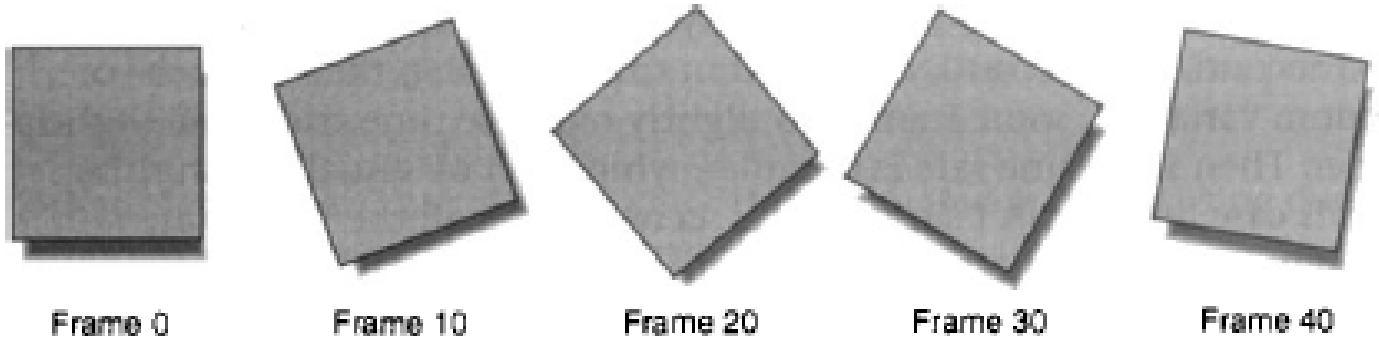
CGL
COMPUTER GRAPHICS LABORATORY

컴퓨터 그래픽스
[03]

2023학년도 1학기

담당교수 : 마 준

- 목표 : 더블 버퍼링을 이용한 사각형 회전
  - 콜백함수 이해하기
  - 셰이딩 모드를 이용한 색상 렌더링 프로그램 만들기
  - 마우스 버튼을 이용한 사각형 회전 프로그램 만들기



Event Handling

- 이벤트 종류
  - 윈도우: 크기 조절, 이동, 겹침 등
  - 마우스: 마우스 버튼 클릭
  - 마우스 움직임: 마우스 이동
  - 키보드: 키보드가 눌리거나 릴리즈
  - Idle(아이들): 아무런 이벤트가 없는 경우
- GLUT Callback 함수
  - glutDisplayFunc
  - glutMouseFunc
  - glutReshapeFunc
  - glutKeyboardFunc
  - glutIdleFunc
  - glutMotionFunc, glutPassiveMotionFunc

- Display 콜백 함수의 호출 경우
  - 1. 처음 윈도우를 열 때
  - 2. 윈도우 위치를 옮길 때
  - 3. 윈도우 크기를 조절할 때
  - 4. 앞 윈도우에 가려져 안 보이던 뒤 윈도우가 활성화 되어 앞으로 드러날 때
  - 5. glutPostRedisplay() 함수에 의해 이벤트 큐에 flag가 게시될 때
- 1~3번 호출 경우는 reshape 이벤트와 동일: reshape 콜백 함수가 실행되면 새로 조정된 뷰포트 및 투상범위를 기준으로 **자동으로 디스플레이 콜백함수가 실행**
- Display callback 함수 등록
  - void glutDisplayFunc(void (*func)(int width, int height));

Reshape Callback

- Reshape 콜백 함수의 호출 경우
  - 처음 윈도우를 열 때
  - 윈도우 위치를 옮길 때
  - 윈도우 크기를 조절할 때
- Reshape callback 함수 등록
  - void glutReshapeFunc(void (*func)(int width, int height));

