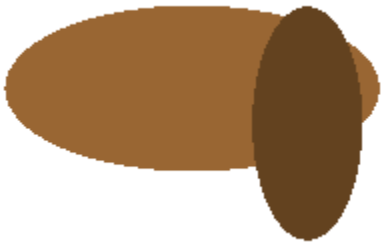
LAPORAN UTS GRAFIKA KOMPUTER

Anggota Kelompok :
- Albert Valentino - c14200135
- Samuel Chandra - c14200139
- Andre Cristian Leo - c14200179

Kecoa bersayap

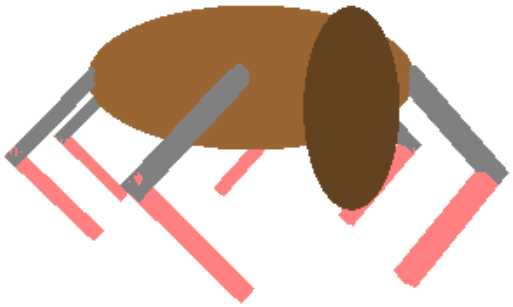
Kepala & badan menggunakan ellipsoid

Untuk **badan**, radius x dibuat lebih besar daripada radius y dan z sehingga membentuk oval yang horizontal. Lalu untuk **kepala**, radius y dibuat lebih besar daripada x dan z sehingga membentuk oval yang vertikal.



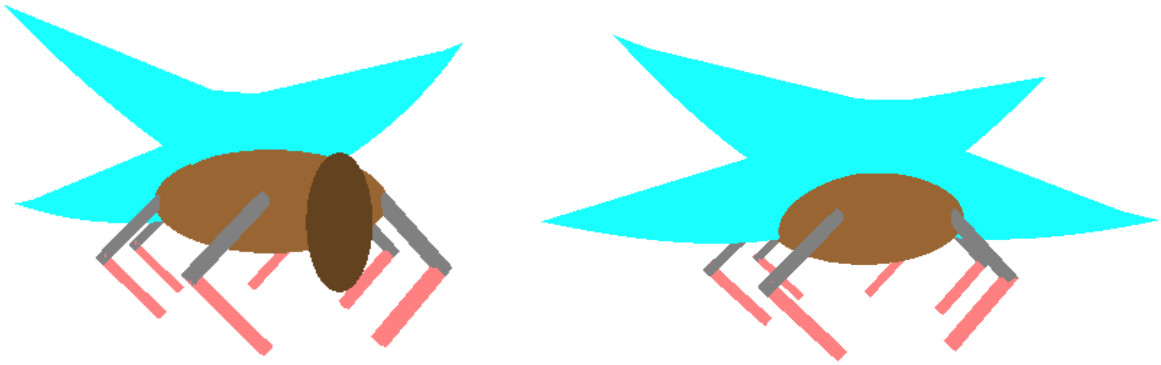
Kaki menggunakan cuboid

2 Cuboid saya rotate sehingga membentuk bentuk seperti pergelangan kaki.
Lalu diduplikasi sebanyak 3 dan ditranslasi agar letaknya sesuai.
Lalu untuk kaki pada bagian kanan, saya kali -1 angle rotasinya sehingga arahnya menjadi kebalikannya. Lalu duplikasi lagi sebanyak 3 dan translasi agar letaknya sesuai.



Sayap menggunakan hyperboloid paraboloid

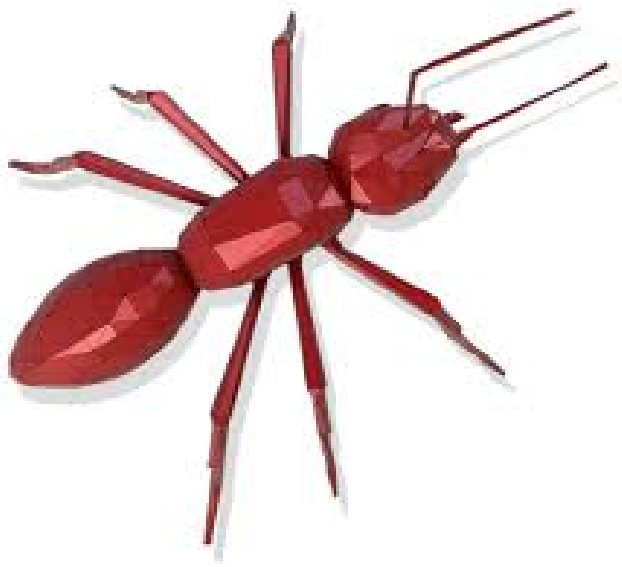
Menggunakan 2 hyperboloid paraboloid pada koordinat yang sama, karena akan dirotasi ke arah yang berbeda nantinya saat dianimasi.



