

**LAPORAN AKHIR PRAKTIKUM
GRAFIKA KOMPUTER**



DISUSUN OLEH
Hanifah Putri Trisnawati

Rabu 12.15 - E

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS AHMAD DAHLAN**

JULI 2025

**LEMBAR PENGESAHAN ASISTEN
RESPONSI
LAPORAN AKHIR PRAKTIKUM GRAFIKA KOMPUTER**

Disusun oleh:

[Hanifah Putri Trisnawati]

[2200018217]

**Laboratorium Informatika
Program Studi S1 Informatika
Fakultas Teknologi Industri
Universitas Ahmad Dahlan**

Telah disetujui oleh:

PJ Asisten

[Moch.Rizqi Ardi Saputra]

[2200018232]

Daftar Isi

Daftar Isi.....	3
Daftar Gambar.....	5
Kata Pengantar.....	8
BAB I. PENGANTAR OPENGL.....	9
1. Pretest 1.....	9
2. Langkah Praktikum 1.....	12
3. Posttest.....	13
BAB II. ALGORITMA GARIS.....	16
1. Pretest 1	16
2. Langkah Praktikum 2.....	17
3. Posttest 2.....	19
BAB III. INTERPOLASI DAN KURVA.....	21
1. Pretest 3	21
2. Langkah Praktikum 3.....	22
3. Posttest 3.....	24
BAB IV. TRANSFORMASI 2D DAN 3D.....	25
1. Pretest 1	25

2.	Langkah Praktikum 4.....	26
3.	Posttest 4.....	28
	BAB V. PROYEKSI 3D.....	30
1.	Pretest 5	30
2.	Langkah praktikum 5.....	31
3.	Posttest 5.....	32
	BAB VI. REPRESENTASI OBYEK 3D.....	34
1.	Pretest 6	34
2.	Langkah praktikum 6.....	35
3.	Posttest 6.....	39
	BAB VII. KURVA SPLINE	40
1.	Pretest 7	40
2.	Langkah praktikum 7.....	41
3.	Posttest 7.....	43
	BAB VII. TEKNIK REPRESENTASI PERMUKAAN.....	44
1.	Pretest 8	44
2.	Langkah praktikum 8.....	46
3.	Posttest 8.....	47
	BAB IX. TEKNIK PERMODELAN OBYEK 3D	48

1.	Pretest 9	48
2.	Langkah praktikum 9.....	49
3.	Posttest 9.....	50
	BAB X. TEKNIK SUBDIVISI.....	52
1.	Pretest 10	52
2.	Langkah praktikum 10.....	53
3.	Posttest 10.....	54

Daftar Gambar

Gambar 1. 1 tahapan OpenGL pipeline	10
Gambar 1. 1 hasil Output langkah praktikum 1.....	12
Gambar 1. 2 hasil output langkah praktikum setelah klik tombol arah.....	13
Gambar 1. 3 kode program membuat objek segitiga sama sisi	13
Gambar 1. 4 kode program menampilkan objek segitiga sama sisi	14
Gambar 1. 5 menampilkan objek segitiga sama sisi.....	14
Gambar 1. 6 kode program menampilkan objek jajar genjang.....	15
Gambar 1. 7 output menampilkan jajar genjang.....	15
Gambar 2. 1 output program menampilkan 4 garis di kuadran berbeda	18
Gambar 2. 2 kode program membuat titik pada kuadran	18
Gambar 2. 3 kode program untuk membuat diamond	19
Gambar 2. 4 output program diamond.....	20
Gambar 2. 5 output program segitiga warna biru.....	20
Gambar 3. 1 kode program menampilkan interpolasi cosine	23
Gambar 3. 2 output program menampilkan interpolasi cosine.....	23
Gambar 3. 3 kode program menampilkan interpolasi cubic	23
Gambar 3. 4 output program menampilkan interpolasi cubic	24
Gambar 3. 5 kode program mengatur titik	24
Gambar 3. 6 hasil output setelah perubahan titik interpolasi	24
Gambar 4. 1 output langkah praktikum 4 sebelum klik tombol arah	26
Gambar 4. 2 output langkah praktikum 4 setelah klik tombol arah	27
Gambar 4. 3 output penerapan scaling pada obyek	27

Gambar 4. 4 kode program ubah untuk translasi.....	28
Gambar 4. 5 kode program ubah untuk rotasi	28
Gambar 4. 6 kode program untuk scalling	28
Gambar 4. 7 output posttest praktikum 4	29
Gambar 5. 1 hasil output proyeksi kubus	32
Gambar 5. 2 fungsi untuk mengatur proyeksi orthogonal	32
Gambar 5. 3 ubah proyeksi kubus menjadi 3 teapot.....	33
Gambar 5. 4 hasil output posttest praktikum 5	33
Gambar 6. 1 hasil output representasi kubus sebelum klik tombol arah.....	36
Gambar 6. 2 hasil output representasi kubus setelah klik tombol arah	36
Gambar 6. 3 mengubah fungsi cube menjadi cylinder	37
Gambar 6. 4 hasil output representasi tabung sebelum klik tombol arah.....	37
Gambar 6. 5 hasil output representasi tabung setelah klik tombol arah.....	38
Gambar 6. 6 hasil output representasi bola.....	38
Gambar 6. 7 hasil output posttest praktikum 6	39
Gambar 5. 3 ubah proyeksi kubus menjadi segitiga	43
Gambar 5. 4 hasil output proyeksi segitiga sebelum klik tombol arah.....	44
Gambar 5. 5 hasil output proyeksi segitiga setelah klik tombol arah	44
Gambar 5. 6 ubah pryorksi kubus menjadi 3 teapot	45
Gambar 5. 7 hasil output posttest 5 proyeksi 3D (teapot)	45
Gambar 6. 1 hasil output representasi kubus sebelum klik tombol panah.....	48
Gambar 6. 2 hasil output representasi kubus setelah klik tombol panah.....	49
Gambar 6. 3 merubah fungsi cube menjadi cylinder	49
Gambar 6. 4 hasil output representasi wire teapot.....	51
Gambar 7. 1 hasil output langkah praktikum 7 kurva spline	42

Gambar 7. 2 kode program yang mengontrol titik pada kurva.....	42
Gambar 7. 3 kode program membuat kurva spline Catmull-rom.....	42
Gambar 7. 4 hasil output kurva spline Catmull-Rom.....	43
Gambar 7. 5 hasil output kurva spline bezier	43
Gambar 7. 6 mengubah koordinat, warna, ukuran dan titik kurva	44
Gambar 8. 1 output langkah praktikum 8 representasi permukaan.....	47
Gambar 8. 2 output langkah pratikum 8 setelah klik tombol arah.....	47
Gambar 8. 3 output posttest 8 representasi permukaan	48
Gambar 9. 1 hasil output langkah praktikum 9 pemodelan objek 3D	50

Gambar 9. 2 ubah kode program jumlah eclipse menjadi 2	50
Gambar 9. 3 memperbarui fungsi eclipse menjadi 2	50
Gambar 9. 4 ubah warna eclipse menjadi ungu.....	51
Gambar 9. 5 hasil output program posttest 9	51
Gambar 10. 1 hasil output langkah praktikum 10	53
Gambar 10. 2 ubah icosahevron pada kode program	54
Gambar 10. 3 output posttest 10 menampilkan subdivisi dengan ring.....	54

Kata Pengantar

Puji syukur saya panjatkan kepada Allah SWT karena berkat rahmat dan karunia-Nya, saya dapat menyusun laporan responsi praktikum Grafika Komputer ini. Laporan ini dibuat sebagai bagian dari proses pembelajaran dan penilaian dalam mata kuliah Grafika Komputer. Praktikum ini bertujuan untuk mendalami pemahaman saya mengenai konsep-konsep dasar grafika komputer. Melalui praktikum ini, kami mendapatkan kesempatan untuk mengaplikasikan teori dan praktik yang telah dipelajari ke dalam implementasi praktis, sehingga dapat meningkatkan keterampilan teknis dan analisis saya.

Dalam laporan ini, saya akan menjelaskan secara rinci mengenai proses dan hasil dari setiap praktikum yang telah dilakukan. Saya berharap laporan ini dapat menunjukkan pemahaman saya terhadap materi yang telah diajarkan serta kemampuan saya dalam menerapkannya secara praktis. Saya menyadari bahwa keberhasilan dalam penyusunan laporan ini tidak terlepas dari dukungan berbagai pihak.

Oleh karena itu, saya mengucapkan terima kasih yang sebesar-besarnya kepada dosen pengampu, Pak Ahmad Azhari, asisten praktikum, serta teman-teman yang telah memberikan bimbingan, dukungan, dan kontribusinya selama proses praktikum berlangsung. Akhir kata, semoga laporan ini dapat memberikan manfaat bagi pengembangan pengetahuan dan keterampilan di bidang grafika komputer, serta menjadi referensi yang berguna bagi pembaca.

BAB I. PENGANTAR OPENGL

1. Pretest 1

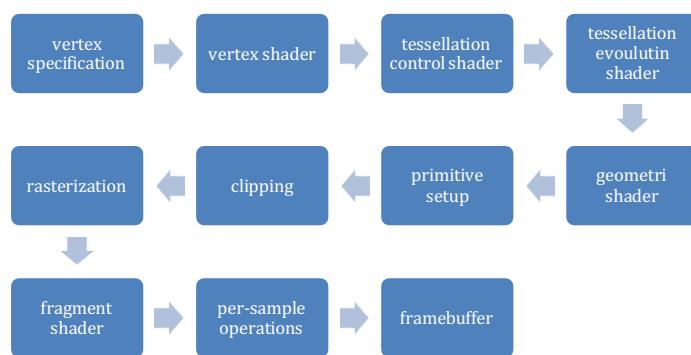
- 1) Apa saja yang bisa dilakukan dengan library OpenGL?
- 2) Gambarkan kemudian jelaskan tahapan OpenGL pipeline!

JAWAB

- 1) Open Graphics Library (OpenGL) merupakan sebuah library yang menyediakan beberapa set prosedur dan berbagai fungsi yang memungkinkan digunakan untuk menggambar sebuah objek dua dimensi (2D) dan tiga dimensi (3D).
Beberapa hal yang bisa dilakukan dengan library OpenGL diantaranya :

- Menggambar Objek 2D dan 3D : OpenGL digunakan untuk menggambar objek 2D dan 3D, seperti titik , garis, lingkaran, persegi, kubus, bola, dan lain-lain.
- Pembuatan shading dan transformasi : library ini memungkinkan pengguna untuk membuat shading (pemberian warna) dan transformasi (rotasi, translasi, skala) pada objek grafis.
- Penggunaan interface sistem window dan input : OpenGL tidak menyediakan interface sistem window dan input, sehingga aplikasi yang menggunakan OpenGL biasanya memerlukan library tambahan seperti GLUT (GL Utikity Toolkit) untuk mengatur window dan input user.
- Menghasilkan Grafis 2D dan 3D : OpenGL mendefinisikan cross-bahasa dan cross-platform API yang memungkinkan pengembangan aplikasi yang menghasilkan grafis 2D dan 3D pada berbagai platform, seperti Windows, Linux, dan lain-lain.
- Dukungan Multi-Language : OpenGL dapat digunakan dengan berbagai bahasa pemrograman, seperti C/C++, Java, Delphi, Visual Basic, dan lain-lain.

- 2) OpenGL pipeline adalah utusan proses rendering dalam OpenGL. Dua informasi grafis, *vertex-based* data dan *pixel-base* data, diproses melalui pipeline, dikombinasikan bersamaan dan kemudian dicatat ke dalam *framebuffer*. OpenGL juga dapat memproses balik data dari *framebuffer* ke aplikasi yang kita miliki. Berikut adalah tahapan OpenGL pipeline dan penjelasannya :



Gambar 1. 1 tahapan OpenGL pipeline

- Per-Vertex Operation
 - Vertex specification : Membuat daftar vertex yang akan digunakan untuk rendering.
 - Vertex shader : Mengolah vertex dengan menggunakan shader yang disebut Vertex shader. Vertex shader diprogram untuk menghitung posisi akhir setiap vertex.
- Per-Primitive Operation
 - Tessellation Control Shader : Mengolah primitive dengan menggunakan shader yang disebut Tessellation Control Shader. Shader ini diprogram untuk menghitung jumlah vertex yang dibutuhkan untuk menghasilkan primitive yang lebih halus.
 - Tessellation Evolution Shader : Mengolah vertex yang dihasilkan oleh tessellation control shader dengan menggunakan shader yang

disebut Tessellation Evolution Shader. Shader ini diprogram untuk menghitung posisi akhir setiap vertex yang dihasilkan oleh tessellation control shader.

- Per-Primitive Operation

Geometry Shader : Mengolah primitive dengan menggunakan shader yang disebut Geometry Shader. Shader ini diprogram untuk menghitung jumlah vertex yang dibutuhkan untuk menghasilkan primitive yang lebih halus.

- Primitive Setup

Primitive Setup : Mengumpulkan vertex yang dihasilkan oleh geometry shader menjadi primitive yang lebih sederhana seperti garis, titik, atau segitiga.

- Clipping

Clipping: Menghapus area primitive yang jatuh di luar View-Volume (yaitu di luar layar).

- Rasterization

Rasterization: Mengkonversi primitive menjadi pixel yang mendekati bentuk primitive. Pixel diproses untuk melihat apakah mereka berada dalam perbatasan primitive. Jika tidak, mereka diabaikan. Jika berada dalam perbatasan, mereka dipindahkan ke tahap berikutnya.

- Fragment processing

Fragment Shader: Mengolah pixel yang lolos tes rasterization dengan menggunakan shader yang disebut Fragment Shader. Shader ini diprogram untuk menghitung warna akhir setiap pixel yang dilihat di layar.

- Per-Sample Operation

Per-Sample Operations : Melakukan beberapa tes pada pixel, seperti tes milik pixel, tes scissor, tes alpha, tes stencil, dan tes depth. Tes ini

memastikan bahwa pixel yang dikirim ke framebuffer sesuai dengan aturan rendering.

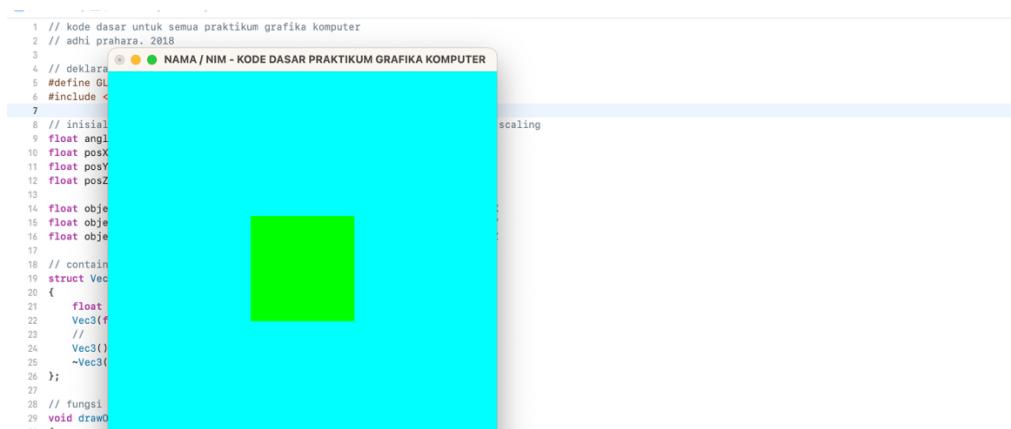
- Framebuffer

Framebuffer: Pixel yang lolos semua tes di simpan dalam Framebuffer, yang lebih spesifiknya adalah Default-Framebuffer. Pixel ini yang dilihat di layar Anda.

