

PP/018/V/R4



**LABORATORIUM  
S1 INFORMATIKA  
FAKULTAS TEKNOLOGI INDUSTRI  
UNIVERSITAS AHMAD DAHLAN**

# **PETUNJUK PRAKTIKUM EDISI KURIKULUM OBE GRAFIKA KOMPUTER**



**Penyusun:**  
**Adhi Prahara, S.Si., M.Cs.**  
**Ahmad Azhari, S.Kom., M.Eng.**  
**Dr. Murinto, S.Si., M.Kom.**  
**Faisal Fajri Rahani, S.Si., M.Cs.**

**2024**

## HAK CIPTA

### PETUNJUK PRAKTIKUM GRAFIKA KOMPUTER

**Copyright© 2024,**

Adhi Prahara, S.Si., M.Cs.

Ahmad Azhari, S.Kom., M.Eng.

Dr. Murinto, S.Si., M.Kom.

Faisal Fajri Rahani, S.Si., M.Cs.

**Hak Cipta dilindungi Undang-Undang**

Dilarang mengutip, memperbanyak atau mengedarkan isi buku ini, baik sebagian maupun seluruhnya, dalam bentuk apapun, tanpa izin tertulis dari pemilik hak cipta dan penerbit.

**Diterbitkan oleh:**

**Program Studi Informatika**

Fakultas Teknologi Industri

Universitas Ahmad Dahlan

Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166

**Penulis**

: Adhi Prahara, S.Si., M.Cs.

Ahmad Azhari, S.Kom., M.Eng.

Dr. Murinto, S.Si., M.Kom.

Faisal Fajri Rahani, S.Si., M.Cs.

**Editor**

: Laboratorium Informatika, Universitas Ahmad Dahlan

**Desain sampul**

: Laboratorium Informatika, Universitas Ahmad Dahlan

**Tata letak**

: Laboratorium Informatika, Universitas Ahmad Dahlan

**Ukuran/Halaman**

: 21 x 29,7 cm / 86 halaman

**Didistribusikan oleh:**



**Laboratorium Informatika**

Universitas Ahmad Dahlan

Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55191

Indonesia

## KATA PENGANTAR

Puji syukur kami panjatkan kehadiran Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga petunjuk praktikum Grafika Komputer dapat diselesaikan dengan lancar. Kami ucapkan terima kasih kepada berbagai pihak yang telah mendukung dalam penyusunan petunjuk praktikum ini.

Petunjuk praktikum Grafika Komputer disusun untuk memberikan panduan dan kemudahan bagi mahasiswa dalam memahami materi grafika komputer seperti OpenGL, sistem koordinat, transformasi, dan teknik pemodelan.

Kami sadar bahwa dalam penyusunan petunjuk praktikum ini masih banyak kekurangan sehingga kritik dan saran sangat kami harapkan.

Yogyakarta, Februari 2024

Penyusun

## DAFTAR PENYUSUN

Adhi Prahara, S.Si., M.Cs.

Ahmad Azhari, S.Kom., M.Eng.

Dr. Murinto, S.Si., M.Kom.

Faisal Fajri Rahani, S.Si., M.Cs.

## HALAMAN REVISI

Yang bertanda tangan di bawah ini:

Nama : Dr. Murinto, S.Si., M.Kom.

NIPM : 19730710 200409 111 0951298

Jabatan : Koordinator Praktikum Grafika Komputer

Dengan ini menyatakan pelaksanaan Revisi Petunjuk Praktikum Grafika Komputer untuk Program Studi Informatika telah dilaksanakan dengan penjelasan sebagai berikut:

No	Keterangan Revisi	Tanggal Revisi	Nomor Modul
1	a. Menyesuaikan kode di semua pertemuan praktikum ke C++ 11. b. Menambahkan cover depan dan belakang.	25 Februari 2019	PP/018/IV/R1
2	a. Mengganti cover depan dan belakang. b. Menambahkan halaman daftar penyusun, halaman revisi, halaman pernyataan, halaman visi dan misi prodi informatika, halaman tata tertib praktikum, halaman daftar isi, tabel dan gambar, c. Menambahkan bagian alokasi waktu, total skor, dan indikator capaian di setiap pertemuan praktikum. d. Menambahkan bagian tugas di setiap pertemuan praktikum. e. Menambahkan halaman lembar jawaban pretest dan post test di setiap pertemuan praktikum.	23 Februari 2020	PP/018/IV/R2
3	a. Menyesuaikan dengan template kurikulum OBE b. Merevisi setiap pertemuan sesuai dengan kurikulum OBE c. Menambahkan pre-test, praktikum dan post-test	20 Februari 2022	PP/018/IV/R3
4	a. Menyesuaikan dengan template kurikulum OBE 2024 b. Menyesuaikan CPL dan CPMK sesuai dengan RPS Mata Kuliah	26 Februari 2024	PP/018/IV/R4

Yogyakarta, 26 Februari 2024

Koordinator Tim Penyusun



Dr. Murinto, S.Si., M.Kom.

NIPM. 19730710 200409 111 0951298

## HALAMAN PERNYATAAN

Yang bertanda tangan di bawah ini:

Nama : Murein Miksa Mardhia S.T., M.T.

NIPM : 19891019 201606 011 1236278

Jabatan : Kepala Laboratorium S1 Informatika

Menerangkan dengan sesungguhnya bahwa Petunjuk Praktikum ini telah direview dan akan digunakan untuk pelaksanaan praktikum di Semester Genap Tahun Akademik 2023/2024 di Laboratorium Praktikum Informatika, Program Studi Informatika, Fakultas Teknologi Industri, Universitas Ahmad Dahlan.

Yogyakarta, 26 Februari 2024

Mengetahui,  
Ketua Kelompok Keilmuan Rekayasa Perangkat  
Lunak dan Data (RELATA)



**Guntur Maulana Zamroni, B.Sc. M. Kom**  
NIPM. 19840309 201810 111 1205917

Kepala Laboratorium S1 Informatika



**Murein Miksa Mardhia, S.T., M.T.**  
NIPM. 19891019 201606 011 1236278

## VISI DAN MISI PRODI INFORMATIKA

### VISI

Menjadi Program Studi Informatika yang diakui secara internasional dan unggul dalam bidang Informatika serta berbasis nilai-nilai Islam.

### MISI

1. Menjalankan pendidikan sesuai dengan kompetensi bidang Informatika yang diakui nasional dan internasional
2. Meningkatkan penelitian dosen dan mahasiswa dalam bidang Informatika yang kreatif, inovatif dan tepat guna.
3. Meningkatkan kuantitas dan kualitas publikasi ilmiah tingkat nasional dan internasional
4. Melaksanakan dan meningkatkan kegiatan pengabdian masyarakat oleh dosen dan mahasiswa dalam bidang Informatika.
5. Menyelenggarakan aktivitas yang mendukung pengembangan program studi dengan melibatkan dosen dan mahasiswa.
6. Menyelenggarakan kerja sama dengan lembaga tingkat nasional dan internasional.
7. Menciptakan kehidupan Islami di lingkungan program studi.

## TATA TERTIB LABORATORIUM INFORMATIKA

### DOSEN/KOORDINATOR PRAKTIKUM

1. Dosen harus hadir saat praktikum minimal 15 menit di awal kegiatan praktikum dan menandatangani presensi kehadiran praktikum.
2. Dosen membuat modul praktikum, soal seleksi asisten, pre-test, post-test, dan responsi dengan berkoordinasi dengan asisten dan pengampu mata praktikum.
3. Dosen berkoordinasi dengan koordinator asisten praktikum untuk evaluasi praktikum setiap minggu.
4. Dosen menandatangani surat kontrak asisten praktikum dan koordinator asisten praktikum.
5. Dosen yang tidak hadir pada slot praktikum tertentu tanpa pemberitahuan selama 2 minggu berturut-turut mendapat teguran dari Kepala Laboratorium, apabila masih berlanjut 2 minggu berikutnya maka Kepala Laboratorium berhak mengganti koordinator praktikum pada slot tersebut.

