Computer Graphic interim report

Progress:

Finish all features the proposal mention.

Include:

Project main target:

Each of the planets should revolution to the sun and rotate itself.

Each of the planets should have different surface.

The speed of revolution and rotate should be adjustable by some key. The camera should control by mouse or keyboard that user can change the perspective of the scene.

Project extra target:

There should have a background of the universe. Add extra satellites to the planets.

Extra features:

Use skybox as the background of the universe.

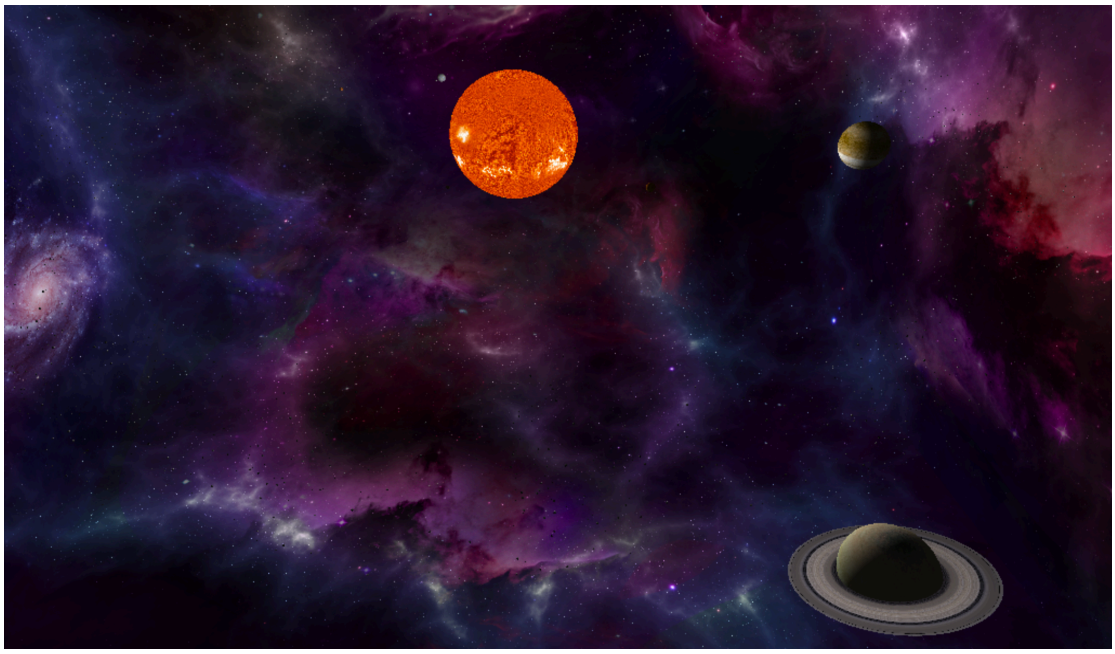
Implement particle system. (transparent texture, control of the particle active or not)

Add asteroid belt to the solar system.

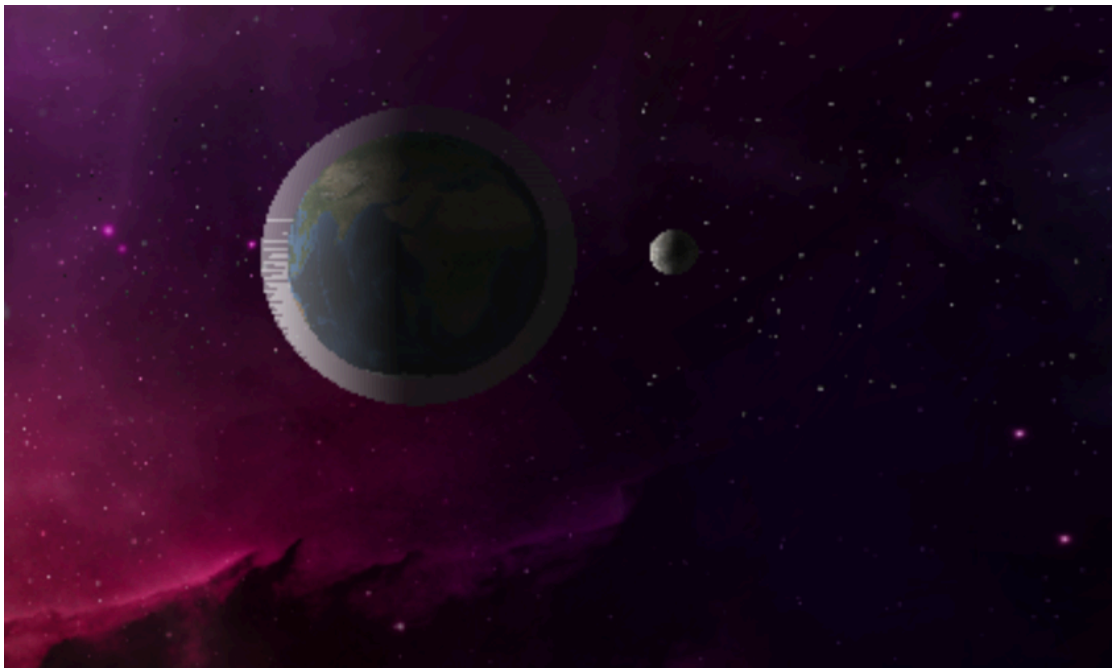
The atmosphere of earth.

