OpenGL 3

가상현실론

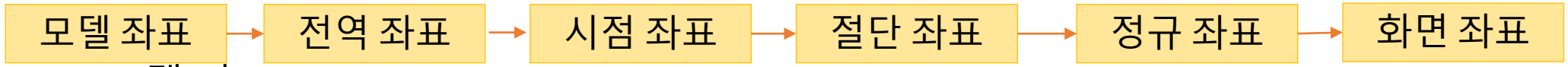
2021/03/10

목차

- 복습
  - 윈도우와 뷰포트
  - 콜백프로그래밍
- 콜백 프로그래밍
- 더블 버퍼링 (double buffering)

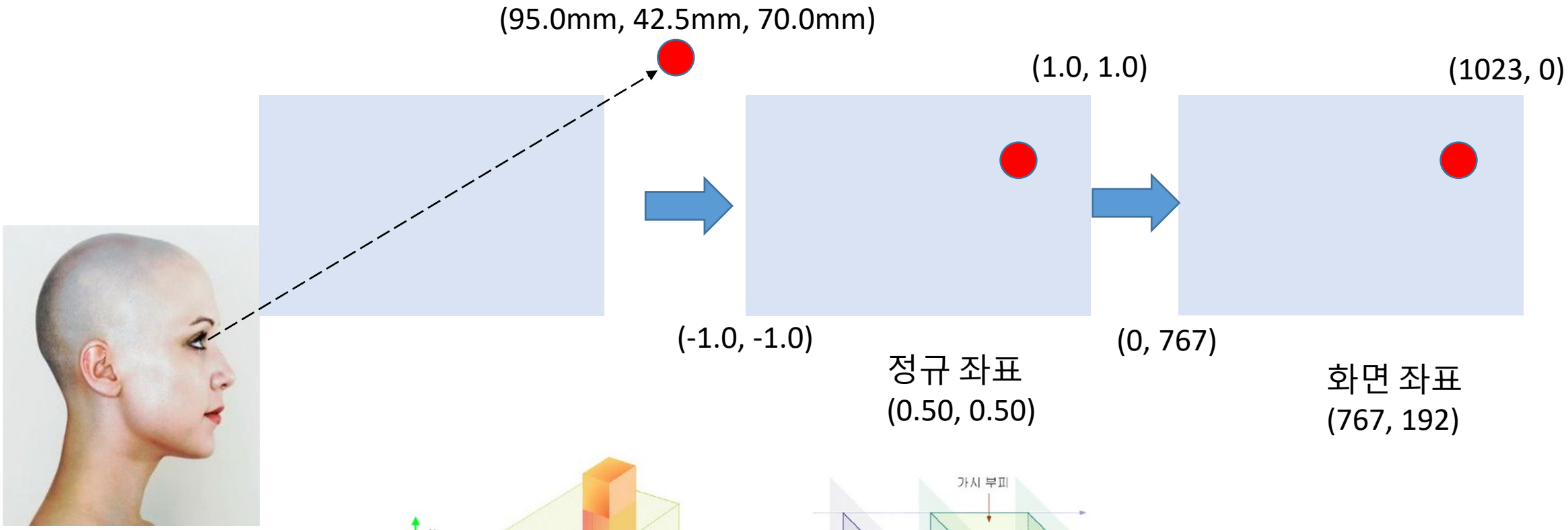
복습

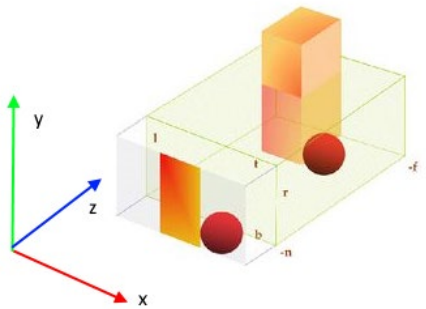
- 윈도우와 뷰포트

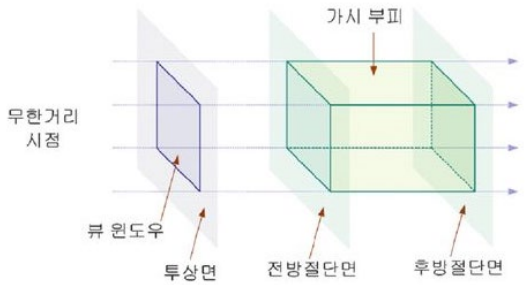


  - 모델 좌표 (local or model coordinate)
    - 모델, 물체별로 설정된 좌표계
  - 전역 좌표 (global coordinate)
    - 모델을 포괄하는 가상 세계의 전역 좌표계
  - 시점 좌표
    - 물체를 바라보는 시점을 기준으로 표현한 것
  - 절단 좌표
    - 시점으로부터 보이지 않는 물체를 잘라내기 편하게 설정한 것
  - 정규 좌표 (NDC: Normalized Device Coordinate)
    - 1을 기준으로 하는 (normalized된) 2차원 좌표
  - 화면 좌표 (SCS: Screen Coordinate System)
    - 화소 단위로 좌표를 표시
    - 예) 1024 x 768 화면에서 x 값은 0~1023, y 값은 0~767

복습


(95.0mm, 42.5mm, 70.0mm)
(1.0, 1.0)
(1023, 0)
(-1.0, -1.0)
(0, 767)
정규 좌표
(0.50, 0.50)
화면 좌표