### PRAKTIKAN

1. Praktikan harus hadir 15 menit sebelum kegiatan praktikum dimulai, dan dispensasi terlambat 15 menit dengan alasan yang jelas (kecuali asisten menentukan lain dan patokan jam adalah jam yang ada di Laboratorium, terlambat lebih dari 15 menit tidak boleh masuk praktikum & dianggap INHAL).
2. Praktikan yang tidak mengikuti praktikum dengan alasan apapun, wajib mengikuti INHAL, maksimal 4 kali praktikum dan jika lebih dari 4 kali maka praktikum dianggap GAGAL.
3. Praktikan harus berpakaian rapi sesuai dengan ketentuan Universitas, sebagai berikut:
  - a. Tidak boleh memakai Kaos Oblong, termasuk bila ditutupi Jaket/Jas Almamater (Laki-laki / Perempuan) dan Topi harus Dilepas.
  - b. Tidak Boleh memakai Baju ketat, Jilbab Minim dan rambut harus tertutup jilbab secara sempurna, tidak boleh kelihatan di jidat maupun di punggung (khusus Perempuan).
  - c. Tidak boleh memakai baju minim, saat duduk pun pinggang harus tertutup rapat (Laki-laki / Perempuan).
  - d. Laki-laki tidak boleh memakai gelang, anting-anting ataupun aksesoris Perempuan.
4. Praktikan tidak boleh makan dan minum selama kegiatan praktikum berlangsung, harus menjaga kebersihan, keamanan dan ketertiban selama mengikuti kegiatan praktikum atau selama berada di dalam laboratorium (tidak boleh membuang sampah sembarangan baik kertas, potongan kertas, bungkus permen baik di lantai karpet maupun di dalam ruang CPU).
5. Praktikan dilarang meninggalkan kegiatan praktikum tanpa seizin Asisten atau Laboran.
6. Praktikan harus meletakkan sepatu dan tas pada rak/loker yang telah disediakan.
7. Selama praktikum dilarang NGENET/NGE-GAME, kecuali mata praktikum yang membutuhkan atau menggunakan fasilitas Internet.
8. Praktikan dilarang melepas kabel jaringan atau kabel power praktikum tanpa sepengetahuan laboran
9. Praktikan harus memiliki FILE Petunjuk praktikum dan digunakan pada saat praktikum dan harus siap sebelum praktikum berlangsung.
10. Praktikan dilarang melakukan kecurangan seperti mencontek atau menyalin pekerjaan praktikan yang lain saat praktikum berlangsung atau post-test yang menjadi tugas praktikum.
11. Praktikan dilarang mengubah setting software/hardware komputer baik menambah atau mengurangi tanpa permintaan asisten atau laboran dan melakukan sesuatu yang dapat merugikan laboratorium atau praktikum lain.



12. Asisten, Koordinator Praktikum, Kepala laboratorium dan Laboran mempunyai hak untuk menegur, memperingatkan bahkan meminta praktikan keluar ruang praktikum apabila dirasa anda mengganggu praktikan lain atau tidak melaksanakan kegiatan praktikum sebagaimana mestinya dan atau tidak mematuhi aturan lab yang berlaku.
13. Pelanggaran terhadap salah satu atau lebih dari aturan diatas maka Nilai praktikum pada pertemuan tersebut dianggap 0 (NOL) dengan status INHAL.
14. Peraturan lain mengikuti aturan yang ditetapkan oleh Fakultas pada Surat Keputusan Dekan No. F2/524/A/IX/2018.
15. Sebelum pelaksanaan praktikum membaca surat pendek.
16. Praktikan saat memasuki lab wajib menggunakan kaos kaki.

### ASISTEN PRAKTIKUM

1. Asisten harus hadir 15 Menit sebelum praktikum dimulai (konfirmasi ke koordinator bila mengalami keterlambatan atau berhalangan hadir).
2. Asisten yang tidak bisa hadir WAJIB mencari pengganti, dan melaporkan kepada Koordinator Asisten.
3. Asisten harus berpakaian rapi sesuai dengan ketentuan Universitas, sebagai berikut:
  - a. Tidak boleh memakai Kaos Oblong, termasuk bila ditutupi Jaket/Jas Almamater (Laki-laki / Perempuan) dan Topi harus Dilepas.
  - b. Tidak Boleh memakai Baju ketat, Jilbab Minim dan rambut harus tertutup jilbab secara sempurna, tidak boleh kelihatan di jidat maupun di punggung (khusus Perempuan).
  - c. Tidak boleh memakai baju minim, saat duduk pun pinggang harus tertutup rapat (Laki-laki / Perempuan).
  - d. Laki-laki tidak boleh memakai gelang, anting-anting ataupun aksesoris Perempuan.
4. Asisten harus menjaga kebersihan, keamanan dan ketertiban selama mengikuti kegiatan praktikum atau selama berada di laboratorium, menegur atau mengingatkan jika ada praktikan yang tidak dapat menjaga kebersihan, ketertiban atau kesopanan.
5. Asisten harus dapat merapikan dan mengamankan presensi praktikum, Kartu Nilai serta tertib dalam memasukan/Input nilai secara Online/Offline.
6. Asisten harus dapat bertindak secara profesional sebagai seorang asisten praktikum dan dapat menjadi teladan bagi praktikan.
7. Asisten harus dapat memberikan penjelasan/pemahaman yang dibutuhkan oleh praktikan berkenaan dengan materi praktikum yang diasistensi sehingga praktikan dapat melaksanakan dan mengerjakan tugas praktikum dengan baik dan jelas.
8. Asisten tidak diperkenankan mengobrol sendiri apalagi sampai membuat gaduh.
9. Asisten dimohon mengkoordinasikan untuk meminta praktikan agar mematikan komputer untuk jadwal terakhir dan sudah dilakukan penilaian terhadap hasil kerja praktikan.
10. Asisten wajib untuk mematikan LCD Projector dan komputer asisten/praktikan apabila tidak digunakan.
11. Asisten tidak diperkenankan menggunakan akses internet selain untuk kegiatan praktikum, seperti Youtube/Game/Medsos/Streaming Film di komputer praktikan.
12. Peraturan lain mengikuti aturan yang ditetapkan oleh Fakultas pada Surat Keputusan Dekan No. F2/524/A/IX/2018.
13. Sebelum pelaksanaan praktikum membaca surat pendek.
14. Asisten saat memasuki lab wajib menggunakan kaos kaki.

### LAIN-LAIN

1. Pada Saat Responsi Harus menggunakan Baju Kemeja untuk Laki-laki dan Perempuan untuk Praktikan dan Asisten.
2. Ketidakhadiran praktikum dengan alasan apapun dianggap INHAL.
3. Izin praktikum mengikuti aturan izin SIMERU/KULIAH.
4. Yang tidak berkepentingan dengan praktikum dilarang mengganggu praktikan atau membuat keributan/kegaduhan.
5. Penggunaan lab diluar jam praktikum maksimal sampai pukul 21.00 dengan menunjukkan surat ijin dari Kepala Laboratorium Prodi Informatika.

Yogyakarta, 24 Februari 2026

Kepala Laboratorium S1 Informatika



**Murein Miksa Mardhia, S.T., M.T.**

NIPM. 19891019 201606 011 1236278

## DAFTAR ISI

HAK CIPTA .....	0
KATA PENGANTAR.....	2
DAFTAR PENYUSUN.....	3
HALAMAN REVISI.....	4
HALAMAN PERNYATAAN.....	5
VISI DAN MISI PRODI INFORMATIKA.....	6
TATA TERTIB LABORATORIUM INFORMATIKA .....	7
DAFTAR ISI .....	10
DAFTAR GAMBAR .....	11
SKENARIO PRAKTIKUM SECARA DARING .....	12
PERSIAPAN PRAKTIKUM .....	13
PRAKTIKUM 1: PENGANTAR OPENGL .....	17
PRAKTIKUM 2: ALGORITMA GARIS .....	21
PRAKTIKUM 3: INTERPOLASI DAN KURVA .....	30
PRAKTIKUM 4: TRANSFORMASI 2D DAN 3D.....	37
PRAKTIKUM 5: PROYEKSI 3D.....	44
PRAKTIKUM 6: REPRESENTASI OBYEK 3D .....	50
PRAKTIKUM 7: KURVA SPLINE .....	56
PRAKTIKUM 8: TEKNIK REPRESENTASI PERMUKAAN .....	65
PRAKTIKUM 9: TEKNIK PEMODELAN OBYEK 3D .....	70
PRAKTIKUM 10: TEKNIK SUBDIVISI .....	77
DAFTAR PUSTAKA.....	86

## DAFTAR GAMBAR

Gambar 1.1 OpenGL Pipeline .....	18
Gambar 1.2 Struktur Program OpenGL.....	18
Gambar 1.3 Pengaturan Include OpenGL di Visual Studio.....	19
Gambar 1.4 Pengaturan Linker untuk OpenGL di Visual Studio. ....	19
Gambar 1.5 Tampilan hasil dari kode dasar.....	20
Gambar 2.1 Hasil pembuatan garis dengan algoritma DDA. ....	26
Gambar 2.2 Hasil pembuatan garis dengan algoritma Bresenham. ....	29
Gambar 3.1 Hasil penerapan interpolasi linear. ....	35
Gambar 3.2 Hasil penerapan interpolasi cosine. ....	35
Gambar 3.3 Hasil penerapan interpolasi cubic. ....	36
Gambar 4.1 Hasil penerapan translasi pada obyek.....	42
Gambar 4.2 Hasil penerapan rotasi terhadap obyek. ....	42
Gambar 4.3 Hasil penerapan scaling pada obyek. ....	43
Gambar 5.1 Hasil proyeksi perspektif pada obyek kubus. ....	47
Gambar 5.2 Hasil proyeksi orthogonal terhadap obyek kubus.....	48
Gambar 6.1 Bentuk polyhedron.....	51
Gambar 6.2 Hasil menggambar obyek kubus .....	53
Gambar 6.3 Hasil menggambar obyek tabung.....	54
Gambar 6.4 Hasil menggambar obyek bola. ....	54
Gambar 7.1 Hasil menggambar kurva spline Cubic. ....	63
Gambar 7.2 Hasil menggambar kurva spline Bezier .....	63
Gambar 7.3 Hasil menggambar kurva spline Catmull-Rom .....	64
Gambar 8.1 Permukaan kurva NURBS .....	69
Gambar 9.1 Permukaan metaball .....	71
Gambar 9.2 Setting file tambahan praktikum09 di Visual Studio .....	72
Gambar 9.3 Hasil menggambar metaball.....	75
Gambar 10.1 Subdivisi permukaan segitiga .....	78
Gambar 10.2 Hasil penerapan teknik subdivisi pada permukaan icosahedron menjadi bola.....	83

