# CSCI3260
# Principles of Computer Graphics

---------Tutorial 11

CHEN Yukang

# Question in Course Project

- About project grouping
  - For those students still not in a team, each two of them are randomly assigned together.

# Question in Course Project

- About project grouping
  - For those students still not in a team, each two of them are randomly assigned together.
  - Please check the google sheet online.

# Question in Course Project

- About project grouping
  - For those students still not in a team, each two of them are randomly assigned together.
  - Please check the google sheet online.
  - It is still possible to change partners if necessary.

# Question in Course Project

- About project grouping
  - For those students still not in a team, each two of them are randomly assigned together.
  - Please check the google sheet online.
  - It is still possible to change partners if necessary.
  - Start to do the course project early.

# Tutorials in Nov

- Multiple textures / shaders
- Skybox
- Normal mapping
- Instance rendering
- Mouse / Keyboard interaction
- Viewpoint setting
- Visual feedback
- Collision detection

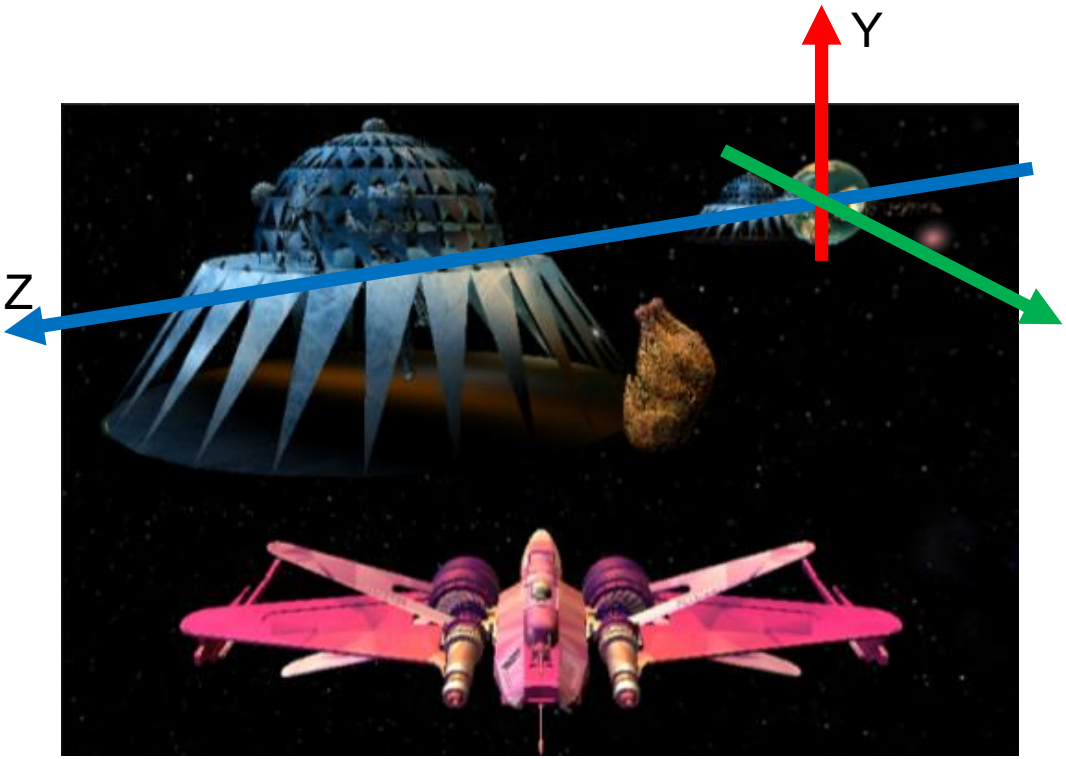
# Background

In the year 3020, you are driving a spacecraft to visit an alien planet. Friendly alien people line in the space with space vehicles and provide foods to welcome you. You need to visit each of them and collect their foods.

