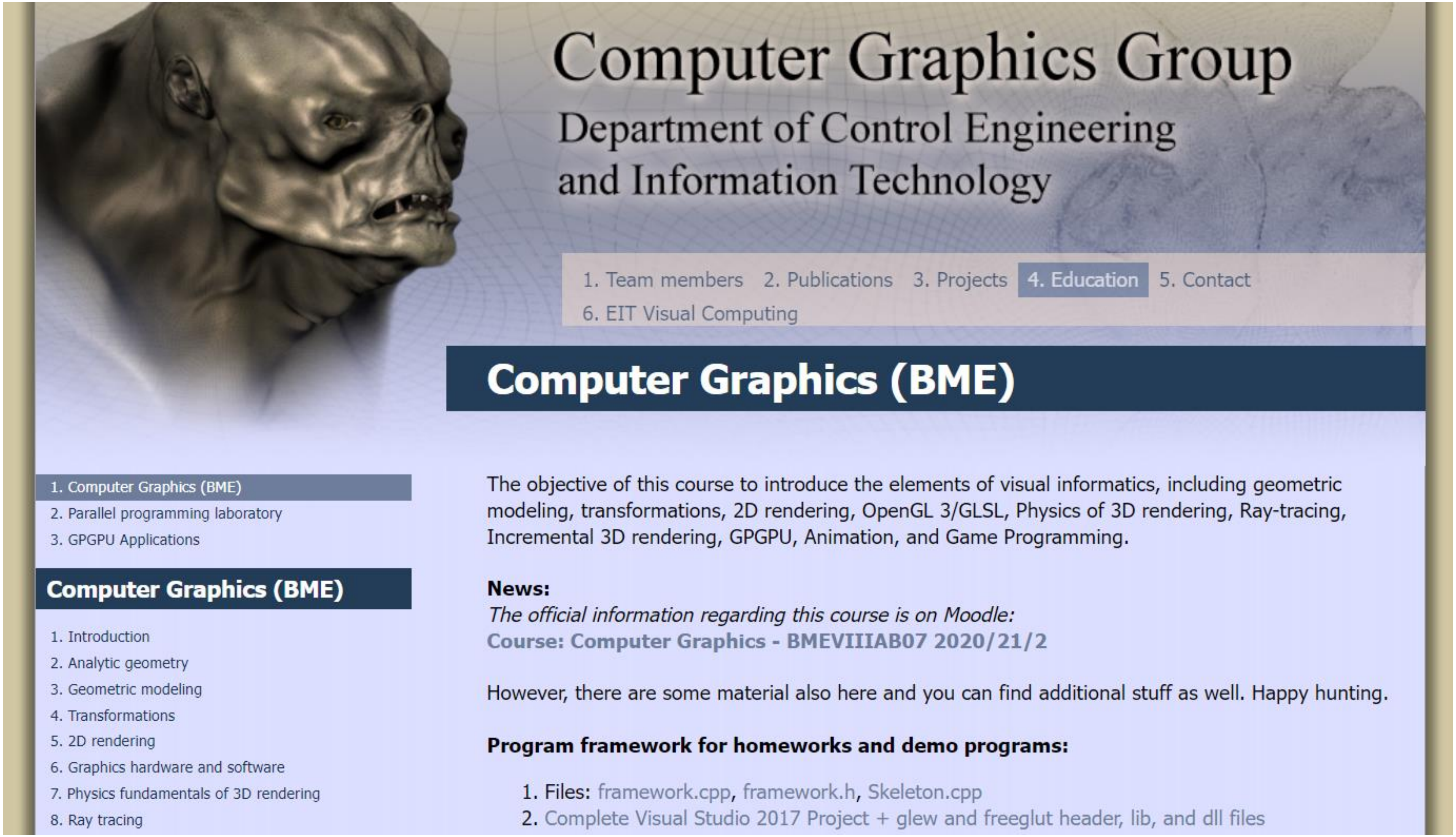
Computer Graphics Laboratory Exercise 1

Balázs Csébfalvi

http://cg.iit.bme.hu/portal/en/cgbme

Download and test the skeleton application



Computer Graphics Group

Department of Control Engineering and Information Technology

Computer Graphics (BME)

