Computer Graphics

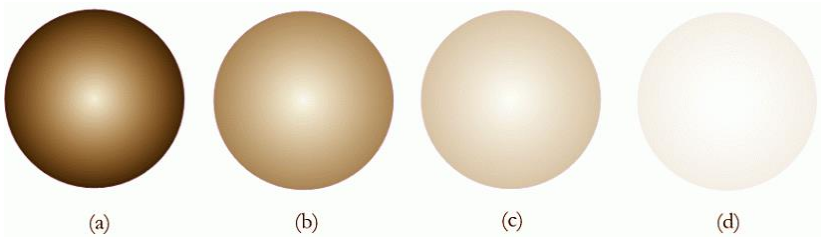
Prof. Jibum Kim

Department of Computer Science & Engineering

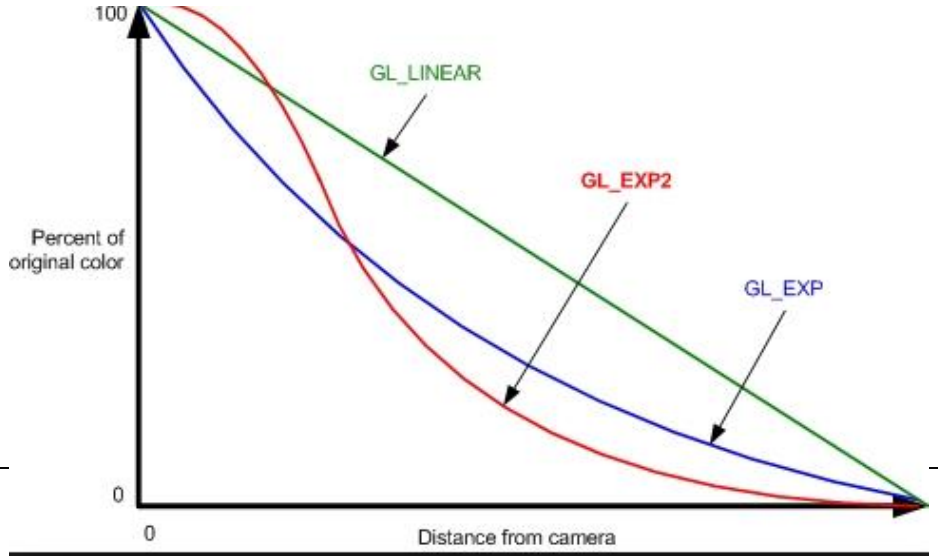
Incheon National University

- **Special Visual Techniques**
- **Fog**

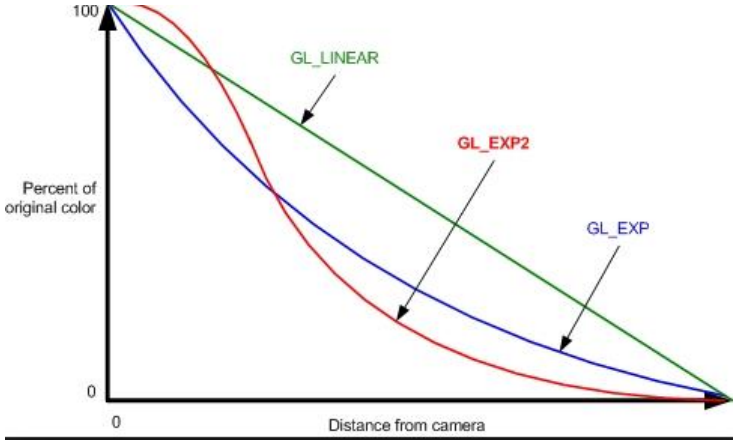
- 안개뿐만 아니라 아지랑이, 연기, 공해, 대기 등 공기 상태로 인한 현상을 통틀어 안개 효과 (fog effect)라 한다
- 안개 효과란 camera (눈)로 부터의 거리가 멀수록 안개색이 더욱 강하게 드러나게 하는 것으로 물체 색과 안개 색의 혼합 비율을 조절하여 blending 한 것이다
- Camera로 부터 멀리 있는 물체는 안개 색 비중을 높임



