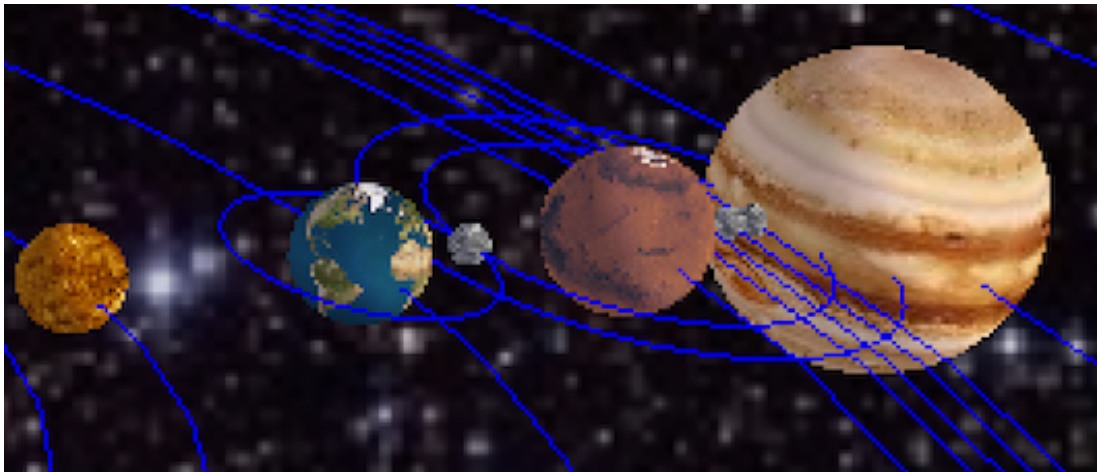
Deep Space Explorer

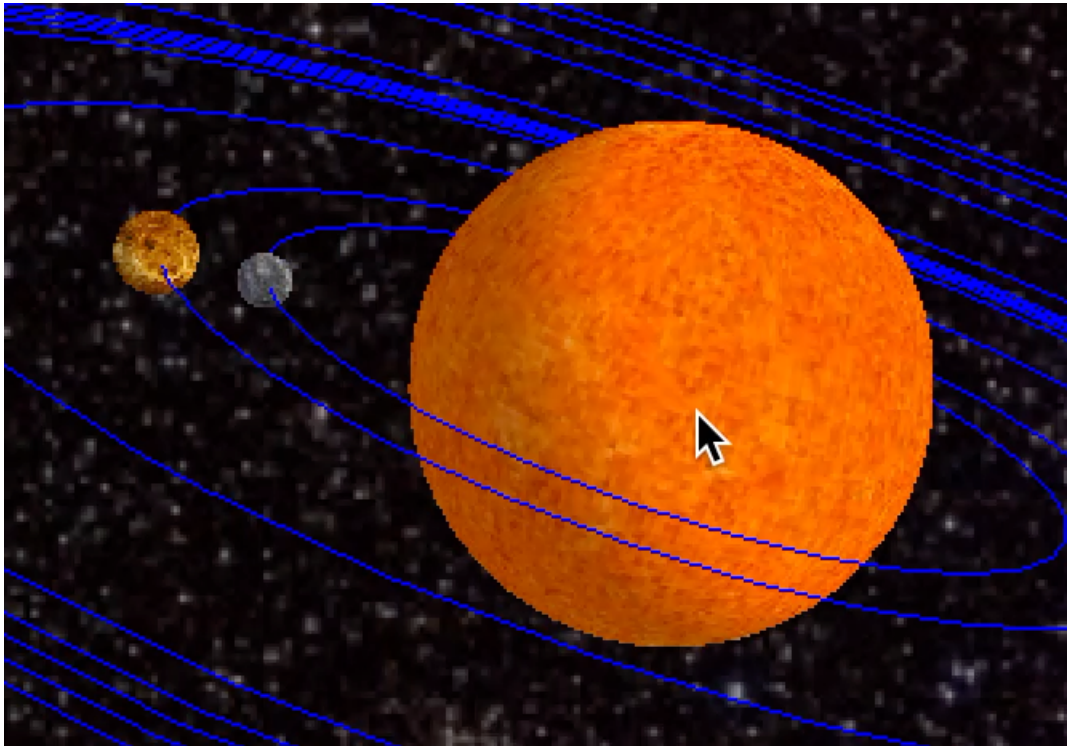
International Space App Challenge 2014

Fernanda Maria Grella
Raffaella Pivetta
Andrea De Lucia
Raffaele Scorrano
Saverio Tubito

13/04/2014



We developed this game for a primary reason change People's ideas about Asteroids . We see the Asteroids like a resource and not only a threat. For this reason we created this Game for adventures across the Solar System.