# Requirements

| | | |
|---|---|---|
| | **Basic (88%)** | |
| 1 | Render one planet, one spacecraft and at least three alien people in their space vehicle | 10% |
| 2 | Self-rotation for the planet and the alien space vehicle | 6% |
| 3 | Render a skybox | 6% |
| 4 | Basic light rendering | 4% |
| 5 | Render an asteroid ring cloud | 10% |
| 6 | The rotation of the rocks | 8% |
| 7 | Correct viewpoint | 8% |
| 8 | Use mouse to control the self-rotation of the spacecraft | 8% |
| 9 | Use keyboard to control the translations of the spacecraft | 8% |
| 10 | Collect foods | 8% |
| 11 | Change the texture of the alien vehicle after visiting | 8% |
| 12 | Change the texture of the spacecraft after the whole visiting | 4% |
| | **Bonus (12%)** | |
| 1 | Add another light source | 5% |
| 2 | Normal mapping for the planet | 5% |
| 3 | More kinds of foods to substitute the provided chicken. | 2% |
| | **Total:** | **100%** |

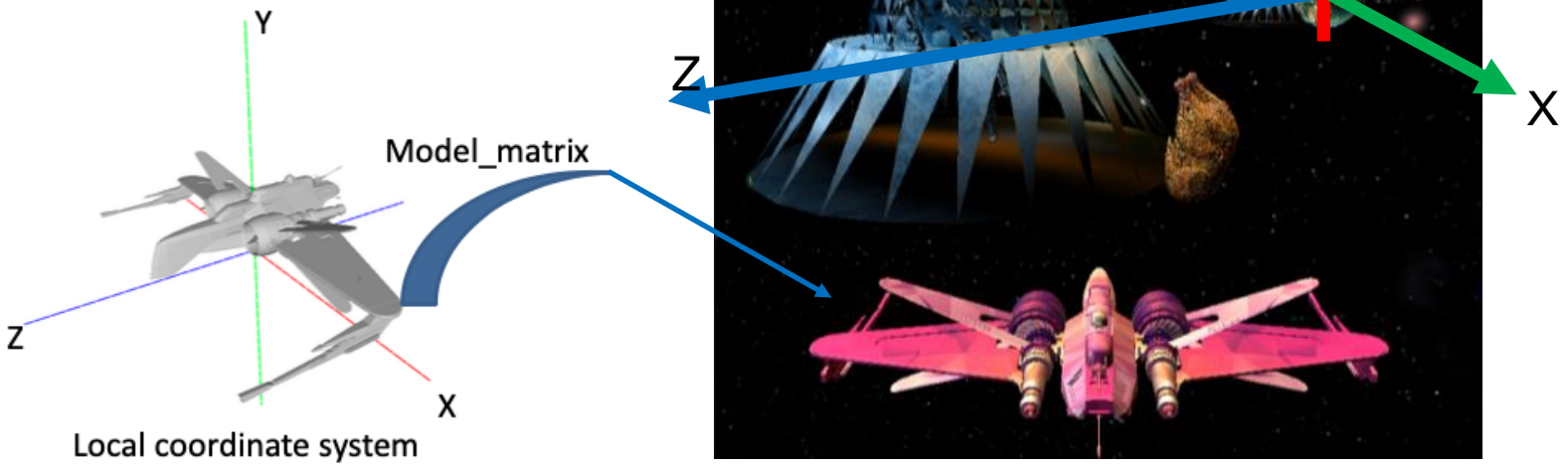
# Recommended layout




1. Put all models on XOZ plane.
2. Put the planet and aliens along Z axis.
3. Move the spacecraft freely on the XOZ Plane using mouse and keyboard.

