

# 셰이더 프로그래밍

Lecture 6

이택희

# 지난 시간

- ◇ 버텍스 셰이더 사용 애니메이션
- ◇ 프래그먼트 셰이더 사용 애니메이션

# 개요

- ◇ 텍스처 매핑
- ◇ 텍스처 생성
- ◇ 텍스처 사용
- ◇ 실습

# 텍스처 매핑

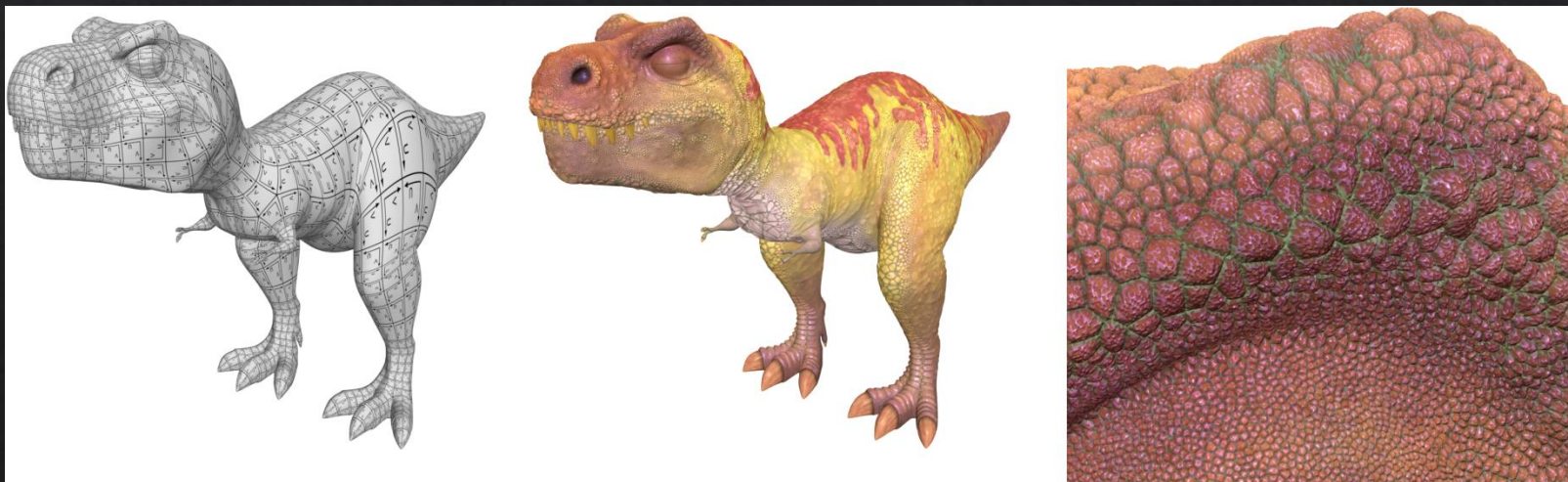
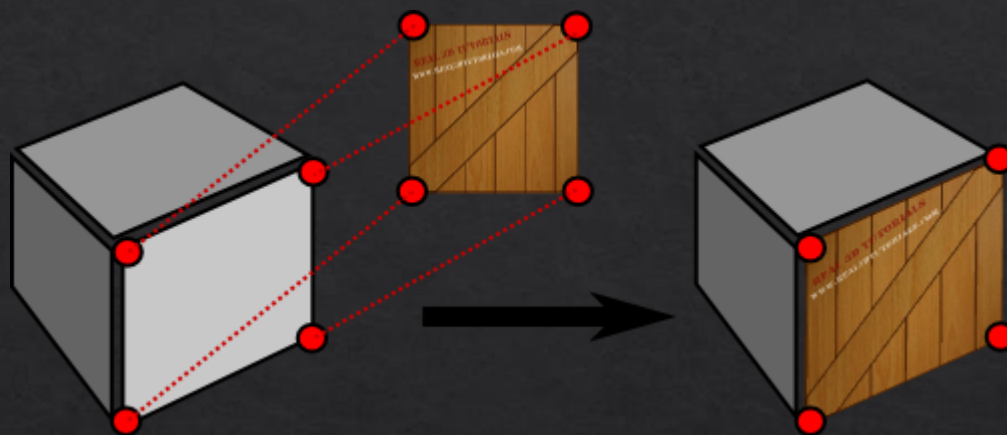
# 텍스처 매핑

## ◇ 텍스처란?

- ◇ 물체의 표면의 디테일을 표현하고자 할 때 모두 버텍스로 표현하기 위해선 매우 많은 데이터가 필요
- ◇ 해당 부분을 이미지로 대체 할 경우 간단하게 복잡한 표면을 표현 할 수 있음
- ◇ 이 이미지를 텍스처라 함