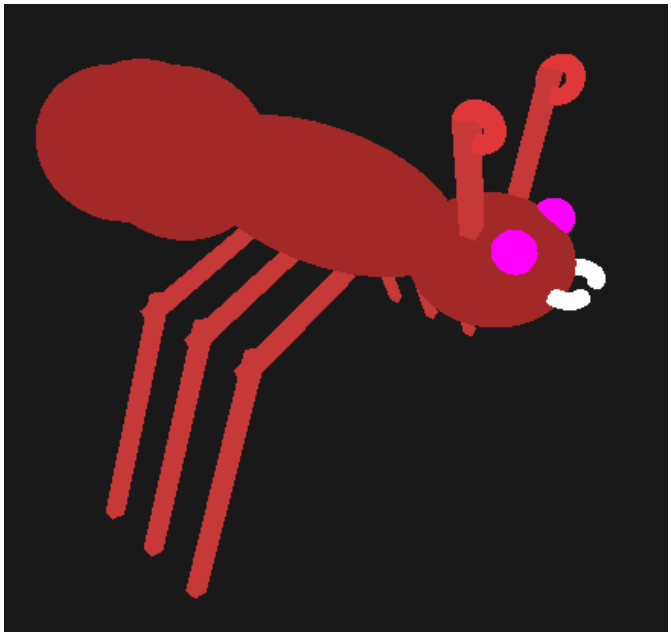
Animasi (terbang)
Rotasi sayap (hyperboloid paraboloid) ke kanan dan ke kiri secara terus menerus.
Seluruh bagian tubuh ditranslasi ke atas.

Semut

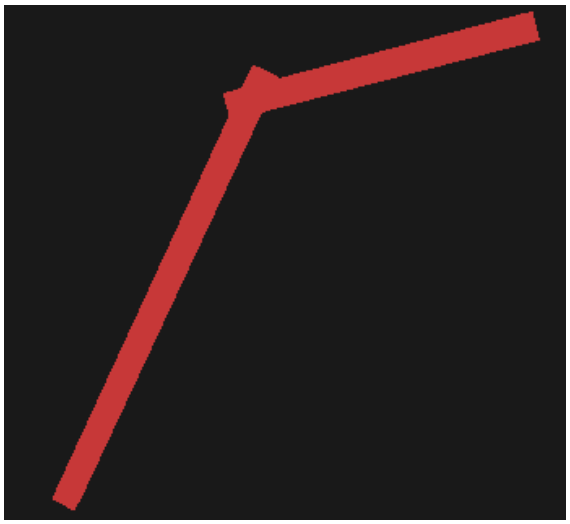
Ekspektasi model



Output code



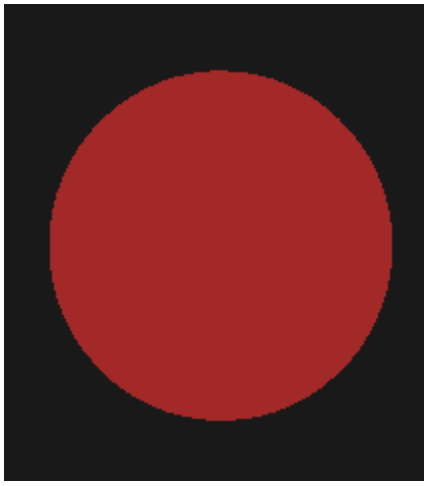
Kaki



Pertama tama saya menggunakan createCuboid_V2 yang udah diubah untuk menyerupai persegi panjang, lalu saya rotate dan saya translasikan untuk mendapatkan posisi yang saya mau.

Setelah mendapatkan 1 bagian dari kaki, saya miror kakinya sehingga mendapatkan sepasang kaki, yang akan dilanjutkan dengan duplikasi dan penggeseran sumbu x sehingga mendapakan 3 pasang kaki