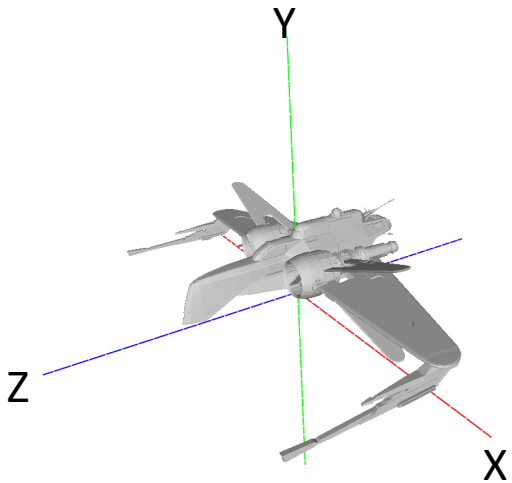
# Coordinate systems

1. Local coordinate system of the spacecraft.
2. World coordinate system.
3. Original is at the center of the planet.

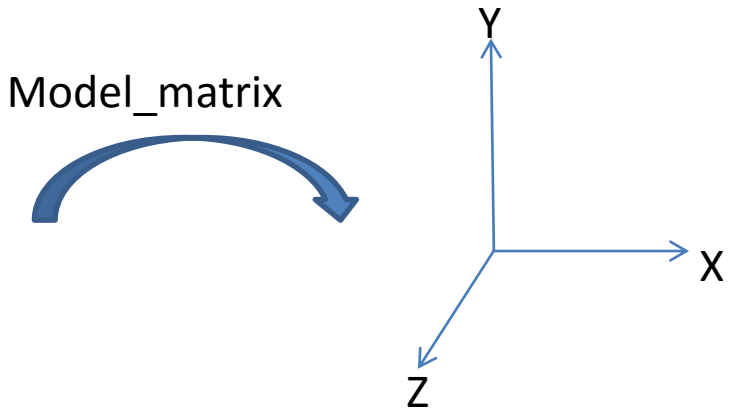


# Calculate the model matrix

- The model is defined with large scale, your may make the model smaller using scaling matrix
- Rotation and translation
- How to get? From the input of mouse and keyboard



Local coordinate system



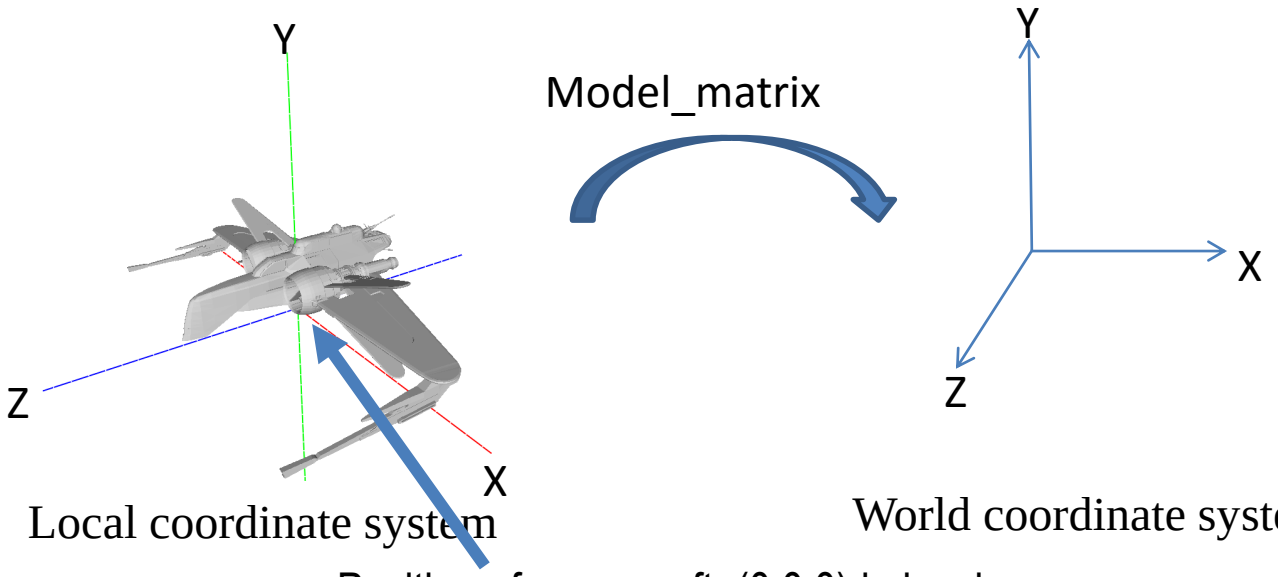
World coordinate system

```
Void UpdateStatus(){
float scale = 0.0005;
glm::mat4 SC_scale_M = glm::scale(glm::mat4(1.0f), glm::vec3(scale));
glm::mat4 SC_trans_M = glm::translate
(
glm::mat4(1.0f),
glm::vec3(SCInitialPos[0] + SCTranslation[0], SCInitialPos[1] + SCTranslation[1], SCInitialPos[2] + SCTranslation[2])
);
Model_matrix = SC_trans_M*SC_Rot_M*SC_scale_M;
}
```