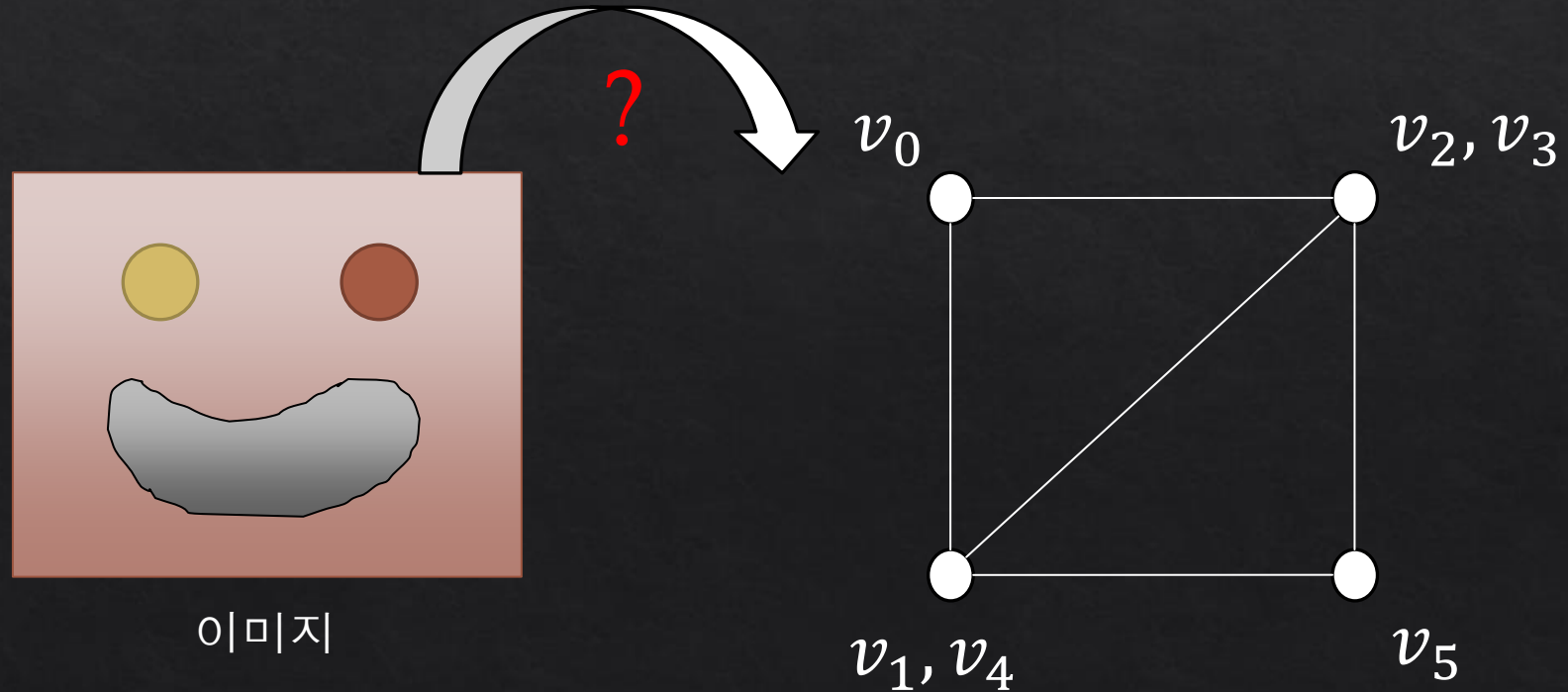


# 텍스처 매핑



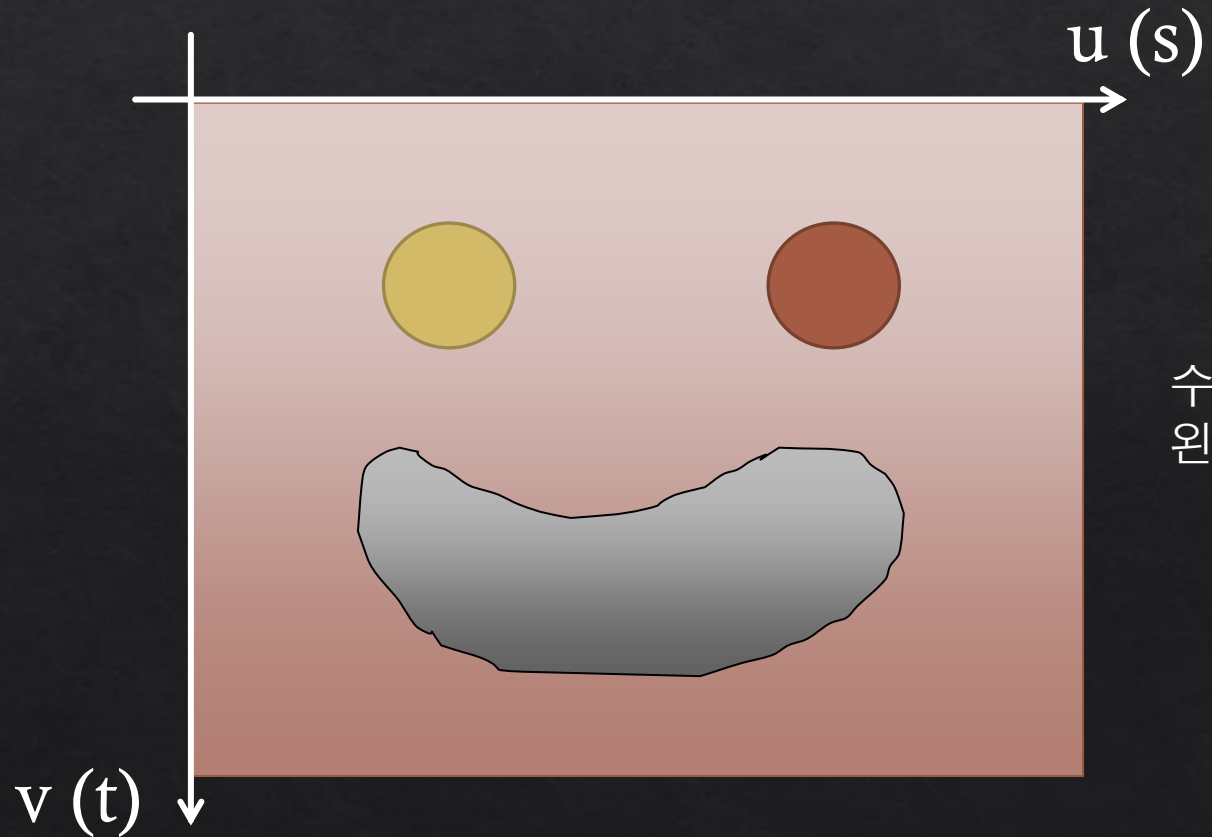
# 텍스처 매핑

- ◇ 필요한 데이터
  - ◇ 이미지 (jpg, bmp, png, tiff, ...)
  - ◇ 매핑 위치



# 텍스처 매핑

- ◆ 텍스처 상에서 사용되는 좌표계는  $u, v$  혹은  $s, t$  로 표현 됨



