Matching the maya camera

Akash Garg

Version 1.0 - June 23, 2018

Maya offers a different set of parameters for their camera, namely focal length, aperature and image aspect ratio. I need to take these values and convert them into a traditional opengl camera projection matrix.

Here is the code to obtain the projection matrix from maya:

```python
def getProjectionMatrix(camera):
    cameraShape = cmds.listRelatives(camera,  children=True)[0]

    list1 = om.MSelectionList()
    list1.add(cameraShape)

    depNode = om.MObject()
    list1.getDependNode(0, depNode)
    camFn = om.MFnCamera(depNode)
    pMtx = om.MFloatMatrix()
    pMtx = camFn.projectionMatrix()
    cameraMatrix = []

    for i in range(4*4):
        cameraMatrix.append(0.0)

    for i in range(4):
        for j in range(4):
            cameraMatrix[i*4 + j] = pMtx(i,j)
    print("This is the camera projection matrix obtained via Open Maya")
    print(cameraMatrix)

getProjectionMatrix("persp")
```

Maya output for the following projection camera: focal length: 70mm Aperature: 36mm x 36mm image aspect : 1.0 (360px x 360px)

Output from maya:

```
[3.8888890743255615, 0.0, 0.0, 0.0,
 0.0, 3.8888890743255615, 0.0, 0.0,
 0.0, 0.0, 1.000019907951355, -1.0,
 0.0, 0.0, 0.20000198483467102, 0.0]
```

Our projection matrix:

```
 3.89  0.00  0.00  0.00
 0.00  3.89  0.00  0.00
 0.00  0.00 -1.00 -0.20
 0.00  0.00 -1.00  0.00
```

Transposed due to row vs. col order. Also, opengl looks down negative z-axis LH vs. RH, which explains the sign difference.

The camera projection matrix can be implemeneted with this function:

```
Eigen::Matrix4d setFrustum(double l, double r, double b, double t, double n,
                           double f) const {
    Eigen::Matrix4d matrix = Eigen::Matrix4d::Zero();
    matrix.data()[0] = 2 * n / (r - l);
    matrix.data()[5] = 2 * n / (t - b);
    matrix.data()[8] = (r + l) / (r - l);
    matrix.data()[9] = (t + b) / (t - b);
    matrix.data()[10] = -(f + n) / (f - n);
    matrix.data()[11] = -1;
    matrix.data()[14] = -(2 * f * n) / (f - n);
    matrix.data()[15] = 0;
    std::cout << matrix << std::endl;
    return matrix;
}
```

Then given maya parameters, focal length, near / far and aperture height/width we can set the camera frustum using:

```
float focalLength = 79.3990980670794;
float filmApertureWidth = 24.0;
float filmApertureHeight = 24.0;
float nearClippingPlane = zNear_;
float farClipingPlane = zFar_;

float top =
    ((filmApertureHeight / 2.0) / focalLength) * nearClippingPlane;
float bottom = -top;
float filmAspectRatio = filmApertureWidth / filmApertureHeight;
float left =
    bottom * filmAspectRatio * (aspectRatio_ / filmAspectRatio);
float right = -left;

// params: left, right, bottom, top, near, far
return setFrustum(left, right, bottom, top, zNear_, zFar_);
```

Revision	**Date**	**Description**
1.0	June 23, 2018	Initial draft