Translation Rotation Scale

# Calculate the model matrix

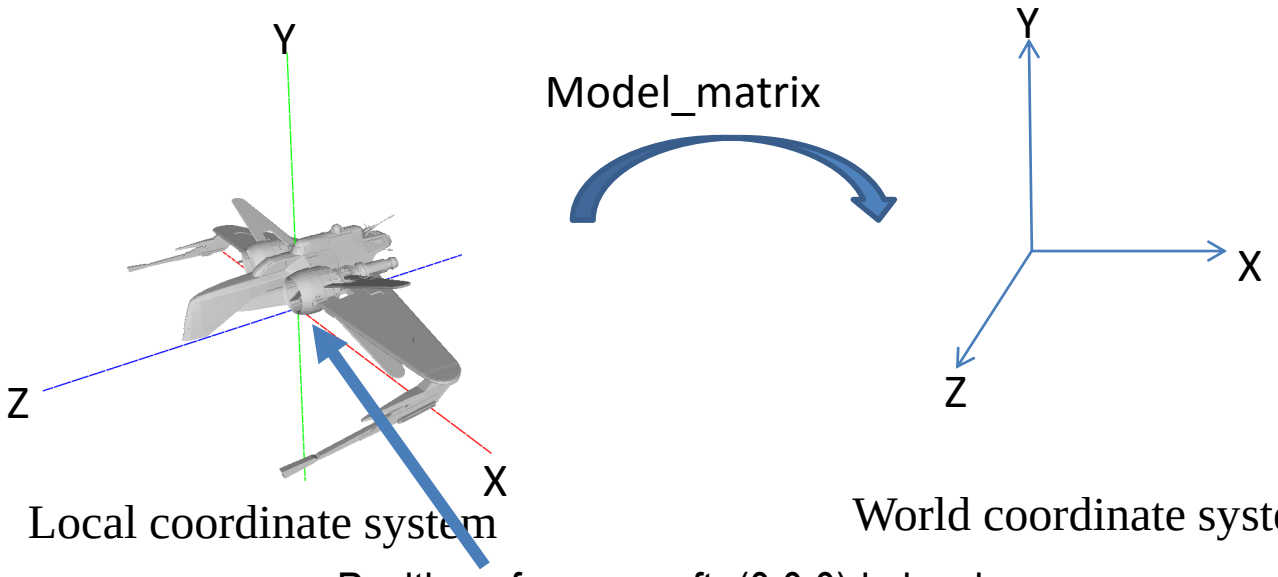
- The model is defined with large scale, your may make the model smaller using scaling matrix
- Rotation and translation
- How to get? From the input of mouse and keyboard




```
Void UpdateStatus(){
float scale = 0.0005;
glm::mat4 SC_scale_M = glm::scale(glm::mat4(1.0f), glm::vec3(scale));
glm::mat4 SC_trans_M = glm::translate
(
glm::mat4(1.0f),
glm::vec3(SCInitialPos[0] + SCTranslation[0], SCInitialPos[1] + SCTranslation[1], SCInitialPos[2] + SCTranslation[2])
);
Model_matrix = SC_trans_M*SC_Rot_M*SC_scale_M;
SC_world_pos = Model_matrix * glm::vec4(SC_local_pos, 1.0f);
}
```

