

# 컴퓨터 그래픽스

## OpenGL 텍스처 매핑

2017년 2학기

## 5. OpenGL 텍스처 매핑 내용

- 텍스처 매핑
  - Raster graphics
  - 비트맵 그리기
  - 문자 그리기
  - 픽스맵 그리기
  - Bmp 파일
  - 텍스처 매핑

# *OpenGL에서의 Raster Graphics*

- **OpenGL에서**
  - bitmap: 흑백 (0/1)으로 표현되는 이미지
  - pixmap: 각 픽셀이 색상을 갖는 컬러 이미지
- **비트맵 그리기**
  - 이미지 데이터를 배열로 정의한다.
  - 래스터 포지션 지정
    - 이미지의 좌측 하단을 기준으로 진행
    - `glRasterPos2i (GLint x, GLint y);`
    - `glRasterPos2f (GLfloat x, GLfloat y);`
    - `glRasterPos3i (GLint x, GLint y, GLint z);`
    - `glRasterPos3f (GLfloat x, GLfloat y, GLfloat z);`

# *OpenGL에서의 Raster Graphics*

## – 비트맵 그리기:

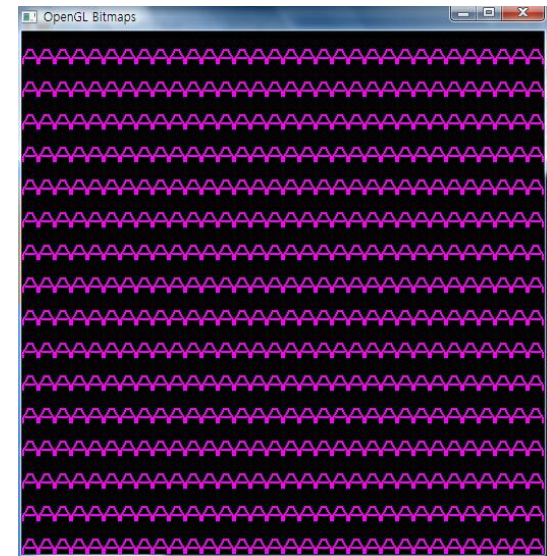
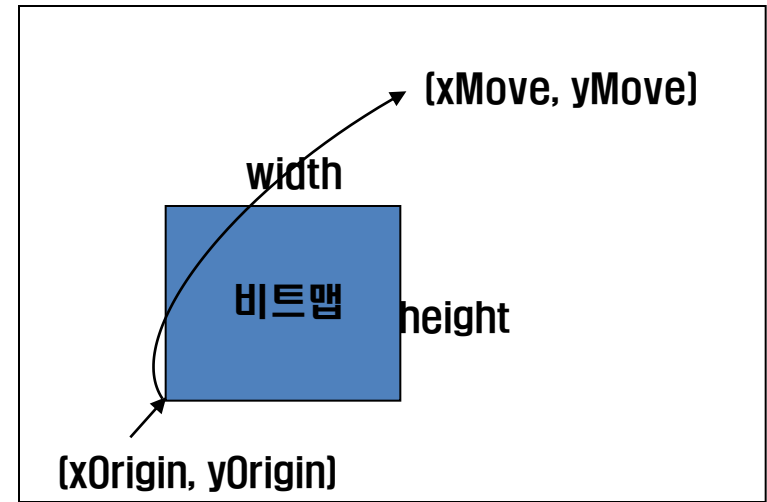
- glBitmap (Gsizei w, Gsizei h, GLfloat x\_orig, GLfloat y\_orig, GLfloat x\_move, GLfloat y\_move, const Glubyte \*bits)
  - w, h: 비트맵의 폭과 높이
  - x\_orig, y\_orig: 비트맵의 중심 위치
  - x\_move, y\_move: 비트맵이 그려진 수 현재의 래스터 위치에 더해질 x, y 오프셋 값
  - bits: 비트맵 이미지 주소
- 함수 호출 이전에 설정된 glColor 색상의 영향
- 아래에서 위 방향으로 그려진다.

# 비트맵 그리기

예제) 문자 A형태를 가지고 있는 16×16 비트맵을 20\*20번  
그리기

```
void drawScene (){
    int i, j;
    unsigned char letterA[] = {
        0xC0, 0x03, 0, 0, // 1100 0000 0000 0011
        0xC0, 0x03, 0, 0, // 1100 0000 0000 0011
        0xC0, 0x03, 0, 0, // 1100 0000 0000 0011
        0xC0, 0x03, 0, 0, // 1100 0000 0000 0011
        0xC0, 0x03, 0, 0, // 1100 0000 0000 0011
        0xDF, 0xFB, 0, 0, // 1101 1111 1111 1011
        0x7F, 0xFE, 0, 0, // 0111 1111 1111 1110
        0x60, 0x06, 0, 0, // 0110 0000 0000 0110
        0x30, 0x0C, 0, 0, // 0011 0000 0000 1100
        0x18, 0x18, 0, 0, // 0001 1000 0001 1000
        0x18, 0x18, 0, 0, // 0001 1000 0001 1000
        0x0C, 0x30, 0, 0, // 0000 1100 0011 0000
        0x0C, 0x30, 0, 0, // 0000 1100 0011 0000
        0x07, 0xE0, 0, 0, // 0000 0111 1110 0000
        0x07, 0xE0, 0, 0, // 0000 0111 1110 0000
        0, 0, 0, 0; // 0000 0000 0000 0000
    }

    for (i = 0; i < 20; i++)
        for (j = 0; j < 20; j++) {
            glRasterPos2i(i*16, j*16);
            glBitmap(16, 16, 0, 0, 16, 16, letterA);
        }
    glutSwapBuffers();
}
```



# 문자 그리기

- 비트맵 캐릭터 렌더링하기
  - void **glutBitmapCharacter** (void \*font, int character);
    - font: 비트맵 폰트
      - GLUT\_BITMAP\_8\_BY\_13
      - GLUT\_BITMAP\_9\_BY\_15
      - GLUT\_BITMAP\_TIMES\_ROMAN\_10
      - GLUT\_BITMAP\_TIMES\_ROMAN\_24
      - GLUT\_BITMAP\_HELVETICA\_10
      - GLUT\_BITMAP\_HELVETICA\_12
      - GLUT\_BITMAP\_HELVETICA\_18
    - character: 출력할 문자
  - int **glutBitmapWidth** (GLUTbitmapFont font, int char);
    - font: 사용할 비트맵 폰트
    - char: 길이를 측정할 문자

# 문자 그리기

- 문자 사용 예

```
char *string = "string sample";
```

```
glRasterPos2f (0, 0);
```

```
// 문자 출력할 위치 설정
```

```
int len = (int) strlen(string);
```

```
for (i = 0; i < len; i++)
```

```
    glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18 , string[i]);
```

# 픽스맵 그리기

- 픽스맵 그리기
  - Pixmap: 2가지 색 이상을 사용한 비트맵
  - 배경 이미지, 텍스처에 사용한다.
  - **glDrawPixels** (GLsizei w, GLsizei h, GLenum format, GLenum type, GLvoid \*pixels)
    - 프레임 버퍼로 한 블록이 픽셀을 그린다.
    - w,h: 픽셀 단위로 나타낸 이미지의 폭, 높이
    - Format: 그려질 픽셀의 색 공간
      - GL\_RGB: RGB 픽셀
      - GL\_RGBA: RGBA 픽셀
      - GL\_COLOR\_INDEX: 컬러 인덱스 픽셀
      - GL\_BGR\_EXT: BGR 픽셀
    - Type: 그려질 픽셀의 데이터 타입
      - GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_BITMAP, GL\_INT, GL\_SHORT
    - Pixels: 이미지의 픽셀 데이터에 대한 포인터



# 픽스맵 그리기

- **glPixelZoom** (GLfloat xscale, GLfloat yscale)
  - 픽셀 크기 변환 값을 설정한다.
  - Xscale: 수평 방향 크기 변환 인자 (1.0일 경우는 그대로)
  - Yscale: 수직 방향 크기 변환 인자 (1.0일 경우는 그대로)
- glPixelZoom (1.0, 1.0)
  - 이미지를 그대로
- glPixelZoom (-1.0, 1.0)
  - 이미지를 좌우를 바꾼다
- glPixelZoom (1.0, -2.0)
  - 이미지의 아래 위를 바꾼 뒤 높이를 2배로 늘린다.
- glPixelZoom (0.5, 0.5)
  - 이미지를 1/2의 크기로 축소

