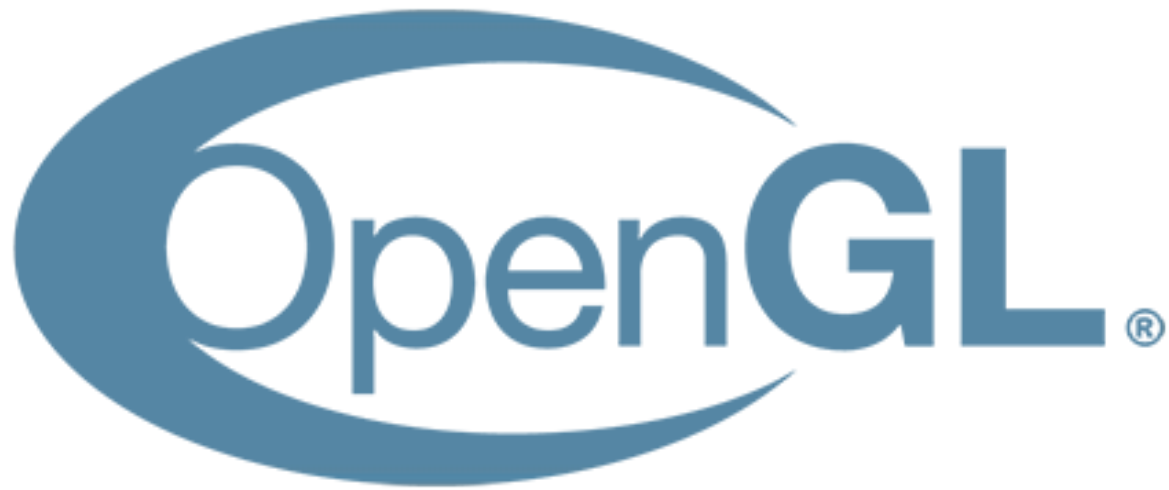

INTRODUCTION TO 3D GRAPHIC PROGRAMMING WITH C++ AND OPENGL

Šimon Potočňák



WHAT IS OPENGL

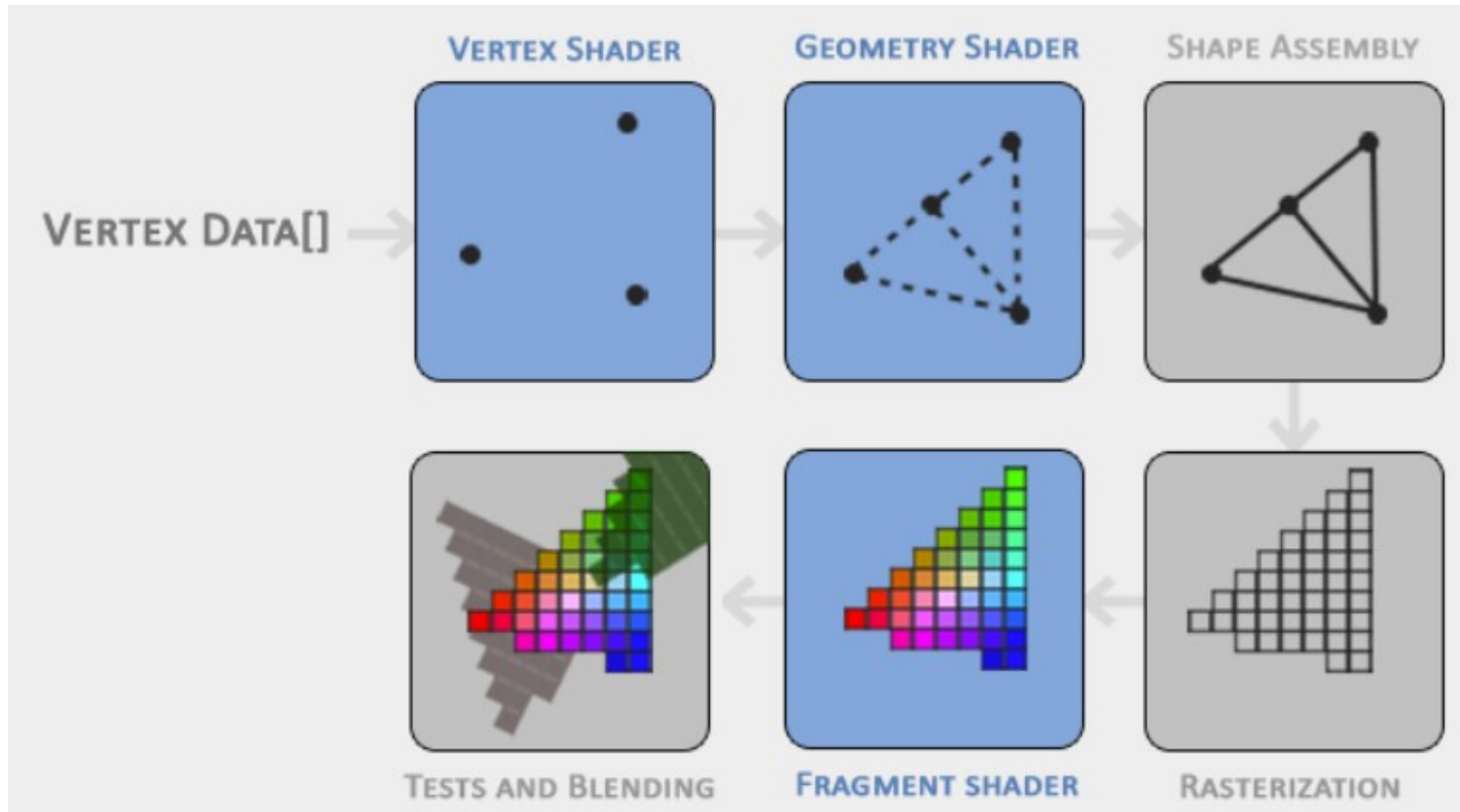


K H R O N O S[®]
G R O U P

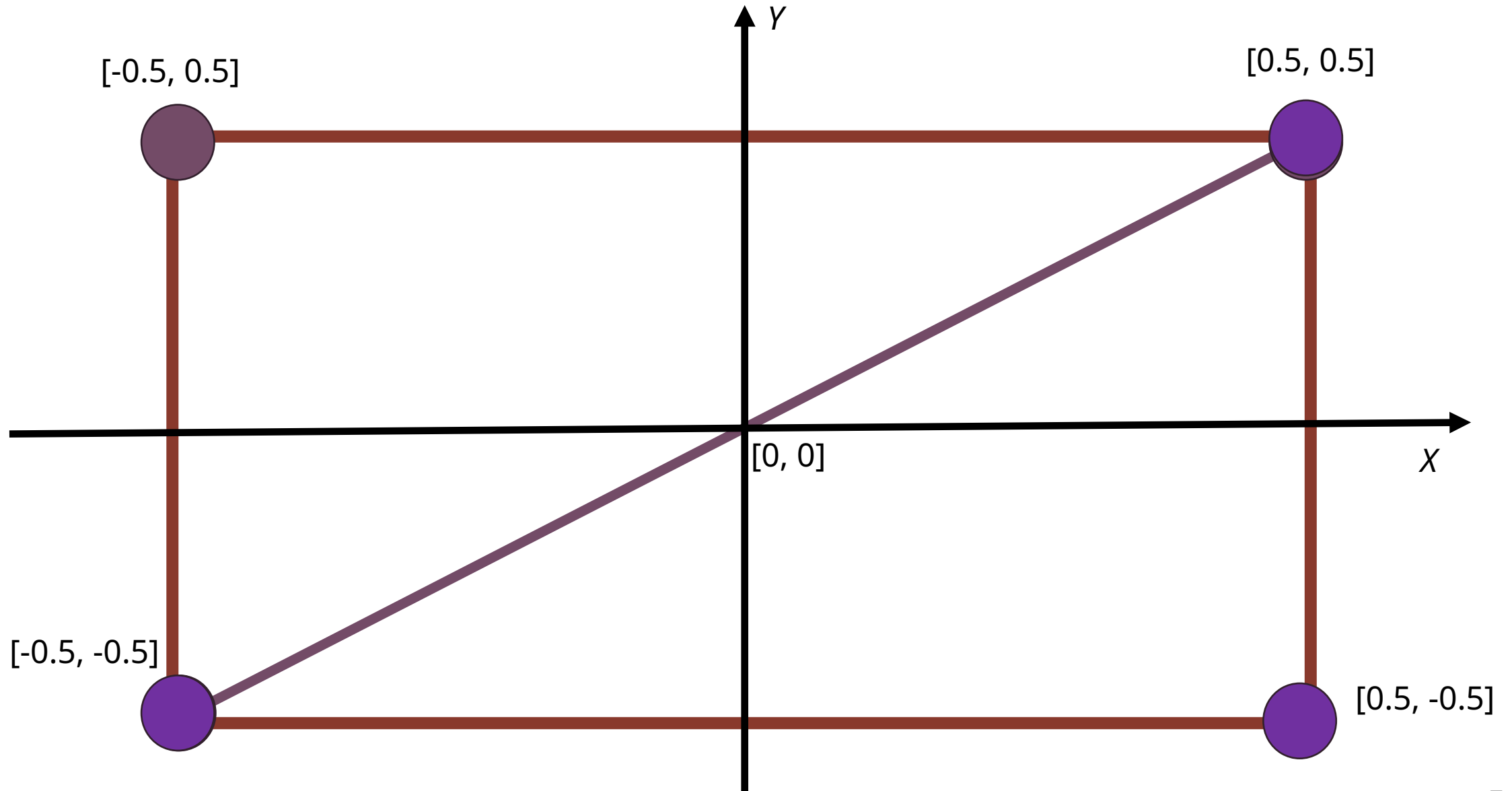
HOW THINGS ARE DRAWN

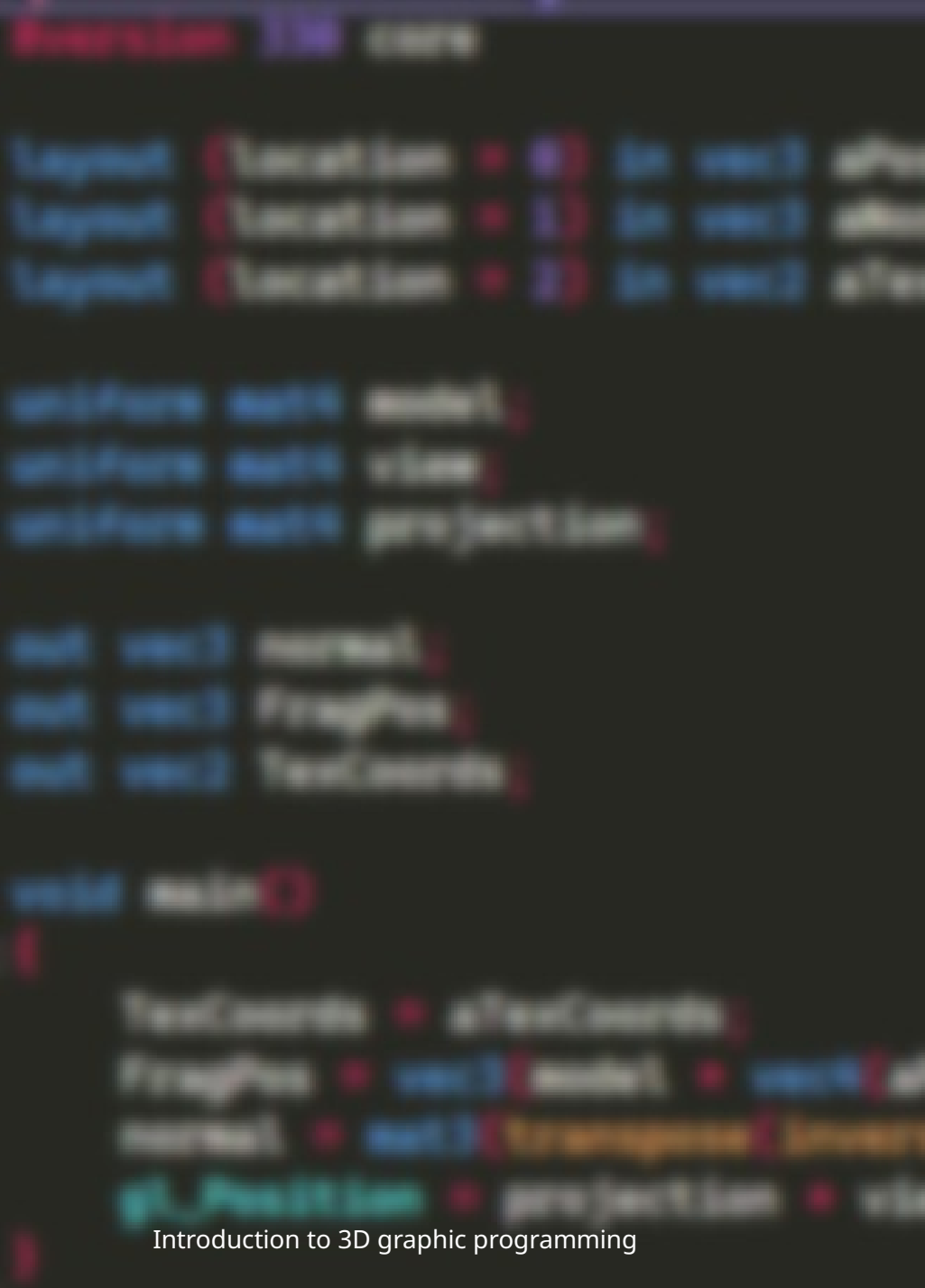


SHADERS



VERTEX SHADER





LET'S WRITE ONE OURSELVES

UNDERSTANDING VERTEX DATA

$-0.5f$	$-0.5f$	$-0.5f$	$0.0f$	$0.0f$	$-1.0f$	$0.0f$	$0.0f$
$0.5f$	$-0.5f$	$-0.5f$	$0.0f$	$0.0f$	$-1.0f$	$1.0f$	$0.0f$
$0.5f$	$0.5f$	$-0.5f$	$0.0f$	$0.0f$	$-1.0f$	$1.0f$	$1.0f$
$0.5f$	$0.5f$	$-0.5f$	$0.0f$	$0.0f$	$-1.0f$	$1.0f$	$1.0f$
$-0.5f$	$0.5f$	$-0.5f$	$0.0f$	$0.0f$	$-1.0f$	$0.0f$	$1.0f$
$-0.5f$	$-0.5f$	$-0.5f$	$0.0f$	$0.0f$	$-1.0f$	$0.0f$	$0.0f$