2. Langkah Praktikum 1

Persiapan:

- 1) Buka C++ dan buat project baru. Setting OpenGL library pada Visual Studio C/C++.
- 2) Bila terdapat error maka copy-kan glut32.dll, glew32.dll, dan opengl32.dll dari folder library OpenGL yang anda ekstrak ke dalam folder dimana .exe program anda berada.
- 3) Hasil tampilan jika sudah berhasil ditunjukkan pada gambar 1.1 dan 1.2



```
1 // kode dasar untuk semua praktikum grafika komputer
2 // adhi prahara. 2018
3
4 // deklara
5 #define GL
6 #include <
7
8 // inisial
9 float angl
10 float posx
11 float posy
12 float posz
13
14 float obje
15 float obje
16 float obje
17
18 // contain
19 struct Vec
20 {
21     float
22     Vec3f
23     //
24     Vec3()
25     ~Vec3()
26 };
27
28 // fungsi
29 void drawO
30 {
31     // obje
32     // fun_
33     // saat diwarnai, ditransformasi dan sebagainya
34     glPushMatrix();
35
36     // operasi transformasi rotasi obyek ke arah kanan-kiri
37     glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);
38
39     glPushMatrix();
40 }
```

Gambar 1. 1 hasil output langkah praktikum 1

- 4) Klik tombol arah bawah, kiri dan kanan agar objek 3D dapat terlihat bentuk keseluruhannya seperti pada Gambar 1.2.

```

1 // Kode dasar untuk semua praktikum grafika komputer
2 // adhi prahara. 2018
3
4 // deklara
5 #define GL
6 #include <
7
8 // inisial
9 float angl
10 float posX
11 float posY
12 float posZ
13
14 float obje
15 float obje
16 float obje
17
18 // contain
19 struct Vec
20 {
21     float
22     Vec3(f
23     //
24     Vec3()
25     ~Vec3(
26 };
27
28 // fungsi
29 void drawO
30 {
31     // oby
32     // fun
33     // saat diwernai, ditransformasi dan sebagainya
34     glPushMatrix();
35
36     // operasi transformasi rotasi obyek ke arah kanan-kiri
37     glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);
38
39     glPushMatrix();
40
41     // operasi transformasi rotasi obvek ke arah atas-bawah

```

Gambar 1.2 hasil output langkah praktikum 1 setelah klik tombol arah

3. Posttest 1

- 1) Buatlah bentuk segitiga sama sisi berwarna merah dengan GL_TRIANGLES!
- 2) Buatlah bentuk jajar genjang berwarna magenta dengan GL_QUADS!

Langkah-langkah :

- 1) Untuk membuat segitiga sama sisi berwarna merah, kita perlu mengubah program.
Ubah program pada bagian yang ditunjukkan oleh Gambar 1.3 di bawah ini :

```

29 // fungsi untuk menyimpan posisi
30 void drawObject()
31 {
32     glPushMatrix();
33     glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);
34     glPushMatrix();
35     glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);
36
37     // Segitiga sama sisi
38     glColor3f(1.0f, 0.0f, 0.0f); // merah
39     glBegin(GL_TRIANGLES);
40         glVertex3f( 0.0f, 0.6f, 0.0f); // atas
41         glVertex3f(-0.5f, -0.3f, 0.0f); // kiri bawah
42         glVertex3f( 0.5f, -0.3f, 0.0f); // kanan bawah
43     glEnd();
44     glPopMatrix();
45
46 }

```

Gambar 1. 3 kode program untuk membuat objek segitiga sama sisi

Penjelasan singkat :

- Fungsi ‘**glColor3f**’ digunakan untuk merubah warna pada objek yang akan kita buat. Pada fungsi tersebut kita menggunakan warna merah dengan kode warna (1.0f, 0.0f, 0.0f).
- Pada program diatas kita juga menggunakan GL_TRIANGLES yang digunakan untuk menggambarkan titik-titik sebelum dihubungkan menjadi sisi-sisi segitiga.

- 2) Ubah kode program pada bagian display untuk menampilkan objek yang telah kita buat tadi.

```

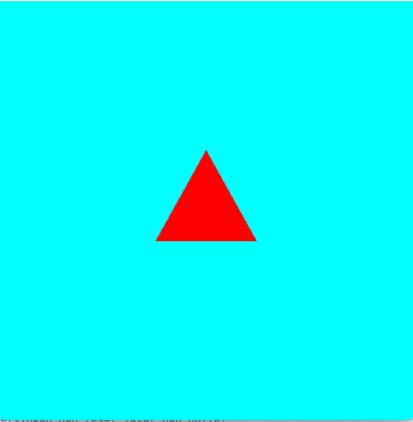
48 // taruh semua obyek yang akan digambar di fungsi display()
49 void display()
50 {
51     // bersihkan dan reset layar dan buffer
52     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
53     glLoadIdentity();
54
55     // posisikan kamera pandang
56     // dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)
57     gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);
58
59     // panggil fungsi untuk menggambar obyek
60     drawObject();
61
62     // tampilkan obyek ke layar
63     // gunakan glFlush() bila memakai single buffer
64     // gunakan glutSwapBuffers() bila memakai double buffer
65     glutSwapBuffers();
66 }

```

Gambar 1. 4 kode program untuk menampilkan objek segitiga

Penjelasan singkat :

- Pada fungsi ‘**void display()**’, panggil fungsi untuk menggambar seitiha yang telah kita buat tadi dengan menggunakan ‘**drawTriangle()**’.
- 3) Setelah kode program telah selesai diubah maka lakukan compile dan run pada program maka akan tertampil objek segitiga sama sisi seperti pada Gambar 1.5.



```

19 struct Vec3
20 {
21     float x; float y; float z;
22     Vec3() : x(0.0f), y(0.0f), z(0.0f) {}
23     // ...
24     ~Vec3();
25 };
26 };
27
28 // fungsi
29 void drawObject()
30 {
31     glPushMatrix();
32
33     glPushMatrix();
34
35     glTranslatef(-0.7f, -0.5f, 0.0f); // kiri bawah
36     glPushMatrix();
37     glTranslatef(0.3f, -0.5f, 0.0f); // kanan bawah
38     glPushMatrix();
39     glTranslatef(0.7f, 0.5f, 0.0f); // kanan atas
40     glPushMatrix();
41     glTranslatef(-0.3f, 0.5f, 0.0f); // kiri atas
42     glPopMatrix();
43     glPopMatrix();
44
45     glPopMatrix();
46 }
47
48 // tampilan
49 void display()
50 {
51     // bereskan dan setel tampilan buffer
52     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
53     glLoadIdentity();
54 }
```

Gambar 1. 5 output menampilkan objek segitiga sama sisi

Langkah – langkah :

- 1) Untuk membuat jajar genjang berwarna magenta, kita perlu mengubah program.

Ubah program pada bagian yang ditunjukkan oleh Gambar 1.6 di bawah ini :

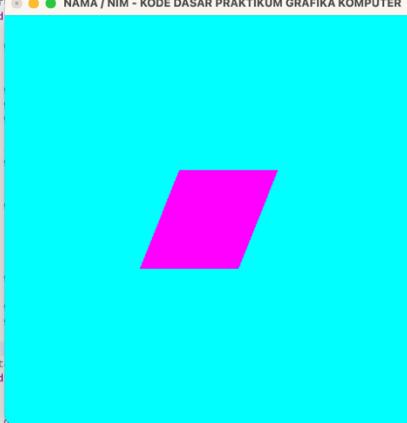
```

28 // fungsi untuk menggambar obyek
29 void drawObject()
30 {
31     glPushMatrix();
32
33     // Rotasi objek jika diperlukan
34     glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);
35     glPushMatrix();
36     glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);
37
38     // Set warna magenta
39     glColor3f(1.0f, 0.0f, 1.0f);
40
41     // Gambar jajar genjang (parallelogram)
42     glBegin(GL_QUADS);
43         glVertex3f(-0.7f, -0.5f, 0.0f); // kiri bawah
44         glVertex3f( 0.3f, -0.5f, 0.0f); // kanan bawah
45         glVertex3f( 0.7f, 0.5f, 0.0f); // kanan atas
46         glVertex3f(-0.3f, 0.5f, 0.0f); // kiri atas
47     glEnd();
48
49     glPopMatrix();
50     glPopMatrix();
51 }
```

Gambar 1. 6 kode program untuk menampilkan objek jajar genjang

- 2) Setelah kode program telah selesai diubah maka lakukan compile dan run pada program maka akan tertampil objek segitiga sama sisi seperti pada Gambar 1.7.

```
26 };
27
28 // f NAMA / NIM - KODE DASAR PRAKTIKUM GRAFIKA KOMPUTER
29 void
30 {
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51 }
52
53 // t
54 void
55 {
56
57
58 glLoadIdentity();
59
60 // memuatkan komponen awal
```



Gambar 1. 7 output menampilkan objek jajar genjang

BAB II. ALGORITMA GARIS

1. Pretest 1

- 1) Jelaskan tahapan pembangkitan garis dengan algoritma DDA!
- 2) Jelaskan tahapan pembangkitan garis dengan algoritma Bresenham!

JAWAB

- 1) Algoritma Digital Differential Analyzer (DDA) adalah algoritma pembentukan garis berdasarkan perhitungan dx maupun dy . Garis dibuat dengan menentukan dua endpoint, yaitu titik awal dan titik akhir. Tahapan untuk melakukan pembangkitan garis dengan algoritma DDA adalah sebagai berikut :
 1. Tentukan dua titik yang akan dihubungkan dalam pembentukan garis.
 2. Tentukan salah satu sebagai titik awal (x_1, y_1) dan titik akhir (x_2, y_2) .
 3. Hitung $dx = x_2 - x_1$ dan $dy = y_2 - y_1$
 4. Tentukan step, yaitu jarak maksimum jumlah penambahan nilai x atau nilai y, dengan ketentuan:
 - a. bila $|dx| > |dy|$ maka $step = |dx|$
 - b. bila tidak, maka $step = |dy|$
 5. Hitung penambahan koordinat pixel dengan persamaan:
 $x_inc = dx / step$
 $y_inc = dy / step$
 6. Koordinat selanjutnya $(x+x_inc, y+y_inc)$
 7. Plot pixel pada layar, nilai koordinat hasil perhitungan dibulatkan.
 8. Ulangi step nomor 6 dan 7 untuk menentukan posisi pixel berikutnya sampai $x = x_1$ atau $y = y_1$.
- 2) Algoritma Bresenham merupakan suatu algoritma (pendekatan) yang dikreasikan

oleh bresenham yang tidak kalah akurat dan efisien dengan algoritma primitif lainnya (seperti DDA). Bagian pengkonversian (scan-knversi) garis akan melakukan kalkulasi untuk penambahan nilai-nilai integer (yang dibutuhkan untuk membentuk garis) yang disesuaikan dengan tipe grafik yang dipakai oleh layar komputer (keadaan monitor pc) kita. . Tahapan untuk melakukan pembangkitan garis dengan algoritma Bresenham adalah sebagai berikut :

1. Tentukan 2 titik yang akan dihubungkan dalam pembentuk garis.
2. Tentukan salah satu titik di sebelah kiri sebagai titik awal, yaitu (X_0, Y_0) dan titik lainnya sebagai titik akhir (X_1, Y_1)
3. Hitung Dx , Dy , $2DX$ dan $2Dy - 2Dx$
4. Hitung parameter $P_0 = 2Dy - 2Dx$
5. Untuk setiap X_1 sepanjang jalur garis, dimulai dengan $k=0$,
 - a. bila $P_k < 0$, maka titik selanjutnya adalah $(X_k + 1, Y_k)$ dan $P_{k+1} = P_k + 2Dy$
 - b. bila tidak, maka titik selanjutnya adalah $(X_k + 1, Y_k + 1)$ dan $P_{k+1} = P_k + 2Dy - 2Dx$
6. Ulangi langkah sebelumnya untuk menentukan posisi selanjutnya, sampai $X=X_1$ dan $Y=Y_1$

2. Langkah Praktikum 2

Langkah-langkah praktikum :

- 1) Buka aplikasi Dev C++ dan buat projek baru dengan console.
- 2) Copy file program txt yang diberikan oleh asisten praktikum ke dalam projek yang saja dibuat..
- 3) Isi linker pada menu parameter dengan -lopengl32, -lfreeglut, dan -lglu32
- 4) Kemudian, pada menu Directories klik pada menu Library Directories dan masukkan folder x64.
- 5) Kemudian, masih pada menu Directories klik pada menu Include Directories dan

masukkan folder include.

- 6) Setelah itu lakukan compile dan run untuk melihat output dari program tersebut.



```
Algoritma DDA
34 void lineDDA(V<-> <-> <-> V> <-> <-> V>)
35     for (int i = 0; i < 4; i++)
36     {
37         px = pX;
38         py = pY;
39         pz = pZ;
40         glVertex3f(pX, pY, pZ);
41     }
42 }
43
44 void lineDDAY()
45 {
46     // hitung
47     int dy = pY - pY0;
48     int dx = pX - pX0;
49     float m = dy / dx;
50     float im = -1 / m;
51
52     // mulai mewarnai
53     glBegin(GL_LINES);
54     // koordinat awal
55     glVertex3f(pX0, pY0, pZ0);
56     // koordinat akhir
57     glVertex3f(px, py, pz);
58 }
59
60 // fungsi untuk menggambar garis
61 void lineDDAY()
62 {
63     // hitung
64     int dy = pY - pY0;
65     int dx = pX - pX0;
66     float m = dy / dx;
67     float im = -1 / m;
68
69     // mulai mewarnai
70     glBegin(GL_LINES);
71     // koordinat awal
72     glVertex3f(pX0, pY0, pZ0);
73     // koordinat akhir
74     glVertex3f(px, py, pz);
75 }
76
77 // kenaikan titik
78 for (int i = 0; i < 4; i++)
79 {
80     px = pX;
81     py = pY + 1; // Yn+1 = Yn + 1
82     pz = pZ;
83     glVertex3f(px, py, pz);
84 }
85
86 // koordinat titik akhir
```

Gambar 2. 1 output program menampilkan 4 garis di 4 kuadran berbeda

Penjelasan singkat :

Pada output diatas terbentuk 4 garis yang berada di 4 kuadran yang berbeda. Cara merubah titik yang berhubungan dengan titik lain bisa dilihat pada Gambar 2.2.

```
122
123     // gambar sumbu
124     Vec3 sbY1 = Vec3( 0.0f, -300.0f, 0.0f);
125     Vec3 sbY2 = Vec3( 0.0f, 300.0f, 0.0f);
126     Vec3 sbX1 = Vec3(-300.0f, 0.0f, 0.0f);
127     Vec3 sbX2 = Vec3( 300.0f, 0.0f, 0.0f);
128     lineDDA(sbX1, sbX2);
129     lineDDA(sbY1, sbY2);
130     // kuadran 1
131     Vec3 point1 = Vec3( 100.0f, 100.0f, 0.0f);
132     Vec3 point2 = Vec3( 200.0f, 120.0f, 0.0f);
133     lineDDA(point1, point2);
134     // kuadran 2
135     point1 = Vec3(-100.0f, 100.0f, 0.0f);
136     point2 = Vec3(-120.0f, 200.0f, 0.0f);
137     lineDDA(point1, point2);
138     // kuadran 3
139     point1 = Vec3(-100.0f, -100.0f, 0.0f);
140     point2 = Vec3(-200.0f, -120.0f, 0.0f);
141     lineDDA(point1, point2);
142     // kuadran 4
143     point1 = Vec3( 100.0f, -100.0f, 0.0f);
144     point2 = Vec3( 120.0f, -200.0f, 0.0f);
145     lineDDA(point1, point2);
146
147     glPopMatrix();
148     glPopMatrix();
149 }
```

Gambar 2. 2 kode program untuk membuat titik dan garis pada kuadran

- Pada baris 124 sampai 129 digunakan untuk membentuk sumbu yang menciptakan garis nyata yang akan memisahkan antara kuadran 1 sampai

- Pada baris 130 sampai 133 digunakan untuk membuat titik 1 dengan berada pada koordinat (100, 100) dan titik 2 berada pada koordinat (200, 120) yang kemudian akan disambungkan dengan garis bresenham.
- Pada baris 134 sampai 137 digunakan untuk membuat titik 1 dengan berada pada koordinat (-100, 100) dan titik 2 berada pada koordinat (-120, 200) yang kemudian akan disambungkan dengan garis bresenham.
- Pada baris 138 sampai 141 digunakan untuk membuat titik 1 dengan berada pada koordinat (-100, -100) dan titik 2 berada pada koordinat (-200, -120) yang kemudian akan disambungkan dengan garis bresenham.
- Pada baris 142 sampai 145 digunakan untuk membuat titik 1 dengan berada pada koordinat (100, -100) dan titik 2 berada pada koordinat (120, -200) yang kemudian akan disambungkan dengan garis bresenham.