1. Computer Graphics (BME)
2. Parallel programming laboratory
3. GPGPU Applications

Computer Graphics (BME)

The objective of this course to introduce the elements of visual informatics, including geometric modeling, transformations, 2D rendering, OpenGL 3/GLSL, Physics of 3D rendering, Ray-tracing, Incremental 3D rendering, GPGPU, Animation, and Game Programming.

News:
The official information regarding this course is on Moodle:
Course: Computer Graphics - BMEVIIIAB07 2020/21/2

However, there are some material also here and you can find additional stuff as well. Happy hunting.

Program framework for homeworks and demo programs:

1. Files: framework.cpp, framework.h, Skeleton.cpp
2. Complete Visual Studio 2017 Project + glew and freeglut header, lib, and dll files

C++/OpenGL

- CPU: C++ skeleton program

```
…
unsigned int vbo;               // vertex buffer object
glGenBuffers(1, &vbo);          // Generate 1 buffer
glBindBuffer(GL_ARRAY_BUFFER, vbo);
// Geometry with 24 bytes (6 floats or 3 x 2 coordinates)
float vertices[] = { 0.0f, 0.0f, 1.0f, 0.0f, 0.0f, 1.0f };
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
…
```

- GPU: GLSL shader programs

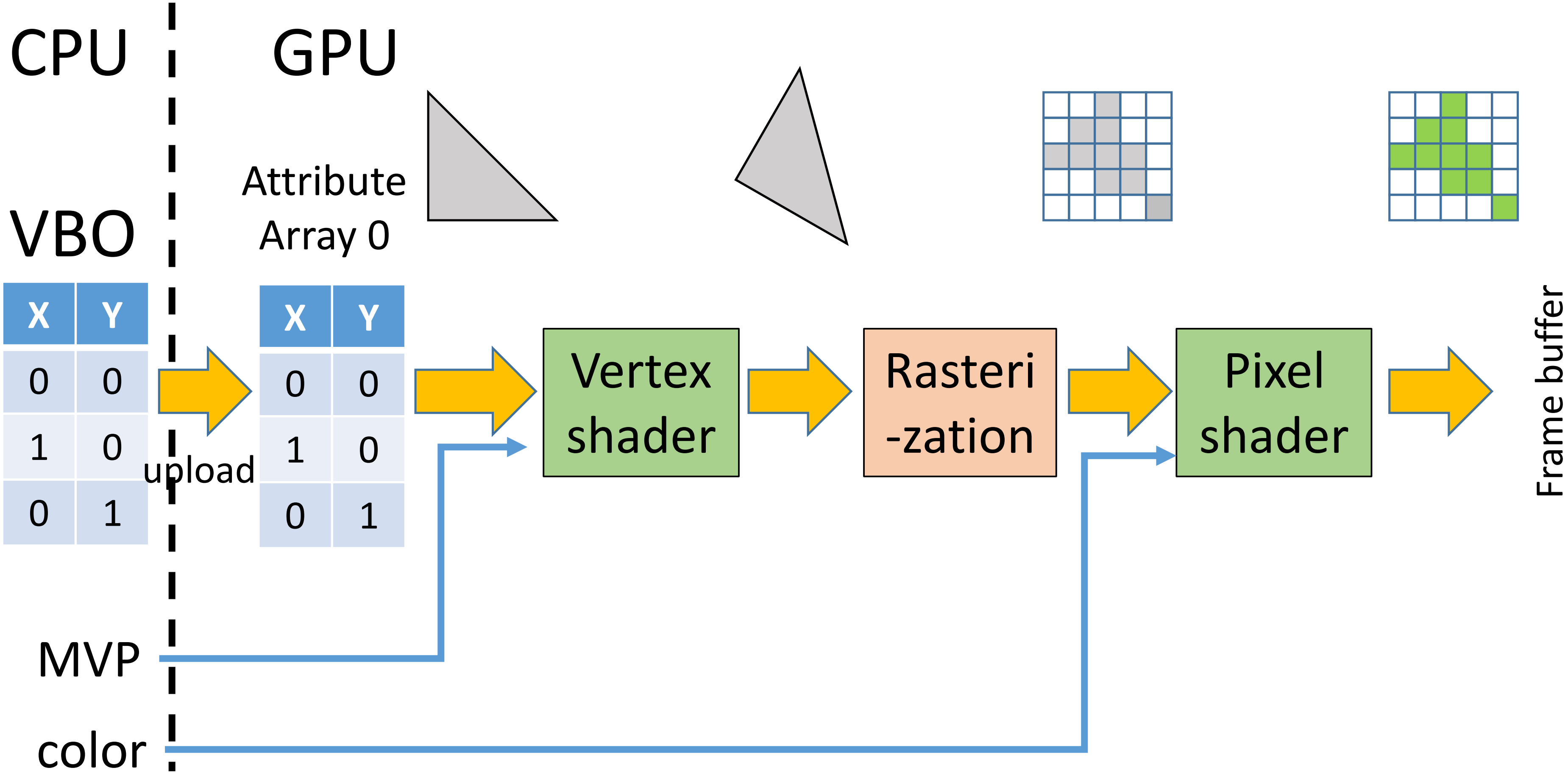
Vertex shader

```
#version 330
precision highp float;
uniform mat4 MVP;
layout(location = 0) in vec2 vp;
void main() {
  gl_Position = vec4(vp.x, vp.y, 0, 1) * MVP;
}
```

