Programowanie grafiki 3D

Karol Przystalski

Poznajmy się

2017 - doktorat uzyskany w PAN oraz UJ

od 2010 - CTO @ **Codete**

2007 - 2009 - Software Engineer @ **IBM**

Praca naukowa

Multispectral skin patterns analysis using fractal methods, K. Przystalski and M. J.Ogorzalek. Expert Systems with Applications, 2017

https://www.sciencedirect.com/science/article/pii/S0957417417304803

kprzystalski@gmail.com

karol@codete.com

Zakres materiału

1. Podstawy OpenGL
2. Shadery
3. Tekstury
4. Transformacje
5. Oświetlenie i cieniowanie

Zaliczenie

Średnia ocena ze wszystkich projektów.

<u>Warunki:</u>

- Termin na każdy projekt: max. 2 tygodnie po zajęciach
- Każdy projekt zaliczony na minimum 3.0

Projekty

House

Colors

Indices

Uniforms

Pyramid

CameraMovements

PVM

Resizing

Zoom

Texture

Mesh

Phong

WebGL

Vulcan

Środowisko

Narzędzia

CLion

PyCharm/Webstorm

Visual Studio

i oczywiście Vim

Języki

Język podstawowy: **C++**

Można niektóre przykłady zbudować w:

Python

http://pyopengl.sourceforge.net/

JavaScript

https://get.webgl.org/

Problemy

OpenGl nie jest już wspierany przez MacOS:

https://developer.apple.com/library/archive/documentation/GraphicsImaging/Conceptual/OpenGL-MacProgGuide/opengl_intro/opengl_intro.html

XQuartz: https://www.xquartz.org/

M1

Wprowadzenie

Trochę historii

1992 - SGI wypuszcza pierwszą wersję OpenGL

1994 - OpenGL pojawia się na Windowsach NT

1995 - DirectX

1996 - Carmach (Quake) roastuje DirectX

1999 - Nvidia wprowadza na swoich GeForce'ach T&L i przenosi transformacje oraz oświetlenie na poziom GPU

2000 - Microsoft wprowadza shadery, a w 2003 HLSL

2006 - Mamy OpenGL 2.1, który przechodzi pod skrzydła Khronos Group

2009 - OpenGL 3.1

2011 - WebGL

Źródło: https://openglbook.com/chapter-0-preface-what-is-opengl.html

Komponenty

GL - OpenGL

GLES - OpenGL ES https://en.wikipedia.org/wiki/OpenGL_ES#OpenGL_ES_3.2

EGL - łączy GL z VG

GLSL - OpenGL Shading Language

GLUT - OpenGL Utility Toolkit

GLEW - OpenGL Extension Wrangler

Vulkan - nowa nakładka na OpenGL

Komponenty

WebGL - OpenGL dla przeglądarek

OpenVG - Vector Graphic acceleration library

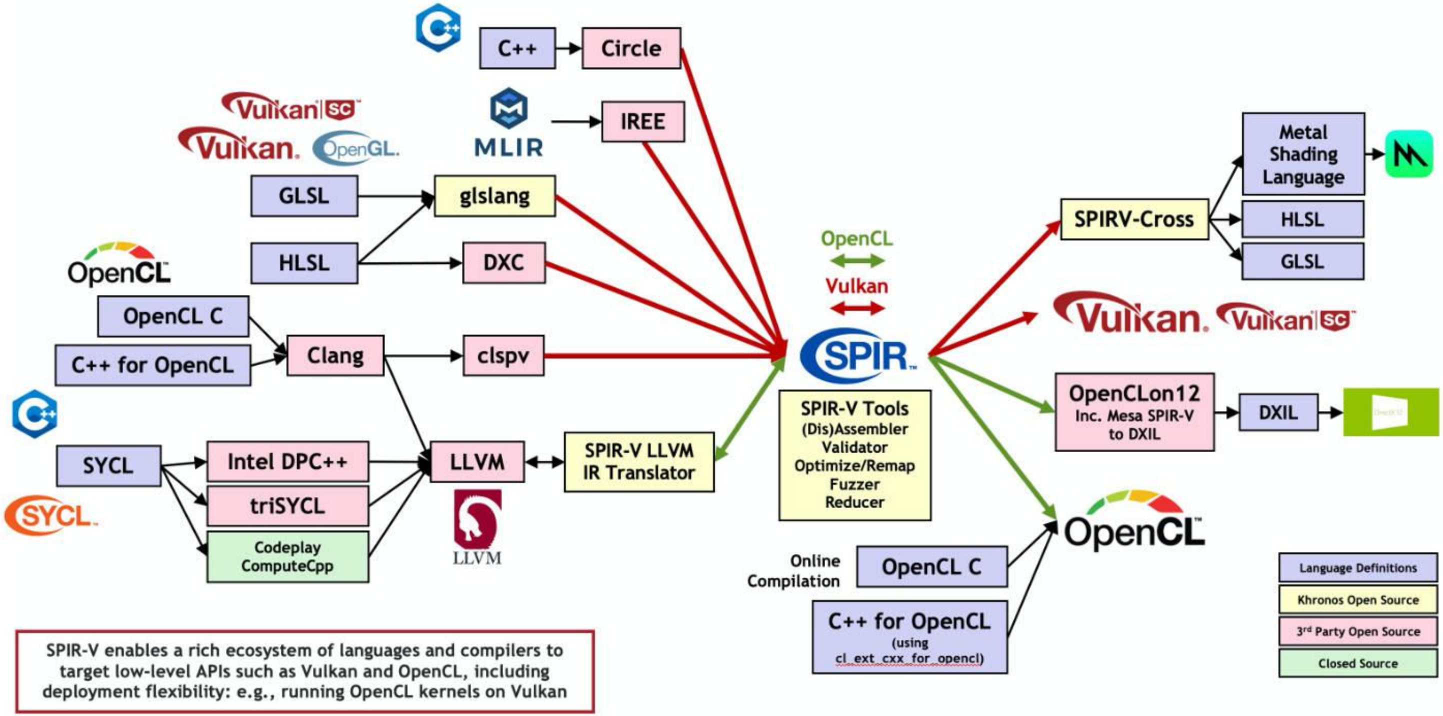
OpenMax - biblioteka do dźwięku, filmów, obrazów

OpenCL - służy do tworzenia aplikacji działających pomiędzy GPU, CPU, FPGA, ...