# Calculate the model matrix

- The model is defined with large scale, your may make the model smaller using scaling matrix
- Rotation and translation
- How to get? From the input of mouse and keyboard




```
Void UpdateStatus(){
float scale = 0.0005;
glm::mat4 SC_scale_M = glm::scale(glm::mat4(1.0f), glm::vec3(scale));
glm::mat4 SC_trans_M = glm::translate
(
glm::mat4(1.0f),
glm::vec3(SCInitialPos[0] + SCTranslation[0], SCInitialPos[1] + SCTranslation[1], SCInitialPos[2] + SCTranslation[2])
);
Model_matrix = SC_trans_M*SC_Rot_M*SC_scale_M;
SC_world_pos = Model_matrix * glm::vec4(SC_local_pos, 1.0f);
}
```

# Control translation by keyboard

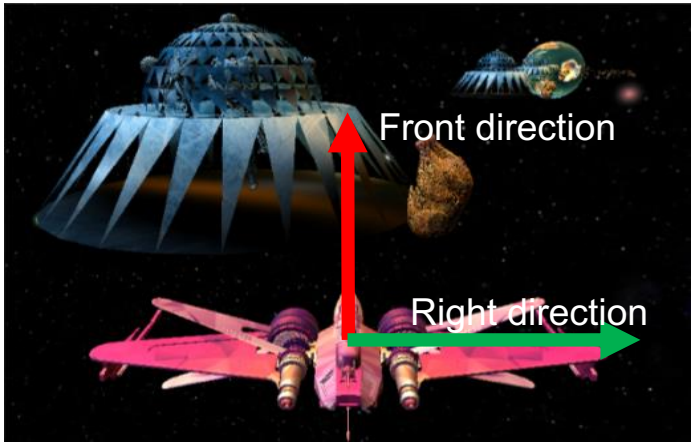
- Translate along the front and right directions of spacecraft. Get unit vectors of the directions in world space

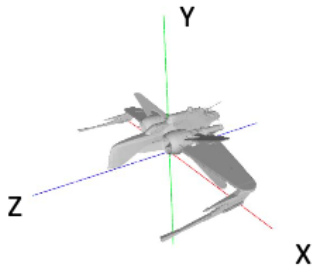
```
void UpdateStatus(){
float scale = 0.0005;
glm::mat4 SC_scale_M = glm::scale(glm::mat4(1.0f), glm::vec3(scale));
glm::mat4 SC_trans_M = glm::translate
(
glm::mat4(1.0f),
glm::vec3(SCInitialPos[0] + SCTranslation[0],
SCInitialPos[1] + SCTranslation[1], SCInitialPos[2] + SCTranslation[2])
);


Model_matrix = SC_trans_M*SC_Rot_M*SC_scale_M;
SC_world_pos = Model_matrix * glm::vec4(SC_local_pos, 1.0f);
SC_world_Front_Direction = Model_matrix * glm::vec4(SC_local_front, 1.0f);
SC_world_Right_Direction = Model_matrix * glm::vec4(SC_local_right, 1.0f);
SC_world_Front_Direction = Normalize(SC_world_Front_Direction);
SC_world_Right_Direction = Normalize(SC_world_Right_Direction);
}
```

- In main.cpp, register a callback function for keyboard cursor keys

```
int main(int argc, char *argv[])
{
…
glutSpecialFunc(SpecialKeys);
…
}
```



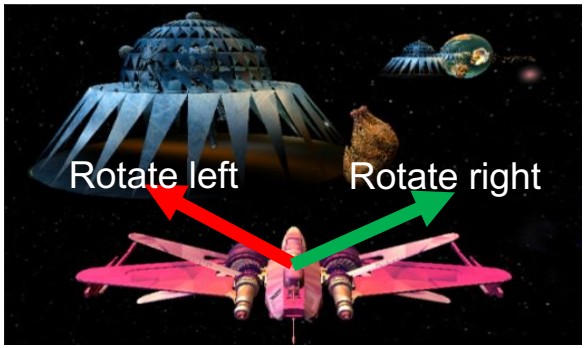

SC_local_front = (0,0,-1)
SC_local_right = (1,0,0)