Positions 12 bites

Normal vectors 12 bites

UV coordinates 8 bites

Every vertex shader input is 8 floating point numbers
32 bites

```

//vec3 vec3;
vec3 vec3;

uniform vec3 lightPos;
uniform vec3 viewPos;

in vec3 normal;
in vec3 fragPos;
out vec3 texCoords;

vec3 lightColor = vec3(1.0f, 1.0f, 1.0f);

uniform sampler2D texture_diffuse;
uniform sampler2D texture_specular;

void main()
{
    vec3 N = normalize(normal);

    vec3 diffuseColor = texture(texture_diffuse, texCoords);
    vec3 specularColor = texture(texture_specular, texCoords);

    //ambient
    vec3 ambientStrength = lightColor * 0.5;
    vec3 ambient = ambientStrength * lightColor + diffuseColor;

    //diffuse
    vec3 lightDirection = normalize(lightPos - fragPos);
    float diffuseStrength = max(dot(N, lightDirection), 0.0);
    vec3 diffuse = diffuseStrength * lightColor + diffuseColor;

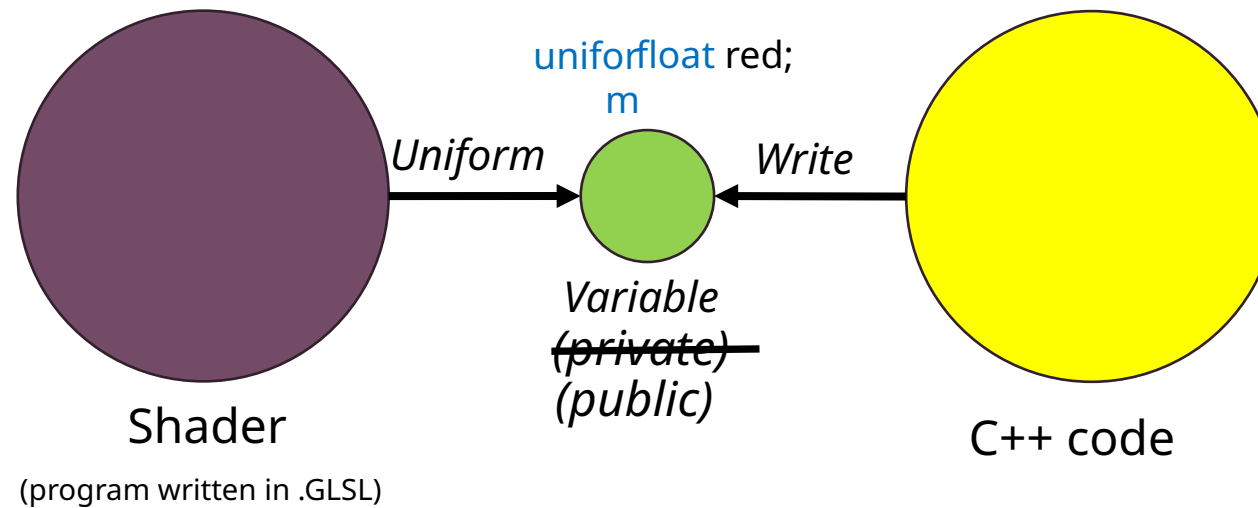
    //specular
    vec3 viewDir = normalize(viewPos - fragPos);
    vec3 reflectDir = reflect(-lightDirection, N);
    float specularStrength = pow(max(dot(viewDir, reflectDir), 0), 4);
    vec3 specular = specularStrength * lightColor + specularColor;

    //final
    vec3 finalColor = ambient + diffuse + specular;
}

```

FRAGMENT SHADER

UNIFORMS



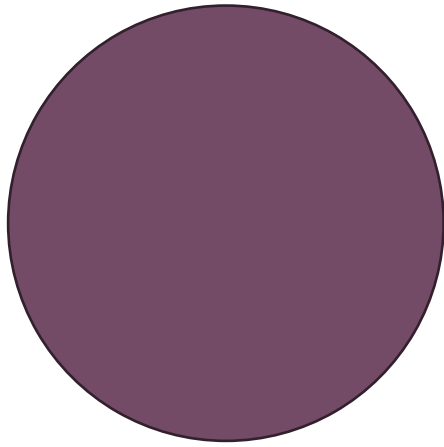


3D SPACE

MATRICES

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

MOVING POINTS



Point in space
`vec4(vec3(0.0,0.0,0.0))`

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

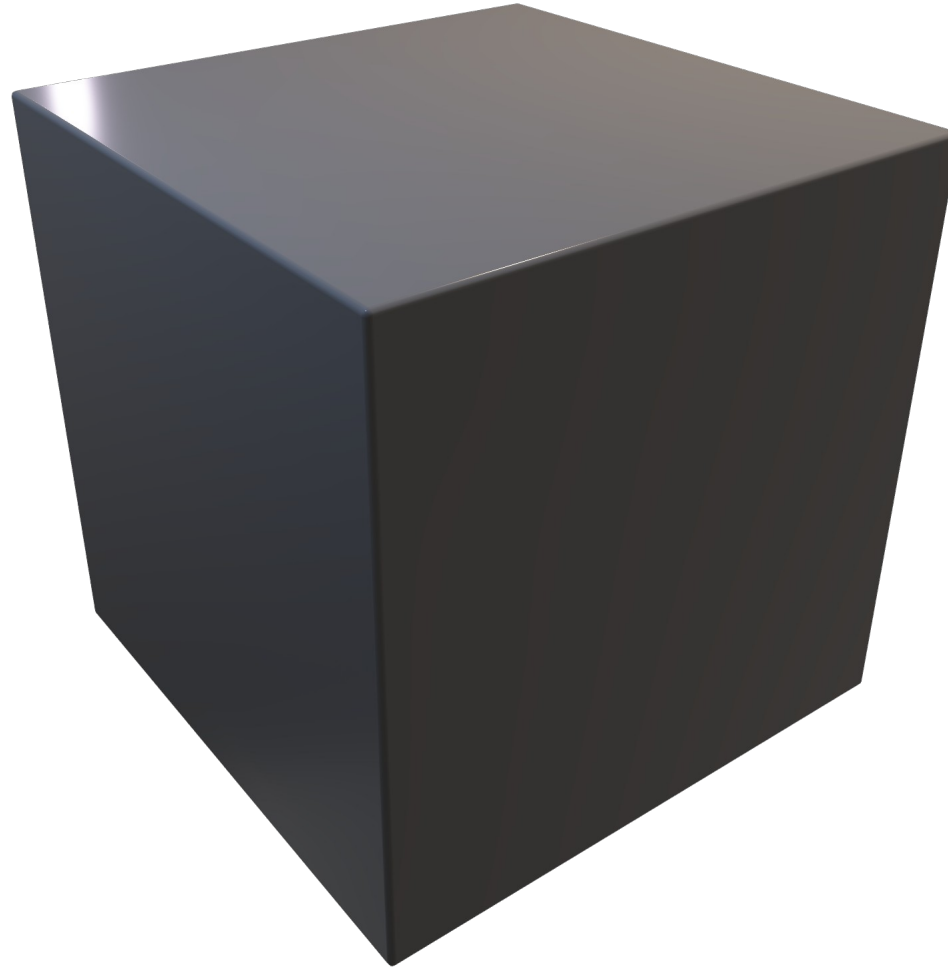
Matrix that will move
point to the right

