Shadery obliczeniowe 1

Modelowanie Komputerowe

Po co?

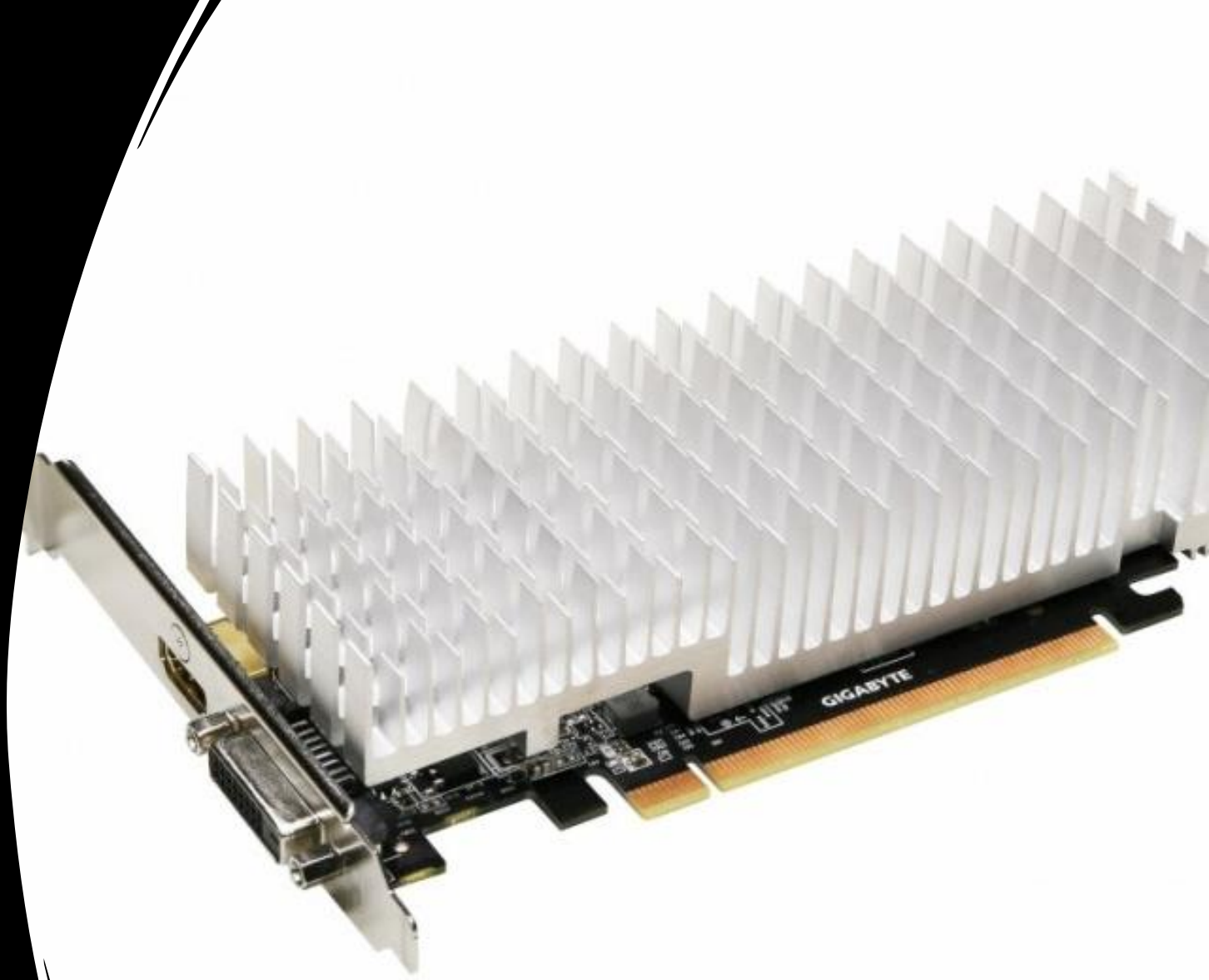
- Symulacje 4K w czasie rzeczywistym
- Ścisłe związanie obliczeń z wizualizacją
- Technologia wykorzystywana w grach
- Efekty specjalne
- Wykorzystanie potencjału karty graficznej

Jak pracować z shaderami?

- Sprzęt np. GTX 1030 --->

Oprogramowanie:

- OpenGL (Vulkan, etc.)
- **OpenFrameworks**



Open Frameworks

- Biblioteka multimediów "wysokiego poziomu"
- Język C++
- Program obiektowy
- Zbudowana na OpenGL
- Posiada wsparcie dla OpenGL>4.3 (compute shaders)
- Video: https://openframeworks.cc/

Minimalny kod w OpenFrameworks

Podstawowy projekt to 3 pliki:

- main.cpp
- ofApp.cpp
- ofApp.h

1. main.cpp

```
1  #include "ofMain.h"
2  #include "ofApp.h"
3
4  //========================================
5  int main( ){
6
7      ofSetupOpenGL(1024,768, OF_WINDOW);
8
9      // this kicks off the running of my app
10     // can be OF_WINDOW or OF_FULLSCREEN
11     // pass in width and height too:
12     ofRunApp( new ofApp());
13
14 }
15
```

2. ofApp.h

```
1 #pragma once
2
3 #include "ofMain.h"
4
5 class ofApp : public ofBaseApp{
6     public:
7         void setup();
8         void update();
9         void draw();
10
11        void keyPressed(int key);
12 };
```