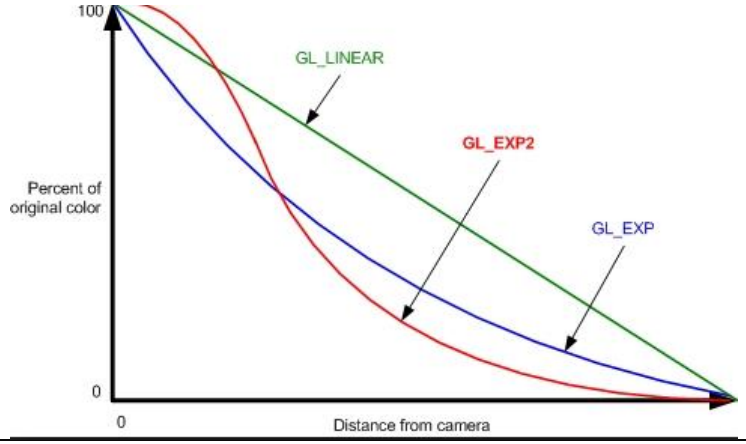
- fog factor (f) : 안개 효과 사용시 현재 물체 색의 비율
- 최종 색 = (1-f)*안개 색 + f* (현재 물채 색)
- f값은 어떻게 결정 되나?
- 기본적으로 어떤 물체가 카메라 (viewer)로부터의 거리가 멀수록 f값을 줄이면 된다 (카메라로 부터 멀수록 안개 색 비중이 높아야 한다)
- 즉, x축(카메라로부터의 거리), y축 (f값)이면 감소함수이다
- 이 안개 함수에는 아래 그림과 같이 여러 종류가 있다



- **안개 함수의 예**
- **1. $f = e^{-(fogDensity*z)}$, z값 (카메라로부터의 거리)이 커지면 f값 급속도로 준다**
- **즉, 안개 색의 비중이 커진다. fogDensity값은 상수**
- **OpenGL에서의 Default 안개 함수이다. 사용시 GL_EXP**
- **2. $f = e^{-(fogDensity*z)^2}$,**
- **1번보다 더 급속도고 감소, 안개 색 비중이 더 커짐. 사용시 GL_EXP2**



- 3. $f = \frac{fogEnd - z}{fogEnd - fogStart}$
- z값이 fogStart이면 f=1, z값이 fogEnd이면 0
- 즉, z ~ [fogStart, fogEnd] 사이에서만 안해 효과를 준다
- 단, z값이 커짐에 따라서 선형 (linear)하게 감소
- 사용시 GL_LINEAR



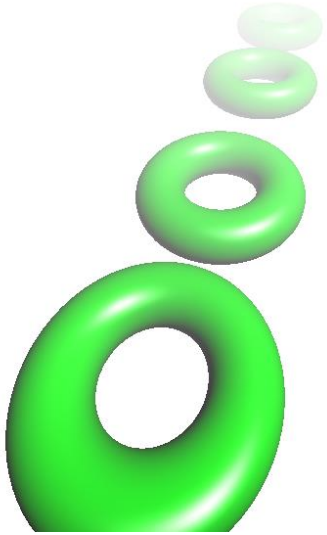
- **OpenGL에서의 Fog 사용**

- Fog 사용 예
- glEnable(GL_FOG); // fog enable 시킴
- fogMode =GL_EXP2; // $f = e^{-(fogDensity*z)^2}$ 사용
- glFogf(GL_FOG_DENSITY, 0.03); // 위의 식에서 fogdensity 값 =0.03
- glHint(GL_FOG_HINT, GL_NICEST); // 가장 나이스 하게

```
glEnable(GL_FOG);
 GLfloat fogColor[4] = {1.0, 1.0, 1.0, 1.0};
 fogMode = GL_EXP2;
 glFogi (GL_FOG_MODE, fogMode);
 glFogfv (GL_FOG_COLOR, fogColor);
 glFogf (GL_FOG_DENSITY, 0.03);
 glHint (GL_FOG_HINT, GL_NICEST);
```