수업에선  $u, v$  의 축이  
왼쪽 그림과 같다



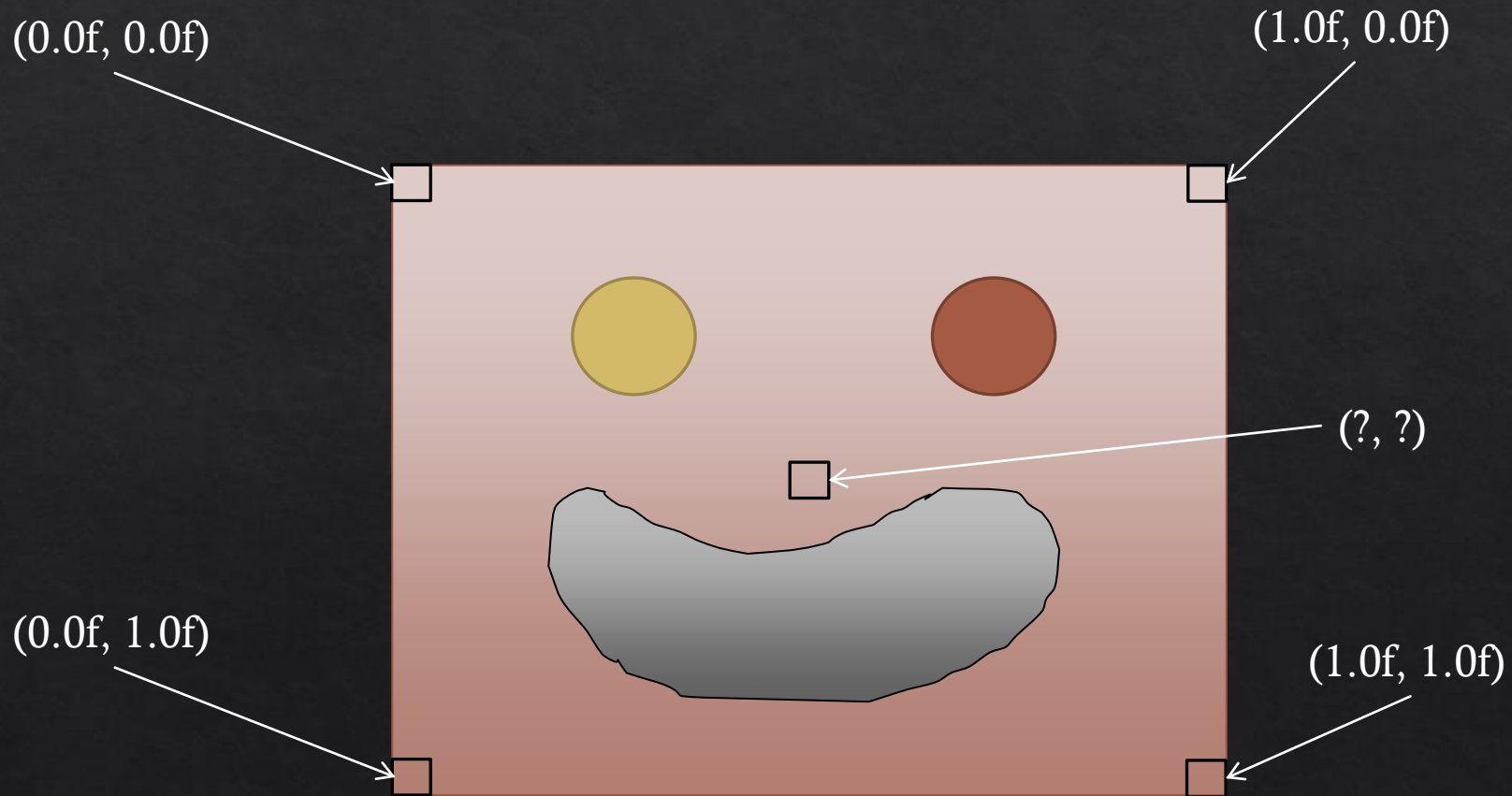
# 텍스처 매핑

◇  $u, v$  혹은  $s, t$  의 범위는  $0.0f \sim 1.0f$  임



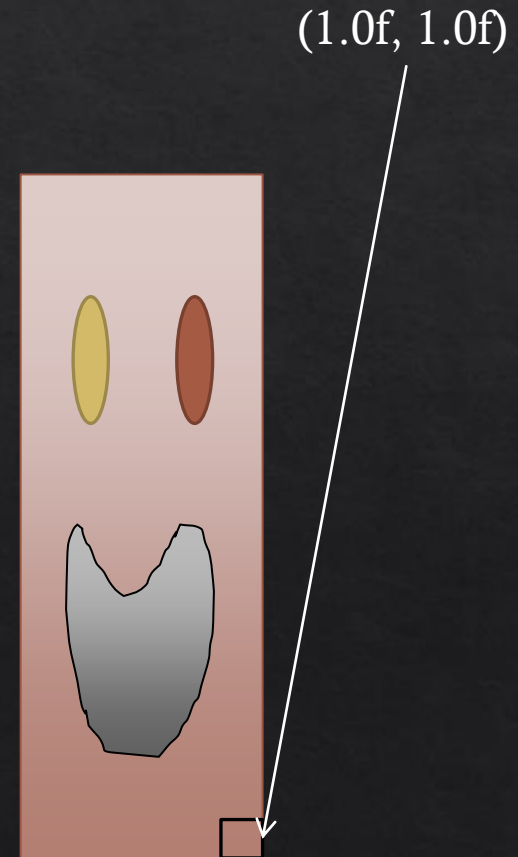
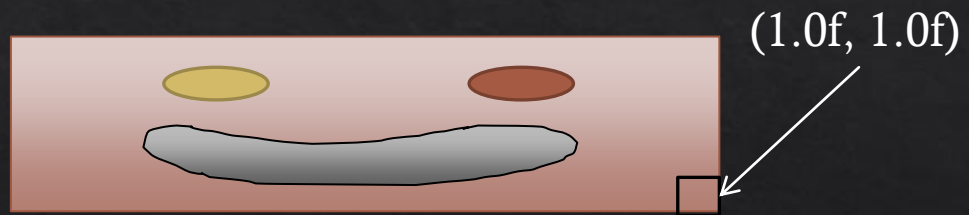
# 텍스처 매핑