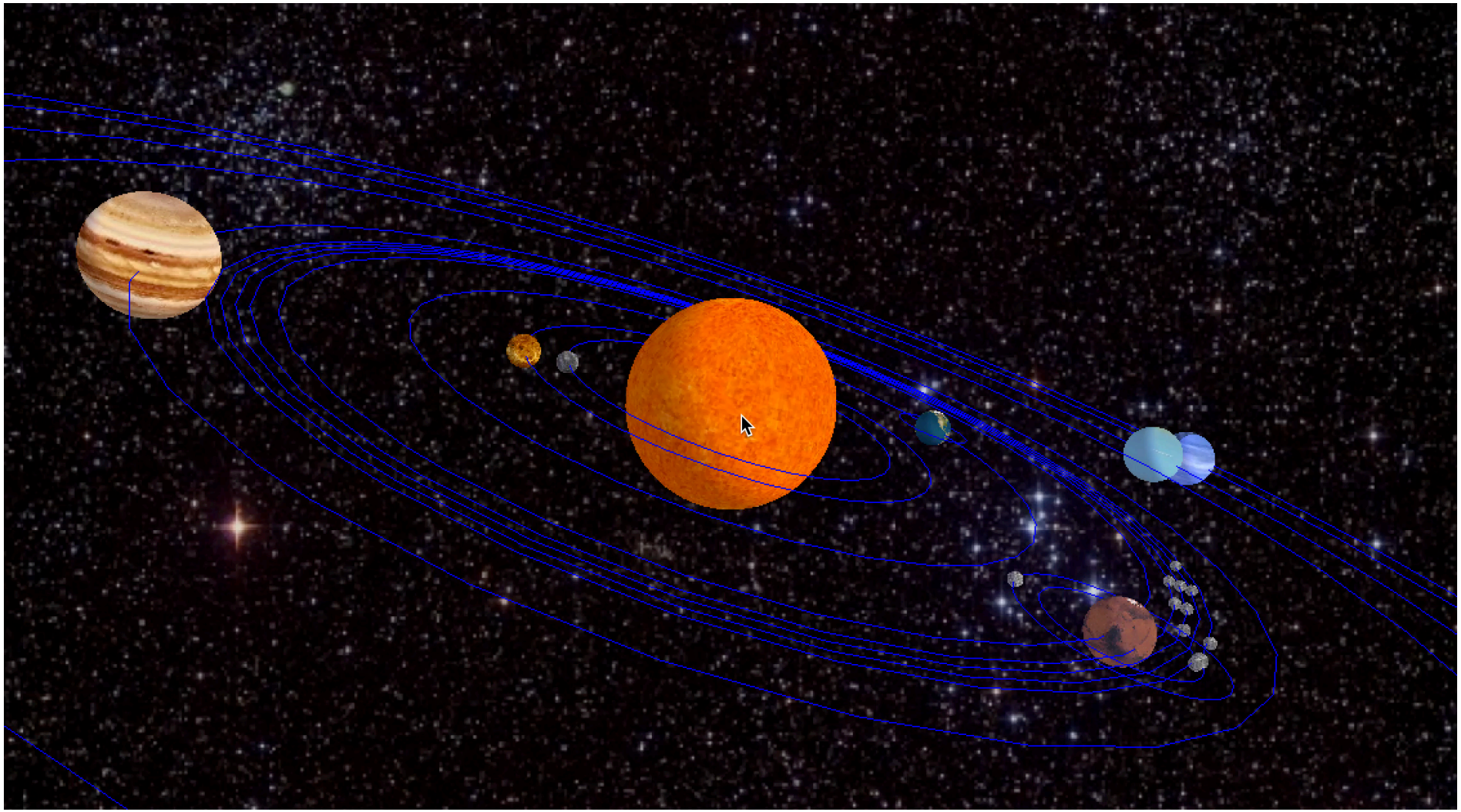


```
using namespace std;

class Planet
{
private:

    float _planet_radius;

    float _spinning_angle;
    float _spinning_angular_velocity;

    float _orbit_radius;
    float _orbit_angle;
    float _orbit_angular_velocity;

    Texture _planet_map;

    GLUquadricObj *quadric;

    float _orbit_radius2;
```

```
void Update()
{
    _spinning_angle += _spinning_angular_velocity;
    _orbit_angle += _orbit_angular_velocity;

    vector<Planet>::iterator pIt;
    for (pIt = _satellites.begin(); pIt != _satellites.end(); pIt++)
        pIt->Update();
}
```

```
void Draw()
{
    glColor3d(0.8, 0.8, 0.9);

    glBindTexture(GL_TEXTURE_2D, _planet_map.ID);

    glPushMatrix();
        glRotatef(_orbit_angle, 0, 1, 0);
        glTranslatef(_orbit_radius, 0, 0);

        glPushMatrix();
            glRotatef(_spinning_angle, 0, 1, 0);
            glRotatef(-90,1,0,0);
            gluSphere(quadric, _planet_radius, 32, 32 );    // sfera
        glPopMatrix();

        // disegna ogni satellite con la relativa orbita
        vector<Planet>::iterator pIt;
        for (pIt = _satellites.begin(); pIt != _satellites.end(); pIt++)
        {
            // disegna orbita
            glColor3f(0.f, 0.f, 1.f);
            glDisable(GL_TEXTURE_2D);
            glBegin(GL_LINE_LOOP);