## SKENARIO PRAKTIKUM SECARA DARING

Nama Mata Praktikum : Grafika Komputer

Jumlah Pertemuan : 10 praktikum + 1 responsi

**TABEL SKENARIO PRAKTIKUM DARING**

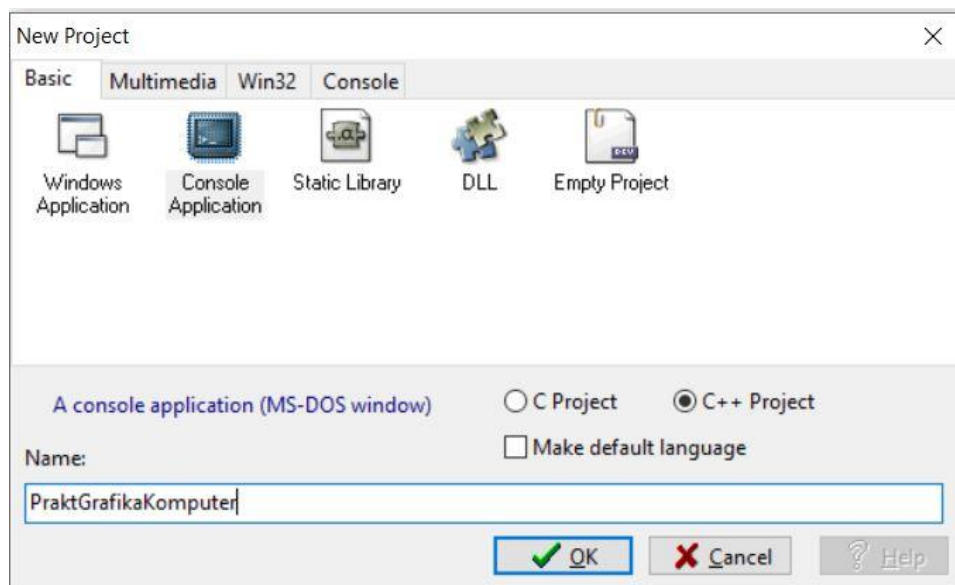
Pertemuan ke	Judul Materi	Waktu (Lama praktikum sampai pengumpulan posttest)	Skenario Praktikum dari pemberian pre-test, post-test dan pengumpulannya serta mencantumkan metode yang digunakan misal video, whatsapp group, Google meet atau lainnya
1	Pengantar OpenGL	1 minggu	Pre-test dan post-test diberikan dan dikumpulkan melalui e-learning (E-Learning UAD atau Google Classroom) sesuai waktu yang ditentukan. Praktikum dilakukan menggunakan video tutorial dan whatsapp/forum di elearning untuk bertanya. Pre-test diberikan di awal praktikum selama 15 menit, keterlambatan diberi waktu sampai waktu praktikum berakhir. Post-test dan laporan praktikum dikerjakan selama 1 minggu sampai praktikum berikutnya dimulai. Laporan praktikum berisi: Cover, Pendahuluan (berisi materi singkat ttg post test yang dikerjakan), Pre-test (berisi jawaban pre-test yang ada di modul praktikum), Praktikum (berisi listing kode jawaban post-test disertai penjelasannya dan screenshot hasil post test) dan Pembahasan (pembahasan singkat dari hasil post test)
2	Algoritma Garis	1 minggu	
3	Interpolasi dan Kurva	1 minggu	
4	Transformasi 2D dan 3D	1 minggu	
5	Proyeksi 3D	1 minggu	
6	Representasi Objek 3D	1 minggu	
7	Kurva Spline	1 minggu	
8	Teknik Representasi Permukaan	1 minggu	
9	Teknik Pemodelan Objek 3D	1 minggu	
10	Teknik Subdivisi	1 minggu	Responsi diberikan dan dikumpulkan melalui e-learning (E-Learning UAD atau Google Classroom) sesuai waktu yang ditentukan. Laporan responsi berisi: Cover, Pendahuluan (berisi materi singkat ttg responsi yang dikerjakan), Responsi (berisi listing kode jawaban responsi disertai penjelasannya dan screenshot hasil responsi) dan Pembahasan (pembahasan singkat dari hasil responsi)
11	Responsi	1 minggu	

## PERSIAPAN PRAKTIKUM

Praktikum grafika komputer dapat dilakukan menggunakan IDE C++ seperti Visual Studio 2010 ke atas, DevC++ dengan compiler 64bit atau yang lain. Default dari petunjuk praktikum ini menggunakan Visual Studio 2010 ke atas. Apabila praktikan menggunakan DevC++ maka akan ada perbedaan sedikit dalam eksekusi programnya seperti yang dijelaskan pada petunjuk di bawah ini.

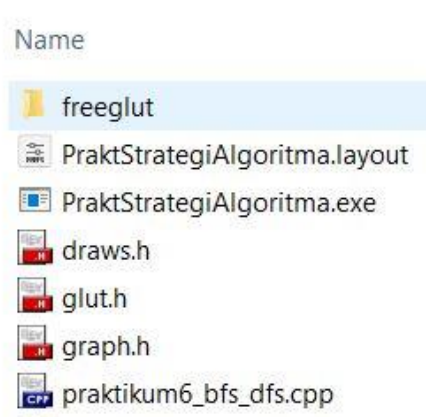
### CARA SETTING OPENGL FREEGLUT PADA DEV++

1. Buka DevC++ dan buat project baru sesuai dengan nama praktikum masing-masing seperti pada contoh di Gambar 1. Pilih console application dan C++ project.



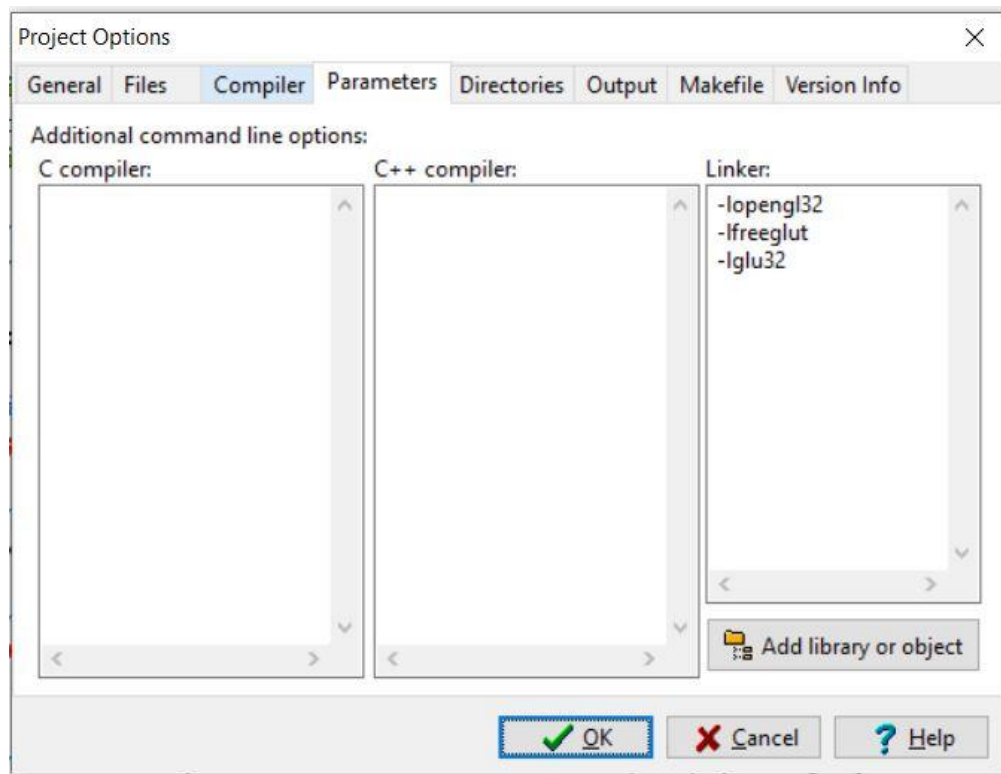
**Gambar 1.** Contoh tampilan new project

2. Download library OpenGL yang **freeglut** di elearning/classroom Grafika Komputer di bagian atas. Ekstrak .rar nya di lokasi folder project anda berada. Contoh tampilan folder project anda ditunjukkan pada Gambar 2.



