ESCOLA DE NOVES TECNOLOGIES INTERACTIVES

centre adscrit a:

UNIVERSITAT DE BARCELONA

Computer Graphics

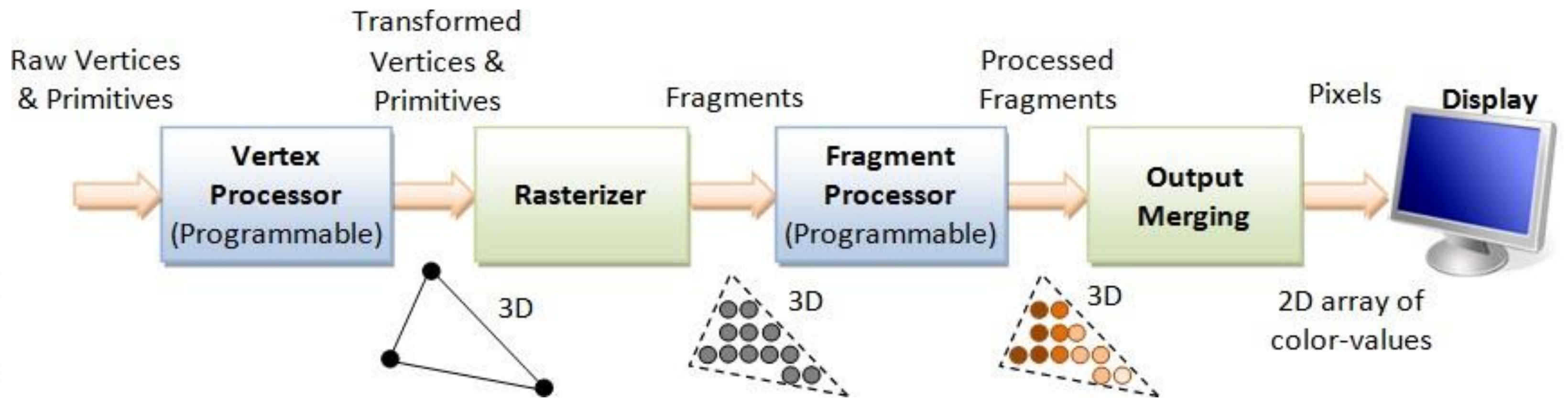
11. Frame Buffer Objects

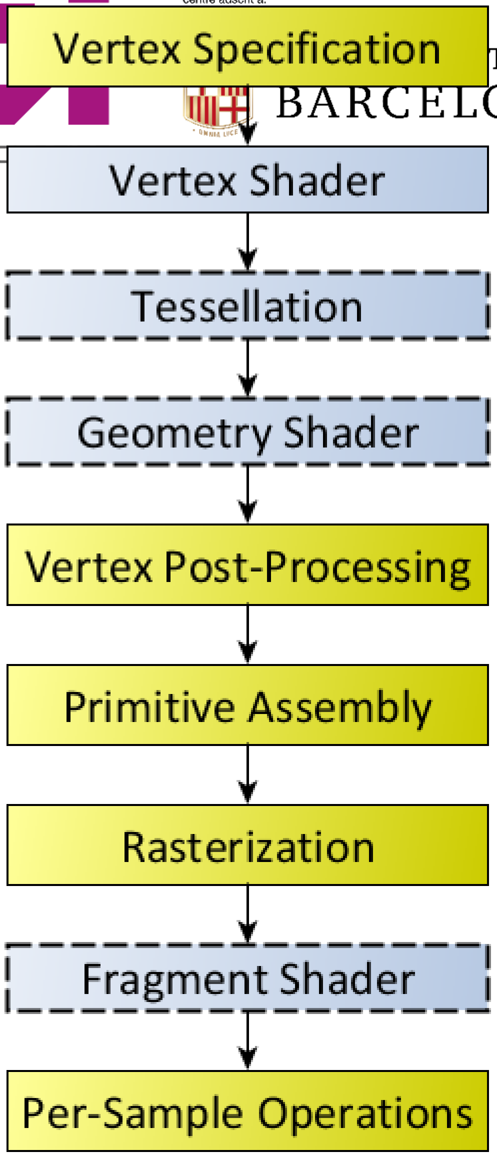
Dr Joan Llobera – joanllobera@enti.cat
Dr Jesús Ojeda – jesusojeda@enti.cat

Spring 2019

Contents

1. Reminder of the graphics pipeline


Raw Vertices & Primitives
Vertex Processor (Programmable)
Transformed Vertices & Primitives
Rasterizer
Fragments
Fragment Processor (Programmable)
Processed Fragments
Output Merging
Pixels
Display
3D
3D
3D
2D array of color-values



What is a Frame buffer object?

