Oblig 2 Matte 3

Adam Aske

20. februar 2022

Innhold

1 Github

Link til min branch : https://github.com/Hedmark-University-College-SPIM/3Dprog22/tree/AdamA

2 Oppgave 3.4.6

Oppgave 3.4.6 Valgte punkter: (-6, 10), (-5.9, 6.6), (-3, 4.8), (-3.1, 1.60), (0.1, 0.5), (2.6, 1.1), (3.8, 4.3), (6.7, 5.2)
Wolfram Aplha er brukt til matrise multiplikasjonene.
y = Ax + e

$$\begin{bmatrix}10\\6.6\\4.8\\1.6\\0.5\\1.1\\4.3\\5.2\end{bmatrix} = \begin{bmatrix}36 & -6 & 1\\43.8 & -5.9 & 1\\9 & -3 & 1\\9.6 & -3.1 & 1\\0 & 0.1 & 1\\6.7 & 2.6 & 1\\14.4 & 3.8 & 1\\44.9 & 6.7 & 1\end{bmatrix} \begin{bmatrix}a\\b\end{bmatrix} + \begin{bmatrix}e_1\\e_2\\e_3\\e_4\\e_5\\e_6\\e_7\end{bmatrix}$$

$$B = A^T * A$$

$$= \begin{bmatrix}36 & 34.8 & 9 & 9.6 & 0 & 6.7 & 14.4 & 44.9\\-6 & -5.9 & -3 & -3.1 & 0.1 & 2.6 & 3.8 & 6.7\\1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix} \begin{bmatrix}36 & -6 & 1\\43.8 & -5.9 & 1\\9 & -3 & 1\\9.6 & -3.1 & 1\\0 & 0.1 & 1\\6.7 & 2.6 & 1\\14.4 & 3.8 & 1\\44.9 & 6.7 & 1\end{bmatrix} = \begin{bmatrix}4948.5 & -97.07 & 155.4\\-105.11 & 158.6 & -4.8\\155.4 & -3.6 & 8\end{bmatrix}$$

$$C = A^T * y = \begin{bmatrix}36 & -6 & 1\\43.8 & -5.9 & 1\\9 & -3 & 1\\9.6 & -3.1 & 1\\0 & 0.1 & 1\\6.7 & 2.6 & 1\\14.4 & 3.8 & 1\\44.9 & 6.7 & 1\end{bmatrix} \begin{bmatrix}10\\6.6\\4.8\\1.6\\0.5\\1.1\\4.3\\5.2\end{bmatrix} = \begin{bmatrix}951\\-64.2\\34.1\end{bmatrix}$$

$$B^{-1} = \begin{bmatrix}0.0005 & 0 & -0.01\\0 & 0.006 & 0.003\\-0.01 & 0.001 & 0.32\end{bmatrix}$$

$$x = B^{-1} * c = \begin{bmatrix}00.0005 & 0 & -0.01\\0 & 0.006 & 0.003\\-0.01 & 0.001 & 0.32\end{bmatrix} \begin{bmatrix}951\\-64.2\\34.1\end{bmatrix} = \begin{bmatrix}0.145\\-0.268 & 1.309\end{bmatrix}$$

$y = 0.145x^2 - 0.268x + 1.309$

3 Beregne punkter og lagre i array

Funksjonen tar inn x som verdi og bruker funksjonen fra utergningen og returnerer y verdien punktet skal ha.

Listing 1: trianglesurface.h

```
static float func2(float x) {
        return 0.174 * x + 1, 743;
    }
```

4 3.4.6 Visualisering

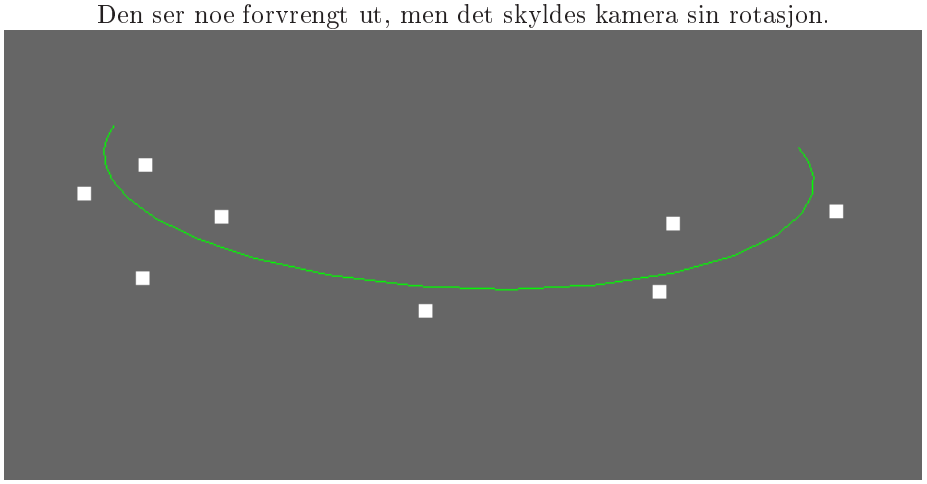
VisualPoint klassen tar inn en vector av Vertex'er, vertexene blir vist som hvite brikker på skjermen. MMap får en QuadraticPolynomial som tar inn 6.9, 1.3 og 3.2 fra minste kvadtraters metode, og blir vist som en grønn kurve på skjermen. De stemmer ikke med hverandre, noe er feil med utregningen.