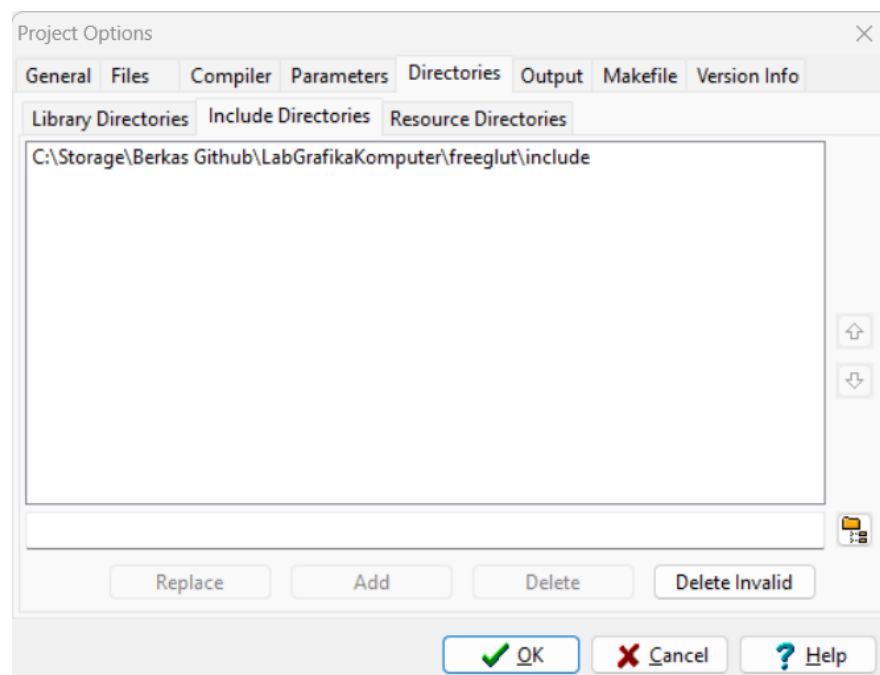
**Gambar 2.** Contoh isian folder project anda

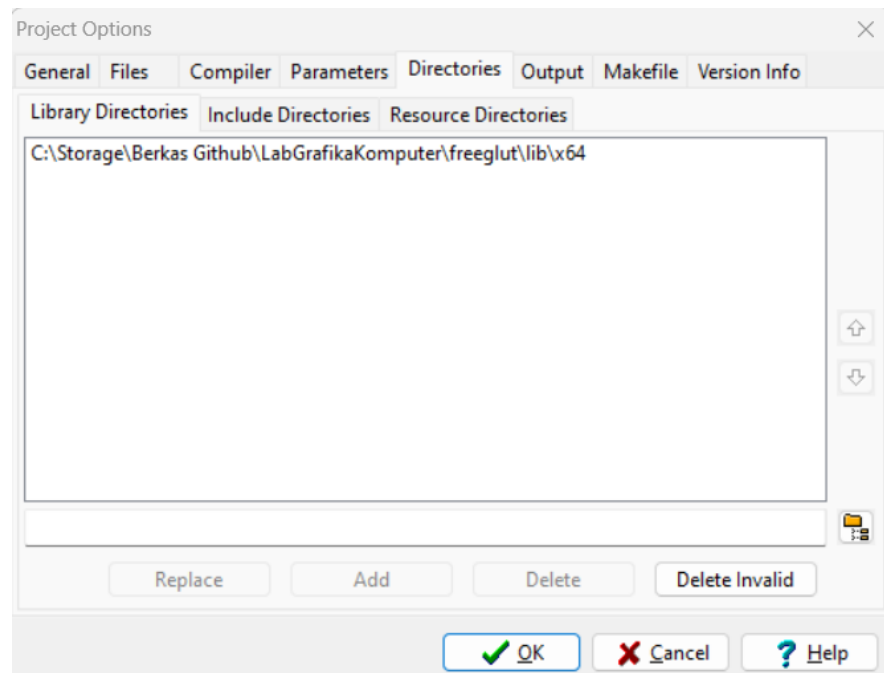
3. Setting OpenGL library pada DevC++ dengan klik Project – Project options – Parameters lalu isikan -lopengl32 -lfreetgl -lglu32 pada Linker seperti pada Gambar 3.



**Gambar 3.** Setting OpenGL pada DevC++

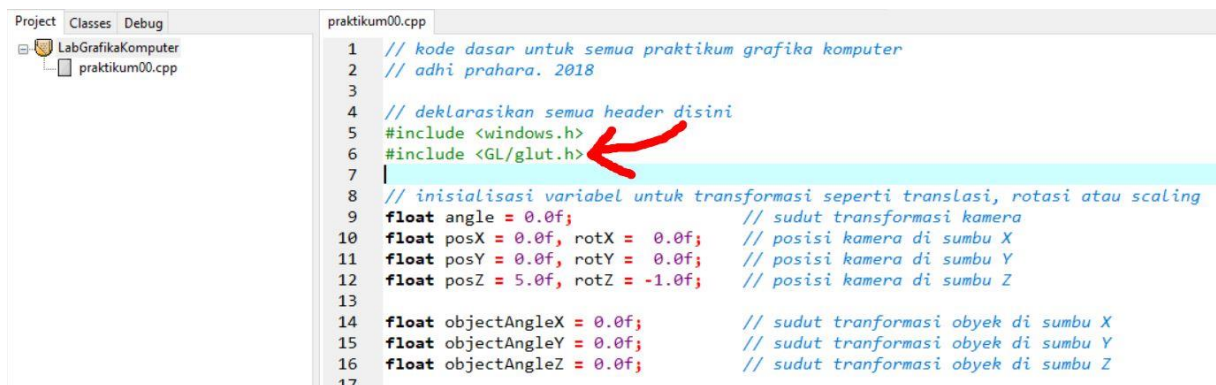
4. Pada bagian Directories, tambahkan lokasi folder include yang ada di folder freeglut yang sudah diekstrak ke include directories dan tambahkan lokasi folder library/x64 ke folder library directories seperti pada Gambar 4.





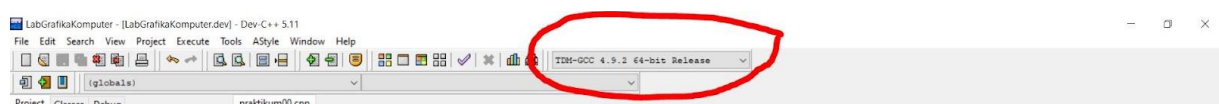
Gambar 4. Tambahkan include dan library directories

- Setting selesai. Anda bisa mulai menuliskan kode C++ yang ada di modul praktikum masing-masing. Ubah header di kode praktikum anda menuju ke letak header `glut.h` dan `glew.h` berada. Misalnya seperti pada Gambar 5.



Gambar 5. Ubah header untuk sesuaikan ke lokasi library Freeglut

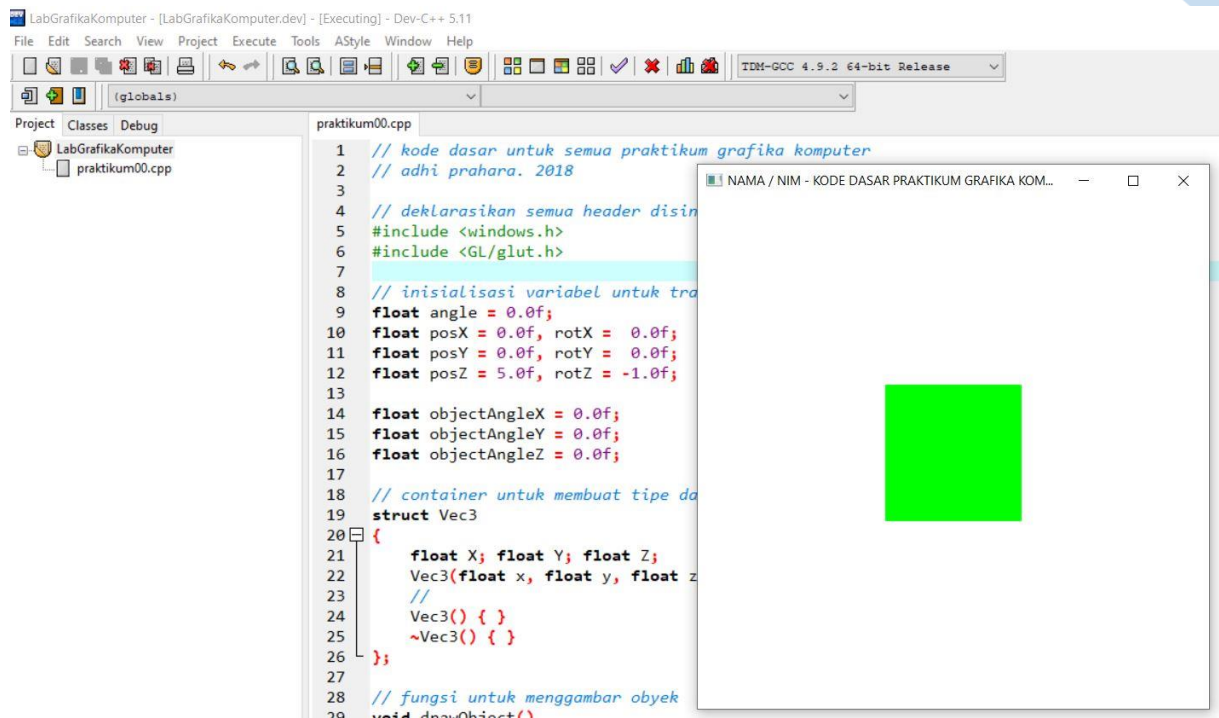
- Karena freeglut yang didownload adalah 64bit maka ubah compiler ke 64bit seperti pada Gambar 6.