◇  $u, v$  혹은  $s, t$  의 범위는  $0.0f \sim 1.0f$  임

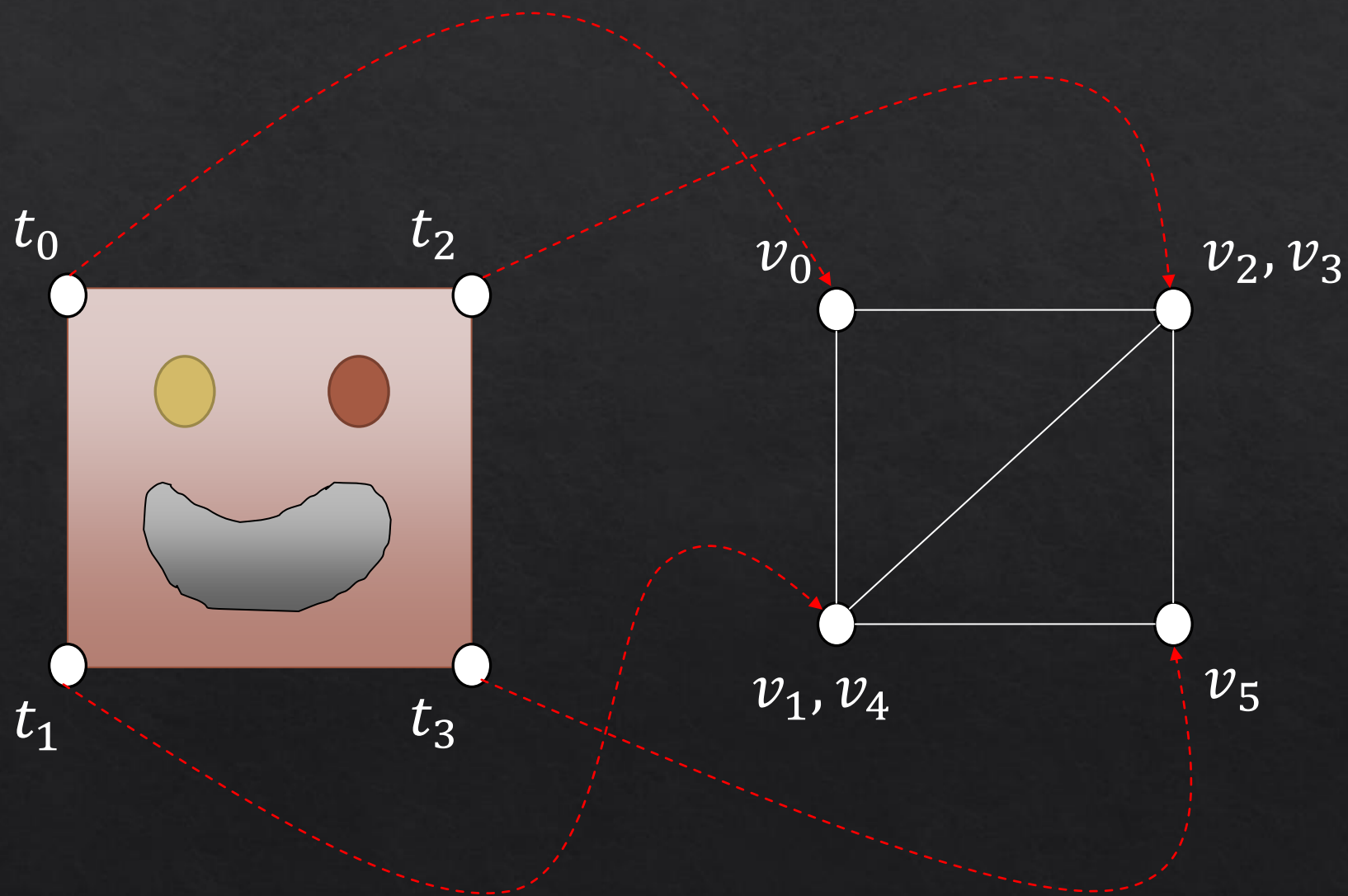


# 텍스처 매핑

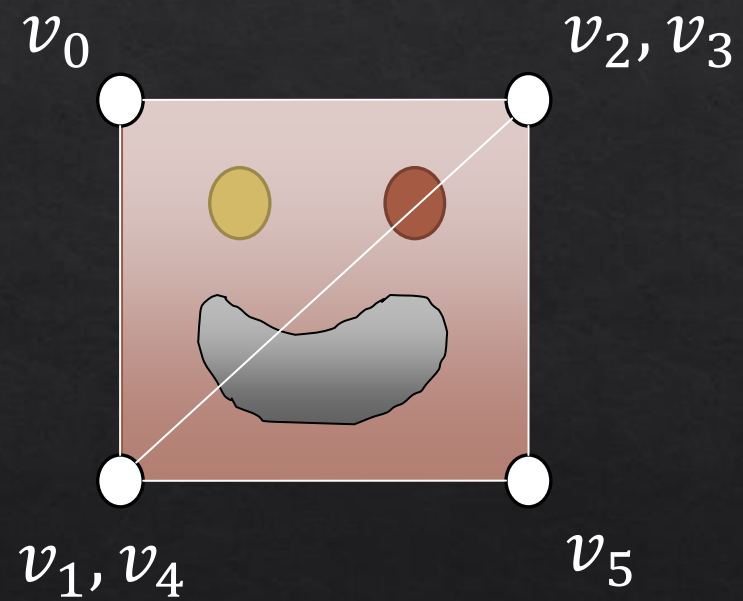
- ◇ 텍스처 좌표는  $0.0f \sim 1.0f$  사이로 normalize 됨
- ◇ 이미지 해상도와 별개임



# 텍스처 매핑



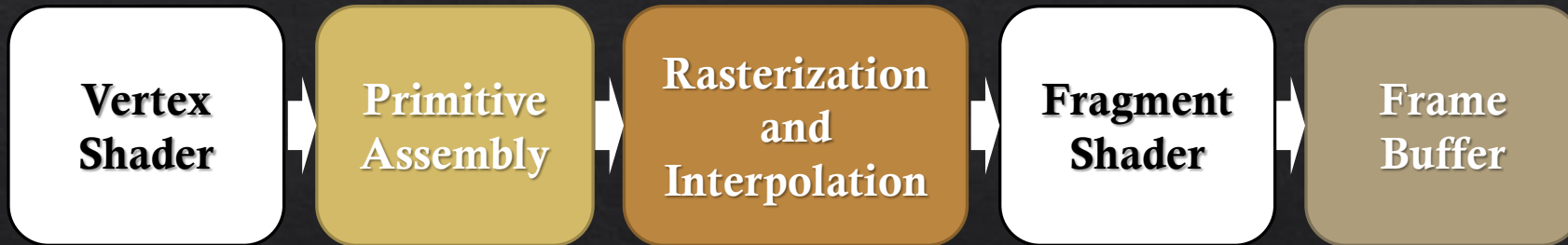
# 텍스처 매핑



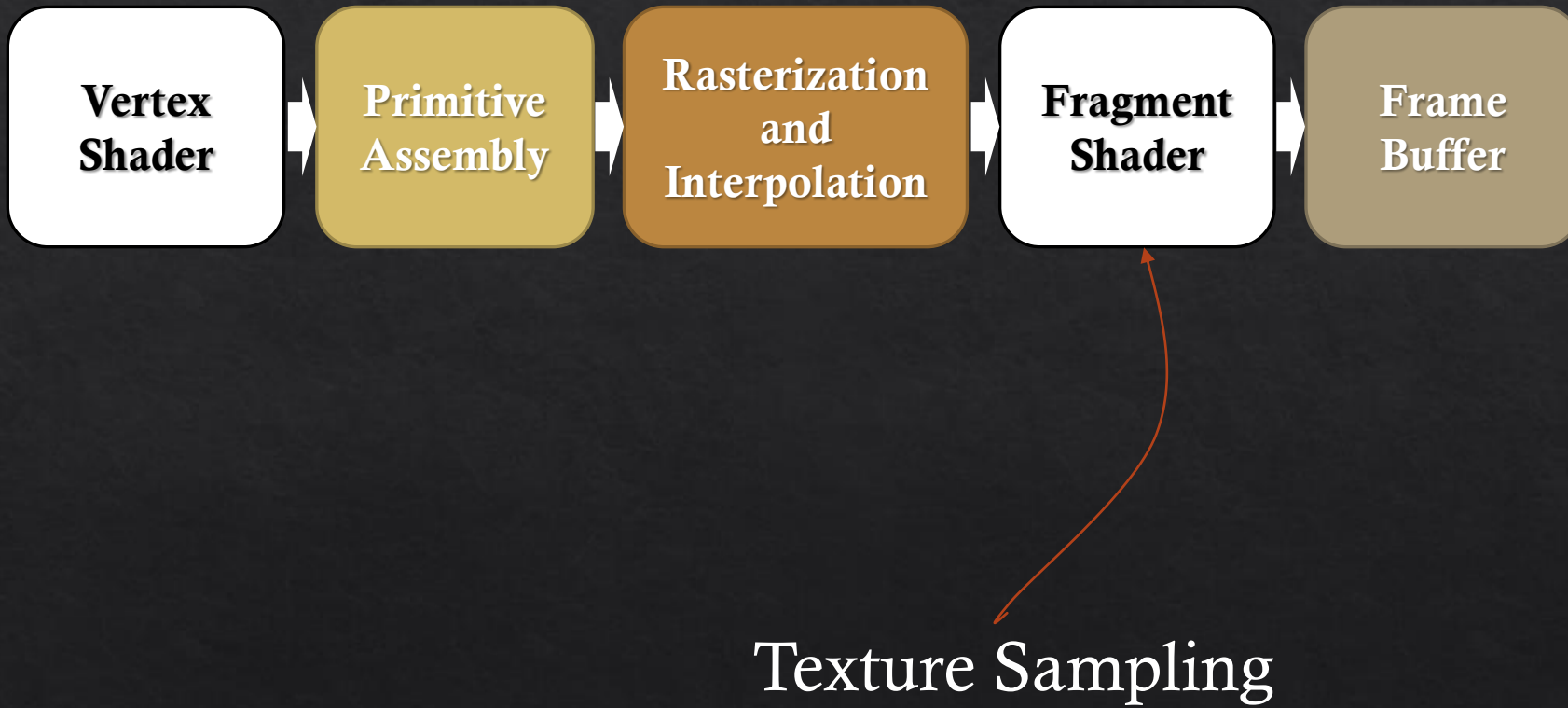


# 텍스처 매핑

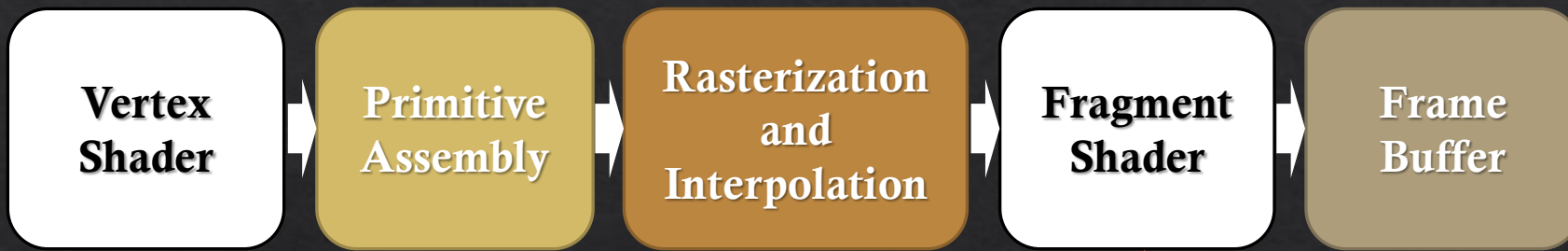
◇ 텍스처 샘플링은 어느 단계에서 이루어 질까?