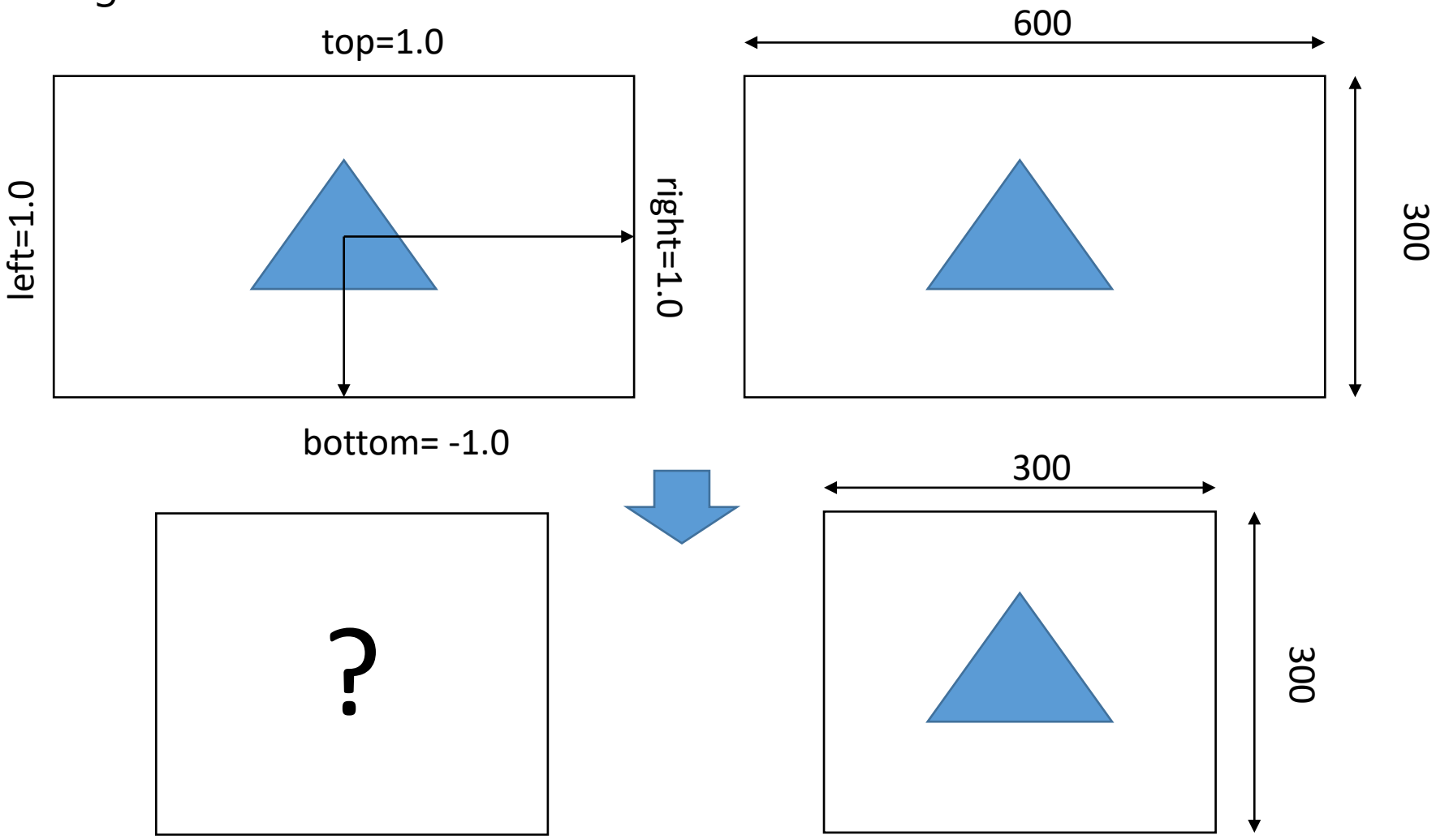
실습 Code: 02_1 Reshape Callback

CGL COMPUTER GRAPHICS LABORATORY

```
#include <glut.h>

void display()
{
    glColor3f(1.0, 1.0, 1.0);

    glBegin(GL_POLYGON);
    glVertex3f(-0.75, -0.75, 0.0);
    glVertex3f(0.75, -0.75, 0.0);
    glVertex3f(0.75, 0.75, 0.0);
    glVertex3f(-0.75, 0.75, 0.0);
    glEnd();

    glFlush();
}

void init()
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
}

void reshape(int new_w, int new_h)
{
    glViewport(0, 0, new_w, new_h);
    float WidthFactor = (float)new_w / 250.0;
    float HeightFactor = (float)new_h / 250.0;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluOrtho2D(-1.0 * WidthFactor, 1.0 * WidthFactor, -1.0 * HeightFactor, 1.0 * HeightFactor);
}
```