# 픽스맵 그리기

- 사용 예)

```
GLubyte *m_bitmap;  
BITMAPINFO *m_bitInfo;  
GLfloat xoffset, yoffset;  
GLfloat xscale = 2.0, yscale = 2.0;
```

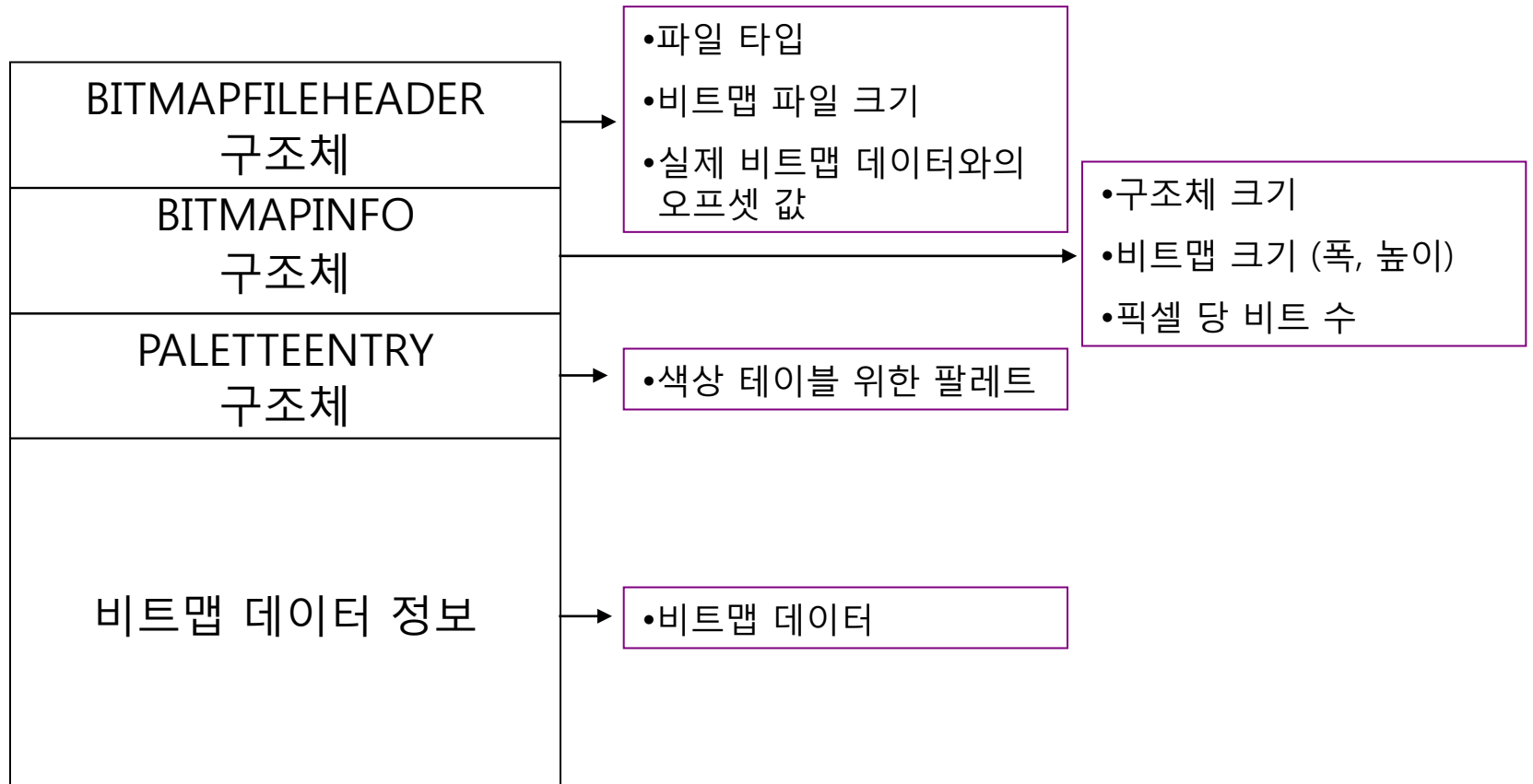
```
m_bitmap = LoadDIBitmap ("bitmap.bmp", &m_bitInfo);
```

```
glRasterPos2f (xoffset, yoffset);  
glPixelZoom (xscale, yscale);  
glDrawPixels (w, h, GL_BGR_EXT, GL_UNSIGNED_BYTE, m_bitmap);
```

## *bmp* 파일 읽기

- 바이너리 모드로 열기
- 비트맵 파일 헤더 읽기
- 비트맵 정보 읽기
- 이미지 크기를 알아내서 memory allocation하기
- 데이터 읽기
- 파일 닫기

# *bmp* 파일 구조



# *bmp* 파일 구조

- 구조체 **BITMAPFILEHEADER**

```
typedef struct {
    WORD    bfType;
    DWORD   bfSize;
    WORD    bfReserved1;
    WORD    bfReserved2;
    DWORD   bfOffBits;
} BITMAPFILEHEADER;
```

// bmfh  
// type: MB (0x42 0x4d)  
// 파일 사이즈: 바이트 단위  
// 0  
// 0  
// 이미지 데이터 위치: 바이트 단위

- 비트맵 정보 헤더 BITMAPINFOHEADER 구조체**

```
typedef struct {
    DWORD   biSize;
    LONG    biWidth;
    LONG    biHeight;
    WORD    biPlanes;
    WORD    biBitCount ;
    DWORD   biCompression;
    DWORD   biSizeImage;
    LONG    biXPelsPerMeter;
    LONG    biYPelsPerMeter;
    DWORD   biClrUsed;
    DWORD   biClrImportant;
} BITMAPINFOHEADER;
```

// bmih  
// BITMAPINFOHEADER 크기: 바이트 단위  
// 픽셀 단위의 이미지 폭  
// 픽셀 단위의 이미지 높이  
// 컬러 평면의 개수: 항상 1  
// 컬러 비트의 개수  
// 사용된 압축 방식의 종류  
// 바이트 단위로 나타낸 이미지 크기  
// 미터당 수평 픽셀 수  
// 미터당 수직 픽셀 수  
// 사용된 컬러 수  
// 중요한 컬러 수

# *bmp* 파일 구조

- **RGBQUAD 구조체**

```
typedef struct tagRGBQUAD {           // rgbq
    BYTE rgbBlue;
    BYTE rgbGreen;
    BYTE rgbRed;
    BYTE rgbReserved;
} RGBQUAD;
```

- **BITMAPINFO 구조체**

```
typedef struct tagBITMAPINFO {        // bmi
    BITMAPINFOHEADER bmiHeader;
    RGBQUAD bmiColors[1];
} BITMAPINFO;
```

# *bmp 파일 로드*

```
GLubyte * LoadDIBitmap (const char *filename, BITMAPINFO **info)
{
    FILE *fp;
    GLubyte *bits;
    int bitsize, infosize;
    BITMAPFILEHEADER header;

    // 바이너리 읽기 모드로 파일을 연다
    if ( (fp = fopen (filename, "rb")) == NULL )
        return NULL;

    // 비트맵 파일 헤더를 읽는다.
    if ( fread (&header, sizeof(BITMAPFILEHEADER), 1, fp) < 1 ) {
        fclose(fp);
        return NULL;
    }

    // 파일이 BMP 파일인지 확인한다.
    if ( header.bfType != 'MB' ) {
        fclose(fp);
        return NULL;
    }
}
```

# *bmp 파일 로드*

// BITMAPINFOHEADER 위치로 간다.

```
infosize = header.bfOffBits - sizeof (BITMAPFILEHEADER);
```

// 비트맵 이미지 데이터를 넣을 메모리 할당을 한다.

```
if ( (*info = (BITMAPINFO *)malloc(infosize)) == NULL ) {  
    fclose(fp);  
    exit (0);  
    return NULL;  
}
```

// 비트맵 인포 헤더를 읽는다.

```
if ( fread (*info, 1, infosize, fp) < (unsigned int)infosize ) {  
    free (*info);  
    fclose(fp);  
    return NULL;  
}
```



# *bmp 파일 로드*

// 비트맵의 크기 설정

```
if ( (bitsize = (*info)->bmiHeader.biSizeImage) == 0 )  
    bitsize = ( (*info)->bmiHeader.biWidth*(*info)->bmiHeader.biBitCount+7) / 8.0 * abs((*info)-  
>bmiHeader.biHeight);
```

// 비트맵의 크기만큼 메모리를 할당한다.

```
if ( (bits = (unsigned char *)malloc(bitsize) ) == NULL ) {  
    free (*info);  
    fclose(fp);  
    return NULL;  
}
```