3. Posttest 2

- 1) Buatlah bentuk diamond berwarna kuning dengan algoritma garis DDA!

```

111 // program menggunakan opengl
112 void drawObject()
113 {
114     glPushMatrix();
115     // operasi transformasi rotasi obyek ke arah kanan-kiri
116     glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);
117     glPushMatrix();
118     // operasi transformasi rotasi obyek ke arah atas-bawah
119     glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);
120     // set warna diamond ke kuning
121     glColor3f(1.0f, 1.0f, 0.0f);
122
123     // Titik-titik diamond lebih kecil (belah ketupat)
124     Vec3 top    = Vec3(0.0f, 50.0f, 0.0f);
125     Vec3 right = Vec3(30.0f, 0.0f, 0.0f);
126     Vec3 bottom = Vec3(0.0f, -50.0f, 0.0f);
127     Vec3 left   = Vec3(-30.0f, 0.0f, 0.0f);
128
129     // Gambar garis-garis diamond dengan algoritma DDA
130     lineDDA(top, right);
131     lineDDA(right, bottom);
132     lineDDA(bottom, left);
133     lineDDA(left, top);
134 }
135
136 glPopMatrix();
137 glPopMatrix();
138 }
...

```

Gambar 2. 3 kode program untuk membuat diamond

- 2) Lakukan compile dan run untuk melihat output

```
110 // fungsi @.●
111 void draw()
112 {
113     glPushAttrib(GL_ALL_ATTRIB_BITS);
114     // on
115     glRotatef(45, 0, 1, 0);
116     glPushAttrib(GL_ALL_ATTRIB_BITS);
117     // on
118     glRotatef(45, 0, 1, 0);
119
120     // set
121     glColor3f(1, 1, 0);
122
123     // Trans
124     Vec3 v1 = Vec3(-1, -1, 0);
125     Vec3 v2 = Vec3(1, -1, 0);
126     Vec3 v3 = Vec3(0, 1, 0);
127     Vec3 v4 = Vec3(0, -1, 0);
128
129     // Gambar
130     line(v1, v2);
131     line(v2, v3);
132     line(v3, v4);
133     line(v4, v1);
134
135     glPopAttrib();
136     glPopAttrib();
137 }
138
139
140 // taruh semua obyek yang akan digambar di fungsi display()
141 void display()
142 {
143     // bersihkan dan reset layar dan buffer
144     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
145     glLoadIdentity();
146 }
```

Gambar 2. 4 output program menampilkan diamond berwarna kuning

3) Buatlah bentuk segitiga berwarna biru dengan algoritma bresenham!

```
1 #define GL_SILENCE_DEPRECATION
2 #include <GL/gl.h>
3 #include <GL/glu.h>
4 #include <math.h>
5
6 int SCR_X = 800;
7 int SCR_Y = 600;
8
9 // Translasi
10 float cx = 400;
11 float cy = 300;
12 float dx = 100;
13 float dy = 100;
14 float dz = 0;
15 float rx = 0;
16 float ry = 0;
17
18 // Strukturnya
19 struct Point {
20     float x;
21     Vec2d v;
22     ~Point() { delete v; }
23 };
24 };
25
26 // Deklarasi
27 void line();
28
29 // Algoritma
30 void lineBresenham();
31 {
32     int dY = point2.Y - point1.Y;
33     int dX = point2.X - point1.X;
34
35     int yi = 1;
36     if (dY < 0) {
37         yi = -1;
38         dY = -dY;
39     }
40
41     glBegin(GL_TRIANGLES);
42 }
```

Gambar 2. 5 output program menampilkan segitiga berwarna biru

BAB III. INTERPOLASI DAN KURVA

1. Pretest 3

- 1) Jelaskan tahapan interpolasi linear!
- 2) Jelaskan tahapan interpolasi kubik!

JAWAB

- 1) Interpolasi linier merupakan polinomial tingkat pertama dan melalui suatu garis lurus pada setiap dua titik masukan yang berurutan. Dua titik masukan tersebut digunakan untuk menaksir harga-harga tengahan diantara titik-titik data yang telah tepat. Tahapan interpolasi linear adalah sebagai berikut :
 - a. Bila terdapat dua titik yang akan diinterpolasi yaitu (x_0, y_0) sampai (x_1, y_1) .
 - b. Bila jarak (x_0, y_0) sampai (x_1, y_1) dimisalkan 1 (dinormalisasi) dan diketahui jarak awal (x_0, y_0) sampai titik sela (x, y) adalah u maka :

$$y = y_0 \cdot (1 - u) + y_1 \cdot u$$

$$\text{Dimana } u = \frac{x - x_0}{x_1 - x_0}$$

- 2) Interpolasi kubik adalah cara menemukan kurva yang menghubungkan satu atau lebih titik data dengan derajat tiga atau kurang. Tahapan interpolasi kubik adalah sebagai berikut :
 - a. Menggunakan fungsi pangkat tiga atau kubik untuk melakukan interpolasi.
 - b. Interpolasi kubik memerlukan 2 titik tambahan di ujung 2 titik utama untuk interpolasi.
 - c. Bila terdapat 4 titik yang akan diinterpolasi yaitu $(x_0, y_0), (x_1, y_1), (x_2, y_2)$ sampai (x_3, y_3) dan jarak tersebut dinormalisasi menjadi 1 sedangkan jarak titik awal (x_0, y_0) sampai titik sela (x, y) adalah u dari dua titik tersebut maka persamaannya : $y = au^3 + bu^2 + cu + d$

Dimana :

- $a = y_3 - y_2 - y_0 + y_1$
 - $b = 2y_0 - 2y_1 - y_3 + y_2$
 - $c = y_2 - y_0$
 - $d = y_1$

2. Langkah Praktikum 3

Langkah-langkah praktikum sesuai modul :

- 1) Buka aplikasi Dev C++ dan buat projek baru dengan console.
 - 2) Copy file program txt yang diberikan oleh asisten praktikum ke dalam projek yang saja dibuat..
 - 3) Isi linker pada menu parameter dengan -lopengl32, -lfreelut, dan -lglu32
 - 4) Kemudian, pada menu Directories klik pada menu Library Directories dan masukkan folder x64.
 - 5) Kemudian, masih pada menu Directories klik pada menu Include Directories dan masukkan folder include.
 - 6) Setelah itu lakukan compile dan run untuk melihat output dari program tersebut.

The screenshot shows a terminal window with the following content:

```
pertermuan3 > pertermuan3 > C\main > No Selection
```

```
1 #define NAMA "NAMA / NIM - LP 3 PRAKTIKUM GRAFIKA KOMPUTER"
2 #include <iostream>
3 #include <math.h>
4
5 // ini untuk menulis ke layar
6 float x,y,z;
7 float x1,y1,z1;
8 float x2,y2,z2;
9 float x3,y3,z3;
10
11 float x4,y4,z4;
12 float x5,y5,z5;
13 float x6,y6,z6;
14
15 #define PI 3.14159265358979323846
16 #define DEG_TO_RAD(x) ((x) * PI / 180)
17 #define RAD_TO_DEG(x) ((x) * 180 / PI)
18
19 // konstruktor
20 struct Vector3D
21 {
22     float x,y,z;
23     Vector3D();
24     // ...
25     Vector3D(float x, float y, float z);
26     ~Vector3D();
27 };
28
29 // langkah
30 // enum
31 enum INTERP_TYPE
32 {
33     INTERP_POINTS = 0,
34     INTERP_LINES = 1,
35 }
```

On the right side of the terminal, there is a 3D plot showing a line segment with six points. The line starts at the origin (0,0,0) and ends at approximately (1.5, 1.5, 1.5). The points are represented by small green dots connected by a green line. The axes are labeled X, Y, and Z.

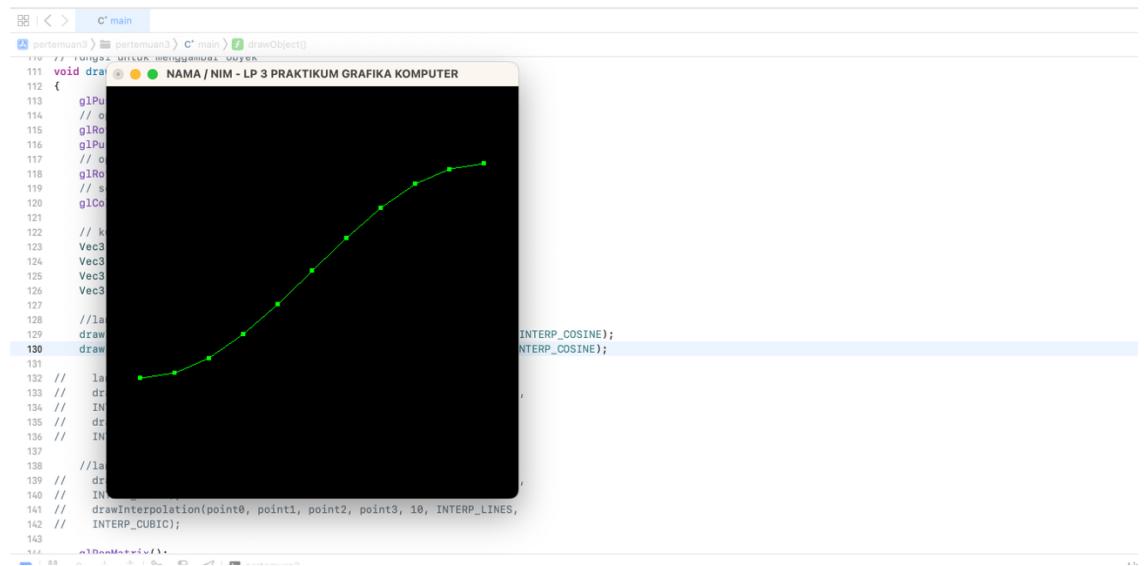
Gambar 3.1 output program menampilkan 11 yang menghasilkan kurva

- 7) Ubah fungsi drawObject() untuk menerapkan interpolasi cosine.

```
122 // kuadran
123 Vec3 point0 = Vec3(-300.0f, -200.0f, 0.0f);
124 Vec3 point1 = Vec3(-200.0f, -100.0f, 0.0f);
125 Vec3 point2 = Vec3( 200.0f, 100.0f, 0.0f);
126 Vec3 point3 = Vec3( 300.0f, 250.0f, 0.0f);
127
128 // langkah 10 (interpolasi linear)
129 drawInterpolation(point0, point1, point2, point3, 10, INTERP_POINTS, INTERP_COSINE);
130 drawInterpolation(point0, point1, point2, point3, 10, INTERP_LINES, INTERP_COSINE);
131 }
```

Gambar 3. 2 kode program menampilkan interpolasi cosine

- 8) Jalankan program hasil seperti ditunjukkan pada gambar 3.3



```
pertemuan3 > pertemuan3 > C:\main > drawObject()
111 void dra @ NAMA / NIM - LP 3 PRAKTIKUM GRAFIKA KOMPUTER
112 {
113     glPU
114     // o
115     glRo
116     glPU
117     // o
118     glRo
119     // s
120     glCo
121
122     // k
123     Vec3
124     Vec3
125     Vec3
126     Vec3
127
128     // la
129     draw
130     draw
131     // la
132     // dr
133     // IN
134     // dr
135     // IN
136     // IN
137
138     // la
139     // dr
140     // IN
141     // drawInterpolation(point0, point1, point2, point3, 10, INTERP_LINES,
142     // INTERP_CUBIC);
143
```

Gambar 3. 3 output program menampilkan interpolasi cosine

- 9) Ubah drawObject untuk menerapkan interpolasi cubic.

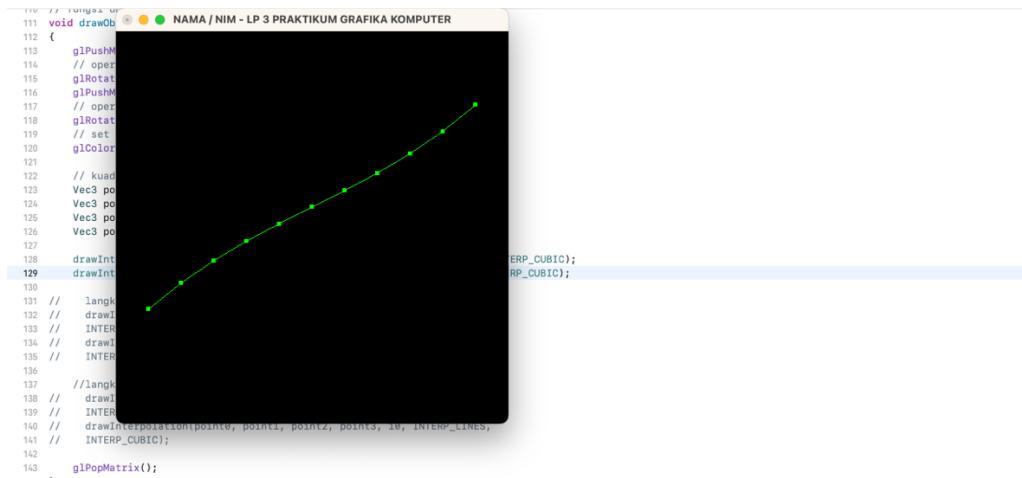
```

121
122 // kuadran
123 Vec3 point0 = Vec3(-300.0f, -200.0f, 0.0f);
124 Vec3 point1 = Vec3(-200.0f, -100.0f, 0.0f);
125 Vec3 point2 = Vec3(200.0f, 100.0f, 0.0f);
126 Vec3 point3 = Vec3(300.0f, 200.0f, 0.0f);
127
128 drawInterpolation(point0, point1, point2, point3, 10, INTERP_POINTS, INTERP_CUBIC);
129 drawInterpolation(point0, point1, point2, point3, 10, INTERP_LINES, INTERP_CUBIC);
130

```

Gambar 3. 4 kode program menampilkan interpolasi cubic

10) Jalankan program hasil seperti ditunjukkan pada gambar 3.4



The screenshot shows a code editor with a C++ file containing OpenGL rendering code. The code includes matrix operations like glPushMatrix, glPopMatrix, glRotat, and glColor, and calls to drawInterpolation for points and lines. A rendered 3D plot is shown on a black background, featuring a green cubic curve connecting ten blue circular points. The curve is smooth and follows a parabolic shape between the points.

```

110 // fungsi ur
111 void drawOb
112 {
113     glPushM
114     // oper
115     glRotat
116     glPushM
117     // oper
118     glRotat
119     // set
120     glColor
121
122     // kuad
123     Vec3 po
124     Vec3 po
125     Vec3 po
126     Vec3 po
127
128     drawInt
129     drawInt
130
131     // langk
132     drawl
133     // INTER
134     // drawl
135     // INTER
136
137     //langk
138     drawl
139     // INTER
140     // drawInterpolation(point0, point1, point2, point3, 10, INTERP_LINES,
141     // INTERP_CUBIC);
142
143     glPopMatrix();

```

Gambar 3. 4 output program menampilkan interpolasi cubic

3. Posttest 3

- 1) Buatlah bentuk lembah dengan interpolasi cubic dengan jumlah titik n=10.
- 2) Ubah kode program untuk 10 titik