Badan

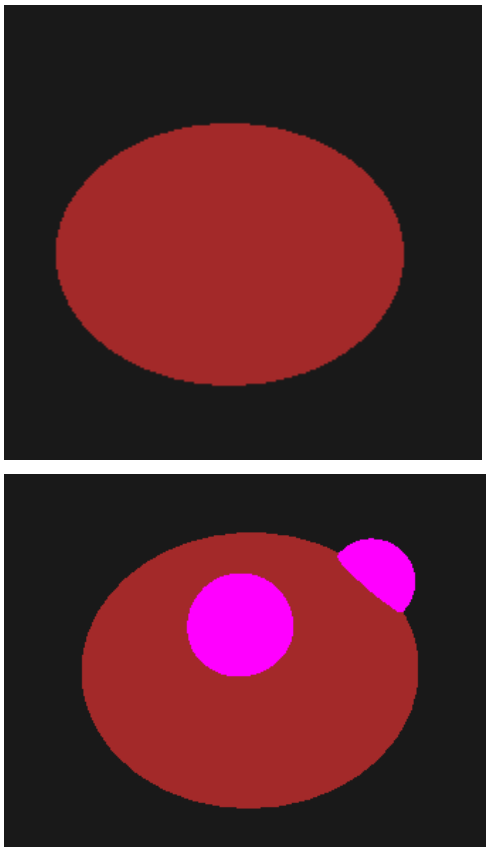


Pada bagian badan ini saya buat mulai dari belakang(Gaster) dimana awalnya saya membuatkan sebuah bola dengan menggunakan createEllipsoid. Lalu saya membuat 2 lagi bola lainnya. Supaya mendapatkan efek tergantung saya mengubah sedikit posisi y dan z pada bola ke 2 dan 3.

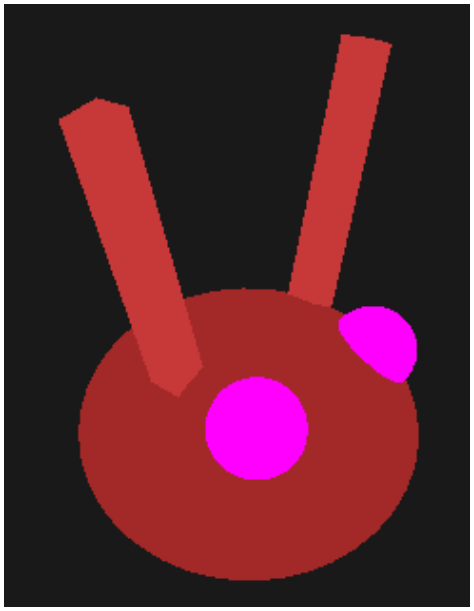
Selanjutnya di-ikuti dengan pembuatan badan bagian tengah(thorax) pada bagian ini saya mengunakan createEllipsoid juga namun saya menarik sumbu x dan mengecilkan sumbu y yang berperan dalam mengkontrol formnya sehingga dapat menghasilkan suatu bentuk oval.

Pada bagian kepala juga saya buat sama seperti badan(thorax) namun saya cuma menarik sumbu y nya saja.

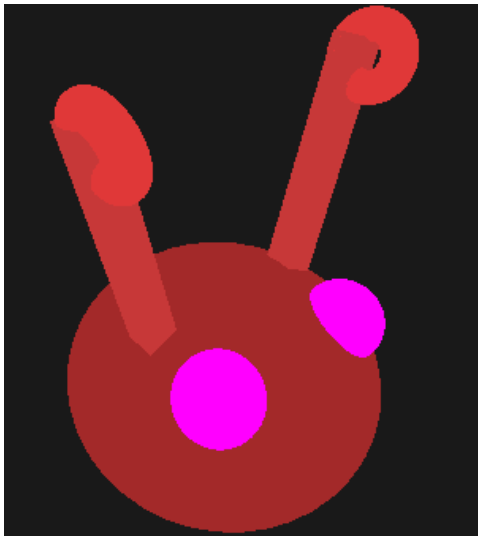
Kepala



Pada bagian mata saya menggukanan createEllipsoid tanpa adanya penarikan sumbu x,y,z sehingga dapat menghasilkan lingkaran sempurna



Pada bagian antena saya menggunakan createCuboid_V2 seperti di kaki untuk menjadikan antena semut. Antenanya juga menggunakan rotasi 45 dan -45 derajat



Lalu saya tambahkan di ujung antena dengan menggukanan createTorus sehingga mendapatan efek antena ini.

Pada bagian mulut ini saya menggunakan rumus createTorusV2 dimana hasilnya akan menjadi setengah torus saja.

Ulat

Badan,kepala,mata dan kaki
Pada bagian ini saya menggunakan ellipsoid dimana setiap radius nya di buat sama sehingga menghasilkan bentuk bola pada badan menggunakan 3 ellipsoid,pada kaki menggunakan 10 ellipsoid ,pada kepala menggunakan 1 dan pada mata menggunakan 4 ellipsoid

Gagang kacamata
Pada gagang panjang kacamata menggunakan cuboid yang di ubah rotasi nya sehingga bisa lurus kedepan dan pada gagang pendek nya rumus nya diubah dimana x nya dibagi dengan angka 1 ,y dibagi dengan angka 3 dan z dibagi dengan angka 2