Listing 2: renderwindow.cpp

```
mMap.insert(std::pair<std::string, VisualObject*>{"QuadraticPolynomial",
new QuadraticPolynomial(0.145f, -0.268f, 1.3f, 0.1f)});
    std::vector<Vertex> points;
    points.push_back(Vertex{ -6, 10, 0 });
    points.push_back(Vertex{ -5.9, 6.6, 0 });
    points.push_back(Vertex{ -3, 4.8, 0 });
    points.push_back(Vertex{ -3.1, 1.6, 0 });
    points.push_back(Vertex{ 0.1, 0.5, 0 });
    points.push_back(Vertex{ 2.6, 1.1, 0 });
    points.push_back(Vertex{ 3.8, 4.3, 0 });
    points.push_back(Vertex{ 6.7, 5.2, 0 });

    for (auto i = 0; i < points.size(); i++) {
        mMap.insert(std::pair<std::string, VisualObject*>
{ std::to_string(i) , new VisualPoint(points)});
    }
```

Den ser noe forvrengt ut, men det skyldes kamera sin rotasjon.



5 Oppgave 4.6.7

Punkter: (0.9, 0.6), (2.2, 2.6), (4.5, -1), (5.9, 1.6)

$$f(x) = ax^3 + bx^2 + cx + d$$
$$0.6 = a * 0.6^3 + b * 0.6^2 + c * 0.6 + d$$
$$2.6 = a * 2.6^3 + b * 2.6^2 + c * 2.6 + d$$
$$-1 = -a * 1^3 - b * 1^2 - c * 1 + d$$
$$1.6 = a * 1.6^3 + b * 1.6^2 + c * 1.6 + d$$

$$A = \begin{bmatrix} 0.729 & 0.81 & 0.9 & 1 \\ 10.648 & 4.84 & 2.2 & 1 \\ 91.125 & 20.25 & 4.5 & 1 \\ 205.379 & 34.81 & 5.9 & 1 \end{bmatrix} B = \begin{bmatrix} 0.6 \\ 2.6 \\ -1 \\ 1.6 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} -0.04 & 0.09 & -0.09 & -0.4 \\ 0.5 & -1.02 & 0.77 & -0.29 \\ -2.11 & 3.25 & -1.7 & 0.6 \\ 2.5 & -2.15 & 1 & -0.34 \end{bmatrix}$$

$$x = A^{-1} * B =$$

$$\begin{bmatrix} -0.04 & 0.09 & -0.09 & -0.4 \\ 0.5 & -1.02 & 0.77 & -0.29 \\ -2.11 & 3.25 & -1.7 & 0.6 \\ 2.5 & -2.15 & 1 & -0.34 \end{bmatrix} \begin{bmatrix} 0.6 \\ 2.6 \\ -1 \\ 1.6 \end{bmatrix} = \begin{bmatrix} -0.34 \\ -3.586 \\ 9.844 \\ -5.634 \end{bmatrix} f(x) = -0.34x^3 - 3.586x^2 + 9.844x - 5.634$$

6 4.6.7 Visualisering

VisualPoint klassen tar inn en vector av Vertex'er, vertexene blir vist som hvite brikker på skjermen. MMap får en QuadraticPolynomial som tar inn 6.9, 1.3 og 3.2 fra minste kvadtraters metode, og blir vist som en grønn kurve på skjermen. De stemmer ikke med hverandre, noe er feil med utregningen.