Point in space
`vec4(vec3(1.0,0.0,0.0))`

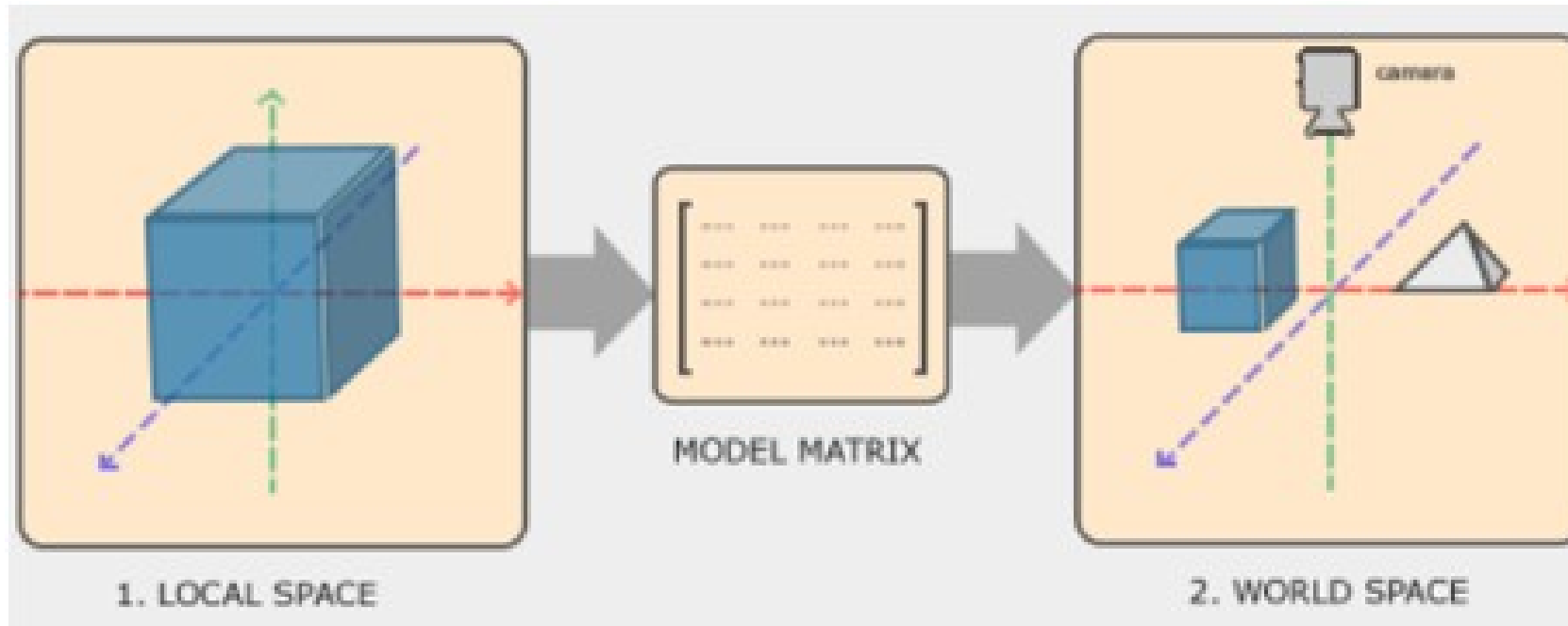
OPENGL MATHEMATIC LIBRARY

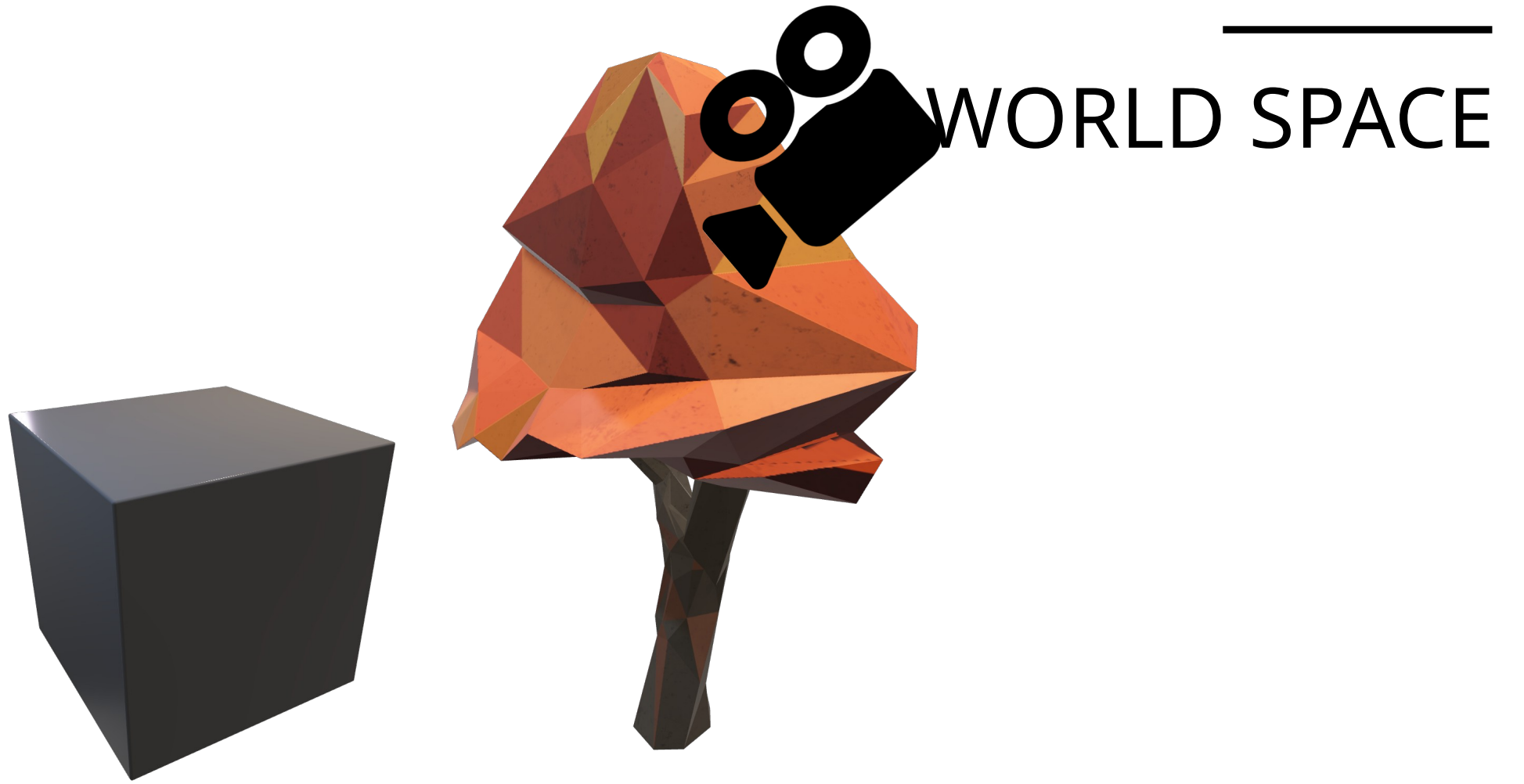


LOCAL SPACE