// 비트맵 데이터를 bit(GLubyte 타입)에 저장한다.

```
if ( fread(bits, 1, bitsize, fp) < (unsigned int)bitsize ) {  
    free (*info); free (bits);  
    fclose(fp);  
    return NULL;  
}
```

```
fclose (fp);  
return bits;
```

```
}
```

- 텍스처 맵핑을 위해서는
  - 텍스처 기능 활성화 (Enable texturing)
  - 텍스처 영상 명시 (Specify texture images)
  - 텍스처 맵핑 할당 (좌표 설정) (Assign texture mapping)
  - 텍스처 파라미터 명시 (Specify texture parameters)
  - 텍스처 환경 명시 (Specify texture environment)

# 텍스처 기능 활성화

- 텍스처 사용을 위해서 사용 기능 활성화
  - Void **glEnable (GLenum mode)**;
    - Mode: GL\_TEXTURE\_1D / GL\_TEXTURE\_2D / GL\_TEXTURE\_3D
      - 1, 2, 3차원 텍스처 매핑
      - glEnable (GL\_TEXTURE\_1D)
      - glEnable (GL\_TEXTURE\_2D)
      - glEnable (GL\_TEXTURE\_3D )
- 사용하지 않는 텍스처는 꺼놓는다.
  - glDisable (GL\_TEXTURE\_1D)
  - glDisable (GL\_TEXTURE\_2D)
  - glDisable (GL\_TEXTURE\_3D )

# 텍스처 영상 명시

- **텍스처를 메모리에 정의한다.**
  - 비트맵 이미지를 저장 해 놓는다.
- **텍스처 맵 정의**
  - glTexImage 함수를 사용하여 텍스처 맵을 정의한다.

# 텍스처 맵핑 1D

- 1D 텍스처 맵핑

- 폭은 있지만 높이가 없는 경우 (또는, visa versa)
- 렌더링 속도가 향상된다.
- 1D 텍스처 맵 정의 함수:
  - **glTexImage1D** (GLenum target, GLint level, GLint components, GLsizei width, GLint border, GLenum format, GLenum type, const GLvoid \*pixels)
    - **target**: 어떤 텍스처 가 정의될 것인지를 나타낸다. (**GL\_TEXTURE\_1D**)
    - **level**: 텍스처 이미지의 상세한 정도를 나타낸다. (0: 기본 이미지)
    - **components**: 각 픽셀에 사용할 컬러 수를 지정 (1, 2, 3, 4)
    - **width**: 텍스처 이미지의 크기 (2의 지수 승)
    - **border**: 경계 픽셀의 수를 조절 (0이어야 함)
    - **format**: 사용할 컬러 값의 종류
      - » GL\_RED/GL\_GREEN/GL\_BLUE/GL\_ALPHA/GL\_RGB/GL\_BGR\_EXT
    - **type**: 각 픽셀 값에 대한 데이터 종류
      - » GL\_BYTE / GL\_UNSIGNED\_BYTE / GL\_BITMAP / GL\_SHORT / GL\_UNSIGNED\_SHORT / GL\_INT / GL\_UNSIGNED\_INT
    - **pixels**: 픽셀 데이터

# 텍스처 맵핑 2D

## • 2D 텍스처 맵핑

- 1픽셀 이상의 높이와 폭을 가지는 이미지
- 복잡한 표면 기하 대신 사용된다.
- 2차원상의 텍스처 이미지를 정의하는 함수
  - **glTexImage2D** (GLenum target, GLint level, GLint components, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid \*pixels)
    - **target**: 어떤 텍스처가 정의될 것인지를 나타낸다. (**GL\_TEXTURE\_2D**)
    - **level**: 텍스처 이미지의 상세한 정도를 나타낸다. (0 사용)
    - **components**: 각 픽셀에 사용할 컬러 수를 지정
      - » 1 ~ 4: RGB이면 3, RGBA이면 4
    - **width**: 텍스처 이미지의 크기 (2의 지수 승)
    - **Height**: 텍스처 의 높이
    - **border**: 경계 픽셀의 수를 조절 (0이어야 한다)
    - **format**: 사용할 컬러 값의 종류
      - » GL\_COLOR\_INDEX / GL\_RED / GL\_GREEN / GL\_BLUE / GL\_ALPHA / GL\_RGB / GL\_BGR\_EXT
    - **type**: 각 픽셀 값에 대한 데이터 종류
      - » GL\_UNSIGNED\_BYTE / GL\_BYTE / GL\_UNSIGNED\_SHORT / GL\_SHORT / GL\_INT / GL\_FLOAT
    - **pixels**: 픽셀 데이터

# 텍스처 매핑 2D

- 예)

```
GLubyte * TexBits;
void setUp ()
{
    BITMAPINFO *texture;

    TexBits = LoadDIBitMap ("bitmap.bmp", &texture);
    glTexImage2D ( GL_TEXTURE_2D, 0, 3, w, h, 0, GL_BGR_EXT,
                  GL_UNSIGNED_BYTE, TexBits);
    glEnable (GL_TEXTURE_2D);
}
```

# 텍스처 모드 환경 설정

- 텍스처 모드 설정

- 조명모델에 따른 다양한 렌더링 방법에 대하여 텍스처 매핑 모드를 지원한다. 즉, 물체색과 조합하여 텍스처를 입힐 수 있다.

- **glTexEnv** 함수를 사용하여 텍스처 모드 설정

- glTexEnvf (GLenum target, GLenum pname, GLfloat param);
- glTexEnvfv (GLenum target, GLenum pname, GLfloat \*param);
- glTexEnvf (GLenum target, GLenum pname, GLint param);
- glTexEnviv (GLenum target, GLenum pname, GLint \*param);
- 텍스처 이미지가 어떻게 폴리곤과 매핑되는지를 제어하는 텍스처 매핑 인자를 설정하는 함수



# 텍스처 모드 환경 설정

target	pname	Param	
GL_TEXTURE_ENV	GL_TEXTURE_ENV_MODE (사용할 텍스처링 종류를 지정)	GL_DECAL	색상 및 조명 정보가 텍스처에 아무런 영향을 미치지 않는다
		GL_REPLACE	기존 물체면의 색을 완전히 텍스처 색으로 대체한다
		<b>GL_MODULATE</b>	텍스처의 색상 정보를 현재 조명을 고려하여 조절 (가장 많이 사용된다)
		GL_BLEND	현재의 텍스처 색상, 조명, 그리고 색상 정보가 모두 혼합되어 표현
	GL_TEXTURE_ENV_COLOR (블렌딩에 사용할 색을 지정)	RGBA 배열	컬러값에 대한 포인터

## – 사용 예)

- `glTexEnvf (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);`

# 텍스처 파라미터 명시

- **텍스처 필터, wrapping**

- 텍스처의 크기가 맵핑 될 다각형의 크기와 다를 때, 크기를 조절
- 텍스처 필터를 사용하여 텍스처 픽셀 사이의 값을 채워 넣는다.
- 텍스처 좌표는 보통 0.0과 1.0사이
- 좌표가 이 범위를 벗어나게 될 경우,
  - GL\_CLAMP: 경계 값으로 대체된다
  - GL\_REPEAT: 반복된다.
- 래핑 모드 설정 함수: **glTexParameter** 함수 사용
  - glTexParameteri (GLenum target, GLenum pname, GLint param);
  - glTexParameteriv (GLenum target, GLenum pname, GLint \*param);
  - glTexParameterf (GLenum target, GLenum pname, GLfloat param);
  - glTexParameterfv (GLenum target, GLenum pname, GLfloat \*param);
    - Target:
      - » GL\_TEXTURE\_1D
      - » GL\_TEXTURE\_2D

# 텍스처 파라미터 명시

- Pname: 설정할 텍스처링 파라미터
  - `GL_TEXTURE_MIN_FILTER`: 텍스처 이미지 축소 필터 지정
  - `GL_TEXTURE_MAG_FILTER`: 텍스처 이미지 확대 필터 지정
  - `GL_TEXTURE_WRAP_S`: 0.0과 1.0 사이의 구간을 벗어나는 텍스처 좌표 s에 대한 처리방법 지정
  - `GL_TEXTURE_WRAP_T`: 0.0과 1.0 사이의 구간을 벗어나는 텍스처 좌표 t에 대한 처리방법 지정
  - `GL_TEXTURE_WRAP_S`, `GL_TEXTURE_WRAP_T`인 경우:
    - `GL_REPEAT`: 필요한 경우 텍스처 이미지가 반복
    - `GL_CLAMP`: 경계 픽셀이 나타난다
  - `GL_TEXTURE_MIN_FILTER`, `GL_TEXTURE_MAG_FILTER`인 경우:
    - `GL_NEAREST`: nearest-neighbor 필터링 (픽셀의 근사치 사용)
    - `GL_LINEAR`: 선형 보간