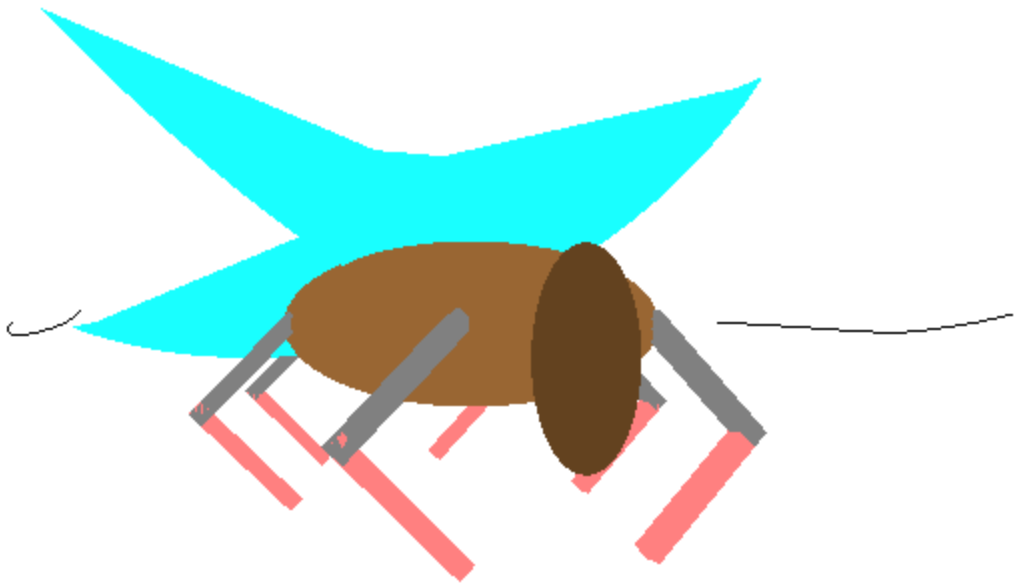
Frame kacamata
Pada frame kacamata menggunakan 2 buah torus yang dirotasikan sehingga menghadap ke depan

Mulut
Pada mulut nya menggunakan cone yang di rotasi kan hingga menghadap ke depan

Animasi
Animasi menggunakan rotate pada bagian kaki seakan - akan bergerak ke samping kanan

Angin

Menggunakan kurva bezier, kurva bezier disamping menggambar hembusan angin saat kecoa sedang terbang



CODE

Untuk membuat kurva bezier

```
public void prepareVertices()
{
    verticesCurve = new float[1080];
    index = 0;
}
public void setControlCoordinate(float x, float y, float z)
{
    verticesCurve[index * 3] = x;
    verticesCurve[index * 3 + 1] = y;
    verticesCurve[index * 3 + 2] = z;
    index++;
}
public List<int> getRow(int rowIndex)
{
    List<int> currow = new List<int>();
    //------
    currow.Add(1);
    if (rowIndex == 0)
    {
        return currow;
    }
    //-----
    List<int> prev = getRow(rowIndex - 1);
    for (int i = 1; i < prev.Count; i++)
    {
```

```
            int curr = prev[i - 1] + prev[i];
            currow.Add(curr);
        }
        currow.Add(1);
        return currow;

    }
    public List<Vector3> createCurveBazier()
    {
        List<Vector3> _verticesBazier = new List<Vector3>();
        List<int> pascal = getRow(index - 1);
        _pascal = pascal.ToArray();
        for (float t = 0; t <= 1; t += 0.01f)
        {
            Vector3 p = getP(index, t);
            _verticesBazier.Add(p);
        }
        return _verticesBazier;
    }
    public Vector3 getP(int n, float t)
    {
        Vector3 p = new Vector3(0, 0, 0);
        float k;
        for (int i = 0; i < n; i++)
        {
            k = (float)Math.Pow((1 - t), n - 1 - i) * (float)Math.Pow(t, i) * _pascal[i];
            p.X += k * verticesCurve[i * 3];
            p.Y += k * verticesCurve[i * 3 + 1];
            p.Z += k * verticesCurve[i * 3 + 2];

        }
        return p;
    }
    public void setVertices(List<Vector3> temp)
    {
        vertices = temp;
    }
```