Gambar 6. Ubah compiler ke 64bit

- Jalankan programnya untuk mendapatkan hasil di Gambar 7.





**Gambar 7.** Hasil tampilan program dengan OpenGL

## PRAKTIKUM 1: PENGANTAR OPENGL

Pertemuan ke : 1

Total Alokasi Waktu : 90 menit

- Materi : 10 menit
- Pre-Test : 10 menit
- Praktikum : 40 menit
- Post-Test : 30 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 06-P01	Mampu merancang dan mengimplementasikan algoritma/metode dalam mengidentifikasi dan memecahkan masalah yang melibatkan perangkat lunak dan pemikiran komputasi
CPMK-01	Mampu menjelaskan sejarah perkembangan aplikasi komputer grafis dan menerapkan algoritma pembangkitan garis, interpolasi, dan kurva untuk membuat objek grafis menggunakan OpenGL

### 1.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini diharapkan:

1. Mahasiswa mampu menjelaskan tentang OpenGL
2. Mahasiswa mampu menjelaskan kegunaan OpenGL
3. Mahasiswa mampu menjelaskan sintaks dalam OpenGL
4. Mahasiswa mampu membuat program sederhana dengan OpenGL API

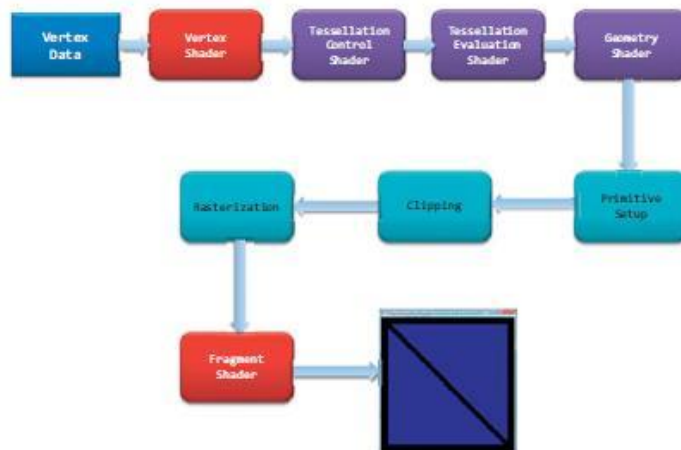
### 1.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

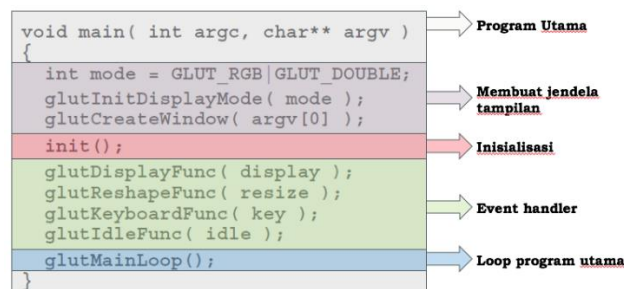
CPL 06-P01	CPMK-01	Kemampuan mahasiswa dalam menerapkan pembuatan program grafis menggunakan library OpenGL.
------------	---------	---

### 1.3. TEORI PENDUKUNG

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). OpenGL bersifat open-source, multi-platform, dan multi-language. OpenGL merupakan suatu antarmuka pemrograman aplikasi (application programming interface/API) yang tidak tergantung pada piranti dan platform yang digunakan, sehingga OpenGL dapat berjalan pada sistem operasi Windows, UNIX, Linux, dan sistem operasi lainnya. Gambar 1.2 menunjukkan tahapan dari OpenGL Pipeline dan Gambar 1.2 menunjukkan struktur program dari OpenGL.



Gambar 1.1 OpenGL Pipeline



Gambar 1.2 Struktur Program OpenGL

#### 1.4. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. Library OpenGL.

#### 1.5. PRE-TEST

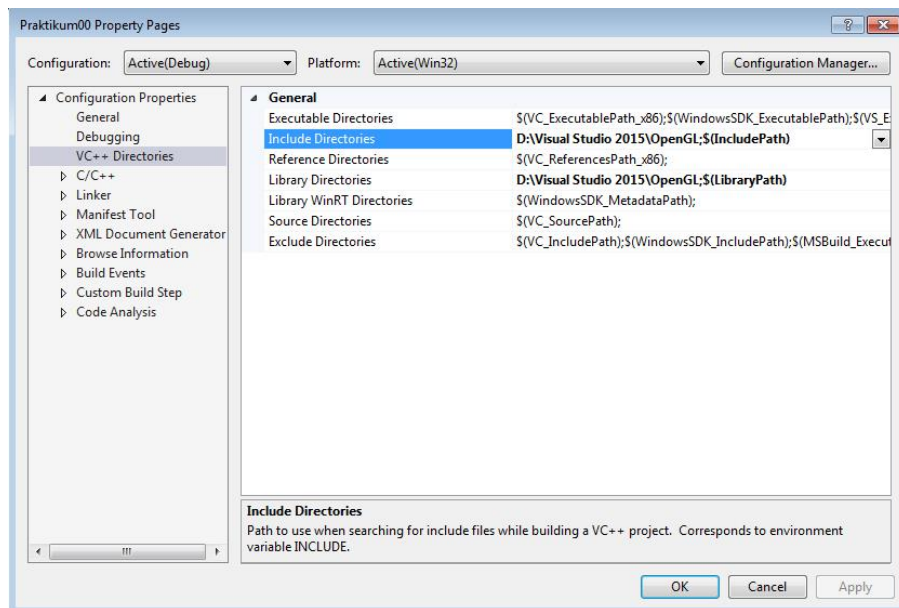
Jawablah pertanyaan berikut (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 06-P01	CPMK-01	Apa saja yang bisa dilakukan dengan library OpenGL?	50
2.	CPL 06-P01	CPMK-01	Gambarkan kemudian jelaskan tahapan OpenGL pipeline!	50

#### 1.6. LANGKAH PRAKTIKUM

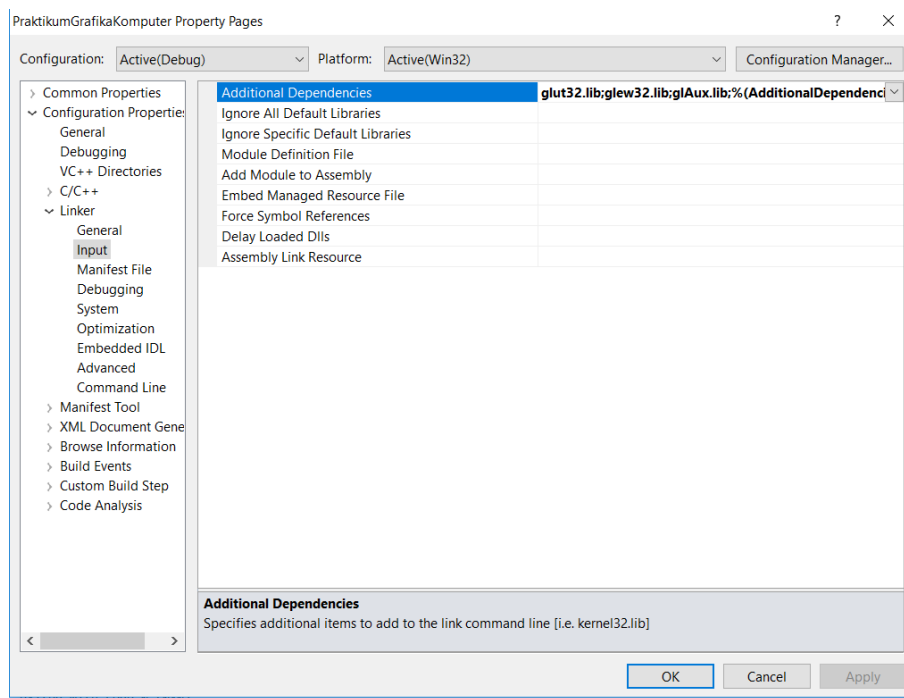
##### PERSIAPAN

1. Buka Visual Studio dan buat project baru dengan nama **praktikum01**.
2. Setting OpenGL library pada Visual Studio C/C++.
  - a. Download OpenGL (**opengl.rar**) dan kode dasar praktikum grafika komputer (**praktikum00.cpp**) di e-learning/classroom dan masukkan file **praktikum00.cpp** ke **Source Files**.
  - b. Tentukan lokasi folder dimana OpenGL library diekstrak atau diletakkan pada hardisk.
  - c. Masuk ke project properties kemudian tambahkan **Include Directories** dan **Library Directories** OpenGL pada Visual Studio seperti ditunjukkan pada Gambar 1.3.



Gambar 1.3 Pengaturan Include OpenGL di Visual Studio.

