20140559 조형준, HW3

1. Modeling Your Floppy cube

   3*3*1 cube전체를 포괄하는 cubebaseRbt를 잡았고 이를 기준으로

   (-1,0,-1) (0,0,-1) (1,0,-1)

   (-1,0,0) (0,0,0) (1,0,0)

   (-1,0,1) (0,0,1) (1,0,1)

   의 좌표를 갖는 9개의 rubic을 잡았고, 각각의 objectId들 1부터 9까지 값으로 초기화했습니다 (각각의 (-1, 0, -1) 부터 (1,0,1)까지의 좌표값은 glm::vec3 cubeposition[9]에 저장하였습니다.)

2. Design Floppy cube Data Structure

   cube가 회전하면서 계속 각각 rubic들의 위치가 바뀌기 때문에, 현재 rubic의 배치를 저장하는 int cube_curr[9]이라는 배열을 만들어, 0부터 8까지의 값으로 i번째 위치에 objectID로 cube_curr[i]+1의 값을 갖는 rubic이 위치한다는 것을 저장하였습니다.

   가장 가운데 rubic을 기준으로 각각의 rubic들은 [-1,1] * [0,0] * [-1,1]사이의 좌표를 갖는데, 이를 glm::mat4 cubeobjectRbt[9]를 이용해 저장하였습니다. 그리고 실제 rubic들의 위치를 나타내기 위해 glm::mat4 cubeRbt[9]를 잡았고 cubeRbt[i] = cubebaseRbt * cubeobjectRbt[i]을 계속해서 적용하였습니다.

   또 glm::mat4*의 type을 갖는 top[3], left[3], bottom[3], right[3]을 선언해 각각 cube_curr의 (0,1,2), (0,3,6), (6,7,8), (2,5,8)에 위치한 rubic(i)들의 cubeobjectRbt[i]의 주소를 담도록 했습니다.

   double rotations[6]은 flat (cube를 평평하게 맞추는 함수)를 실행하기 전까지 top, left, bottom, right, middle (가로, 세로)의 회전 각을 저장하고, 이를 flat을 실행할 때 사용하였습니다.

   glm::vec3 axis[4]에 각각 x축, -x축, z축, -z축을 저장해 rotation을 진행할 때 적절한 axis를 찾아 회전을 진행하도록 하였습니다.

void Floppy_drawPicking()과 void Floppy_draw()는 각각 drawPicking과 draw를 모든 rubic에 대해 적용하는 함수입니다.

bool Floppy_rotation_0_(int I, int j)과 bool Floppy_rotation_180_(int i, int j)은 각각 rotations[i]와 rotations[j]의 차이가 0에 근접한지, 180에 근접한지 (오차 10)를 판단하여 근접하다면 true, 아니라면 false를 반환하는 함수입니다.

bool Floppy_valid()는 현재 Floppy cube가 회전 가능한 상태인지 확인하는 함수로, 현재 가로축을 회전하려 하면 세로축들이 서로 10도 이하로 뒤틀린 상태인지 확인하고, 세로축을 회전하려 한다면 가로축들이 서로 10도 이하로 뒤틀린 상태인지 확인합니다.

void Floppy_set_flat은 rotation을 진행하기 전에 Floppy_valid()가 true일 때 진행되는 함수로 Floppy cube를 정확히 평평한 상태로 만들어주는 함수이며, 가운데 줄을 기준으로 정렬시킵니다. 이 함수를 구현할 때 회전을 통해 구현하게 되면 아무래도 실수 연산이다 보니 조금씩 오차가 생기는 것을 확인하여 각 큐브(i)의 cubeobjectRbt[i]에서 mat4의 원소들을 모두 확인해 -0.8미만의 값은 -1.0으로, -0.2 ~ 0.2사이의 값은 0.0으로, 0.8초과의 값은 1.0으로 설정했습니다 (각각 초기에 glm::translate(glm::mat4(1.0f), cubeposition[i])로 만들어졌고, 여기서 rotation만을 적용하였기에 모든 원소는 -1부터 1사이의 값을 갖는 것이 자명합니다.) 그리고, flat을 진행하면서 cube_curr도 재설정해주었습니다 (Floppy cube의 rotation이 가로축회전에서 세로축회전, 세로축회전에서 가로축회전으로 바뀔 때만 flat을 진행하므로 flat을 진행하기 전까지는 cube_curr을 재설정할 필요가 없습니다.)

void Floppy_set_current_layout()은 top, left, right, bottom을 재설정합니다.

3. Modify Arcball Interface

void Floppy_rotation(double xpos, double ypos)은 우선 Floppy_valid()을 확인하고, target이 1부터 9사이의 값인지 확인한 후에 selected_node를 target으로 저장하고, 만약 회전축이 가로에서 세로, 세로에서 가로로 바뀌었다면 Floppy_set_flat()을 실행합니다.

우선 회전은 마우스 왼쪽 버튼과 오른쪽 버튼으로 나누었고, 왼쪽버튼은 (초기상태에서) 세로축, 오른쪽 버튼은 (초기상태에서) 가로축을 회전하며, 회전시킬 때 마우스의 이동은 각 축의 중심 rubic을 기준으로 하여 (window의 xy좌표 평면에서) 마우스가 시계방향, 반시계방향으로 회전한 각도에 따라 축의 회전이 일어나도록 하였습니다. 만약 시계방향으로 회전했다면 x(z)축을 기준으로 회전, 반시계방향으로 회전했다면 -x(-z)축을 기준으로

회전 (θ > 0)하였습니다. 회전에는 quat를 사용하였습니다.

4. Creativity

Floppy cube 전체가 transition하고 rotation하는 것을 picking상태가(p를 짝수 번 눌렀을 때) 아닐 때 구현하였습니다.

void Floppy_rotation_all(double xpos, double ypos)은 HW2의 arcball interface를 그대로 이용해 cubebaseRbt를 회전시키고, 이후 각 rubic들을 cubeRbt[i] = cubebaseRbt * cubeobjectRbt[i]을 이용해 재설정 함으로써 적용하였습니다.

void Floppy_transition_xy(double xpos, double ypos)와 void Floppy_transition_xy(double ypos)역시 비슷한 방법으로 진행하였습니다.