Faculty of Science

Using textures with OpenGL 3.3

Alexander Christensen

Department of Computer Science
University of Copenhagen

2019

Overview

What is a Texture?

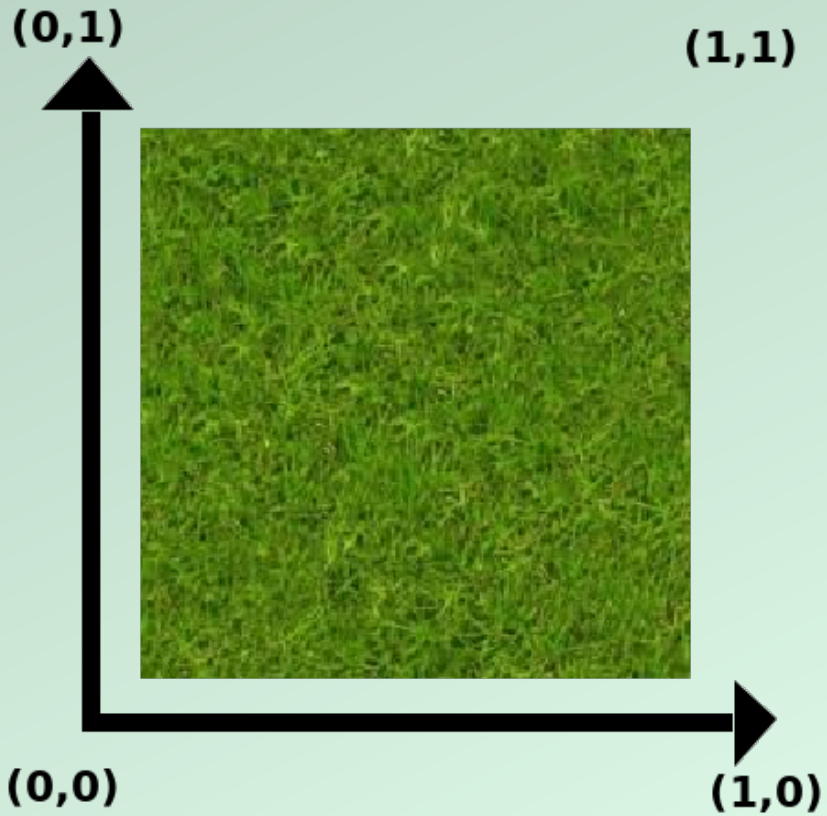
A texture is an image that has been loaded to the GPU as a consecutive block of memory.
OpenGL has no built-in functions for loading textures from the hard-disk.
Assume a 2D texture with four color channels (RGBA):

```
int width, height;
GLubyte* ourTextureData = someFunctionToReadFromDisk("ourImage.png",
    &width, &height);

GLuint ourTexture;
glGenTextures(1, ourTexture);
glBindTexture(GL_TEXTURE_2D, ourTexture);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, GL_RGBA,
    GL_UNSIGNED_BYTE, ourTextureData);
```

Texture Coordinates

We define a normalized coordinate system $T : \mathbb{R}^2$. A coordinate in a texture is called a *texel*, and is a discrete value. To get a texel given real coordinates (u, v) we can apply a filtering function $f(u, v) \ : \ \mathbb{R}^2 \mapsto \mathbb{N}^2$.



Texture Coordinates - Filtering

Define a function for interpolating between texture coordinates:

`GL_NEAREST` : choose nearest texel.

`GL_LINEAR` : linearly interpolate over neibhbouring texels.

Texture Coordinates - Clamping and Wrapping

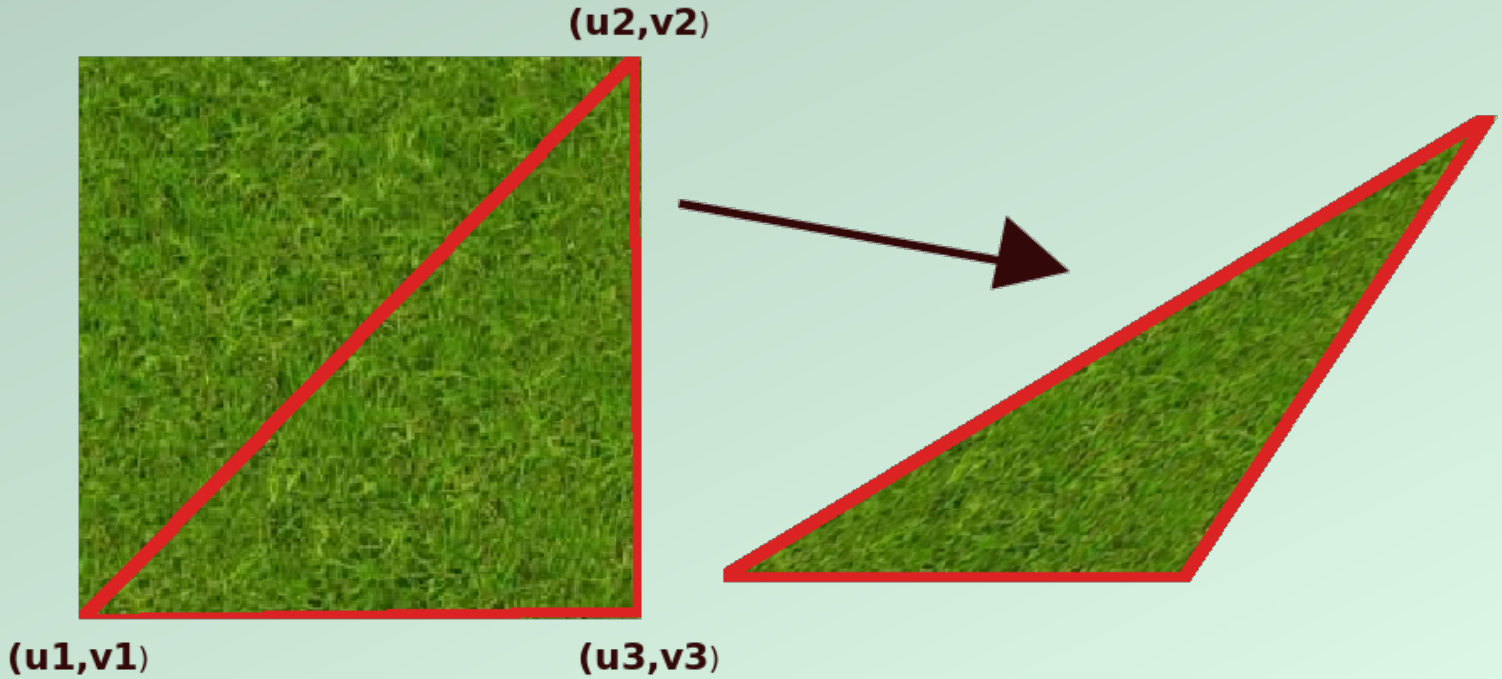
Define a function for interpolating between texture coordinates:

`GL_NEAREST` : choose nearest texel.

`GL_LINEAR` : linearly interpolate over neibhbouring texels.

Texture Mapping

Given a triangle with texture coordinates $(u1, v1)$, $(u2, v2)$, and $(u3, v3)$, we wish to define a mapping, such that each vertex gets its own texture coordinate.



Texture Mapping - Vertex Shader

The texture coordinate is buffered together with the position, as a per-vertex attribute:

```
#version 330 core

layout (location = 0) in vec2 vertexPos;
layout (location = 1) in vec2 texCoord;

out vec2 interpolatedTexCoord;

void main()
{
    gl_Position = vec4(vertexPos, 0.0f, 1.0f);
    interpolatedTexcoord = texCoord; // will be interpolated by OpenGL
}
```

Texture Mapping - Fragment Shader

Any input/output variable between vertex- and fragment shader is automatically interpolated by OpenGL.

```
#version 330 core

uniform sampler2D textureSampler;
in vec2 interpolatedTexCoord;
out vec4 color;

void main()
{
    color = texture(textureSampler, interpolatedTexcoord);
}
```

Example with 1 texture

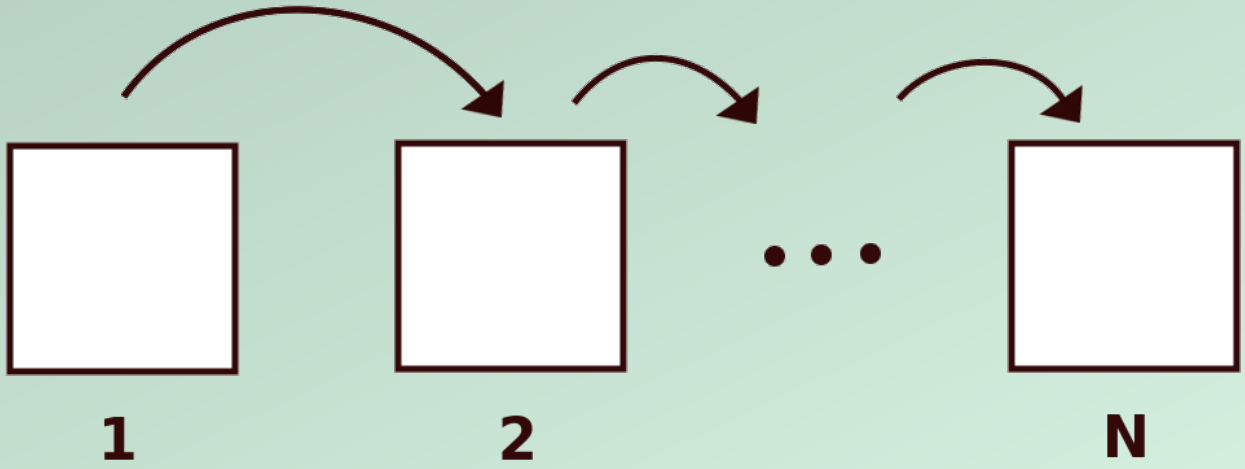
Draw a triangle with the grass texture that I have been showing so much (or something else!).

Example with 2 textures

buffer the mouse position as a uniform such that moving the mouse will capture the x-position and divide the screen vertically between the two textures. There can be a smoothening transition which would look very cool.
We could also perform this transition on the y-axis, but then a choice must be made: 2 textures or 4 textures!

The bigger perspective - Framebuffers

Textures are images loaded to GPU memory.
We are not restricted only to reading from textures, we an also write to them.



If we have $N-1$ textures, and let N represent the application window, then for $1 \leq i \leq N-1$ we can use the image data in texture i to run a shader which writes to texture $i+1$. The final image in texture N is rendered to the screen.

Framebuffer Example: Conway's Game of Life

Summary

We have seen the texture coordinate system.

We have seen methods for texture sampling.

Some code examples have shown how textures can be used in OpenGL 3.3.

References


Texture.
https://www.khronos.org/opengl/wiki/Texture.
Accessed: 2019-01-19.


Learnopengl.
https://learnopengl.com/Getting-started/Textures.
Accessed: 2019-01-19.


Learnopengl.
https://learnopengl.com/Advanced-OpenGL/Framebuffers.
Accessed: 2019-01-19.


Conway's game of life.
https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life.
Accessed: 2019-01-19.