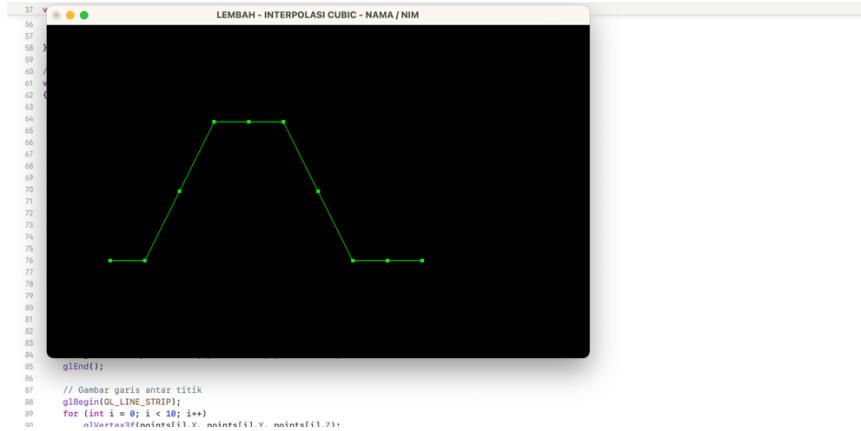
```

66
67 // 10 titik menyerupai bentuk lembah [
68 Vec3 points[10] = {
69     Vec3(-300, -100, 0), // Titik 1
70     Vec3(-250, -100, 0), // Titik 2
71     Vec3(-200, 0, 0), // Titik 3
72     Vec3(-150, 100, 0), // Titik 4
73     Vec3(-100, 100, 0), // Titik 5
74     Vec3(-50, 100, 0), // Titik 6
75     Vec3(0, 0, 0), // Titik 7
76     Vec3(50, -100, 0), // Titik 8
77     Vec3(100, -100, 0), // Titik 9
78     Vec3(150, -100, 0) // Titik 10
79 };
80

```

Gambar 3. 5 kode program untuk mengatur titik

- 3) Jika sudah melakukan perubahan pada program, lakukan compile dan run untuk melihat hasil output program



Gambar 3. 6 hasil output setelah perubahan titik interpolasi

BAB IV. TRANSFORMASI 2D DAN 3D

1. Pretest 1

- 1) Diketahui suatu garis P dengan titik-titik ada di koordinat A (-1, -1) dan B (1, 1). Apabila garis P dikenakan transformasi berikut, Berapa koordinat titik-titik yang baru?
- Garis P ditranslasi sejauh T (2, 2)!
 - Garis P discaling sebesar S (2, 1)!
 - Garis P dirotasi sejauh 30 derajat!

JAWAB

- 1) Diketahui garis P dengan titik-titik A (-1, -1) dan B (1, 1). Kita ingin mencari koordinat titik-titik baru setelah garis P dikenakan transformasi berikut:
- Translasi T (2, 2): Translasi menggeser semua titik pada garis dengan jarak yang sama. Dalam kasus ini, semua titik digeser 2 unit ke kanan dan 2 unit ke atas. Koordinat baru titik A: $(-1 + 2, -1 + 2) = (1, 1)$ dan Koordinat baru titik B: $(1 + 2, 1 + 2) = (3, 3)$
 - Scaling S (2, 1): Scaling meregangkan atau menyusutkan garis dengan faktor tertentu. Dalam kasus ini, garis P diregangkan 2 kali pada sumbu x dan 1 kali pada sumbu y. Titik A diregangkan dari (-1, -1) menjadi (-2, -1) dan Titik B diregangkan dari (1, 1) menjadi (2, 1).
 - Rotasi 30 derajat: Rotasi memutar semua titik pada garis dengan sudut tertentu terhadap titik asal. Dalam kasus ini, garis P diputar 30 derajat searah jarum jam. Untuk menghitung koordinat baru setelah rotasi, kita perlu menggunakan matriks rotasi. Matriks rotasi 30 derajat searah jarum jam adalah:
 - a. $(\cos(30) - \sin(30))$

$$b. (\sin(30) + \cos(30))$$

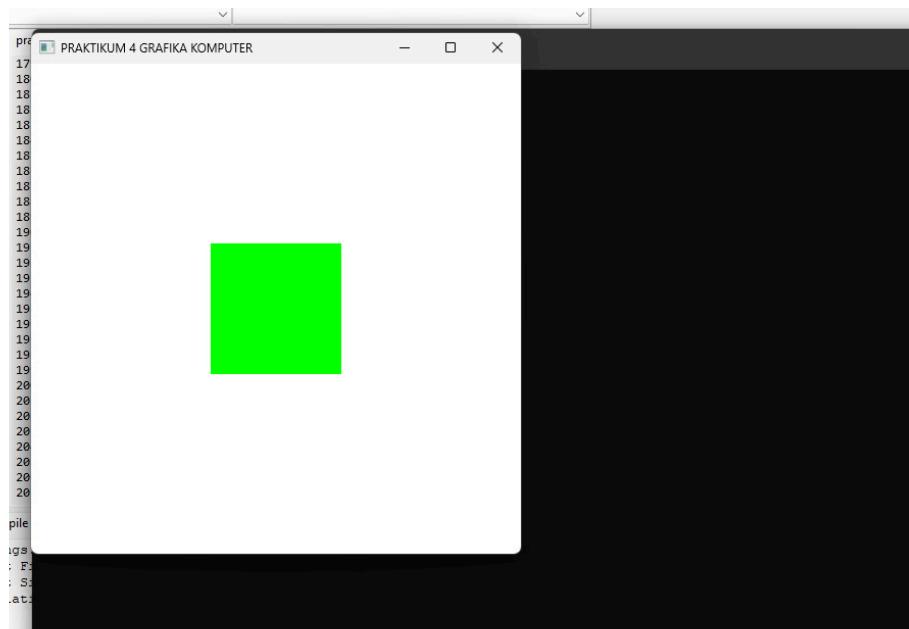
Misalkan vektor yang mewakili titik A adalah $[-1, -1]^T$ dan vektor yang mewakili titik B adalah $[1, 1]^T$. Koordinat baru titik-titik setelah rotasi dapat dihitung dengan mengalikan vektor-vektor ini dengan matriks rotasi: Koordinat baru titik

$$\begin{aligned} a. \quad & A: (-1, 1) \cdot (\cos(30) - \sin(30)) = \left(-\sqrt{\frac{3}{2}}, \frac{1}{2} \right)^T \\ b. \quad & B: (1, 1) \cdot (\sin(30) + \cos(30)) = \left(\sqrt{\frac{3}{2}}, \frac{1}{2} \right)^T \end{aligned}$$

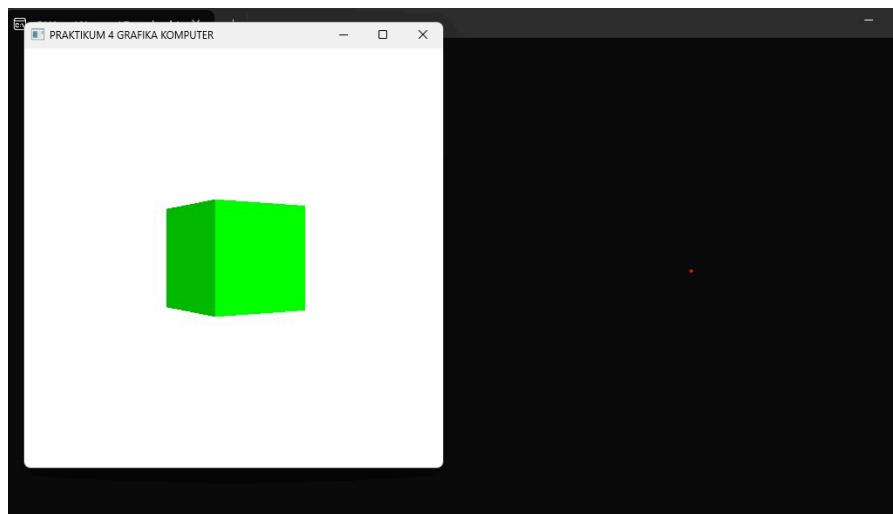
2. Langkah Praktikum 4

Langkah-langkah praktikum sesuai modul :

- 1) Buka aplikasi Dev C++ dan buat projek baru dengan console.
- 2) Copy file program txt yang diberikan oleh asisten praktikum ke dalam projek yang saja dibuat..
- 3) Isi linker pada menu parameter dengan -lopengl32, -lfreeglut, dan -lglu32
- 4) Kemudian, pada menu Directories klik pada menu Library Directories dan masukkan folder x64.
- 5) Kemudian, masih pada menu Directories klik pada menu Include Directories dan masukkan folder include.
- 6) Setelah itu lakukan compile dan run untuk melihat output dari program tersebut.
- 7)



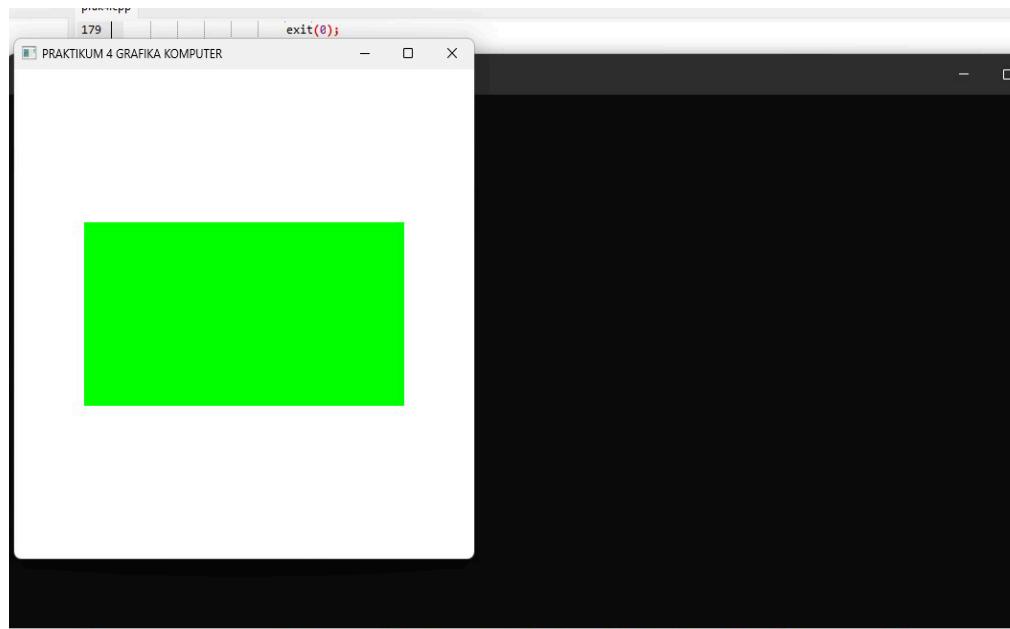
Gambar 4. 1 output langkah praktikum 4 sebelum klik tombol arah



Gambar 4. 2 output langkah praktikum 4 setelah klik tombol arah

Penjelasan singkat :

- Terdapat fungsi glutSolidCube yang digunakan untuk menggambar objek kubus pada program tersebut.
 - Terdapat fungsi glEnable(GL_DEPTH_BUFFER) yang digunakan untuk mengaktifkan depth buffer.
- 8) Gunakan tombol t, g, f, h, r dan y untuk melakukan scalling.



Gambar 4. 3 output penerapan scaling pada obyek

3. Posttest 4

Langkah-langkah :

- 1) Pada kasus ini kita ingin membuat obyek persegi berwarna cyan kemudian transformasi translasi, rotasi, dan scalling.
- 2) Kode program untuk Translasi

```
61 // Keyboard biasa (translasi dan scaling)
62 void keyboard1(unsigned char key, int x, int y)
63 {
64     float moveStep = 0.2f;
65     float scaleStep = 0.1f;
66
67     switch (key)
68     {
69         case 'w': objectPositionY += moveStep; break;
70         case 's': objectPositionY -= moveStep; break;
71         case 'a': objectPositionX -= moveStep; break;
72         case 'd': objectPositionX += moveStep; break;
73         case 'q': objectPositionZ -= moveStep; break;
74         case 'e': objectPositionZ += moveStep; break;
75     }
```

Gambar 4. 4 ubah untuk translasi

- 3) Kemudian ubah juga untuk rotasi

```

2 #include <GLUT/glut.h>
3
4 // Variabel transformasi
5 float objectAngleZ = 0.0f; // rotasi obyek di sumbu Z
6 float objectScaleX = 1.0f, objectScaleY = 1.0f, objectScaleZ = 1.0f; // skala obyek
7 float objectPositionX = 0.0f, objectPositionY = 0.0f, objectPositionZ = -5.0f; // posisi obyek
8
9 // Fungsi menggambar persegi berwarna cyan
10 void drawObject()
11 {
12     glPushMatrix();
13
14     glTranslatef(objectPositionX, objectPositionY, objectPositionZ); // Translasi
15     glScalef(objectScaleX, objectScaleY, objectScaleZ); // Skala
16     glRotatef(objectAngleZ, 0.0f, 0.0f, 1.0f); // Rotasi Z
17
18     glColor3f(0.0f, 1.0f, 1.0f); // Cyan
19

```

Gambar 4. 5 ubah untuk rotasi

4) Ubah program digunakan untuk scalling

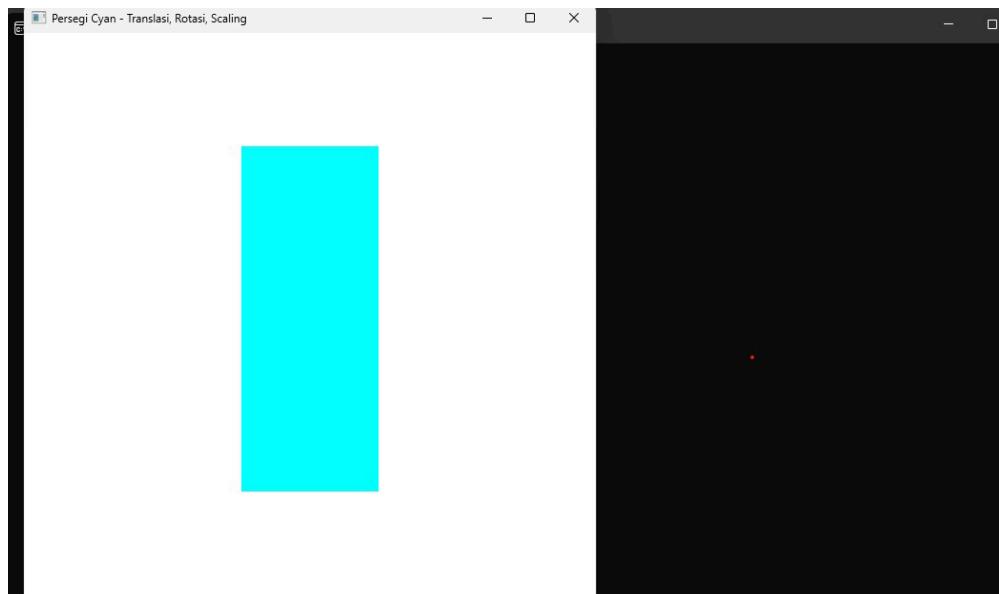
```

75     case 't': objectScaleY += scaleStep; break;
76     case 'g': objectScaleY -= scaleStep; break;
77     case 'f': objectScaleX += scaleStep; break;
78     case 'h': objectScaleX -= scaleStep; break;
79     case 'r': objectScaleZ += scaleStep; break;
80     case 'y': objectScaleZ -= scaleStep; break;
81
82

```

Gambar 4. 6 ubah untuk scallingi

5) Setelah mengubah kode prongam, lakukan compile dan run pada program untuk melihat outputnya.



Gambar 4. output posttest 4

BAB V. PROYEKSI 3D

1. Pretest 5

- 1) Jelaskan yang dimaksud dengan proyeksi orthogonal!
- 2) Jelaskan yang dimaksud dengan proyeksi perspektif!
- 3) Jelaskan perbedaan antara proyeksi orthogonal dengan perspektif!

