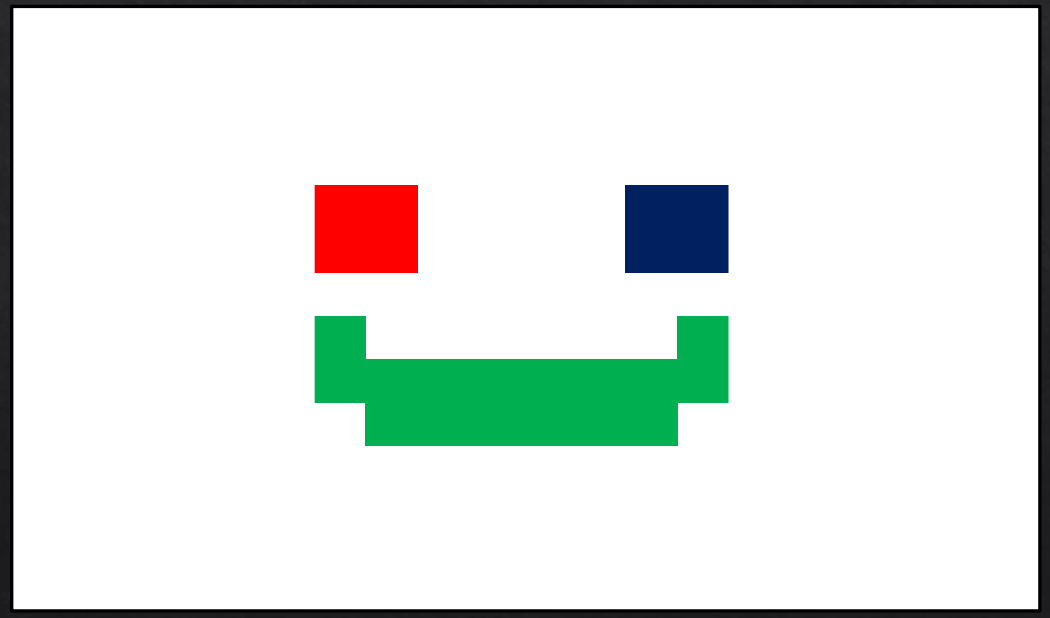
셰이더프로그래밍

Lecture 7

이택희

지난시간

- 텍스처



개요

- 다중 텍스처 사용
- 실습
- 단일 텍스처 사용 애니메이션
- 실습

다중 텍스처 사용

다중 텍스처 사용

- 지금까지는 Texture 하나만 사용
  - Default 값이 0 으로 지정되어 있기 때문에 따로 지정을 해 주지 않아도 동작함

Fragment Shader

```
#version 330

in vec2 vTexPos;

out vec4 FragColor;

uniform sampler2D uTexture;

void main()
{
   FragColor = texture(uTexture, vTexPos);
}
```

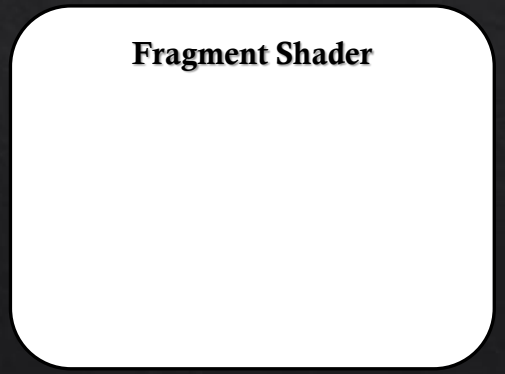
```
int uniformTex = glGetUniformLocation(gShaderProgram, "uTexSampler");

glUniform1i(uniformTex, 0);

glActiveTexture(GL_TEXTURE0);

glBindTexture(GL_TEXTURE_2D, gTextureID);
```