# 텍스처 매핑



# 텍스처 매핑

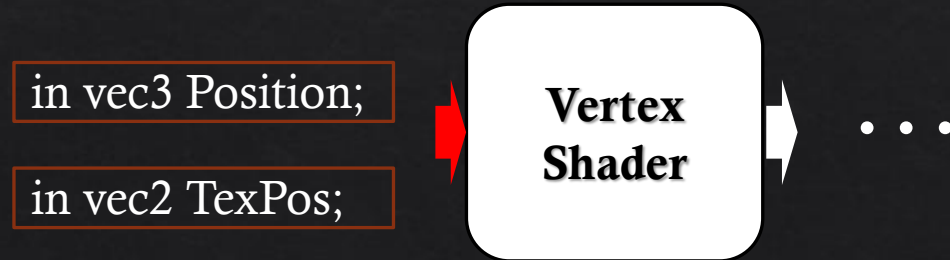


Sampling 을 위해선 텍스처 좌표가  
필요하고 좌표를 얻기 위해선  
버텍스 셰이더로 부터 받아야 함

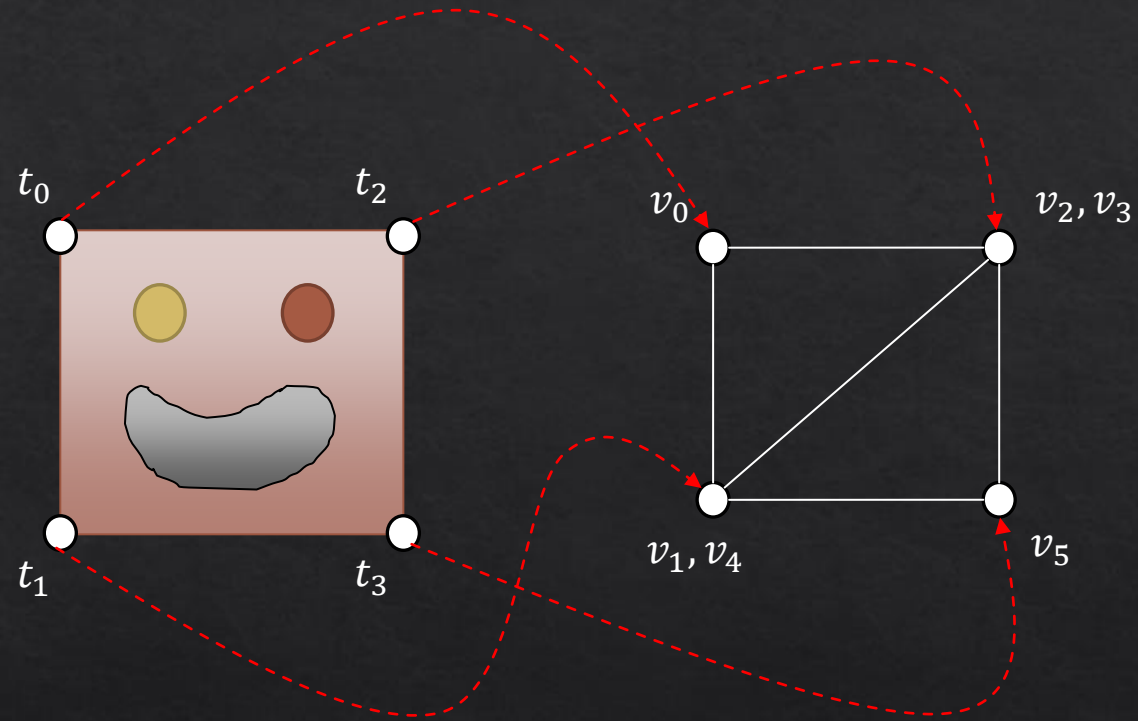
→ **Attribute**

Texture Sampling

# 텍스처 매핑



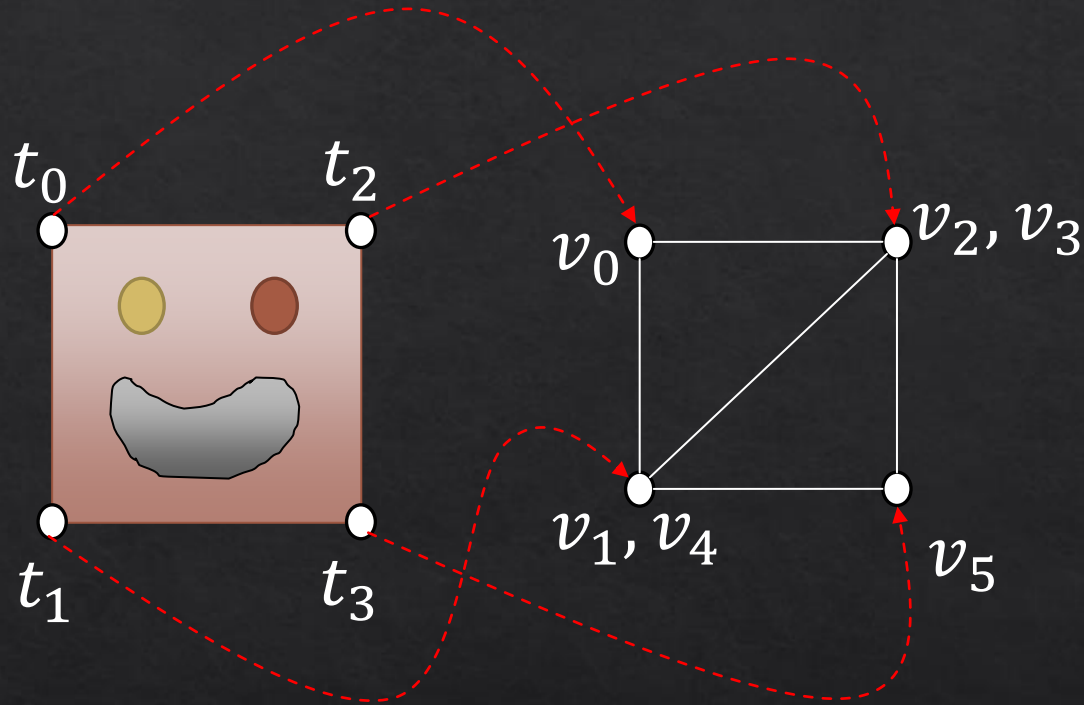
# 텍스처 매핑



버텍스당 텍스처 좌표 하나씩 필요함  
→ 버텍스 정보에 텍스처 좌표 추가 필요



# 텍스처 매핑



$$v_0 = (-0.5f, 0.5f, 0.0f)$$