Untuk membuat cuboid dengan panjang sisi yang sama

```
public void createCuboid(float x_, float y_, float z_, float length)
    {
        var tempVertices = new List<Vector3>();
        Vector3 temp_vector;
```

```
//Titik 1
temp_vector.X = x_ - length / 2.0f;
temp_vector.Y = y_ + length / 2.0f;
temp_vector.Z = z_ - length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 2
temp_vector.X = x_ + length / 2.0f;
temp_vector.Y = y_ + length / 2.0f;
temp_vector.Z = z_ - length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 3
temp_vector.X = x_ - length / 2.0f;
temp_vector.Y = y_ - length / 2.0f;
temp_vector.Z = z_ - length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 4
temp_vector.X = x_ + length / 2.0f;
temp_vector.Y = y_ - length / 2.0f;
temp_vector.Z = z_ - length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 5
temp_vector.X = x_ - length / 2.0f;
temp_vector.Y = y_ + length / 2.0f;
temp_vector.Z = z_ + length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 6
temp_vector.X = x_ + length / 2.0f;
temp_vector.Y = y_ + length / 2.0f;
temp_vector.Z = z_ + length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 7
temp_vector.X = x_ - length / 2.0f;
temp_vector.Y = y_ - length / 2.0f;
temp_vector.Z = z_ + length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 8
temp_vector.X = x_ + length / 2.0f;
```

```
        temp_vector.Y = y_ - length / 2.0f;
        temp_vector.Z = z_ + length / 2.0f;
        tempVertices.Add(temp_vector);

        var tempIndices = new List<uint>
        {
                                //Back
                                1, 2, 0,
            2, 1, 3,

                                //Top
                                5, 0, 4,
            0, 5, 1,

                                //Right
                                5, 3, 1,
            3, 5, 7,

                                //Left
                                0, 6, 4,
            6, 0, 2,

                                //Front
                                4, 7, 5,
            7, 4, 6,

                                //Bottom
                                3, 6, 2,
            6, 3, 7
        };
        vertices = tempVertices;
        indices = tempIndices;
    }
```

Membuat cuboid menjadi bentuk persegi panjang

```
    public void createCuboid_v3(float x_, float y_, float z_, float length)
    {
        var tempVertices = new List<Vector3>();
        Vector3 temp_vector;

        //Titik 1
        temp_vector.X = x_ - length / 1.0f;
        temp_vector.Y = y_ + length / 3.0f;
        temp_vector.Z = z_ - length / 2.0f;
```

```
tempVertices.Add(temp_vector);

//Titik 2
temp_vector.X = x_ + length / 1.0f;
temp_vector.Y = y_ + length / 3.0f;
temp_vector.Z = z_ - length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 3
temp_vector.X = x_ - length / 1.0f;
temp_vector.Y = y_ - length / 3.0f;
temp_vector.Z = z_ - length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 4
temp_vector.X = x_ + length / 1.0f;
temp_vector.Y = y_ - length / 3.0f;
temp_vector.Z = z_ - length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 5
temp_vector.X = x_ - length / 1.0f;
temp_vector.Y = y_ + length / 3.0f;
temp_vector.Z = z_ + length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 6
temp_vector.X = x_ + length / 1.0f;
temp_vector.Y = y_ + length / 3.0f;
temp_vector.Z = z_ + length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 7
temp_vector.X = x_ - length / 1.0f;
temp_vector.Y = y_ - length / 3.0f;
temp_vector.Z = z_ + length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 8
temp_vector.X = x_ + length / 1.0f;
temp_vector.Y = y_ - length / 3.0f;
temp_vector.Z = z_ + length / 2.0f;
tempVertices.Add(temp_vector);
```

```
        var tempIndices = new List<uint>
        {
                                //Back
                                1, 2, 0,
            2, 1, 3,

                                //Top
                                5, 0, 4,
            0, 5, 1,

                                //Right
                                5, 3, 1,
            3, 5, 7,

                                //Left
                                0, 6, 4,
            6, 0, 2,

                                //Front
                                4, 7, 5,
            7, 4, 6,

                                //Bottom
                                3, 6, 2,
            6, 3, 7
        };
        vertices = tempVertices;
        indices = tempIndices;
    }
```