JAWAB

- 1) Proyeksi orthogonal adalah jenis proyeksi yang digunakan untuk menggambarkan objek tiga dimensi pada bidang dua dimensi, di mana setiap garis proyeksi tegak lurus (orthogonal) terhadap bidang gambar. Dalam proyeksi orthogonal, jarak dan ukuran objek di sepanjang garis pandang tetap konstan, sehingga bentuk dan ukuran asli dari objek dapat tetap terjaga tanpa distorsi perspektif.

Ciri-ciri proyeksi orthogonal :

- Garis-garis proyeksi sejajar satu sama lain.
- Tidak ada efek perspektif; objek yang jauh tidak tampak lebih kecil dari objek yang dekat.
- Digunakan dalam gambar teknik dan CAD (Computer-Aided Design) untuk representasi yang akurat dari objek.

- 2) Proyeksi perspektif adalah jenis proyeksi yang digunakan untuk menggambarkan objek tiga dimensi pada bidang dua dimensi, di mana garis-garis proyeksi menyatu pada titik vanishing point di cakrawala. Dalam proyeksi perspektif, objek yang jauh dari pengamat tampak lebih kecil dibandingkan dengan objek yang dekat, menghasilkan efek tiga dimensi yang realistik.

Ciri-ciri proyeksi perspektif :

- Garis-garis proyeksi bertemu pada titik vanishing point.

- Ukuran objek bervariasi tergantung pada jaraknya dari titik pandang; objek yang jauh tampak lebih kecil.
- Digunakan dalam seni dan desain untuk menciptakan representasi visual yang realistik.

3) Perbedaan Antara Proyeksi Orthogonal dengan Perspektif

a. Proyeksi orthogonal

- Garis Proyeksi: Sejajar satu sama lain.
- Ukuran dan Bentuk: Ukuran objek tetap konstan; tidak ada distorsi.
- Aplikasi: Gambar teknik, CAD, blueprint, di mana akurasi dan proporsi penting.
- Tampilan: Objek terlihat datar tanpa efek tiga dimensi.
- Jarak dan Sudut: Tidak terpengaruh oleh jarak; sudut dan panjang tetap sama.

b. Proyeksi perspektif

- Garis Proyeksi: Menyatu pada satu atau lebih titik vanishing point.
- Ukuran dan Bentuk: Ukuran objek berubah tergantung pada jarak dari pengamat; objek yang jauh tampak lebih kecil.
- Aplikasi: Seni, desain arsitektur, game, dan simulasi 3D, di mana efek visual realistik diinginkan.
- Tampilan: Objek terlihat memiliki kedalaman dan efek tiga dimensi.
- Jarak dan Sudut: Terpengaruh oleh jarak; objek yang lebih jauh tampak lebih kecil dan sudutnya berubah sesuai dengan perspektif.

2. Langkah praktikum 5

Langkah-langkah praktikum sesuai modul :

- 1) Buka aplikasi Dev C++ dan buat projek baru dengan console.

- 2) Copy file program txt yang diberikan oleh asisten praktikum ke dalam projek yang saja dibuat..
- 3) Isi linker pada menu parameter dengan -lopengl32, -lfreeglut, dan -lglu32
- 4) Kemudian, pada menu Directories klik pada menu Library Directories dan masukkan folder x64.
- 5) Kemudian, masih pada menu Directories klik pada menu Include Directories dan masukkan folder include.
- 6) Setelah itu lakukan compile dan run untuk melihat output dari program tersebut.

```

pertemuan5/
1 #define
2 #include
3
4 #define
5 #define
6
7 // posisi
8 // posisi
9 //float
10 //posisi
11 float
12
13 // inisiasi
14 float
15 float
16 float
17 float
18
19 float
20 float
21 float
22
23 // fungsi
24 void draw()
25 {
26
27
28 // saat diwarnai, ditransformasi dan sebagainya
29 glPushMatrix();
30
31 // operasi transformasi rotasi obyek ke arah kanan-kiri
32 glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);
33
34 glPopMatrix();

```

Gambar 5. 1 hasil output proyeksi kubus

Penjelasan singkat :

```

88 }
89
90 // fungsi ini digunakan bila layar akan diresize (default)
91 void reshape(int w, int h)
92 {
93     glViewport(0, 0, (GLsizei)w, (GLsizei)h);
94     glMatrixMode(GL_PROJECTION);
95     glLoadIdentity();
96     glOrtho((GLfloat)-w/2, (GLfloat)w/2, (GLfloat)-h/2, (GLfloat)h/2, 1.0, 100.0);
97     glMatrixMode(GL_MODELVIEW);
98 }
99

```

Gambar 5. 2 fungsi untuk mengatur proyeksi orthogonal

- Pada fungsi ‘**reshape()**’ digunakan untuk mengatur proyeksi orthogonal ketika ukuran jendela berubah. Terdapat fungsi ‘**glOrtho()**’ yang digunakan untuk mendefinisikan volume pandang proyeksi

orthogonal sesuai dengan ukuran baru jendela

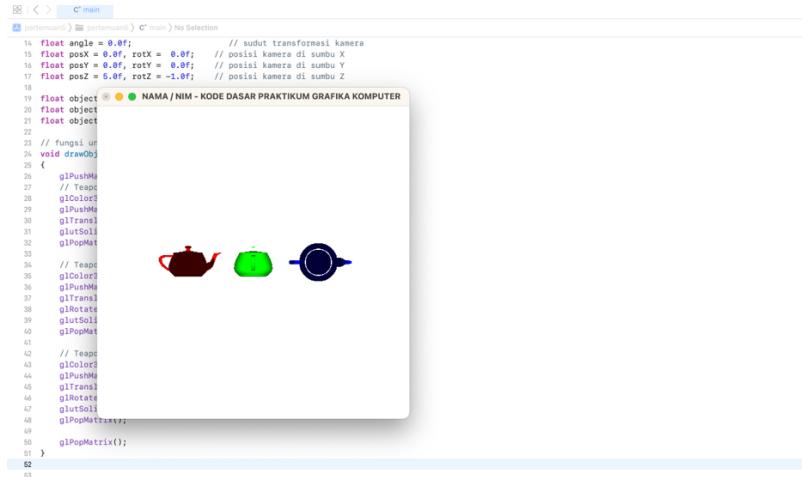
3. Posttest 5

- 1) Pada kasus ini kita akan mengubah proyeksi 3D dalam bentuk kubus menjadi bentuk teapot. Jumlah teapot yang akan dibuat adalah 3.
- 2) Ubah bagian drawObject seperti pada Gambar 5.6 dibawah ini :

```
23 // fungsi untuk menggambar obyek kubus
24 void drawObject()
25 {
26     glPushMatrix();
27     // Teapot merah - tampak depan (default)
28     glColor3f(1.0f, 0.0f, 0.0f); // merah
29     glPushMatrix();
30     glTranslatef(-100.0f, 0.0f, 0.0f); // geser ke kiri
31     glutSolidTeapot(30.0f);
32     glPopMatrix();
33
34     // Teapot hijau - tampak samping (kanan)
35     glColor3f(0.0f, 1.0f, 0.0f); // hijau
36     glPushMatrix();
37     glTranslatef(0.0f, 0.0f, -100.0f); // geser ke dalam
38     glRotatef(90, 0.0f, 1.0f, 0.0f); // rotasi ke arah samping
39     glutSolidTeapot(30.0f);
40     glPopMatrix();
41
42     // Teapot biru - tampak atas
43     glColor3f(0.0f, 0.0f, 1.0f); // biru
44     glPushMatrix();
45     glTranslatef(100.0f, 0.0f, 0.0f); // geser ke kanan
46     glRotatef(-90, 1.0f, 0.0f, 0.0f); // rotasi ke arah atas
47     glutSolidTeapot(30.0f);
48     glPopMatrix();
49
50     glPopMatrix();
51 }
52 --
```

Gambar 5. 3 ubah proyeksi kubus menjadi 3 teapot

- 3) Setelah program diubah maka lakukan compile dan run untuk melihat hasil output dari program.



```
14 float angle = 0.0f; // sudut transformasi kamera
15 float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X
16 float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y
17 float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z
18
19 float object; // NAMA/NIM - KODE DASAR PRAKTIKUM GRAFIKA KOMPUTER
20
21 void drawObj()
22 {
23     // Fungsi utama
24     void drawObj()
25     {
26         glPushMatrix();
27         // Teapot
28         glColor3f(1.0f, 0.0f, 0.0f);
29         glPushMatrix();
30         glTranslatef(-100.0f, 0.0f, 0.0f);
31         glutSolidTeapot(30.0f);
32         glPopMatrix();
33
34         // Teapot
35         glColor3f(0.0f, 1.0f, 0.0f);
36         glPushMatrix();
37         glTranslatef(0.0f, 0.0f, -100.0f);
38         glRotatef(90, 0.0f, 1.0f, 0.0f);
39         glutSolidTeapot(30.0f);
40         glPopMatrix();
41
42         // Teapot
43         glColor3f(0.0f, 0.0f, 1.0f);
44         glPushMatrix();
45         glTranslatef(100.0f, 0.0f, 0.0f);
46         glRotatef(-90, 1.0f, 0.0f, 0.0f);
47         glutSolidTeapot(30.0f);
48         glPopMatrix();
49     }
50 }
51
52 --
```

Gambar 5. 4 hasil output posttest 5 proyeksi 3D (teapot)

BAB VI. REPRESENTASI OBYEK 3D

1. Pretest 6

- 1) Sebutkan metode representasi obyek 3D yang anda ketahui!
- 2) Jelaskan setiap metode representasi obyek 3D yang anda sebutkan di soal nomor 1!

JAWAB

- 1) Beberapa metode representasi objek 3D yang umum digunakan meliputi :
 - Wireframe (Kerangka Kawat)
 - Surface Rendering (Penggambaran Permukaan)
 - Solid Modeling (Modeling Padat)
 - Point Clouds (Awan Titik)
 - Polygonal Mesh (Jaringan Poligonal)
- 2) Penjelasan mengenai setiap metode representasi objek 3D yang ada di nomor 1:
 - Wireframe (Kerangka Kawat)

Wireframe adalah representasi visual objek 3D yang terbuat dari garis-garis dan kurva-kurva yang menghubungkan titik-titik (vertices) dan membentuk batas-batas objek. Wireframe digunakan dalam desain 3D untuk memvisualisasikan struktur dasar objek dan memudahkan pengembangan dan manipulasi solid dan permukaan solid. Wireframe mudah diproses dan digunakan dalam aplikasi yang memerlukan frame rate tinggi, seperti dalam video game atau sistem real-time yang memodelkan fenomena luar ruang.

- Surface Rendering (Penggambaran Permukaan)

Surface Rendering adalah metode penggambaran yang mempresentasikan

objek 3D dengan menggunakan permukaan yang terbuat dari polygon-polygon. Permukaan ini dapat diproyeksikan ke dalam ruang 2D untuk menghasilkan gambar yang lebih detail dan realistik. Surface Rendering digunakan dalam aplikasi yang memerlukan detail visual yang lebih tinggi, seperti dalam film animasi atau video game yang memerlukan rendering yang lebih kompleks.

- Solid Modeling (Modeling Padat)

Solid Modeling adalah metode pengembangan objek 3D yang mempresentasikan objek sebagai solid yang terbuat dari polygon-polygon dan volume yang terdefinisi. Solid Modeling digunakan dalam aplikasi yang memerlukan detail geometris yang lebih tinggi, seperti dalam desain produk industri atau arsitektur. Solid Modeling memungkinkan pengembangan objek yang lebih realistik dan detail, serta memudahkan analisis geometris dan kinematika.

- Point Clouds (Awan Titik)

Point Clouds adalah representasi objek 3D yang terbuat dari titik-titik 3D yang tersebar di ruang. Point Clouds digunakan dalam aplikasi seperti scanning 3D, surveying, dan visualisasi data. Point Clouds memungkinkan analisis geometris dan kinematika yang lebih detail dan memudahkan pengembangan model 3D yang lebih realistik.

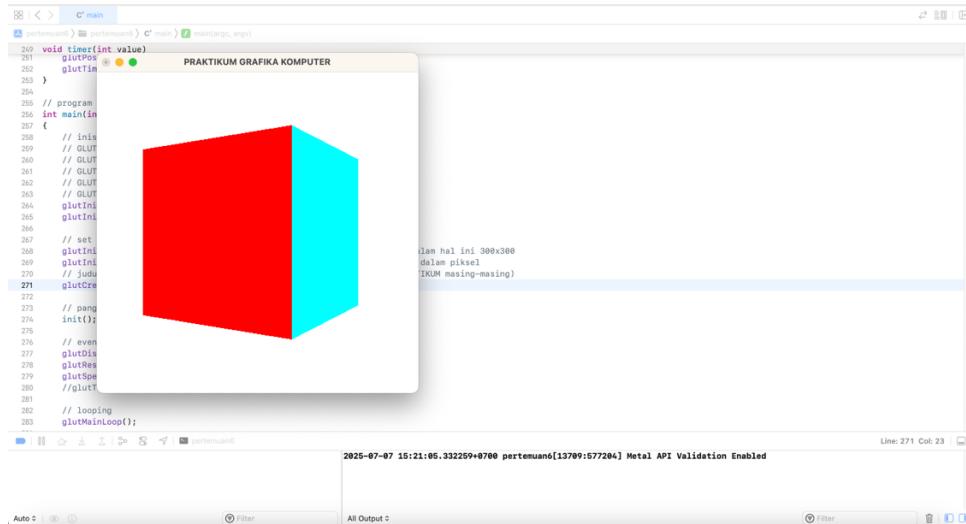
- Polygonal Mesh (Jaringan Poligonal)

Polygonal Mesh adalah representasi objek 3D yang terbuat dari jaringan polygon-polygon yang menghubungkan titik-titik (vertices). Polygonal Mesh digunakan dalam aplikasi seperti desain produk industri, arsitektur, dan animasi. Polygonal Mesh memungkinkan pengembangan objek yang lebih realistik dan detail, serta memudahkan analisis geometris dan kinematika.

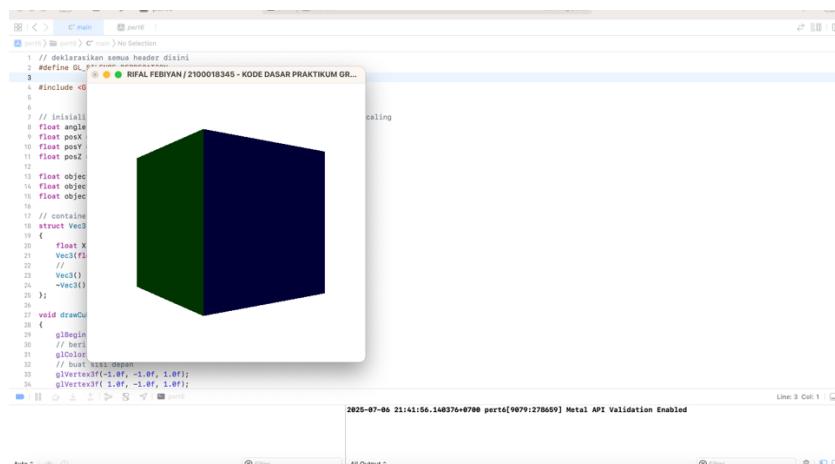
2. Langkah praktikum 6

Langkah-langkah praktikum :

- 1) Buka aplikasi Dev C++ dan buat projek baru dengan console.
- 2) Copy file program txt yang diberikan oleh asisten praktikum ke dalam projek yang saja dibuat..
- 3) Isi linker pada menu parameter dengan -lopengl32, -lfreeglut, dan -lglu32
- 4) Kemudian, pada menu Directories klik pada menu Library Directories dan masukkan folder x64.
- 5) Kemudian, masih pada menu Directories klik pada menu Include Directories dan masukkan folder include.
- 6) Setelah itu lakukan compile dan run untuk melihat output dari program tersebut.