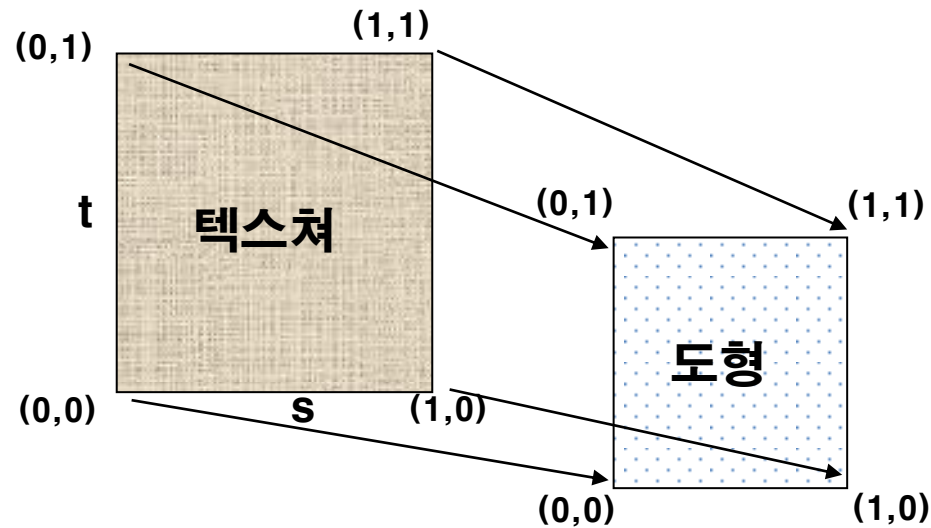
## - 사용 예)

```
glTexParameteri (GL_TEXTURE_2D , GL_TEXTURE_MIN_FILTER, GL_LINEAR);  
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);  
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT );  
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT );
```

# 텍스처 좌표 설정

- 텍스처 좌표 결정

- 임의의 폴리곤에 텍스처를 맵핑 할 경우에는 텍스처와 폴리곤의 정점을 일치시킨다.



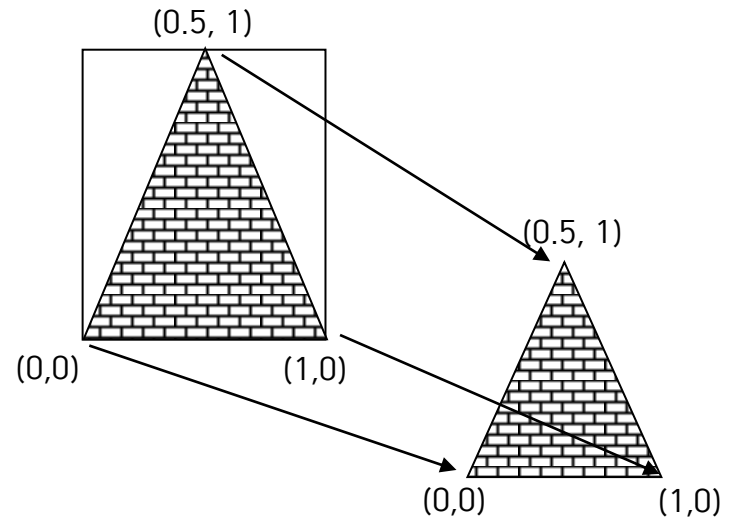
# 텍스처 좌표 설정

## - 텍스처의 위치 설정

- **glTexCoord2d** (GLdouble s, GLdouble t)
- **glTexCoord2f** (GLfloat s, GLfloat t)
- **glTexCoord2i** (GLint s, GLint t)
  - s: 수평 방향 텍스처 이미지 좌표
  - t: 수직 방향 텍스처 이미지 좌표

예)

```
glBegin (GL_TRIANGLE);  
    glTexCoord2d (0.5f, 1.0f);  
    glVertex3f (100.0f, 100.0f, 100.0f);  
  
    glTexCoord2d (0.0f, 0.0f);  
    glVertex3f (50.0f, 50.0f, 100.0f);  
  
    glTexCoord2d (1.0f, 0.0f);  
    glVertex3f (150.0f, 50.0f, 100.0f);  
glEnd ();
```



# 텍스처 좌표 설정

## • 자동 텍스처 매핑

- 물체 내부의 모든 정점마다 텍스처 좌표가 자동으로 할당
- void **glTexGeni** (Glenum coord, Glenum pname, const Glint \*params);
- void **glTexGenf** (Glenum coord, Glenum pname, const GLfloat \*params);
- void **glTexGendv** (Glenum coord, Glenum pname, const GLdouble \*params);
  - coord: 매핑할 텍스처 좌표
    - GL\_S, GL\_T, GL\_R, GL\_Q 중의 하나
  - pname: 설정할 인자
    - GL\_TEXTURE\_GEN\_MODE
    - GL\_OBJECT\_PLANE
    - GL\_EYE\_PLANE
  - params: 텍스처 생성 인자값
    - GL\_TEXTURE\_GEN\_MODE이면,
      - » GL\_OBJECT\_LINEAR: 개체의 꼭지점 좌표로부터 텍스처 좌표가 계산
      - » GL\_EYE\_LINEAR: 시각 좌표를 사용하여 텍스처 좌표가 계산
      - » **GL\_SPHERE\_MAP**: 관측 위치 주변의 구체 내에서 텍스처 좌표 생성
  - 사용 예)
    - glTexGeni (GL\_S, GL\_TEXTURE\_GEN\_MODE, GL\_SPHERE\_MAP);

# 1개 이상의 텍스처 파일

- 1개 이상의 텍스처 파일을 읽어 각각의 폴리곤에 맵핑할 경우
  - 텍스처 생성
    - `glGenTextures (n, textures);`
  - 텍스처 바인딩
    - `glBindTexture (GL_TEXTURE_2D, textures);`
  - 비트맵 읽기
    - `LoadDIBitmap (...);`
  - 텍스처 이미지 정의
    - `glTexImage2D (...);`
  - 텍스처의 각 파라미터 설정
    - `glTexParameterf (GL_TEXTURE_2D, ...);`
  - 텍스처 모드 설정
    - `glTexEnvf (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);`
  - 텍스처 기능을 켜다
    - `glEnable (GL_TEXTURE_2D);`
  - 폴리곤을 그리기 전에 텍스처를 연결한다.
    - `glBindTexture (GL_TEXTURE_2D, textures[0]);`
  - 텍스처의 위치를 설정한다
    - `glTexCoord2f (0.0, 0.0);`

# 1개 이상의 텍스처 파일

- **glGenTextures (GLsizei n, GLuint \*textures)**
  - 텍스처 이름을 생성한다.
  - n: 생성되어야 할 텍스처 이름의 수
  - textures: 생성될 텍스처 이름이 저장된 배열의 첫 번째 값을 가리키는 포인터
- **glBindTextures (GLenum target, GLuint texture)**
  - 텍스처링할 객체에 텍스처를 연결해 준다.
    - 텍스처를 생성하고 사용하게 해 준다.
  - target: 텍스처가 연결될 목표
    - GL\_TEXTURE\_1D / GL\_TEXTURE\_2D
  - texture: 텍스처 이름 (unsigned int 타입)



# 1개 이상의 텍스처 파일

- 객체 그리기

- 객체에 텍스처 이미지 결합하기 위해서 객체를 그리기 전에 텍스처를 결합한다.