ActiveTexture 가 없어도 동작함

다중 텍스처 사용

- Texture 가 여러 장일 경우 glActiveTexture 함수를 통해 사용할 Texture 의 번호를 지정해 주어야 함

gTextureID → GL_TEXTURE0
gTextureID1 → GL_TEXTURE1
gTextureID2 → GL_TEXTURE2
gTextureID3 → GL_TEXTURE3
…



다중 텍스처 사용

gTextureID → GL_TEXTURE0 지정 방법

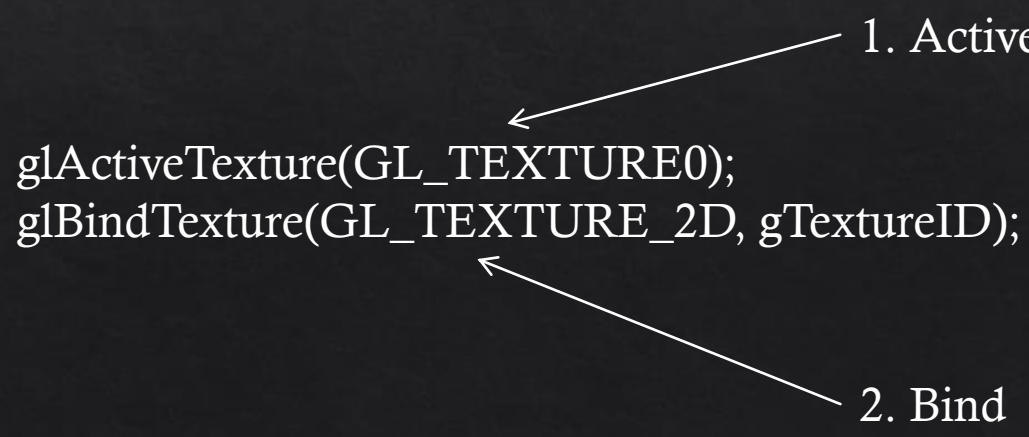
```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, gTextureID);
```

gTextureID → GL_TEXTURE1 지정 방법

```
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_2D, gTextureID1);
```

다중 텍스처 사용

- 즉, GL_TEXTURE_2D 는 설정 가능한 텍스처가 최소 80개가 있음
- 각 텍스처를 설정하기 위해 Bind 전에 Active 시키는 과정이 필요함



다중 텍스처 사용


GL_TEXTURE0
GL_TEXTURE1
GL_TEXTURE2
GL_TEXTURE3
GL_TEXTURE4
GL_TEXTURE5
GL_TEXTURE6
…
GL_TEXTURE_2D
…
…
…
…
…
…
…
…
GL_TEXTURE31
glActiveTexture(GL_TEXTURE0)
glActiveTexture(GL_TEXTURE1)
glActiveTexture(GL_TEXTURE2)
glActiveTexture(GL_TEXTURE3)
glActiveTexture(GL_TEXTURE4)
glActiveTexture(GL_TEXTURE4)
glBindTexture(GL_TEXTURE_2D, gTextureID);
glBindTexture(GL_TEXTURE_2D, gTextureID1);
glBindTexture(GL_TEXTURE_2D, gTextureID2);
glBindTexture(GL_TEXTURE_2D, gTextureID3);
glBindTexture(GL_TEXTURE_2D, gTextureID4);
glBindTexture(GL_TEXTURE_2D, gTextureID5);

다중 텍스처 사용

GL_TEXTURE0 : gTextureID
GL_TEXTURE1 : gTextureID1
GL_TEXTURE2 : gTextureID2
GL_TEXTURE3 : gTextureID3
GL_TEXTURE4 : gTextureID4
GL_TEXTURE5 : gTextureID5
GL_TEXTURE6 : nothing
…

GL_TEXTURE_2D …

…
…
…
…
…
…
…
…
GL_TEXTURE31 : nothing

다중 텍스처 사용(실습 준비)

Vertex Shader

```
#version 330

in vec3 Position;
in vec2 TexPos;

out vec2 vTexPos;

void main()
{
    gl_Position = vec4(Position, 1.0);
    vTexPos = TexPos;
}
```