HLSL - High Level Shader Language

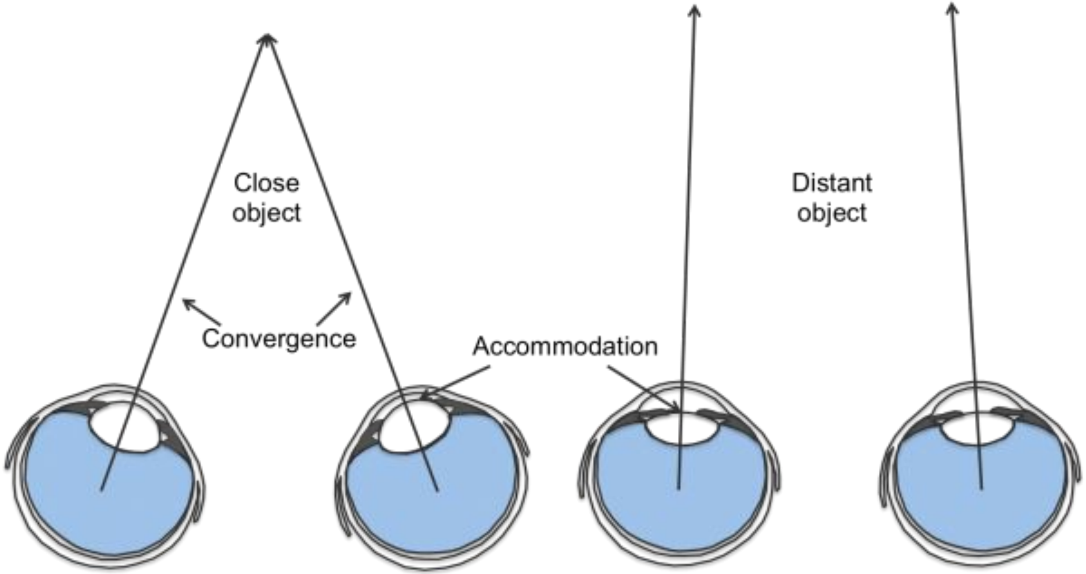
GLM - GL Mathematics

SPIR-V



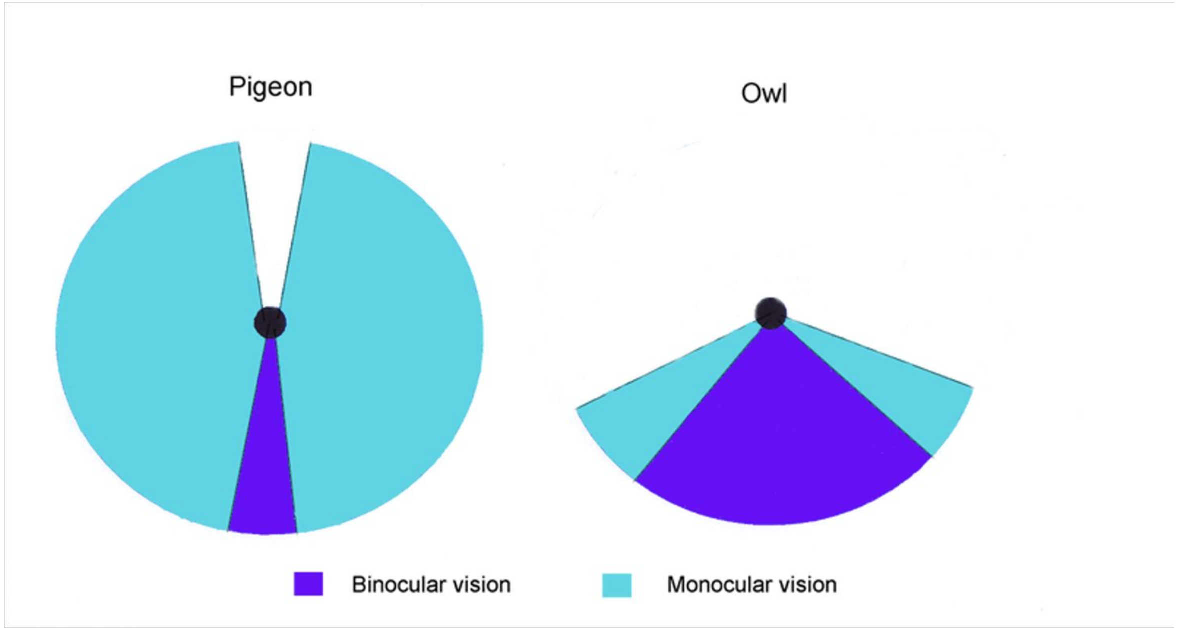
C++
Circle
MLIR
IREE
GLSL
glslang
HLSL
DXC
OpenCL C
C++ for OpenCL
Clang
clspv
SYCL
Intel DPC++
triSYCL
Codeplay ComputeCpp
LLVM
SPIR-V LLVM IR Translator
OpenCL
Vulkan
SPIR
SPIR-V Tools
(Dis)Assembler
Validator
Optimize/Remap
Fuzzer
Reducer
SPIRV-Cross
Metal Shading Language
HLSL
GLSL
OpenCLon12
Inc. Mesa SPIR-V to DXIL
DXIL
Online Compilation
OpenCL C
C++ for OpenCL (using cl_ext_cxx_for_opencl)
Language Definitions
Khronos Open Source
3rd Party Open Source
Closed Source
SPIR-V enables a rich ecosystem of languages and compilers to target low-level APIs such as Vulkan and OpenCL, including deployment flexibility: e.g., running OpenCL kernels on Vulkan

Percepcja



Źródło: https://www.binocularsguru.com/binocular-cues-vs-monocular-cues-difference-and-uses/

Percepcja



Percepcja jednym okiem

1. *Absolute Size,* not knowing the size of an object is problematic for us, in such cases, the smaller object is considered at a greater distance than larger objects at the same location.

2. *Motion Parallax*, it describes the way stationary objects appear to be moving at different speeds against a background when we observe it moving.