# Control translation by keyboard

- Define the callback function to get the keyboard event and update

```
void SpecialKeys(int key, int x, int y)
{
    if (key == GLUT_KEY_DOWN)
    {
           SCTranslation[0] = SCTranslation[0] + 0.5* SC_world_Front_Direction[0];
           SCTranslation[2] = SCTranslation[2] + 0.5* SC_world_Front_Direction[2];
    }
 if (key == GLUT_KEY_UP)
    {
           SCTranslation[0] = SCTranslation[0] - 0.5* SC_world_Front_Direction[0];
           SCTranslation[2] = SCTranslation[2] - 0.5* SC_world_Front_Direction[2];
    }
    if (key == GLUT_KEY_LEFT)
    {
           SCTranslation[0] = SCTranslation [0] - 0.5* SC_world_Right_Direction[0];
           SCTranslation[2] = SCTranslation [2] - 0.5* SC_world_Right_Direction[2];
    }
    if (key == GLUT_KEY_RIGHT)
    {
           SCTranslation[0] = SCTranslation [0] + 0.5* SC_world_Right_Direction[0];
           SCTranslation[2] = SCTranslation [2] + 0.5* SC_world_Right_Direction[2];
    }
}
```

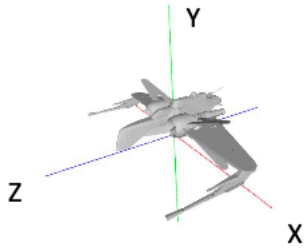
# Control rotation by mouse




- In main.cpp, register a callback function for mouse movement

```
int main(int argc, char *argv[])
{
…
glutPassiveMotionFunc(PassiveMouse);
…
}
```

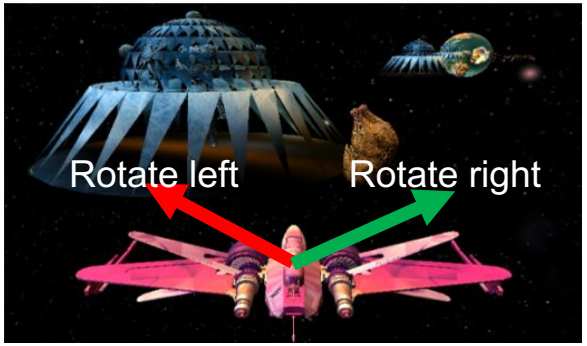
- Rotate around Y axis in local coordinate system

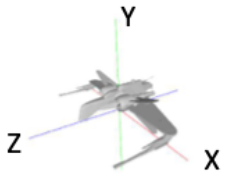



```
void UpdateStatus(){
float scale = 0.0005;
glm::mat4 SC_scale_M = glm::scale(glm::mat4(1.0f), glm::vec3(scale));
glm::mat4 SC_trans_M = glm::translate
(
glm::mat4(1.0f),
glm::vec3(SCInitialPos[0] + SCTranslation[0], SCInitialPos[1] + SCTranslation[1], SCInitialPos[2] + SCTranslation[2])
);
Model_matrix = SC_trans_M*SC_Rot_M*SC_scale_M;
SC_world_pos = Model_matrix * glm::vec4(SC_local_pos, 1.0f);
SC_world_Front_Direction = Model_matrix * glm::vec4(SC_local_front, 1.0f);
SC_world_Right_Direction = Model_matrix * glm::vec4(SC_local_right, 1.0f);
SC_world_Front_Direction = Normalize(SC_world_Front_Direction);
SC_world_Right_Direction = Normalize(SC_world_Right_Direction);
}
```