- https://www.dropbox.com/s/27qcwnbwelziaj4/fog_1.txt?dl=0

- 조명 및 안개 효과 추가
- fogMode 바꿔보자



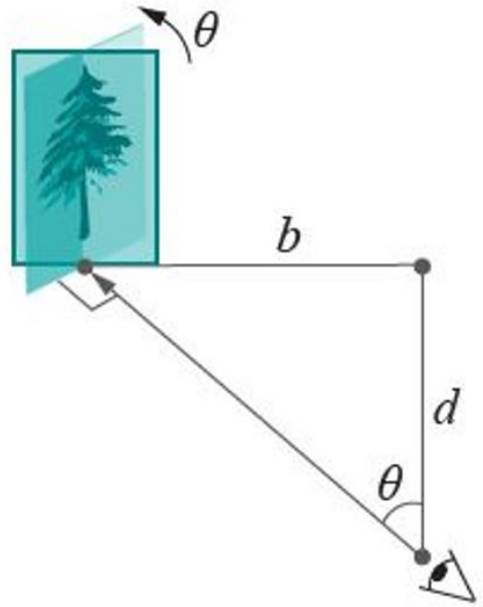
- **Special Visual Techniques**
- **Billboarding**

- **Billboard의 정의: a texture mapped polygon, which always faces the viewer**
- **물체가 나무, 길거리 사인, 가구와 같은 배경인 경우에는 실제로 3D로 만들면 시간이 오래 걸리기 때문에 3D처럼 보이게 만드는 기술이다**
- **Billboarding의 idea: 물체를 계속 회전시켜서 물체의 법선 벡터가 항상 viewer를 향하도록 한다**
- **Unity 매뉴얼: Billboards are a level-of-detail (LOD) method for drawing complicated 3D Meshes in a simpler way when they are far away from the Camera. When a Mesh is far**

- 물체 (texture)가 최초에 z=0인 위치에서 xy 평면에 정의되어 있다고 하자. 이 texture가 z축으로 –d만큼 이동하고 x축으로 –b만큼 이동했다고 하자. 이 경우에 물체의 normal vector가 항상 viewer를 향하려면 texture가 아래 그림과 같이 θ도 만큼만큼 회전해야 한다

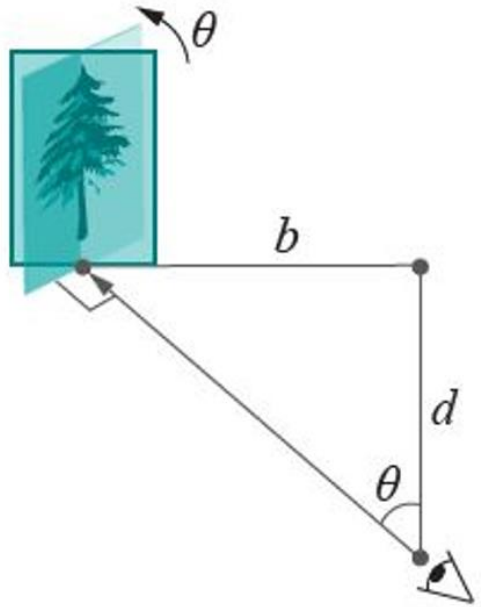
- $tan(\theta) = \frac{b}{d}$

- $\theta = tan^{-1}(\frac{b}{d})$



- https://www.dropbox.com/s/kjbn92vcg8rdmjl/Billboard.zip?dl=0
- 압축 풀고 파일 열기 프로젝트/솔루션으로 불러옴
- space key: billboarding on/off
- Billboarding on시 항상 texture의 normal vector 는 viewer로 향하여 입체감을 준다
- 위/아래 key: texture 이동 (d값 조절)

- glTranslatef(-b, 0.0, -d);
- If billboarding on, rotate the trees image so that it is normal to the viewing direction
- if (isBillboard) glRotatef(atan(b/d)*(180.0/PI), 0.0, 1.0, 0.0); // y축 회전



