

OpenGL - Rubik's Cube

Grafica pe calculator - proiect 3d

Eduard-Valentin Dumitrescul - grupa 332

1. Conceptul proiectului

Reprezentarea 3d a unui cub rubik interactiv

2. Elemente incluse



- iluminare
- umbra
- ceata

- cuaternioni

```
void AnimationManager::rotateBack(bool reverse)
{
    float sign = 1;
    if (reverse)
    {
        sign = -1;
    }

    for (int i = 0; i < 3; i++)
    {
        int k = 0;
        for (int j = 0; j < 3; j++)
        {
            glm::quat deltaRotation = glm::angleAxis(glm::radians( degrees: 90.0f) * sign, glm::vec3(x: 0.0f, y: 0.0f, z: 1.0f));
            glm::quat interpolatedRotation = glm::slerp(glm::quat(x: 1, y: 0, z: 0), glm::quat(x: 0, y: 1, z: 0), deltaRotation, 1.0f * animationProgress / animationDuration);
            cubes[i][j][k]->setAnimationRotation(interpolatedRotation);

            glm::vec3 currentPosition = cubes[i][j][k]->getPosition();
            glm::vec3 positionRelativeToAxis = glm::vec3(currentPosition.x, currentPosition.y, z: 0.0f);
            glm::vec3 rotatedPosition = interpolatedRotation * positionRelativeToAxis;
            cubes[i][j][k]->setAnimationPosition(rotatedPosition - positionRelativeToAxis );
        }
    }
}
```

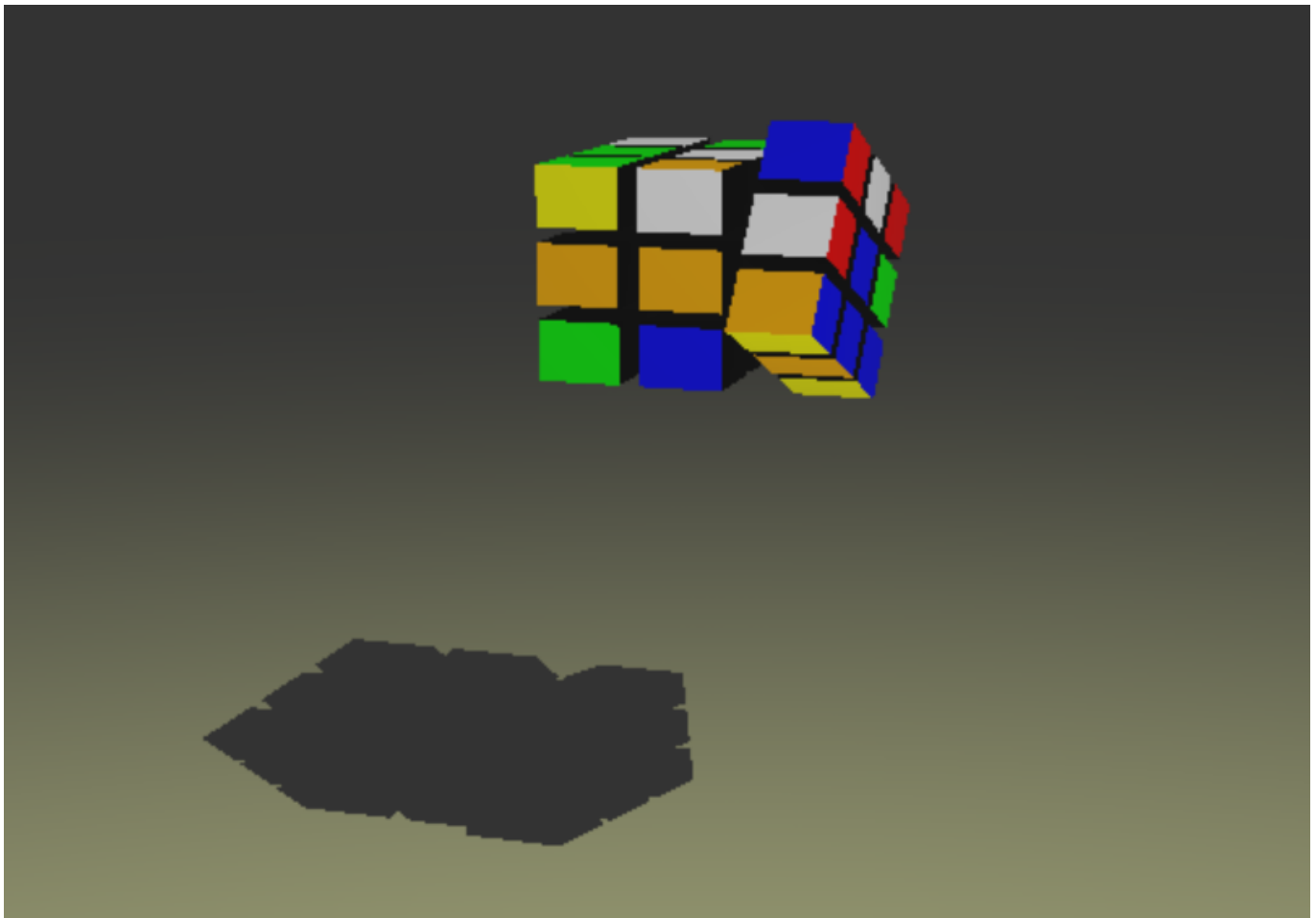
- animatii

```
void AnimationManager::startAnimation(Move move, std::function<void()> callback)
{
    if (animationRunning == false)
    {
        this->animationProgress = 0;
        this->move = move;
        animationRunning = true;
        this->animationFinishedCallback = callback;
    }
}
```