Pixel shader

```
#version 330
precision highp float;
uniform vec3 color;
out vec4 outColor;
void main() {
  outColor = vec4(color, 1);
}
```

GPU pipeline



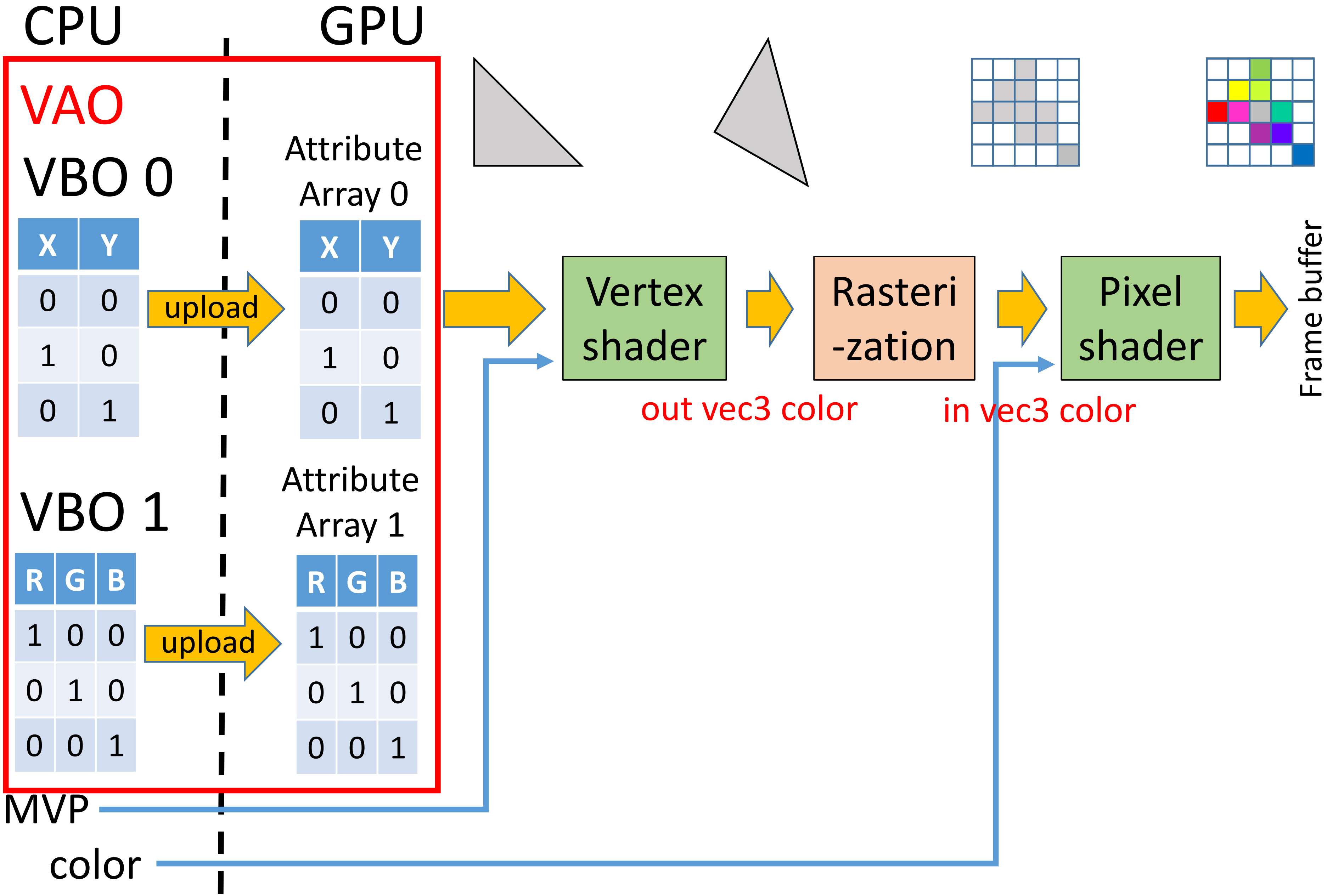
VBO

X	Y
0	0
1	0
0	1

Attribute Array 0

X	Y
0	0
1	0
0	1

Add another VBO for vertex colors



CPU

GPU

VAO

VBO 0

X	Y
0	0
1	0
0	1

upload

Attribute Array 0

X	Y
0	0
1	0
0	1

VBO 1

R	G	B
1	0	0
0	1	0
0	0	1

upload

Attribute Array 1

R	G	B
1	0	0
0	1	0
0	0	1

Vertex shader

Rasteri -zation

Pixel shader

Frame buffer

out vec3 color

in vec3 color

MVP

color

C++/OpenGL

CPU: C++ skeleton program

```
glGenVertexArrays(1, &vao); glBindVertexArray(vao);

unsigned int vbo[2]; glGenBuffers(2, vbo);

glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
float vertices[] = { 0.0f, 0.0f, 1.0f, 0.0f, 0.0f, 1.0f };
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 0, NULL);

glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
float colors[] = { 1.0, 0.0, 0.0,0.0, 1.0, 0.0,0.0, 0.0, 1.0 };
glBufferData(GL_ARRAY_BUFFER, sizeof(colors), colors, GL_STATIC_DRAW);
glEnableVertexAttribArray(1);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 0, NULL);
```

C++/OpenGL

GPU: GLSL shader programs

Vertex shader

```
#version 330
precision highp float;

uniform mat4 MVP;
layout(location = 0) in vec2 vp;
layout(location = 1) in vec3 vc;
out vec3 color;

void main() {
  gl_Position = vec4(vp.x, vp.y, 0, 1) * MVP;
  color = vc;
}
```

