Computer Graphics II
– Framebuffer (Questions)

Kai Lawonn

Possible Questions

How to create and bind a framebuffer?

```
unsigned int framebuffer;
```

Possible Questions

How to create and bind a framebuffer?

```
unsigned int framebuffer;
glGenFramebuffers(1, &framebuffer);
glBindFramebuffer(GL_FRAMEBUFFER, framebuffer);
```

Possible Questions

How to attach a texture to a framebuffer?

Possible Questions

How to attach a texture to a framebuffer?

- Create a texture, but set the dimensions equal to the screen size (not required) and pass NULL as the texture's data parameter
- Attach the texture to the framebuffer:

```
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D,
texture, 0);
```

Possible Questions

What is an RBO?

Possible Questions

What is an RBO?

- Renderbuffer objects (RBOs) are a possible type of framebuffer attachments
- Like a texture image, an RBO is an actual buffer (an array of bytes, integers, pixels,...)
- RBO advantage that it stores its data in OpenGL's native rendering format → optimized for off-screen rendering to a framebuffer

Kernel Effects

```
const float offset = 1.0 / 300.0;
void main()
{
vec2 offsets[9] = vec2[](vec2(-offset, offset), // top-left
                         vec2( 0.0f,offset), // top-center
                         vec2( offset, offset), // top-right
                         vec2(-offset, 0.0f), // center-left
                         vec2( 0.0f,0.0f), // center-center
                         vec2( offset, 0.0f), // center-right
                         vec2(-offset, -offset), // bottom-left
                         vec2( 0.0f, -offset), // bottom-center
                         vec2( offset, -offset) // bottom-right
);

float kernel[9] = float[](
-1, -1, -1,
-1, 9, -1,
-1, -1, -1
);
```

Possible Questions

Complete the code to apply a kernel:

```
vec3 sampleTex[9];
for(int i = 0; i < 9; i++)
    sampleTex[i] = vec3(texture(screenTexture, TexCoords.st + offsets[i]));

vec3 col = vec3(0.0);
                                    
    
                                    * kernel[i];

FragColor = vec4(col, 1.0);
}
```

Possible Questions

Complete the code to apply a kernel:

```
vec3 sampleTex[9];
for(int i = 0; i < 9; i++)
    sampleTex[i] = vec3(texture(screenTexture, TexCoords.st + offsets[i]));

vec3 col = vec3(0.0);
for(int i = 0; i < 9; i++)
    col += sampleTex[i] * kernel[i];

FragColor = vec4(col, 1.0);
}
```