            for (float i = 0; i < 6.28; i += 0.2)
                glVertex3f(pIt->_orbit_radius * cos(i), 0, pIt->(_orbit_radius) * sin(i));
            glEnd();
            glEnable(GL_TEXTURE_2D);

            // disegna il satellite
            pIt->Draw();
        }
    glPopMatrix();
}
```

```
void initGL(int width, int height)
{
    // colore e tipo delle luci
    GLfloat light_ambient_0[] = {0.65, 0.65, 0.65, 1.0};// colore ambiente della luce 0
    GLfloat light_diffuse_0[] = {1.0, 1.0, 1.0, 1.0};   // colore diffusione della luce 0
    GLfloat light_specular_0[] = {1.0, 1.0, 1.0, 1.0};  // colore speculare della luce 0
    GLfloat light_position_0[] = {15.0, 5.0, 10.0, 0.0};    // posizione della luce 0

    glLightfv (GL_LIGHT0, GL_AMBIENT,   light_ambient_0);
    glLightfv (GL_LIGHT0, GL_DIFFUSE,   light_diffuse_0);
    glLightfv (GL_LIGHT0, GL_SPECULAR,  light_specular_0);
    glLightfv (GL_LIGHT0, GL_POSITION,  light_position_0);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);

    reshape(width, height);

    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glClearDepth(1.0f);

    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LEQUAL);

    glEnable(GL_TEXTURE_2D);

    //earth
    if (!earth.LoadTextureTGA("/Users/Fefy/Desktop/Solar_System/Solar_System/data/earthmap1k.tga"))