Untuk membuat cuboid dengan panjang sisi yang berbeda

```
    public void createCuboid_v2(float x_, float y_, float z_, float length, float extra)
    {
        var tempVertices = new List<Vector3>();
        Vector3 temp_vector;

        //Titik 1
        temp_vector.X = x_ - length / 2.0f;
        temp_vector.Y = y_ + length / 2.0f;
        temp_vector.Z = z_ - length / 2.0f;
        tempVertices.Add(temp_vector);

        //Titik 2
        temp_vector.X = x_ + length / 2.0f;
```

```
temp_vector.Y = y_ + length / 2.0f;
temp_vector.Z = z_ - length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 3
temp_vector.X = x_ - length / 2.0f;
temp_vector.Y = y_ - (length + extra) / 2.0f;
temp_vector.Z = z_ - length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 4
temp_vector.X = x_ + length / 2.0f;
temp_vector.Y = y_ - (length + extra) / 2.0f;
temp_vector.Z = z_ - length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 5
temp_vector.X = x_ - length / 2.0f;
temp_vector.Y = y_ + length / 2.0f;
temp_vector.Z = z_ + length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 6
temp_vector.X = x_ + length / 2.0f;
temp_vector.Y = y_ + length / 2.0f;
temp_vector.Z = z_ + length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 7
temp_vector.X = x_ - length / 2.0f;
temp_vector.Y = y_ - (length + extra) / 2.0f;
temp_vector.Z = z_ + length / 2.0f;
tempVertices.Add(temp_vector);

//Titik 8
temp_vector.X = x_ + length / 2.0f;
temp_vector.Y = y_ - (length + extra) / 2.0f;
temp_vector.Z = z_ + length / 2.0f;
tempVertices.Add(temp_vector);

var tempIndices = new List<uint>
{
                        //Back
                        1, 2, 0,
```

```
            2, 1, 3,

                                //Top
                                5, 0, 4,
            0, 5, 1,

                                //Right
                                5, 3, 1,
            3, 5, 7,

                                //Left
                                0, 6, 4,
            6, 0, 2,

                                //Front
                                4, 7, 5,
            7, 4, 6,

                                //Bottom
                                3, 6, 2,
            6, 3, 7
        };
        vertices = tempVertices;
        indices = tempIndices;
    }
 public void createEllipsoid(float x, float y, float z, float radX, float radY, float radZ, float
sectorCount, float stackCount)
    {
        objectCenter = new Vector3(x, y, z);

        float pi = (float)Math.PI;
        Vector3 temp_vector;
        float sectorStep = 2 * pi / sectorCount;
        float stackStep = pi / stackCount;
        float sectorAngle, stackAngle, tempX, tempY, tempZ;

        for (int i = 0; i <= stackCount; ++i)
        {
            stackAngle = pi / 2 - i * stackStep;
            tempX = radX * (float)Math.Cos(stackAngle);
            tempY = radY * (float)Math.Sin(stackAngle);
            tempZ = radZ * (float)Math.Cos(stackAngle);

            for (int j = 0; j <= sectorCount; ++j)
```

```
            {
                sectorAngle = j * sectorStep;

                temp_vector.X = x + tempX * (float)Math.Cos(sectorAngle);
                temp_vector.Y = y + tempY;
                temp_vector.Z = z + tempZ * (float)Math.Sin(sectorAngle);

                vertices.Add(temp_vector);
            }
        }

        uint k1, k2;
        for (int i = 0; i < stackCount; ++i)
        {
            k1 = (uint)(i * (sectorCount + 1));
            k2 = (uint)(k1 + sectorCount + 1);

            for (int j = 0; j < sectorCount; ++j, ++k1, ++k2)
            {
                if (i != 0)
                {
                    indices.Add(k1);
                    indices.Add(k2);
                    indices.Add(k1 + 1);

                }

                if (i != stackCount - 1)
                {
                    indices.Add(k1 + 1);
                    indices.Add(k2);
                    indices.Add(k2 + 1);
                }
            }
        }
    }
```

Untuk membuat bentuk cone

```
public void createCone(float x, float y, float z, float radX, float radY, float radZ, float sectorCount,
float stackCount)
    {
        objectCenter = new Vector3(x, y, z);

        float pi = (float)Math.PI;
```

```
Vector3 temp_vector;
float sectorStep = 2 * pi / sectorCount;
float stackStep = pi / stackCount;
float sectorAngle, stackAngle, tempX, tempY, tempZ;

for (int i = 0; i <= stackCount; ++i)
{
    stackAngle = pi / 2 - i * stackStep;
    tempX = radX * (float)(stackAngle);
    tempY = radY * (float)(stackAngle);
    tempZ = radZ * (float)(stackAngle);

    for (int j = 0; j <= sectorCount; ++j)
    {
        sectorAngle = j * sectorStep;

        temp_vector.X = x + tempX * (float)Math.Cos(sectorAngle);
        temp_vector.Y = y + tempY * (float)Math.Sin(sectorAngle);
        temp_vector.Z = z + tempZ;

        if (temp_vector.Z > 0)
        {
            vertices.Add(temp_vector);
        }
    }
}

uint k1, k2;
for (int i = 0; i < stackCount; ++i)
{
    k1 = (uint)(i * (sectorCount + 1));
    k2 = (uint)(k1 + sectorCount + 1);

    for (int j = 0; j < sectorCount; ++j, ++k1, ++k2)
    {
        if (i != 0)
        {
            indices.Add(k1 + 1);
            indices.Add(k2);
            indices.Add(k1);
        }

        if (i != stackCount - 1)
        {
```

```
                indices.Add(k1 + 1);
                indices.Add(k2);
                indices.Add(k2 + 1);
            }
        }
    }
}
```