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

    glutInitWindowSize(250, 250);
    glutInitWindowPosition(100, 100);

    glutCreateWindow("02 Reshape Callback");
    init();

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();

    return 0;
}
```

- gluOrtho



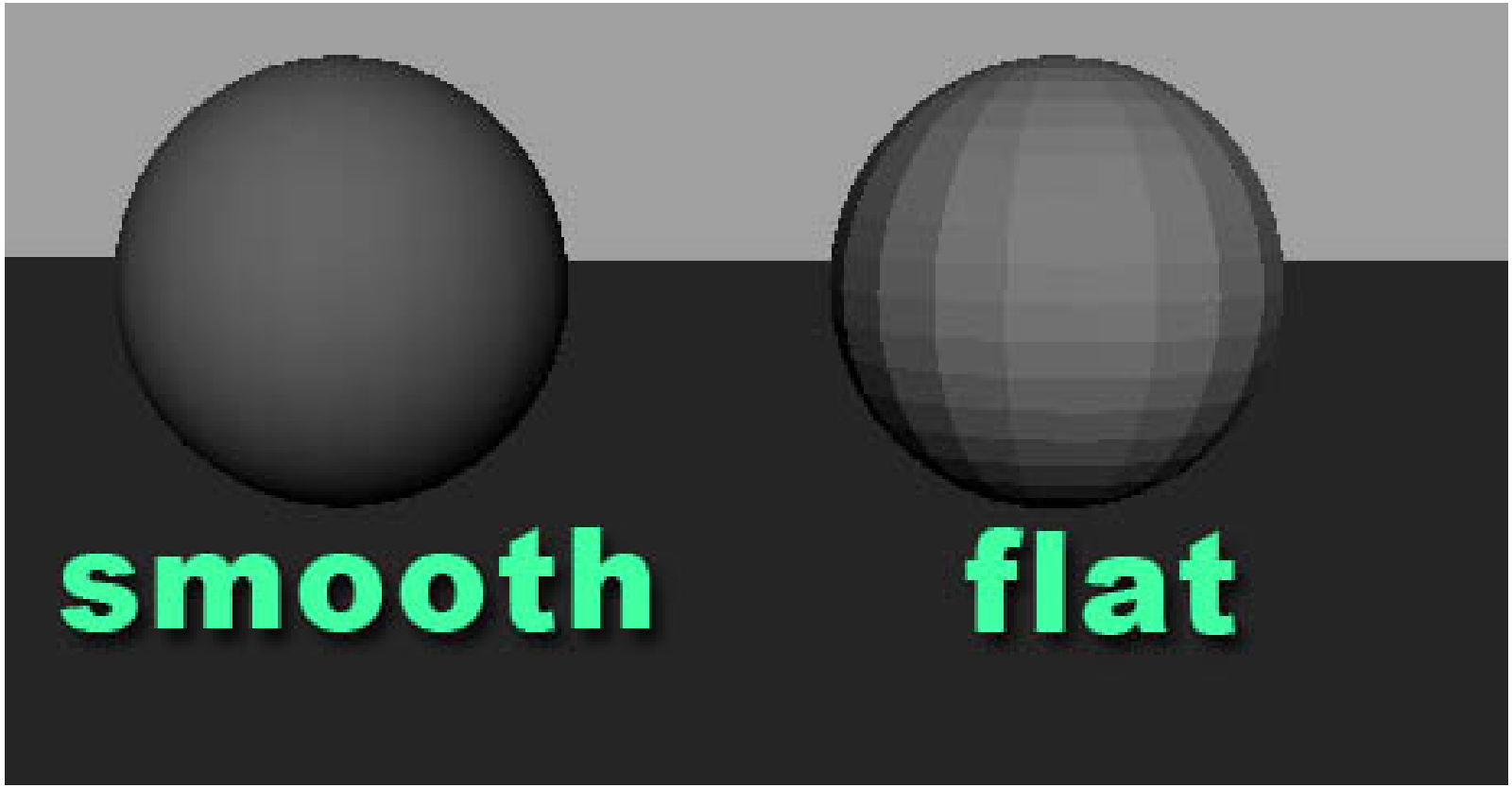
셰이딩 모델 선택

- void glShadeModel(Glenum mode);
  - 셰이딩 모델을 선택
  - 물체의 색상 적용은 정점 기준
  - Flat Shading 또는 Smooth Shading을 선택
  - Flat Shading: 지정된 하나의 색으로 렌더링
  - Smooth Shading: 각 정점의 색을 부드럽게 혼합하여 렌더링

mode	설 명
GL_FLAT	플랫 셰이딩
GL_SMOOTH	스무드 셰이딩

셰이딩 모델 선택

- Smooth 셰이딩 vs Flat 셰이딩



실습 Code: 02_2 Shading Mode