$$v_1 = (-0.5f, -0.5f, 0.0f)$$

$$v_2 = (0.5f, 0.5f, 0.0f)$$

$$v_3 = (0.5f, 0.5f, 0.0f)$$

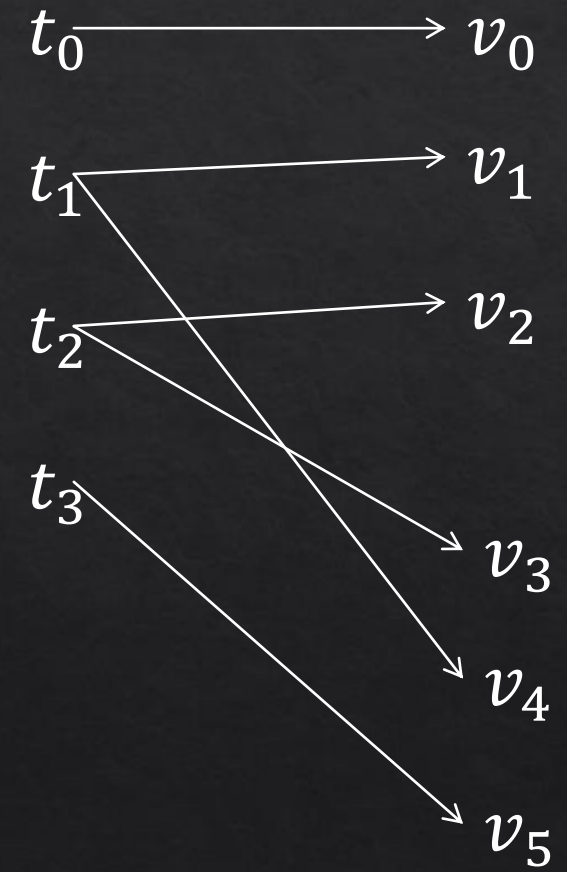
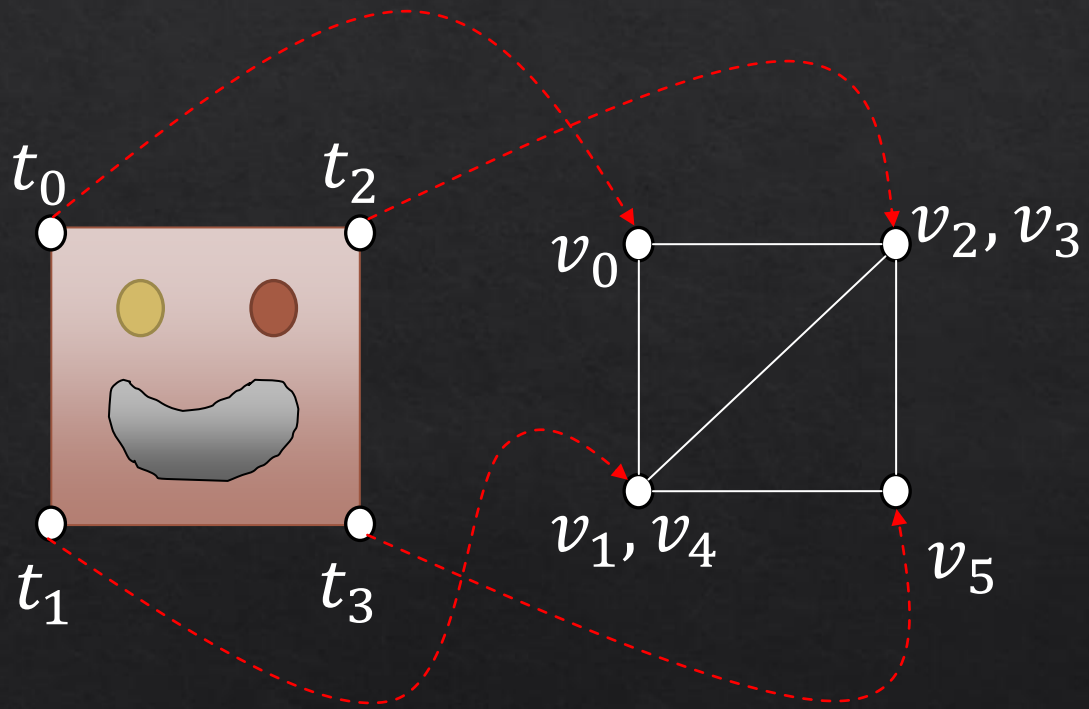
$$v_4 = (-0.5f, -0.5f, 0.0f)$$

$$v_5 = (0.5f, -0.5f, 0.0f)$$

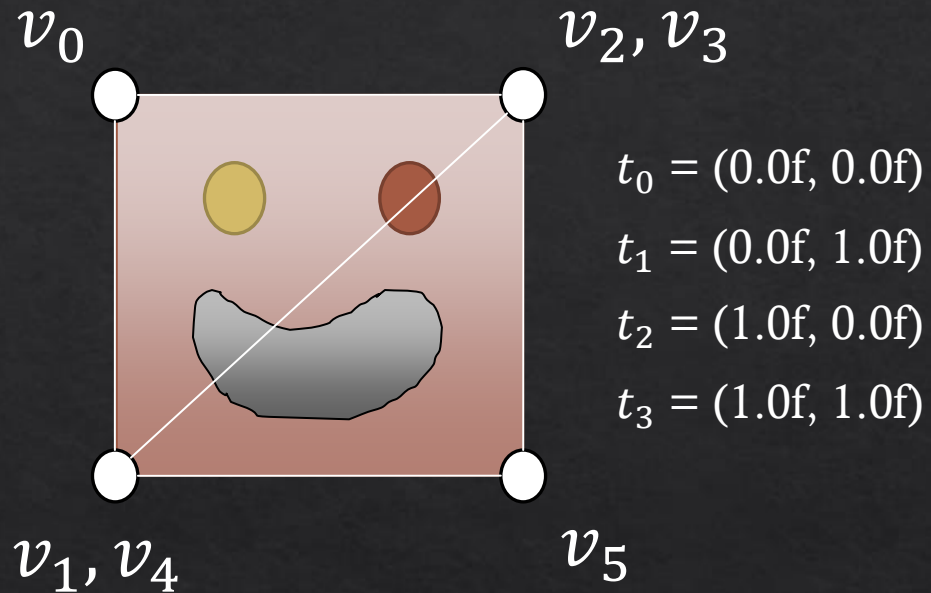
$$t_0 = (0.0f, 0.0f) \quad t_2 = (1.0f, 0.0f)$$