- **Keyboard 콜백 함수**

- **키보드 콜백 함수: 문자,숫자 키에 대한 콜백 함수 등록하기 위한 것으로 키보드 입력에 해당하는 호출 되는 함수**
- **사용자가 현재 OpenGL 윈도우에 글자나 문자를 타이핑하는 이벤트가 발생하면, 각각의 키에 해당하는 ASCII코드를 만들어 콜백함수로 전달**
- **GLUT의 키보드 콜백 함수의 원형**

```
void glutKeyboardFunc(void (*func)(unsigned char key,
                                   int x, int y));
```

- **눌러진 key를 파라미터 key를 통해 콜백 함수로 전달 (자동으로)**
- **키가 눌러진 순간의 마우스의 위치는 파라미터 x, y에 전달된다 (자동으로)**
-

1. main 함수에 키보드 콜백 함수 myKeyboard 등록

```
void main ()
 {
    glutKeyboardFunc(myKeyboard);
}
```

2. // 키보드를 눌렀을 때의 key 값과 키보드를 눌렀을 때의 마우스 위치 (x, y)가 넘어감

```
void myKeyboard(unsigned char key, int x, int y)
{
   switch(key)
   {
     …
  case '+':
        numVertices++;
        glutPostRedisplay();
        break;
    …
   }
}
```

- glutPostRedisplay() :

- Marks the current OpenGL window as needing to be redrawn (redisplay) by indirectly calling the display callback function
- **주의: 이 함수는 윈도우가 다시 그려져야 함을 표시하기만 할 뿐 즉시 그리기를 하지는 않는다. 이 경우 Keyboard를 눌렀을 때에만 다시 그린다**

- 예: circle을 그리는데 사용되는 점의 개수를 keyboard 콜백 함수 이용하여 변화 시킨다.
- ‘+’ key를 누르면 점 개수 증가 즉
- ‘-’ key 를 누르면 점 개수 감소..

- **https://www.dropbox.com/s/gwob2xgc7n1zfbx/keyboard_1.txt?dl=0**

```
void keyInput(unsigned char key, int x, int y)    // x, y is a location of mouse (you don't have to care)
{
  switch(key)
  {
    case 27:
      exit(0);
      break;
    case '+':
      numVertices++;
      glutPostRedisplay();
      break;
    case '-':
      if (numVertices > 3) numVertices--;
      glutPostRedisplay();
      break;
    default:
      break;
  }
}
```

- 이 경우에는 “ESC” 키에 대한 ASCII 코드인 27을 사용하였다
- http://www.asciitable.com/

- **Mouse 콜백 함수**

- 마우스 이벤트는 마우스 버튼을 누를 때 또는 마우스가 움직일 때 발생한다
- 마우스 콜백함수를 등록하기 위한 원형은 다음과 같다
- glutMouseFunc(int Button, int State, int X, int Y)

1. 첫번째 파라미터

GLUT_LEFT_BUTTON

GLUT_RIGHT_BUTTON

GLUT_MIDDLE_BUTTON 설정 가능

2. 두번째 파라미터

GLUT_DOWN // when the button is pressed

GLUT_UP // when the button is released

3. 세,네 번째 파라미터: X, Y

Event 발생시의 마우스 위치

- 예: 왼쪽 버튼이 눌러질 때 프로그램을 종료하라

- if (state == GLUT_DOWN && button == GLUT_LEFT_BUTTON)
- exit();

- 예) 다음 예제는 마우스 콜백함수를 사용하여
- Mouse 왼쪽 키를 누르면 z축의 CCW 방향으로 1도 증가
- Mouse 오른쪽 키를 누르면 z축의 CW 방향으로 1도 증가

- **https://www.dropbox.com/s/icnuptq7z4g5mmz/mouse_1.txt?dl=0**

- **애니메이션 and Double buffering**

- 비디오 컨트롤러 : frame buffer (메모리) 에 있는 내용을 읽어서 화면에 뿌림. 비디오 컨트롤러가 frame buffer를 읽는 작업은 매우 빨라서 frame buffer를 읽음과 거의 동시에 화면에 그림이 뿌려진다
- 만일 여러 개의 그림이 있는 애니메이션 같은 상황을 하나의 frame buffer를 이용하여 화면에 뿌려진다고 생각해 보자
- 아래 1, 2 과정을 반복하게 된다. 어떤 문제가 생길 까?