Fragment Shader

```
#version 330

in vec2 vTexPos;

out vec4 FragColor;

uniform sampler2D uTexSampler;

void main()
{
    FragColor = texture(uTexSampler, vTexPos);
}
```

다중 텍스처 사용(실습 준비)

```
float vertPosTex[30] =
{
-0.5f, 0.5f, 0.0f, 0.0f, 1.0f, -0.5f, -0.5f, 0.0f, 0.0f, 0.0f, 0.5f, 0.5f, 0.0f, 1.0f, 1.0f,
0.5f, 0.5f, 0.0f, 1.0f, 1.0f, -0.5f, -0.5f, 0.0f, 0.0f, 0.0f, 0.5f, -0.5f, 0.0f, 1.0f, 0.0f
};

glGenBuffers(1, &VBO_PosTex);
glBindBuffer(GL_ARRAY_BUFFER, VBO_PosTex);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertPosTex), vertPosTex,
GL_STATIC_DRAW);
```

다중 텍스처 사용(실습 준비)

```
GLulong textureSmile[]
=
{
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFFFFFFFF,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF
};
glGenTextures(1, &gTextureID);
glBindTexture(GL_TEXTURE_2D, gTextureID);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 8, 8, 0, GL_RGBA, GL_UNSIGNED_BYTE, textureSmile);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_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);

GLulong textureSmile1[]
=
{
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF
};
glGenTextures(1, &gTextureID1);
glBindTexture(GL_TEXTURE_2D, gTextureID1);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 8, 8, 0, GL_RGBA, GL_UNSIGNED_BYTE, textureSmile1);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_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);
```

다중 텍스처 사용(실습 준비)

```
GLulong textureSmile2[]
=
{
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00, 0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF
};
glGenTextures(1, &gTextureID2);
glBindTexture(GL_TEXTURE_2D, gTextureID2);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 8, 8, 0, GL_RGBA, GL_UNSIGNED_BYTE, textureSmile2);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_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);

GLulong textureSmile3[]
=
{
0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFFFFFFFF, 0xFF00FF00, 0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00, 0xFF00FF00, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF
};
glGenTextures(1, &gTextureID3);
glBindTexture(GL_TEXTURE_2D, gTextureID3);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 8, 8, 0, GL_RGBA, GL_UNSIGNED_BYTE, textureSmile3);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_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);
```

다중 텍스처 사용(실습 준비)

```
GLulong textureSmile4[]
=
{
0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFFFFFFFF, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF
};
glGenTextures(1, &gTextureID4);
glBindTexture(GL_TEXTURE_2D, gTextureID4);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 8, 8, 0, GL_RGBA, GL_UNSIGNED_BYTE, textureSmile4);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_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);

GLulong textureSmile5[]
=
{
0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00,
0xFF00FF00, 0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00, 0xFF00FF00,
0xFFFFFFFF, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF
};
glGenTextures(1, &gTextureID5);
glBindTexture(GL_TEXTURE_2D, gTextureID5);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 8, 8, 0, GL_RGBA, GL_UNSIGNED_BYTE, textureSmile5);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_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);
```

다중 텍스처 사용(실습 준비)

```
glUseProgram(gShaderProgram);

glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, gTextureID);

glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_2D, gTextureID1);

glActiveTexture(GL_TEXTURE2);
glBindTexture(GL_TEXTURE_2D, gTextureID2);

glActiveTexture(GL_TEXTURE3);
glBindTexture(GL_TEXTURE_2D, gTextureID3);

glActiveTexture(GL_TEXTURE4);
glBindTexture(GL_TEXTURE_2D, gTextureID4);

glActiveTexture(GL_TEXTURE5);
glBindTexture(GL_TEXTURE_2D, gTextureID5);
```

다중 텍스처 사용(실습 준비)

```
int uniformTex = glGetUniformLocation(gShaderProgram, "uTexSampler");
glUniform1i(uniformTex, 0);
```