(767, 192)


y
z
x

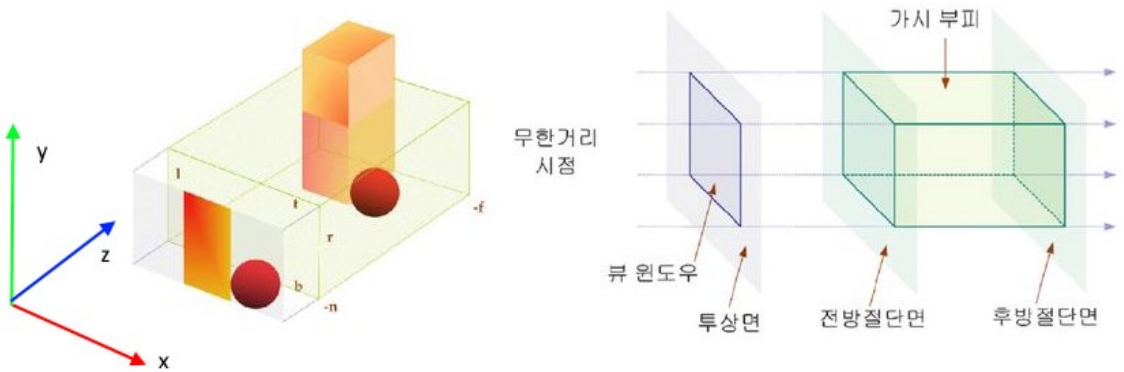

가시 부피
무한거리
시점
뷰 윈도우
투상면
전방절단면
후방절단면

복습

- 콜백프로그래밍-reshape callback
  - GLUT는 아래의 경우에 리셰이프 이벤트 (reshape event)가 발생한 것으로 취급한다.
    - 처음 윈도우를 열 때
    - 윈도우 위치를 옮길 때
    - 윈도우 크기를 조절할 때
  - 리셰이프 이벤트를 등록하기 위한 콜백함수 프로토타입
    - void glutReshapeFunc(void(*func)(int width, int height));
    - func에 reshape 콜백 함수 이름을 넣으면 됨. 예를 들어 reshape function을 myReshape이라고 한다면 glutReshapeFunc(myReshape). myReshape은 프로토타입에 명시된대로 myReshape(int width, int height)의 형태로 선언되어야 하며 width와 height는 변형된 윈도우의 폭과 높이를 나타낸다.

복습

- 콜백프로그래밍- reshape callback
  - glOrtho()
    - glOrtho()는 평행 투상의 일종으로 구체적으로는 그림처럼 직육면체 형태의 가시부피(view volume)을 설정함으로써 정의.
    - 가시부피는 '화면에 보이고자 하는 물체의 범위 ' 를 말함. 따라서 지정된 가시 부피 밖의 물체는 화면에 보이지 않음 (그림의 주황색 큐브의 윗부분은 가시부피 밖에 있기 때문에 보이지 않음).



