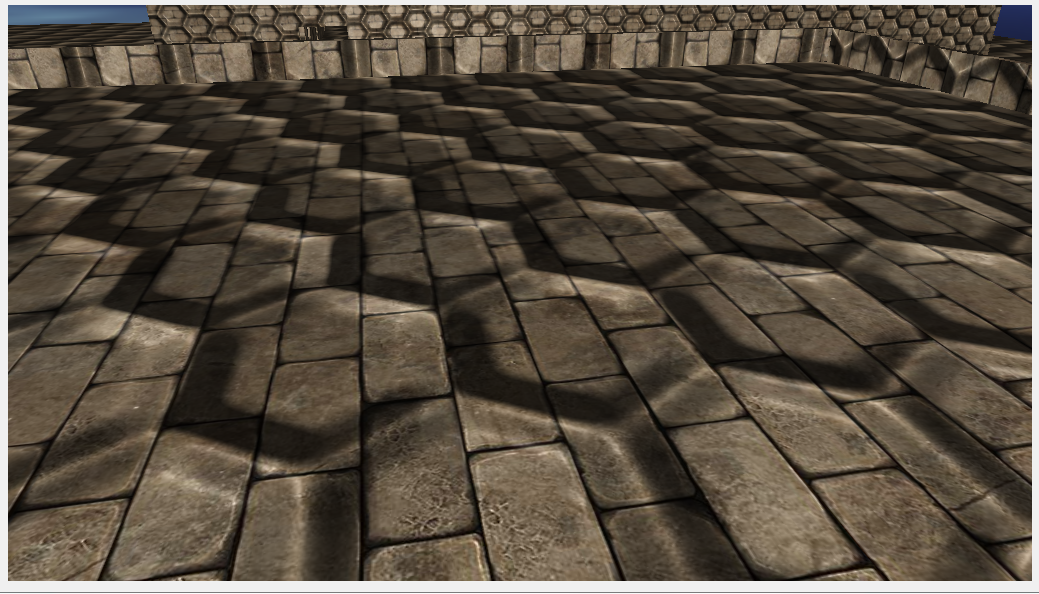
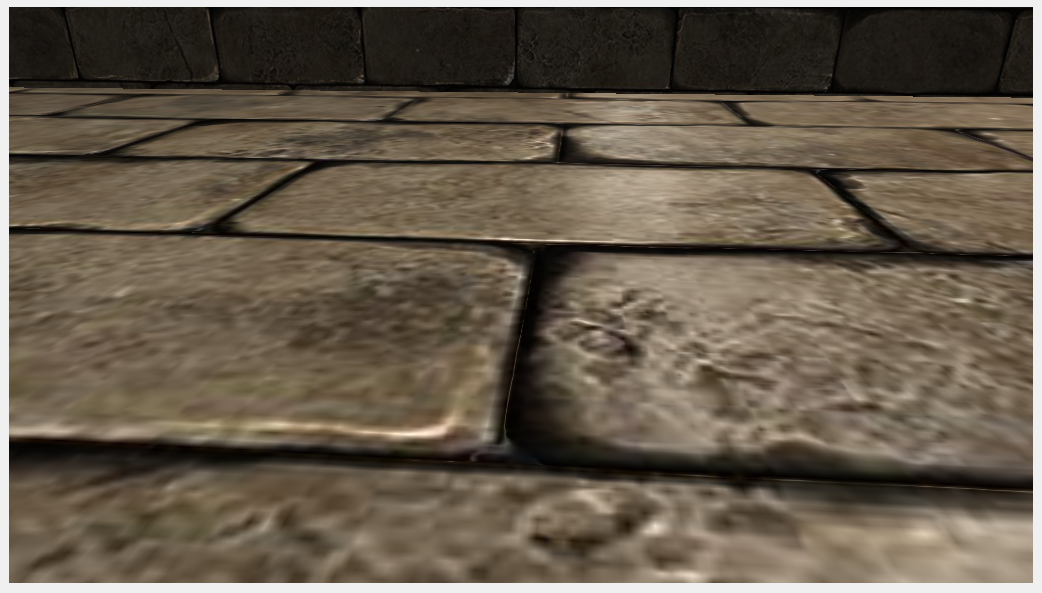
Lab 07 – Normal Mapping – Joshua Kauer

For this lab I did normal mapping. For this lab I had to add on to the .obj parser in order for the .scene files to contain tangents and bitangents. I then pass those values in as well as the position, texture, and normal into the shader. From there the only thing I had to change was add another texture possibility and then make sure I was binding correctly. After that I tested to see if I did everything right. Initially I had my calculations for the tangents and bitangets wrong but they were fixed. Next I then tried multiple maps to make sure it was working and it was.

Show that normal maps are working



Close up with a map that aligns with the texture to show they are working properly



//Vertex Shader

#version 430 core

layout (location = 0) in vec3 position;

layout (location = 1) in vec2 texCoords;

layout (location = 2) in vec3 normal;

layout (location = 3) in vec3 tangent;

layout (location = 4) in vec3 bitangent;

out VS\_OUT {

vec3 FragPos;

vec2 TexCoords;

vec3 TangentLightPos;

vec3 TangentViewPos;

vec3 TangentFragPos;

} vs\_out;

uniform mat4 projection;

uniform mat4 view;

uniform mat4 model;

uniform vec3 lightPos;

uniform vec3 viewPos;

void main()

{

gl\_Position = projection \* view \* model \* vec4(position, 1.0f);

vs\_out.FragPos = vec3(model \* vec4(position, 1.0));

vs\_out.TexCoords = texCoords;

mat3 normalMatrix = transpose(inverse(mat3(model)));

vec3 T = normalize(normalMatrix \* tangent);

vec3 B = normalize(normalMatrix \* bitangent);

vec3 N = normalize(normalMatrix \* normal);

mat3 TBN = transpose(mat3(T, B, N));

vs\_out.TangentLightPos = TBN \* lightPos;

vs\_out.TangentViewPos = TBN \* viewPos;

vs\_out.TangentFragPos = TBN \* vs\_out.FragPos;

}

//Fragment Shader

#version 430 core

out vec4 FragColor;

in VS\_OUT {

vec3 FragPos;

vec2 TexCoords;

vec3 TangentLightPos;

vec3 TangentViewPos;

vec3 TangentFragPos;

} fs\_in;

uniform sampler2D diffuseMap;

uniform sampler2D normalMap;

void main()

{

// Obtain normal from normal map in range [0,1]

vec3 normal = texture(normalMap, fs\_in.TexCoords).rgb;

// Transform normal vector to range [-1,1]

normal = normalize(normal \* 2.0 - 1.0); // this normal is in tangent space

// Get diffuse color

vec3 color = texture(diffuseMap, fs\_in.TexCoords).rgb;

// Ambient

vec3 ambient = 0.3 \* color;

// Diffuse

vec3 lightDir = normalize(fs\_in.TangentLightPos - fs\_in.TangentFragPos);

float diff = max(dot(lightDir, normal), 0.0);

vec3 diffuse = diff \* color;

// Specular

vec3 viewDir = normalize(fs\_in.TangentViewPos - fs\_in.TangentFragPos);

vec3 reflectDir = reflect(-lightDir, normal);

vec3 halfwayDir = normalize(lightDir + viewDir);

float spec = pow(max(dot(normal, halfwayDir), 0.0), 32.0);

vec3 specular = vec3(0.2) \* spec;

FragColor = vec4(ambient + diffuse + specular, 1.0f);

}