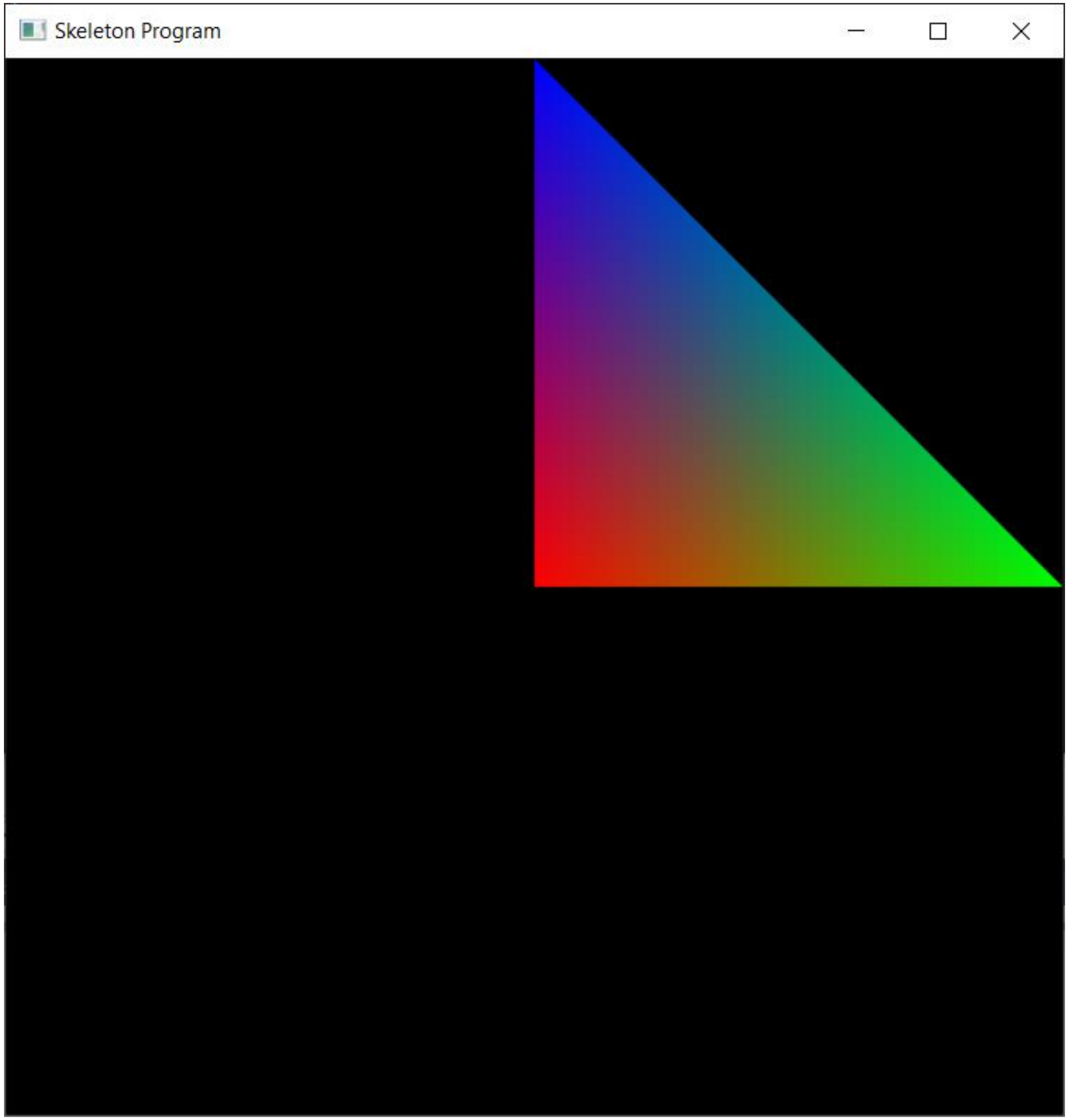
Pixel shader

```
#version 330
precision highp float;

//uniform vec3 color;
in vec3 color;
out vec4 outColor;

void main() {
  outColor = vec4(color, 1);
}
```