```
void display()
{
    glColor3f(1.0, 1.0, 1.0);

    glShadeModel(GL_FLAT);

    glBegin(GL_POLYGON);
    glColor3f(1, 0, 0);
    glVertex3f(-0.75, -0.75, 0.0);
    glColor3f(0, 1, 0);
    glVertex3f(0.75, -0.75, 0.0);
    glColor3f(0, 0, 1);
    glVertex3f(0.75, 0.75, 0.0);
    glColor3f(1, 1, 1);
    glVertex3f(-0.75, 0.75, 0.0);
    glEnd();

    glFlush();
}
```

Keyboard Callback

- 키보드 입력에 의하여 호출
- Keyboard callback 함수 등록
  - void glutKeyboardFunc(void(*func)(unsigned char key, int x, int y));
  - void glutSpecialFunc(void(*func)(int key, int x, int y)); 특수키 등록 (함수키, 방향키)

함수 키	방향 및 이동 키
#define GLUT_KEY_F1 1	#define GLUT_KEY_LEFT 100
#define GLUT_KEY_F2 2	#define GLUT_KEY_UP 101
#define GLUT_KEY_F3 3	#define GLUT_KEY_RIGHT 102
#define GLUT_KEY_F4 4	#define GLUT_KEY_DOWN 103
#define GLUT_KEY_F5 5	#define GLUT_KEY_PAGE_UP 104
#define GLUT_KEY_F6 6	#define GLUT_KEY_PAGE_DOWN 105
#define GLUT_KEY_F7 7	#define GLUT_KEY_HOME 106
#define GLUT_KEY_F8 8	#define GLUT_KEY_END 107
#define GLUT_KEY_F9 9	#define GLUT_KEY_INSERT 108
#define GLUT_KEY_F10 10	
#define GLUT_KEY_F11 11	
#define GLUT_KEY_F12 12	

Mouse Callback

- glutMouseFunc은 윈도우 안에서 마우스 버튼을 누르거나 놓을때, 각각의 누름이나 놓음은 mouse callback을 호출
- Mousecallback 함수 등록
  - void glutMouseFunc(void (*func)(int button, int state, int x, int y));
  - x, y는 마우스의 현재 위치

button	설 명
GLUT_LEFT_BUTTON	왼쪽 마우스 버튼
GLUT_MIDDLE_BUTTON	중앙 마우스 버튼
GLUT_RIGHT_BUTTON	오른쪽 마우스 버튼

state	설 명
GLUT_UP	마우스 버튼이 눌리지 않은 상태
GLUT_DOWN	마우스 버튼이 눌린 상태

Motion Callback

- glutMotionFunc과 glutPassiveMotionFunc은 현재 윈도우에서 마우스의 움직임에 대해서 motion callback과 passive motion callback을 수행
- Motion callback은 윈도우 안에서 하나 또는 그 이상의 마우스 버튼이 눌린 상태로 움직일 때 호출됨
- Motion callback 함수 등록
  - void glutMotionFunc(void (*func)(int x, int y));
- Passive motion callback은 아무런 마우스 버튼이 눌리지 않은 상태로 움직일 때 호출됨
- Passive Motion callback 함수 등록
  - void glutPassiveMotionFunc(void (*func)(int x, int y));

실습 Code: 02_3 Key/Mouse Callback

CGL
COMPUTER GRAPHICS LABORATORY