3. ofApp.cpp

```cpp
#include "ofApp.h"

//--------------------------------------------------------------
void ofApp::setup(){

}

//--------------------------------------------------------------
void ofApp::update(){

}

//--------------------------------------------------------------
void ofApp::draw(){

}

//--------------------------------------------------------------
void ofApp::keyPressed(int key){

}

```

Program startowy OF

main.cpp

```cpp
#include "ofMain.h"
#include "ofApp.h"

//========================================================================
int main( ){

    ofSetupOpenGL(1024,768, OF_WINDOW);

    // this kicks off the running of my app
    // can be OF_WINDOW or OF_FULLSCREEN
    // pass in width and height too:
    ofRunApp( new ofApp());

}
```

ofApp.cpp

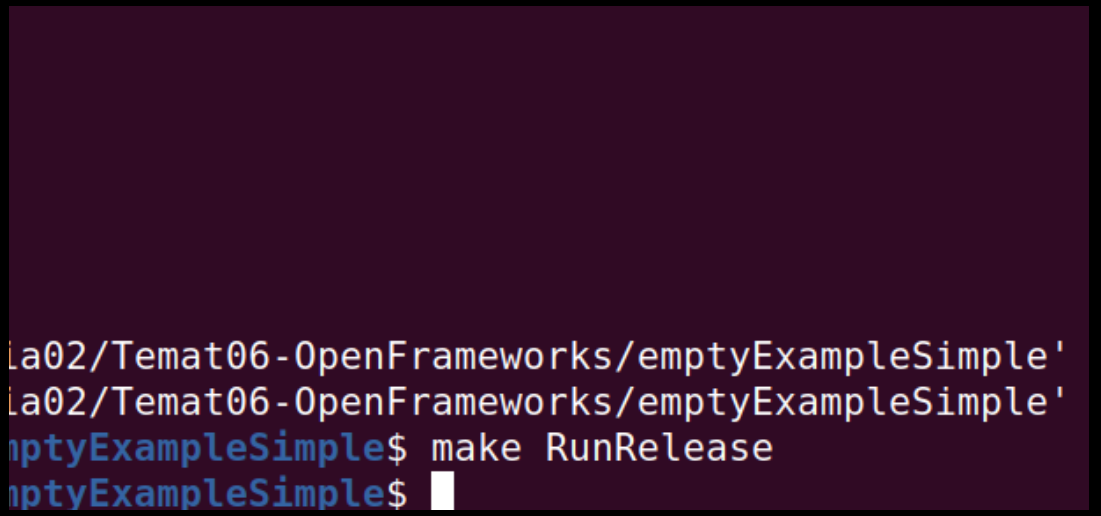
```cpp
#pragma once

#include "ofMain.h"

class ofApp : public ofBaseApp{
    public:
        void setup();
        void update();
        void draw();

        void keyPressed(int key);
};

```

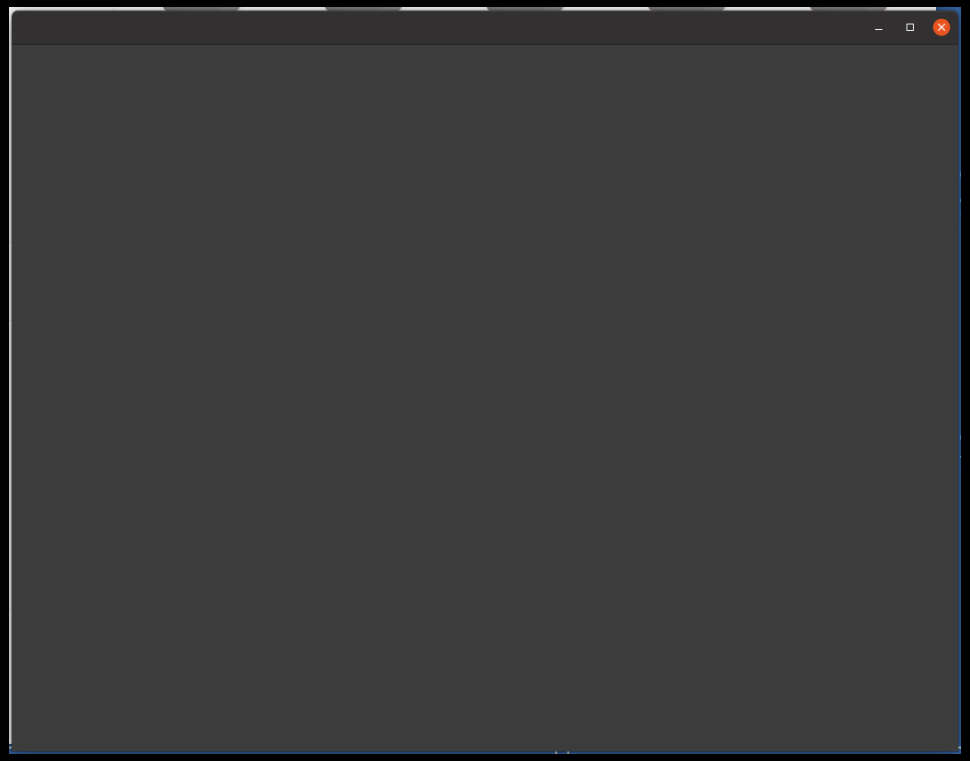
ofApp.h

```cpp
#include "ofApp.h"

//--------------------------------------------------------------
void ofApp::setup(){

}

//--------------------------------------------------------------
void ofApp::update(){

}

//--------------------------------------------------------------
void ofApp::draw(){

}

//--------------------------------------------------------------
void ofApp::keyPressed(int key){

}

