**Student ID: 2016025105**

**Name: 강재훈**

1. **Introduction to the program, a brief description of how to run it, and how to use it.**

제가 구현한 코드의 내용은 고양이 놀아주기입니다. 고양이가 main object이고 나머지 사물들과 고양이가 좋아하는 작은 공으로 프로그램이 이루어져있습니다. 먼저 고양이는 상하좌우앞뒤로 translation 할 수 있고 y축을 기준으로 rotation 할 수 있습니다. 또한 size up과 size down이 가능하며 x축에 대해 shear 할 수 있습니다. 마지막으로 xy평면에 대해서 reflection 할 수 있습니다. 이러한 각각의 동작은 key\_call\_back 함수에 의해 구현되었습니다. 또한 고양이는 자체적으로 가지고 놀 공을 갖습니다. 이 공은 적절하게 scaling, rotate, translation 됩니다. (extra credit 1) 각각의 사물들에 대해 설명하겠습니다. 먼저 실행 시 좌측에 있는 육면체는 원점으로부터 translation 한 뒤 시간값 t를 받아 z축에 대해서 cos(t) 만큼 자체적으로 회전을 합니다. 그리고 main object인 고양이가 움직일때마다 추가적으로 자신의 local frame에 대해 x축으로 cos(t) 만큼, z축으로 cos(t) 만큼 translation 합니다. 그리고 이 object는 hierachical하게 자식을 갖습니다.(extra credit 1) 이 자식은 고양이가 가지고 노는 공으로서 적절하게 scaling, rotate, translation 됩니다. 실행 시 오른쪽에 있는 사물도 왼쪽과 마찬가지로 자체적인 움직임과 main object가 움직일때에 따른 움직임, hierarchical 하게 자식을 갖습니다. (extra credit 1) 마찬가지로 이 부분의 자식도 적절한 scaling, rotate, translation을 합니다. 마지막으로 처음 실행 시 고양이 주변을 도는 구 모형을 볼 수 있는데 이 모형의 움직임은 curve로 표현됩니다. (extra credit 2) 이 object 역시 hierarchical 하게 자식을 갖습니다. (extra credit 1) 이 외에도 우리가 여태까지 사용해왔던 카메라 뿐만 아니라 first person view, quarter view를 구현하였고 자세한 구현방법은 2번에서 설명하겠습니다. 또한 lighting과 shading을 구현했습니다. 빛은 총 4개로 설정했고 그 중 2번째 light는 시간값 t를 받아 pos를 옮기며 animate합니다. defalut로 smooth shading을 채택했고 s키를 입력 시 flat shading으로 전환할 수 있습니다.

1. **Description of the implementation of each requirement:** 1) How you implemented that requirement, 2) How TA can verify that the requirement has been implemented, by performing a program's specific feature.. **Requirement items not implemented should be left blank.**
   1. (15 pts) Draw 3 or more 3D objects.

class assignment 2에서 사용한 코드를 재사용했습니다. 이전에는 drag and drop으로 obj파일을 불러와 정보를 읽어들였다면 이번에는 상대주소 값을 이용해 파일을 읽어오고 flat shading과 smooth shading을 모두 구현하기 위해 전역변수로 선언한 VertexArrayIndexed, VertexArraySeperate, NormalArray에 각 값을 저장했습다. 이후 render 함수에서 각 object들을 그리며 extra credit 1의 hierarchical 모델을 제외하고서는 각각에게 영향을 주지 않기 위해 push와 pop을 적절히 사용하며 modeling을 했습니다. (main object가 움직일때 다른 object가 그에 반응하여 움직이는 움직임은 key\_call\_back 함수에서 따로 matrix를 만들어 그 object를 modeling 할 때 따로 사용하였습니다.) 그렇게 고양이(main object)와 실행시 보이는 양 옆의 육면체 (sub1, sub2)를 구현하였고 그 외의 작은 sphere는 extra credit 1을 위해 구현했고, curve 형태를 그리며 움직이는 큰 sphere는 extra credit 2를 위해 구현했습니다.

* 1. (15 pts) One of the objects should be "main object". The user should be able to transform the main object using the mouse or keyboard.

key call back에서 입력받은 값을 토대로 main object가 transform합니다. main object는 자신의 local frame에 대해 transform 해야 하기 때문에 main object의 transform을 저장할 전역변시 mv\_matirx를 선언하고 key call back 함수의 지역변수로 tmp\_mat을 선언해 각 입력에 대해 알맞은 조치를 취해줍니다. 이후 입력에 대한 정보를 tmp\_mat에 저장하였다면 mv\_matrix의 오른쪽에 곱해줘 local frame에 대한 움직임을 구현합니다. main object가 가능한 transform은 다음과 같습니다.

* + - * 1. 앞으로 전진 – keyboard u
        2. 뒤로 후진 – keyboard j
        3. 오른쪽으로 이동 – keyboard k
        4. 왼쪽으로 이동 – keyboard h
        5. 위로 이동 – keyboard n
        6. 아래로 이동 – keyboard m
        7. y축을 기준으로 양의 각도 회전 – keyboard y
        8. y축을 기준으로 음의 각도 회전 – keyboard i
        9. object size up – keyboard o
        10. object size down – keyboard p
        11. 양의 x축으로 shear – keyboard v
        12. 음의 x축으로 shear – keyboard b
        13. xz평면에 대한 reflect – keyboard g
  1. (15 pts) The rest of the objects should be automatically moved in response to the movement of the main object or other objects around it.

main object와 extra credit을 제외하고 제가 구현한 object는 2개입니다. 실행시 먼저 왼쪽에 있는 object는 자체적으로 움직이고 main object가 움직일때 그에 따라서도 움직입니다. Translation과 rotation으로 자체적으로 움직이고 glMultMatrix로 key\_call\_back 함수에서 main object가 움직일 때마다 새로운 움직임을 저장한 mv\_sub1 행렬을 곱해줘 main object가 1번 움직일 때마다의 추가적인 움직임을 구현해주었습니다. 이 object는 main object가 움직일때마다 기본 움직임에서 추가적으로 y축, z축으로 각각 cos(t) (t는 시간값) 만큼 translation 합니다. 실행시 오른쪽에 보이는 object 역시 앞에서 설명한 object와 마찬가지로 본인의 움직임과 main object가 움직였을때 추가적인 움직임을 갖습니다. 구체적인 구현원리는 같습니다.