- d. Isi Linker -> **Input** dengan **glut32.lib, glew32.lib, glAux.lib** seperti yang ditunjukkan pada Gambar 1.4.



Gambar 1.4 Pengaturan Linker untuk OpenGL di Visual Studio.

## PRAKTIKUM

### Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 06-P01	CPMK-01	Selesaikan langkah praktikum 1 – 2	Hasil praktikum langkah 1 – 2	70
2.	CPL 06-P01	CPMK-01	Selesaikan langkah praktikum 3 – 4	Hasil praktikum langkah 3 - 4	30

### Langkah-Langkah:

1. Jalankan kode dasar praktikum grafika komputer untuk mengecek apakah pengaturan pada Visual Studio sudah benar.
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 kode dasar sudah berhasil dijalankan ditunjukkan pada Gambar 1.5.

NAMA / NIM - KODE DASAR PRAKTIKUM GRAFIKA LANJ... — □ ×



Gambar 1.5 Tampilan hasil dari kode dasar.

4. Geser depan, belakang, kanan, dan kiri, dengan tombol arah untuk mengetahui apakah fungsi keyboard berjalan dengan benar.

### 1.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 06-P01	CPMK-01	Buatlah bentuk segitiga siku-siku sama sisi berwarna merah dengan GL_TRIANGLES!	50
2.	CPL 06-P01	CPMK-01	Buatlah bentuk jajar genjang berwarna magenta dengan GL_QUADS!	50

### 1.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 06-P01	CPMK-01	20%		
2.	Praktik	CPL 06-P01	CPMK-01	30%		
3.	Post-Test	CPL 06-P01	CPMK-01	50%		
<b>Total Nilai</b>						

## PRAKTIKUM 2: ALGORITMA GARIS

Pertemuan ke : 2

Total Alokasi Waktu : 90 menit

- Materi : 10 menit
- Pre-Test : 10 menit
- Praktikum : 40 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 06-P01	Mampu merancang dan mengimplementasikan algoritma/metode dalam mengidentifikasi dan memecahkan masalah yang melibatkan perangkat lunak dan pemikiran komputasi
CPMK-01	Mampu menjelaskan sejarah perkembangan aplikasi komputer grafis dan menerapkan algoritma pembangkitan garis, interpolasi, dan kurva untuk membuat objek grafis menggunakan OpenGL

### 2.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan:

1. Mahasiswa mampu menjelaskan konsep menggambar garis dalam komputer grafis.
2. Mahasiswa mampu menjelaskan algoritma untuk membuat garis.
3. Mahasiswa mampu menjelaskan kelebihan dan kekurangan algoritma garis.
4. Mahasiswa mampu menerapkan algoritma garis dengan OpenGL.

### 2.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 06-P01	CPMK-01	Kemampuan mahasiswa dalam menerapkan algoritma pembangkitan garis menggunakan library OpenGL.
------------	---------	---

### 2.3. TEORI PENDUKUNG

Algoritma garis adalah algoritma untuk menentukan lokasi piksel yang paling dekat dengan garis sebenarnya (*actual line*). Algoritma untuk membuat garis ada banyak diantaranya adalah algoritma DDA dan Bresenham.

#### Algoritma DDA (*Digital Differential Analyzer*)

Algoritma DDA merupakan algoritma scan-konversi garis dengan melakukan *sampling* pada garis di rentang  $\Delta x$  atau  $\Delta y$ . Algoritma ini menghitung posisi piksel di sepanjang garis dengan menggunakan posisi piksel sebelumnya. Algoritma DDA dapat dilakukan dengan langkah-langkah berikut:

1. Ketika slope berada pada nilai  $-1 \leq m \leq 1$  maka koordinat  $x$  naik satu demi satu dan koordinat  $y$  naik berdasarkan slope dari garis
2. Dengan menaikkan koordinat  $x$  dengan 1 maka  $y$  dapat dihitung:
 
$$y_{k+1} = y_k + m$$
3. Bila  $m$  diluar nilai tersebut maka lakukan sebaliknya
4. Dengan menaikkan koordinat  $y$  dengan 1 maka  $x$  dapat dihitung:
 
$$x_{k+1} = x_k + \frac{1}{m}$$
5. Selanjutnya nilai hasil perhitungan harus dibulatkan agar cocok dengan nilai piksel

### Algoritma Bresenham

Algoritma Bresenham merupakan algoritma yang dikembangkan oleh Bresenham pada tahun 1965. Algoritma ini merupakan perbaikan dari algoritma perhitungan koordinat piksel garis lurus dengan cara menggantikan operasi bilangan real perkalian dengan operasi penjumlahan. Algoritma Bresenham merupakan algoritma yang sekarang digunakan di komputer grafis modern. Algoritma ini lebih baik dari algoritma DDA, karena menggunakan *incremental algorithm*, yakni nilai sekarang menggunakan nilai sebelumnya. Algoritma DDA digunakan untuk tipe data integer, hal ini bertujuan untuk menghindari operasi *floating point*. Algoritma Bresenham menggunakan fungsi keputusan untuk menentukan letak koordinat selanjutnya. Algoritma Bresenham dapat dilakukan dengan menggunakan langkah-langkah berikut:

1. Bila titik awal garis  $(x_1, y_1)$  dan akhir garis  $(x_2, y_2)$ , untuk inialisasi awal, hitung:
  - Selisih lebar =  $\Delta x = x_2 - x_1$
  - Selisih tinggi =  $\Delta y = y_2 - y_1$
  - $2\Delta y = 2(y_2 - y_1)$
2. Inisial parameter keputusan =  $p_0 = 2\Delta y - \Delta x$
3. Setiap  $x_k$  di sepanjang garis, mulai dari  $k = 0$ , cek kondisi berikut:
  - Jika  $p_k < 0$  maka titik selanjutnya untuk digambar di :  $(x_k + 1, y_k)$ 

$$p_{k+1} = p_k + 2\Delta y$$
  - Selain itu maka titik selanjutnya untuk digambar di :  $(x_k + 1, y_k + 1)$ 

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$
4. Ulangi langkah diatas sebanyak  $\Delta x$

## 2.4. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 06-P01	CPMK-01	Jelaskan tahapan pembangkitan garis dengan algoritma DDA!	50
2.	CPL 06-P01	CPMK-01	Jelaskan tahapan pembangkitan garis dengan algoritma Bresenham!	50

## 2.5. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. Library OpenGL

## 2.6. LANGKAH PRAKTIKUM

### PERSIAPAN

1. Buka Visual Studio C++ dan buat project baru dengan nama **praktikum02**.
2. Download kode dasar praktikum Grafika Komputer dan Library OpenGL seperti pada Praktikum 1.
3. Ubah nama dari kode dasar "**praktikum00.cpp**" menjadi "**praktikum02.cpp**" dan copy-kan ke **Source Files** di project yang anda buat.
4. Setting OpenGL library pada Visual Studio C/C++ seperti pada Praktikum 1.

## PRAKTIKUM

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 06-P01	CPMK-01	Selesaikan langkah praktikum 1 – 9	Hasil praktikum langkah 1 – 9	50
2.	CPL 06-P01	CPMK-01	Selesaikan langkah praktikum 10 – 14	Hasil praktikum langkah 10 - 14	50

### Langkah-Langkah:

1. Untuk menggambar garis yang berupa obyek 2D, proyeksi dari kamera perlu di ubah ke proyeksi Orthogonal.
2. Di fungsi **init()** dalam praktikum02.cpp, ubah baris kode berikut :

```
gluPerspective(45.0, 1.0, 1.0, 100.0);

menjadi

glOrtho((GLfloat)-SCREEN_WIDTH/2, (GLfloat)SCREEN_WIDTH/2,
        (GLfloat)-SCREEN_HEIGHT/2, (GLfloat)SCREEN_HEIGHT/2, 1.0, 100.0);
```

3. Di fungsi **reshape()** dalam praktikum02.cpp, ubah baris kode berikut :

```
gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

menjadi

glOrtho((GLfloat)-w/2, (GLfloat)w/2, (GLfloat)-h/2, (GLfloat)h/2, 1.0,
        100.0);
```

4. Di fungsi **init()** dalam praktikum02.cpp, ubah baris kode berikut untuk mengubah warna latar belakang pada layar menjadi warna hitam.

```
glClearColor(1.0, 1.0, 1.0, 0.0);

menjadi

glClearColor(0.0, 0.0, 0.0, 0.0);
```

5. Tambahkan fungsi **lineDDAX()** di file praktikum02.cpp anda untuk menggambar garis dengan algoritma DDA bila kenaikan terhadap X.

```
// fungsi untuk menggambar garis dengan algoritma DDA bila terhadap X
void lineDDAX(Vec3 point1, Vec3 point2)
{
    // hitung gradient garis m
    int dY = point2.Y - point1.Y;
    int dX = point2.X - point1.X;
```



```

float m = (float)dY / dX;
float im = 1.0f/m;

// mulai menggambar titik-titik
glBegin(GL_POINTS);
// koordinat titik awal
glVertex3f(point1.X, point1.Y, point1.Z);

float pX = point1.X, pY = point1.Y, pZ = point1.Z;
// kenaikan terhadap X
for (int i = point1.X; i < point2.X; i++)
{
    pX = pX + 1; // Xn+1 = Xn + 1
    pY = pY + m; // Yn+1 = Yn + m
    glVertex3f(pX, pY, pZ);
}
// koordinat titik akhir
glVertex3f(point2.X, point2.Y, point2.Z);
glEnd();
}