Triangle rendering with color interpolation



Directly uploading the transformation matrix

```
float MVPtransf[4][4] = { 1, 0, 0, 0,    // MVP matrix,
  0, 1, 0, 0,    // row-major!
  0, 0, 1, 0,
  -0.5, 0, 0, 1 };

// Get the GPU location of uniform variable MVP:
int location = glGetUniformLocation(gpuProgram.getId(), "MVP");

// Upload a 4x4 row-major float matrix to the specified location:
glUniformMatrix4fv(location, 1, GL_TRUE, &MVPtransf[0][0]);
```

Vertex shader

```
#version 330
precision highp float;
uniform mat4 MVP;
layout(location = 0) in vec2 vp;
layout(location = 1) in vec3 vc;
out vec3 color;
void main() {
  gl_Position = vec4(vp.x, vp.y, 0, 1) * MVP;
  color = vc;
}
```

Use the setUniform function of the GPUProgram instead

```
mat4 MVP = ScaleMatrix(vec3(1.0, 1.0, 1.0));
gpuProgram.setUniform(MVP, "MVP");
```

Vertex shader

```
#version 330
precision highp float;
uniform mat4 MVP;
layout(location = 0) in vec2 vp;
layout(location = 1) in vec3 vc;
out vec3 color;
void main() {
  gl_Position = vec4(vp.x, vp.y, 0, 1) * MVP;
  color = vc;
}
```

Measure the elapsed time in a global variable

```
long time = 0;

…

// Idle event indicating that some time elapsed: do animation here
void onIdle() {
time = glutGet(GLUT_ELAPSED_TIME); // elapsed time since the start of the program
glutPostRedisplay();
}
```

Change the position of the triangle periodically

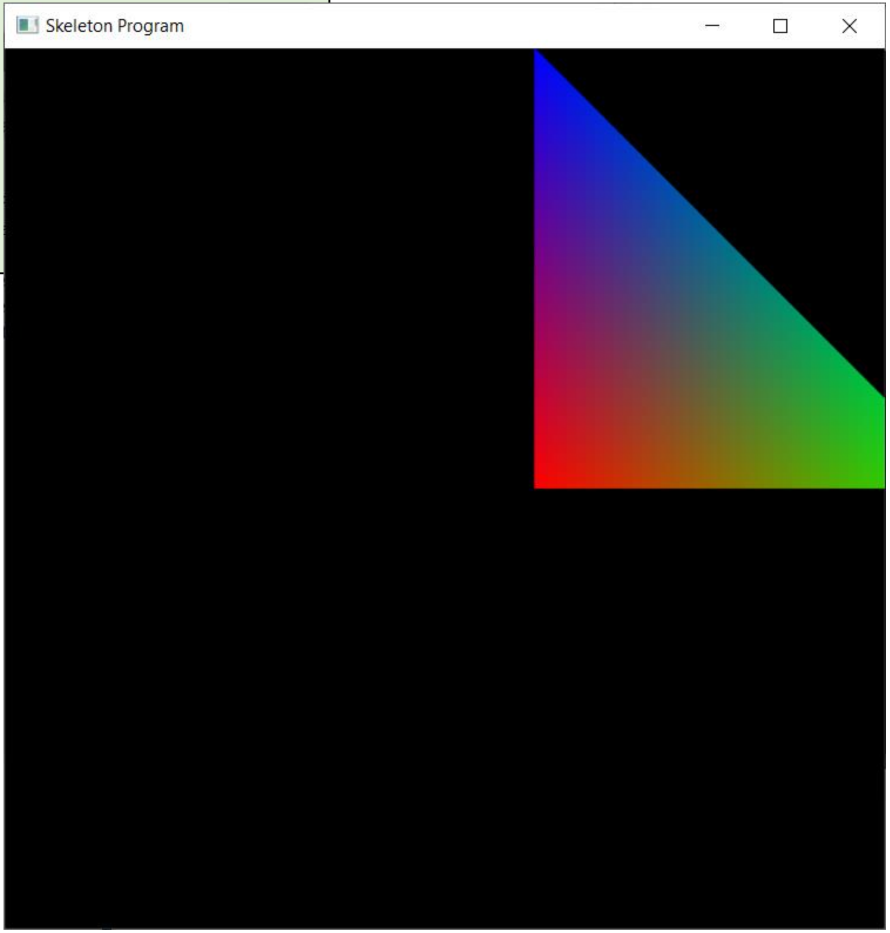
```
void onDisplay() {

…

mat4 MVP = TranslateMatrix(vec3(sin(time / 1000.0), 0.0, 0.0));
gpuProgram.setUniform(MVP, "MVP");

…

}
```