```
void keyboardProcess(unsigned char Key, int x, int y)
{
    switch (Key)
    {
    case '1':
        printf("숫자 1키를 입력하였습니다.\n");
    case '2':
        printf("숫자 2키를 입력하였습니다.\n");
    case '5':
        printf("숫자 5키를 입력하였습니다.\n");
    case 'Q':
        printf("문자 Q키를 입력하였습니다.\n");
    }
}
```

```
glutKeyboardFunc(keyboardProcess);
glutMouseFunc(mouseProcess);
```

```
void mouseProcess(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        printf("왼쪽 마우스 버튼을 클릭하였습니다.\n");
    }
    else if (button == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN)
    {
        printf("가운데 마우스 버튼을 클릭하였습니다.\n");
    }
    else if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
    {
        printf("오른쪽 마우스 버튼을 클릭하였습니다.\n");
    }
    else if (button == GLUT_LEFT_BUTTON && state == GLUT_UP)
    {
        printf("왼쪽 마우스 버튼을 뗐습니다.\n");
    }
    else if (button == GLUT_MIDDLE_BUTTON && state == GLUT_UP)
    {
        printf("가운데 마우스 버튼을 뗐하였습니다.\n");
    }
    else if (button == GLUT_RIGHT_BUTTON && state == GLUT_UP)
    {
        printf("오른쪽 마우스 버튼을 뗐하였습니다.\n");
    }
}
```

Timer Callback

- 일정한 시간 간격이 지나면 발생하는 이벤트가 타이머 이벤트이며, 이 이벤트에 대응하는 콜백함수가 타이머 콜백 함수
- 타이머 이벤트를 이용하면 일정 시간 간격으로 이벤트를 발생시켜 애니메이션의 속도 제어가 가능
- 타이머 콜백 함수를 등록한 부분이 msec 후에 호출됨
- Timer callback 함수 등록
  - void glutTimerFunc(unsigned int msecs, void (*func)(int value), value);

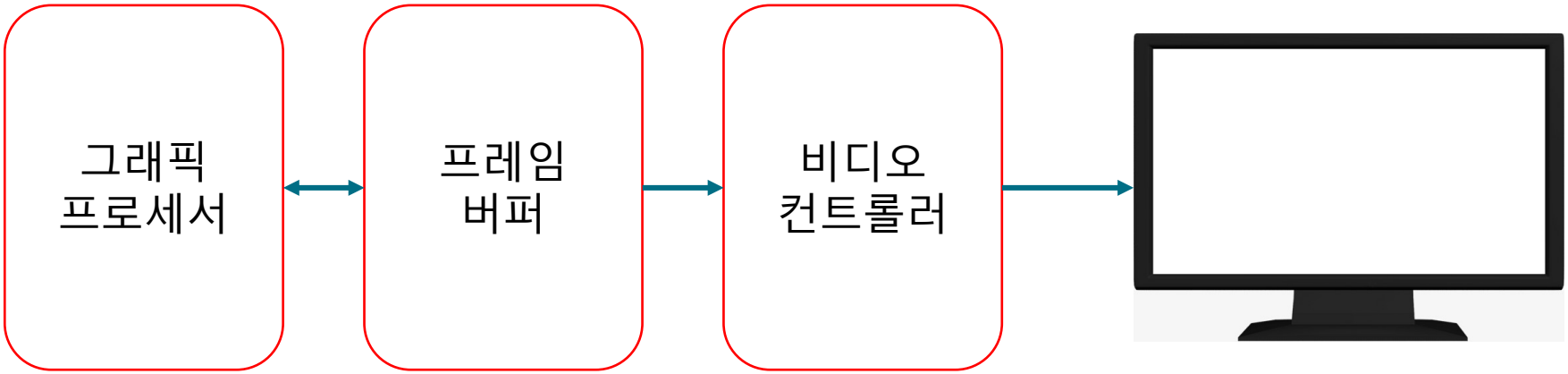
실습 Code: 02_4 Timer Callback

```
GLfloat Red = 0.0f;
GLfloat Green = 0.0f;
GLfloat Blue = 0.0f;