```

Kompilacja

- Linux:
  - make
  - make RunRelease





Ustawienia kompilacji

- Trzeba ustawić ścieżkę OF_ROOT do katalogu z biblioteką
- U mnie:
  - OF_ROOT=~/OF // plik Makefile i config.make

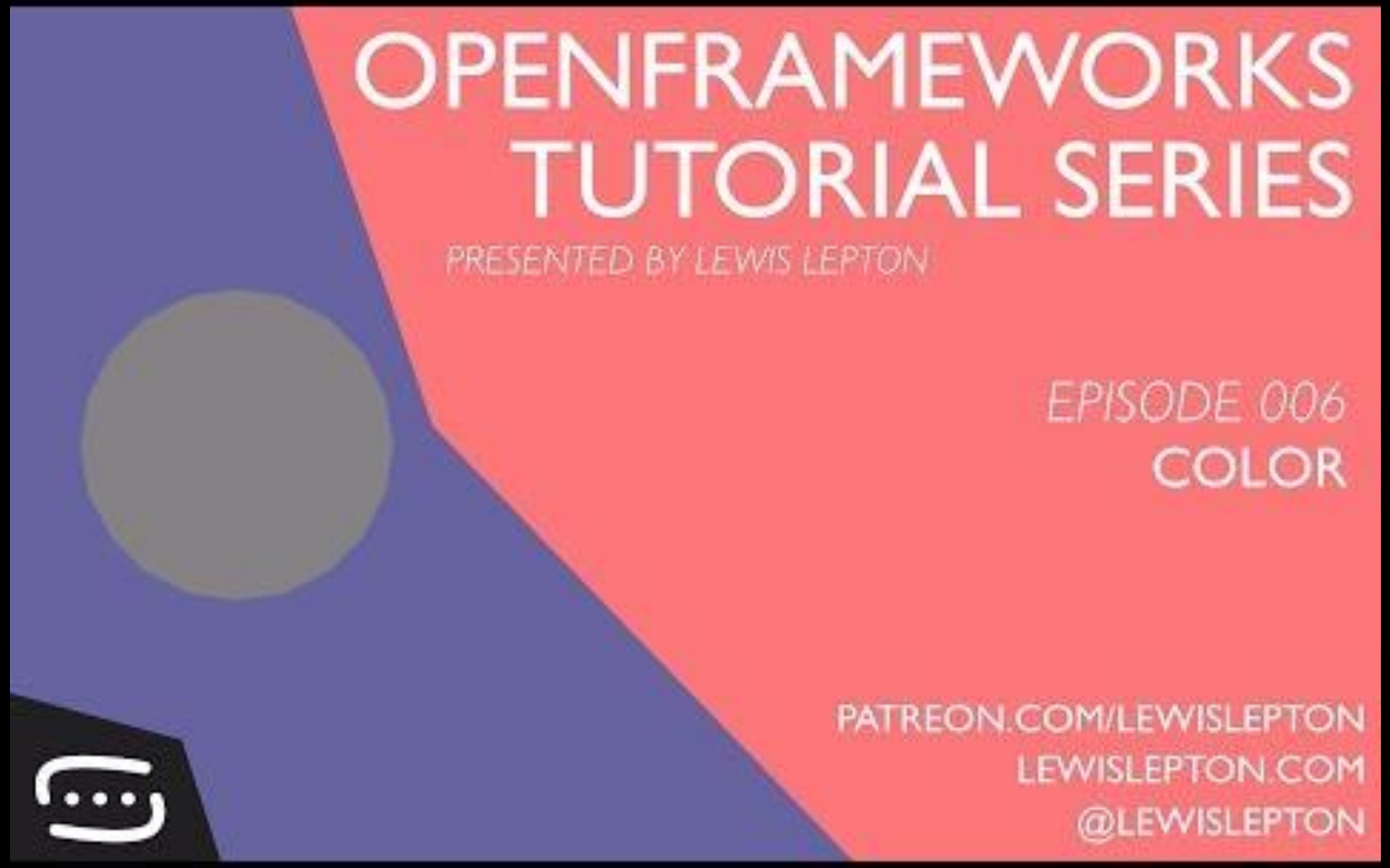
- Inne miejsce?

  /usr/local/of_v0.11.2_linux64gcc6_release/

Windows?

