OpenGL - Rubik's Cube

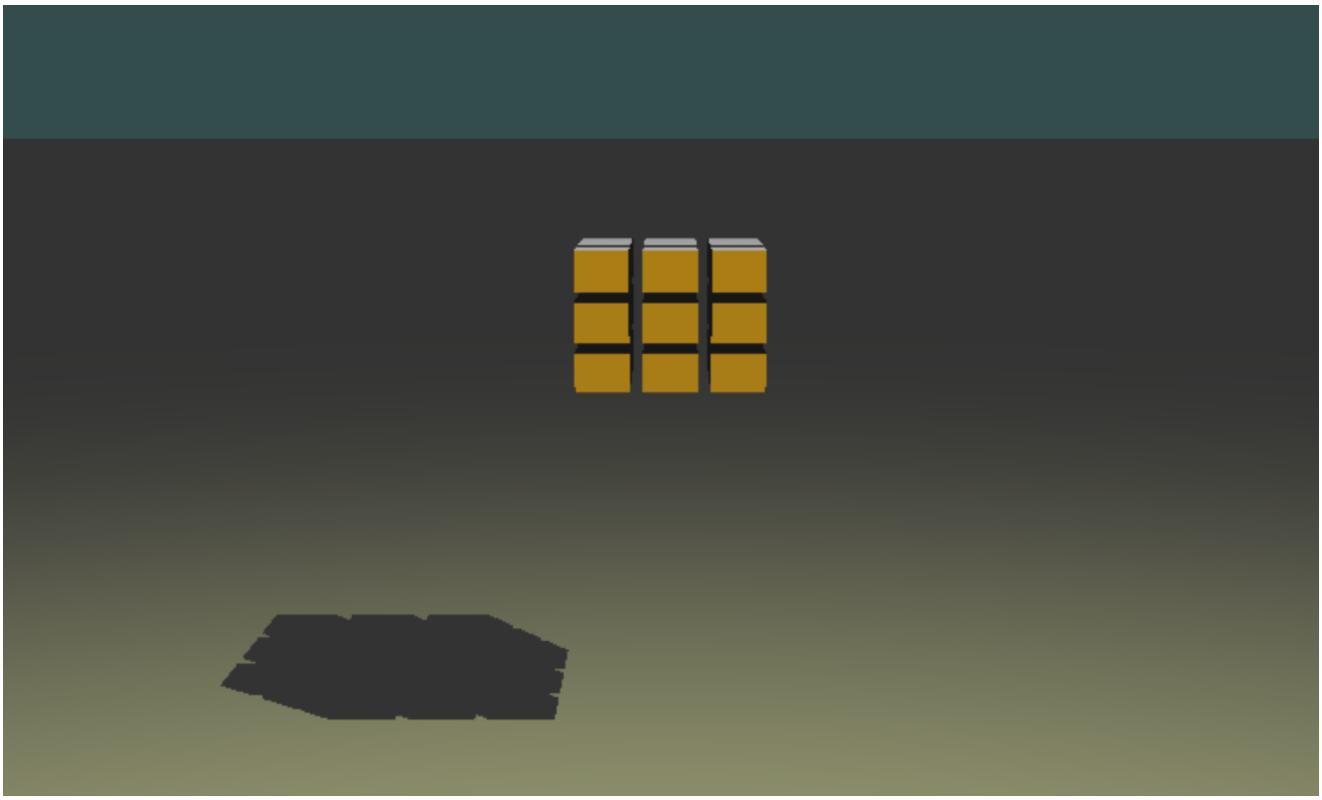
Grafica pe calculator - proiect 3d

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, v: glm::vec3( x: 0.0f, y: 0.0f, z: 1.0f));
            glm::quat interpolatedRotation = glm::slerp( x: ... { w: 1, x: 0, y: 0, z: 0}, y: deltaRotation, a: 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 );
        }
    }
}
```

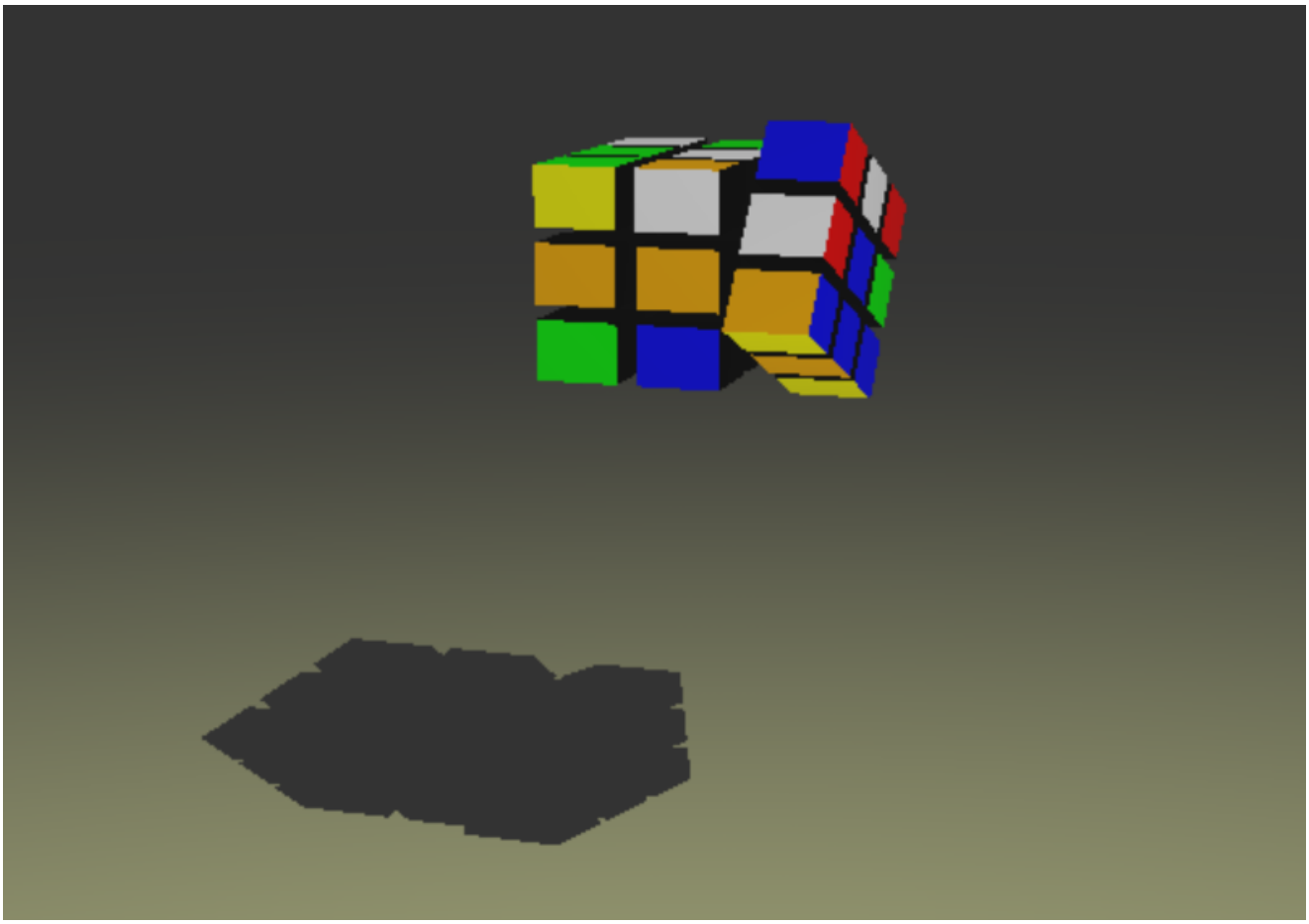
- animatii

```
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 folosing strict principii de baza ale OpenGL si C++.

4. Capturi de ecran



```cpp
void Rubik::performMove(Move move, std::function<void()>onMoveFinished, bool animation)
{
    if (animation == false)
    {
        moveHandler->performMove(move);
        for (int i = 0; i < 3; ++i)
        {
            for (int j = 0; j < 3; ++j)
            {
                for (int k = 0; k < 3; ++k)
                {
                    cubes[i][j][k]->removeRotation();
                    cubes[i][j][k]->setPosition( { x: i-1,  y: j-1,  z: k-1});
                }
            }
        }
        return;
    }
    auto callback = [this, move, onMoveFinished]() ->void {
        moveHandler->performMove(move);

        for (int i = 0; i < 3; ++i)
        {
            for (int j = 0; j < 3; ++j)
            {
                for (int k = 0; k < 3; ++k)
                {
