<모바일응용 소프트웨어 설계>

# 8주차 실습 과제

정보통신공학과

12161774 이 채 은

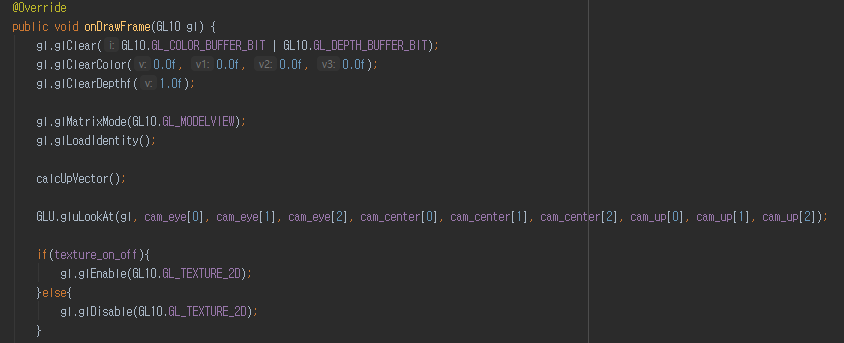
**[소스 코드]**

1. SolarSystemRenderer.java



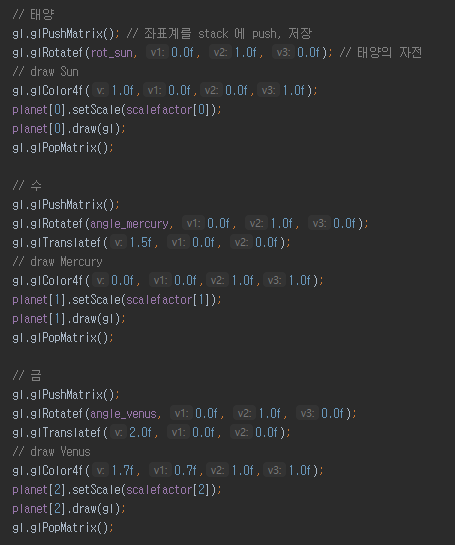
=> rotation 버튼을 누를 때 toggle 이 되도록 하기 위해서 플래그 rot\_flag 을 이용한다.

=> 또한 태양, 달, 행성들의 자전을 회전(rotation)을 이용해서 나타내기 위해서 각 행성 별로 자전각도 angle을 설정한다. 그리고 공전 속도를 나타내기 위해서 행성 별로 orbital을 설정했다.



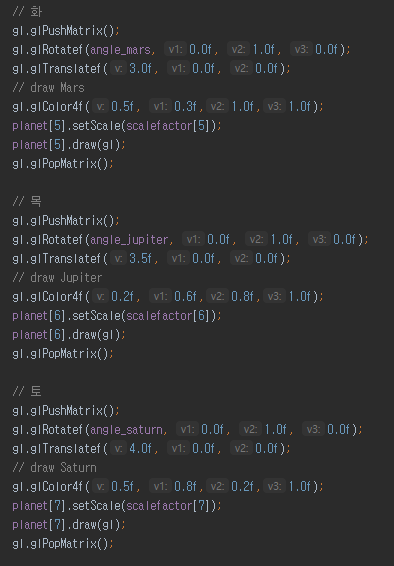
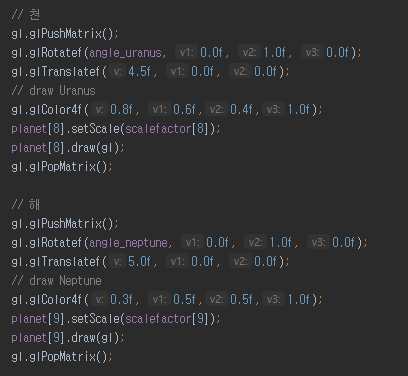
=> 행성들을 직접 그려낼 때 쓰이는 onDrawFrame 함수이다.

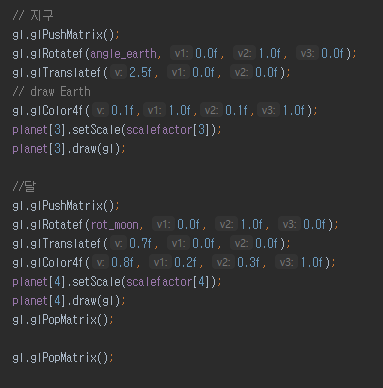
=> glClear, glClearColor, glClearDepthf 를 이용해서 view를 나타낼 때마다 color, depth, buffer를 모두 초기화한다. gluLookAt은 up vector를 계산하고, 카메라의 위치를 지정한다. Texture\_on\_off는 TEXTURE 버튼을 누르면 texture을 전환해 토글시키기 위해서 if-else문으로 코딩한다.