- 가시부피는 6개의 면으로 정의됨.
- 투상면에 나란하면서 관찰자에 가까운 면을 전방절단면(front clipping plane, front plane, near plane, hither), 먼 면을 후방 절단면(back clipping plane, back plane, far plane, yon)이라고 함.

콜백 프로그래밍 (Callback Programming)

Keyboard callback, Mouse callback, Menu callback, Idle callback, Timer callback

키보드 콜백

- 키보드 입력에 대해 호출되는 함수가 keyboard callback 함수.
- 키보드 콜백 함수를 등록하기 위한 프로토타입 함수
  - void glutKeyboardFunc(void(*func)(unsigned char key, int x, int y))
  - void glutSpecialFunc(void(*func)(int key, int x, int y));
  - glutKeyboardFunc는 문자 및 숫자 키에 대한 콜백 함수를 등록하기 위한 것. 등록된 콜백 함수는 keyboard(unsigned char key, int x, int y)의 형태로 선언되어야 함.
  - 키보드 이벤트가 발생했을 때 GLUT는 눌러진 키를 key 파라미터를 통해서 콜백 함수에 전달. 또한 키가 눌러진 순간 마우스의 위치는 GLUT의 화면 좌표계로 표시되어있는 파라미터 x, y에 의해 전달.
  - 방향 키, 특수 키에 대한 콜백 함수 등록은 glutSpecialFunc()의 프로토타입을 사용함.

키보트 콜백

- 특수키에 대한 GLUT 정의

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	

예제

```
void keyboard(unsigned char key, int x, int y)
{
   switch (key)
   {
   case 27:
   case 'Q' :
   case 'q' :
      exit(0);
      break;
   }
}
```

마우스 콜백

- 마우스 이벤트는 마우스 버턴을 누를 때 또는 마우스가 움직일 때 발생
- 마우스 콜백 함수를 등록하기 위한 프로토타입 함수
  - void glutMouseFunc(void(*func)(int button, int state, int x, int y));
  - 등록된 마우스 콜백 함수는 void mouse(int button, int state, int x, int y)의 형태로 선언되어야 함.
  - Button 파라미터에는 GLUT_LEFT_BUTTON, GLUT_RIGHT_BUTTON, GLUT_MIDDLE_BUTTON 등 버튼 종류를 뜻하는 상수 전달
  - State 파라미터에는 GLUT_DOWN, GLUT_UP 등 해당 버튼이 눌러진 상태인지, 아닌지를 의미하는 상수 전달
  - x, y 파라미터에는 이벤트 발생시의 커서 위치가 전달

마우스 콜백

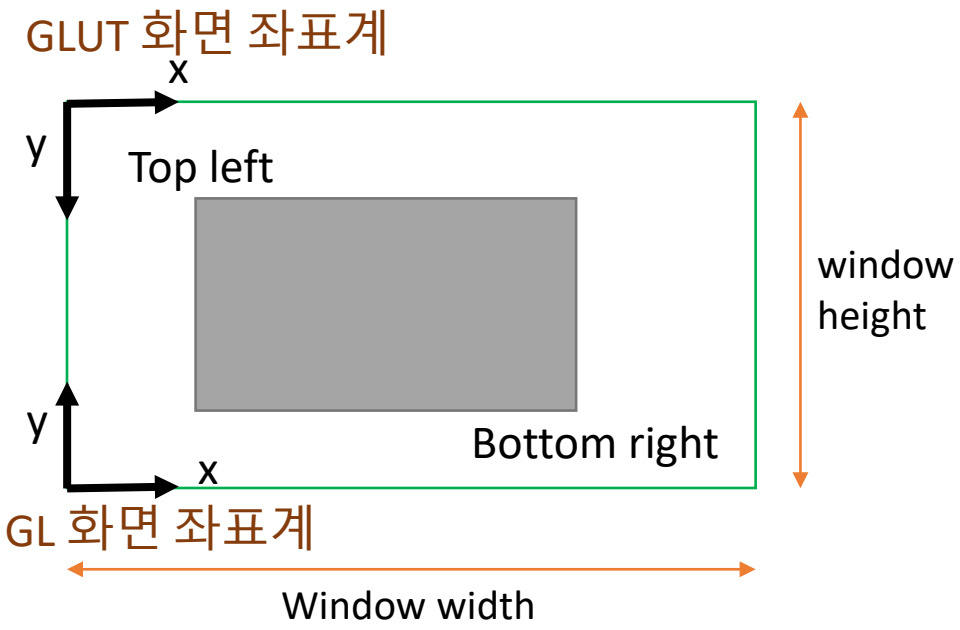
- 마우스를 움직일 때 버튼 상태에 따른 별도의 콜백 함수 정의
  - void glutMotionFunc(void(*func)(int x, int y));
  - void glutPassitveMotionFunc(void(*func)(int x, int y);
  - glutMotionFunc()은 버튼을 누른 상태에서 마우스를 움직일 때 호출되는 콜백 함수 등록.
  - glutPassiveMotionFunc()은 아무런 버튼을 누르지 않은 상태에서 마우스를 움직일 때(passive motion) 호출되는 콜백 함수 등록
- 마우스가 윈도우 안으로 들어오거나 밖으로 나가는 이벤트에 대한 콜백 함수
  - void glutEntryFunc(void(*func)(int state));
  - State 파라미터에는 GLUT_ENTERED(윈도우 밖에서 안으로) 또는 GLUT_LEFT(윈도우에서 나감) 상수 할당.