    {
        printf("failed to load earth texture!\n");
        exit(0);
    }
    earth.SetRadius(1.f);
    earth.SetSpinningAngle(0);
    earth.SetSpinningAngularVelocity(1);
    earth.SetOrbitRadius(15);
    earth.SetOrbitAngle(0);
    earth.SetOrbitAngularVelocity(2);
```



```
//-------------------------------------------------------------------
//draw Asteroid Belt
if (!asteroids1.LoadTextureTGA("/Users/Fefy/Desktop/Solar_System/Solar_System/data/moonmap4k.tga"))
{
    printf("failed to load moon texture!\n");
    exit(0);
}
asteroids1.SetRadius(.25f);
asteroids1.SetSpinningAngle(0);
asteroids1.SetSpinningAngularVelocity(1);
asteroids1.SetOrbitRadius(21);
asteroids1.SetOrbitAngle(0);
asteroids1.SetOrbitAngularVelocity(5.98);
```

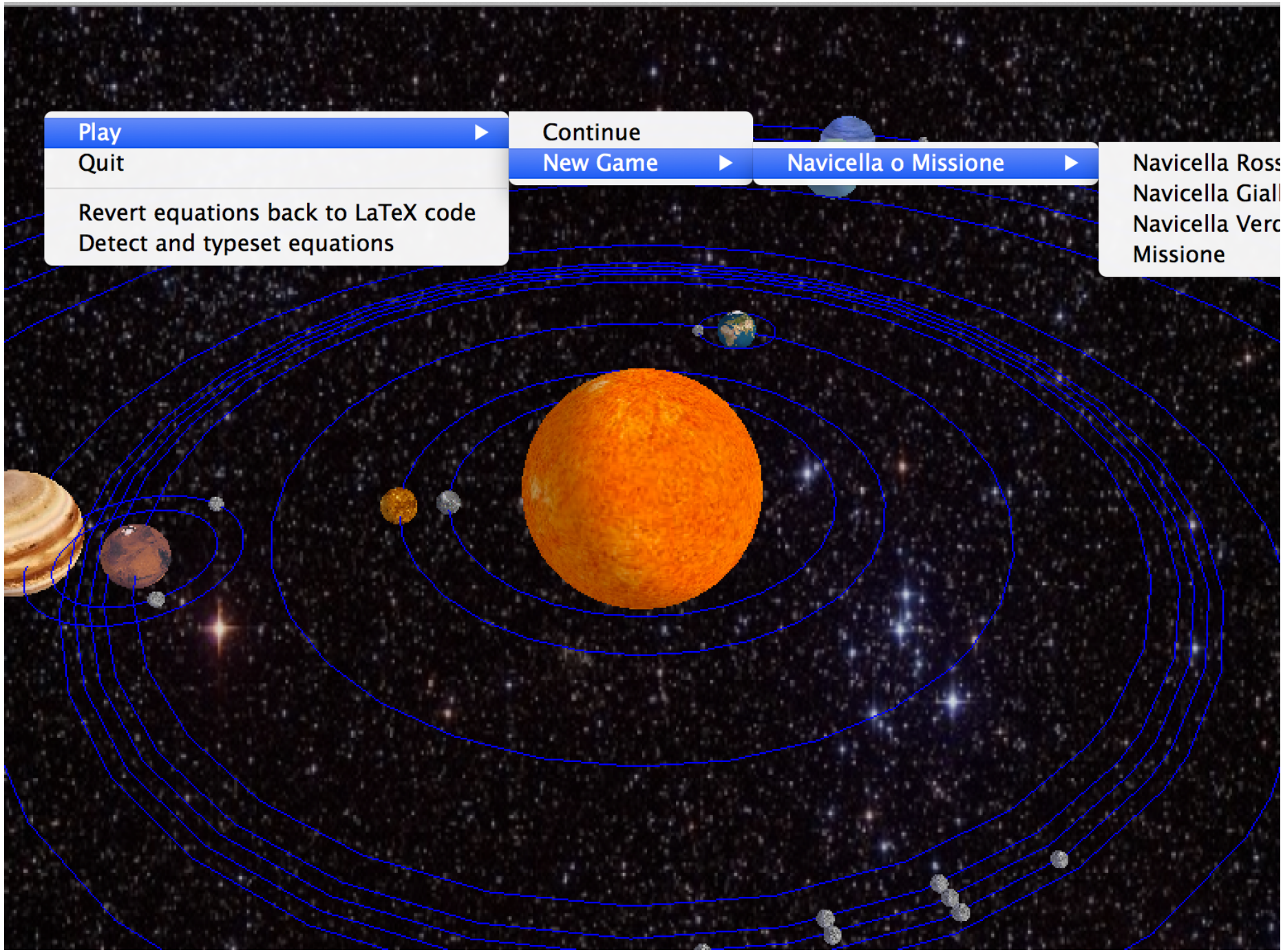
To calculate the trajectory of the spacecraft we use the Hohmann Transfer principle

```
void fuel(double r1, double r2, double m1)
{
    double const nepero = 2.71828183;
    double const Ve = 9.81 * 5000;
    double const Mu = 132712440018;
    double deltaV1 = sqrt(Mu/r1) * ( sqrt( ( 2 * r2 ) / (r1 + r2) ) - 1);
    double deltaV2 = sqrt(Mu/r2) * (1 - sqrt( ( 2 * r1 ) / (r1 + r2) ));
    double deltaVT =  (deltaV1 + deltaV2);
    //float m0 = nepero^(deltaVT/ Ve);
    //fuel = m1 - m0;

}
```

