LAPORAN UAS GRAFKOM

PEMBENTUKAN OBJECT 3D, ANIMASI, CAMERA, SHADING

Felicia Laksana – C14190054
Levina Charin – C14190145
Handrian Alandi – C14190231

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS KRISTEN PETRA 2021

Kelompok:
Handrian C14190231
Felicia Laksana C14190054
Levina Charin C14190145

Rancangan Animasi Object 3D

- Membuat spongebob environment
- Object:
    - Rumah patrick
    - Rumah squidward
    - Rumah spongebob
    - Chum Bucket
    - Krusty krab
    - Plankton
    - Mr crab
    - Trampoline
    - Tulisan Judul ‘Spongebob Squarepants’
    - Batu-batuan
    - Jellyfish
    - Gary
    - Patrick,
      Animasi: Patrick menyapa mobil polisi yang lewat, dengan merotasi bagian tangan patrick terhadap sumbu X
    - Squidward,
      Animasi: Seperti biasanya, squidward kesal terhadap keramaian patrick dan spongebob dan sekitarnya, ia kesal dengan menggeleng-gelengkan kepala (me-rotasi sumbu Y)
    - Spongebob,
      Animasi: Spongebob dengan gembira loncat-loncat di trampoline di dekat rumahnya, ini salah satu alasan squidward kesal (translate sumbu Y)
    - Mobil Polisi,
      Animasi: Mobil polisi yang melewati rumah patrick,squidward dan spongebob, animasi di mobil polisi ini ada 2 yaitu translate keseluruhan mobil, dan satunya adalah merotasi kincir belakang mobil sehingga terlihat berputar

  - Untuk mengatur peletakkan tiap object dan tampilan, kami menggunakan translasi, rotasi dan skala

Proses Pembuatan:

- Pembuatan batasan environment
- Pembuatan camera
- Kami menggunakan blender untuk membuat object, dan beberapa kami import dari object jadian dari internet, lalu kita modifikasi warna, lokasi dan tampilan, di code c kami
- Pertama kami membuat satu object sebagai sumber cahaya
- Lalu membuat object tanah
- Lalu menambah object jalan
- Lalu dilanjutkan menambah object-object rumah dan icon-icon spongebob lainnya dan juga menambahkan animasi untuk beberapa object tersebut

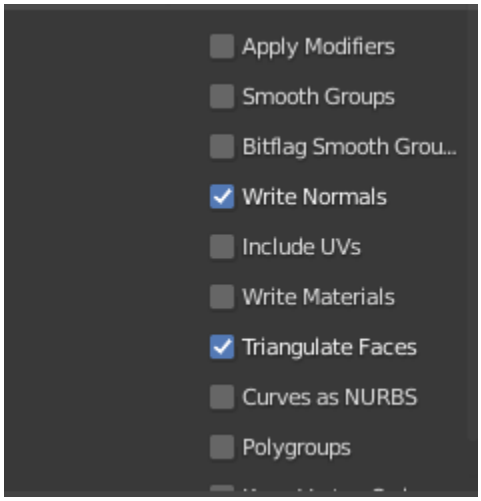
- Pembuatan proyek ini kami lakukan bersama - sama, dengan bantuan github. Tiap anggota kelompok menambahkan object-object sendiri sesuai adanya ide diri perlu ditambahkan apa.

Problem yang dihadapi:

- Ketika membuat object di blender secara lengkap, lalu import dalam opentk pada C. Dalam code opentk C kami, tidak bisa mengimport sekaligus dengan shader/warnanya yang di set dari blender, kami hanya bisa memberikan satu warna pada 1 objek yang kita import. **Solusi**: 1 object kami pisah-pisah per bagian, lalu kami import per bagian(yang warna nya beda), dan berikan warna, lalu menyatukannya kembali sehingga menjadi satu object utuh.

Bagaimana cara kita memproses data obj dari blender dan ditampilkan ke window OpenTK?

- Ketika kita mau export dari blender kita harus mencentang pilihan Triangulate Faces dan Write Normals:



- Dari situ kita bisa mendapatkan data obj seperti berikut:

```
# Blender v2.93.0 OBJ File: ''
# www.blender.org
o Cube
v 1.000000 1.000000 -1.000000
v 1.000000 -1.000000 -1.000000
v 1.000000 1.000000 1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 1.000000 -1.000000
v -1.000000 -1.000000 -1.000000
v -1.000000 1.000000 1.000000
v -1.000000 -1.000000 1.000000
vn 0.0000 1.0000 0.0000
vn 0.0000 0.0000 1.0000
vn -1.0000 0.0000 0.0000
vn 0.0000 -1.0000 0.0000
vn 1.0000 0.0000 0.0000
vn 0.0000 0.0000 -1.0000
s off
f 5//1 3//1 1//1
f 3//2 8//2 4//2
f 7//3 6//3 8//3
f 2//4 8//4 6//4
f 1//5 4//5 2//5
f 5//6 2//6 6//6
f 5//1 7//1 3//1
f 3//2 7//2 8//2
f 7//3 5//3 6//3
f 2//4 4//4 8//4
f 1//5 3//5 4//5
f 5//6 1//6 2//6
```

