Кретање унутар модела. Трансформација камере.

Опис проблема

Направити следеће методе које ће се извршавати притиском појединих тастера на тастатури:

- Тастер „w“ – Кретање напред.
- Тастер „s“ – Кретање назад.
- Тастер „a“ – Бочно кретање лево.
- Тастер „d“ – Бочно кретање десно.
- Тастер „4“ – Окретање лево.
- Тастер „6“ – Окретање десно.
- Тастер „8“ – Поглед ка горе.
- Тастер „2“ – Поглед ка доле.

Решење

Кретање кроз модел се своди на трансформацију камере којом се модел посматра. Да би се ефекат трансформације камере видео потребно је након трансформације поново позвати функцију за постављање камере са измењеним параметрима.
Прво је потребно да оперативни систем ухвати догађај притиска тастера на тастатури, а потом и да одреагује на одговарајући начин позивом одговарајуће функције. Након тога се позива поновно постављање камере и поновно исцртавање модела. Испод је приказана метода која имплементираописану функционалност.

```
void KeyboardKeyPressed(unsigned char key, int x, int y)
{
    Matrix4x4 MT;
    Vector3D V;
    Vector3D L, T;

switch(key)
    {
    case 27:    //ESC key
        exit(0);
        break;
    case 'a':
        cout << "a pressed -> moving left" << endl;
        MoveLeft_CameraTransform();
        break;
    case 'd':
        cout << "d pressed -> moving right" << endl;
        MoveRight_CameraTransform();
        break;
    case 'w':
        cout << "w pressed -> moving forward" << endl;
        MoveForward_CameraTransform();
        break;
    case 's':
```

```
        cout << "s pressed -> moving backward" << endl;
        MoveBackward_CameraTransform();
        break;
    case 52:
        cout << "4 pressed -> turning left" << endl;
        TurnLeft_CameraTransform();
        break;
    case 54:
        cout << "6 pressed -> turning right" << endl;
        TurnRight_CameraTransform();
        break;
    case 56:
        cout << "8 pressed -> look up" << endl;
        LookUp_CameraTransform();
        break;
    case 50:
        cout << "2 pressed -> look down" << endl;
        LookDown_CameraTransform();
        break;
    }

    setCamera();
    glutPostRedisplay();
}
```

У зависности од притиска на неки тастер на тастатури, извршиће се одговарајућа метода. О тим методама ће бити речи испод.

Треба напоменути да ће се у овим методама наћи једна додатна фунционалност. То је бирање брзине којом ће бити вршено кретање кроз модел. Односно, уколико је притиснут леви тастер миша, кретање ће бити брже од стандардног.

У програму су дефинисани макрои који ће водити рачуна о брзини кретања као и о углу при окретању:

```
#define MOVING_CONST 0.1
#define ROTATION_CONST 3.14/180.0
#define SPEED_CONST_NORMAL 1
#define SPEED_CONST_FAST 10
```

Да би се извршило кретање напред прво је потребно одредити вектор правца у коме је усмерена камера $\vec{v}$ као разлику вектора тачке ка којој је камерена усмерена и тачке којом је орећен положај камере. Уколико дефинишемо кретање по чврстој и равној подлози y координата се неће мењати, па из тог разлога добијени вектор треба пројектовати у раван xOz. Да брзина не би зависила од удаљености позиције камере и тачке ка којој је усмерена, потребно је нормализовати вектор $\vec{v}$, а потом у зависности од изабране брзине кретања помножити одговарајућом константом. Сада је још потребно транслирати позицију камере и тачку ка којој је камера усмерена за вектор $\vec{v}$.

Кретање уназад се дефинише на исти начин осим што се транслација врши за вектор $-\vec{v}$. Програмски код ових метода дат је испод.

```
void MoveForward_CameraTransform()
{
    Matrix4x4 MT;
    Vector3D v = LookAt_vector - CameraPosition;
    v.m_y = 0.0;

    v.Normalize();

    double speed = SPEED_CONST_NORMAL;
    if (speedIndicator == 1)
        speed = SPEED_CONST_FAST;

    v = v * speed * MOVING_CONST;

    MT.loadTranslate(v.m_x, v.m_y, v.m_z);

    CameraPosition = MT.Transform(CameraPosition);
    LookAt_vector = MT.Transform(LookAt_vector);
}
```

```
void MoveBackward_CameraTransform()
{
    Matrix4x4 MT;
    Vector3D v = LookAt_vector - CameraPosition;
v.m_y = 0.0;

    v.Normalize();

v = -v * SPEED_CONST_NORMAL * MOVING_CONST;

    MT.loadTranslate(v.m_x, v.m_y, v.m_z);

    CameraPosition = MT.Transform(CameraPosition);
    LookAt_vector = MT.Transform(LookAt_vector);
}
```

Нека је $\vec{f}$ вектор правца у коме је усмерена камера. Векторским производом вектора $\vec{f}$ и вектора камере $\overrightarrow{up}$ добија се вектор $\vec{w} = \overrightarrow{up} \times \vec{f}$. Вектор $\vec{w}$ се користи приликом бочног кретања лево и десно. Потребно је још нормализовати добијени вектор и помножити га константом којом је одређена брзина кретања. Камера се транслира за вектор $\vec{w}$ уколико је реч