- Visual Studio (od 2017-2022)
- Po instalacji gotowe

Tutorial YT
Lewis
Lepton



Materiały do nauki

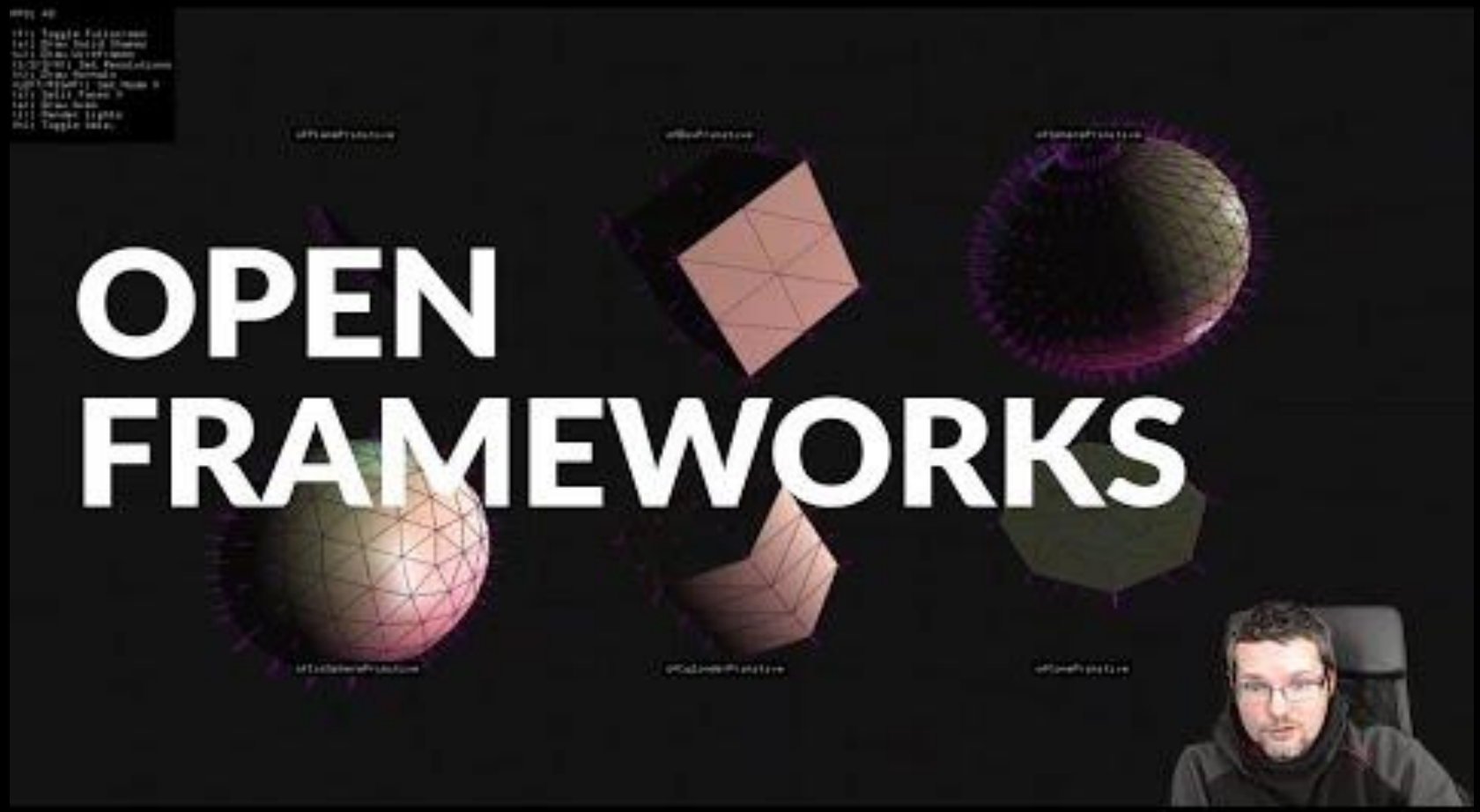
- Strona OF: https://openframeworks.cc/learning/
- OFBook: https://openframeworks.cc/learning/#ofBook

- Lewis Lepton (tutoriale): https://www.youtube.com/watch?v=dwt2NAd1ZYY&list=PL4neAtv21WOlqpDzGqbGM_WN2hc5ZaVv7&index=1

- Github https://github.com/lewislepton/openFrameworksTutorialSeries

- Dokumentacja: https://openframeworks.cc/documentation/

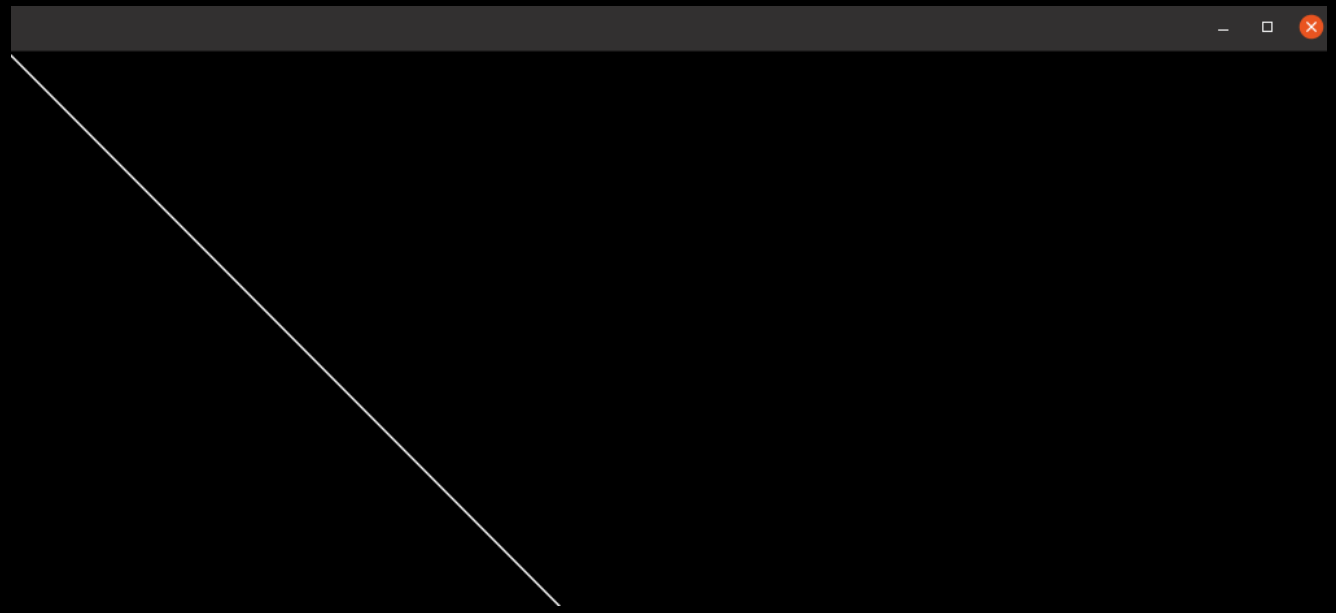
Wykład online



- https://youtu.be/Q_eIhnBsFH8

Krótko

- Kolor tła: ofBackground(0,0,0);
- Kolor rysowania: ofSetColor(255,255,255);
- Rysowanie linii: ofDrawLine(0, 0, 100, 100);