                    cubes[i][j][k]->removeRotation();
                    cubes[i][j][k]->setPosition( { x: i-1,  y: j-1,  z: k-1});
                }
            }
        }
        onMoveFinished();
    };
    animationManager->startAnimation(move,  callback);

```

```cpp
void MoveHandler::performMove(Move move)
{
    std::shared_ptr<Cube> face[3][3];

    switch (move) {
    case Move::UP:
        getFace( axis: 1,  index: 2,  [&] face);
        rotateFaceCounterClockwise( [&] face);
        rotateCubes( axis: 1,  clockwise: true,  [&] face);
        setFace( axis: 1,  index: 2, face);
        break;

    case Move::UP_REVERSE:
        getFace( axis: 1,  index: 2,  [&] face);
        rotateFaceClockwise( [&] face);
        rotateCubes( axis: 1,  clockwise: false,  [&] face);
        setFace( axis: 1,  index: 2, face);
        break;

    case Move::DOWN:
        getFace( axis: 1,  index: 0,  [&] face);
        rotateFaceClockwise( [&] face);
        rotateCubes( axis: 1,  clockwise: false,  [&] face);
        setFace( axis: 1,  index: 0, face);
        break;

    case Move::DOWN_REVERSE:
        getFace( axis: 1,  index: 0,  [&] face);
        rotateFaceCounterClockwise( [&] face);
        rotateCubes( axis: 1,  clockwise: true,  [&] face);
        setFace( axis: 1,  index: 0, face);
        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};
```