예제

```
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGB);
   glutInitWindowSize(300, 300);
   glutInitWindowPosition(0, 0);
   glutCreateWindow("OpenGL mouse ex");
   glutDisplayFunc(display);
   glutMouseFunc(mouse);
   glutMotionFunc(move);
   glutKeyboardFunc(keyboard);
   myInit();
   glutMainLoop();
}
```

```
void myInit()
{
   glClearColor(1.0, 1.0, 1.0, 1.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

예제



※ 마우스를 누르고 놓은 위치를 되돌려 줄 때 GLUT은 좌상단 기준의 좌표계 사용. 반면 GL 함수는 좌하단 기준의 화면 좌표계 사용.

```
void display()
{
   glViewport(0, 0, 300, 300);
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(0.5, 0.5, 0.5);
   glBegin(GL_POLYGON);
      glVertex3f(TopLeftX/300.0, (300-TopLeftY)/300.0, 0.0);
      glVertex3f(TopLeftX/300.0, (300-BottomRightY)/300.0, 0.0);
      glVertex3f(BottomRightX/300.0, (300-BottomRightY)/300.0, 0.0);
      glVertex3f(BottomRightX/300.0, (300-TopLeftY)/300.0, 0.0);
   glEnd();
   glFlush();
}
void mouse(GLint button, GLint state, GLint x, GLint y)
{
   if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
      TopLeftX = x;
      TopLeftY = y;
   }
}
void move(GLint x, GLint y)
{
   BottomRightX = x;
   BottomRightY = y;
   glutPostRedisplay();
}
```

메뉴 콜백 (menu callback)

- GLUT의 메뉴 콜백과 관련된 함수 프로토타입
  - int glutCreateMenu(void(*func)(int value))

  메뉴콜백 함수를 등록하며, 그 결과 등록된 메뉴 아이디 값이 리턴됨. 여러 메뉴가 필요하면 메뉴마다 별도로 콜백 함수를 등록해야 함.
  - void glutSetMenu(int id)

  현 메뉴(current menu)를 파라미터 id에 지정한 메뉴로 설정. 파라미터 id는 glutCreateMenu()에 의해 해당 메뉴를 만들 때 리턴된 값
  - void glutAddMenuEntry(char *name, int value)

  현 메뉴에 메뉴 항목(menu entry)추가. 파라미터 name은 항목의 이름이며, value는 goekd 항목이 선택될 때 메뉴 콜백 함수에 전달되는 정수 값
  - void glutAttachMenu(int button)

  GLUT_LEFT_BUTTON, GLUT_RIGHT_BUTTON, GLUT_MIDDLE_BUTTON 등 지정한 마우스 버튼에 현 메뉴 할당.
  - Void glutAddSubMenu(char *name, int menu)

  현 메뉴 항목 중 하나의 서브 메뉴 항목을 추가함. 첫 파라미터인 name은 추가할 서브메뉴명. 둘째 파라미터 menu는 glutCreateMenu()에 의해 해당 서브 메뉴를 만들 때 리턴된 메뉴 아이디값.

예제