- https://openframeworks.cc/documentation/graphics/ofGraphics/#show_ofDrawLine

Compute shaders
(shadery obliczeniowe)

Struktura programu z shaderami

- ofApp.cpp
- ofApp.h
- main.cpp
- computeshader.cs

Inicjalizowanie main.cpp (OpenGL > 4.4)

```
1  #include "ofMain.h"
2  #include "ofApp.h"
3
4  //========================================================================
5  int main( )
6  {
7      ofGLWindowSettings settings;
8      settings.setGLVersion(4,5);
9     #ifdef __linux__
10    settings.setSize(W,H);
11    #elif _WIN32
12    settings.width=W;
13    settings.height=H;
14    #else
15    #endif
16     ofSetDataPathRoot("../");
17     ofCreateWindow(settings);
18     ofRunApp(new ofApp());
19 }
20
```

Shader obliczeniowy (ang. Compute shader)

- program, który uruchamia się na procesorze GPU (zamiast CPU)
- Język programowania: GLSL ze składnią jak C++
- https://www.khronos.org/opengl/wiki/Compute_Shader
- https://www.khronos.org/opengl/wiki/Core_Language_(GLSL)

Shader jest obiektem w OpenFrameworks

```
ofShader shader;
```

Obiekt ofShader

```
4   #define W 1280
5   #define H 720
6
7   class ofApp : public ofBaseApp
8   {
9       public:
10          void setup();
11          void update();
12          void draw();
13
14          ofShader shader;
15  };
```

ofApp.h

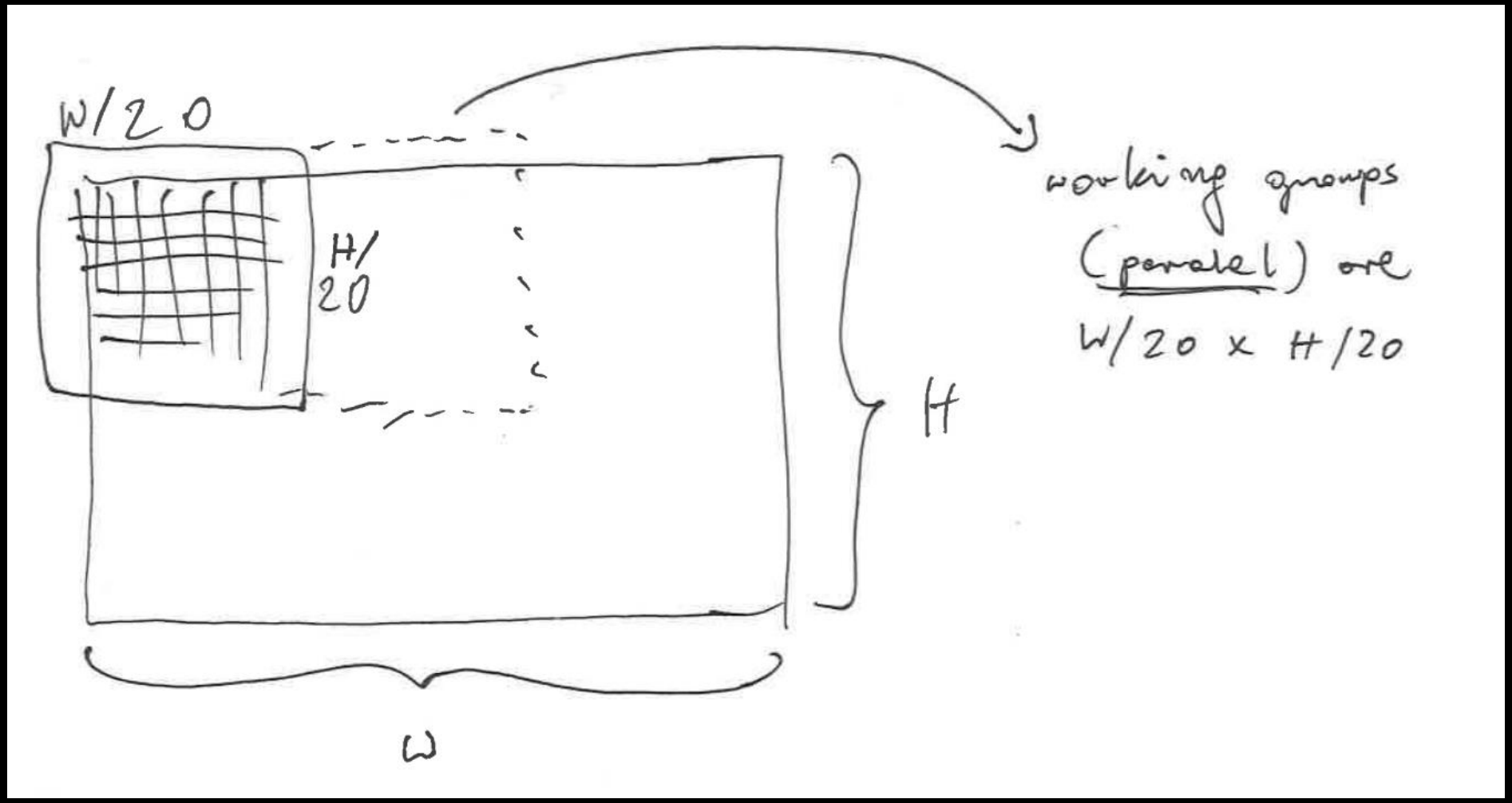
Kompilowanie Shadera

- Napisany program musi być przesłany do GPU i skompilowany

```
4   //--------------------------------------------------------------
5   void ofApp::setup()
6   {
7       shader.setupShaderFromFile(GL_COMPUTE_SHADER,"computeShader.cs");
8       shader.linkProgram();
9
```