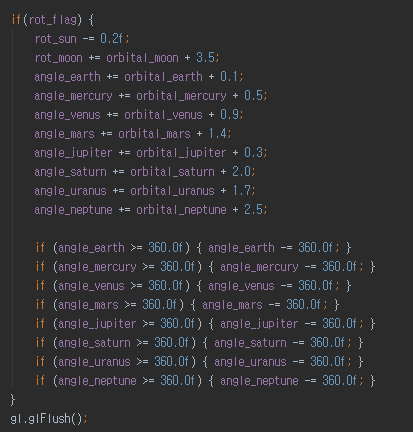


=> 화면에 카메라가 보는 행성을 하나씩 추가한다. 태양부터 시작해서 수성, 금성 등 행성을 순서대로 화면에 위치시키고 각 행성 별로 자전과 공전을 하도록 코딩해야 하기 떄문에 계층적인 modeling transtorm 이 필요했다. 따라서 이 때에는 glPushMatrix를 이용했다. 지구를 그리고, 그 지점에서 상태를 push 한 뒤에 transforming 으로 rotate, translate, color 변환을 한 뒤에 setscale로 크기 설정을 하고, draw를 한다. 그리고 다시 pop을 해서 이전 변환만 영향을 미치도록 한다. Push 와 pop 사이에 있는 변환들은 효과를 상실하게 되고, 따라서 중간 변환을 임시 저장하고 다시 행성을 추가할 수 있는 위치로 회귀하는 원리이다.

따라서 각 행성을 PushMatrix로 push 한 이후에, Rotatf 로 angle만큼 자전하고, translatef로 태양을 기준으로 거리를 이동시킨 뒤, Color4f로 색깔을 지정해준다. 그리고 setScale로 크키를 지정하고, draw한 뒤에 다시 PopMatrix로 pop을 해서 한 행성에 대한 내용을 정의해주는 것이다.

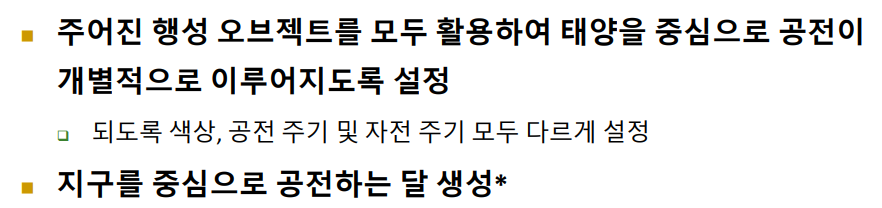
=> 화성, 목성, 토성, 천왕성, 해왕성도 순서대로 위와 같이 push, transforming, pop을 반복하면서 화면에 나타낸다.

=> 지구 주위를 공전하는 달을 나타내기 위해서, 지구를 먼저 push한 다음에 변환을 하고 draw를 한다. 그리고 pop을 하기 전에 달을 push 해서 그려주고, 변환과 draw를 한 뒤 pop을 해서 달이 자전하면서 공전을 하도록 먼저 설정을 한다. 그 후 마지막으로 pop을 한번 더 해서 지구에 대한 draw를 마친다. 따라서 지구와 달을 순서대로 그리고, 그 다음 화성을 그릴 수 있는 상태로 setting 이 된다.



=> rot\_flag에 따라서 ROTATION 버튼을 누르면 행성들이 공전, 자전을 하고 한번 더 누르면 회전을 멈추도록 토글한다. 각 행성의 공전 속도를 모두 다르게 정의하기 위해서 angle 값에 orbital 값을 다르게 설정했다. 또한 자전을 정의하기 위해서 각 angle이 360도 이상이면 다시 처음으로 돌아와서 회전할 수 있도록 설정했다.

**[ 실습 결과 및 고찰 ]**



=> 이번 실습을 통해서 modeling transformation에 대해서 배울 수 있었다. 객체는 맨 처음에 각각의 local 좌표계에 표현되어 있다. 하지만 기하적으로 변환을 해서 최종적으로 world 좌표계에 표현을 해야 여러 물체의 좌표계가 일치되고, 한 장면에 여러 물체를 모아 놓을 수 있다. 따라서 이를 위한 기하적 변환을 하는 과정을 modeling transformation이라고 한다. Pipeline Architecture에서 가장 첫 번째 단계라고 할 수 있다. 각 모델별로 가지고 있는 좌표가 local 좌표계이다. 따라서 modeling transformation 을 통해서 world 좌표계에 표현해서 공통된 좌표 내에 배치를 한다. glScale, glRotatef, glTranslatef 등의 함수가 있으며 크기변환, 회전, 평행이동을 통해서 local 좌표계를 world 좌표계로 변환한다. 코드 상에서 glRotatef로 자전, 공전을 할 수 있도록 했다. 또한 glTranslatef로 태양에서 부터의 거리를 다르게 해서 태양계 행성을 배치했다. 또한 matrix postmultiplication으로 구현했다. 따라서 코드에서 glRotatef, glTranslatef 순서대로 썼기 때문에 실제로는 translate가 먼저, rotate가 나중에 일어나는 것이다.

그리고 각 행성을 그리기 전에 onDrawFrame에서 gluLookAt( )을 호출하는데, World 좌표계에 표현된 물체를 Camera 좌표계로 보기 위해서 viewing transformation을 한다. Pipeline의 두번째 과정인 viewing transformation 이다. 이는 카메라를 기준으로 eye, center, up을 맞춰주는 것이다.

태양계를 나타내기 위해 여러 개의 행성을 그려야 했기 때문에 각 행성의 공전과 자전을 그린 후에 다시 그 행성의 좌표계로 돌아와서 새로운 다음 행성을 그렸다. 따라서 특정 local 좌표계로 반복해서 회귀할 수 있는 Matrix stack 의 push, pop을 이용했다. 행렬의 곱들은 matrix stack에 쌓여 있다. 따라서 push를 하면 이전까지 변환이 stack에 차례로 담긴다. 그 후 pop을 하면 그 전의 변환들의 효과는 사라지고, push 전에 있던 변환만 영향을 미치게 되는 것이다.