Untuk membuat hyperboloid paraboloid

```
public void createHyper(float x, float y, float z, float radX, float radY, float radZ, float
sectorCount, float stackCount)
{
    objectCenter = new Vector3(x, y, z);

    float pi = (float)Math.PI;
    Vector3 temp_vector;
    float sectorStep = 2 * pi / sectorCount;
    float stackStep = pi / stackCount;
    float sectorAngle, stackAngle, tempX, tempY, tempZ;

    for (int i = 0; i <= stackCount; ++i)
    {
        stackAngle = pi / 2 - i * stackStep;
        tempX = radX * (float)stackAngle;
        tempY = radY * (float)stackAngle;
        tempZ = radZ * (float)(stackAngle * stackAngle);

        for (int j = 0; j <= sectorCount; ++j)
        {
            sectorAngle = j * sectorStep;

            temp_vector.X = x + tempX * (float)(Math.Sin(sectorAngle) /
Math.Cos(sectorAngle));
            temp_vector.Y = y + tempY * (float)(1 / Math.Cos(sectorAngle));
            temp_vector.Z = z + tempZ;

            vertices.Add(temp_vector);
        }
    }

    uint k1, k2;
    for (int i = 0; i < stackCount; ++i)
    {
        k1 = (uint)(i * (sectorCount + 1));
```

```
            k2 = (uint)(k1 + sectorCount + 1);

            for (int j = 0; j < sectorCount; ++j, ++k1, ++k2)
            {
                if (i != 0)
                {
                    indices.Add(k1);
                    indices.Add(k2);
                    indices.Add(k1 + 1);

                }

                if (i != stackCount - 1)
                {
                    indices.Add(k1 + 1);
                    indices.Add(k2);
                    indices.Add(k2 + 1);
                }
            }
        }
    }
```

Untuk membuat bentuk torus

```
    public void createTorus(float x, float y, float z, float radMajor, float radMinor, float
sectorCount, float stackCount)
    {
        objectCenter = new Vector3(x, y, z);

        float pi = (float)Math.PI;
        Vector3 temp_vector;
        stackCount *= 2;
        float sectorStep = 2 * pi / sectorCount;
        float stackStep = 2 * pi / stackCount;
        float sectorAngle, stackAngle, tempX, tempY, tempZ;

        for (int i = 0; i <= stackCount; ++i)
        {
            stackAngle = pi / 2 - i * stackStep;
            tempX = radMajor + radMinor * (float)Math.Cos(stackAngle);
            tempY = radMinor * (float)Math.Sin(stackAngle);
            tempZ = radMajor + radMinor * (float)Math.Cos(stackAngle);

            for (int j = 0; j <= sectorCount; ++j)
            {
```

```
            sectorAngle = j * sectorStep;

            temp_vector.X = x + tempX * (float)Math.Cos(sectorAngle);
            temp_vector.Y = y + tempY;
            temp_vector.Z = z + tempZ * (float)Math.Sin(sectorAngle);

            vertices.Add(temp_vector);
        }
    }

    uint k1, k2;
    for (int i = 0; i < stackCount; ++i)
    {
        k1 = (uint)(i * (sectorCount + 1));
        k2 = (uint)(k1 + sectorCount + 1);

        for (int j = 0; j < sectorCount; ++j, ++k1, ++k2)
        {
            indices.Add(k1);
            indices.Add(k2);
            indices.Add(k1 + 1);

            indices.Add(k1 + 1);
            indices.Add(k2);
            indices.Add(k2 + 1);
        }
    }
}
```

Untuk membuat setengah torus

```
    public void createTorusV2(float x, float y, float z, float radMajor, float radMinor, float
sectorCount, float stackCount)
    {
        objectCenter = new Vector3(x, y, z);

        float pi = (float)Math.PI;
        Vector3 temp_vector;
        stackCount *= 2;
        float sectorStep = 2 * pi / sectorCount;
        float stackStep = 2 * pi / stackCount;
        float sectorAngle, stackAngle, tempX, tempY, tempZ;

        for (int i = 0; i <= stackCount; ++i)
```

```
        {
            stackAngle = pi / 2 - i * stackStep;
            tempX = radMajor + radMinor * (float)Math.Cos(stackAngle);
            tempY = radMinor * (float)Math.Sin(stackAngle);
            tempZ = radMajor + radMinor * (float)Math.Cos(stackAngle);

            for (int j = 0; j <= sectorCount; ++j)
            {
                sectorAngle = j * sectorStep;

                temp_vector.X = x + tempX * (float)Math.Cos(sectorAngle);
                temp_vector.Y = y + tempY;
                temp_vector.Z = z + tempZ * (float)Math.Sin(sectorAngle);

                vertices.Add(temp_vector);
            }
        }

        uint k1, k2;
        for (int i = 0; i < stackCount; ++i)
        {
            k1 = (uint)(i * (sectorCount + 1));
            k2 = (uint)(k1 + sectorCount + 1);

            for (int j = 0; j < sectorCount / 2; ++j, ++k1, ++k2)
            {
                indices.Add(k1);
                indices.Add(k2);
                indices.Add(k1 + 1);

                indices.Add(k1 + 1);
                indices.Add(k2);
                indices.Add(k2 + 1);
            }
        }
    }
```