다중 텍스처 사용(실습 준비)

```
int attrribPosition = glGetAttribLocation(gShaderProgram, "Position");
int attrribTexPos = glGetAttribLocation(gShaderProgram, "TexPos");

glEnableVertexAttribArray(attrribPosition);
glEnableVertexAttribArray(attrribTexPos);

glBindBuffer(GL_ARRAY_BUFFER, VBO_PosTex);
glVertexAttribPointer(attrribPosition, 3, GL_FLOAT, GL_FALSE, 5 * sizeof(float), 0);
glVertexAttribPointer(attrribTexPos, 2, GL_FLOAT, GL_FALSE, 5 * sizeof(float), (GLvoid*)(3 * sizeof(float)));
```

다중 텍스처 사용(실습 준비)

```
glDrawArrays(GL_TRIANGLES, 0, 6);
```

실습

- 내용 구현
- uniform 값(sampler2D) 을 지속적으로 변경하여 gTextureID0 ~ gTextureID5 까지 애니메이션 되도록 구현
  - 너무 빠르게 바뀐다면 Sleep(1000); 을 넣어서 천천히 확인할 수 있음

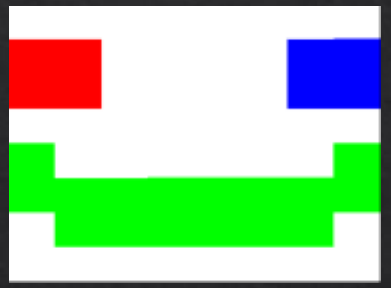
단일 텍스처 사용 애니메이션

단일 텍스처 사용 애니메이션

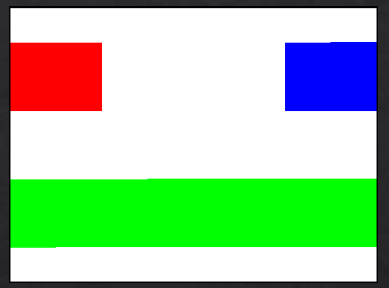
- 다중 텍스처를 사용하여 애니메이션 구현
  - 텍스처 Switch 가 발생하여 효율성이 떨어짐
  - 캐시 효율성이 떨어짐
- 하나의 텍스처에 여러장의 텍스처를 합쳐서 그리는 방식
  - 텍스처 Switch 발생이 없음
  - 캐시 효율성 높음