- Dimana (v) merupakan data vertices object dan (vn) adalah vertices untuk normals. (f) melambangkan indices seperti saat kita mau membuat kotak yang terdiri dari 2 segitiga, satu deret f adalah 1 segitiga yang yang dibaca seperti: indicesVertices//indicesNormals
- Lalu kita pindah ke vs, di dalam visual studio ini kita memproses data-data tersebut supaya bisa tertampilkan di window nya yang kita lakukan adalah kita membuat array dimana terdiri dari vertices untuk obj dan vertices untuk normalnya, contohnya adalah dibawah ini:

```
//private readonly float[] _vertices =
//{
//     // Position          Normal
//    -0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f, // Front face
//     0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
//     0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
//     0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
//    -0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
//    -0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f,

//    -0.5f, -0.5f,  0.5f,  0.0f,  0.0f,  1.0f, // Back face
//     0.5f, -0.5f,  0.5f,  0.0f,  0.0f,  1.0f,
//     0.5f,  0.5f,  0.5f,  0.0f,  0.0f,  1.0f,
//     0.5f,  0.5f,  0.5f,  0.0f,  0.0f,  1.0f,
//    -0.5f,  0.5f,  0.5f,  0.0f,  0.0f,  1.0f,
//    -0.5f, -0.5f,  0.5f,  0.0f,  0.0f,  1.0f,

//    -0.5f,  0.5f,  0.5f, -1.0f,  0.0f,  0.0f, // Left face
//    -0.5f,  0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
//    -0.5f, -0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
//    -0.5f, -0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
//    -0.5f, -0.5f,  0.5f, -1.0f,  0.0f,  0.0f,
//    -0.5f,  0.5f,  0.5f, -1.0f,  0.0f,  0.0f,

//     0.5f,  0.5f,  0.5f,  1.0f,  0.0f,  0.0f, // Right face
//     0.5f,  0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
//     0.5f, -0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
//     0.5f, -0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
//     0.5f, -0.5f,  0.5f,  1.0f,  0.0f,  0.0f,
//     0.5f,  0.5f,  0.5f,  1.0f,  0.0f,  0.0f,
```

Data - data diatas bisa kita dapatkan dari file obj nya itu dengan cara pertama memasukan ke List<Vector3> sesuai tipe vertices nya:

```
case "v":
    vertices.Add(new Vector3(float.Parse(words[0]) / 10, float.Parse(words[1]) / 10, float.Parse(words[2]) / 10));
    break;

case "vt":
    textureVertices.Add(new Vector3(float.Parse(words[0]), float.Parse(words[1]),
                                    words.Count < 3 ? 0 : float.Parse(words[2])));
    break;

case "vn":
    normals.Add(new Vector3(float.Parse(words[0]), float.Parse(words[1]), float.Parse(words[2])));
    break;
```

Lalu kita kan punya data segitiga yang dilambangkan (f) di file obj tersebut, itu kita jadikan 1 array yang besar, cara memprosesnya seperti ini:

```
// face
case "f":
    foreach (string w in words)
    {
        if (w.Length == 0)
            continue;
        string[] comps = w.Split('/');

        //vertices
        realVertices.Add(vertices[int.Parse(comps[0]) - 1].X);
        realVertices.Add(vertices[int.Parse(comps[0]) - 1].Y);
        realVertices.Add(vertices[int.Parse(comps[0]) - 1].Z);

        //normal
        realVertices.Add(normals[int.Parse(comps[2]) - 1].X);
        realVertices.Add(normals[int.Parse(comps[2]) - 1].Y);
        realVertices.Add(normals[int.Parse(comps[2]) - 1].Z);
    }
    break;
```

* realVertices adalah List<float>

```
List<float> realVertices = new List<float>();
```

Dengan begitu isi dari realVertices akan sama dengan _vertices yang di comment di atas tersebut

- Lalu selanjutnya yang harus kita lakukan adalah memproses VBO & VAO (vao nya terdiri dari 2 yaitu aPosition & aNormal):

```
//inisialisasi array
_vertexArrayObject = GL.GenVertexArray();
GL.BindVertexArray(_vertexArrayObject);


var vertexLocation = _shader.GetAttribLocation("aPosition");
GL.EnableVertexAttribArray(vertexLocation);
GL.VertexAttribPointer(vertexLocation, 3, VertexAttribPointerType.Float, false, 6 * sizeof(float), 0);

var normalLocation = _shader.GetAttribLocation("aNormal");
GL.EnableVertexAttribArray(normalLocation);
GL.VertexAttribPointer(normalLocation, 3, VertexAttribPointerType.Float, false, 6 * sizeof(float), 3 * sizeof(float));
```

- Untuk rendernya kita memakai GL.DrawArrays dengan Triangles
- Kita juga menambahkan fungsi set untuk merubah Ambient Strength, Shininess, dan Specular Strength guna supaya tiap benda bisa memiliki nilai yang berbeda2 untuk efek shading nya
- Lalu juga ada fungsi setColor untuk merubah warna dari obj yang di proses tersebut