3. *Familiar Size,* familiarity with the size of objects helps us determine how far away they are from us.

4. *Texture Gradient,* the amount of detail we can see easily on an object when it is close to us; when far we can't see the detail.

5. *Reach Trajectory,* it shows direction bias during monocular viewing, especially in the approach phase. This bias is consistent with the presence of esophoria in monocular viewing. Esophoria is present when occluded eye deviates medially and exophoria is present when occluded eye shifts temporally.

6. *Relative Size*, size does matter; by knowing how big two objects are in relation to each other, how far away they are from each other and we can be figured.

7. *Linear Perspective,* parallel lines seem to converge at a distance; the farther they are, the closer they seem to us.

Percepcja jednym okiem

8. *Natural Effects,* like heat haze, water vapor, dust, sand, and fog, can affect our vision, especially at longer distances.

9. *Interposition,* when an object partially overlaps or obscures another object; it helps us to put the distances of objects in order of the nearest one first.

10. *Aerial Perspective, o*bjects at larger distances from us are affected by natural scattering of light and form less of a contrast with their background; making it harder to gauge a distance between the two and us.

11. *Accommodation* refers to the amount of work our eye muscles like ciliary muscles have to do to focus on an object.

12. *Shading and Lighting,* the nearer an object is to light, its surface appears to be brighter. In a group of objects, darker objects tend to appear farther away than the brighter objects.

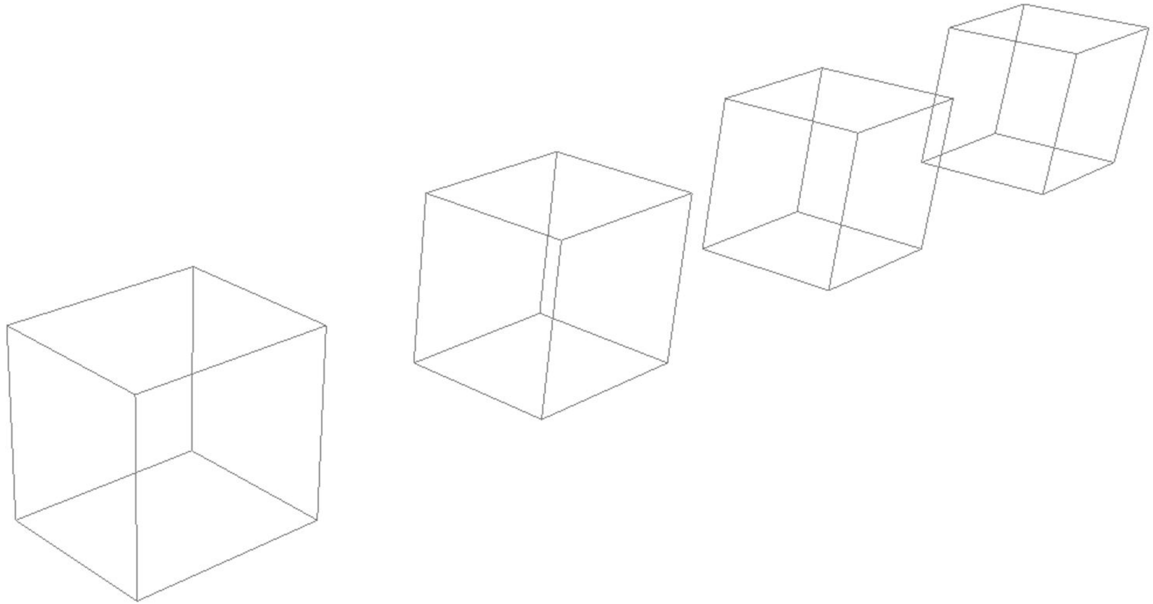
13. *Depth from Motion*, as an object moves closer, its size increases in the eyes of an observer; this helps determine the pace of its movement and its distance from us.

Percepcja parą oczu

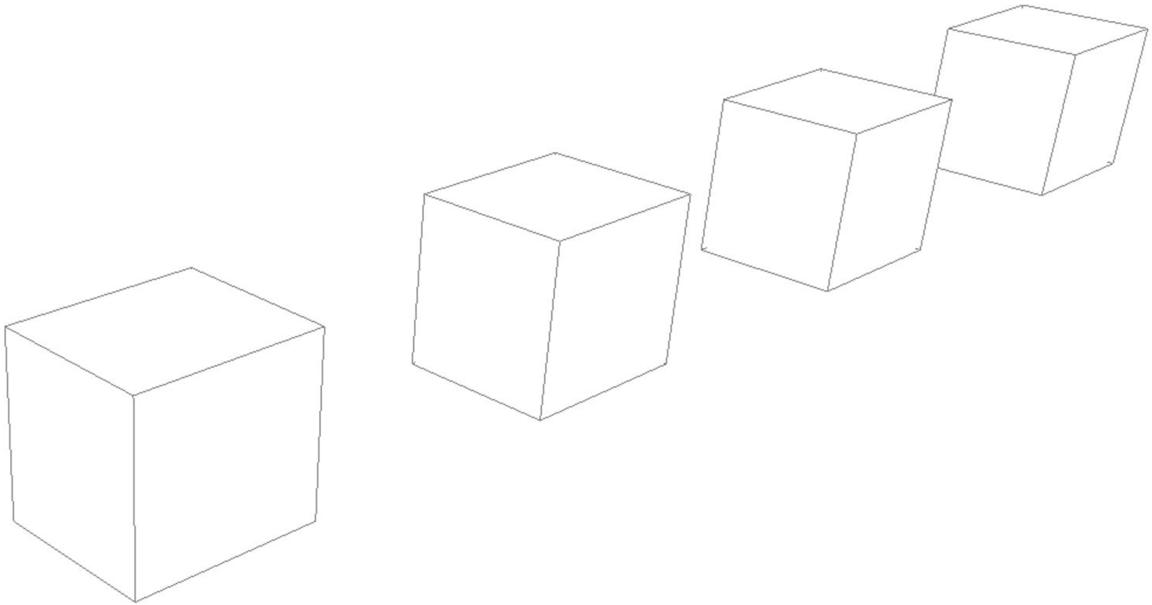
1. *Retinal Disparity* also called binocular parallax, that refers to the fact that each of our eyes sees the world from a slightly different angle, which is triangulated by the brain to figure out the correct distance

2. *Binocular Convergence* refers to the amount of rotation our eyes have to do in order to focus on an object. It enables us to determine how near or far things are away from us. A proprioceptive sense, it is the amount of inward rotation our eyes have to do in order to focus on an object.