ofApp.cpp

Uruchamianie Shadera

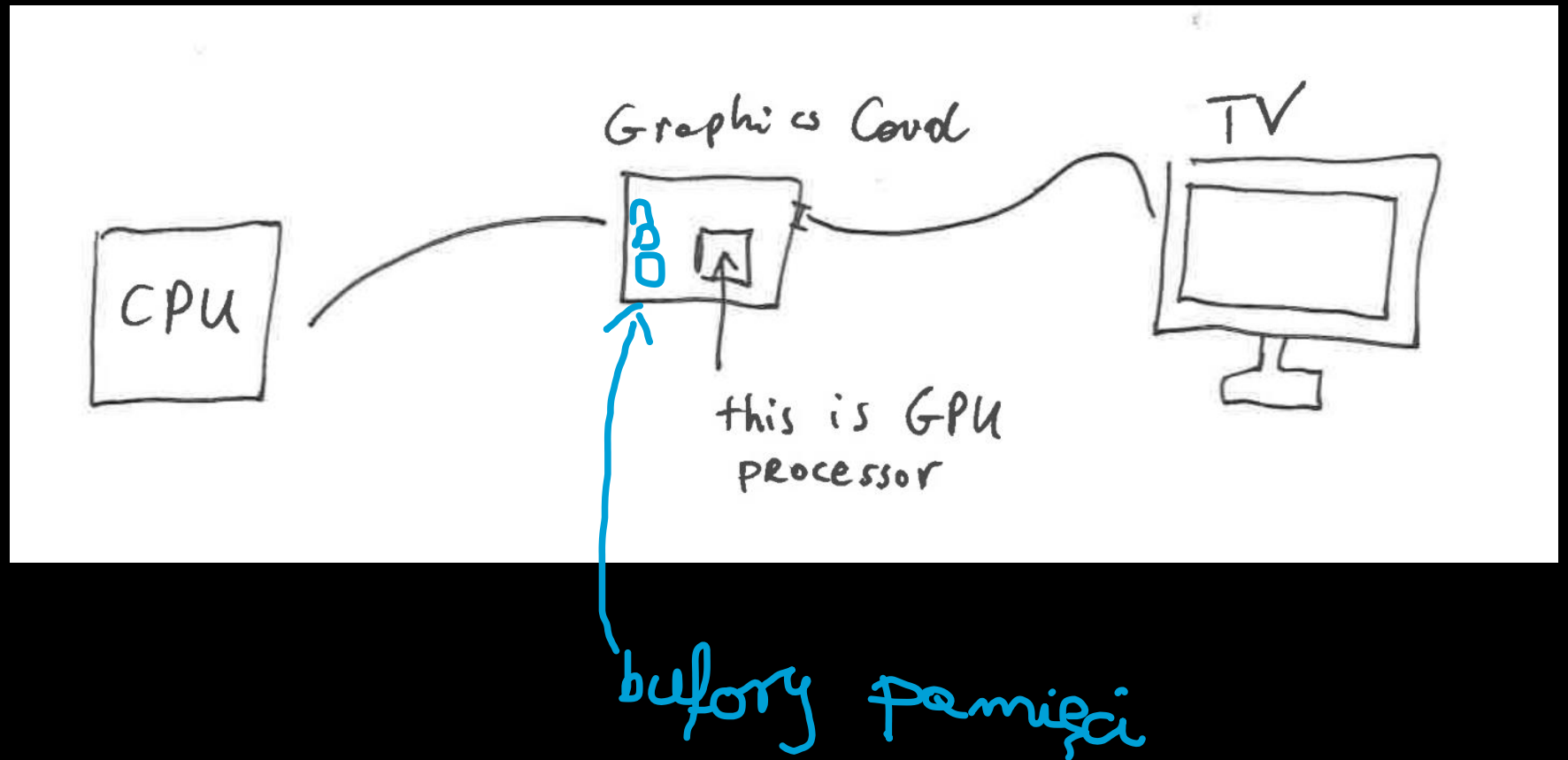


Uruchamianie Shadera

```
13  //--------------------------------------------------------------
14 void ofApp::update()
15  {
16      // run shader
17      shader.begin();
18          shader.dispatchCompute(W/20, H/20, 1);
19      shader.end();
20  }
```

ofApp.cpp

Shader pracuje na buforach ofBufferObject



Shader pracuje na buforach ofBufferObject

```
 7  class ofApp : public ofBaseApp
 8  {
 9      public:
10          void setup();
11          void update();
12          void draw();
13
14          ofBufferObject A1;
15          ofShader shader;
16  };
```