- 객체 그리고 텍스처 입히기

```
glBindTexture (GL_TEXTURE_2D, textures[0]);
glBegin (GL_QUADS);
    glTexCoord2f (0.0f, 1.0f);           // 텍스처 위치 설정
    glVertex3f (-400.0, -200.0, -400.0); // 꼭지점
    glTexCoord2f (0.0f, 0.0f);           // 텍스처 위치 설정
    glVertex3f (-400.0, -200.0, 400.0);  // 꼭지점
    glTexCoord2f (1.0f, 0.0f);           // 텍스처 위치 설정
    glVertex3f (400.0, -200.0, 400.0);    // 꼭지점
    glTexCoord2f (1.0f, 1.0f);           // 텍스처 위치 설정
    glVertex3f (400.0, -200.0, -400.0);  // 꼭지점
glEnd ();
```

# 1개 이상의 텍스처 파일

- 예) 3개의 이미지를 텍스처로 사용하기

```
GLuint texture_object[3];
```

```
// 텍스처 이름
```

```
// 3개 텍스처 만들고 이미지 결합하기
```

```
glGenTextures (3, texture_object);
```

```
glBindTextures (GL_TEXTURE_2D, texture_object[0]);
```

```
LoadBitmap();
```

```
glTexImage2D (...); glTexParameteri (...);
```

```
glBindTextures (GL_TEXTURE_2D, texture_object[1]);
```

```
LoadBitmap();
```

```
glTexImage2D (...); glTexParameteri (...);
```

```
glBindTextures (GL_TEXTURE_2D, texture_object[2]);
```

```
LoadBitmap();
```

```
glTexImage2D (...); glTexParameteri (...);
```

```
// 만든 텍스처를 객체에 결합하기
```

```
glBindTexture (GL_TEXTURE_2D, texture_object[0]);
```

```
glBegin (GL_QUADS);
```

```
...
```

```
glEnd ();
```

```
glBindTexture (GL_TEXTURE_2D, texture_object[1]);
```

```
glBegin (GL_QUADS);
```

```
...
```

```
glEnd ();
```

# 프로그램 만들어 보기

- 초기화

```
#include <stdio.h>           // 헤더 파일 삽입
#include <windows.h>          // 비트맵 파일 관련 자료 저장
GLubyte *pBytes;             // 데이터를 가리킬 포인터
BITMAPINFO *info;            // 비트맵 헤더 저장할 변수
```

// 조명 설정

// n개의 이미지 텍스처 매핑을 한다.

```
glGenTextures (n, textures);
```

//텍스처와 객체를 결합한다. --- (1)

```
glBindTexture (GL_TEXTURE_2D, textures[0]);
```

//이미지 로딩을 한다. --- (2)

```
pBytes = LoadDIBitmap ( "이미지.bmp", &info );
```

//텍스처 설정 정의를 한다. --- (3)

```
glTexImage2D ( GL_TEXTURE_2D, 0, 3, W, H, 0, GL_BGR_EXT,
                                                       GL_UNSIGNED_BYTE, pBytes);
```

# 프로그램 만들어 보기

//텍스처 파라미터 설정

--- (4)

```
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);  
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);  
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);  
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

// 나머지 n-1개의 텍스처에도 (1) ~ (4)까지의 과정을 진행하여 텍스처를 설정한다.

// 텍스처 모드 설정

```
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, GL_MODULATE);
```

// 텍스처 매핑 활성화

```
glEnable(GL_TEXTURE_2D);
```

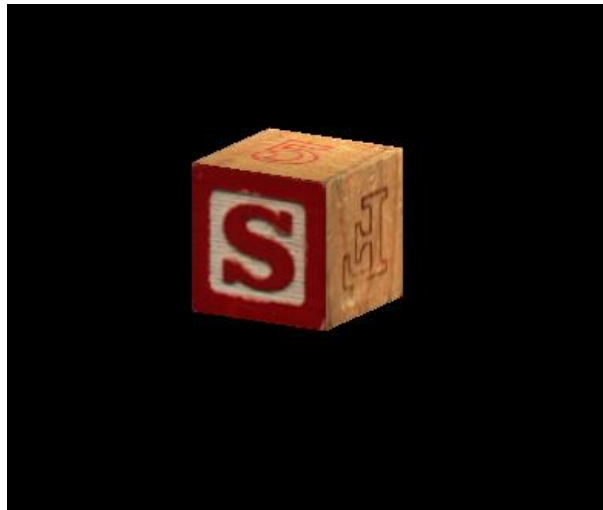
// 텍스처를 객체에 맵핑

```
glBindTexture (GL_TEXTURE_2D, texture[0]);  
glBegin (GL_QUADS);  
...  
glEnd ();
```

## 실습 32

- **텍스처 맵핑 하기**

- 정육면체를 그려서 각 면에 다른 텍스처를 맵핑해 본다.
  - 쿼드를 사용하여 각 면의 꼭짓점을 만든다.
- 육면체는 x축과 y축으로 각각 30도씩 회전되어 있고 y축을 기준으로 회전하고 있다.
- 비트맵 로딩:
  - 강의노트의 코드를 사용한다 (LoadDIBitmap 함수 사용)
  - 1개 이상 (최소 2개)의 비트맵을 사용한다.



## 실습 33

- 실습 30 또는 31 (눈 내리는 화면에 조명 애니메이션)에 텍스처 입히기

- 바닥에 텍스처를 입힌다.
- 중앙의 피라미드에 텍스처를 입힌다.
- 구에 텍스처를 입힌다.
- 구 매핑 예)

// 자동 매핑 설정

```
glEnable (GL_TEXTURE_GEN_S);  
glEnable (GL_TEXTURE_GEN_T);
```

// 구 매핑

```
glTexGeni (GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);  
glTexGeni (GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);  
glBindTexture (GL_TEXTURE_2D, textures[1]);  
glColor3f (1.0, 1.0, 1.0);  
glutSolidSphere (100.0, 30.0, 30.0);
```

// 자동 매핑 해제

```
glDisable (GL_TEXTURE_GEN_S);  
glDisable (GL_TEXTURE_GEN_T);
```

# 블렌딩 (Blending)

- 블렌딩

- 두 가지 색상을 섞어서 그리는 기능
- **투명도 조절, 안티 앨리어싱 효과**
  - 투명도는 RGBA 색상을 이용하여 A 값을 조절하여 투명한 효과를 넣는다.
    - Alpha 값이 1.0: 완전 불투명, Alpha 값이 0.0: 완전 투명
- 기능 활성화
  - glEnable (**GL\_BLEND**);
- 원본 색상과 대상 색상 블렌딩 함수
  - **void glBlendFunc** (GLenum source, GLenum destination)
    - Source와 destination 색상 값
      - » Source color: glColor 함수로 설정된 색상에 source 블렌딩 적용 (들어오는 화소값)
      - » Destination color: 색상 버퍼 내에 저장되어 있는 색상값에 적용 (목적지의 위치에 있던 화소값)
      - » 이 둘 색상을 하나로 합쳐서 만들어진 최종 색상으로 픽셀을 그린다.

# 블렌딩 (Blending)

- 뒤에 있는 것을 먼저 그리고, 불투명한 객체를 먼저 그린 후 투명한 객체를 그린다.
- 반투명으로 설정하려면  
**glBlendFunc (GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA)**로 설정
  - source 색상의 알파값에 비례하여 투명도 결정
- 표준 블렌딩 공식:  $C_f = (C_s * s) + (C_d * d)$



# 블렌딩 (Blending)

- Source 블렌딩 값

GL_ZERO	(0, 0, 0, 0)
GL_ONE	(1, 1, 1, 1)
GL_DST_COLOR	SOURCE * DESTINATION
GL_ONE_MINUS_DST_COLOR	Source * {(1, 1, 1, 1) – destination}
GL_SRC_ALPHA	Source * source alpha
GL_ONE_MINUS_SRC_ALPHA	Source * (1 – source alpha)

- Destination 블렌딩 값

GL_ZERO	(0, 0, 0, 0)
GL_ONE	(1, 1, 1, 1)
GL_SRC_COLOR	DESTINATION * SOURCE
GL_ONE_MINUS_SRC_COLOR	Destination * {(1, 1, 1, 1) – source}
GL_SRC_ALPHA	Destination * source alpha
GL_ONE_MINUS_SRC_ALPHA	Destination * (1 – source alpha)

# 블렌딩 (Blending)

- 사용 예

```
glEnable(GL_CULL_FACE);
```

```
glEnable (GL_BLEND);
```

```
glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

```
glColor4f (0.0, 0.0, 1.0, 0.3);
```

```
glutSolidSphere(2.5, 16, 8);
```

```
// 기존의 색상 버퍼에 들어있는 색상에 0.7의 알파값을
```

```
// 그리려는 구에 0.3의 알파값이 적용된다.
```

# 블렌딩 (Blending)

```
glEnable (GL_BLEND);  
glEnable (GL_CULL_FACE);
```

```
glClearColor (1.0, 1.0, 1.0, 0.0);  
glBlendFunc (GL_ONE, GL_ZERO);  
glBegin (GL_TRIANGLES);  
    glColor4f (1.0, 0.0, 0.0, 0.2);  
    glVertex3f (0.0, 50.0, 0.0);  
    glVertex3f (-50.0, 0.0, 0.0);  
    glVertex3f (50.0, 0.0, 0.0);  
glEnd ();
```