Demo pic:

Whole scene:

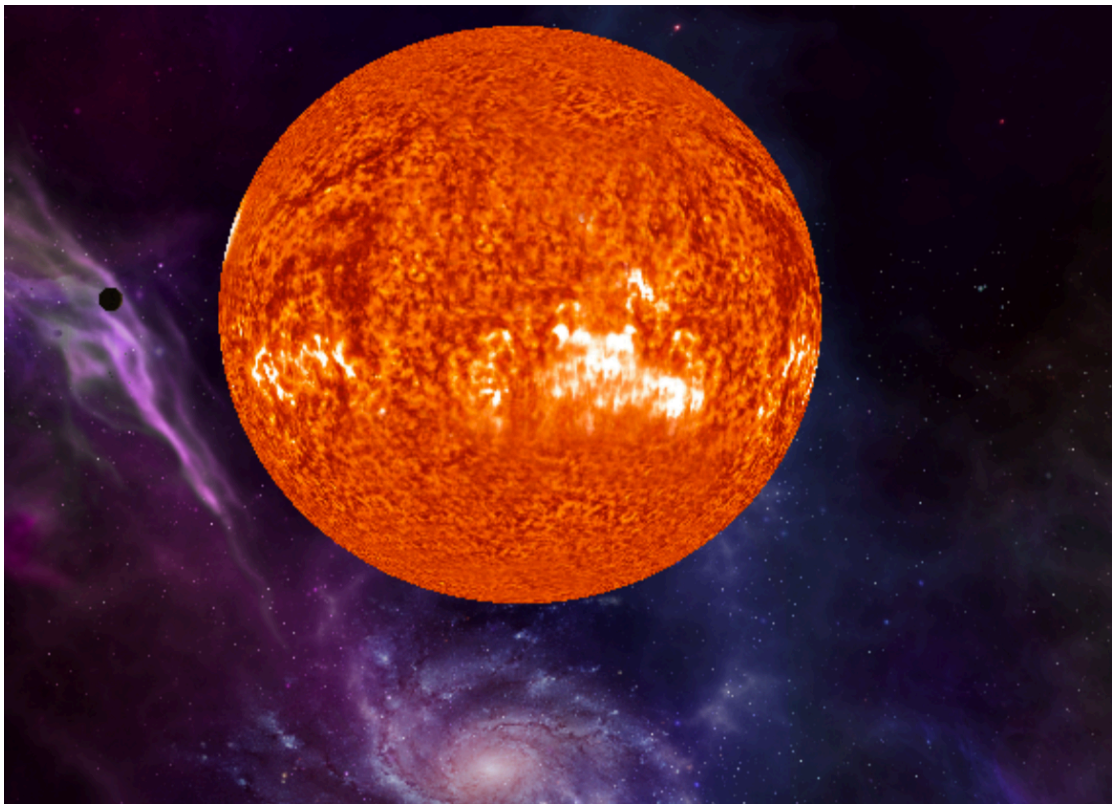


Earth atmosphere

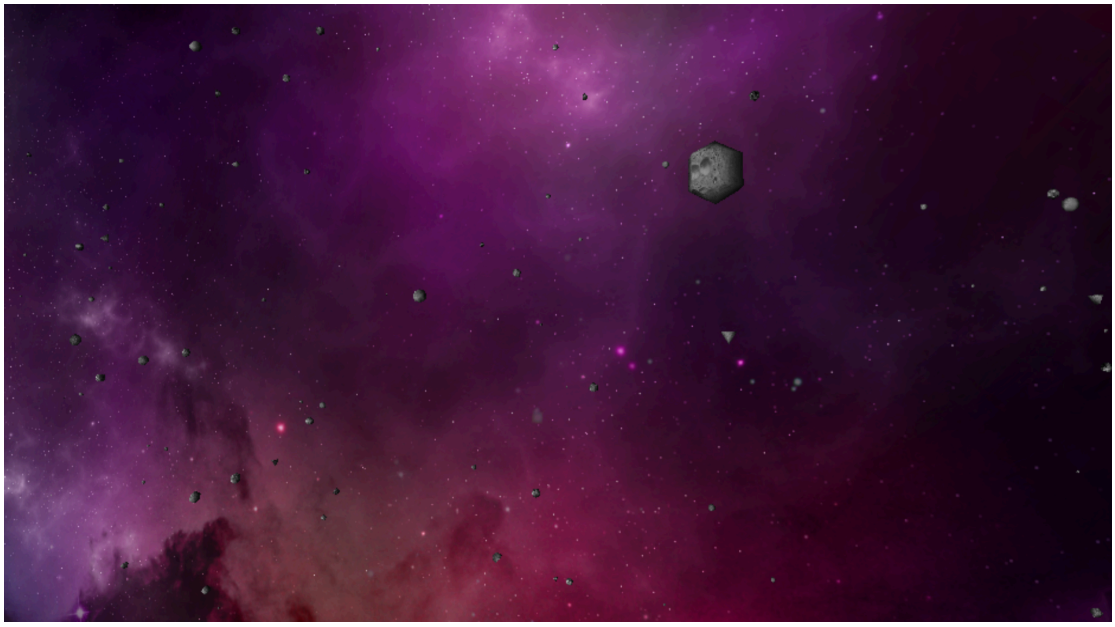


Lighting of planet:

The sun does not have shadow:



Asteroid belt: each asteroid has random shape

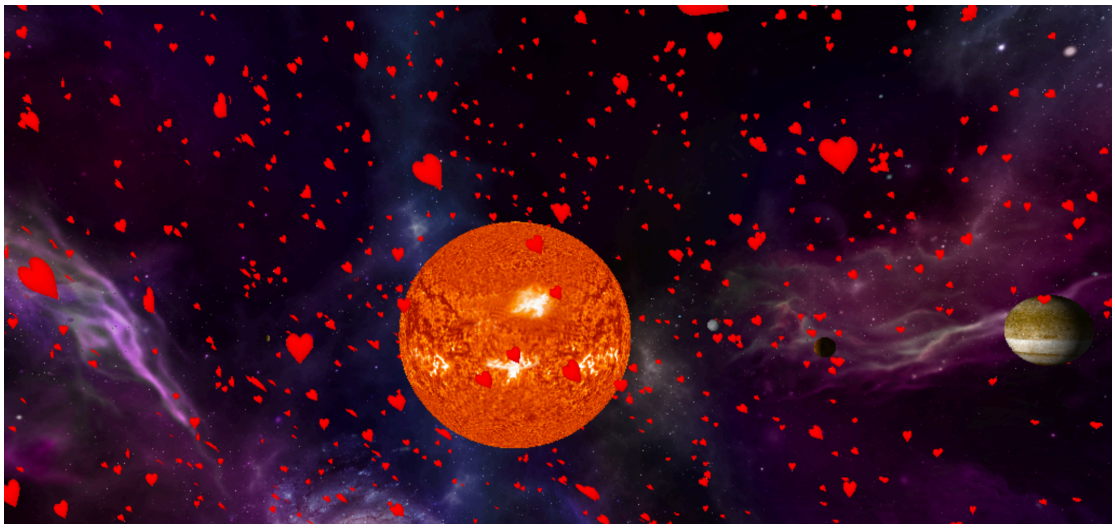


Another view of asteroid belt:

Saturn and Saturn ring:

Skybox：

Particle system:



Code:

Different class for the feature:

```
//
#include "baseheader.h"
#include "Texture.h"
#include "Draw.h"
#include "Planet.h"
#include "SpaceShip.h"
#include "math.h"
#include "ParticleController.h"
#define MAX_POS 1600
#define ASTEROID_NULBER 3000
```

Baseheader: include some base library

SpaceShip: the free camera controller, I intent to make a space ship model, so I abstract this class as SpaceShip.

```
void SpaceShip::RotateShip(float x, float y)
{
    yaw += x;
    pitch += y;
    if (pitch > 89.0f) {
        pitch = 89.0f;
    }
    if (pitch < -89.0f) {
        pitch = -89.0f;
    }
    if (yaw < 0.0f) {
        yaw += 360.0f;
    }
    if (yaw > 360.0f) {
        yaw -= 360.0f;
    }
    vec3 forward = vec3();
    forward.x = -sin(deg2rad * yaw) * cos(deg2rad * pitch);
    forward.y = sin(deg2rad * pitch);
    forward.z = -cos(deg2rad * yaw) * cos(deg2rad * pitch);
    forward = Normalize(forward);
    lookAt = pos + forward;
}
```

```
void SpaceShipControl(unsigned char key, int x, int y)
{
    vec3 buffer;
    switch (key)
    {
    case 27:        //ESC turn off space ship
        exit(0);
        break;
    case 'w'://go forward
        buffer = ship->forward();
        buffer *= shipSpeed;
        if (Length(ship->pos + buffer) >= MAX_POS) return;
        ship->pos += buffer ;
        ship->lookAt += buffer;
        glutPostRedisplay();
        break;
    case 's'://go backward
        buffer = ship->forward();
        buffer *= shipSpeed;
        if (Length(ship->pos - buffer) >= MAX_POS) return;
        ship->pos -= buffer;
        ship->lookAt -= buffer;
        glutPostRedisplay();
        break;
    case 'a'://go left
        buffer = ship->right();
        buffer *= shipSpeed;
        if (Length(ship->pos + buffer) >= MAX_POS) return;
        ship->pos += buffer;
        ship->lookAt += buffer;
        glutPostRedisplay();
        break;
    case 'd'://go right
        buffer = ship->right();
        buffer *= shipSpeed;
        if (Length(ship->pos - buffer) >= MAX_POS) return;
        ship->pos -= buffer;
        ship->lookAt -= buffer;
        glutPostRedisplay();
```