단일 텍스처 사용 애니메이션

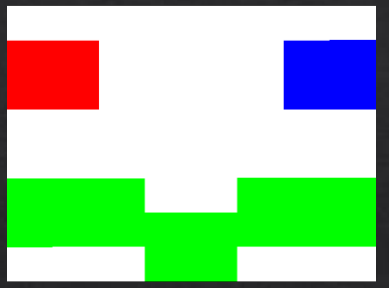
0



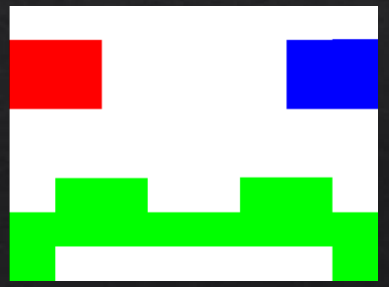
1



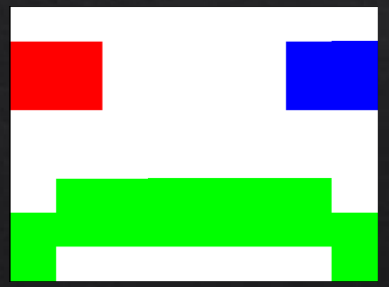
2



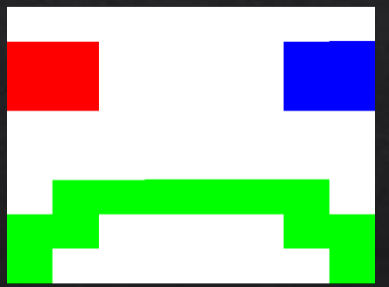
3



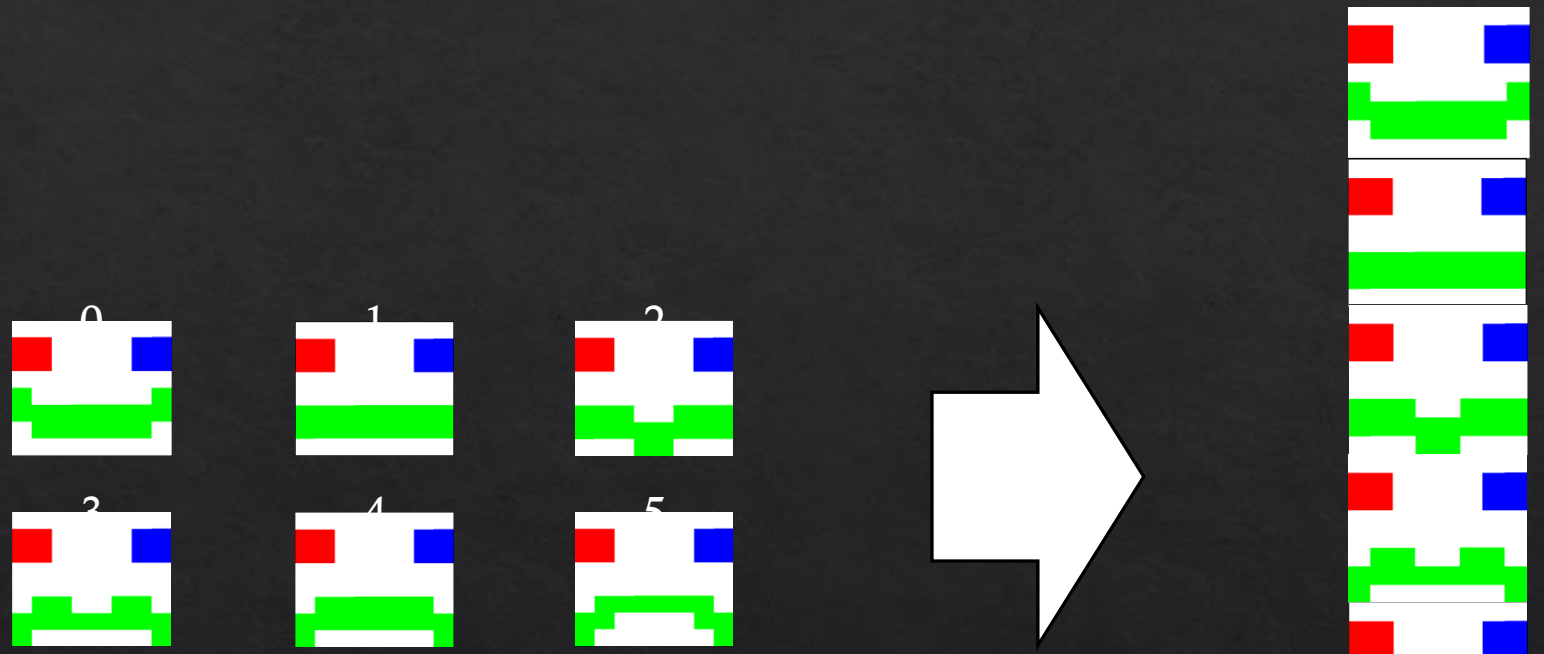
4



5



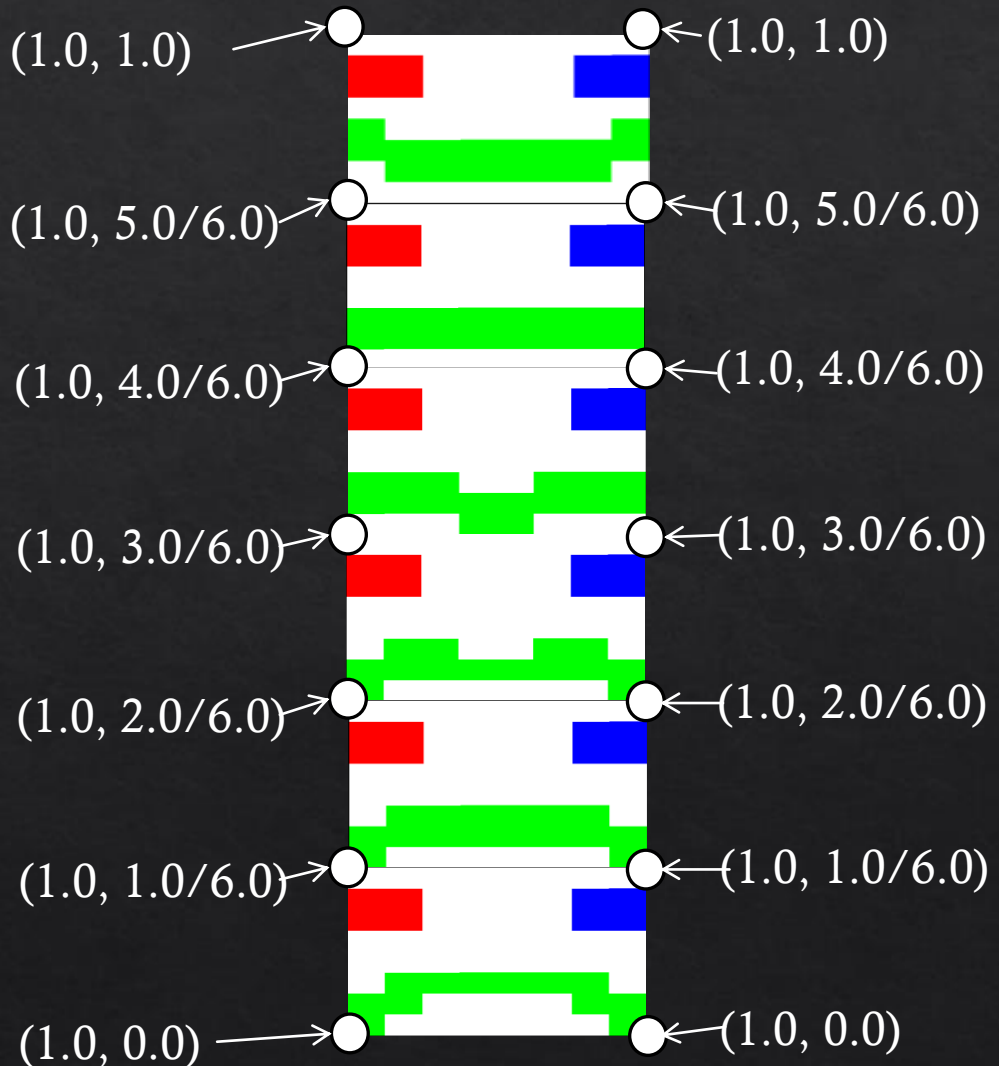
단일 텍스처 사용 애니메이션



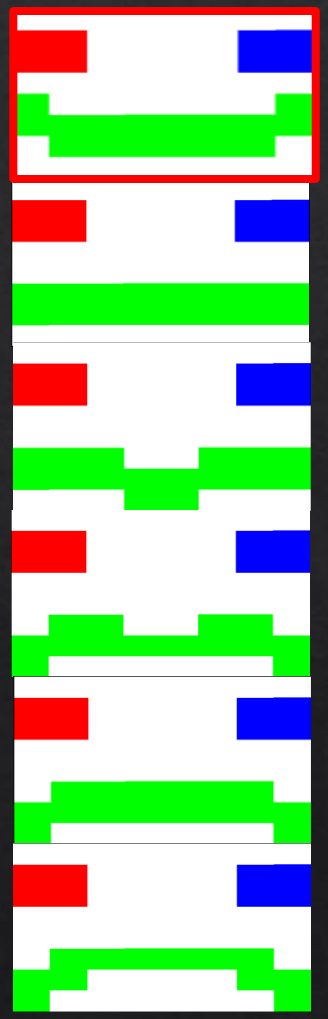
8X8 6개

8X48 1개

단일 텍스처 사용 애니메이션



단일 텍스처 사용 애니메이션



Fragment Shader

```
#version 330

in vec2 vTexPos;

out vec4 FragColor;

uniform float uTime;

uniform sampler2D uTexSampler;

void main()
{
    vec2 newTexPos = vec2(vTexPos.x, uTime/6.0 + vTexPos.y/6.0);
    FragColor = texture(uTexSampler, newTexPos);
}
```

uTime 을 0~5.f 사이로 넘김

단일 텍스처 사용(실습 준비)

Vertex Shader

```
#version 330

in vec3 Position;
in vec2 TexPos;

out vec2 vTexPos;

void main()
{
    gl_Position = vec4(Position, 1.0);
    vTexPos = TexPos;
}
```

Fragment Shader

```
#version 330

in vec2 vTexPos;

out vec4 FragColor;

uniform float uTime;

uniform sampler2D uTexSampler;

void main()
{
    vec2 newTexPos = vec2(vTexPos.x, uTime/6.0 + vTexPos.y/6.0);