Listing 3: renderwindow.cpp

```
//Matte oblig 2 4.6.7
    mMap.insert(std::pair<std::string, VisualObject*>
        {"CubicPolynomial", new CubicPolynomial(-0.34, -3.586, 9.844, -5.634,0.1f)});
    std::vector<Vertex> points2;
    points2.push_back(Vertex{ 0.9f, 0.6f, 0 });
    points2.push_back(Vertex{ 2.2f, 2.6f, 0 });
    points2.push_back(Vertex{ 4.5f, -1.f, 0 });
    points2.push_back(Vertex{ 5.9f, 1.6, 0 });

    for (auto i = 0; i < points2.size(); i++) {
        mMap.insert(std::pair<std::string, VisualObject*>
        { std::to_string(i*10), new VisualPoint(points2)});
    }
```

Listing 4: cubicpolynomial.cpp

```
CubicPolynomial::CubicPolynomial(double a, double b, double c, double d, float dx)
{

    for (auto x = -10.f; x <= 10; x += 0.1)
```

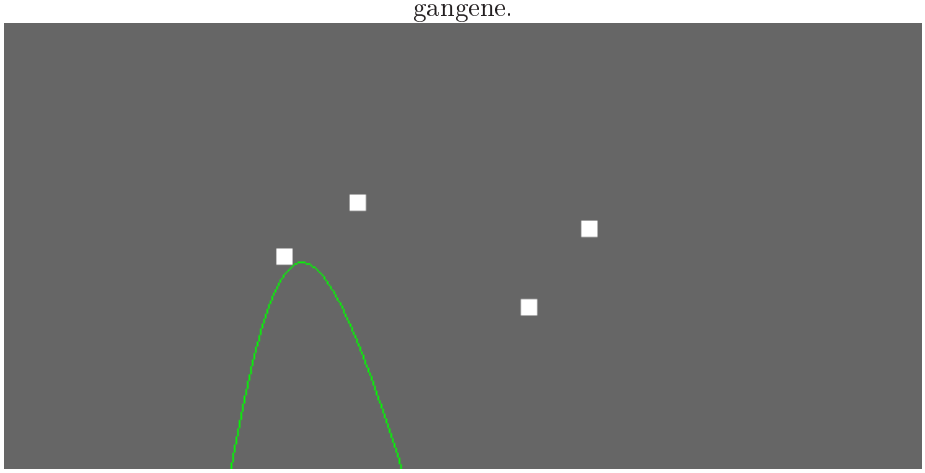
```
    {
        auto y = p(a, b, c, d, x);
        mVertices.push_back(Vertex(x, y, 0, 0, 1, 0));
    }
    mMatrix.setToIdentity();

}

double CubicPolynomial::p(double a, double b, double c, double d, double x)
{
    return a * x * x * x + b * x * x + c * x + d;

}
```

Resultatet ser ikke riktig ut i mine øyne, men har omregnet matrisene flere gangene.



7 Oppgave 4.11.6

Bezier kurve.
Initialiseringen av Bezier kurven.

Listing 5: renderwindow.cpp

```
//Bezier curve
    std::vector<QVector3D> controlPoints;
    controlPoints.push_back(QVector3D(0.f, 0.f, 0.f));
    controlPoints.push_back(QVector3D(2.f, 3.f, 0.f));
    controlPoints.push_back(QVector3D(4.f, -3.f, 0.f));
    controlPoints.push_back(QVector3D(6.f, 3.f, 0.f));
    mMap.insert(std::pair<std::string, VisualObject*>
        {"BezierCurve", new BezierCurve(controlPoints)});
```

Listing 6: beziercurve.cpp

```
BezierCurve::BezierCurve(std::vector<QVector3D> controlPoints) {
    mControlPoints = controlPoints;
    //Create vertexs from control points
    for (auto it : mControlPoints) {
        mControlPointsVertices.push_back
        (Vertex(it.x(), it.y(), it.z(), 1.f, 1.f, 1.f));
    }
    //Visualpoint for displaying control points
```

```
    mControlPointVisual = new VisualPoint(mControlPointsVertices);

    for (float t{}; t < 1.00f; t += 0.01f) {
        QVector3D point = EvaluateBezier(t);

        mVertices.push_back(Vertex(point.x(), point.y(), point.z()));
    }
}
QVector3D BezierCurve::EvaluateBezier(float t)
{
    std::vector<QVector3D> temp;

    //Gets the control points
    for (int i = 0; i < mControlPoints.size(); i++) {
        temp.push_back(mControlPoints[i]);
    }
    for(int k = temp.size()-1; k > 0; k--)
    {
        for(int i = 0; i < k; i++)
            //Bezier algoritmen
            temp[i] = temp[i] * (1 - t) + temp[i + 1] * t;
    }
    return temp[0];
}
```