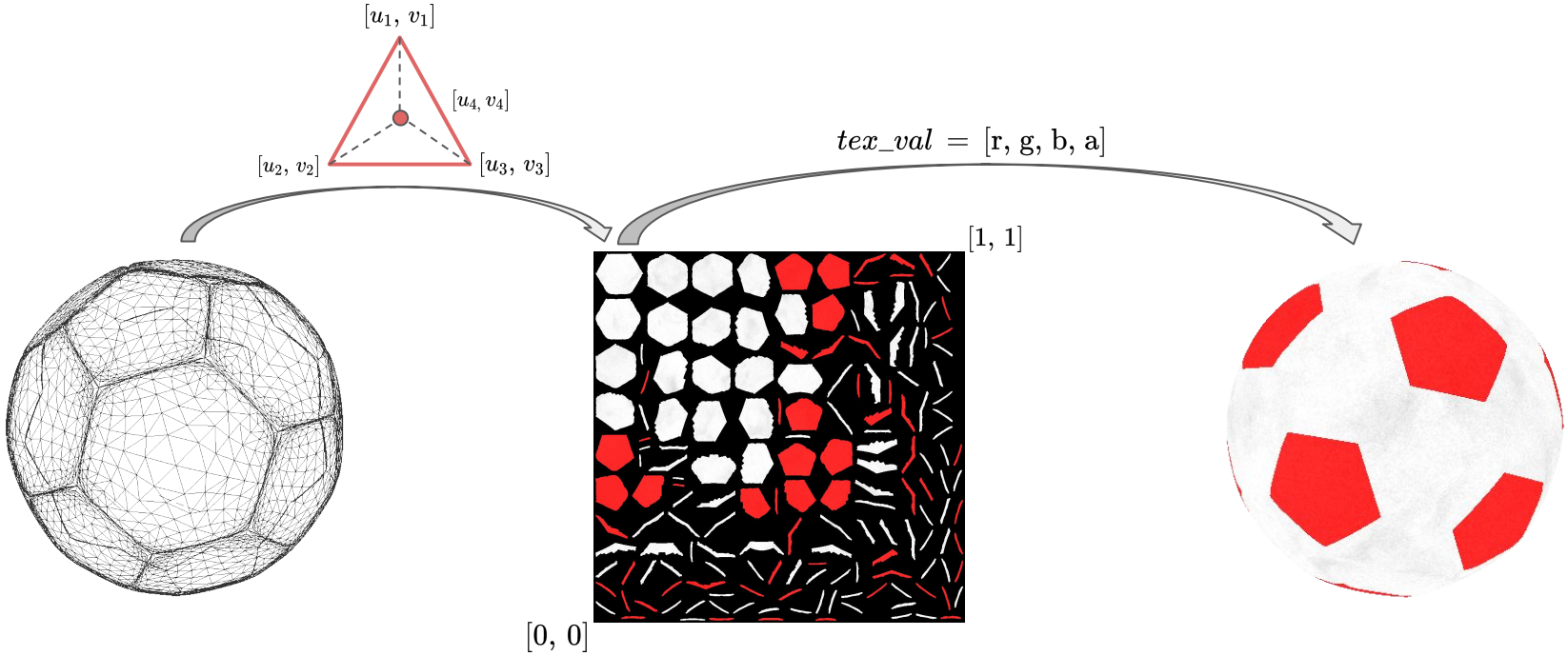
Textures

Evangelou Iordanis

Presentation co-author Anastasios Gkaravelis

Sampler units

- OpenGL supports (optionally compressed)
  - 1D, 2D, 3D, cubemaps
  - texture arrays
- Texture coordinates are in range [0, 1]



Sampler units

- Load images with SDL library

```
// 2D texture example
SDL_Surface* surf = IMG_Load(filename);
unsigned char* data = surf->pixels;

Gluint texHandle;
glGenTextures(1, &texHandle);
glBindTexture(GL_TEXTURE_2D, texHandle);

SDL_LockSurface(surf);
glTexImage2D(GL_TEXTURE_2D,      // target
       0,                        // level-of-detail
       GL_RGBA32F,               // internal format / number of components
       surf->w, surf->h,         // size
       0,                        // border (must be zero)
       GL_BGRA,                  // format
       GL_UNSIGNED_BYTE,         // type
       data);                    // data buffer
SDL_UnlockSurface(surf);

glGenerateMipmap(GL_TEXTURE_2D); // generate mipmap if needed
glBindTexture(GL_TEXTURE_2D, 0);
```