Gambar 6. 1 hasil output representasi kubus sebelum klik tombol panah



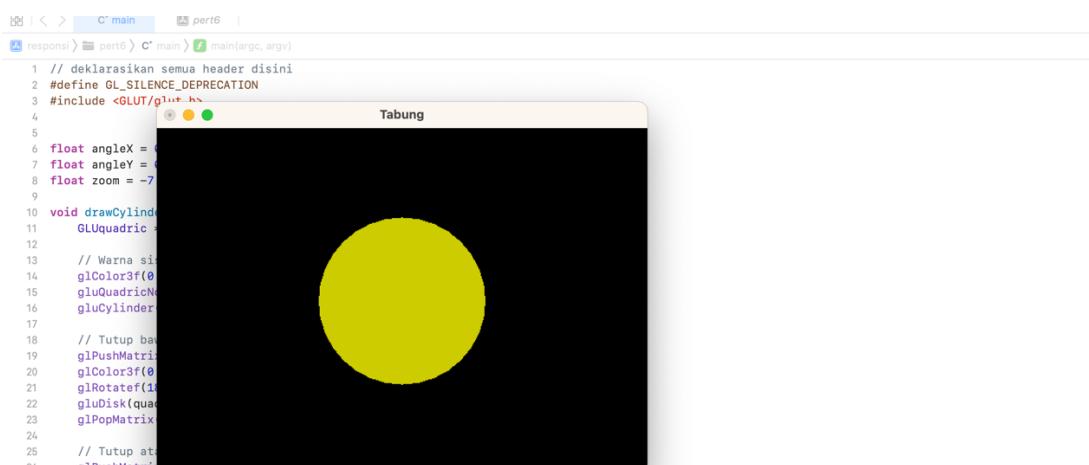
Gambar 6. 2 hasil output representasi kubus setelah klik tombol panah

- 7) Ubah fungsi drawObject() untuk menampilkan tabung Ganti fungsi drawCube menjadi fungsi drawCylinder

```
9
10 void drawCylinder() {
11     GLUquadric *quad = gluNewQuadric();
12
13     // Warna sisi samping
14     glColor3f(0.6f, 0.1f, 0.1f);
15     gluQuadricNormals(quad, GLU_SMOOTH);
16     gluCylinder(quad, 1.0, 1.0, 2.0, 32, 32);
17
18     // Tutup bawah
19     glPushMatrix();
20     glColor3f(0.8f, 0.8f, 0.2f);
21     glRotatef(180.0f, 1.0f, 0.0f, 0.0f);
22     gluDisk(quad, 0.0, 1.0, 32, 1);
23     glPopMatrix();
24
25     // Tutup atas
26     glPushMatrix();
27     glColor3f(0.8f, 0.8f, 0.2f);
28     glTranslatef(0.0f, 0.0f, 2.0f);
29     gluDisk(quad, 0.0, 1.0, 32, 1);
30     glPopMatrix();
31
32     gluDeleteQuadric(quad);
33 }
34
35
```

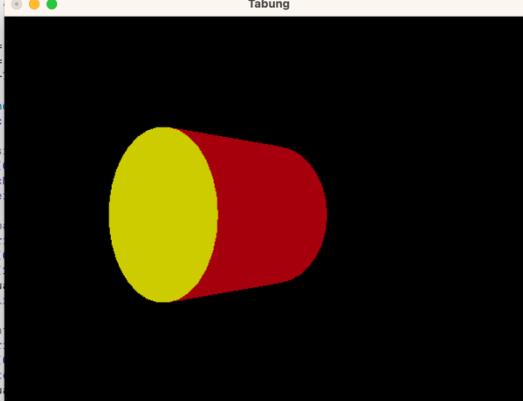
Gambar 6. 3 merubah fungsi cube menjadi cylinder

- 8) Kemudian tambahkan fungsi drawCylinder pada fungsi drawObject.
9) Setelah melakukan perubahan pada program, lakukan compile dan run pada program untuk melihat hasil outputnya.



```
1 // deklarasiakan semua header disini
2 #define GL_SILENCE_DEPRECATION
3 #include <GLUT/glut.h>
4
5 float angleX = 0;
6 float angleY = 0;
7 float zoom = -7;
8
9
10 void drawCylinder()
11     GLUquadric *
12
13     // Warna sisi
14     glColor3f(0.6f, 0.1f, 0.1f);
15     gluQuadricNormals(quad, GLU_SMOOTH);
16     gluCylinder(quad, 1.0, 1.0, 2.0, 32, 32);
17
18     // Tutup bawah
19     glPushMatrix();
20     glColor3f(0.8f, 0.8f, 0.2f);
21     glRotatef(180.0f, 1.0f, 0.0f, 0.0f);
22     gluDisk(quad, 0.0, 1.0, 32, 1);
23     glPopMatrix();
24
25     // Tutup atas
26     glPushMatrix();
27     glColor3f(0.8f, 0.8f, 0.2f);
28     glTranslatef(0.0f, 0.0f, 2.0f);
29     gluDisk(quad, 0.0, 1.0, 32, 1);
30     glPopMatrix();
31
32     gluDeleteQuadric(quad);
33 }
```

Gambar 6. 4 hasil output representasi tabung sebelum klik tombol panah

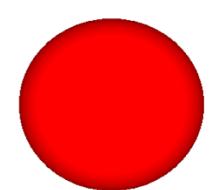


```
C:\ main | perl6 | C:\ main | perl6 | main(argc, argv)

1 // deklarasikan semua header disini
2 #define GL_SILENCE_DEPRECATION
3 #include <GLUT.h>
4
5
6 float angleX =
7 float angleY =
8 float zoom = -1.0;
9
10 void drawCylinder() {
11     gluQuadricOrientation(GLU_OUTSIDE);
12
13     // Warna sisi
14     glColor3f(1.0, 0.0, 0.0);
15     gluQuadricDrawStyle(GL_QUADS);
16     gluCylinder(quadr, 1.0, 0.5, 2.0, 10, 10);
17
18     // Tutup bawah
19     glPushMatrix();
20     glColor3f(0.0, 1.0, 0.0);
21     glRotatef(90.0, 1.0, 0.0, 0.0);
22     gluDisk(quadr, 0.0, 1.0, 10, 10);
23     glPopMatrix();
24
25     // Tutup atas
26     glPushMatrix();
27     glColor3f(1.0, 1.0, 0.0);
28     glTranslatef(0.0, 0.5, 0.0);
29     gluDisk(quadr, 0.0, 1.0, 10, 10);
30     glPopMatrix();
31
32     gluDeleteQuadric(quadr);
33 }
34
35
36 void drawSphere() {
37     glColor3f(0.2, 0.8, 0.3); // hijau muda
38     glutSolidSphere(1.0, 30, 30);
39 }
40
41 void drawObjects() {
```

Gambar 6. 5 hasil output representasi tabung setelah diklik tombol panah

10) Ubah fungsi drawSphere() untuk menampilkan bola



```
void drawSphere(float radius, int slices, int stacks) {
    yesdrawSphere(radius, slices, stacks);
}
NAMA/NIM - PRAKTIKUM GRAFIKA KOMPUTER

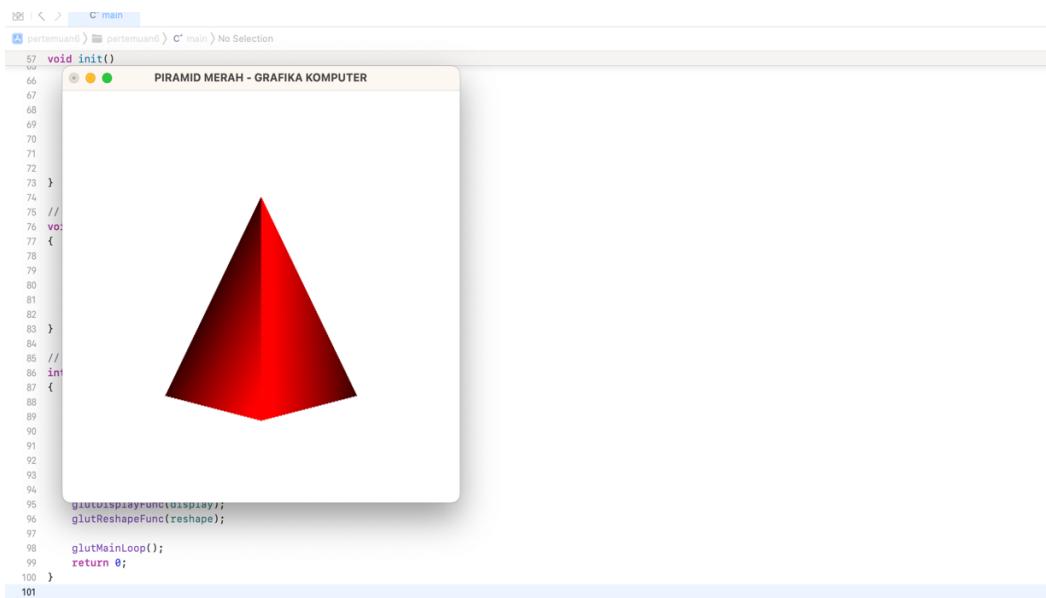
63 void drawSphere(float radius, int slices, int stacks) {
64     yesdrawSphere(radius, slices, stacks);
65 }
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97     drawObject(); // hanya gambar bola
98
99     glutSwapBuffers();
100 }

101 // Fungsi inisialisasi
102 void init() {
103     glClearColor(1, 1, 1, 0); // putih
```

Gambar 6. 6 hasil output representasi bola

3. Posttest 6

- 1) Buatlah obyek piramid berwarna merah menggunakan gluCylinder.
- 2) Pada kode program bagian drawObject ubah drawPyramid
- 3) Setelah dilakukan pengubahan maka compile dan run program untuk melihat hasil outputnya.



A screenshot of a C++ IDE interface. The code editor shows a file named 'main.c' with the following content:

```
57 void init()
58 {
59     // ...
60     void
61     {
62         // ...
63     }
64     // ...
65     int
66     {
67         // ...
68     }
69
70     glutDisplayFunc(display);
71     glutReshapeFunc(reshape);
72
73     glutMainLoop();
74     return 0;
75 }
```

The window title bar reads "PIRAMID MERAH - GRAFIKA KOMPUTER". Inside the window, a 3D rendering of a red pyramid is displayed, representing the rendered output of the program.

Gambar 6. 7 hasil output representasi pyramid

BAB VII. KURVA SPLINE

1. Pretest 7

1) Jelaskan perbedaan antara kurva spline Cubic, Catmull-Rom, Hermit, dan Bezier!

JAWAB

1) Perbedaan Antara Kurva Spline Cubic, Catmull-Rom, Hermite, dan Bezier

a. Kurva Spline Cubic

- Deskripsi: Kurva spline cubic adalah kurva interpolasi yang terbentuk dari beberapa segmen cubic polynomial (polinomial derajat tiga) yang terhubung secara mulus di titik kontrolnya.
- Kontrol Poin: Menggunakan titik kontrol (control points) untuk menentukan jalur kurva yang dihasilkan.
- Interpolasi: Dapat digunakan untuk interpolasi atau aproksimasi data titik-titik yang diberikan.
- Kontinuitas: Terdapat dua jenis kontinuitas utama: kontinuitas posisi (C_0) dan kontinuitas turunan pertama (C_1).

b. Kurva Catmull-Rom

- Deskripsi: Kurva Catmull-Rom adalah jenis kurva spline cubic yang dikenal karena kemampuannya untuk melewati setiap titik kontrol yang ditentukan.
- Kontrol Poin: Membutuhkan minimal empat titik kontrol untuk menentukan sebuah kurva.
- Interpolasi: Melalui setiap titik kontrol secara tepat, yang membuatnya cocok untuk animasi dan pergerakan yang mulus dalam grafika komputer.
- Kontinuitas: Menjamin kontinuitas posisi (C_0) dan kontinuitas turunan pertama (C_1) di antara segmen kurva yang bersebelahan.

c. Kurva Hermite

- Deskripsi: Kurva Hermite menggunakan interpolasi Hermite untuk menentukan jalur antara titik-titik kontrolnya, yang mencakup posisi dan vektor turunan (gradient) pada titik-titik tersebut.
- Kontrol Poin: Memerlukan setiap titik kontrol dilengkapi dengan vektor turunan yang menentukan kemiringan kurva di titik tersebut.
- Interpolasi: Bisa digunakan untuk mengontrol kecepatan perubahan di sepanjang kurva, membuatnya ideal untuk animasi yang memerlukan gerakan yang halus dan terkontrol.
- Kontinuitas: Bergantung pada pengaturan vektor turunan, dapat menyediakan kontinuitas posisi (C_0) dan kontinuitas turunan pertama (C_1).

d. Kurva Bezier

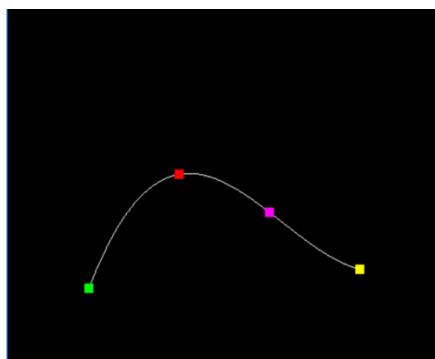
- Deskripsi: Kurva Bezier menggunakan polinomial polinomial berorde rendah untuk menggambarkan kurva yang lebih kompleks, dengan mempertimbangkan titik kontrol dan bobotnya.
- Kontrol Poin: Didefinisikan oleh titik kontrol yang dapat menarik atau mengarahkan kurva tanpa melewati mereka.
- Interpolasi: Tergantung pada jumlah titik kontrol yang digunakan; kurva Bezier bisa mendekati kurva spline cubic tetapi tidak secara intrinsik melewati setiap titik kontrol.
- Kontinuitas: Tergantung pada pengaturan titik kontrol, kurva Bezier dapat memberikan berbagai tingkat kontinuitas, dari C_0 hingga C_2 , tergantung pada jumlah dan posisi titik kontrol.

2. Langkah praktikum 7

Langkah-langkah praktikum sesuai modul :

- 1) Buka aplikasi Dev C++ dan buat projek baru dengan console.

- 2) Copy file program txt yang diberikan oleh asisten praktikum ke dalam projek yang saja dibuat..
- 3) Isi linker pada menu parameter dengan -lopengl32, -lfreeglut, dan -lglu32
- 4) Kemudian, pada menu Directories klik pada menu Library Directories dan masukkan folder x64.
- 5) Kemudian, masih pada menu Directories klik pada menu Include Directories dan masukkan folder include.
- 6) Setelah itu lakukan compile dan run untuk melihat output dari program tersebut.