- A Frame Buffer Object, commonly called FBO, is a frame buffer considered as an object.
- The outcome is a fragment (i.e., almost a pixel).
- An FBO can handle the entire rendering pipeline, but not assign it to the screen. Yes, this means that the result is off-screen.
- A frame buffer has render buffers: color buffer, but also (optionally) stencil buffer or depth buffer

Why do I want to use Frame buffer objects?

Why would we want to render if it is not to the screen?

Having a frame buffer object allows having multiple rendering pipelines, which can be combined in a common scene.

It also allows pre-processing parts of a scene, to be used later on

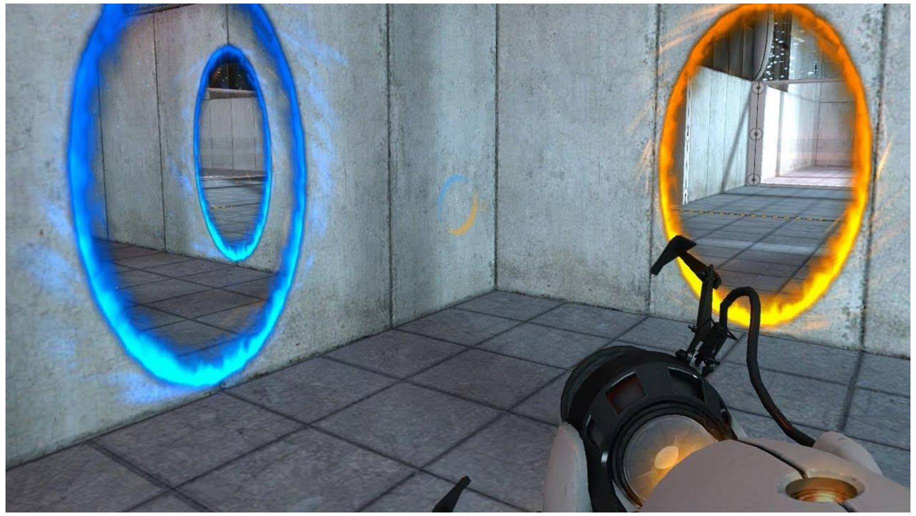
Specifically:

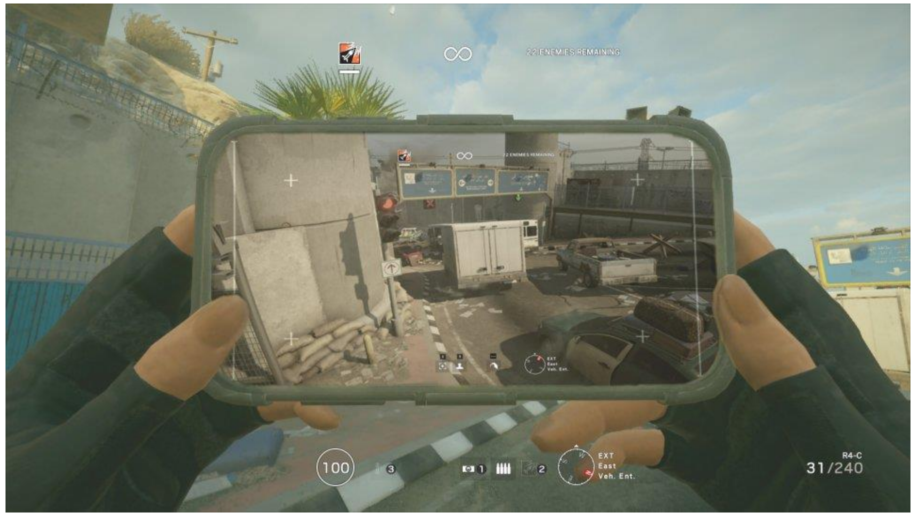
- Render to texture (today)
- Visual effects (next class)