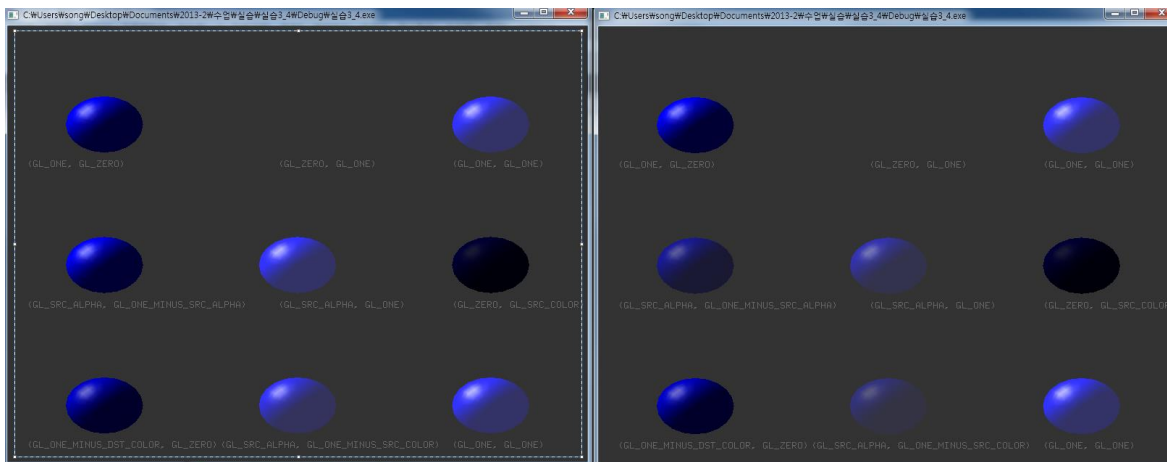
```
glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);  
glBegin (GL_QUADS);  
    glColor4f (0.0, 0.0, 1.0, 0.5);  
    glVertex3f (-40.0, 40.0, 0.0);  
    glVertex3f (-40.0, 5.0, 0.0);  
    glVertex3f (40.0, 5.0, 0.0);  
    glVertex3f (40.0, 40.0, 0.0);  
glEnd ();
```

# 블렌딩 (Blending)

```
glEnable (GL_BLEND);  
glEnable (GL_CULL_FACE);  
  
glClearColor (1.0, 1.0, 1.0, 0.0);  
glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);  
glBegin (GL_TRIANGLES);  
    glColor4f (1.0, 0.0, 0.0, 0.2);  
    glVertex3f (0.0, 50.0, 0.0);  
    glVertex3f (-50.0, 0.0, 0.0);  
    glVertex3f (50.0, 0.0, 0.0);  
glEnd ();  
  
glBegin (GL_QUADS);  
    glColor4f (0.0, 0.0, 1.0, 0.5);  
    glVertex3f (-40.0, 40.0, 0.0);  
    glVertex3f (-40.0, 5.0, 0.0);  
    glVertex3f (40.0, 5.0, 0.0);  
    glVertex3f (40.0, 40.0, 0.0);  
glEnd ();
```

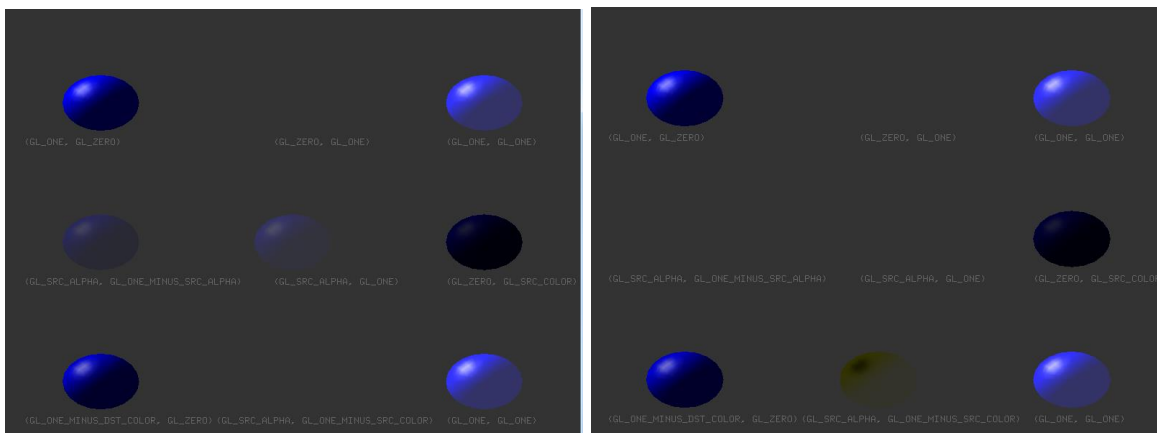


# 블렌딩 (Blending)



glColor4f 에서  $a=1.0$

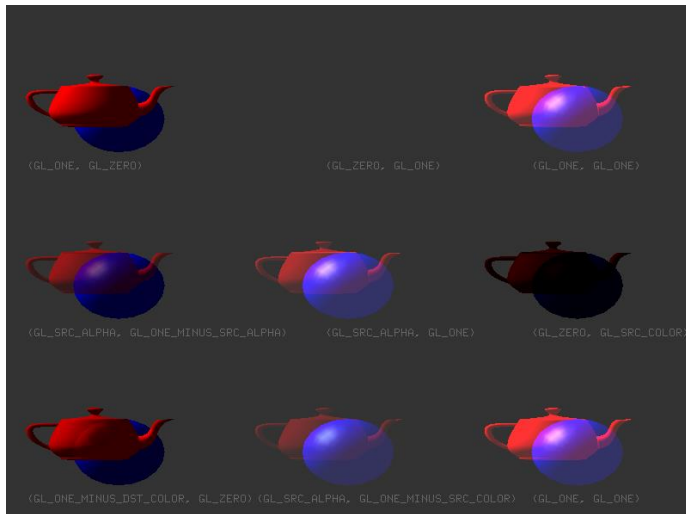
glColor4f 에서  $a=0.5$



glColor4f 에서  $a=0.2$

glColor4f 에서  $a=0.0$

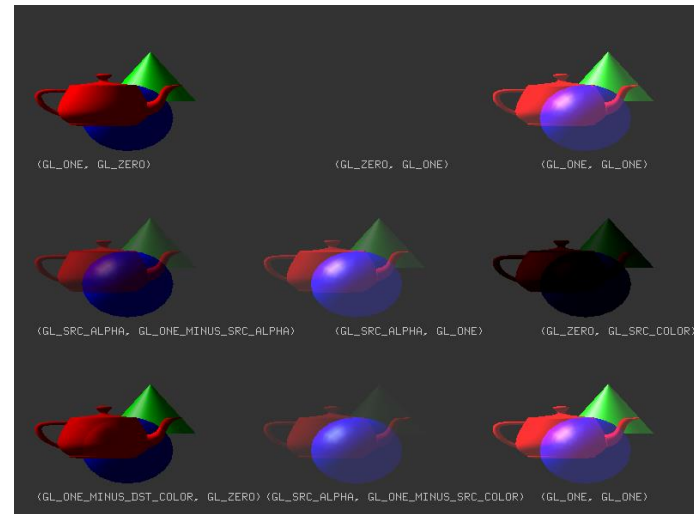
# 블렌딩 (Blending)



구의 alpha = 1.0,  
주전자의 alpha = 0.5

사용 블렌딩 값:

- 1: (GL\_ONE, GL\_ZERO)
- 2: (GL\_ZERO, GL\_ONE)
- 3: (GL\_ONE, GL\_ONE)
- 4: (GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA)
- 5: (GL\_SRC\_ALPHA, GL\_ONE)
- 6: (GL\_ZERO, GL\_SRC\_COLOR)
- 7: (GL\_ONE\_MINUS\_DST\_COLOR, GL\_ZERO)
- 8: (GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_COLOR)
- 9: (GL\_ONE, GL\_ONE)";



구의 alpha = 1.0,  
주전자의 alpha = 0.5  
콘의 alpha = 0.25

# 안개 효과 (Fog)

- 안개
  - 이미 객체에 설정된 색상에 안개 색상과 블렌딩함으로서 나타나는 효과
  - OpenGL에서는 세가지 모드의 안개 효과가 있다
    - GL\_LINEAR : depth cueing에 대한 것으로 거리에 비례한다.
    - GL\_EXP : 짙은 안개나 구름에 사용된다.
    - GL\_EXP2 : 연기나 흐릿한 안개를 나타낸다.
- 안개효과를 설정
  - 우선 안개효과 기능 활성화
    - glEnable (GL\_FOG);
  - 안개 모드를 지정
    - 안개의 색상 , 시작 및 끝 위치, 그리고 밀도를 설정한다.