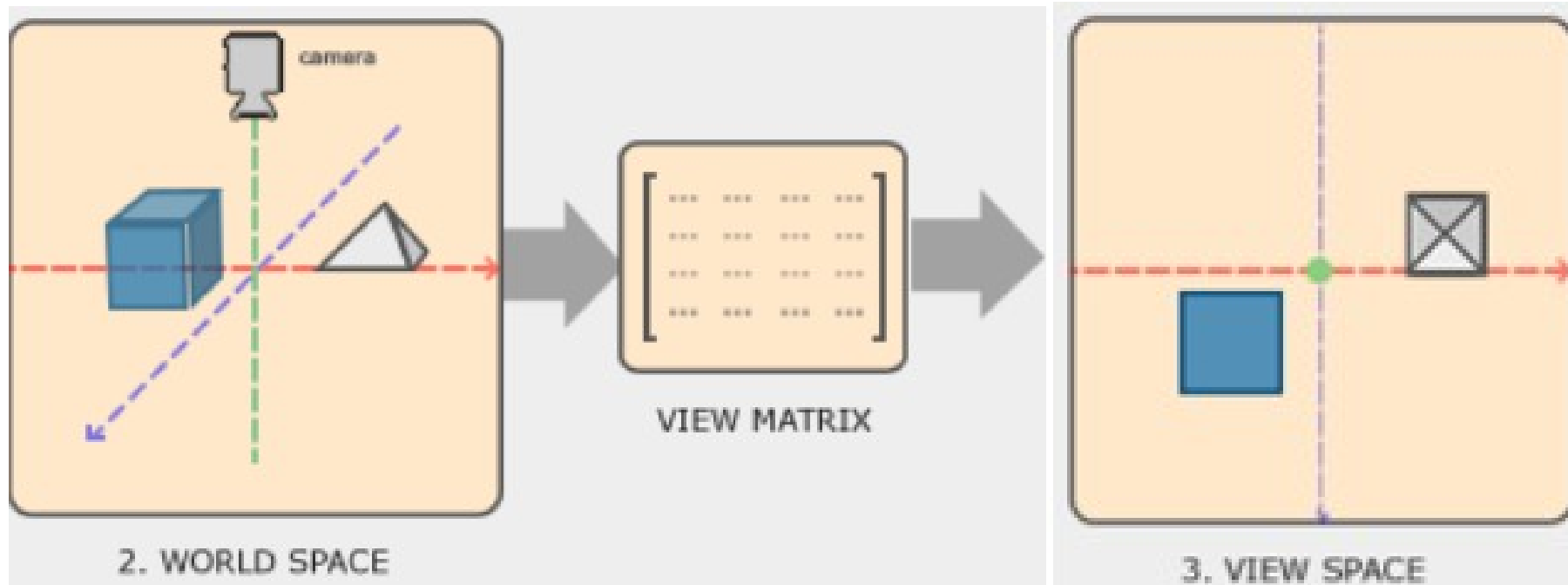


WORLD SPACE

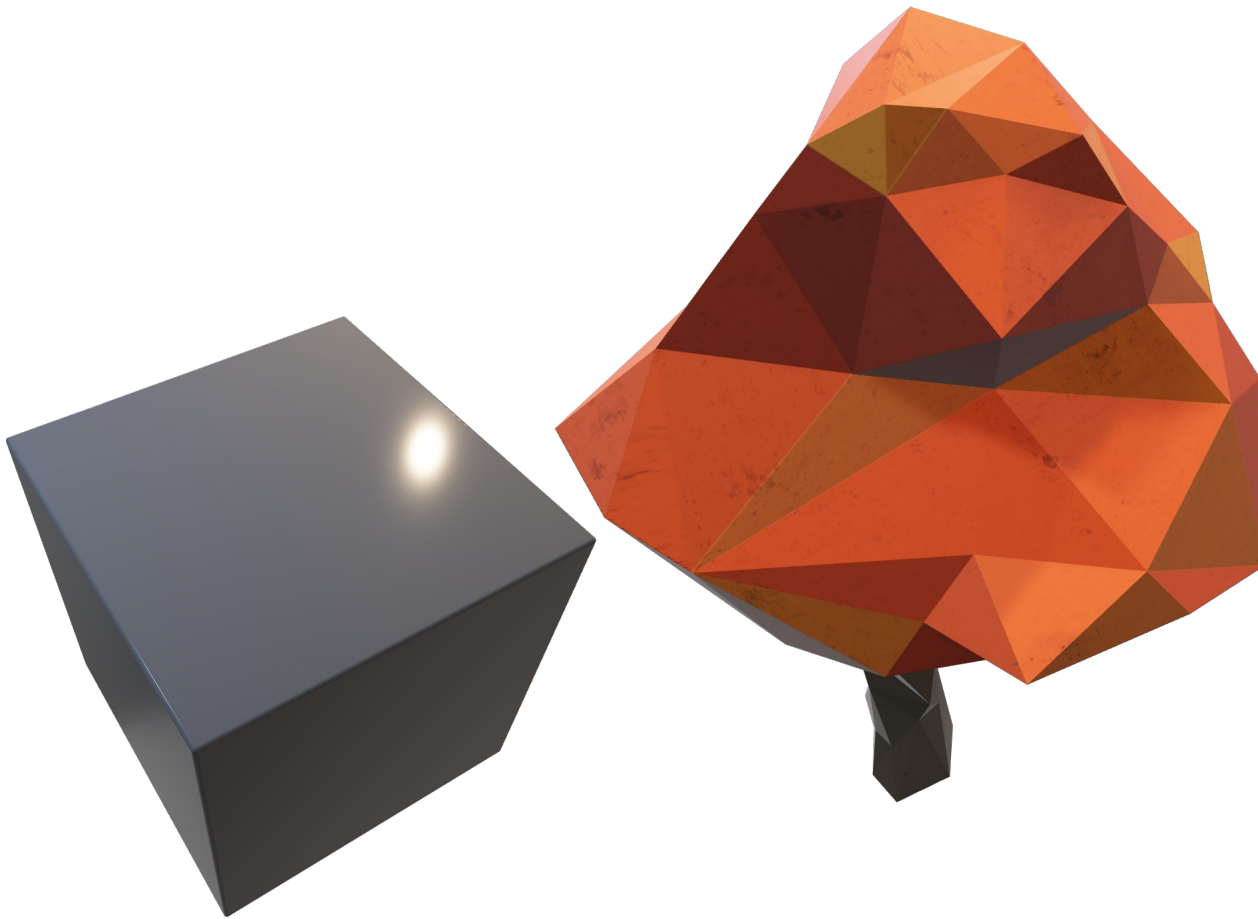




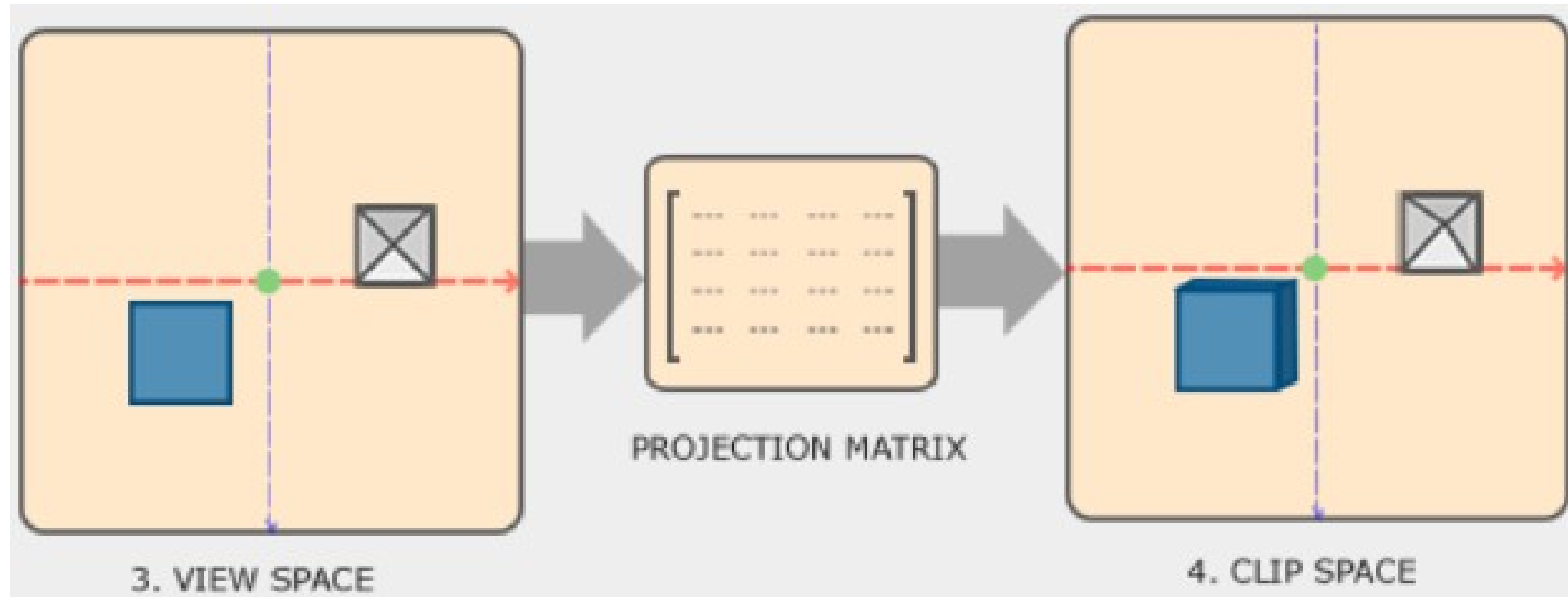
VIEW SPACE (ILLUSION OF THE CAMERA)