Untuk merotate object

```
public void rotate(Vector3 pivot, Vector3 vector, float angle)
    {
        var radAngle = MathHelper.DegreesToRadians(angle);

        var arbRotationMatrix = new Matrix4
            (
```

```
            new Vector4((float)(Math.Cos(radAngle) + Math.Pow(vector.X, 2.0f) * (1.0f -
Math.Cos(radAngle))), (float)(vector.X * vector.Y * (1.0f - Math.Cos(radAngle)) + vector.Z *
Math.Sin(radAngle)), (float)(vector.X * vector.Z * (1.0f - Math.Cos(radAngle)) - vector.Y *
Math.Sin(radAngle)), 0),
            new Vector4((float)(vector.X * vector.Y * (1.0f - Math.Cos(radAngle)) - vector.Z *
Math.Sin(radAngle)), (float)(Math.Cos(radAngle) + Math.Pow(vector.Y, 2.0f) * (1.0f -
Math.Cos(radAngle))), (float)(vector.Y * vector.Z * (1.0f - Math.Cos(radAngle)) + vector.X *
Math.Sin(radAngle)), 0),
            new Vector4((float)(vector.X * vector.Z * (1.0f - Math.Cos(radAngle)) + vector.Y *
Math.Sin(radAngle)), (float)(vector.Y * vector.Z * (1.0f - Math.Cos(radAngle)) - vector.X *
Math.Sin(radAngle)), (float)(Math.Cos(radAngle) + Math.Pow(vector.Z, 2.0f) * (1.0f -
Math.Cos(radAngle))), 0),
            Vector4.UnitW
            );

        model *= Matrix4.CreateTranslation(-pivot);
        model *= arbRotationMatrix;
        model *= Matrix4.CreateTranslation(pivot);

        for (int i = 0; i < 3; i++)
        {
            _euler[i] = Vector3.Normalize(getRotationResult(pivot, vector, radAngle, _euler[i],
true));
        }

        objectCenter = getRotationResult(pivot, vector, radAngle, objectCenter);

        foreach (var i in child)
        {
            i.rotate(pivot, vector, angle);
        }
    }

    public Vector3 getRotationResult(Vector3 pivot, Vector3 vector, float angle, Vector3 point,
bool isEuler = false)
    {
        Vector3 temp, newPosition;

        if (isEuler)
        {
            temp = point;
        }
        else
        {
```

```
            temp = point - pivot;
        }

        newPosition.X =
            temp.X * (float)(Math.Cos(angle) + Math.Pow(vector.X, 2.0f) * (1.0f -
Math.Cos(angle))) +
            temp.Y * (float)(vector.X * vector.Y * (1.0f - Math.Cos(angle)) - vector.Z *
Math.Sin(angle)) +
            temp.Z * (float)(vector.X * vector.Z * (1.0f - Math.Cos(angle)) + vector.Y *
Math.Sin(angle));

        newPosition.Y =
            temp.X * (float)(vector.X * vector.Y * (1.0f - Math.Cos(angle)) + vector.Z *
Math.Sin(angle)) +
            temp.Y * (float)(Math.Cos(angle) + Math.Pow(vector.Y, 2.0f) * (1.0f -
Math.Cos(angle))) +
            temp.Z * (float)(vector.Y * vector.Z * (1.0f - Math.Cos(angle)) - vector.X *
Math.Sin(angle));

        newPosition.Z =
            temp.X * (float)(vector.X * vector.Z * (1.0f - Math.Cos(angle)) - vector.Y *
Math.Sin(angle)) +
            temp.Y * (float)(vector.Y * vector.Z * (1.0f - Math.Cos(angle)) + vector.X *
Math.Sin(angle)) +
            temp.Z * (float)(Math.Cos(angle) + Math.Pow(vector.Z, 2.0f) * (1.0f -
Math.Cos(angle)));

        if (isEuler)
        {
            temp = newPosition;
        }
        else
        {
            temp = newPosition + pivot;
        }
        return temp;
    }
```

Untuk mentranslasikan object

```
    public void translate(float x, float y, float z)
    {
        model *= Matrix4.CreateTranslation(x, y, z);
        objectCenter.X += x;
        objectCenter.Y += y;
```

```
        objectCenter.Z += z;

        foreach (var i in child)
        {
            i.translate(x, y, z);
        }
    }
```

Untuk memperbesar atau memperkecil object

```
public void scale(float scaleX, float scaleY, float scaleZ)
    {
        model *= Matrix4.CreateTranslation(-objectCenter);
        model *= Matrix4.CreateScale(scaleX, scaleY, scaleZ);
        model *= Matrix4.CreateTranslation(objectCenter);

        foreach (var i in child)
        {
            i.scale(scaleX, scaleY, scaleZ);
        }
    }
    #endregion
}
```