* 1. (15 pts) Use perspective projection. The camera should be able to be switched between two modes:

우리가 기존에 사용했던 카메라와는 다른 시점의 카메라 모드 2개를 구현해야 했습니다. 첫번째는 first-persion view입니다. first-persion vies는 FPS 게임에서 많이 사용되는 1인칭 시점의 카메라 모드입니다. 두번째는 고전게임에서 많이 사용되는 quarter view 모드 입니다. 일단 편의를 위해 남겨둔 기존의 카메라 모드(defalut)와 first-persion view, quarter view 중 mode를 선택하기 위해 select\_view 함수를 만들었습니다. select\_view함수는 인자로 들어온 select\_v 인자의 값에 따라 세 mode중 하나를 선택해 실행합니다. select\_v의 값은 key\_call\_back 함수에서 정해지는데 키보드 8을 누르면 default mode, 9를 누르면 first-persion view, 0을 누르면 quarter view 입니다. default 모드는 기존에 사용해왔던 코드를 재사용하였습니다. first-person view는 gluLookat함수를 이용해서 구현하였습니다. gluLookat은 카메라의 위치, 카메라가 보고 있는 곳, 카메라의 노말벡터로 이루어집니다. first-persion view에서는 main object가 translation 한 만큼 카메라의 위치를 옮겨주었고 main object의 모든 움직임을 반영하여 원래 보고 있던 at을 바꿔주었습니다. (mv\_matrix@at) 마지막으로 카메라의 노말벡터를 구해주기 위해 key\_call\_back 함수에서 main object가 rotation, reflect 할때마다 그 정보를 저장한 cam\_matirx를 만들어 노말벡터를 완성해주었습니다. (cam\_matrix@up). quarter view 모드 역시 gluLookat 함수를 이용해 구현하였습니다. 카메라의 위치에 main object의 translation 정보와 적절하게 위에서 내려다 볼 수 있는 값을 더해서 정해주었고, 어디를 볼지는 main object가 translation 한 곳의 xy 평면의 점을 내려다 볼 수 있게 해주었습니다. 마지막으로 노말벡터는 0, 0, 1의 값을 고정적으로 주었습니다.

* 1. (15 pts) Use OpenGL lighting / shading.

각각의 물체마다 다른 색상을 주었습니다. main object는 (1,1,1,1)을, 실행시 왼쪽에 보이는 object에는 (0,1,1,1)을, 오른쪽에 보이는 object에는 (1,1,0,1)을, 마지막으로 extra credit 2를 구현하기 위해 만든 object에는 (1,0,1,1)의 색상을 주었습니다. light는 class assignment 2 에서와 마찬가지로 4개의 빛을 사용하였고 그중 두번째 빛은 시간값 t를 받아 animate하게 움직이게 만들었습니다. flat shading과 smooth shading 중 한개만 구현하면 되었지만 이전 과제에서 구현한 코드가 있었기에 두 mode 모두 가능하게 만들었고 s키를 누르면 mode가 바뀝니다. ( default는 smooth mode) normal vector는 class assignment2에서 구현했던 것과 마찬가지로 obj 파일을 읽어오며 생성하였습니다.

* 1. Extra credit
     + 1. Use an animatinghierarchical model composed of multiple meshesas an object, for allobjects

hierarchical 모델을 만들어주기 위해 위에서 object들을 만들었 던 것과 같이 obj 파일을 읽어 필요한 값들을 저장해주었습니다. 그리고 기존 object들이 render에서 그려지는 공간 (glPushMatrix, glPopMatrix 사이) 에 부모 object의 modeling을 모두 끝낸 후 새로운 glPushMatrix, glPopMatirx 블록을 만들어 해당 hierarchical 모델을 modeling 해주었습니다. main object, sub object 1, sub object2, 그리고 extra credit2를 위한 object 모두에게 hierarchical한 자식 을 만들어주었습니다.

2. Use curves to express the movement of at least one object.

제가 curve를 사용해 움직임을 구현한 object는 실행시 x,y 평면에서 곡선을 그리는 object입니다. 다른 object와 마찬가지로 obj 파일을 읽어 필요한 정보를 저장하였습니다. 그리고 render 함수에서 curve를 그리기 위한 4점(p0, p1, p2, p3)을 지정 해준 뒤 수업시간에 배운 Bezier curve 공식을 이용하여 곡선이 그려질 점 P(t) (제 코드에서는 t는 th입니다.)를 찾습니다. 그리고 그 점의 위치로 해당 object를 translation 해주어 곡선을 따라 object가 움직이게 해주었습니다. t는 0<=t<=1인 값으로써 count를 사용하여 0에서부터 계속 증가하는 t 값을 시기적절하게 0으로 초기화 해줍니다. 처음 4개의 점 (p0,p1,p2,p3)는 모두 다른 점이었지만 p0와 p3가 달라서 한번 커브를 그리고 끊기고 다시 시작하는 점이 보기 좋지 않아 p0와 p3를 연결해 이어지는 커브 모양을 구현하였습니다.