Change the orientation of the triangle periodically

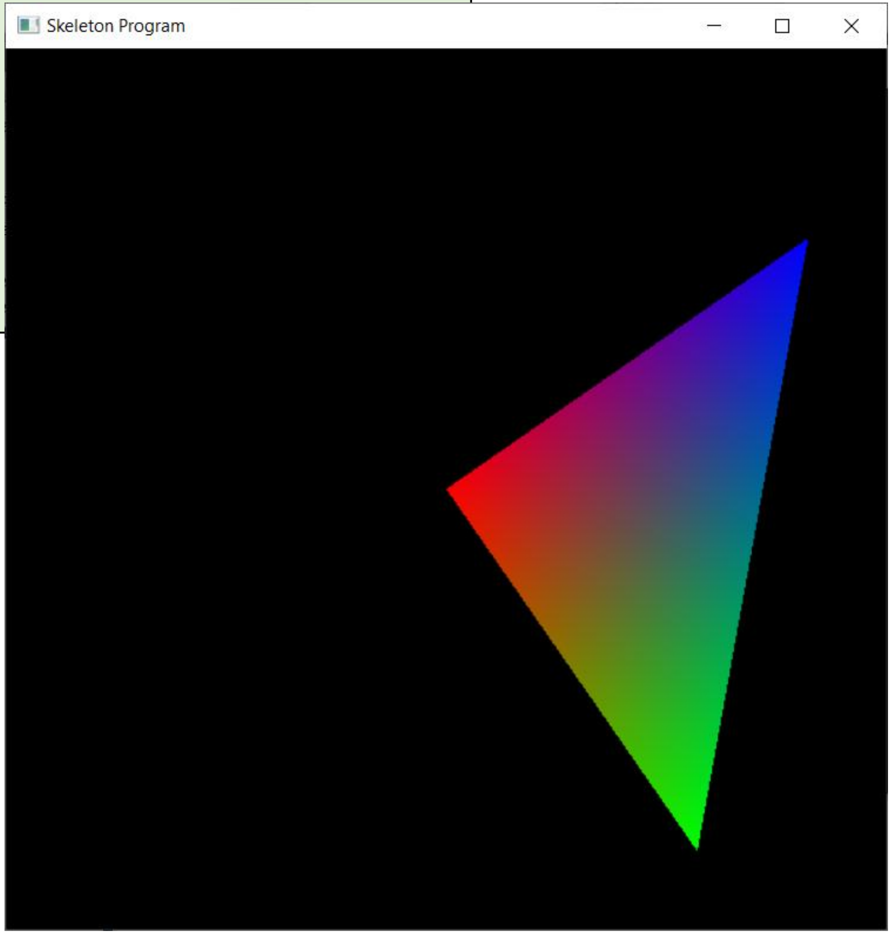
```
void onDisplay() {

…

mat4 MVP = RotationMatrix(M_PI * time / 1000.0, vec3(0.0, 0.0, 1.0));
gpuProgram.setUniform(MVP, "MVP");


…

}
```



Define class Triangle,
add functions Create and Render

```
class Triangle
{
	unsigned int vao;

public:
	void Create()
	{
		// create and upload VBOs here
	}

	void Render()
	{
		glBindVertexArray(vao);
		glDrawArrays(GL_TRIANGLES, 0, 3);
	}
};

Triangle triangle;
```

```
void onInitialization() {

…

triangle.Create();

…
}
```

```
void onDisplay() {

…

triangle.Render();

…
}
```