FBO for Render To Texture

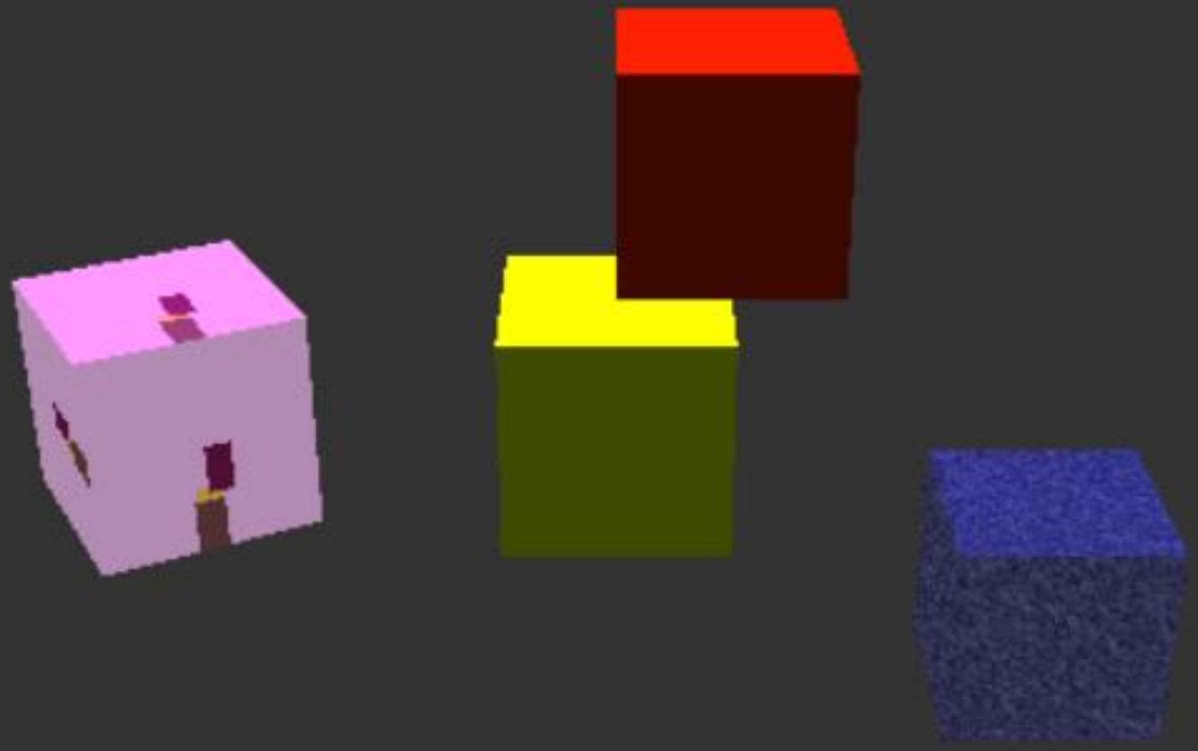
Why would we want to render if it is not to the screen?

- Render to texture





How to set it up?



How to set it up?

You will need:

1. An object rendering a texture
2. A frame buffer object
3. A texture where to store the outcome of the frame buffer shaders

How to set it up?

An object rendering a texture

```
#define STB_IMAGE_IMPLEMENTATION
#include "stb_image.h"
```

```
//exactly the usual shader, only adding the
normal transmission from in to out
const char* cube_vertShader =
"#version 330\n\
in vec3 in_Position;\n\
in vec2 in_Tex;\n\
in vec3 in_Normal;\n\
out vec4 vert_Normal;\n\
out vec2 vert_Tex;\n\
uniform mat4 objMat;\n\
uniform mat4 mv_Mat;\n\
uniform mat4 mvpMat;\n\
void main() {\n\
gl_Position = mvpMat * objMat *
vec4(in_Position, 1.0);\n\
vert_Normal = mv_Mat * objMat * vec4(in_Normal,
0.0);\n\
vert_Tex = in_Tex;\n\
}";
```

How to set it up? (2)

An object rendering a texture

```
const char* cube_fragShaderWithTexture =
"#version 330\n\
in vec4 vert_Normal;\n\
in vec2 vert_Tex;\n\
out vec4 out_Color;\n\
uniform mat4 mv_Mat;\n\
uniform vec4 color;\n\
uniform sampler2D tex;\n\
void main() {\n\
out_Color = texture(tex, vert_Tex * vec2(1.0, -
1.0))*0.5+ vec4(color.xyz * dot(vert_Normal,
mv_Mat*vec4(0.0, 1.0, 0.0, 0.0))*0.3 +
color.xyz * 0.2, 1.0 );\n\
}";
```

```
//we create a second program to draw a cube
using a texture:
cubeShaders[0] = compileShader(cube_vertShader,
GL_VERTEX_SHADER, "cubeVert");
cubeShaders[1] =
compileShader(cube_fragShaderWithTexture,
GL_FRAGMENT_SHADER, "cubeFrag");


cubeProgramWithTexture = glCreateProgram();
glAttachShader(cubeProgramWithTexture,
cubeShaders[0]);
glAttachShader(cubeProgramWithTexture,
cubeShaders[1]);
glAttachShader(cubeProgramWithTexture,
cubeShaders[2]);
glBindAttribLocation(cubeProgramWithTexture, 0,
"in_Position");
glBindAttribLocation(cubeProgramWithTexture, 1,
"in_Normal");
glBindAttribLocation(cubeProgramWithTexture, 2,
"in_Tex");
linkProgram(cubeProgramWithTexture);
```