Camera

- Untuk Camera kita menggunakan free move dengan menggunakan fungsi OpenTK yaitu isKeyDown, dengan memencet W, A, S, D, Spasi, Left Control, Mouse
- Untuk W & S kita menambah(W)/mengurang(S) posisi camera dengan mengalikan camera.Front dan speed camera(speed kita tambahi variable sendiri) lalu kita kalikan dengan args.Time sehingga kamera bergerak mulus dan tidak terlalu cepat
- Sama dengan A & D hanya saja yang beda adalah kita bukan menambah/mengurang dengan camera.Front, tetapi dengan camera.Right (A mengurangi, D menambahi)
- Untuk menambah/mengurangi pitch atau yaw dari kamera kita menggunakan mouse dengan menghitung posisi mouse
- Juga ada untuk zoom in dan out menggunakan I(in) dan O(out)
- Untuk Spasi dan Left Control kita menambahkan(Spasi)/mengurangkan(Left Control) posisi dari kamera dengan new Vector3(0, cameraSpeed * (float)args.Time * .5f, 0) sehingga kamera bergerak naik turun terhadap sumbu Y dan tidak dipengaruhi pitch kamera
- Kita juga menambahkan fungsi pada saat menekan tombol F1, F2, F3
- F1 : kamera berotasi memutari keseluruhan scene
- F2 : kamera berotasi memutari Spongebob
- F3 : kamera berotasi memutari Patrick

- Ketiga tersebut kita menggunakan fungsi generateArbRotationMatrix yang sudah dijelaskan di kelas oleh asdos, hanya saja rotasi kamera tersebut tidak akan berguna ketika fungsi untuk menggerakan kamera (yaw & pitch) dibawah ini aktif

```
if (!usingcamera)
{
    if (!IsFocused)
    {
        return;
    }
    if (_firstmove)
    {
        _lastMousePosition = new Vector2(MouseState.X, MouseState.Y);
        _firstmove = false;
    }
    else
    {
        //calc selisih mouse pos
        var deltaX = MouseState.X - _lastMousePosition.X;
        var deltaY = MouseState.Y - _lastMousePosition.Y;
        _lastMousePosition = new Vector2(MouseState.X, MouseState.Y);

        _camera.Yaw += deltaX * sensitivity;
        _camera.Pitch -= deltaY * sensitivity;
    }
}
```

Oleh karena itu kita menambahkan if lagi diatasnya untuk mencegah penggunaan mouse untuk mempengaruhi yaw dan pitch dari kamera
Variabel usingcamera adalah boolean yang akan di set True ketika F1 atau F2 atau F3 ditekan contohnya adalah seperti ini:

```
//spongebob
if (KeyboardState.IsKeyPressed(Keys.F2))
{
    cameraFocus = spongebobmain;
    usingcamera = true;
    _camera.Position = new Vector3(0);
    _objectPos = cameraFocus.getRealPos();
    _camera.Position += new Vector3(0, _objectPos.Y, .2f);
    _camera.Pitch = 0;
    _camera.Yaw = 0;
}
if (KeyboardState.IsKeyDown(Keys.F2))
{
    _objectPos = cameraFocus.getRealPos();
    var axis = new Vector3(0, 1, 0);
    _camera.Position -= _objectPos;
    _camera.Yaw -= _rotationSpeed;
    _camera.Position = Vector3.Transform(_camera.Position,
        generateArbRotationMatrix(axis, _objectPos, -_rotationSpeed).ExtractRotation());
    _camera.Position += _objectPos;


    _camera._front = -Vector3.Normalize(_camera.Position - _objectPos);
}
if (KeyboardState.IsKeyReleased(Keys.F2))
{
    usingcamera = false;
    _camera.Yaw -= 50;
}
```

Di gambar atas ini kita bisa lihat bahwa ketika menekan tombol F2 (IsKeyPressed) kita mengeset usingcamera = true;

- Setelah menonaktifkan fungsi untuk menggerakan yaw dan pitch kamera dengan mouse, selanjutnya yang harus dilakukan adalah menentukan posisi object dan mengembalikan posisi kamera ke 0,0,0, yaitu dengan cara :

```
//spongebob
if (KeyboardState.IsKeyPressed(Keys.F2))
{
    cameraFocus = spongebobmain;
    usingcamera = true;
    _camera.Position = new Vector3(0);
    _objectPos = cameraFocus.getRealPos();
    _camera.Position += new Vector3(0, _objectPos.Y, .2f);
    _camera.Pitch = 0;
    _camera.Yaw = 0;
}
```

- _camera.Position + = new Vector3(0, _objectPos.Y, .2f); berfungsi untuk memberi jarak dari sumbu putar kamera
- Dan kita bisa lihat di akhir ada fungsi IsKeyReleased yang digunakan untuk mengaktifkan kembali fungsi mouse yang tadi di nonaktifkan