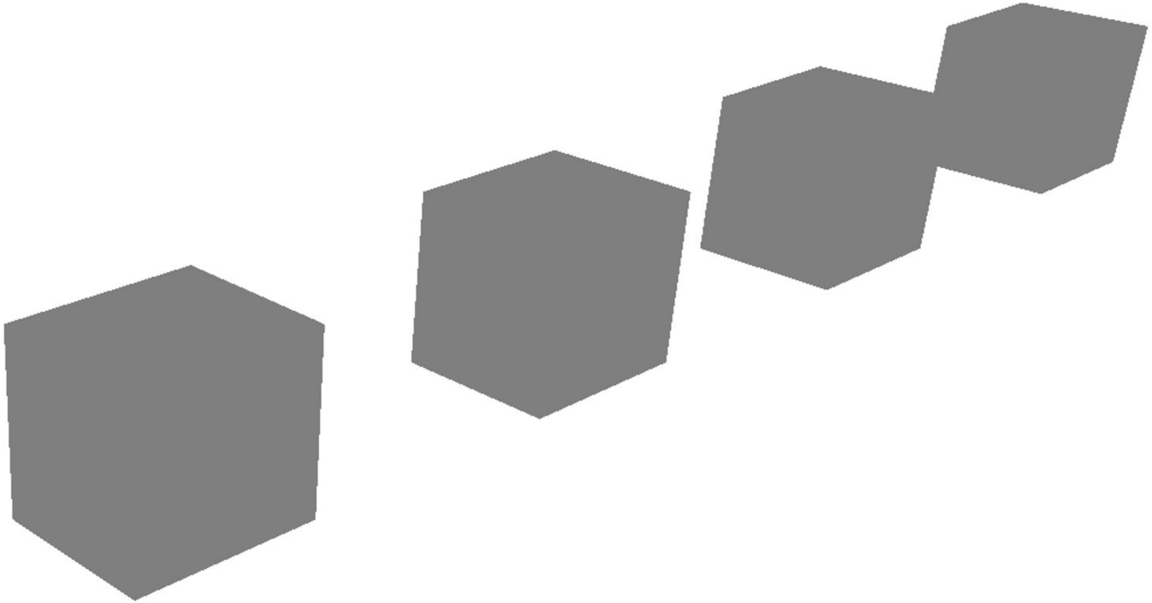
Perspektywa



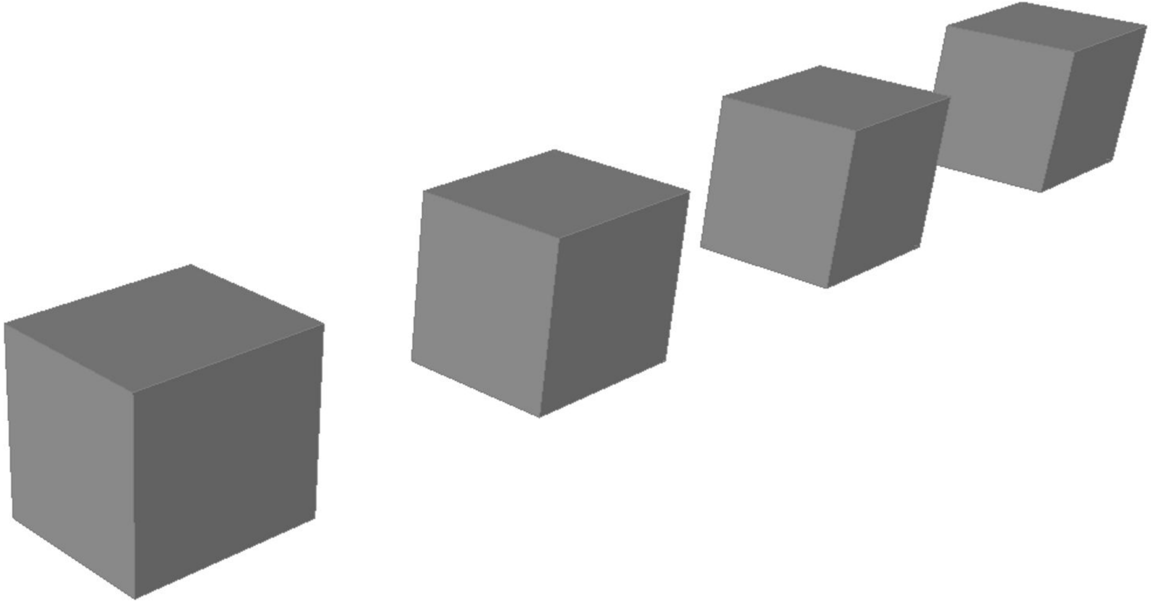
Okluzja



Okluzja



Światło i cienie



Paralax

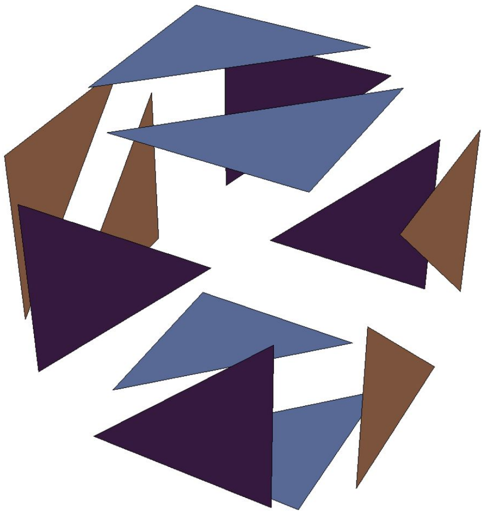
https://pixelcog.github.io/parallax.js/

Proces

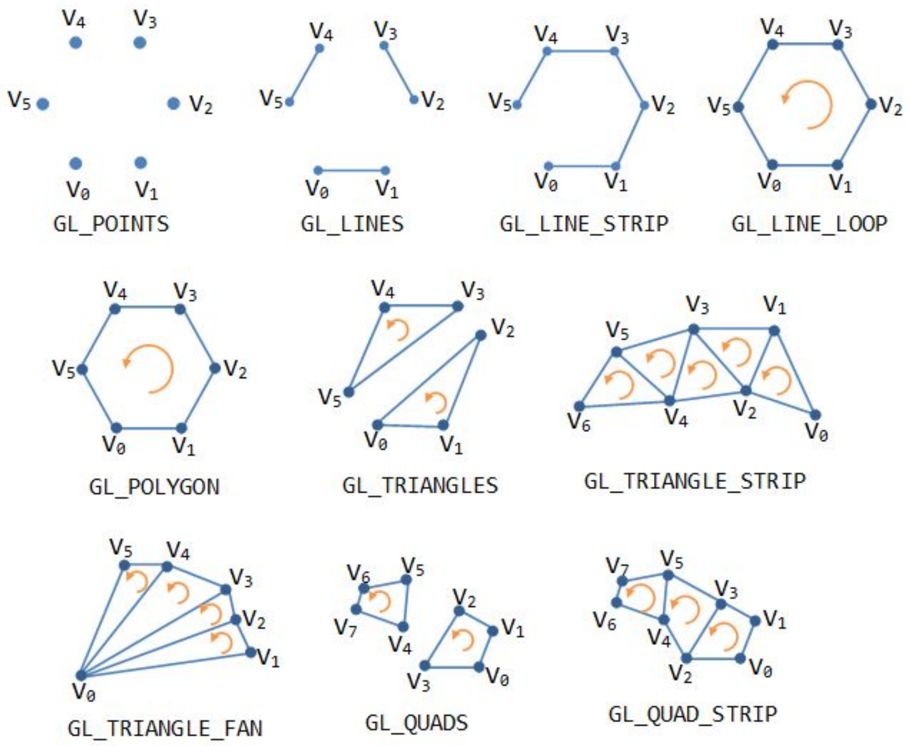
Proces

Proces

Proces



Prymitywy



OpenGL Primitives

https://www.khronos.org/opengl/wiki/Primitive

Shadery kiedyś

- glBegin
  - glColor, glNormal, glVertex, glTexCoord, etc.
- glEnd

Shadery dziś

VBO (Vertex Buffer Objects) is a collection of data representing your object ○ Positions, normals, colors, texture coordinates, etc.

VAO (Vertex Array Objects) - you can set vertex attributes for the VAO to point to the different data fields of your struct and the proper uniforms of the shader

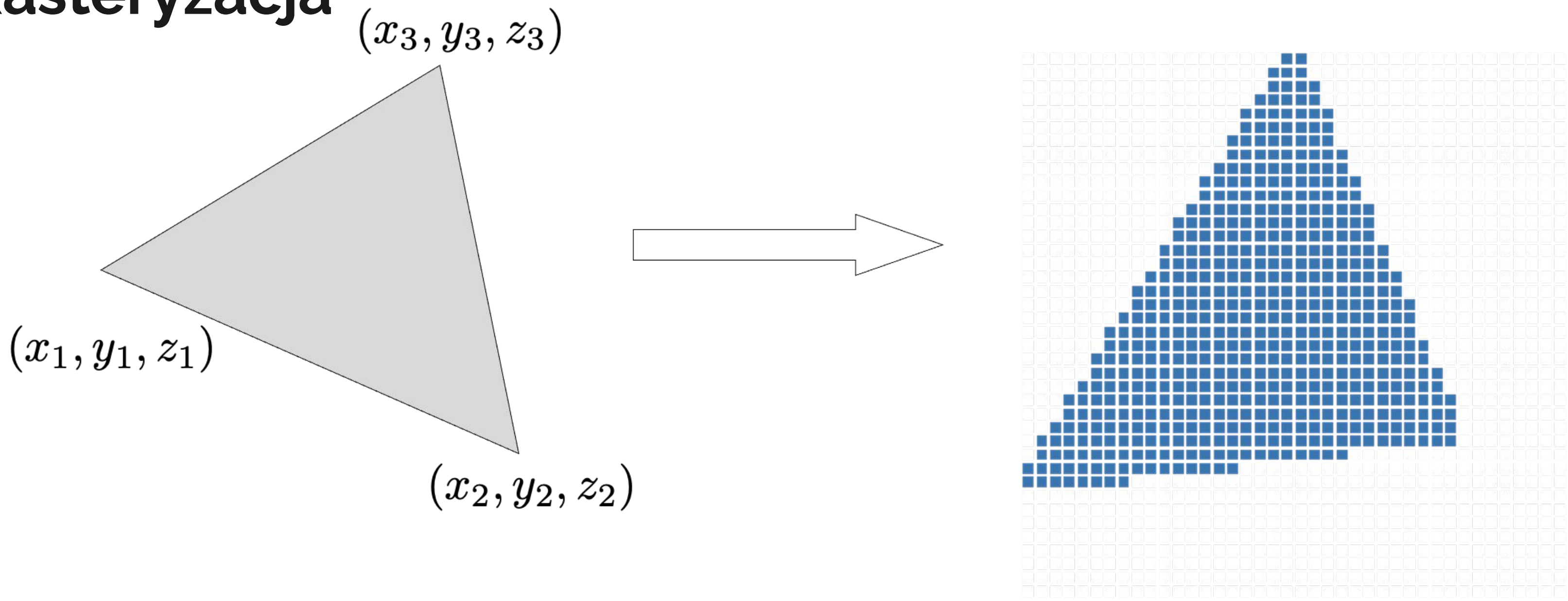