GLint index = 0;          //색상 팔레트의 인덱스
GLfloat Delta = 0.0f;
unsigned char PALETTE[9][3] =
{
    { 255, 255, 255 },  //WHITE
    { 0, 255, 255 },    //CYAN
    { 255, 0, 255 },    //PURPLE
    { 192, 192, 192 },  //LIGHT GRAY
    { 128, 128, 128 },  //DARK GRAY
    { 128, 0, 0 },      //DARK RED
    { 0, 128, 0 },      //DARK GREEN
    { 0, 0, 128 },      //DARK BLUE
    { 0, 0, 0 }         //BLACK
};

void timerProcess(int value)
{
    if (Delta < 2.0f)
    {
        Delta = Delta + 0.01f;

        if (++index >= 8)
        {
            index = 0;
            glClear(GL_COLOR_BUFFER_BIT);
        }
    }
    else
    {
        Delta = 0.0f;
    }
    glutPostRedisplay();
    glutTimerFunc(200, timerProcess, 1);
}
```

```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    //Timer Function에 의한 색상 변경
    Red = PALETTE[index][0] / 255.0f;
    Green = PALETTE[index][1] / 255.0f;
    Blue = PALETTE[index][2] / 255.0f;

    glColor3f(Red, Green, Blue);

    glBegin(GL_POLYGON);
    glVertex3f(-0.75, -0.75, 0.0);
    glVertex3f(0.75, -0.75, 0.0);
    glVertex3f(0.75, 0.75, 0.0);
    glVertex3f(-0.75, 0.75, 0.0);
    glEnd();

    glutSwapBuffers();
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

    glutInitWindowSize(250, 250);
    glutInitWindowPosition(100, 100);

    glutCreateWindow("02_4 Timer Callback");
    init();

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutTimerFunc(200, timerProcess, 1);
    glutMainLoop();

    return 0;
}
```

화면에 영상 렌더링 절차

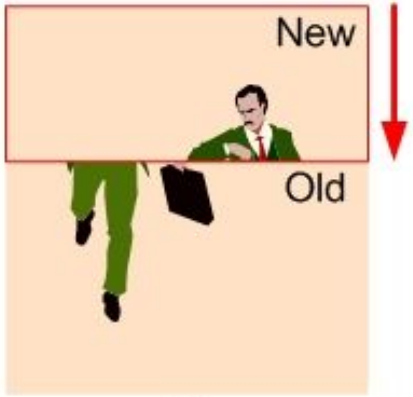
- 화면 렌더링 절차
  - 그래픽 프로세서: 프레임 버퍼에 물체 영상을 써나가는 역할
  - 비디오 컨트롤러: 프레임 버퍼에 쓰인 내용을 화면에 뿌림
  - 비디오 컨트롤러가 프레임 버퍼를 읽는 작업은 매우 빠름
  - 그래픽 프로세서가 프레임 버퍼에 기록하는 것은 상대적으로 느림
  - **애니메이션에서 문제가 됨**



깜빡거림(flickering) 현상

(a)

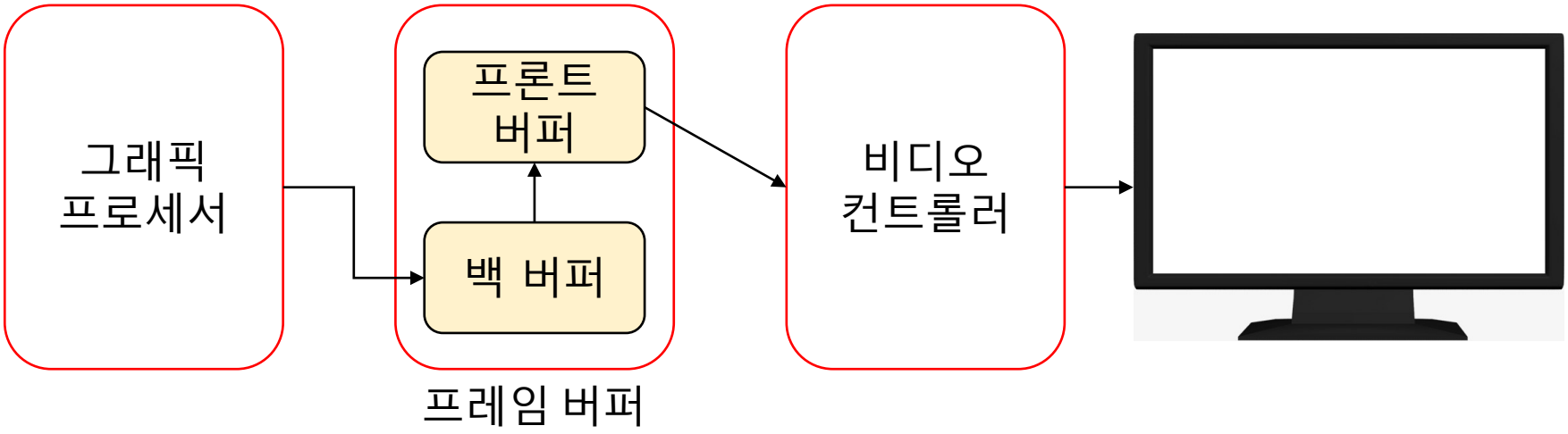
(b)



(c)

(d)

- 두개의 버퍼(Front, Back) 활용
  - 프레임 버퍼의 기록 속도와 검색 속도의 차이 해소
  - 2배의 메모리 필요
  - 프론트 버퍼: 현재 비디오 컨트롤러가 읽고 있는 버퍼
  - 백 버퍼: 현재 화면에서 그 내용이 보이지 않고 있지만, 그래픽 프로세서가 기록 중인 버퍼



실습 Code: 02_4 Timer Callback 수정