Gambar 7. 1 hasil output langkah praktikum 7 kurva cubic

Penjelasan :

```

...
417 // membuat 4 titik kontrol kurva
418 Vec3 point1 = Vec3(-150.0f, -70.0f, 0.0f);
419 Vec3 point2 = Vec3(-50.0f, 50.0f, 0.0f);
420 Vec3 point3 = Vec3(50.0f, 10.0f, 0.0f);
421 Vec3 point4 = Vec3(150.0f, -50.0f, 0.0f);
422
423 // tandai setiap titik kontrol kurva dengan warna
424 markPoint(point1, Vec3(0.0f, 1.0f, 0.0f), 5.0f);
425 markPoint(point2, Vec3(1.0f, 0.0f, 0.0f), 5.0f);
426 markPoint(point3, Vec3(1.0f, 0.0f, 1.0f), 5.0f);
427 markPoint(point4, Vec3(1.0f, 1.0f, 0.0f), 5.0f);

```

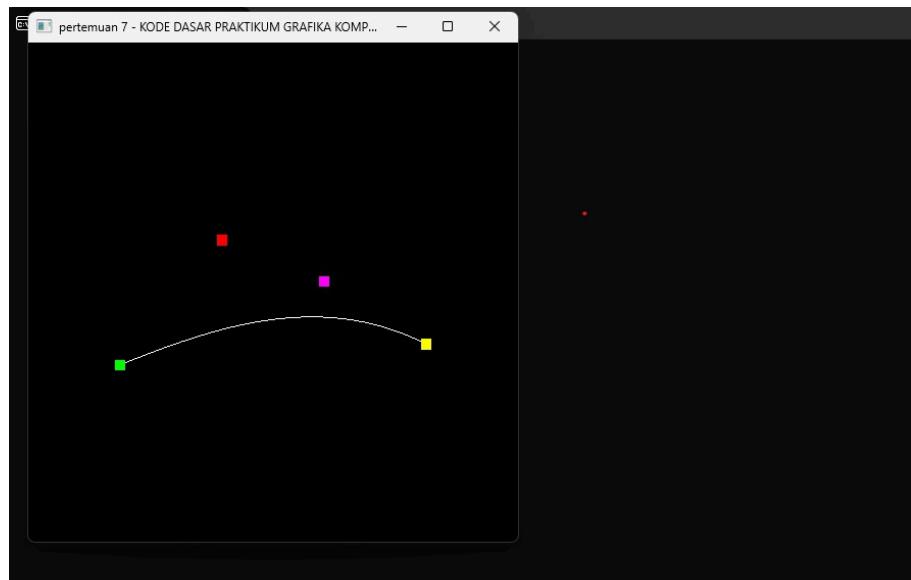
Gambar 7. 2 kode program yang mengontrol titik pada kurva

- Pada baris 417 sampai 421 digunakan untuk membuat 4 titik dan koordinatnya.
- Pada baris ke 423 sampai 427 digunakan untuk mengontrol warna dan besar titik tersebut.

- 7) Ubah kode di fungsi drawObject() untuk menggambar kurva spline Catmull-Rom

```
437 // membuat kurva spline catmullrom dari titik kontrol diatas  
438 drawSplineCatmullRom(point1, point2, point3, point4, 30);  
439  
440 glEnd();  
441 glPopMatrix();
```

Gambar 7. 3 kode program membuat kurva spline Catmull



Gambar 7. 4 hasil output kurva spline Catmull-Rom

- 8) Ubah kode di fungsi drawObject() untuk menggambar kurva spline Bezier

```
434 // membuat kurva spline cubic dari titik kontrol diatas  
435 drawSplineCubic(point1, point2, point3, point4, 30);  
436  
437 // membuat kurva spline bezier dari titik kontrol diatas  
438 drawSplineBezier(point1, point2, point3, point4, 30);  
439  
440 glEnd();  
441 glPopMatrix();  
442 glPopMatrix();  
443  
444 }  
445
```

Gambar 7. 5 hasil output kurva spline Bezier

3. Posttest 7

Langkah-langkah :

- 1) Buatlah kurva spline cubic, bezier dan catmull-rom dengan posisi titik kontrol:

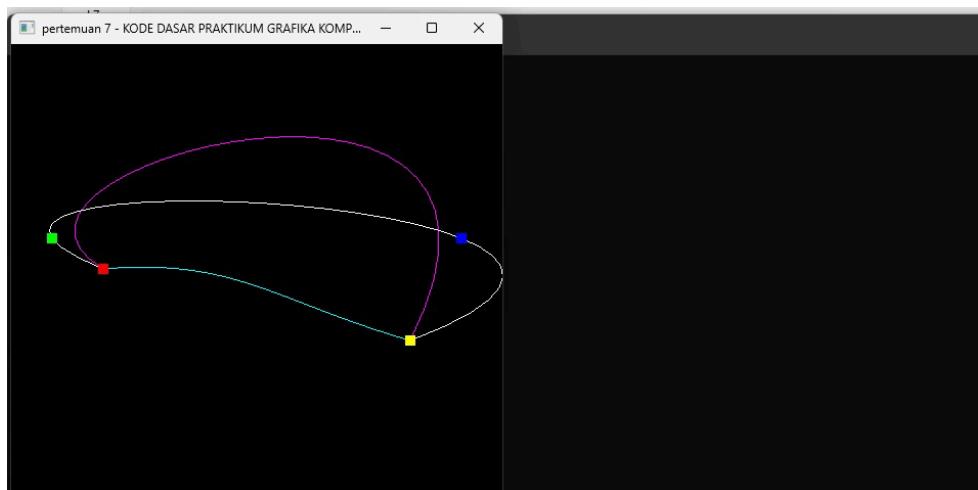
A (-150.0f, 20.0f, 0.0f)

B (-200.0f, 50.0f, 0.0f)

C (200.0f, 50.0f, 0.0f)

D (150.0f, -50.0f, 0.0f)

2) Hasil 3 kurva spline



Gambar 7. 6 merubah koordinat, warna, dan ukuran titik kurva

Penjelasan : spline dengan warna putih yaitu spline cubic, spline dengan warna magenta yaitu spline bezier terakhir spline dengan warna cyan yaitu catmull-rom.

BAB VII. TEKNIK REPRESENTASI PERMUKAAN

1. Pretest 8

- 1) Sebutkan teknik representasi permukaan yang anda ketahui!
- 2) Jelaskan setiap teknik representasi permukaan yang anda sebutkan di soal nomor 1!

JAWAB

- 1) Teknik representasi permukaan melibatkan beberapa cara untuk menggambarkan dan menampilkan permukaan 3D. Berikut adalah beberapa teknik yang umum

digunakan:

- Polygon Surfaces / Polyhedra
- Kurva
- Constructive Solid Geometry (CSG)
- Sweep Representation
- Boundary Representation
- Spatial Partitioning Representation
- Digital Elevation Model (DEM)
- Triangulated Irregular Network (TIN)
- Contour (Isoline) Lines

2) Penjelasan mengenai setiap teknik representasi permukaan yang disebutkan di soal nomor 1 :

- Polygon Surfaces / Polyhedra :
 - Merepresentasikan objek 3D dengan gabungan polygon tertutup yang membentuk objek baru. Polygon mudah diproses dan biasanya berbentuk segitiga.
 - Spesifikasi polygon meliputi tabel data polygon, tabel geometri, dan tabel atribut seperti transparansi, warna, dan tekstur[1].
- Kurva :
 - Spline : Menggambar bentuk kurva kompleks dengan memasukkan rangkaian titik kendali. Kurva yang melewati tiap titik kendali disebut interpolasi kurva spline, sedangkan kurva yang melewati dekat namun tidak melewati titik kendali disebut pendekatan kurva spline.
 - Bezier : Kurva yang menggunakan titik kendali untuk menggambar bentuk kurva yang kompleks.
- Constructive Solid Geometry (CSG) :

Merepresentasikan objek 3D dengan menggabungkan beberapa objek yang lebih sederhana menggunakan operasi set (union, intersection, dan difference).

- Sweep Representation :

Merepresentasikan objek 3D dengan menggambarkan objek sebagai hasil dari gerakan kurva atau garis melalui ruang 3D.

- Boundary Representation : Merepresentasikan objek 3D dengan menggambarkan batas-batas objek sebagai garis atau kurva yang membentuk objek.
- Spatial Partitioning Representation :

Merepresentasikan objek 3D dengan menggambarkan objek sebagai kumpulan segmen ruang yang terbagi secara spati.

- Digital Elevation Model (DEM) :

Merepresentasikan permukaan dengan menggunakan array sampel dari elevasi yang secara regular ditentukan intervalnya pada arah x dan y. Keuntungan DEM meliputi model konsep sederhana dan mudah berelasi dengan data raster lain.

- Triangulated Irregular Network (TIN) :

Merepresentasikan permukaan dengan menggunakan segitiga irregular yang memiliki elevasi pada tiga titik. Keuntungan TIN meliputi efisiensi dan dapat menangkap fitur lereng secara signifikan.

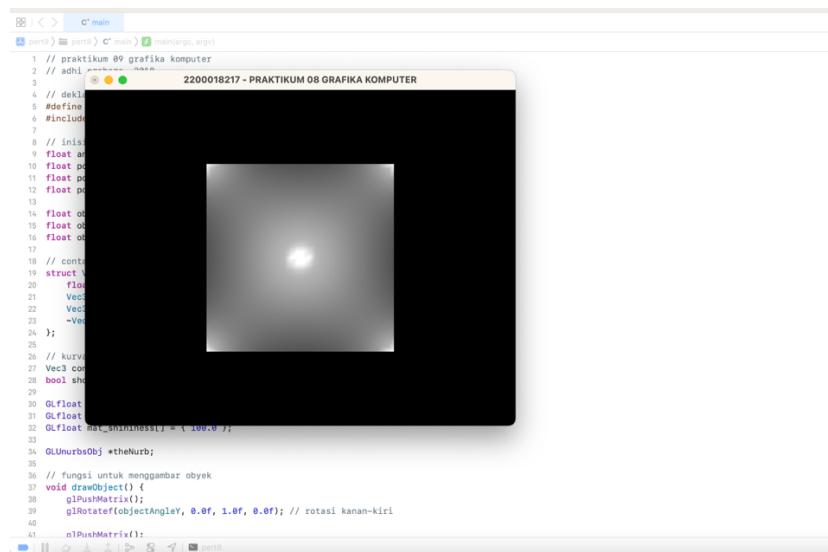
- Contour (Isoline) Lines :

Merepresentasikan permukaan dengan menggunakan garis kontur yang membentuk isoline dari elevasi konstan pada interval tertentu. Keuntungan contour lines meliputi familiaritas pada banyak orang dan mudah membentuk gambar permukaan.

2. Langkah praktikum 8

Langkah-langkah praktikum :

- 1) Buka aplikasi Dev C++ dan buat projek baru dengan console.
- 2) Copy file program txt yang diberikan oleh asisten praktikum ke dalam projek yang saja dibuat..
- 3) Isi linker pada menu parameter dengan -lopengl32, -lfreeglut, dan -lglu32
- 4) Kemudian, pada menu Directories klik pada menu Library Directories dan masukkan folder x64.
- 5) Kemudian, masih pada menu Directories klik pada menu Include Directories dan masukkan folder include.
- 6) Setelah itu lakukan compile dan run untuk melihat output dari program tersebut.



The screenshot shows the Dev C++ IDE interface. The code editor window displays a C++ program named 'main'. The code includes definitions for a struct 'Nurb' containing vectors and floats, and a function 'drawObject' which uses OpenGL functions like glPushMatrix, glPopMatrix, and glRotatef to render an object. The rendered output is a grayscale surface plot of a hyperbolic paraboloid shape (a saddle surface) displayed in a separate window titled 'main'. The title bar of the Dev C++ window also shows 'main'.

```

1 // praktikum 09 grafika komputer
2 // adhi_
3 // date : 2200018217 - PRAKTIKUM 08 GRAFIKA KOMPUTER
4 // definisi
5 #define _CRT_SECURE_NO_WARNINGS
6
7 // inisiasi
8 float angleY = 0.0f;
9 float posx = 0.0f;
10 float posy = 0.0f;
11 float posz = 0.0f;
12 float radius = 10.0f;
13
14 float obj[10][10];
15 float obj2[10][10];
16 float obj3[10][10];
17
18 // konten
19 struct Nurb {
20     float a;
21     Vec3 b;
22     Vec3 c;
23     ~Vec3();
24 };
25
26 // kurva
27 Vec3 crossProduct(Vec3 v1, Vec3 v2);
28 bool shareSameNormal(Nurb *n1, Nurb *n2);
29
30 GLfloat mat_shininess[] = { 100.0 };
31 GLfloat mat_ambient[] = { 0.0 };
32 GLfloat mat_shadeModel[] = { 100.0 };
33
34 GLUnurbsObj *theNurb;
35
36 // fungsi untuk menggambar objek
37 void drawObject() {
38     glPushMatrix();
39     glPushMatrix();
40     glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f); // rotasi kanan-kiri
41
42     glPopMatrix();
43 }

```

Gambar 8. 1 output langkah praktikum 7 representasi permukaan

```

1 // praktikum 09 grafika komputer
2 // adhi
3 // deklarasi
4 // inisiasi
5 #include <GL/gl.h>
6 #include <GL/glu.h>
7 #include <GL/unrbs.h>
8 // kurva
9 float angleY = 0.0f;
10 float posx = 0.0f;
11 float posy = 0.0f;
12 float posz = 0.0f;
13 float objAngleY = 0.0f;
14 float objAngleX = 0.0f;
15 float objAngleZ = 0.0f;
16 float objShininess = 100.0f;
17
18 // konten
19 struct Vec3 {
20     float x;
21     Vec3();
22     Vec3(float x, float y, float z);
23     ~Vec3();
24 };
25
26 // kurva
27 Vec3 curve(GLfloat t) {
28     bool show = true;
29     GLfloat u;
30     GLfloat v;
31     GLfloat w;
32     GLfloat mat_shininess[] = { 100.0 };
33
34     GLUnurbsObj *theNurb;
35
36     // fungsi untuk menggambar obyek
37     void drawObject() {
38         glPushMatrix();
39         glRotatef(objAngleY, 0.0f, 1.0f, 0.0f); // rotasi kanan-kiri
40
41         glPushMatrix();

```

Gambar 8. 2 output langkah pratikum 7 setelah klik tombol arah

3. Posttest 8

Langkah-langkah :

- 1) Pada kasus ini kita akan merubah merubah control point, compile dan run program untuk melihat hasil outputnya
- 2) Ubah perulangan dan control point dari kode program
- 3) Setelah mengatur control point, lakukan compile dan run untuk melihat output dari program

```

1 #define _CRT_SECURE_NO_WARNINGS
2 #include <GL/gl.h>
3 #include <GL/glu.h>
4 #include <GL/unrbs.h>
5 #include <math.h>
6 #include <math.h>
7 #include <math.h>
8 #include <math.h>
9 #include <math.h>
10 #include <math.h>
11 #include <math.h>
12 struct Vec3 {
13     float x;
14     Vec3();
15     Vec3(float x, float y, float z);
16     ~Vec3();
17 };
18
19 // kurva
20 Vec3 curve(GLfloat t) {
21     bool show = true;
22     Vec3 v;
23     bool mat_shininess = true;
24
25     GLfloat mat_shininess[] = { 100.0, 100.0, 100.0, 1.0 };
26     GLfloat mat_shininess[] = { 100.0, 100.0, 100.0, 1.0 };
27     GLfloat mat_shininess[] = { 100.0 };
28
29     GLUnurbsObj *theNurb;
30
31     void drawObject() {
32         glPushMatrix();
33         glTranslatef(0.0f, 0.0f, 0.0f);
34         glScalef(1.0f, 1.0f, 1.0f);
35         glRotatef(0.0f, 0.0f, 0.0f);
36         glRotatef(0.0f, 0.0f, 0.0f);
37         glRotatef(0.0f, 0.0f, 0.0f);
38         glColor3f(0.0f, 1.0f, 0.0f);
39
40         GLfloat knotval[] = { 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0 };
41         glBegin(GL_QUADS);
42         gluSphereSurface(theNurb, R, knots, n_knots,

```

Gambar 8. 3 output posttest 8 representasi permukaan

BAB IX. TEKNIK PERMODELAN OBYEK 3D

1. Pretest 9

- 1) Sebutkan teknik permodelan 3D yang anda ketahui!
- 2) Jelaskan setiap teknik permodelan 3D yang anda sebutkan di nomor 1!