о бочном кретању у лево, а за вектор $-\vec{w}$ ако је реч о бочном кретању у десно.За исти вектор се транслира и тачка ка којој је усмерена камера.

```
void MoveLeft_CameraTransform()
{
    Matrix4x4 MT;
    Vector3D f = LookAt_vector - CameraPosition;
    Vector3D w = LookUp_vector.Cross(f);

w.Normalize();

w = w * SPEED_CONST_NORMAL * MOVING_CONST;

    MT.loadTranslate(w.m_x, w.m_y, w.m_z);

    CameraPosition = MT.Transform(CameraPosition);
    LookAt_vector = MT.Transform(LookAt_vector);
}
```

```
void MoveRight_CameraTransform()
{
    Matrix4x4 MT;
    Vector3D f = LookAt_vector - CameraPosition;
    Vector3D w = LookUp_vector.Cross(f);

w.Normalize();

w = -w * SPEED_CONST_NORMAL * MOVING_CONST;

    MT.loadTranslate(w.m_x, w.m_y, w.m_z);

    CameraPosition = MT.Transform(CameraPosition);
    LookAt_vector = MT.Transform(LookAt_vector);
}
```

Окретање лево и десно се реализује ротирањем камере око осе која пролази кроз тачку којом је одређен положај камере, а има правац y осе. Таква ротација се формира из три трансформације. Нека је вектор $\vec{n}$ паралелан са XZ равни и такав да камеру транслира до y осе. Прва трансформација врши транслирање камере за вектор $\vec{n}$. Потом се врши одговарајућа ротација око y осе. И на крају се врши инверзна транслација за вектор $-\vec{n}$. Знаком угла ротације се одређује да ли се врши окрет на лево или на десно.

```
void TurnLeft_CameraTransform()
{
    Matrix4x4 MR,MT1,MT2,MT;

    MT1.loadTranslate(-CameraPosition.m_x, 0.0, -CameraPosition.m_z);
    MT2.loadTranslate(CameraPosition.m_x,  0.0, CameraPosition.m_z);
```

```
    MR.loadRotateY(ROTATION_CONST);

    MT = MT2 * MR * MT1;

    LookAt_vector = MT.Transform(LookAt_vector);
}
```

```
void TurnRight_CameraTransform()
{
    Matrix4x4 MR,MT1,MT2,MT;

    MT1.loadTranslate(-CameraPosition.m_x, 0.0, -CameraPosition.m_z);
    MT2.loadTranslate(CameraPosition.m_x,  0.0, CameraPosition.m_z);
    MR.loadRotateY(-ROTATION_CONST);

    MT = MT2 * MR * MT1;

    LookAt_vector = MT.Transform(LookAt_vector);
}
```

Поглед на горе или на доле се реализује ротацијом тачке ка којој је усмерена камера $\overrightarrow{lookAp}$ и вектора камере $\overrightarrow{up}$ око осе која има правац већ описаног вектора $\vec{w}$, а пролази кроз тачку којом је одређен положај камере. Знак угла ротације одређује да ли се врши поглед ка горе или ка доле. У програмском коду испод, дозвољено је само ротирање на доле и горе до угла $\frac{\pi}{2}$.

```
void LookUp_CameraTransform()
{
    Matrix4x4 MR;
    Vector3D f = LookAt_vector - CameraPosition;
    Vector3D w = LookUp_vector.Cross(f);

    if(rotationUpDown + ROTATION_CONST < 0.5*M_PI)
    {
        MR.loadRotate(CameraPosition, w, -ROTATION_CONST);

        LookAt_vector = MR.Transform(LookAt_vector);

        Vector4D LookUp_vector4d = LookUp_vector;
        LookUp_vector4d.m_w = 0.0;
        LookUp_vector4d = MR.Transform(LookUp_vector4d);
        LookUp_vector = LookUp_vector4d;

        rotationUpDown += ROTATION_CONST;
    }
}
```

```
void LookDown_CameraTransform()
{
    Matrix4x4 MR;
    Vector3D f = LookAt_vector - CameraPosition;
    Vector3D w = LookUp_vector.Cross(f);

    if(rotationUpDown - ROTATION_CONST > -0.5*M_PI)
    {
        MR.loadRotate(CameraPosition, w, ROTATION_CONST);

        LookAt_vector = MR.Transform(LookAt_vector);

        Vector4D LookUp_vector4d = LookUp_vector;
        LookUp_vector4d.m_w = 0.0;
        LookUp_vector4d = MR.Transform(LookUp_vector4d);
        LookUp_vector = LookUp_vector4d;

        rotationUpDown -= ROTATION_CONST;
    }
}
```

Испод се налази и функција којом се реагује на притисак тастатуре миша, а којом се може изабрати брже кретање унапред.

```
void mousePress(int button, int state, int x, int y)
{
    switch (button)
    {
    case GLUT_LEFT_BUTTON:
        if(state == GLUT_DOWN)
            speedIndicator = 1;
        else
            speedIndicator = 0;
        break;
    default:
        break;
    }
}
```