VBO

1. glGenBuffers
2. glBindBuffer
3. glBufferData

VBO

```
void CreateVBO(void) {
	GLfloat Vertices[] = { -0.8f, -0.8f, 0.0f, 1.0f, 0.0f, 0.8f, 0.0f, 1.0f, 0.8f, -0.8f, 0.0f, 1.0f };
	GLfloat Colors[] = { 1.0f, 0.0f, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f };

	glGenVertexArrays(1, &VaoId); glBindVertexArray(VaoId);
	glGenBuffers(1, &VboId);
	glBindBuffer(GL_ARRAY_BUFFER, VboId);
	glBufferData(GL_ARRAY_BUFFER, sizeof(Vertices), Vertices, GL_STATIC_DRAW);
	glVertexAttribPointer(0, 4, GL_FLOAT, GL_FALSE, 0, 0); glEnableVertexAttribArray(0);

	glGenBuffers(1, &ColorBufferId); glBindBuffer(GL_ARRAY_BUFFER, ColorBufferId);
	glBufferData(GL_ARRAY_BUFFER, sizeof(Colors), Colors, GL_STATIC_DRAW);
	glVertexAttribPointer(1, 4, GL_FLOAT, GL_FALSE, 0, 0); glEnableVertexAttribArray(1);
}
```