JAWAB

- 1) Teknik permodelan 3D
- 2) Penjelasan mengenai setiap teknik permodelan 3D yang telah disebutkan di nomor 1:
 - Polygon Surfaces / Polyhedra :
 - Merepresentasikan objek 3D dengan gabungan polygon tertutup yang membentuk objek baru. Polygon mudah diproses dan biasanya berbentuk segitiga.
 - Spesifikasi polygon meliputi tabel data polygon, tabel geometri, dan tabel atribut seperti transparansi, warna, dan tekstur.
 - Ray Tracing :
 - Merepresentasikan objek 3D dengan mengikuti jalur sinar cahaya yang bergerak di dalam objek. Teknik ini dapat menghasilkan efek yang sangat realistik, seperti bayangan, refleksi, dan pencahayaan global.
 - Kelebihan ray tracing adalah dapat menghasilkan citra yang sangat realistik, tetapi kelemahannya adalah memerlukan prosesor yang kuat dan memori yang besar.
 - Radiosity :

Merepresentasikan objek 3D dengan menggambarkan penyebaran cahaya di dalam objek. Teknik ini biasanya digunakan untuk menampilkan

pencahayaan yang lebih realistik dan detail

- Scanline Rendering :

Merepresentasikan objek 3D dengan menggambarkan objek secara garis-garis, yaitu dengan menghitung warna dan tekstur pada setiap garis yang membentuk objek

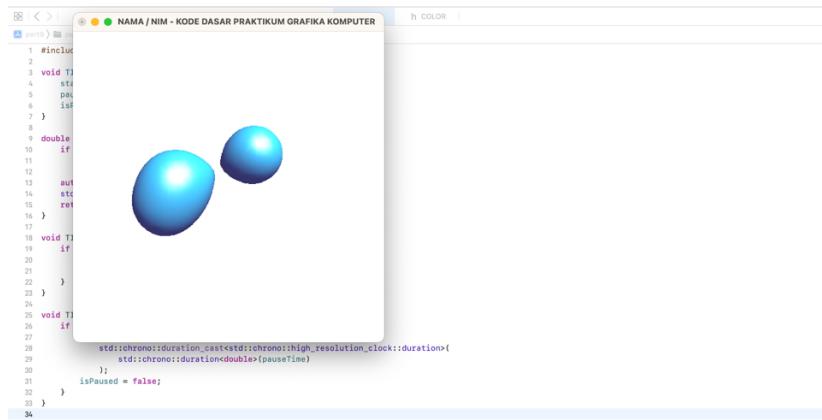
- Voxel-Based Rendering :

Merepresentasikan objek 3D dengan menggambarkan objek sebagai kumpulan kubus kecil (voxel) yang membentuk objek. Teknik ini dapat menghasilkan efek yang lebih dinamis dan detail, seperti pencahayaan volumetrik dan kabut.

1. Langkah praktikum 9

Langkah-langkah praktikum :

- 1) Buka aplikasi Dev C++ dan buat projek baru dengan console.
- 2) Copy file program txt yang diberikan oleh asisten praktikum ke dalam projek yang saja dibuat..
- 3) Isi linker pada menu parameter dengan -lopengl32, -lfreeglut, dan -lglu32
- 4) Kemudian, pada menu Directories klik pada menu Library Directories dan masukkan folder x64.
- 5) Kemudian, masih pada menu Directories klik pada menu Include Directories dan masukkan folder include.
- 6) Setelah itu lakukan compile dan run untuk melihat output dari program tersebut.



```

1 #include <iostream>
2 #include <Windows.h>
3 #include <chrono>
4 #include <thread>
5 #include <vector>
6 #include <cmath>
7 #include <random>
8 #include <assert.h>
9 double pauseTime = 0.001;
10
11
12
13
14
15
16
17
18 void Timer::start()
19 {
20     if (!isRunning)
21     {
22         isRunning = true;
23     }
24 }
25 void Timer::stop()
26 {
27     if (isRunning)
28     {
29         std::chrono::duration<std::chrono::high_resolution_clock::duration>(std::chrono::duration<double>(pauseTime));
30     }
31     isPaused = false;
32 }
33
34

```

Gambar 9. 1 hasil output langkah praktikum 9 permodelan objek 3D

Penjelasan singkat :

- Pada output tersebut terdapat 2 objek eclips yang bisa menjadi 1 model objek 3D.

1.Posttest 9

Langkah-langkah:

- 1) Pada kasus ini, kita akan merubah jumlah eclipse menjadi 2 dengan warna eclips menjadi warna ungu.
- 2) Ubah jumlah eclipse menjadi 2.

```

30
31 const int numMetaballs = 2;
32 METABALL metaballs[numMetaballs];
33 TIMER timers;
34

```

Gambar 9. 2 ubah jumlah eclipse menjadi 2

- 3) Memperbarui fungsi menjadi 2 untuk mengatur posisi eclipse sehingga eclipse dapat berpisah dan bergabung.

```

64
65 // update posisi metaball
66 float c = 2.0f*(float)cos(timers.GetTime() / 600);
67
68 metaballs[0].position.x = -4.0f*(float)cos(timers.GetTime()/700) - c;
69 metaballs[0].position.y = 4.0f*(float)sin(timers.GetTime()/600) - c;
70
71 metaballs[1].position.x = 5.0f*(float)sin(timers.GetTime()/400) + c;
72 metaballs[1].position.y = 5.0f*(float)cos(timers.GetTime()/400) - c;
73

```

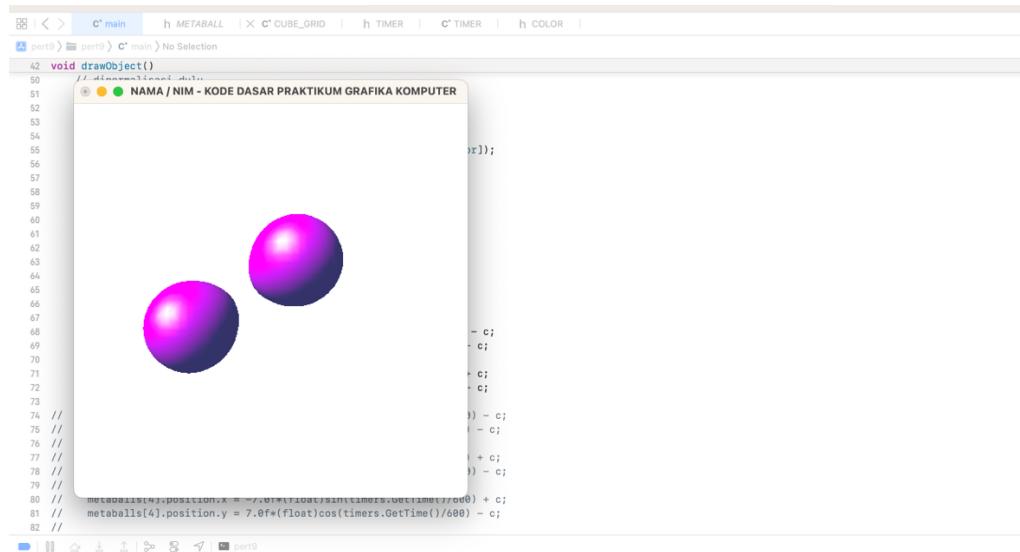
Gambar 9. 3 memperbarui fungsi eclipse menjadi 2

4) Ubah warna eclipse.

```
187 //Set Up Colors
188 diffuseColors[0].Set(1.0f, 0.0f, 1.0f, 1.0f);
189 diffuseColors[1].Set(0.047f, 0.839f, 0.271f, 1.0f);
190 diffuseColors[2].Set(0.976f, 0.213f, 0.847f, 1.0f);
191
192 timers.Reset();
193 }
194
195
```

Gambar 9. 4 ubah warna eclipse menjadi ungu

5) Setelah selesai merubah kode program, lakukan compile dan run pada program untuk melihat hasil output dari program tersebut.



Gambar 9. 5 hasil output program posttest 9

BAB X. TEKNIK SUBDIVISI

1. Pretest 10

- 1) Bagaimana cara menerapkan subdivisi 2x pada permukaan segitiga?

JAWAB:

- 1) Cara menerapkan subdivisi 2x pada permukaan segitiga

- a. Mengidentifikasi Titik Kendali :

Titik kendali adalah titik yang digunakan sebagai referensi dalam menggambarkan permukaan. Pada subdivisi 2x, titik kendali adalah titik-titik yang membentuk segitiga asli.

Contoh : titik A, titik B, titik C.

- b. Menghitung Titik Baru :

Dalam subdivisi 2x, titik baru dibentuk dengan menghitung titik tengah dari setiap rusuk segitiga asli. Titik baru ini akan membentuk segitiga baru yang lebih kecil.

– Titik tengah D dari sisi A dan B adalah $D = \frac{A+B}{2}$

2

– Titik tengah E dari sisi B dan C adalah $E = \frac{B+C}{2}$

2

– Titik tengah F dari sisi C dan A adalah $F = \frac{C+A}{2}$

2

- c. Menghitung Sisi Baru :

Sisi baru dibentuk dengan menghitung sisi-sisi segitiga baru yang terbentuk dari titik baru. Sisi baru ini akan membentuk segitiga yang lebih kecil lagi.

- d. Menghitung Titik Baru Kedua :

Dalam subdivisi 2x, titik baru kedua dibentuk dengan menghitung titik tengah dari setiap rusuk segitiga baru yang terbentuk. Titik baru kedua ini akan membentuk segitiga yang lebih kecil lagi.

e. Menghitung Sisi Baru Kedua :

Sisi baru kedua dibentuk dengan menghitung sisi-sisi segitiga baru yang terbentuk dari titik baru kedua. Sisi baru kedua ini akan membentuk segitiga yang lebih kecil lagi.

f. Mengulangi Proses :

Proses subdivisi 2x dapat diulangi beberapa kali untuk mencapai tingkat detail yang diinginkan. Setiap pengulangan disebut sebagai level subdivisi.

g. Menghitung Permukaan Akhir :

Permukaan akhir dibentuk dengan menggabungkan semua titik dan sisi yang dihasilkan dari subdivisi 2x. Permukaan akhir ini akan memiliki tingkat detail yang lebih halus daripada permukaan asli.

2. Langkah praktikum 10

Langkah-langkah praktikum :

- 1) Buka aplikasi Dev C++ dan buat projek baru dengan console.
- 2) Copy file program txt yang diberikan oleh asisten praktikum ke dalam projek yang saja dibuat..
- 3) Isi linker pada menu parameter dengan -lopengl32, -lfreeglut, dan -lglu32
- 4) Kemudian, pada menu Directories klik pada menu Library Directories dan masukkan folder x64.
- 5) Kemudian, masih pada menu Directories klik pada menu Include Directories dan masukkan folder include.
- 6) Setelah itu lakukan compile dan run untuk melihat output dari program tersebut.

```

1 #include <
2 #include <
3 #define GL_
4 #include <
5
6 // --- Dek.
7 // Transformasi
8 float posX;
9 float posY;
10 float posZ;
11
12 // Transformasi
13 float objekt;
14 float objekt;
15
16 // --- Data
17 #define vX;
18 #define vZ;
19
20 // Array di dalam
21 static QLff;
22 {
23     {-vX, 0,
24      { 0.0,
25       { vZ, 0,
26     };
27
28 // Indeks
29 static int;
30 {1,4,0,
31 {1,10,0,
32 {3,10,0,
33 {10,1,0,
34 };
35
36 // --- Properti Material dan Cahaya ---
37 GLfloat mat_specular[] = { 0.0, 0.0, 0.0, 1.0 };
38 GLfloat mat_diffuse[] = { 0.8, 0.6, 0.4, 1.0 };
39 GLfloat mat_ambient[] = { 0.8, 0.6, 0.4, 1.0 };
40 GLfloat mat_shininess = 100.0;
41

```

Gambar 10. 1 hasil output langkah praktikum 10 penerapan teknik subdivisi

3. Posttest 10

Langkah-langkah :

- 1) Pada kasus kali ini kita akan membuat sebuah obyek torus (gunakan SolidTorus) kemudian terapkan subdivisi dengan ring 5, 10, 20, 50
- 2) Ubah torusRings pada kode program

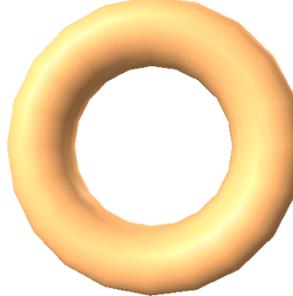
```

85
86     switch (key) {
87         case GLUT_KEY_LEFT: objectAngleY -= 5.0f; break;
88         case GLUT_KEY_RIGHT: objectAngleY += 5.0f; break;
89         case GLUT_KEY_UP: objectAngleX -= 5.0f; break;
90         case GLUT_KEY_DOWN: objectAngleX += 5.0f; break;
91         case GLUT_KEY_PAGE_UP: posZ -= step * 5; break;
92         case GLUT_KEY_PAGE_DOWN: posZ += step * 5; break;
93
94         case GLUT_KEY_F5: torusRings = 5; break;
95         case GLUT_KEY_F6: torusRings = 10; break;
96         case GLUT_KEY_F7: torusRings = 20; break;
97         case GLUT_KEY_F8: torusRings = 50; break;
98     }
99
100    glutPostRedisplay();
101 }

```

Gambar 10. 2 ubah icosahedron pada kode program

- 3) Setelah melakukan perubahan pada kode program, lakukan compile dan run untuk melihat hasil output pada program.



```
C main
post10 C main > No Selection
1 #include <iostream>
2 #include <cmath>
3 #define pi 3.141592653589793
4 #include <GL/gl.h>
5
6 // === K
7 float pi;
8 float ro;
9 float ob;
10
11 // === T
12 int toru;
13 int toru;
14 float to;
15
16 // === M
17 GLfloat
18 GLfloat
19 GLfloat
20 GLfloat
21
22 GLfloat
23 GLfloat
24 GLfloat
25
26 GLfloat
27 GLfloat
28
29 // === G
30 void draw();
31
32
33
34
35
36 // T
37
38
39
40 glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
```

Gambar 10. 3 output posttest 10 subdivisi ring

Daftar Pustaka

- [1]. Dufatancom, “Pengertian OpenGL dan Cara Kerjanya,” <https://www.dufatancom.id/2019/08/pengertian-opengl-dan-cara-kerjanya.html>, 2019.
- [2]. J. Widadi, Murinto “Media Pembelajaran Materi Pengenalan Opengl Pada Mata Kuliah Grafika Komputer,” Jurnal Sarjana Teknik Informatika, vol.6, no.1, Feb. 2018.
- [3]. Akunerio, “Pengenalan OpenGL & Glut,” <https://www.akunerio.com/pengenalan-opengl-glut/>, 2014
- [4]. Herriyance, M. Dahria, “Penerapan transformasi Translasi Dan Rotasi Untuk Visualisasi Objek 3D Pada Aplikasi Desktop,” J-SISKO TECH (Jurnal Teknologi Sistem Informasi dan Sistem Komputer TGD), vol. 1, no. 2, Jul. 2018, doi: 10.53513/jsk.v1i2.35.
- [5]. GeeksforGeeks, “OpenGL Rendering Pipeline | An Overview,” <https://www.geeksforgeeks.org/opengl-rendering-pipeline-overview/>, 2022.
- [6]. Khronos, “Rendering Pipeline Overview,” https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview, 2022.
- [7]. P. Novantara, T.Sugiharto , E.Januar, “IMPLEMENTASI ALGORITMA DIGITAL DIFFERENTIAL ANALYZER DALAM PENENTUAN RUTE PADA PETA TOPOGRAFI BERBASIS ANDROID”, vol. 6, no. 2, Nov 2021, doi: 10.25134/jejaring.v6i2.6740.
- [8]. K. Hartomo, “IMPLEMENTASI METODE INTERPOLASI LINEAR UNTUK PEMBESARAN RESOLUSI CITRA,” Journal Teknoin, vol. 11, no.3, Sep 2006. [9]. Harmastuti, D. Setyowati, “REPRESENTASI OBJEK 3D MENGGUNAKAN TEKNIK RENDERING RAY-TRACING DAN RADIOSITY,” <https://ejournal.akprind.ac.id/index.php/technoscientia/article/view/3161>, vol.15, no.1, Feb 2022, doi: 10.34151/technoscientia.v13i2.3161.
- [10]. Gustiam74, “ALGORITMA PEMBENTUKAN GARIS (DDA DAN BRESENHAM),” <https://gustiam74.blogspot.com/2016/04/algoritma-pembentukan-garis-dda-dan.html>, 2016.