# Control rotation by mouse

Define the callback function for mouse

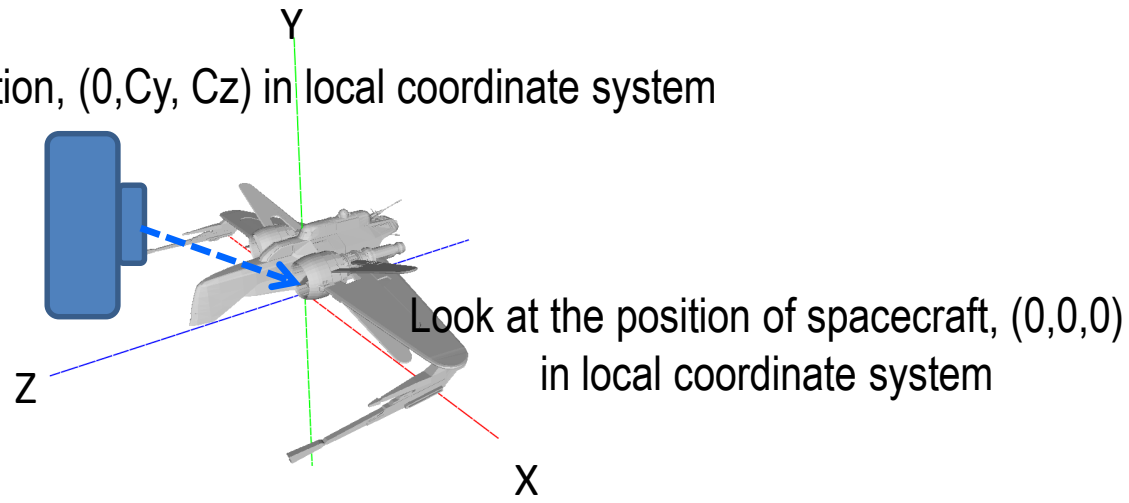



```
void PassiveMouse(int x, int y)
{
    if (x<oldx)
    {
        viewRotateDegree[1] += 1.0f;
        SC_Rot_M = glm::rotate(glm::mat4(1.0f), glm::radians(viewRotateDegree[1]),   glm::vec3(0.0f, 1.0f, 0.0f));
    }
    if (x>oldx)
    {
        viewRotateDegree[1] -= 1.0f;
        SC_Rot_M = glm::rotate(glm::mat4(1.0f), glm::radians(viewRotateDegree[1]), glm::vec3(0.0f,1.0f, 0.0f));
    }
    oldx = x;
}
```

# Viewpoint setting

- Fix the virtual camera relatively static to the spacecraft
- Set Cz, Cy by yourself