VIEW SPACE

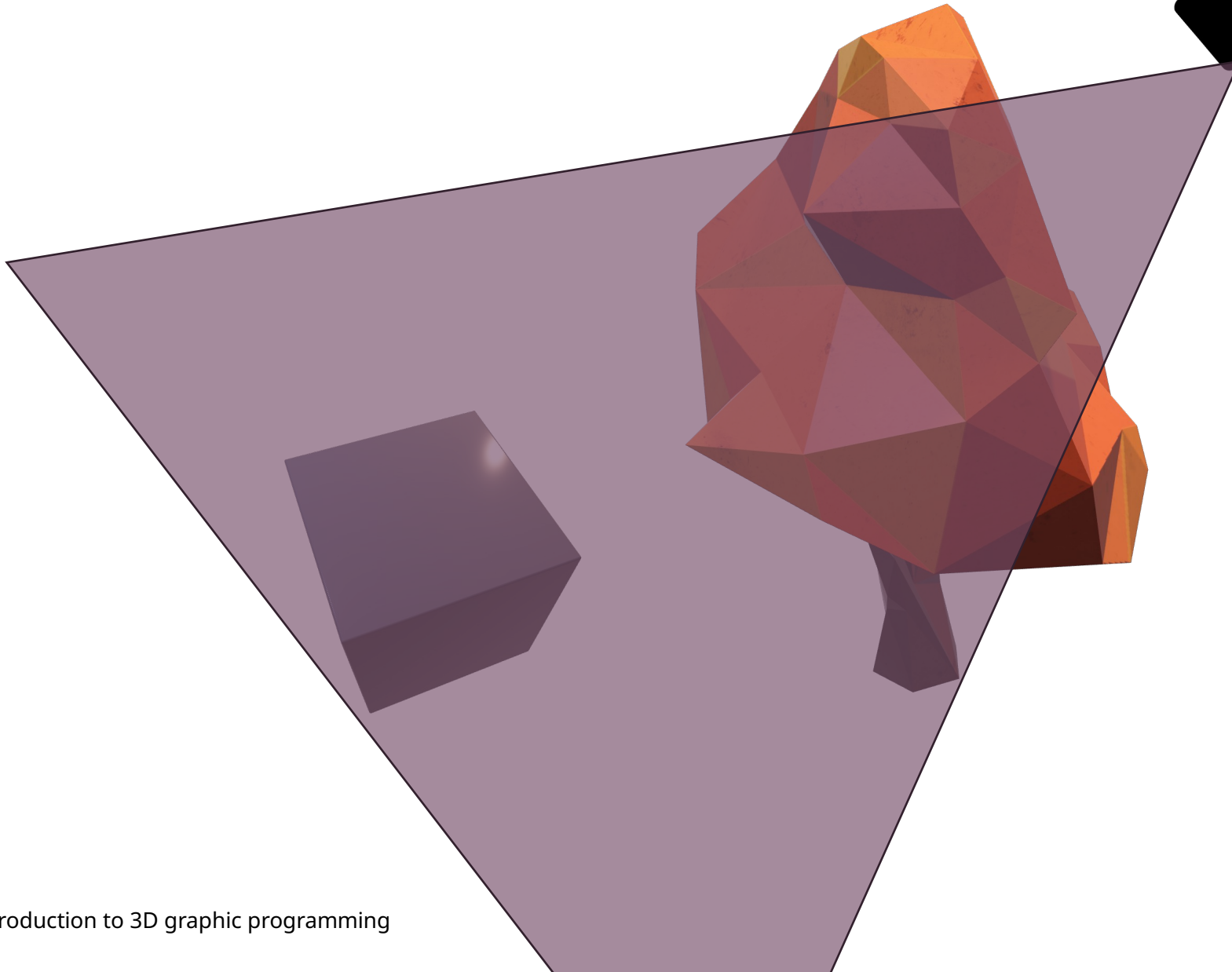


PROJECTION MATRIX





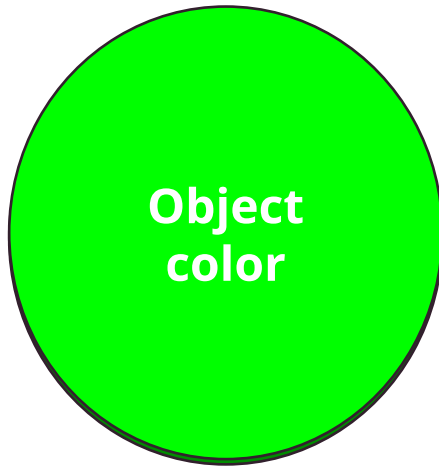
CLIP SPACE



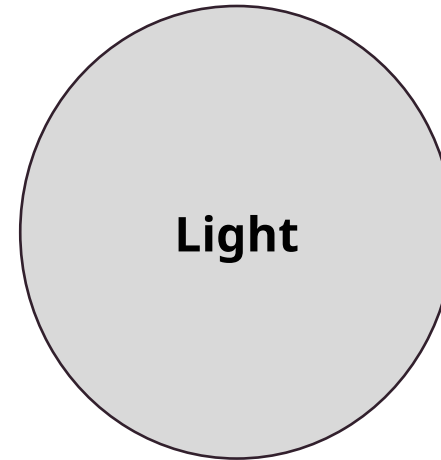


LIGHTNING

OBJECT COLOR AND LIGHT COLOR

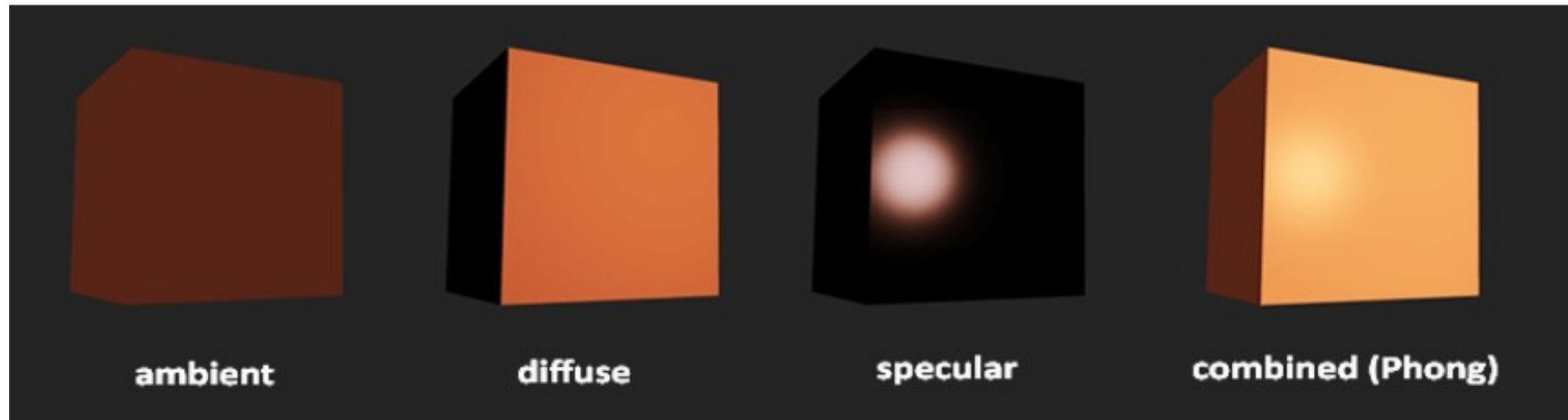


~~(1.0, 0.49, 0.00)~~



(0.7, 0.7, 0.7)