Sampler units

- Access textures in shader units through uniform sampler objects

```
#version 330 core
layout(location = 0) out vec4 out_color;

in vec2 f_texcoord;

uniform sampler2D uniform_texture1;
uniform sampler2D uniform_texture2;

void main(void)
{
        vec4 tex_val = texture(uniform_texture1, f_texcoord);
        out_color = tex_val;
}
```

```
glActiveTexture(GL_TEXTURE0);
glUniform1i(glGetUniformLocation(program, "uniform_texture1"), 0);
glBindTexture(GL_TEXTURE_2D, texHandle1);

glActiveTexture(GL_TEXTURE1);
glUniform1i(glGetUniformLocation(program, "uniform_texture2"), 1);
glBindTexture(GL_TEXTURE_2D, texHandle2);
```

Sampler wrapping

```
glBindTexture(GL_TEXTURE_2D, texHandle);

glTexParameteri(GL_TEXTURE_2D,
       GL_TEXTURE_WRAP_S,
       GL_REPEAT);


glTexParameteri(GL_TEXTURE_2D,
       GL_TEXTURE_WRAP_T,
       GL_REPEAT);
```

GL_REPEAT

GL_MIRRORED_REPEAT

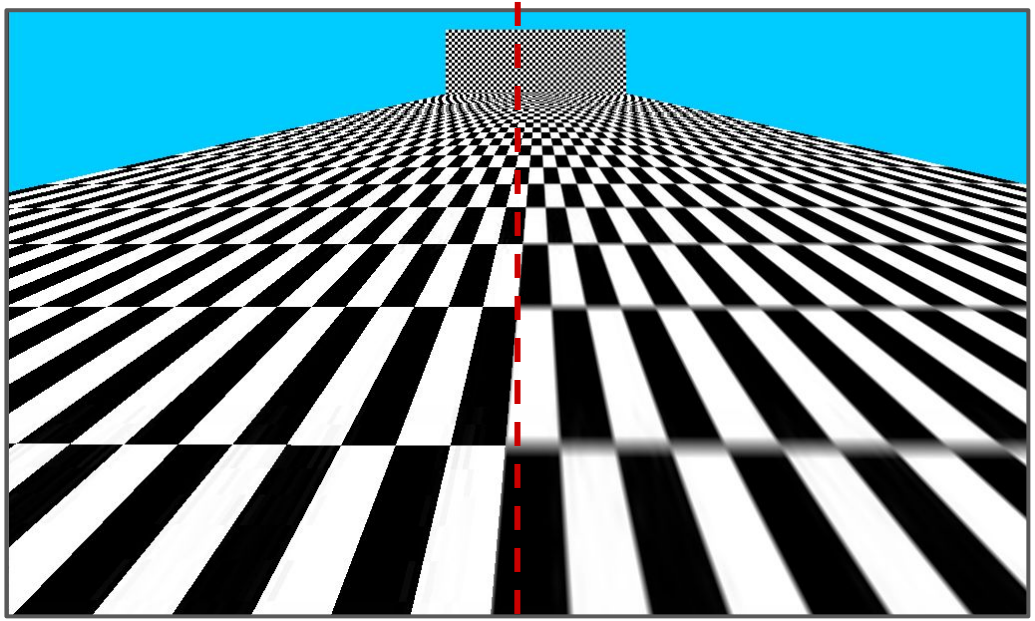
GL_CLAMP_TO_EDGE

GL_CLAMP_TO_BORDER

Sampler filtering

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