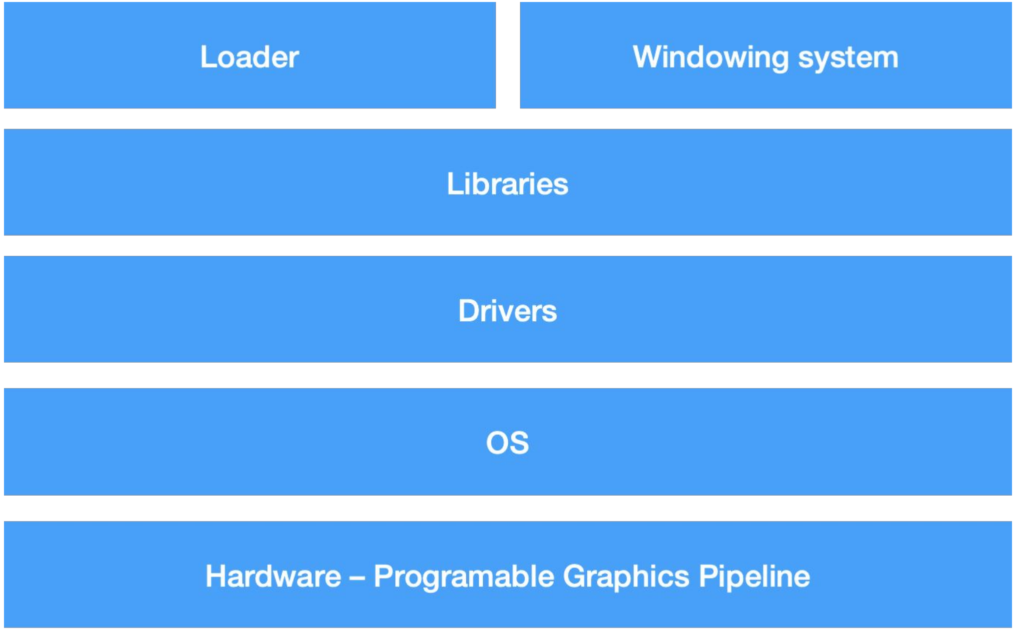
Rasteryzacja



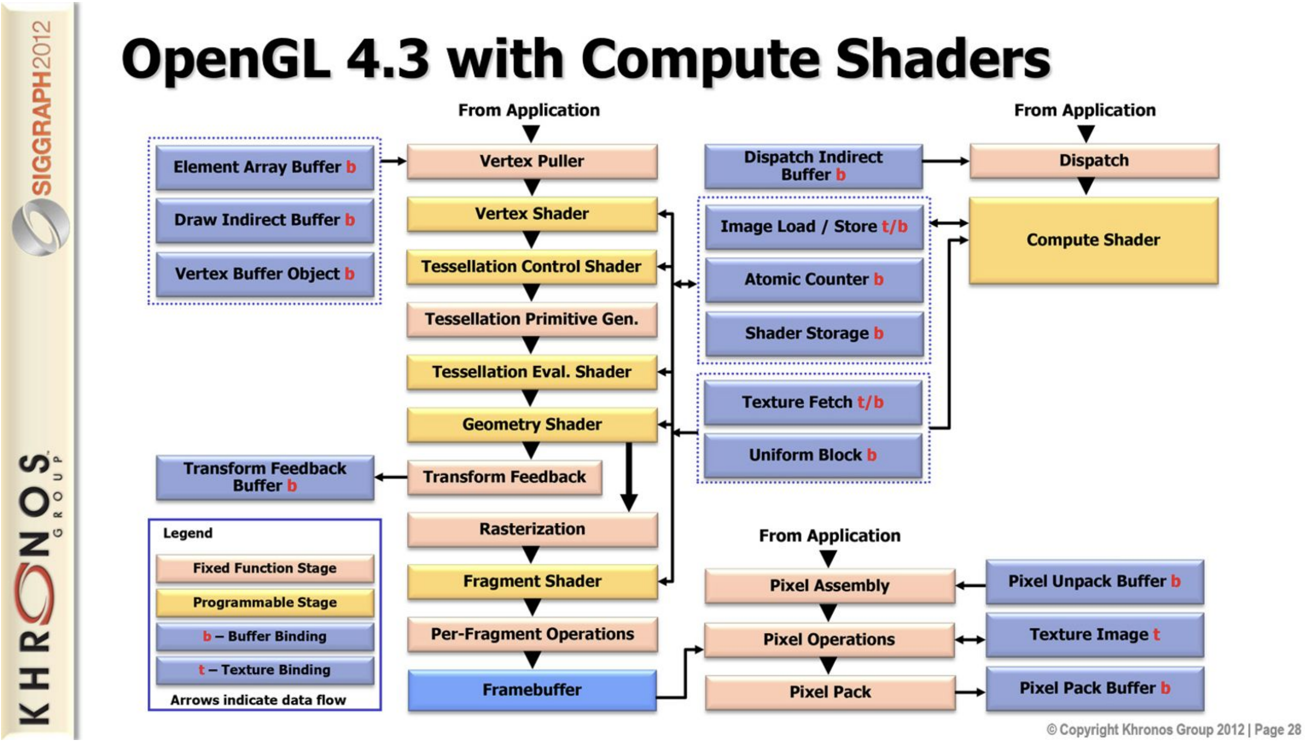
Szczegóły

1. Obiekty 3D trzeba przenieść na obraz 2D (kordynaty) - projekcja perspektywy
2. Trzeba ustalić piksele obiektów - rasteryzacja
3. Widoczne piksele powinny zostać pokazane (z-buffer)
4. Każdy piksel należy pokolorować (cieniowanie/teksturowanie)