PHONG LIGHTNING MODEL

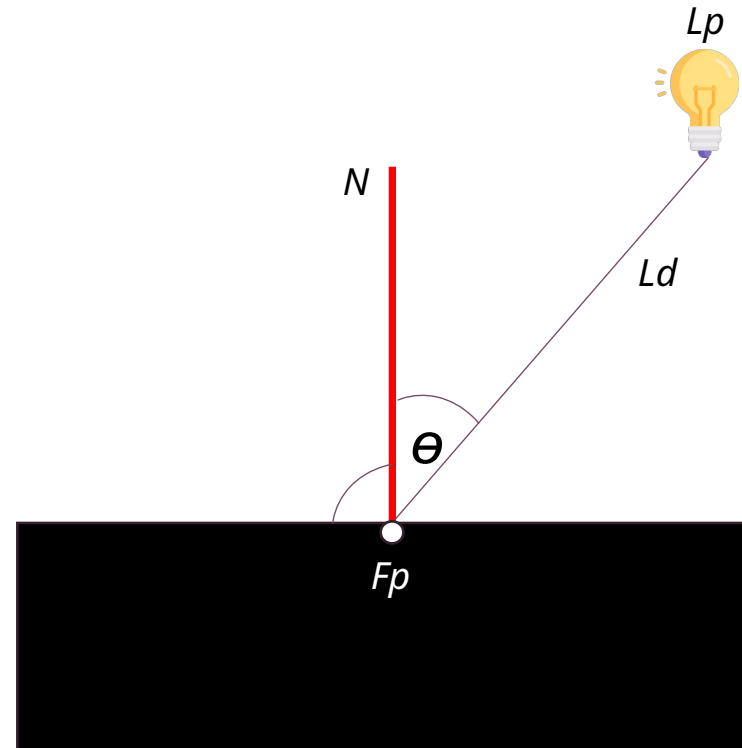


AMBIENT

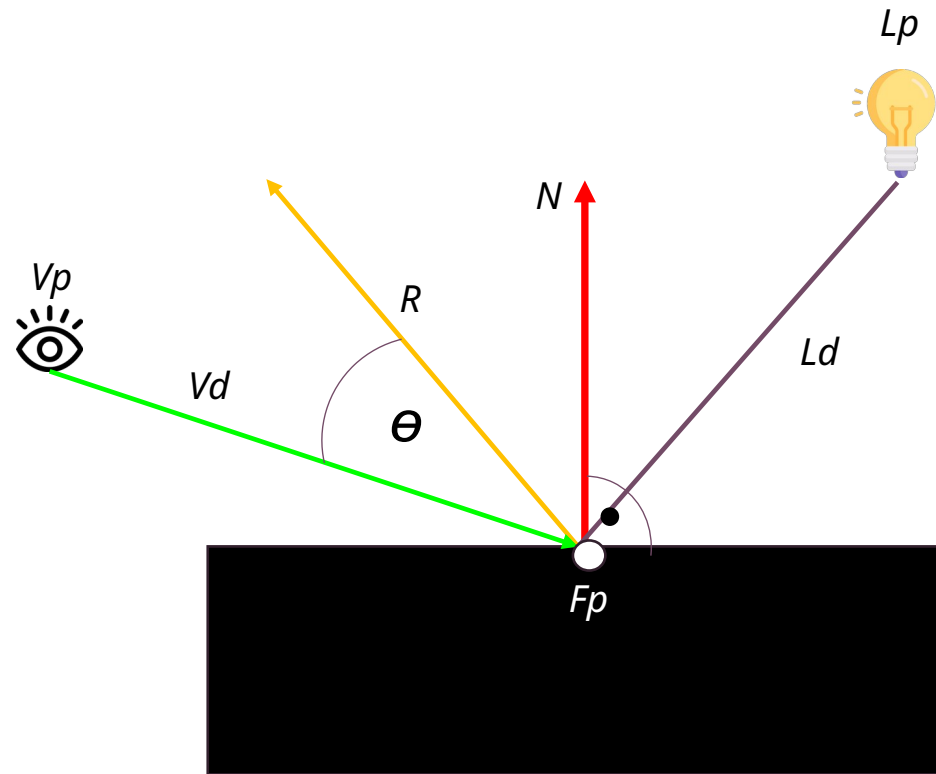
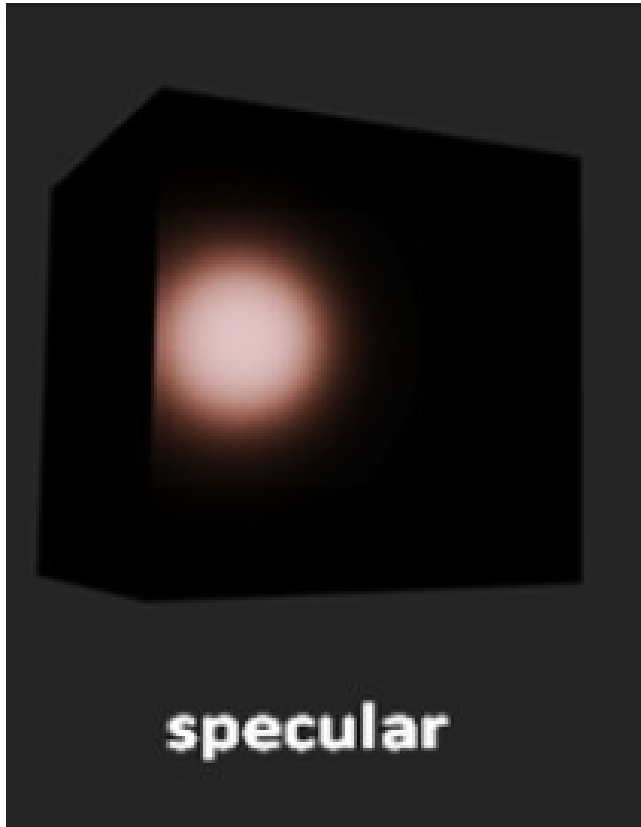


$$\begin{aligned} \textit{AmbientStrength} &= \textit{lightColor} * K \\ \textit{FragmentColor} &= \textit{AmbientStrength} * \textit{LightColor} * \textit{Object color} \end{aligned}$$