How to set it up? (3)

An object rendering a texture

```
// to draw a normal texture, from a .jpg file:
void setupTexture4Cube() {
int x, y, n;
unsigned char *dat = stbi_load("text1.jpg", &x, &y, &n, 3);
glGenTextures(1, &cubeTexture);
glBindTexture(GL_TEXTURE_2D, cubeTexture);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, x, y, 0, GL_RGB,
GL_UNSIGNED_BYTE, dat);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
stbi_image_free(dat);

}
```

How to set it up? (4)

An object rendering a texture

```
// to draw a normal texture, from a .jpg file:
void setupTexture4Cube() {
int x, y, n;
unsigned char *dat = stbi_load("text1.jpg", &x, &y, &n, 3);
glGenTextures(1, &cubeTexture);
glBindTexture(GL_TEXTURE_2D, cubeTexture);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, x, y, 0, GL_RGB,
GL_UNSIGNED_BYTE, dat);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
stbi_image_free(dat);

}
```

Exercise:

a) write a function that allows drawing a cube changing the following parameters:

```
void drawCubeAt(glm::vec3 pos, glm::vec3 color, float r, GLuint myProgram){…}
```

b) Use it to draw the typical cube in the center of the scene, and an additional cube that rotates, on top of it
c) Use it to draw the typical cube. Use the following call:

```
drawCubeAt(glm::vec3(.0f, .0f, .0f), glm::vec3(0.8f, 1.f, 0.f), 0, cubeProgram);
```

d) Use it to draw a cube with a texture. Use the following call:

```
glBindTexture(GL_TEXTURE_2D, cubeTexture);
drawCubeAt(glm::vec3(2.0f, -1.0f, .0f), glm::vec3(0.1f, .1f, 0.8f), 0,
cubeProgramWithTexture);
```

How to set it up? (6)

Up to now we just reminded you how to create a texture for an object.

Now, we need to create:

1. A frame buffer
2. A texture where to store the outcome of the frame buffer shaders

To create a frame buffer:

```
// to draw a scene to a texture, we need a frame buffer:
void setupFBO() {

/////setup FBO texture
glGenFramebuffers(1, &fbo);
//create texture exactly as before:
glGenTextures(1, &fbo_tex);
glBindTexture(GL_TEXTURE_2D, fbo_tex);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, 800, 800, 0, GL_RGB, GL_UNSIGNED_BYTE, NULL);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);

//if we need a depth or stencil buffer, we do it here

//we bind texture (& renderbuffer) to framebuffer
glBindFramebuffer(GL_FRAMEBUFFER, fbo);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, fbo_tex, 0);
//if we had depth or stencil, we would do it here.
}
```

To draw in a frame buffer (1/2):

```
void drawCubeFBOTex() {

//we store the current values in a temporary variable
glm::mat4 t_mvp = RenderVars::_MVP;
glm::mat4 t_mv = RenderVars::_modelView;



//we set up our framebuffer and draw into it
glBindFramebuffer(GL_FRAMEBUFFER, fbo);
glClearColor(1.f, 1.f, 1.f, 1.f);
glViewport(0, 0, 800, 800);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glEnable(GL_DEPTH_TEST);
RenderVars::_MVP = RenderVars::_projection;
RenderVars::_modelView = glm::mat4(1.f);

//everything you want to draw in your texture should go here
Cube::objMat = glm::lookAt(glm::vec3(0.f, 1.5f, 3.5f), glm::vec3(0.f, 1.5f, 0.f), glm::vec3(0.f
1.f, 0.f));
Cube::draw2Cubes();
```

To draw in a frame buffer (2/2):

```
//we restore the previous conditions
RenderVars::_MVP = t_mvp;
RenderVars::_modelView = t_mv;
glBindFramebuffer(GL_FRAMEBUFFER, 0);

//we set up a texture where to draw our FBO:
glViewport(0, 0, g_width, g_height);
glBindTexture(GL_TEXTURE_2D, fbo_tex);

glm::vec3 c1_pos = glm::vec3(-2.f, 0.f, 0.f);
drawCubeAt(c1_pos, glm::vec3(1.0f, 0.2f, 1.f),0.5f, cubeProgramWithTexture);

}
```

Resources

- Graham Sellers, Richard S. Writght, Jr. Nicholas Haemel. **OpenGL SuperBible**, 6th Edition. Pearson education.
- John Kessenich, Graham Sellers, Dave Shreiner. **OpenGL Programming guide**. Ninth Edition. Pearson Education.