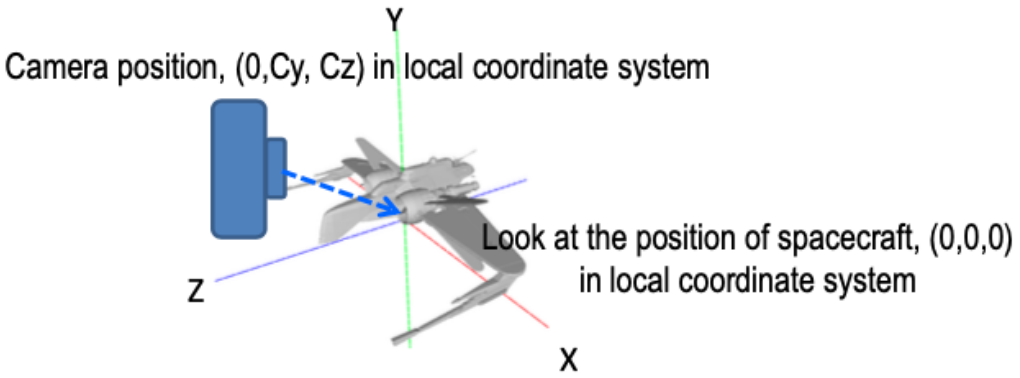


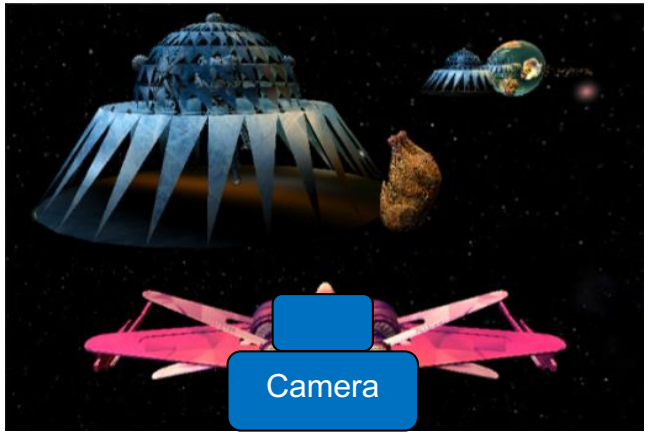

```
void UpdateStatus(){
float scale = 0.0005;
glm::mat4 SC_scale_M = glm::scale(glm::mat4(1.0f), glm::vec3(scale));
glm::mat4 SC_trans_M = glm::translate
(
glm::mat4(1.0f),
glm::vec3(SCInitialPos[0] + SCTranslation[0], SCInitialPos[1] + SCTranslation[1], SCInitialPos[2] + SCTranslation[2])
);
Model_matrix = SC_trans_M*SC_Rot_M*SC_scale_M;
SC_world_pos = Model_matrix * glm::vec4(SC_local_pos, 1.0f);
SC_world_Front_Direction = Model_matrix * glm::vec4(SC_local_front, 1.0f);
SC_world_Right_Direction = Model_matrix * glm::vec4(SC_local_right, 1.0f);
SC_world_Front_Direction = Normalize(SC_world_Front_Direction);
SC_world_Right_Direction = Normalize(SC_world_Right_Direction);
Camera_world_position = Model_matrix * glm::vec4(Camera_local_position, 1.0f);
}
```

# Viewpoint setting

- Calculate view matrix

```
ViewMatrix = glm::lookAt(glm::vec3(Camera_world_position), glm::vec3(SC_world_pos), glm::vec3(0.0f, 1.0f, 0.0f));
```



# Set perspective matrix

```
void paintGL(void)
{
    glClearColor(0.0f, 0.0f, 0.8f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // spacecraft modelling
    glm::mat4 translateMatrix = glm::translate(mat4(), vec3(2 + x_translate, 0.5, 15 +
        z_translate));
    glm::mat4 rotateMatrix = glm::rotate(mat4(), glm::radians(y_rotate), vec3(0, 1, 0));
    glm::mat4 scaleMatrix = glm::scale(mat4(), vec3(0.0005f, 0.0005f, 0.0005f));
    glm::mat4 spacecraftModel = translateMatrix * rotateMatrix;
    // world space modelling
    glm::vec4 camera = spacecraftModel * vec4(0.0f, 0.5f, 0.8f, 1.0f);
    glm::vec4 viewport = spacecraftModel * vec4(0.0f, 0.0f, -0.8f, 1.0f);
    // Projection matrix
    glm::mat4 projection = glm::perspective(glm::radians(45.0f), 4.0f / 3.0f, 0.1f, 500.0f);
    glm::mat4 view = glm::lookAt(glm::vec3(camera), glm::vec3(viewport), glm::vec3(0, 1, 0));
    // Model matrix
    glm::mat4 model;
    spacecraftModel = spacecraftModel * glm::rotate(mat4(), glm::radians(180.0f), vec3(0, 1, 0))
        * scaleMatrix;
```

# Next tutorials

- Multiple textures / shaders
- Skybox
- Normal mapping
- Instance rendering
- Mouse / Keyboard interaction
- Viewpoint setting
- Visual feedback
- Collision detection

# Requirements

| | | |
|---|---|---|
| | **Basic (88%)** | |
| 1 | Render one planet, one spacecraft and at least three alien people in their space vehicle | 10% |
| 2 | Self-rotation for the planet and the alien space vehicle | 6% |
| 3 | Render a skybox | 6% |
| 4 | Basic light rendering | 4% |
| 5 | Render an asteroid ring cloud | 10% |
| 6 | The rotation of the rocks | 8% |
| 7 | Correct viewpoint | 8% |
| 8 | Use mouse to control the self-rotation of the spacecraft | 8% |
| 9 | Use keyboard to control the translations of the spacecraft | 8% |
| 10 | Collect foods | 8% |
| 11 | Change the texture of the alien vehicle after visiting | 8% |
| 12 | Change the texture of the spacecraft after the whole visiting | 4% |
| | **Bonus (12%)** | |
| 1 | Add another light source | 5% |
| 2 | Normal mapping for the planet | 5% |
| 3 | More kinds of foods to substitute the provided chicken. | 2% |
| | **Total:** | **100%** |

# Thanks