3. Originalitate

Implementarea folosind strict principiile de baza ale OpenGL si C++.

4. Capturi de ecran



```

49 void Rubik::performMove(Move move, std::function<void()>onMoveFinished, bool animation)
50 {
51     if (animation == false)
52     {
53         moveHandler->performMove(move);
54         for (int i = 0; i < 3; ++i)
55         {
56             for (int j = 0; j < 3; ++j)
57             {
58                 for (int k = 0; k < 3; ++k)
59                 {
60                     cubes[i][j][k]->removeRotation();
61                     cubes[i][j][k]->setPosition({ x: i-1, y: j-1, z: k-1});
62                 }
63             }
64         }
65         return;
66     }
67     auto callback = [this, move, onMoveFinished]() ->void {
68         moveHandler->performMove(move);
69
70         for (int i = 0; i < 3; ++i)
71         {
72             for (int j = 0; j < 3; ++j)
73             {
74                 for (int k = 0; k < 3; ++k)
75                 {
76                     cubes[i][j][k]->removeRotation();
77                     cubes[i][j][k]->setPosition({ x: i-1, y: j-1, z: k-1});
78                 }
79             }
80         }
81         onMoveFinished();
82     };
83     animationManager->startAnimation(move, callback);

```

```

102 void MoveHandler::performMove(Move move)
103 {
104     std::shared_ptr<Cube> face[3][3];
105
106     switch (move) {
107     case Move::UP:
108         getFace( axis: 1, index: 2, [&] face);
109         rotateFaceCounterClockwise( [&] face);
110         rotateCubes( axis: 1, clockwise: true, [&] face);
111         setFace( axis: 1, index: 2, face);
112         break;
113
114     case Move::UP_REVERSE:
115         getFace( axis: 1, index: 2, [&] face);
116         rotateFaceClockwise( [&] face);
117         rotateCubes( axis: 1, clockwise: false, [&] face);
118         setFace( axis: 1, index: 2, face);
119         break;
120
121     case Move::DOWN:
122         getFace( axis: 1, index: 0, [&] face);
123         rotateFaceClockwise( [&] face);
124         rotateCubes( axis: 1, clockwise: false, [&] face);
125         setFace( axis: 1, index: 0, face);
126         break;
127
128     case Move::DOWN_REVERSE:
129         getFace( axis: 1, index: 0, [&] face);
130         rotateFaceCounterClockwise( [&] face);
131         rotateCubes( axis: 1, clockwise: true, [&] face);
132         setFace( axis: 1, index: 0, face);
133         break;

```

Cuaternioni

```

glm::quat deltaRotation = glm::angleAxis(glm::radians(90.0f) * sign,
glm::vec3(0.0f, 1.0f, 0.0f));
    glm::quat interpolatedRotation = glm::slerp({1, 0, 0, 0},
deltaRotation, 1.0f * animationProgress / animationDuration);
    cubes[i][j][k]->setAnimationRotation(interpolatedRotation);

```

```

glm::quat rotation = {1,0 , 0, 0};
glm::quat animationRotation = {1, 0, 0, 0};

```

Link Github

[GitHub repository](#)