```

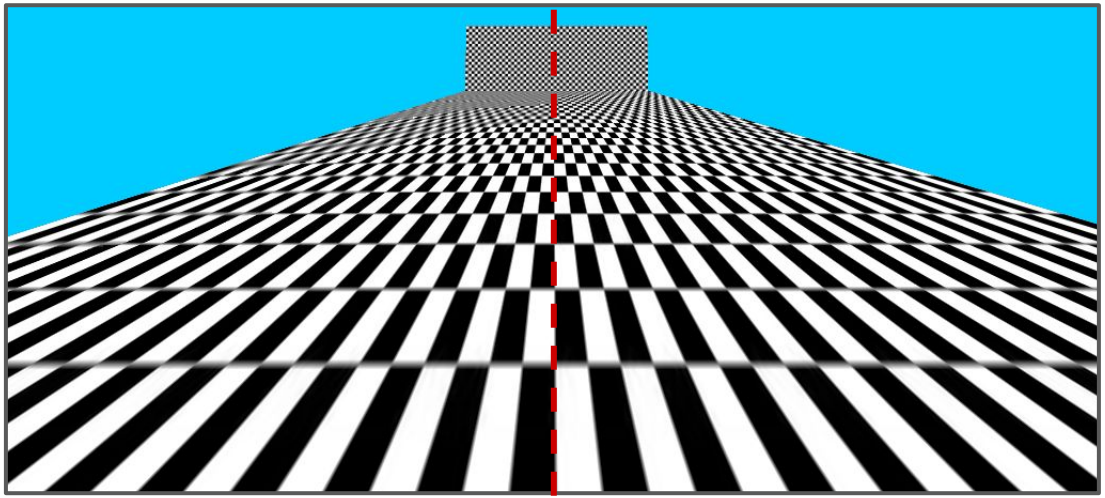
```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```



Sampler mipmap

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
        GL_LINEAR_MIPMAP_NEAREST);
```

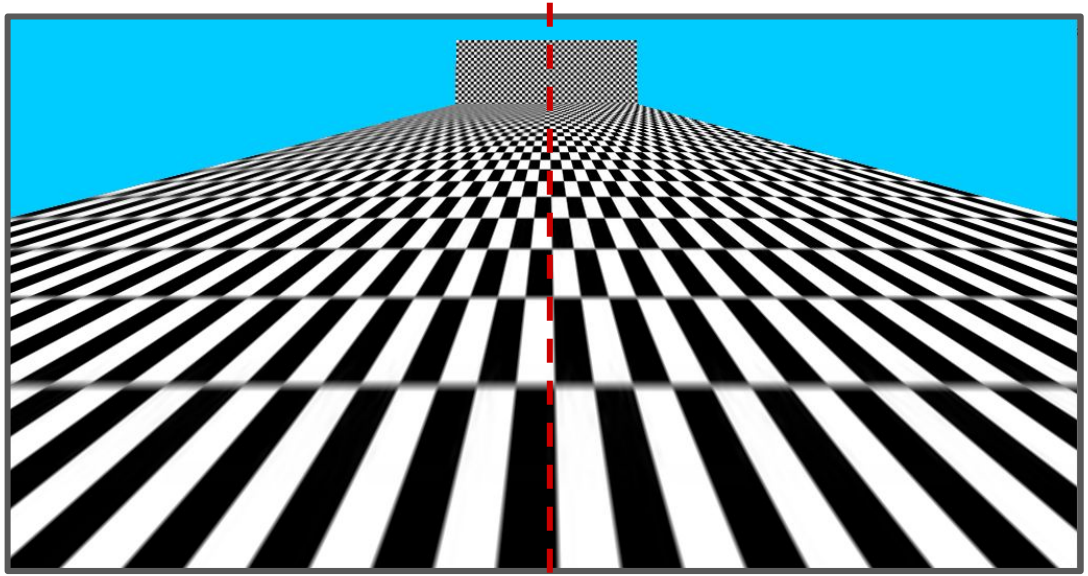
```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
        GL_LINEAR_MIPMAP_NEAREST);

GLint max_anisotropy = 1;
glGetIntegerv(GL_MAX_TEXTURE_MAX_ANISOTROPY_EXT, &max_anisotropy);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAX_ANISOTROPY_EXT, max_anisotropy);
```



Sampler mipmap

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
        GL_LINEAR_MIPMAP_LINEAR);
```

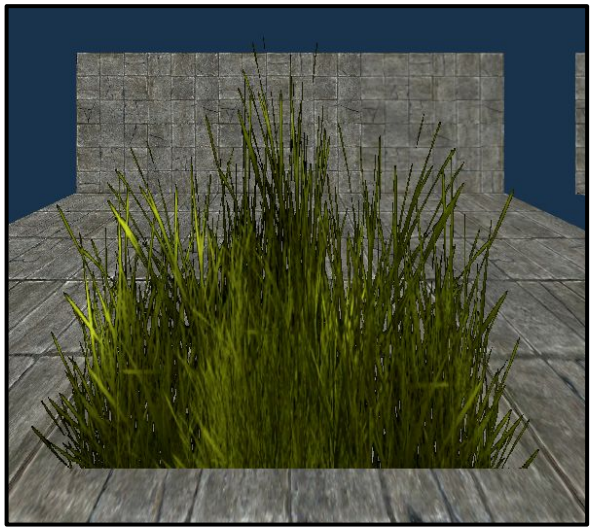
```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

glTexParameteri(GL_TEXTURE_2D,
        GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);

GLint max_anisotropy = 1;
glGetIntegerv(GL_MAX_TEXTURE_MAX_ANISOTROPY_EXT, &max_anisotropy);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAX_ANISOTROPY_EXT, max_anisotropy);
```