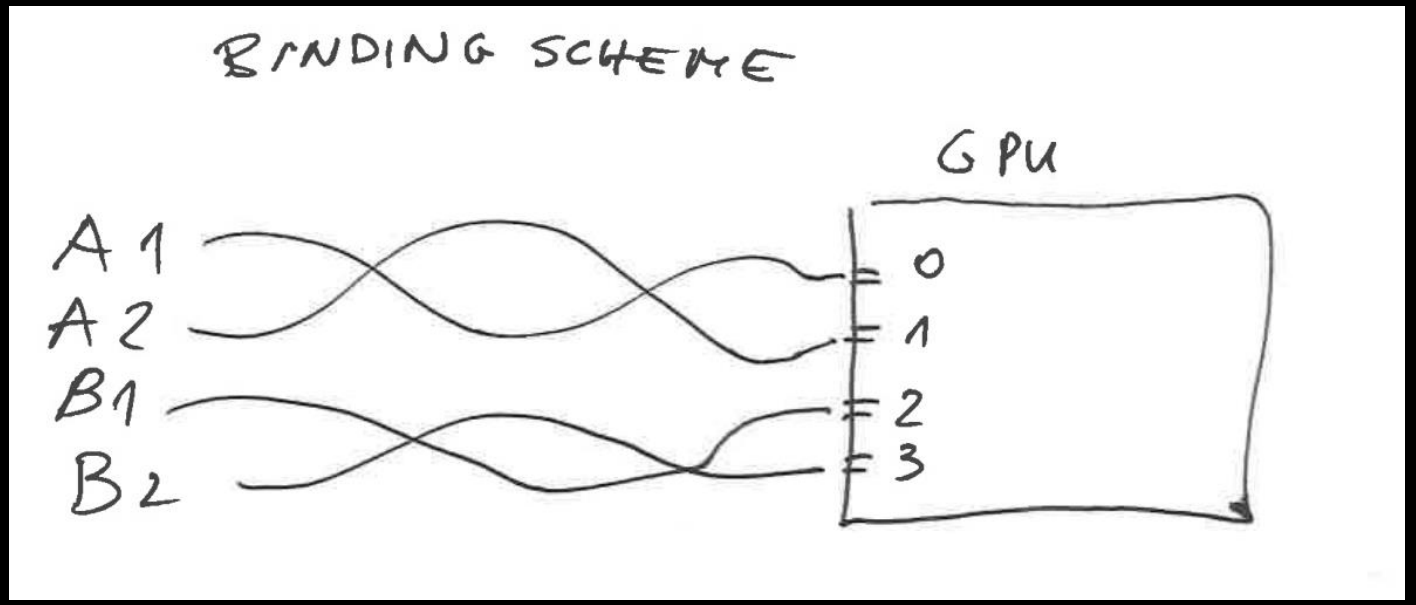
ofApp.cpp

Alokacja pamięcia dla ofBufferObject

```
3   float A1cpu[W*H]=0;
4   //--------------------------------------------------------------
5   void ofApp::setup()
6   {
7       shader.setupShaderFromFile(GL_COMPUTE_SHADER,"computeShader.cs");
8       shader.linkProgram();
9
10      A1.allocate(W*H*sizeof(float),A1cpu,GL_DYNAMIC_DRAW);
11
```

ofApp.cpp

Bufor trzeba podpiąć pod slot na GPU



Po stronie CPU

```
A1.bindBase(GL_SHADER_STORAGE_BUFFER, 0);
```

ofApp.cpp

Po stronie GPU - określamy co odbieramy

```
layout(binding = 0) buffer dcA1 { float A1 [  ]; };
```

computeshader.cs

Kod shadera

Shader czyszczący bufor A1

```
1  #version 440
2
3  layout(binding = 0) buffer dcA1 { float A1 [  ]; };
4  layout(local_size_x = 20, local_size_y = 20, local_size_z = 1) in;
5  const int W = 1280;
6  const int H = 720;
7
8  void main()
9  {
10         int i, j;
11         i = int(gl_GlobalInvocationID.x);
12         j = int(gl_GlobalInvocationID.y);
13
14         int idx = i+j*W;
15         A1[idx] = 0;
16 }
17
```

wersja

bufor

stałe
→ lepiej uniform?

schemat uruchomienia

pozycja w kracie

indeks tablicy 2D

algorytm

computeshader.cs

Shader czyszczący bufor A1

```
#version 440

layout(binding = 0) buffer dcA1 { float A1 [  ]; };
layout(local_size_x = 20, local_size_y = 20, local_size_z = 1) in;
const int W = 1280;
const int H = 720;

void main()
{
        int i, j;
        i = int(gl_GlobalInvocationID.x);
        j = int(gl_GlobalInvocationID.y);

        int idx = i+j*W;
        A1[idx] = 0;
}
```

computeshader.cs

Tekstura

Dodajemy teksturę

```
 7  class ofApp : public ofBaseApp
 8  {
 9      public:
10          void setup();
11          void update();
12          void draw();
13
14          ofBufferObject A1;
15          ofShader shader;
16
17          ofTexture tekstura;
18  };
```

ofApp.h

Alokacja tekstury + bindowanie

```
12
13      tekstura.allocate(W,H,GL_RGBA8);
14      tekstura.bindAsImage(1,GL_WRITE_ONLY);
15  }
16  //------------------------------------------------------------
```