```
void my_main_menu(int entryID);
void my_sub_menu(int entryID);
void keyboard(unsigned char key, int x, int y);

GLfloat m_width = 300;
GLfloat m_height = 300;
GLboolean IsSphere = true;
GLboolean IsSmall = true;
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGB);
   glutInitWindowSize(m_width, m_height);
   glutInitWindowPosition(0, 0);
   glutCreateWindow("OpenGL menu ex");
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   GLint MySubMenuID = glutCreateMenu(my_sub_menu);
   glutAddMenuEntry("Small one", 1);
   glutAddMenuEntry("Big one", 2);
   GLint MyMainMenuID = glutCreateMenu(my_main_menu);
   glutAddMenuEntry("Draw sphere", 1);
   glutAddMenuEntry("Draw torus", 2);
   glutAddSubMenu("Change size", MySubMenuID);
   glutAddMenuEntry("Exit", 3);
   glutAttachMenu(GLUT_RIGHT_BUTTON);
   glutKeyboardFunc(keyboard);
   myInit();
   glutMainLoop();
   return 0;
}
void display()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(0.5, 0.5, 0.5);
   if (IsSphere) {// sphere
      if (IsSmall) glutWireSphere(0.2, 15, 15);
      else glutWireSphere(0.4, 15, 15);
   }
   else {// torus
      if (IsSmall) glutWireTorus(0.1, 0.3, 40, 20);
      else glutWireTorus(0.2, 0.5, 40, 20);
   }
   glFlush();
}
```

예제

```
void my_main_menu(int entryID)
{
   if (entryID == 1) IsSphere = true;
   else if (entryID == 2) IsSphere = false;
   else if (entryID == 3) exit(0);
   glutPostRedisplay();
}
//void my_main_menu2(int entryID)
//{
//glutPostRedisplay();
//}
void my_sub_menu(int entryID)
{
   if (entryID == 1) IsSmall = true;
   else if (entryID == 2)  IsSmall = false;
   glutPostRedisplay();
}
```

아이들 콜백 (idle callback)

- glMainLoop()은 프로그램을 무한 이벤트 루프(infinite event loop)으로 가져가는 함수. 각 이벤트 루프마다 GLUT는 큐에 쌓인 이벤트를 검사함. 만약 해당 이벤트의 콜백 함수가 정의되어 있으면 실행하고 정의도어 있지 않으면 무시.
- 큐에 새로운 이벤트가 들어가 있지 않은 경우에도 실행하고자 하는 명령이 있을 경우 아이들 콜백 함수를 만들어 실행. (ex) 애니메이션)
- Idle 콜백 함수 등록 프로토타입
  - void glutIdleFunc(void(*func)())

예제

```
GLfloat Delta = 0.0;
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
   glutInitWindowSize(300, 300);
   glutInitWindowPosition(0, 0);
   glutCreateWindow("OpenGL Drawing Example");
   glutKeyboardFunc(keyboard);
   glutReshapeFunc(reshape);
   glutDisplayFunc(display);
   glutIdleFunc(idle);
   glClearColor(1.0, 1.0, 1.0, 1.0);
   glutMainLoop();
   return 0;
}
```

예제