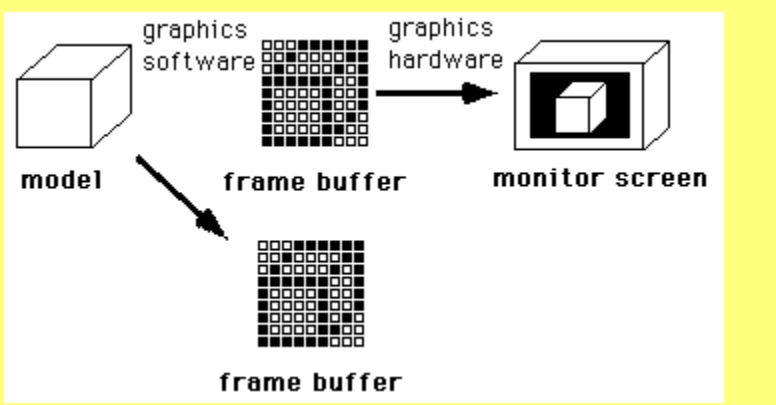
1. Frame buffer에 있는 현재 그림을 비디오 컨트롤러가 읽어서 화면에 뿌림
2. Frame buffer에 있는 내용을 지우고 frame buffer에 다음 그림으로 업데이트 함 (이 작업은 느림)

- Frame buffer의 내용을 다음 그림으로 바꾸는데 시간이 걸리기 때문에 화면이 깜빡 거리는 것 (flickering)처럼 보인다

Double buffering: 두 개의 frame buffer 를 사용

Front buffer에는 현재 화면을 화면에 display 하면서 Back buffer에는 다음 화면을 미리 그려 놓는다

다음 화면이 준비되면 back buffer=>front buffer로 copy 한다



- OpenGL에서의 double buffering 사용법

// main function

1. glutInitDisplayMode(GLUT_DOUBLE);

// display callback function

2. glutSwapBuffers();

// switch back buffer and front buffer

// 이 함수가 실행되면 묵시적으로 glFlush() 실행

- **Timer 콜백 함수**

- Timer 이벤트: 일정한 시간 간격이 지나면 발생하는 이벤트 (애니메이션에 사용)
- Timer 콜백 함수를 등록하기 위한 원형

```
void glutTimerFunc(unsigned int msecs,
                   void (*func)(int value), value);
```

- 첫번째 인자: 얼마 후에 이벤트 발생시킬지 (msec)
- 두번째 인자: Timer이벤트시 호출되어야할 함수
- 세번째 인자: 이벤트 발생시 콜백 함수에 넘겨주고 싶은 파라미터가 있으면 사용

1. main 함수에 Timer 콜백 함수 myKeyboard 등록

```
void main ()
  {
     glutTimerFunc(30, DoTimer, 1);
}
```

2. 30ms 마다 DoTimer 함수 호출

```
void DoTimer(int value)
{
    ….
    glutPostRedisplay();
    glutTimerFunc(30, DoTimer, 1);
}
```

- **https://www.dropbox.com/s/4pptbk0emhzubm1/timer_1.txt?dl=0**

- **Throwing a ball animation**
- **애니메이션을 물리학식을 이용해서도 표현할 수 있다**

- x(t)=ht
- $y(t)=vt-(g/2)t^2$
- g:gravity (중력가속도), 상수
- h:initial velocity (초기 속도 x방향), 상수
- V:initial velocity (초기 속도 y방향), 상수
- t값을 증가시키면서 물체를 translate 시킴
- glTranslatef(h*t, v*t - (g/2.0)*t*t, 0.0);

- More realistic animation using light
- https://www.dropbox.com/s/377jmppeqyjmzvm/gravity_ball_light.txt?dl=0