```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    //Timer Function에 의한 색상 변경
    Red = PALETTE[index][0] / 255.0f;
    Green = PALETTE[index][1] / 255.0f;
    Blue = PALETTE[index][2] / 255.0f;

    glColor3f(Red, Green, Blue);

    glBegin(GL_POLYGON);
    glVertex3f(-0.75, -0.75, 0.0);
    glVertex3f(0.75, -0.75, 0.0);
    glVertex3f(0.75, 0.75, 0.0);
    glVertex3f(-0.75, 0.75, 0.0);
    glEnd();

    glutSwapBuffers();
}
```

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

    glutInitWindowSize(250, 250);
    glutInitWindowPosition(100, 100);

    glutCreateWindow("02_4 Timer Callback");
    init();

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutTimerFunc(200, timerProcess, 1);
    glutMainLoop();

    return 0;
}
```

```
void reshape(int new_w, int new_h)
{
    glViewport(0, 0, new_w, new_h);
    float WidthFactor = (float)new_w / 250.0;
    float HeightFactor = (float)new_h / 250.0;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluOrtho2D(-5.0 * WidthFactor, 5.0 * WidthFactor, -5.0 * HeightFactor, 5.0 * HeightFactor);
}
```

- void glutInitDisplayMode(unsigned int mode);
  - 그리기 표면의 주요 특징들을 결정
  - 한번 윈도우를 생성한 후에는 변경할 수 없음
  - 여러 개의 모드를 OR 연산자로 묶어서 지정(상호 배타적인 속성의 플래그들은 하나만 지정 가능)

Mode	설 명	Mode	설 명
GLUT_RGBA	RGBA의 창 모드를 지원	GLUT_ALPHA	색상버퍼에 알파값 사용
GLUT_RGB	GLUT_RGBA와 동일	GLUT_DEPTH	깊이버퍼 사용
GLUT_INDEX	인덱스를 가진 색상만 지원	GLUT_STENCIL	stencil buffer 사용
GLUT_SINGLE	단일버퍼 사용	GLUT_MULTISAMPLE	다중맵플링 지원
GLUT_DOUBLE	이중버퍼 사용	GLUT_STEREO	입체창(stereo window) 지원
GLUT_ACCUM	누적버퍼 사용	GLUT_LUMINANCE	휘도를 가진 색상모델 지원

- void glutSwapBuffers(void);
  - 더블 버퍼링 사용시 버퍼를 바꿈
  - 따라서 전면, 후면 버퍼가 교체되면서 미리 준비한 장면을 출력
  - 더블 버퍼링 사용시 glFlush() 함수 대신 호출해야 함