Alpha culling / blending

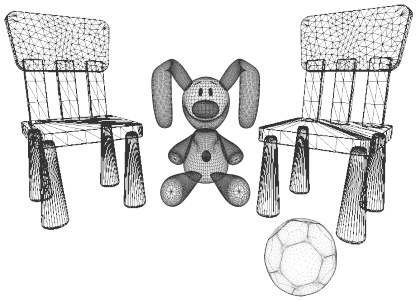
- Fragments can be discarded (using the discard command inside the fragment shader)
- Discard based on the alpha channel
  - `if(texture_color.a < 0.1) discard;`
- Alternatively enable blending

Alpha culling

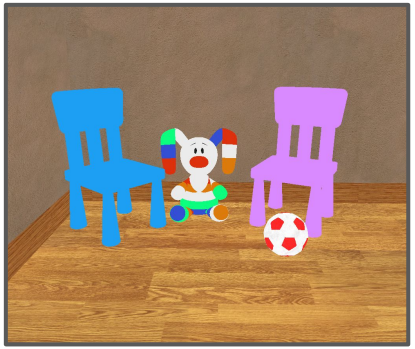


Alpha blending

Post processing



Forward pass



Post process

Screen

Post processing

- Construct a custom framebuffer
  - Requires at least one color and depth attachment

```
GLuint fbo_texture = 0;
GLuint fbo_depth_texture = 0;
GLuint fbo = 0;

glGenTextures(1, &fbo_texture);
glGenTextures(1, &fbo_depth_texture);
glGenFramebuffers(1, &fbo);
```

```
glBindTexture(GL_TEXTURE_2D, m_fbo_texture);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA32F,
        m_screen_width, m_screen_height,
        0, GL_RGBA, GL_UNSIGNED_BYTE, NULL);

glBindTexture(GL_TEXTURE_2D, m_fbo_depth_texture);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);

glTexImage2D(GL_TEXTURE_2D, 0,
        GL_DEPTH_COMPONENT24,
        m_screen_width, m_screen_height,
        0,
        GL_DEPTH_COMPONENT,
        GL_FLOAT, NULL);

glBindTexture(GL_TEXTURE_2D, 0);
```

Post processing

- Construct a custom framebuffer
  - Requires at least one color and depth attachment
- Bind textures to framebuffer

```
GLuint fbo_texture = 0;
GLuint fbo_depth_texture = 0;
GLuint fbo = 0;

glGenTextures(1, &fbo_texture);
glGenTextures(1, &fbo_depth_texture);
glGenFramebuffers(1, &fbo);

// [..] initialize color attachments and depth textures

// framebuffer to link to everything together
glBindFramebuffer(GL_FRAMEBUFFER, m_fbo);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, m_fbo_texture, 0);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, GL_TEXTURE_2D, m_fbo_depth_texture, 0);
```

Post processing

- Construct a custom framebuffer
  - Requires at least one color and depth attachment
- Bind textures to framebuffer
- Bind the specific framebuffer and draw geometry

```
glBindFramebuffer(GL_FRAMEBUFFER, m_fbo);
GLenum drawbuffers[1] = { GL_COLOR_ATTACHMENT0 };
glDrawBuffers(1, drawbuffers);

//[..] enable draw operations (e.g. blending)
//[..] draw calls
//[..] disable appropriate buffers
```

Post processing

- Construct a custom framebuffer
  - Requires at least one color and depth attachment
- Bind textures to framebuffer
- Bind the specific framebuffer and draw geometry
- Bind the default framebuffer and process the final result

```
glBindFramebuffer(GL_FRAMEBUFFER, m_fbo);
GLenum drawbuffers[1] = { GL_COLOR_ATTACHMENT0 };
glDrawBuffers(1, drawbuffers);

//[..] enable draw operations (e.g. blending)
//[..] draw calls
//[..] disable appropriate buffers

glBindFramebuffer(GL_FRAMEBUFFER, 0);

//[..] draw full screen quad
```