$$t_1 = (0.0f, 1.0f) \quad t_3 = (1.0f, 1.0f)$$

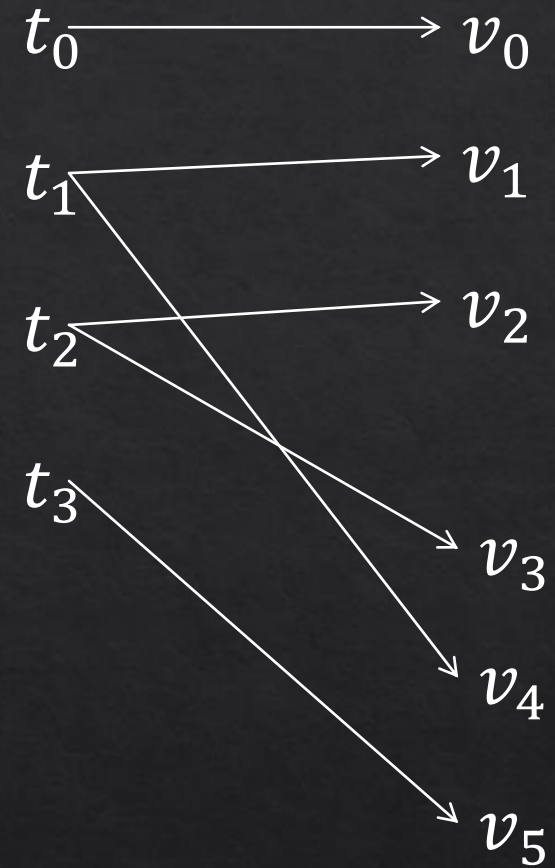
# 텍스처 매핑



# 텍스처 매핑

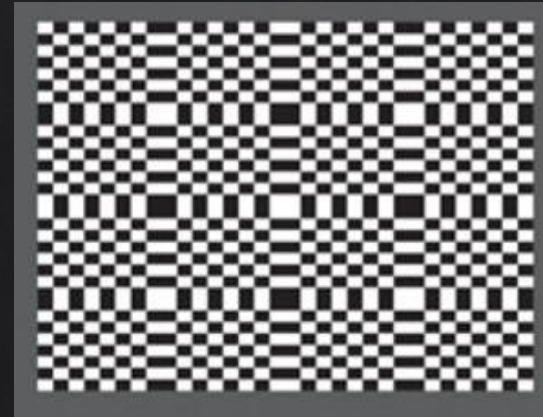


$$\begin{aligned}
 v_0 &= (-0.5f, 0.5f, 0.0f, \mathbf{0.0f}, \mathbf{0.0f}) \\
 v_1 &= (-0.5f, -0.5f, 0.0f, \mathbf{0.0f}, \mathbf{1.0f}) \\
 v_2 &= (0.5f, 0.5f, 0.0f, \mathbf{1.0f}, \mathbf{0.0f}) \\
 v_3 &= (0.5f, 0.5f, 0.0f, \mathbf{1.0f}, \mathbf{0.0f}) \\
 v_4 &= (-0.5f, -0.5f, 0.0f, \mathbf{0.0f}, \mathbf{1.0f}) \\
 v_5 &= (0.5f, -0.5f, 0.0f, \mathbf{1.0f}, \mathbf{1.0f})
 \end{aligned}$$



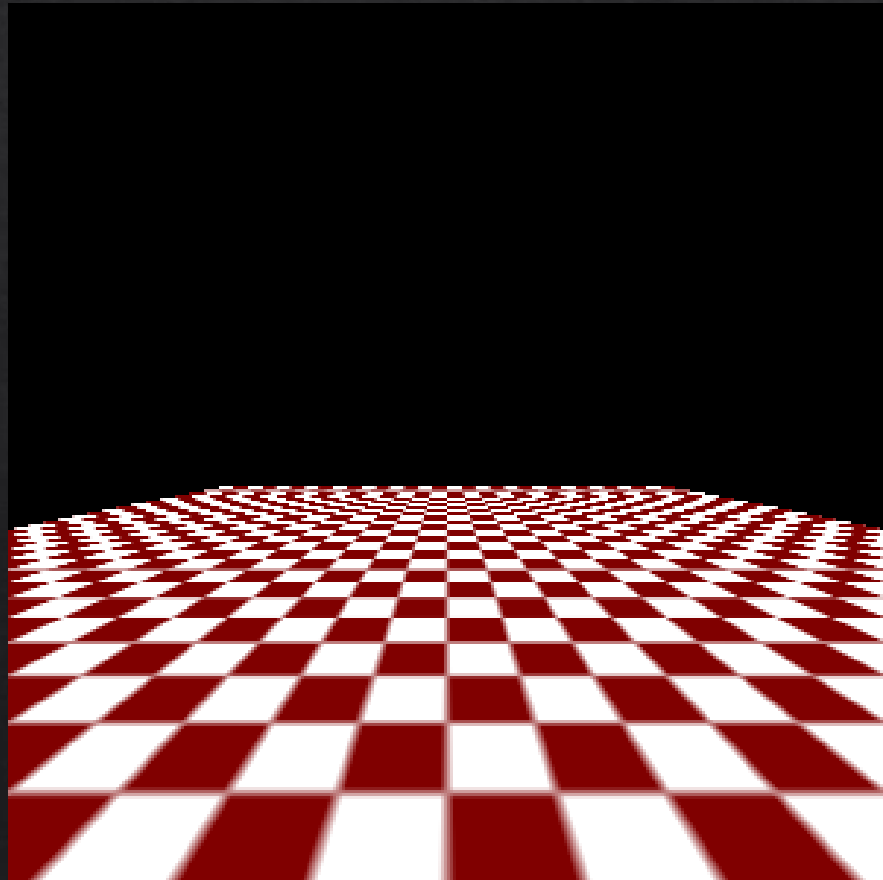
# 텍스처 매핑

- ◇  $u, v$  좌표가 0보다 작거나 1보다 클 경우
  - ◇ wrap 형식에 따라 채워짐



# 텍스처 매핑

## ◇ 텍스처 샘플링

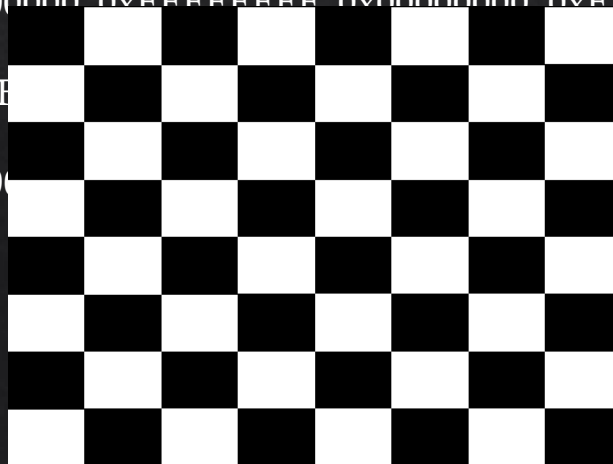




텍스처 생성

# 텍스처 생성

```
static const GLulong checkerboard[] =
{
0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF,
0x00000000,
0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000,
0xFFFFFFFF,
0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF,
0x00000000,
0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000,
0xFFFFFFFF,
0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF,
0x00000000,
0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000,
0xFFFFFFFF,
0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF,
0x00000000,
0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000,
0xFFFFFFFF
};
```



# 텍스처 생성

- ◇ 전체적인 순서는 VBO 생성하는 법과 비슷함
  - ◇ `glGenTextures(1, &gTextureID);`
  - ◇ `glBindTexture(GL_TEXTURE_2D, gTextureID);`
  - ◇ `glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 8, 8, 0, GL_RGBA, GL_UNSIGNED_BYTE, checkerboard);`