    FragColor = texture(uTexSampler, newTexPos);
}
```

단일 텍스처 사용(실습 준비)

```
float vertPosTex[30] =
{
-0.5f, 0.5f, 0.0f, 0.0f, 1.0f, -0.5f, -0.5f, 0.0f, 0.0f, 0.0f, 0.5f, 0.5f, 0.0f, 1.0f, 1.0f,
0.5f, 0.5f, 0.0f, 1.0f, 1.0f, -0.5f, -0.5f, 0.0f, 0.0f, 0.0f, 0.5f, -0.5f, 0.0f, 1.0f, 0.0f
};

glGenBuffers(1, &VBO_PosTex);
glBindBuffer(GL_ARRAY_BUFFER, VBO_PosTex);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertPosTex), vertPosTex,
GL_STATIC_DRAW);
```

다일 텍스처 사용(실습 준비)

```
GLulong textureSmileTotal[]
=
{
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFFFFFFFF,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,

0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,

0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00, 0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFF00FF00, 0xFF00FF00, 0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00, 0xFF00FF00, 0xFF00FF00,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFF0000FF, 0xFF0000FF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFF0000, 0xFFFF0000,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,

0xFF00FF00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFF00FF00,
```

단일 텍스처 사용(실습 준비)

```
glUseProgram(gShaderProgram);

glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, gTextureIDTotal);

int uniformTex = glGetUniformLocation(gShaderProgram, "uTexSampler");
glUniform1i(uniformTex, 0);

int uniformTime = glGetUniformLocation(gShaderProgram, "uTime");
glUniform1f(uniformTime, gTimeStamp);

gTimeStamp += 1.f;
if (gTimeStamp > 5.f)
	gTimeStamp = 0.f;
```

단일 텍스처 사용(실습 준비)

```
int attrribPosition = glGetAttribLocation(gShaderProgram, "Position");
int attrribTexPos = glGetAttribLocation(gShaderProgram, "TexPos");

glEnableVertexAttribArray(attrribPosition);
glEnableVertexAttribArray(attrribTexPos);

glBindBuffer(GL_ARRAY_BUFFER, VBO_PosTex);
glVertexAttribPointer(attrribPosition, 3, GL_FLOAT, GL_FALSE, 5 * sizeof(float), 0);
glVertexAttribPointer(attrribTexPos, 2, GL_FLOAT, GL_FALSE, 5 * sizeof(float), (GLvoid*)(3 * sizeof(float)));
```

단일 텍스처 사용(실습 준비)

```
glDrawArrays(GL_TRIANGLES, 0, 6);
```

실습

- 내용 구현