**glFogf (GLenum pname, GLfloat param);**

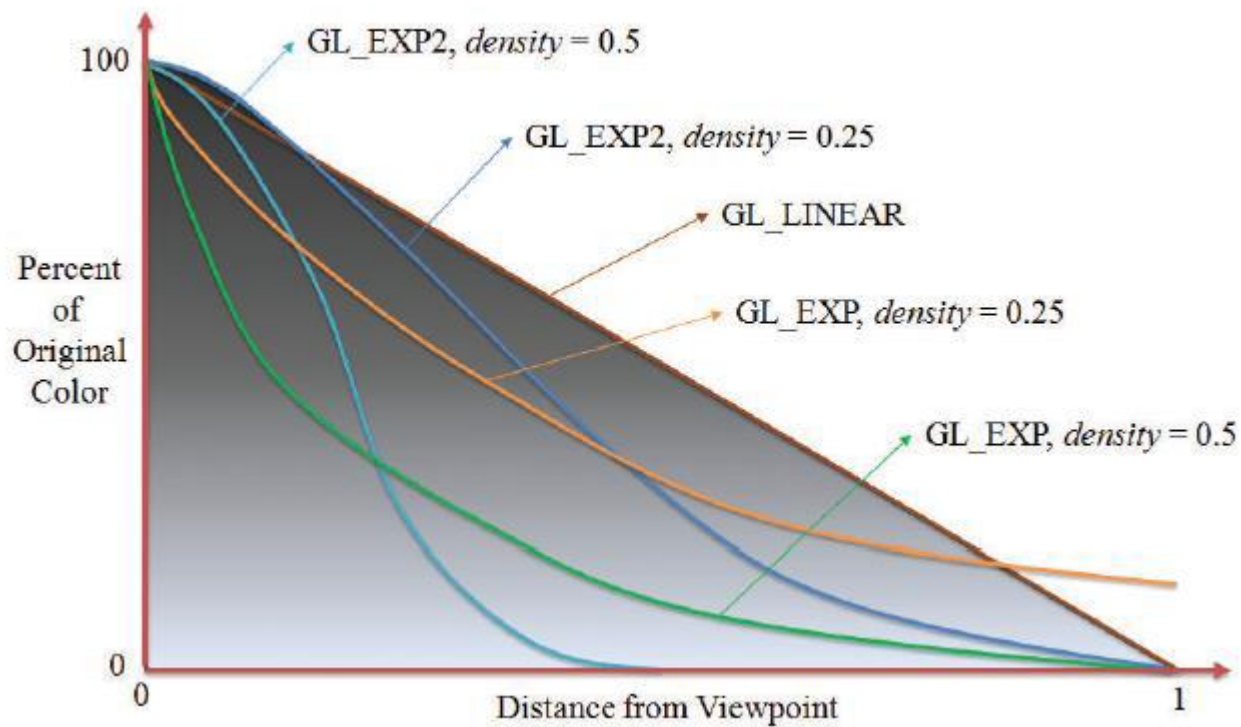
**glFogfv (GLenum pname, const GLfloat \*params);**

# 안개 효과 (Fog)

- pname: 안개 설정 인자
  - GL\_FOG\_MODE**: 안개 블렌드 인자, 초기값은 GL\_EXP  
(GL\_LINEAR / GL\_EXP / GL\_EXP2)
  - GL\_FOG\_COLOR**: 안개의 색, 초기값은 (0, 0, 0, 0)  
(RGBA컬러를 나타내는 4개의 숫자로 된 배열)
  - GL\_FOG\_DENSITY**: 안개 밀도, 0.0보다 큰 수로 설정, 초기값: 1  
(fog mode가 GL\_EXP, GL\_EXP2일 경우 밀도의 설정이 가능)
  - GL\_FOG\_START**: world coordinate상에서 안개 시작 위치  
(관측자로부터 안개 시작 거리, 세계좌표상의 z값, 초기값: 0)
  - GL\_FOG\_END**: 는 world coordinate상에서 안개 끝 위치  
(세계좌표상의 z값, 초기값: 1)  
GL\_FOG\_START로 지정한 위치보다 가까이 있는 물체는 안개효과를 사용하지 않는다.  
GL\_FOG\_END로 지정한 위치보다 멀리 떨어진 곳에 있는 물체는 안개효과를 최대로 사용한다.
- param: pname 값



# 안개 효과 (Fog)




# 안개 효과 (Fog)

Fog equation

$$f = \frac{\text{end} - z}{\text{end} - \text{start}}$$

z is the distance in eye coordinates from origin to fragment being fogged.

Screen-space view



Command manipulation window

```
GLfloat color[4] = { 0.70 , 0.70 , 0.70 , 1.00 };
glFogfv(GL_FOG_COLOR, color);
glFogf(GL_FOG_START, 0.50 );
glFogf(GL_FOG_END, 2.00 );
glFogi(GL_FOG_MODE, GL_LINEAR);
```


Click on the arguments and move the mouse to modify values.

Fog equation

$$f = e^{-(\text{density} * z)}$$

z is the distance in eye coordinates from origin to fragment being fogged.

Screen-space view



Command manipulation window

```
GLfloat color[4] = { 0.70 , 0.70 , 0.70 , 1.00 };
glFogfv(GL_FOG_COLOR, color);
glFogf(GL_FOG_DENSITY, 1.00 );
glFogi(GL_FOG_MODE, GL_EXP);
```


Click on the arguments and move the mouse to modify values.

Fog equation

$$f = e^{-(\text{density} * z)^2}$$

z is the distance in eye coordinates from origin to fragment being fogged.

Screen-space view



Command manipulation window

```
GLfloat color[4] = { 0.70 , 0.70 , 0.70 , 1.00 };
glFogfv(GL_FOG_COLOR, color);
glFogf(GL_FOG_DENSITY, 1.00 );
glFogi(GL_FOG_MODE, GL_EXP2);
```

Click on the arguments and move the mouse to modify values.

## 안개 효과 (Fog)

```
GLfloat fog_color[4] = {0.7, 0.7, 0.7, 1.0};
```

```
GLfloat density = 0.7;
```

```
GLfloat start = 10.0;
```

```
GLfloat end = 50.0;
```

```
glEnable (GL_FOG);
```

```
glFogf (GL_FOG_MODE, GL_LINEAR);
```

```
glFogfv(GL_FOG_COLOR, fog_color); // fog_color는 안개의 색을 의미한다.
```

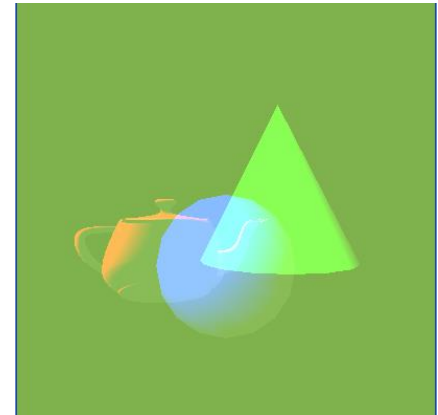
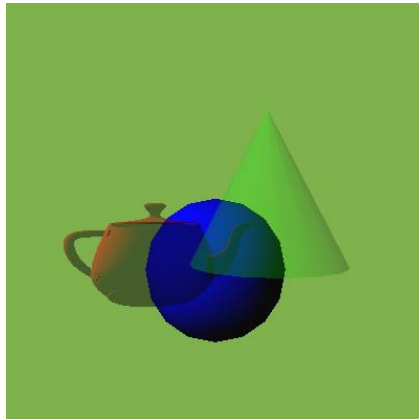
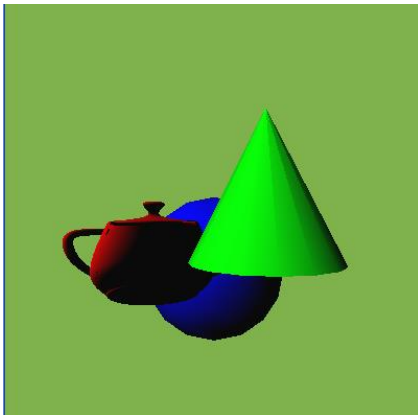
```
glFogf(GL_FOG_START, start); // start는 world coordinate상에서 안개 시작 위치를 의미한다.
```

```
glFogf(GL_FOG_END, end); // end는 world coordinate상에서 안개 끝 위치를 의미한다.
```