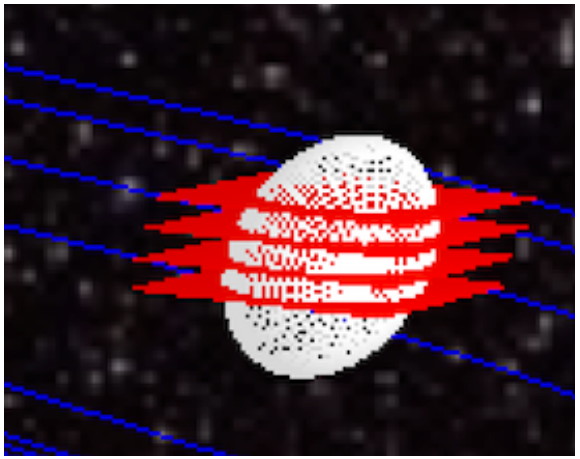
$$\Delta v_1 = \sqrt{\frac{\mu}{r_1}}\left(\sqrt{\frac{2r_2}{r_1+r_2}} - 1\right). \qquad \Delta v_2 = \sqrt{\frac{\mu}{r_2}}\left(1 - \sqrt{\frac{2r_1}{r_1+r_2}}\right).$$

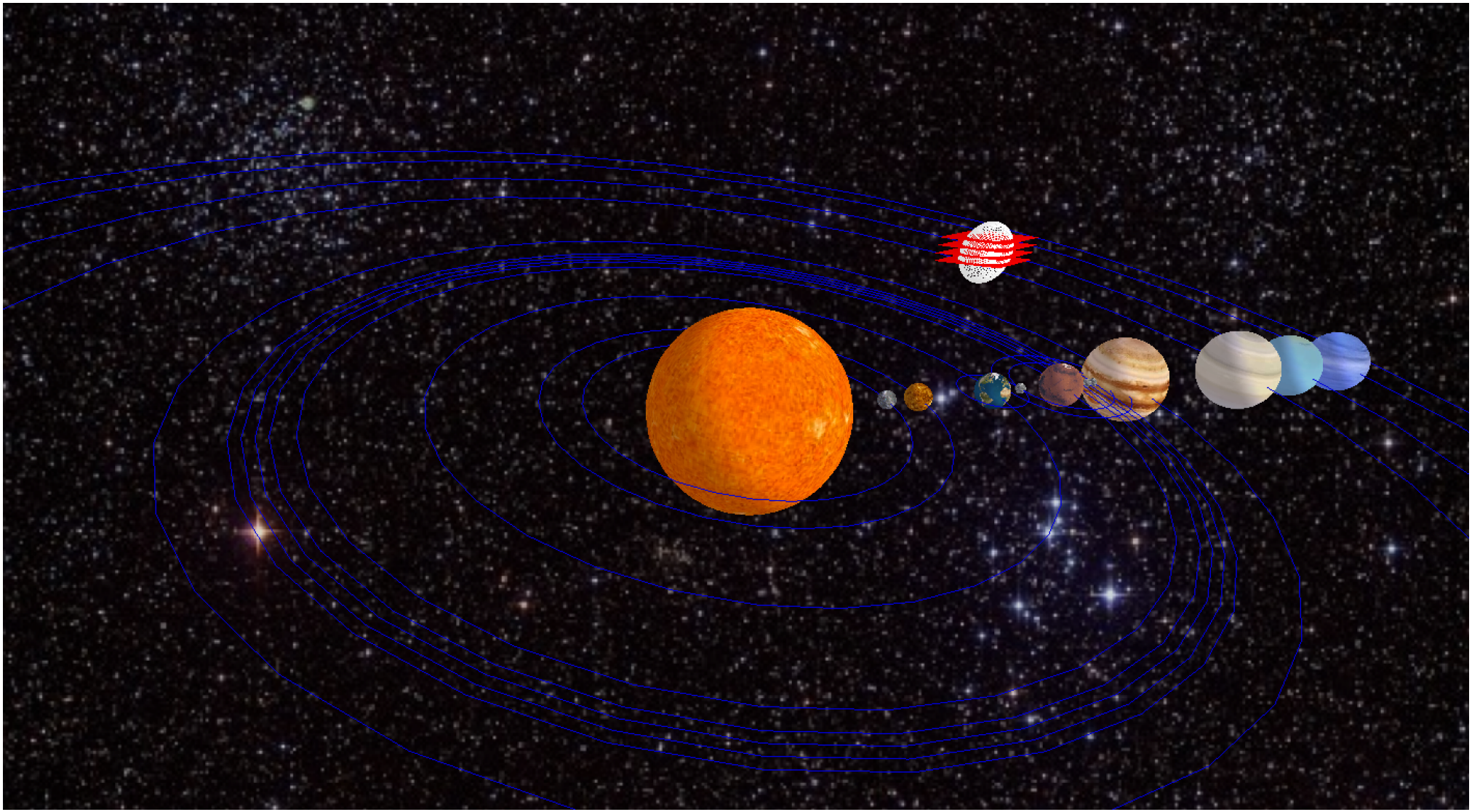
$$\Delta v_{total} = \Delta v_1 + \Delta v_2.$$

```
//---------------
int sub4 = glutCreateMenu(menuCB);
glutAddMenuEntry("Hard", MENU_H);
glutAddMenuEntry("Medium", MENU_G);
glutAddMenuEntry("Easy", MENU_F);


int sub3 = glutCreateMenu(menuCB);
glutAddMenuEntry("Navicella Rossa", MENU_E);
glutAddMenuEntry("Navicella Gialla", MENU_D);
glutAddMenuEntry("Navicella Verde", MENU_C);
glutAddSubMenu("Missione",  sub4);

int sub2 = glutCreateMenu(menuCB);
glutAddSubMenu("Navicella o Missione",sub3);

int sub1 = glutCreateMenu(menuCB);
glutAddMenuEntry("Continue", MENU_A);
glutAddSubMenu("New Game",  sub2);

glutCreateMenu(menuCB);
glutAddSubMenu("Play",  sub1);
glutAddMenuEntry("Quit",     MENU_QUIT);

glutAttachMenu(GLUT_RIGHT_BUTTON); /* tie popup menu to right mouse button */

//---------------
```


Play
Quit
Revert equations back to LaTeX code
Detect and typeset equations
Continue
New Game
Navicella o Missione
Navicella Ross
Navicella Giall
Navicella Verd
Missione

```
#ifndef Solar_System_spaceship_h
#define Solar_System_spaceship_h



class spaceship