```
void display()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(0.0, 0.5, 0.8);
   glBegin(GL_POLYGON);
      glVertex3f(-1.0+Delta, -0.5, 0.0);
      glVertex3f(0.0+Delta, -0.5, 0.0);
      glVertex3f(0.0+Delta, 0.5, 0.0);
      glVertex3f(-1.0+Delta, 0.5, 0.0);
   glEnd();
   glutSwapBuffers();
}
void idle()
{
   Delta = Delta + 0.001;
   glutPostRedisplay();
}
```

- glutSwapBuffers() 앞에 있는 모든 함수는 백 버퍼에 적용됨. 즉, 증가된 Delta 값을 사용하여 새로이 그려질 사각형은 백 버퍼에 기록되며 그 동안 현재 화면에 보이는 내용은 현재 프론트 버퍼에 있는 이전 사각형임. 새로운 사각형의 기록이 완료된 순간 glutSwapBuffers() 함수에 의해 버퍼 스와핑을 명령하게 됨

타이머 콜백 (Timer callback)

- Idle callback을 사용하는 경우 사실상 화면 내용이 언제 바뀔지 알 수 없음.
- 타이머 이벤트(timer event)는 일정한 시간 간격에 따라 발생하는 이벤트로 타이머 콜백은 이 이벤트에 대응하는 콜백함수
- 타이머 콜백 함수 등록 프로토타입
  - void glutTimerFunc(unsigned int msecs, void(*func)(int value), int value);
  - msec는 얼마 후에 이벤트를 발생시킬 것인지 나타내는 파라미터
  - 이벤트 발생시 넘겨주고 싶은 파라미터는 value를 사용.

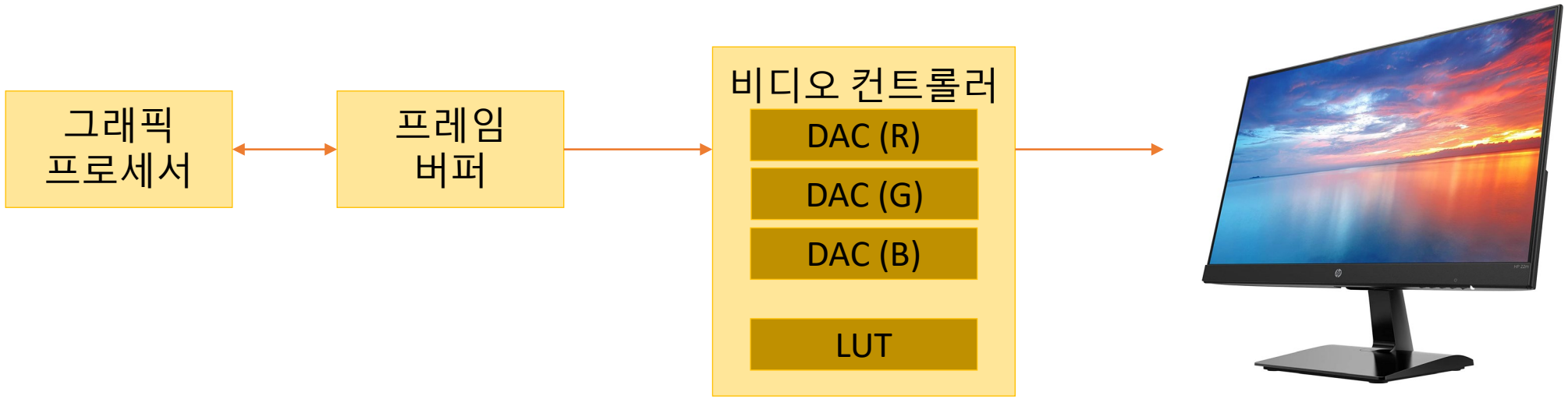
예제

```
GLfloat Delta = 0.0;
void myTimer(int value);
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
   glutInitWindowSize(300, 300);
   glutInitWindowPosition(0, 0);
   glutCreateWindow("OpenGL Drawing Example");
   glutKeyboardFunc(keyboard);
   glutReshapeFunc(reshape);
   glutDisplayFunc(display);
   glutTimerFunc(40, myTimer, 1);
   glClearColor(1.0, 1.0, 1.0, 1.0);
   glutMainLoop();
   return 0;
}
```