Proces

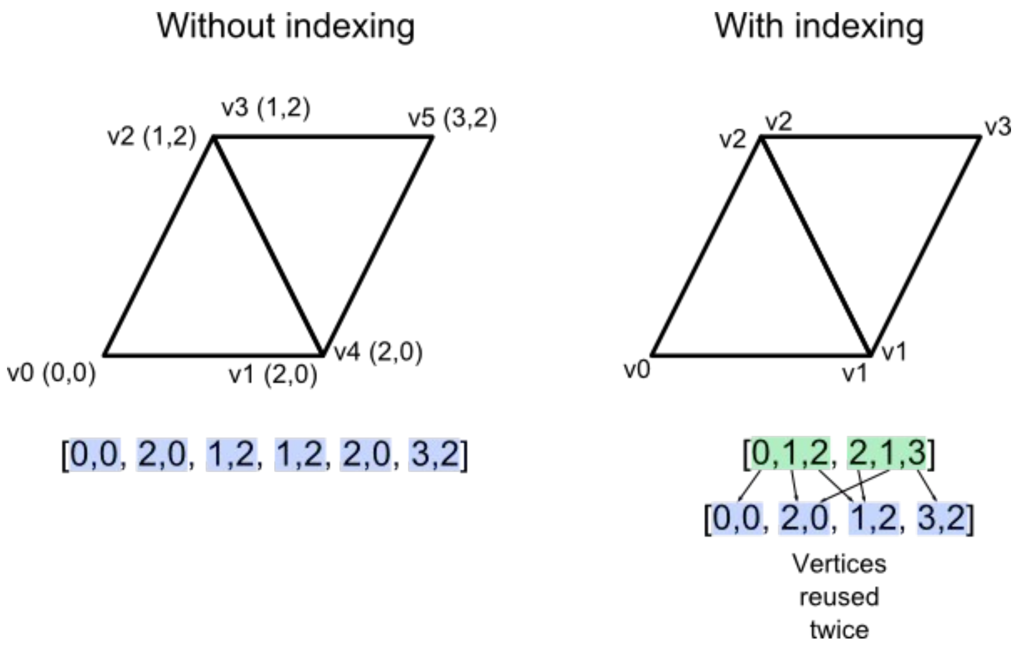

Loader
Windowing system
Libraries
Drivers
OS
Hardware – Programable Graphics Pipeline

Proces


SIGGRAPH2012
KHRONOS GROUP
OpenGL 4.3 with Compute Shaders
From Application
Vertex Puller
Vertex Shader
Tessellation Control Shader
Tessellation Primitive Gen.
Tessellation Eval. Shader
Geometry Shader
Transform Feedback
Rasterization
Fragment Shader
Per-Fragment Operations
Framebuffer
Element Array Buffer b
Draw Indirect Buffer b
Vertex Buffer Object b
Transform Feedback Buffer b
Legend
Fixed Function Stage
Programmable Stage
b – Buffer Binding
t – Texture Binding
Arrows indicate data flow
Dispatch Indirect Buffer b
Image Load / Store t/b
Atomic Counter b
Shader Storage b
Texture Fetch t/b
Uniform Block b
From Application
Dispatch
Compute Shader
From Application
Pixel Assembly
Pixel Operations
Pixel Pack
Pixel Unpack Buffer b
Texture Image t
Pixel Pack Buffer b
© Copyright Khronos Group 2012 | Page 28

Proces

Indeksowanie



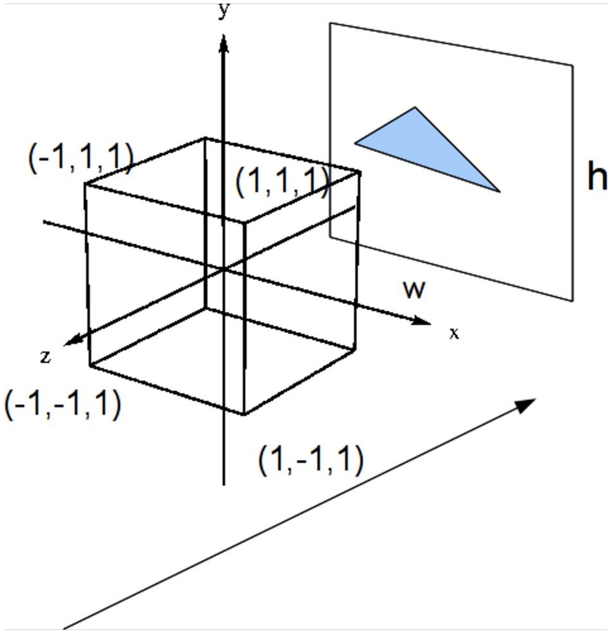
http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-9-vbo-indexing/

Uniforms

A **uniform** is a global Shader variable declared with the "uniform" storage qualifier. These act as parameters that the user of a shader program can pass to that program. Their values are stored in a program object.

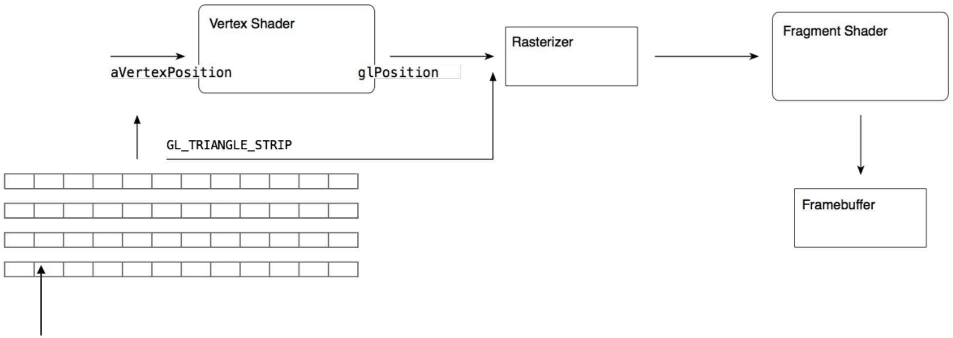
https://www.khronos.org/opengl/wiki/Uniform_(GLSL)