Idle(아이들) Callback

CGL
COMPUTER GRAPHICS LABORATORY

- GL함수의 마지막에 오는 glutMainLoop()는 프로그램을 무한 이벤트 루프로 유도
- 이벤트 루프를 돌 때마다 GLUT는 큐에 쌓인 이벤트를 검사
- 만약 해당 이벤트의 콜백 함수가 정의되어 있으면 그것을 수행하고 정의되어 있지 않으면 해당 이벤트를 무시
- 큐에 새로운 이벤트가 없을 경우, 아이들 콜백을 정의하면 콜백 함수를 만들어 필요한 작업 처리: 애니메이션에 이용 가능
- Idle callback 함수 등록
  - void glutIdleFunc(void (*func)(void));

실습 Code: 02_5 Idle Callback

```
void idleProcess()
{

    angle += 0.0001;
    if (angle > 360.0) angle = 0.0f;

    glutPostRedisplay();
}


void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glRotatef(angle, 0, 0, 1);

    glColor3f(1.0f, 1.0f, 1.0f);

    glBegin(GL_POLYGON);
    glVertex3f(-0.75, -0.75, 0.0);
    glVertex3f(0.75, -0.75, 0.0);
    glVertex3f(0.75, 0.75, 0.0);
    glVertex3f(-0.75, 0.75, 0.0);
    glEnd();

    glutSwapBuffers();
}
```

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

    glutInitWindowSize(250, 250);
    glutInitWindowPosition(100, 100);

    glutCreateWindow("02_5 Idle Callback");
    init();

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutIdleFunc(idleProcess);
    glutMainLoop();

    return 0;
}
```

- void glRotatef(double angle, double x, double y, double z);
  - 회전각과 회전 방향을 정해 행렬을 회전하는 함수
  - x, y, z축 플래그를 설정해 회전을 수행해야함(필수)
    - Ex) glRotatef(10.0, 1, 0, 0);

- void glTranslatef(double x, double y, double z);
  - 이동할 벡터를 설정해 행렬을 이동하는 함수
  - x, y, z축 이동 벡터를 설정해 이동을 수행해야함(필수)
    - Ex) glTranslatef(1.0, 2.0, 0.0);

- 윈도우상에 터키 국기를 그리시오.
  - 별은 여러 개의 삼각형을 사용해 그릴것
- 프로그램의 기능은 다음과 같다.
  - 키패드 1번을 누르면 큰 원이 그려진다.(흰색)
  - 키패드 2번을 누르면 작은 원이 그려진다.(빨간색)
  - 키패드 3번을 누르면 별이 그려진다.(흰색)
  - 엔터키를 누르면 모든 도형이 그려진다.

02 과제-2

- 윈도우상에 삼각형 두개를 사용하여 작은별을 하나 그리시도.
- 마우스 이벤트를 이용하여 왼쪽 마우스 클릭시에 클릭된 위치로 별이 생성되면서 별 사이 직선거리의 선(중점을 연결하는 선)을 그리시오. 모든 별이 계속 추가되어야함.
- 마우스 이벤트를 이용하여 오른쪽 마우스 클릭시에 모든 별이 회전하고 한번 더 클릭하면 토글되어 모든별이 반대로 회전되도록 만드시오. 오른쪽 마우스 클릭마다 계속 토글 되도록.
- 마우스 이벤트를 이용하여 마우스 가운데 버튼 클릭시 모든 별의 색이 바뀌도록 만드시오

CGL
COMPUTER GRAPHICS LABORATORY

- 600x400 윈도우 사이즈
- 100, 100 위치에 윈도우 실행

- [추가점수]
  - 태극기 그려오면 + 점수
  - 참고자료 : http://blog.naver.com/PostView.nhn?blogId=hhp1919&logNo=100184200418

- 과제 제출 기간 : 4월 4일(화요일) 11:59까지 eclass 제출
  - 소스코드만 제출
- 레포트 제출 기간 : 4월 5일(수요일) 수업시간 까지!!
- 3월 29일(수) 휴강.
- COPY = 0 SCORE !!!!

CGL
COMPUTER GRAPHICS LABORATORY

THANK YOU

마 준 박사과정 | majun@sch.ac.kr