# 텍스처 생성

GL\_LINEAR

- ◇ `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);`

텍스처 사용



# 텍스처 사용

```
#version 330
```

```
in vec3 Position;  
in vec2 TexPos;
```

```
out vec2 vTexPos;
```

```
void main()
```

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

```
#version 330
```

```
uniform sampler2D uTexSampler;  
in vec2 vTexPos;
```

```
out vec4 FragColor;
```

```
void main()
```

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

```
int uniformTex = glGetUniformLocation(gShaderProgram, "uTexSampler");  
glUniform1i(uniformTex, 0);  
glActiveTexture(GL_TEXTURE0);  
glBindTexture(GL_TEXTURE_2D, gTextureID);
```

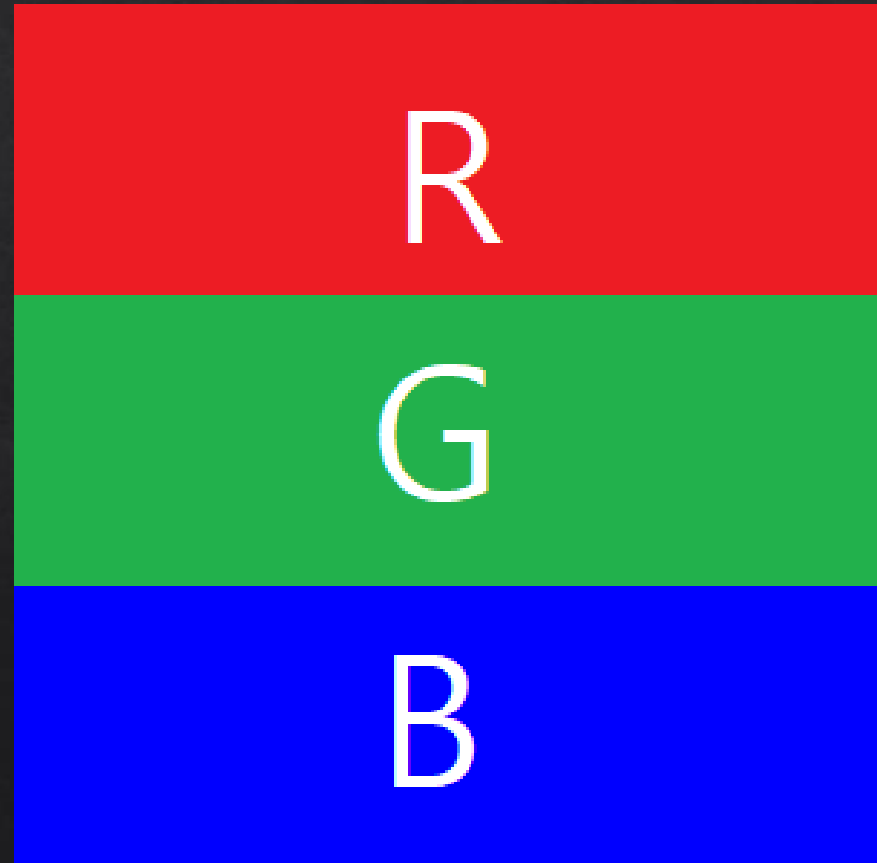




실습

# 실습

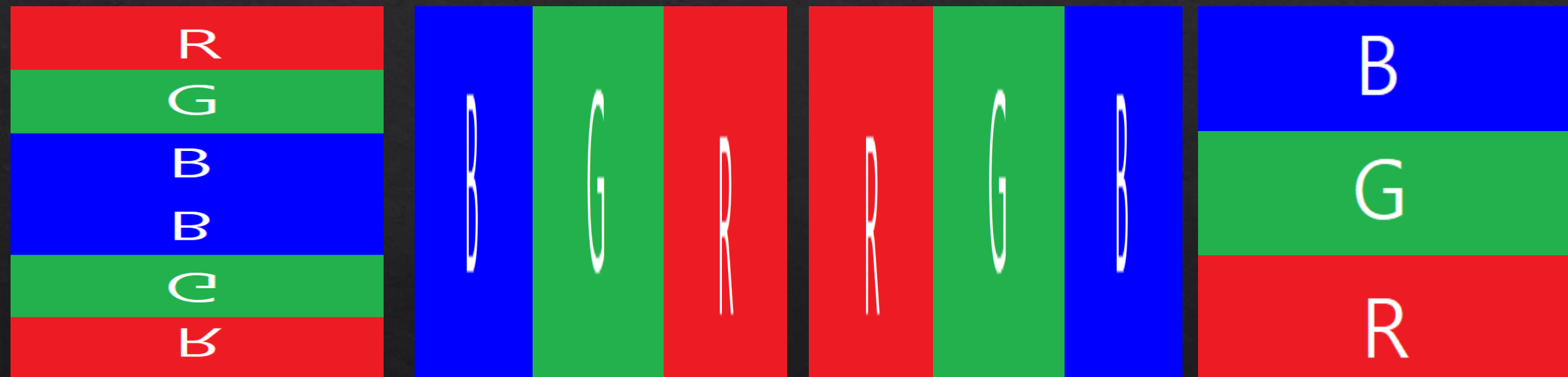
- ◆ 새로운 Shader 및 API 생성 후 구현
- ◆ 문제 : 블러 효과 구현 해보기
- ◆ 문제 : 좌표 꼬아 보기

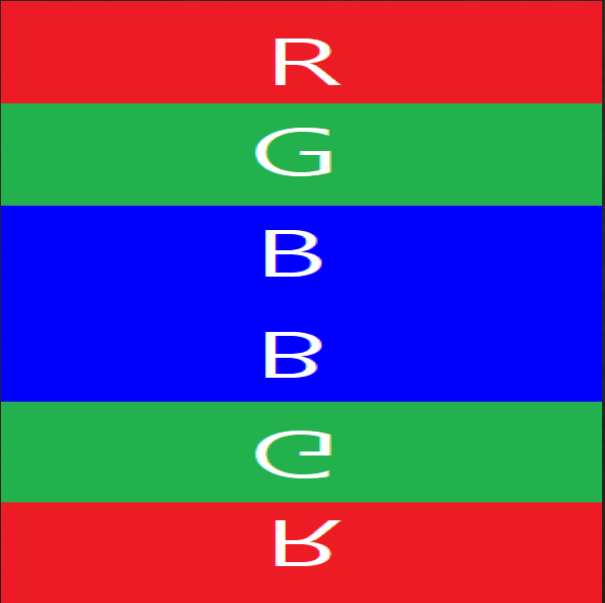
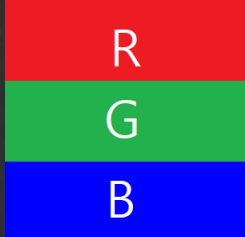




# 실습

문제 : 좌표 꼬아 보기





R

G

B

B

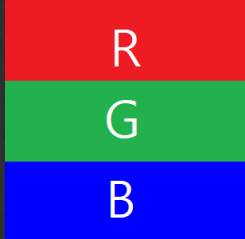
G

R

R
G
B

실습

R	R	R
G	G	G
B	B	B
R	R	
G	G	
B	B	



# 실습

R	G
G	B
B	R
R	G
G	B
B	R
	G