```
void myTimer(int value)
{
   Delta = Delta + 0.001;
   glutPostRedisplay();
   glutTimerFunc(40, myTimer, 1);
}
```

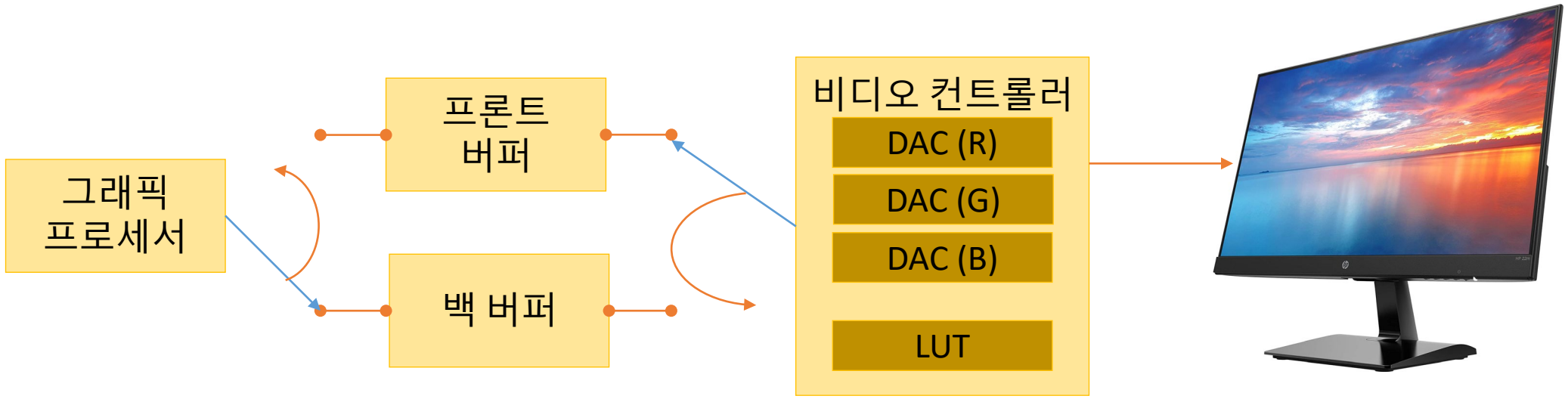
윈도우 타이머와 달리 glut 타이머는 단 한 번만 타이머 이벤트를 발생시키므로 반복된 호출을 위해서는 콜백함수 내부에 glutTimerFunc를 통해서 재호출 필요.

애니메이션과 더블버퍼링



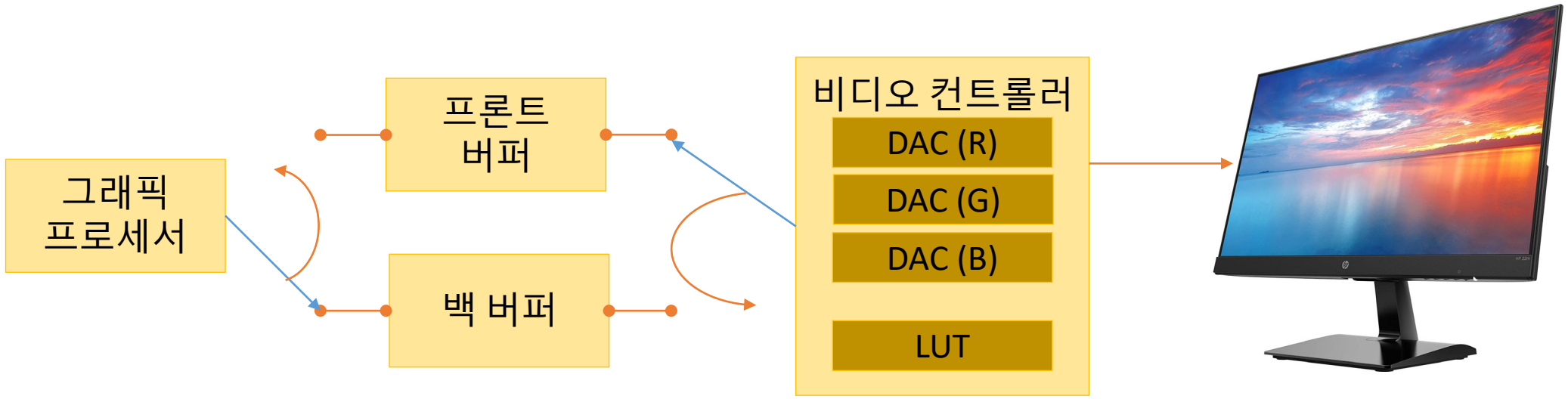
- 비디오 컨트롤러는 프레임 버퍼 내용을 화면에 뿌리기 위한 요소로 램댁(RAMDAC: Random Access Memory Digital to Analog Converter)이라고 불림. 그림처럼 DA 변환기(Digital to Analog Converter)와 RAM으로 구성된 voltage lookup table을 포함한 하나의 칩으로 구성.
- 일반적인 비디오 카드에서 프레임 버퍼(Frame buffer, color buffer)는 2중 포트(dual port) 구조. 즉, 그림처럼 프레임 버퍼의 한쪽에는 그래픽 프로세서가, 다른 쪽에는 비디오 컨트롤러가 연결. 그래픽 프로세서가 프레임 버퍼에 물체 영상을 써나가는 역할을 한다면 비디오 컨트롤러는 쓰인 내용을 DLFR어써 화면예 뿌림.
-

애니메이션과 더블더퍼링



- 그래픽 프로세서가 프레임 버퍼에 쓰는 작업이 비디오 컨트롤러가 프레임 버퍼를 읽어가는 속도에 비해 느릴 경우 화면에 깜빡거림(flickering)이 생김. 이를 방지하기 위해 2개의 프레임 버퍼를 사용하는 것이 더블버퍼링(double buffering).
- 그림과 같이 비디오 컨트롤러가 프론트 버퍼의 내용을 읽고 출력하는 동안 그래픽 프로세서가 백 버퍼에 내용을 써나감.

애니메이션과 더블더퍼링



- Glut에서 더블 버퍼를 사용하려면 다음의 두 함수를 사용
  - void glutInitDisplayMode(unsigned int mode);
  - 윈도우 버퍼 모드를 초기화할 때 사용(glutInitDisplayMode(GLUT_DOUBLE);)
  - Void glutSwapBuffers();
  - 프론트 버퍼와 백 버퍼를 스위칭하기 위함. 이 함수가 실행되면 묵시적으로 glFlush()함수 실행. 즉, 이전의 프런트 버퍼에 가해질 명령을 모두 실행한 후 버퍼 스위칭이 일어남.

디스플레이 리스트 (display list)

- 지엘 함수의 실행모드는 직접모드(immediate mode)와 보류모드(retained mode)로 구분.