Projekcja



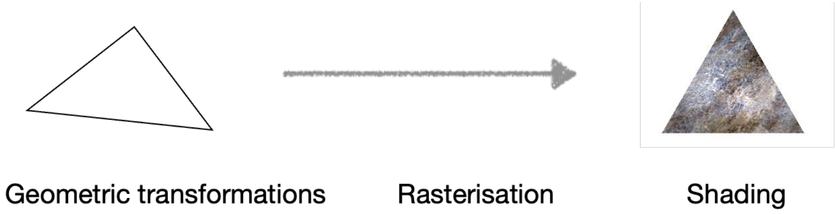
Bufory i proces

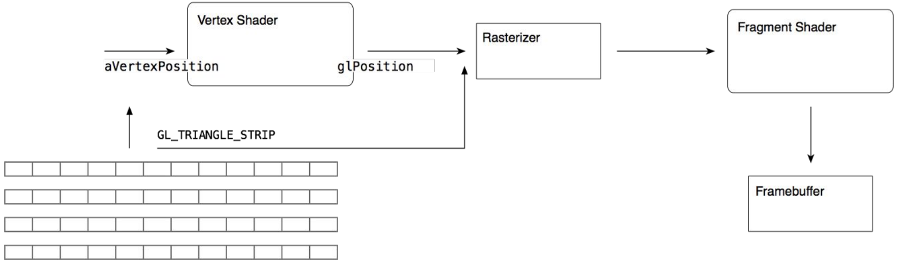
Geometric transformations Rasterisation Shading



Buffers

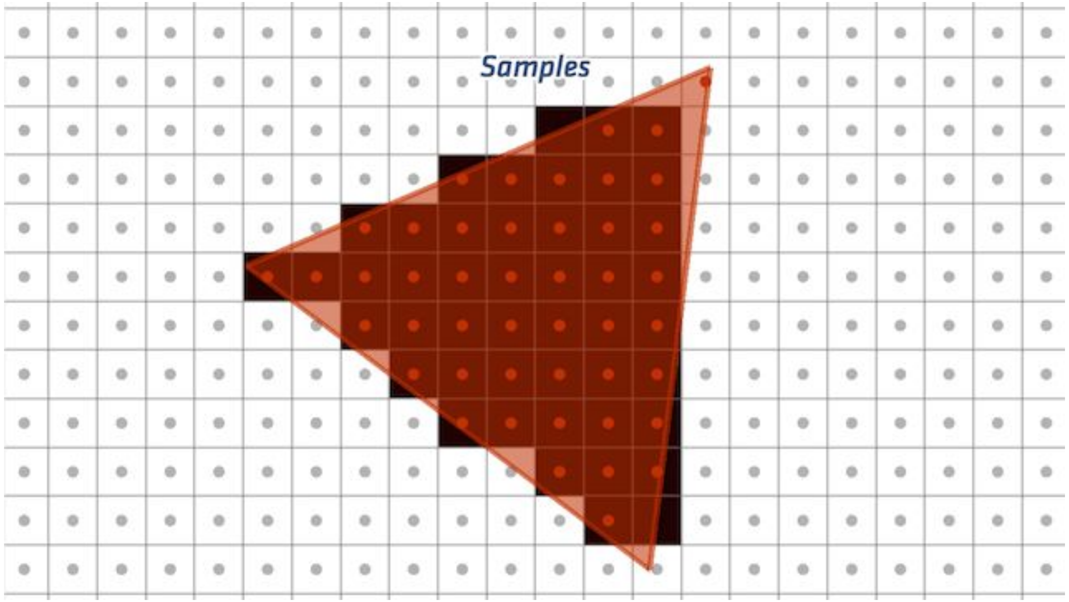
Vertex shader



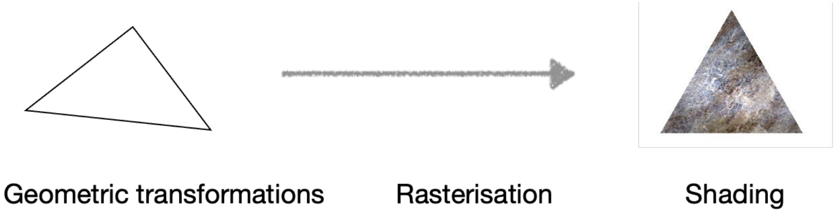


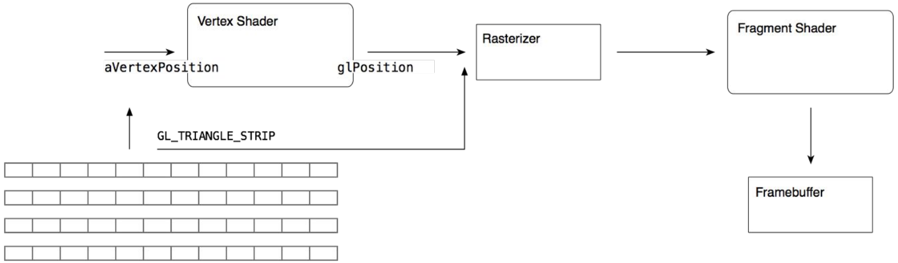

```
#version 410

layout(location=0) in vec4 aVertexPosition;

void main() {
    gl_Position = aVertexPosition;
}
```

Fragment shader



Fragment shader






```
#version 410

out vec4 vFragColor;

void main() {
   vFragColor = vec4(1.f, 0.f, 0.f, 1.f);
}
```

Bibliografia

[1] John Kessenich, Graham Sellers and Dave Shreiner, **OpenGL® Programming Guide: The Official Guide to Learning OpenGL®, Version 4.5 with SPIR-V, Ninth Edition**, 2016

[2] Graham Sellers, Richard S. Wright Jr. and Nicholas Haemel, **OpenGL SuperBible: Comprehensive Tutorial and Reference, Seventh Edition**, 2015

[3] David Wolff, **OpenGL 4 Shading Language Cookbook - Third Edition**, 2018

[4] Frahaan Hussain, **Learn OpenGL**, 2018

[5] Farhad Ghayour , Diego Cantor, **Real-Time 3D Graphics with WebGL 2 - Second Edition**, 2018