# Assignment 3

# Moving Planet

Affiliation: Mathematics, CS

student ID: 2017313260

Name: Lee Jae Min

It started with "trackball" and last assignment. Same algorithm with last assignment, create_sphere_vertices, update_index_buffer. Of course, It has same variable.

1. Data Structure (Additions)

1.1 Declaration constant and variable

radius[], rot_speed[], rev_speed[] are radius, rotation speed, revolve speed of each planets.

au[] is distance from sun, each planets.

r is unit radius, multiply with radius[] for scaling. (different from last assignment)

Initial window size set 1280*720.

1.2 Function

2.1 main.cpp

create_sphere_vertices() is function that makes vertices.

In this assignment, vertex buffer should not be updated every time. So, update_index_buffer() is update index buffer, not vertex buffer.

update() function is update b_color_type, matrix in vertex/fragment shaders.

In render() function, update rotate matrix, and rendering planets.

mouse(), motion() function are handling mouse event.

2.2 trackball.h

update(), update_pan(), update_zoom() handling camera rotate event, pan event, zoom event.

2. Algorithm

2.1 render()

Use for statement, render each planets with attributes.

### 2.2 mouse(), motion()

Get mouse button, action and mods. And start callback function, handling events. Event handling functions are in trackball structures.

### 2.3 update_~()

Multiplication with view_matrix, translate matrix, scale matrix.

### 2.5 user_init (additions)

Vertex buffer should not be updated every time. So, update vertex buffer in this function. And get index buffer, load mesh.

### 2.6 keyboard (additions)

If press D, change color type to b_color_type ++.

When press PAUSE, if it was rotating, stored stopped time. If it was not rotating, add (current time – passed time) to t_gap (stationary time). After this, change rotate status.

When press HOME, camera structure initialize.

## 3. Discussions

There are many ways to handle mouse events. It can be changed depending on how it is used.