```

6. Tambahkan fungsi **lineDDAY()** di file praktikum02.cpp anda untuk menggambar garis dengan algoritma DDA bila kenaikan terhadap Y (copy paste dari fungsi lineDDAX() di langkah 9 dan ubah seperti fungsi dibawah ini).

```

// fungsi untuk menggambar garis dengan algoritma DDA bila terhadap Y
void lineDDAY(Vec3 point1, Vec3 point2)
{
    // hitung gradient garis m
    int dY = point2.Y - point1.Y;
    int dX = point2.X - point1.X;
    float m = (float)dY / dX;
    float im = 1.0f/m;

    // mulai menggambar titik-titik
    glBegin(GL_POINTS);
    // koordinat titik awal
    glVertex3f(point1.X, point1.Y, point1.Z);

    float pX = point1.X, pY = point1.Y, pZ = point1.Z;
    // kenaikan terhadap Y
    for (int i = point1.Y; i < point2.Y; i++)
    {
        pX = pX + im; // Xn+1 = Xn + 1/m
        pY = pY + 1; // Yn+1 = Yn + 1
        glVertex3f(pX, pY, pZ);
    }
    // koordinat titik akhir
    glVertex3f(point2.X, point2.Y, point2.Z);
    glEnd();
}

```

7. Tambahkan fungsi **lineDDA()** di file praktikum02.cpp anda untuk menggambar garis dengan algoritma DDA dengan memanggil fungsi yang anda buat di langkah 9 dan langkah 10.

```

// fungsi untuk menggambar garis dengan algoritma DDA
void lineDDA(Vec3 point1, Vec3 point2)
{
    // hitung selisih panjang
    int dY = point2.Y - point1.Y;
    int dX = point2.X - point1.X;
    // bila deltaY lebih pendek dari deltaX

```

```

if (abs(dY) < abs(dX))
{
    if (point1.X < point2.X) // bila X1 < X2
        lineDDAX(point1, point2);
    else // bila X1 > X2 maka dibalik
        lineDDAX(point2, point1);
}
else // bila deltaY lebih panjang dari deltaX
{
    if (point1.Y < point2.Y) // bila Y1 < Y2
        lineDDAY(point1, point2);
    else // bila Y1 > Y2 maka dibalik
        lineDDAY(point2, point1);
}
}

```

8. Ubah fungsi **drawObject()** di praktikum02.cpp menjadi seperti dibawah ini.

```

// fungsi untuk menggambar obyek
void drawObject()
{
    glPushMatrix();
    // operasi transformasi rotasi obyek ke arah kanan-kiri
    glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);
    glPushMatrix();
    // operasi transformasi rotasi obyek ke arah atas-bawah
    glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

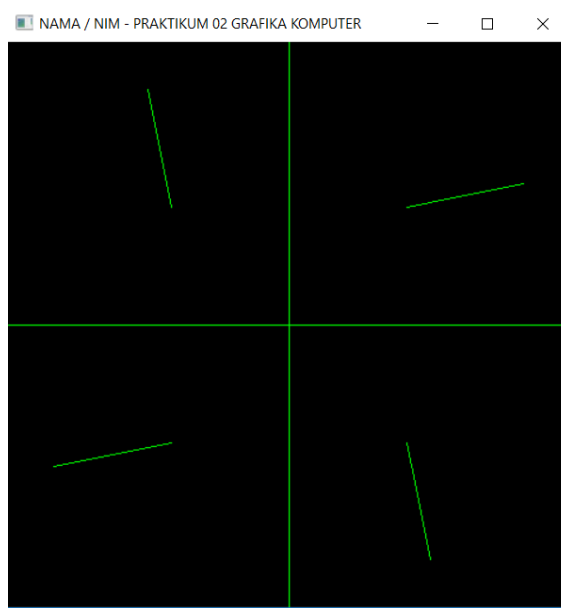
    // set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)
    glColor3f(0.0f, 1.0f, 0.0f);

    // gambar sumbu
    Vec3 sbY1 = Vec3( 0.0f, -300.0f, 0.0f);
    Vec3 sbY2 = Vec3( 0.0f, 300.0f, 0.0f);
    Vec3 sbX1 = Vec3(-300.0f, 0.0f, 0.0f);
    Vec3 sbX2 = Vec3( 300.0f, 0.0f, 0.0f);
    lineDDA(sbX1, sbX2);
    lineDDA(sbY1, sbY2);
    // kuadran 1
    Vec3 point1 = Vec3( 100.0f, 100.0f, 0.0f);
    Vec3 point2 = Vec3( 200.0f, 120.0f, 0.0f);
    lineDDA(point1, point2);
    // kuadran 2
    point1 = Vec3(-100.0f, 100.0f, 0.0f);
    point2 = Vec3(-120.0f, 200.0f, 0.0f);
    lineDDA(point1, point2);
    // kuadran 3
    point1 = Vec3(-100.0f, -100.0f, 0.0f);
    point2 = Vec3(-200.0f, -120.0f, 0.0f);
    lineDDA(point1, point2);
    // kuadran 4
    point1 = Vec3( 100.0f, -100.0f, 0.0f);
    point2 = Vec3( 120.0f, -200.0f, 0.0f);
    lineDDA(point1, point2);

    glPopMatrix();
    glPopMatrix();
}

```

9. Jalankan program untuk melihat hasil dari pembuatan garis dengan algoritma DDA seperti yang ditunjukkan pada Gambar 2.1.



Gambar 2.1 Hasil pembuatan garis dengan algoritma DDA.

10. Tambahkan fungsi **lineBresenhamX()** di file praktikum02.cpp anda untuk menggambar garis dengan algoritma Bresenham bila kenaikan terhadap X.

```
// fungsi untuk menggambar garis dengan algoritma Bresenham
// bila slopenya terhadap X
void lineBresenhamX(Vec3 point1, Vec3 point2)
{
    // hitung selisih panjang
    int dY = point2.Y - point1.Y;
    int dX = point2.X - point1.X;

    int yi = 1; // skala penambahan
    // bila delta Y kurang dari 0
    if (dY < 0)
    {
        yi = -1;
        dY = -dY;
    }

    // mulai menggambar titik-titik
    glBegin(GL_POINTS);
    // koordinat titik awal
    glVertex3f(point1.X, point1.Y, point1.Z);

    int pX = point1.X, pY = point1.Y, pZ = point1.Z;
    int dY2 = 2*dY; // hitung 2*deltaY
    int dX2 = 2*dX; // hitung 2*deltaX
    int pk = dY2 - dX; // hitung p0
    // kenaikan terhadap X
    for (int i = point1.X; i < point2.X; i++)
    {
        if (pk < 0) // bila p < 0
        {
            pk = pk + dY2; // update pk+1 = pk + 2dY
            pX = pX + 1; // Xn+1 = Xn + 1
            pY = pY; // Yn+1 = Yn
        }
        else // bila p >= 0
        {

```

```

        pk = pk + dY2 - dX2;    // update pk+1 = pk + 2dY - 2dX
        pX = pX + 1;           // Xn+1 = Xn + 1
        pY = pY + yi;          // Yn+1 = Yn + yi
    }
    glVertex3f(pX, pY, pZ);
}
// koordinat titik akhir
glVertex3f(point2.X, point2.Y, point2.Z);
glEnd();
}

```

11. Tambahkan fungsi **lineBresenhamY()** di file praktikum02.cpp anda untuk menggambar garis dengan algoritma Bresenham bila kenaikan terhadap Y (copy paste dari fungsi lineBresenhamX() di langkah 14 dan ubah seperti fungsi dibawah ini).

```

// fungsi untuk menggambar garis dengan algoritma Bresenham
// bila slopenya terhadap Y
void lineBresenhamY(Vec3 point1, Vec3 point2)
{
    // hitung selisih panjang
    int dY = point2.Y - point1.Y;
    int dX = point2.X - point1.X;

    int xi = 1; // skala penambahan
    // bila delta X kurang dari 0
    if (dX < 0)
    {
        xi = -1;
        dX = -dX;
    }

    // mulai menggambar titik-titik
    glBegin(GL_POINTS);
    // koordinat titik awal
    glVertex3f(point1.X, point1.Y, point1.Z);

    int pX = point1.X, pY = point1.Y, pZ = point1.Z;
    int dY2 = 2*dY; // hitung 2*deltaY
    int dX2 = 2*dX; // hitung 2*deltaX
    int pk = dX2 - dY; // hitung p0
    // kenaikan terhadap Y
    for (int i = point1.Y; i < point2.Y; i++)
    {
        if (pk < 0) // bila p < 0
        {
            pk = pk + dX2;    // update pk+1 = pk + 2dX
            pX = pX;          // Xn+1 = Xn
            pY = pY + 1;      // Yn+1 = Yn + 1
        }
        else // bila p >= 0
        {
            pk = pk + dX2 - dY2; // update pk+1 = pk + 2dX - 2dY
            pX = pX + xi;        // Xn+1 = Xn + xi
            pY = pY + 1;        // Yn+1 = Yn + 1
        }
        glVertex3f(pX, pY, pZ);
    }
    // koordinat titik akhir
    glVertex3f(point2.X, point2.Y, point2.Z);
    glEnd();
}