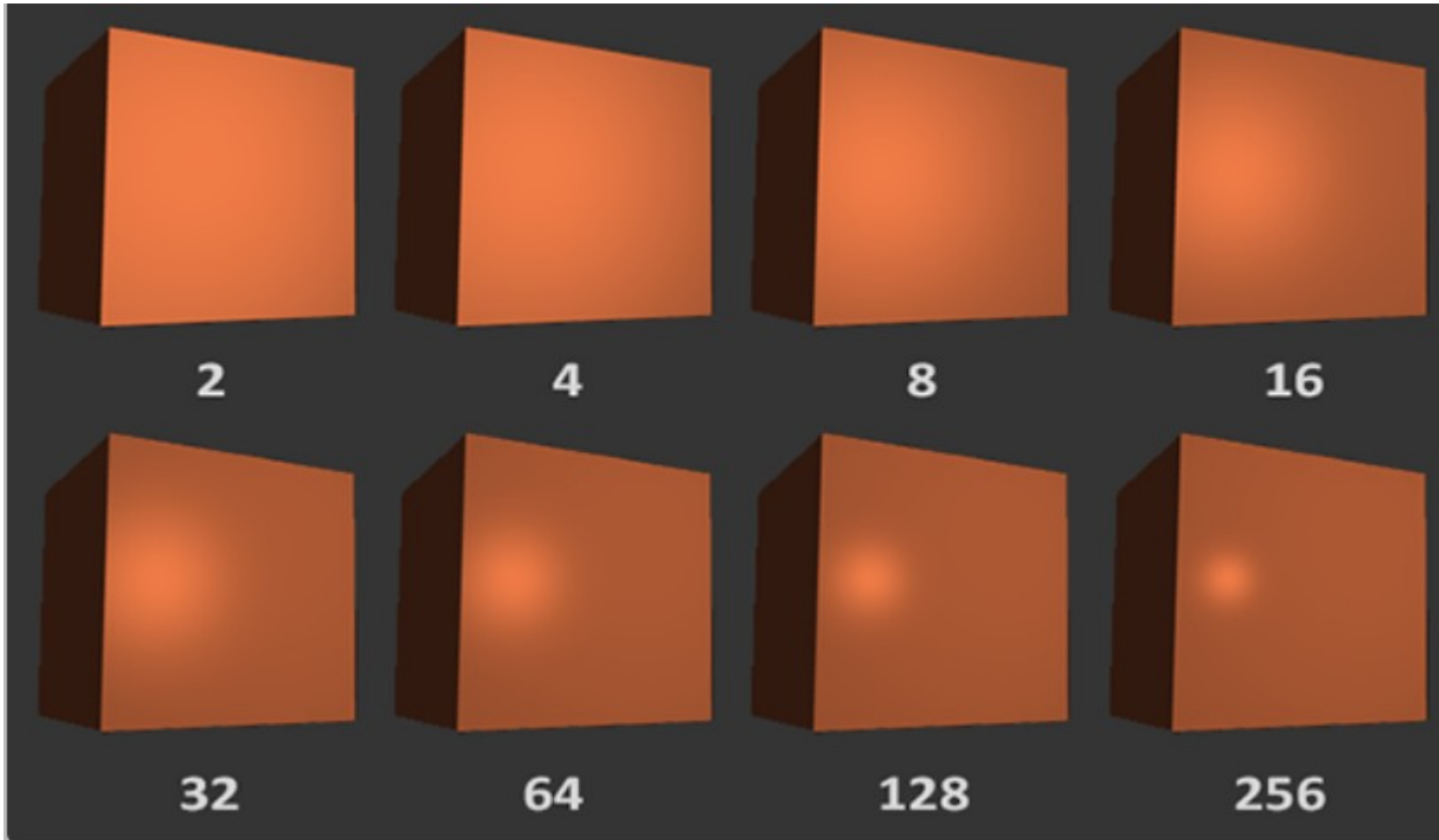
DIFFUSE LIGHT



SPECULAR LIGHT



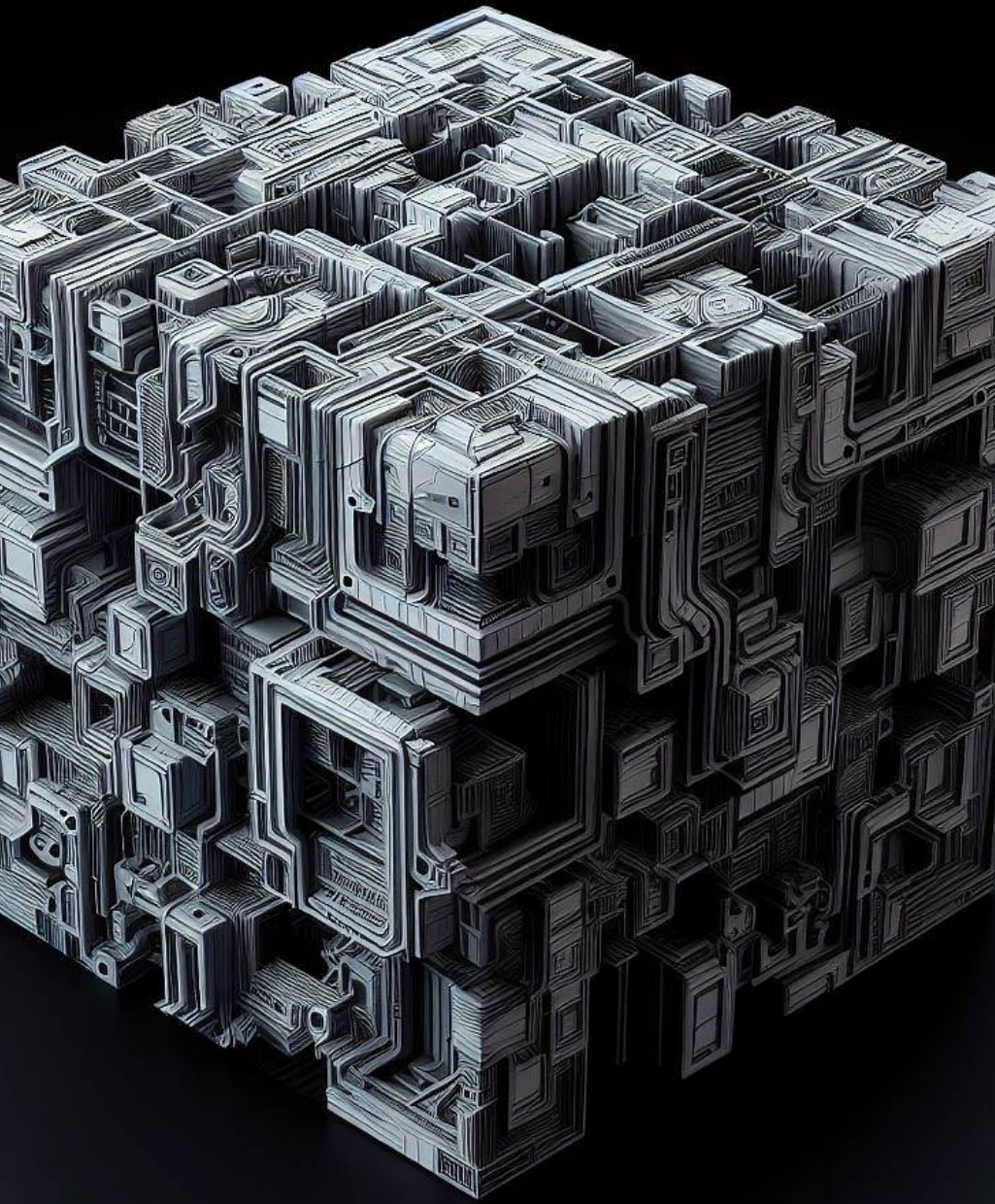
WHY 16 ?



RESULT



$$\textit{FinalColor} = \textit{Ambient} + \textit{Diffuse} + \textit{Specular}$$

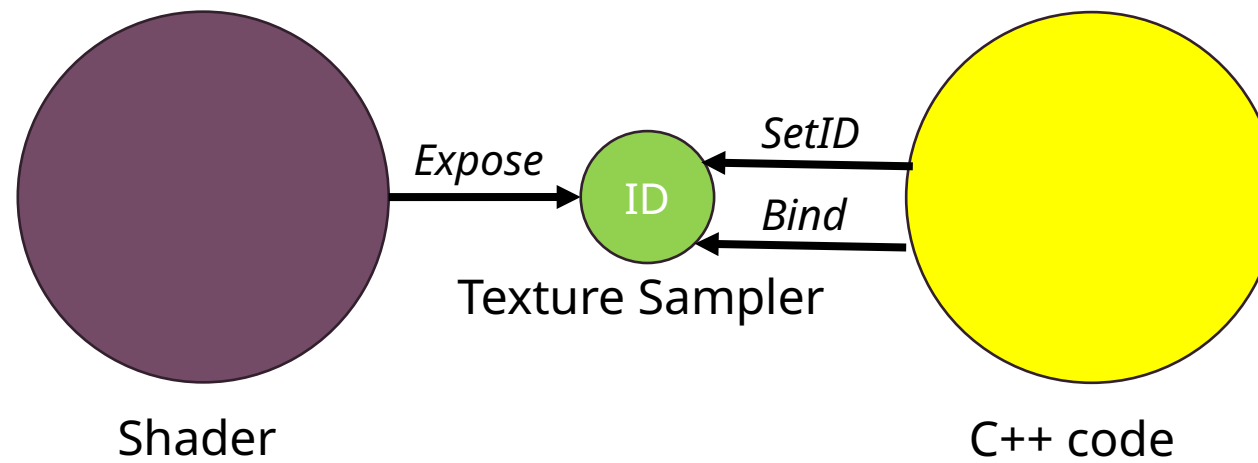


LET'S CODE THIS

TEXTURES



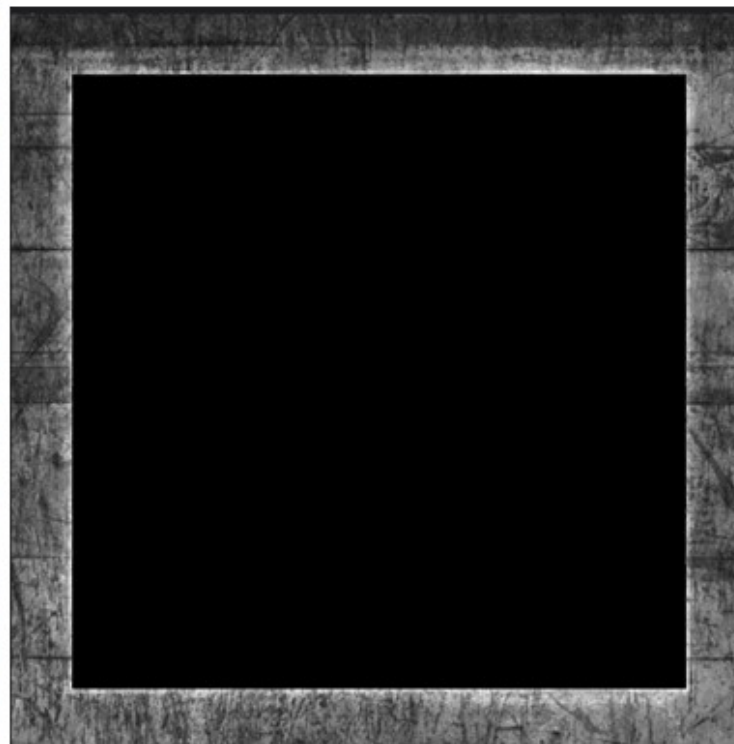
TEXTURES



MATERIALS



Diffuse (does not have
shiny spot)



Specular (has shiny
spot)

MODEL LOADING





THANK YOU

Šimon Potočňák