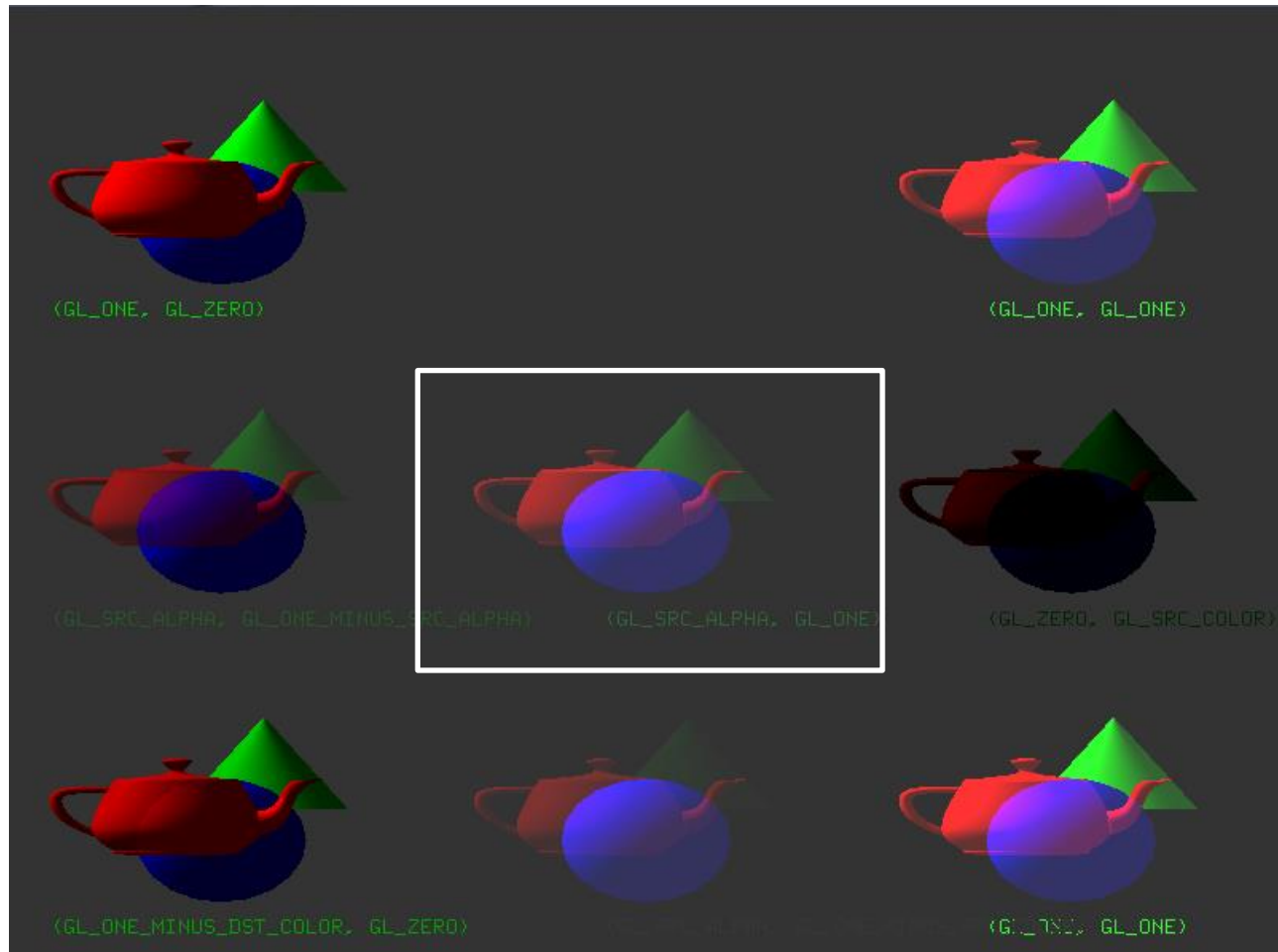
```
glFogf(GL_FOG_DENSITY, density); // fog mode가 GL_EXP, GL_EXP2일 경우 밀도의 설정이 가능
```

## 실습 34

- 3차원 객체를 3개 그린다.
  - 블렌딩 함수의 인자들을 변형시킨 도형을 순서대로 화면에 출력하도록 하는 프로그램을 구현한다.
    - 각 객체들의 투명도를 다양하게 한다. (glColor4f 함수에서 알파값을 0.0 ~ 1.0 사이의 값으로 정한다.
    - 블렌딩 함수의 소스와 데스티네이션 인자의 값을 화면에 출력한다.
    - 9개의 샘플을 만들고, 키보드를 이용하여 한 개를 선택하면 그 객체의 둘레에 사각형을 두르고, 알파값을 조정할 수 있도록 한다.
    - 직각 투영을 한다. (glOrtho 함수 사용, z 값에도 볼륨을 넣어 직육면체 형태의 투영 볼륨을 사용한다.)

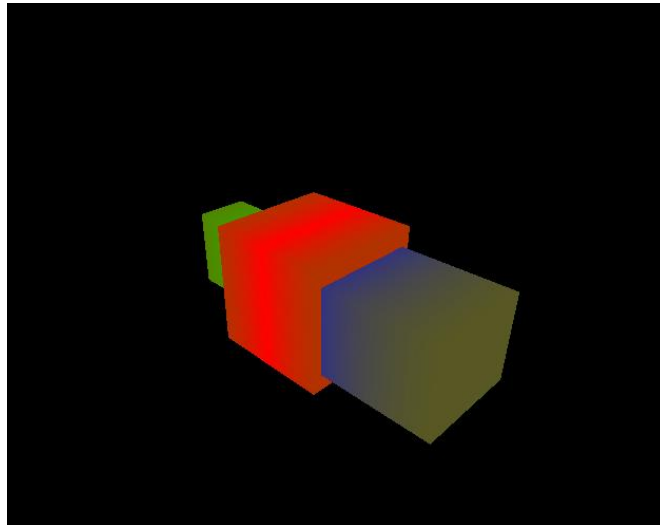


# 실습 34



## 실습 35

- 화면에 육면체 3개를 연결하여 그린다.
  - 키보드 명령어를 이용하여 다양한 포그의 성질을 테스트 한다.
    - d/D: 포그의 density값을 올리기/내리기
    - s/S: 포그의 시작 위치 올리기/내리기
    - e/E: 포그의 끝 위치를 올리기/내리기
    - M: 포그의 모드를 바꾸기

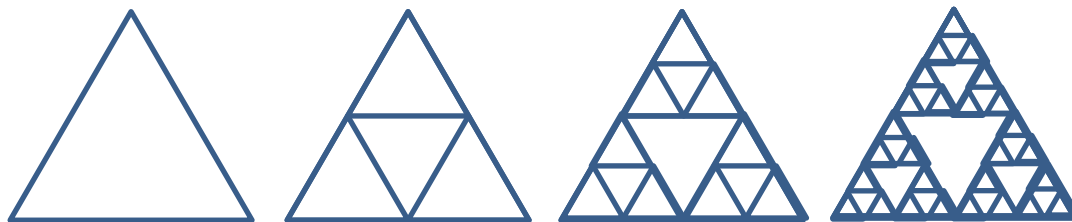


## 실습 36~40 (마침내 마지막 실습)

- 마지막 실습입니다. (각 1점씩 배점)

- 바닥을 그린다.
- 조명을 넣는다.
- 뒷 배경을 만든다. → 텍스처 맵핑을 한다.
- 다음의 5개의 이벤트를 구현한다.

1. 한 꼭지점으로 세운 육면체를 반투명으로 그리고 그 안에 다른 방향의 불투명한 육면체를 그린다. → y축으로 회전하고 있다.
2. 놀이동산의 관람차가 회전하고 있다. 토러스와 육면체를 이용하여 관람차를 그린다. 큰 바퀴가 계속 회전하는데, 관람차는 바닥과평행을 유지하면서 함께 회전한다. → z축으로 회전하고 있다.
3. 해먹이 좌/우에서 반대 방향으로 흔들거린다. 눈사람이 한쪽에 타고 있다가 특정 시점이 되면 다른쪽 해먹으로 점프하여 위치를 바꾼다.
4. 바닥의 가운데에 피라미드를 Sierpinski 프렉탈 삼각형으로 놓는다.
  1. 시어핀스키 삼각형: 삼각형의 세 변의 중점을 찾아서 새로운 삼각형을 그린다. 단계가 2~5단계로 애니메이션 된다.



5. 화면 한 쪽에 곡면을 사용한 연못을 그린다.
  1. 연못 주위에는 다양한 크기의 블록들이 놓여있다.
  2. 연못의 곡면은 시간에 따라 출렁거린다.

# 실습 36~40 (마침내 마지막 실습)