{
    public:

    double storage; //spaceship's capability
    double life;
    double free_space; //inside our storage
   // std::vector< list < int > > arr(sizeof, list < int > ()); //


   GLUquadric *myQuad;

    void draw_spaceship(double x_shapeship, double y_shapeship, double z_shapeship)
    {
        glPushMatrix();

        glScalef(1,2,1);
        glTranslatef(x_shapeship, y_shapeship, z_shapeship);
        glRotatef(90, 0, 0, 1);

        glColor3f(1.0, 1.0, 1.0);
        glutWireSphere(2.0,5.0,6.0);


        glColor3f(1.0, 0.0, 0.0);
        glRotatef(90, 1, 0, 0);
        glutWireSphere(0.2, 2.0, 2.0);
        myQuad = gluNewQuadric();
        gluSphere(myQuad, 0.4, 2.0, 2.0);

        glColor3f(1.0, 0.0, 0.0);
        glRotatef(90, 1, 0, 0);
        glTranslatef(0.1, 0.0, 0.0);
        myQuad = gluNewQuadric();
        gluSphere(myQuad, 0.4, 2.0, 2.0);

        glColor3f(1.0, 0.0, 0.0);
        glRotatef(90, 1, 0, 0);
        glTranslatef(-0.05, 0.0, 0.0);
        myQuad = gluNewQuadric();
        gluSphere(myQuad, 0.4, 2.0, 2.0);

        glColor3f(1.0, 0.0, 0.0);
        glRotatef(90, 1, 0, 0);
        glTranslatef(-0.1, 0.0, 0.0);
        myQuad = gluNewQuadric();
        gluSphere(myQuad, 0.4, 2.0, 2.0);


        glPopMatrix();


    }//end make_base
```





Thank you