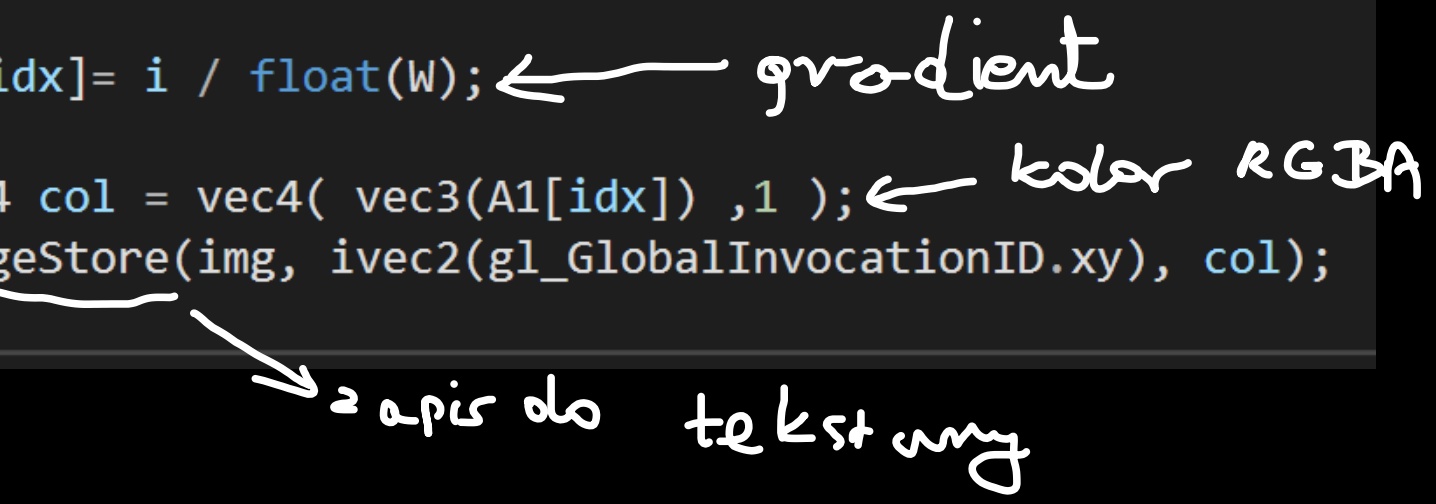
Rysowanie tekstury w oknie

```
24  //--------------------------------
25  void ofApp::draw()
26  {
27      tekstura.draw(0,0);
28  }
```

ofApp.cpp

Teraz możemy rysować po teksturze (shader)

```
10  void main()
11  {
12      int i, j;
13
14      i = int(gl_GlobalInvocationID.x);
15      j = int(gl_GlobalInvocationID.y);
16      int idx = i+j*W;
17
18      A1[idx]= i / float(W);
19
20      vec4 col = vec4( vec3(A1[idx]) ,1 );
21      imageStore(img, ivec2(gl_GlobalInvocationID.xy), col);
22  }
```

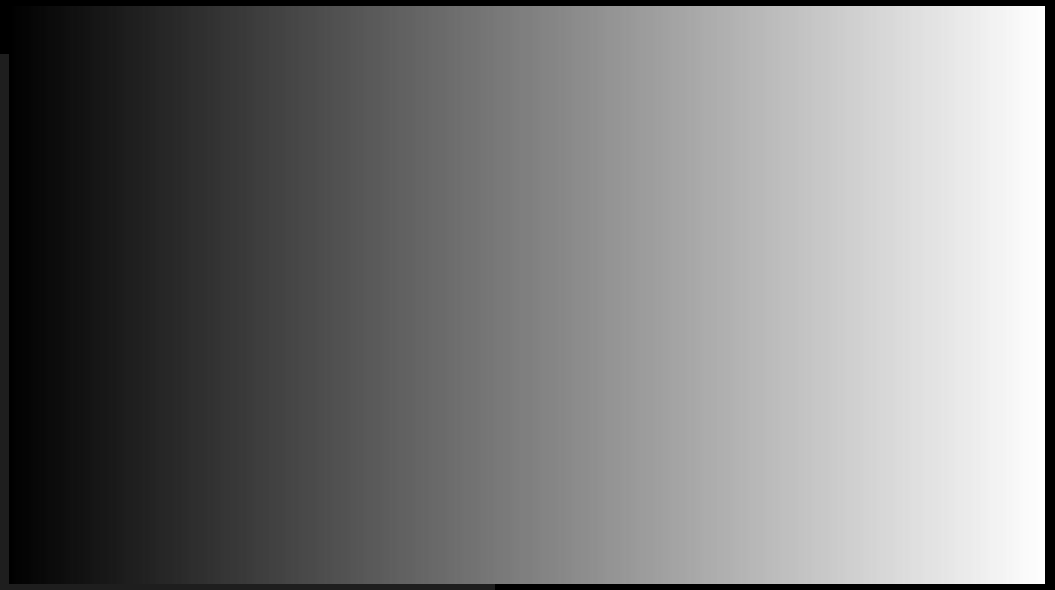


Binding tekstury po stronie shadera

```
4   layout(rgba8,binding=1) uniform writeonly image2D img;
```

Teraz możemy rysować po teksturze (shader)

```
10  void main()
11  {
12      int i, j;
13
14      i = int(gl_GlobalInvocationID.x);
15      j = int(gl_GlobalInvocationID.y);
16      int idx = i+j*W;
17
18      A1[idx]= i / float(W);
19
20      vec4 col = vec4( vec3(A1[idx]) ,1 );
21      imageStore(img, ivec2(gl_GlobalInvocationID.xy), col);
22  }
```



Część 2 - Tutorial Gray-Scott

- https://maciej-matyka.medium.com/gpu-compute-shaders-in-open-frameworks-tutorial-6ca7d21a196d

.