- 직접모드: 물체를 화면에 그림과 동시에 물체 생성과 관련되 sahems 정보를 파기. 그 물체를 다시 그리려면 처음부터 다시 실행해야 함.
- 보류모드: 정의된 물체 정보를 그대로 유지하고 재사용. 물체를 다시 그려낼 때 코드를 실행하지 않고 정의된 물체를 컴파일 형태로 재사용함으로써 빠른 속도를 보장.
- GL의 보류모드는 디스플레이 리스트(display list)에 의해 이루어짐.

디스플레이 리스트 (display list)

- GLuint glGenLists(GLsizei range);
  - 아직 사용이 안된 디스플레이리스트 아이디 값 검색하여 아직 사용 안 된 아이디 중 첫번째 아이디 리턴.
  - Range는 원하는 아이디의 개수.
- glNetList(GLuint list GLenum mode);
  - 실제로 리스트를 만들고 내용을 채우기 위함.
  - Mode가 GL_COMPILE_AND_EXECUTE라면 직WJQ 모뜨처럼 명령어가 즉시 실행됨과 동시에 디스플레이 리스트에 저장. GL_COMPILE이면 실행되지 않고 디스플레이 리스트에만 저장.

예제

```
int myListID;
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGB);
   glutInitWindowSize(300, 300);
   glutInitWindowPosition(0, 0);
   glutCreateWindow("OpenGL ex-display list");
   myInit();
   glutDisplayFunc(display);
   glutKeyboardFunc(keyboard);
   glutMainLoop();
   return 0;
}
void display()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glViewport(0, 0, 300, 300);
   glCallList(myListID);
   glFlush();
}
```

디스플레이 리스트 정의

```
void myInit()
{
   glClearColor(1.0, 1.0, 1.0, 1.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
   myListID = glGenLists(1);
   glNewList(myListID, GL_COMPILE);
   glColor3f(0.5, 0.5, 0.5);
   glBegin(GL_POLYGON);
      glVertex3f(-0.5, -0.5, 0.0);
      glVertex3f(0.5, -0.5, 0.0);
      glVertex3f(0.5, 0.5, 0.0);
      glVertex3f(-0.5, 0.5, 0.0);
   glEnd();
   glEndList();
}
```