```

12. Tambahkan fungsi **lineBresenham()** di file praktikum02.cpp anda untuk menggambar garis dengan algoritma Bresenham dengan memanggil fungsi di langkah 14 dan langkah 15.

```
// fungsi untuk menggambar garis dengan algoritma Bresenham
void lineBresenham(Vec3 point1, Vec3 point2)
{
    // hitung selisih panjang
    int dY = point2.Y - point1.Y;
    int dX = point2.X - point1.X;
    if (abs(dY) < abs(dX)) // bila deltaY lebih pendek dari deltaX
    {
        if (point1.X < point2.X) // bila X1 < X2
            lineBresenhamX(point1, point2);
        else // bila X1 > X2 maka dibalik
            lineBresenhamX(point2, point1);
    }
    else // bila deltaY lebih panjang dari deltaX
    {
        if (point1.Y < point2.Y) // bila Y1 < Y2
            lineBresenhamY(point1, point2);
        else // bila Y1 > Y2 maka dibalik
            lineBresenhamY(point2, point1);
    }
}
```

13. Ubah fungsi **drawObject()** di praktikum02.cpp di baris kode berikut untuk menerapkan algoritma bresenham.

```
lineDDA(sbX1, sbX2);
lineDDA(sbY1, sbY2);

menjadi

lineBresenham(sbX1, sbX2);
lineBresenham(sbY1, sbY2);

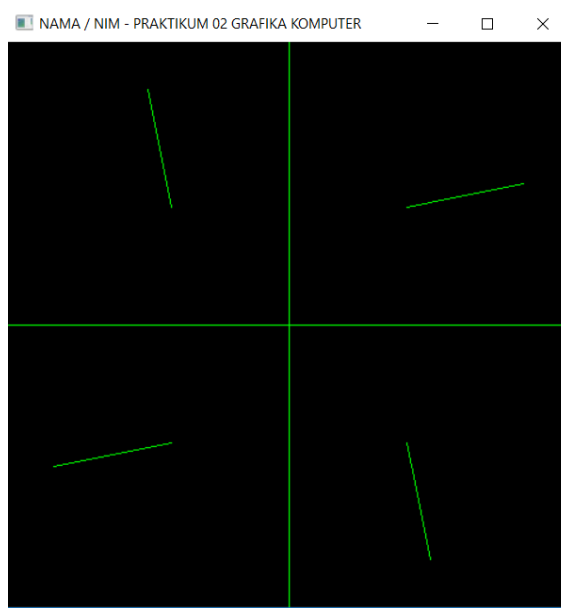
dan

lineDDA(point1, point2);

menjadi

lineBresenham(point1, point2);
```


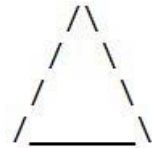
14. Jalankan program untuk melihat hasil dari pembuatan garis dengan algoritma Bresenham seperti yang ditunjukkan pada Gambar 2.2.



Gambar 2.2 Hasil pembuatan garis dengan algoritma Bresenham.

## 2.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 06-P01	CPMK-01	Buatlah bentuk diamond berwarna kuning dengan algoritma garis DDA! 	50
2.	CPL 06-P01	CPMK-01	Buatlah bentuk segitiga berwarna biru dengan algoritma garis Bresenham! 	50

## 2.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 06-P01	CPMK-01	20%		
2.	Praktik	CPL 06-P01	CPMK-01	30%		
3.	Post-Test	CPL 06-P01	CPMK-01	50%		
<b>Total Nilai</b>						

## PRAKTIKUM 3: INTERPOLASI DAN KURVA

Pertemuan ke : 3

Total Alokasi Waktu : 90 menit

- Materi : 10 menit
- Pre-Test : 10 menit
- Praktikum : 40 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 06-P01	Mampu merancang dan mengimplementasikan algoritma/metode dalam mengidentifikasi dan memecahkan masalah yang melibatkan perangkat lunak dan pemikiran komputasi
CPMK-01	Mampu menjelaskan sejarah perkembangan aplikasi komputer grafis dan menerapkan algoritma pembangkitan garis, interpolasi, dan kurva untuk membuat objek grafis menggunakan OpenGL

### 3.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan:

1. Mahasiswa mampu menjelaskan tentang konsep interpolasi.
2. Mahasiswa mampu menjelaskan tentang jenis-jenis interpolasi.
3. Mahasiswa mampu menjelaskan tentang kurva polynomial.
4. Mahasiswa mampu menerapkan interpolasi dan kurva dengan OpenGL.

### 3.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 06-P01	CPMK-01	Kemampuan mahasiswa dalam menerapkan algoritma interpolasi dan pembangkitan kurva menggunakan library OpenGL.
------------	---------	---

### 3.3. TEORI PENDUKUNG

Interpolasi digunakan untuk menaksir nilai antara (*intermediate value*) diantara titik-titik data. Nilai sela yang diberikan tergantung dari fungsi interpolasi. Kurva merupakan rentetan titik 1D yang berkelanjutan pada bidang 2D atau 3D. Kurva memiliki atribut warna, ketebalan, pola, dan bentuk. Representasi kurva yaitu eksplisit, implisit dan parametrik. Rentetan titik pada kurva dapat dibuat dengan interpolasi. Macam-macam interpolasi diantaranya:

#### Interpolasi Linear

Menggunakan fungsi linear untuk melakukan interpolasi. Bila terdapat dua titik yang akan diinterpolasi yaitu  $(x_0, y_0)$  sampai  $(x_1, y_1)$ . Bila jarak  $(x_0, y_0)$  sampai  $(x_1, y_1)$  dinormalisasi menjadi 1 (dinormalisasi) dan diketahui jarak awal  $(x_0, y_0)$  sampai titik sela  $(x, y)$  adalah  $u$  maka:

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

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

### Interpolasi cosine

Menggunakan fungsi cosine untuk melakukan interpolasi. Bila terdapat dua titik yang akan diinterpolasi yaitu  $(x_0, y_0)$  sampai  $(x_1, y_1)$  dan jarak tersebut dinormalisasi menjadi 1 sedangkan jarak titik sela  $(x, y)$  dengan titik awal adalah  $u$  maka persamaannya :

$$y = y_0 \cdot \left(1 - \left((1 - \cos(u\pi))/2\right)\right) + y_1 \cdot \left((1 - \cos(u\pi))/2\right)$$

### Interpolasi cubic

Menggunakan fungsi pangkat tiga / kubik untuk melakukan interpolasi. Interpolasi kubik memerlukan 2 titik tambahan di ujung 2 titik utama untuk interpolasi. 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 :

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

## 3.4. PRE-TEST

Jawablah pertanyaan berikut (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 06-P01	CPMK-01	Jelaskan tahapan interpolasi linear!	50
2.	CPL 06-P01	CPMK-01	Jelaskan tahapan interpolasi kubik!	50

## 3.5. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. Library OpenGL

## 3.6. LANGKAH PRAKTIKUM

### PERSIAPAN

1. Buka Visual Studio C++ dan buat project baru dengan nama **praktikum03**.
2. Download kode dasar praktikum Grafika Komputer dan Library OpenGL seperti pada Praktikum 1.
3. Ubah nama dari kode dasar "**praktikum00.cpp**" menjadi "**praktikum03.cpp**" dan copy-kan ke **Source Files** di project yang anda buat.



4. Setting OpenGL library pada Visual Studio C/C++ seperti pada Praktikum 1.

## PRAKTIKUM

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 06-P01	CPMK-01	Selesaikan langkah praktikum 1 – 11	Hasil praktikum langkah 1 – 11	50
2.	CPL 06-P01	CPMK-01	Selesaikan langkah praktikum 12 – 15	Hasil praktikum langkah 12 - 15	50

### Langkah-Langkah:

1. Untuk menggambar kurva yang berupa obyek 2D, proyeksi dari kamera perlu di ubah ke proyeksi Orthogonal.
2. Di fungsi **init()** dalam praktikum03.cpp, ubah baris kode berikut :

```
gluPerspective(45.0, 1.0, 1.0, 100.0);

menjadi

glOrtho((GLfloat)-SCREEN_WIDTH/2, (GLfloat)SCREEN_WIDTH/2,
        (GLfloat)-SCREEN_HEIGHT/2, (GLfloat)SCREEN_HEIGHT/2, 1.0, 100.0);
```

3. Di fungsi **reshape()** dalam praktikum03.cpp, ubah baris kode berikut :

```
gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

menjadi

glOrtho((GLfloat)-w/2, (GLfloat)w/2, (GLfloat)-h/2, (GLfloat)h/2, 1.0,
        100.0);
```

4. Di fungsi **init()** dalam praktikum03.cpp, ubah baris kode berikut untuk mengubah warna latar belakang pada layar menjadi warna hitam.

```
glClearColor(1.0, 1.0, 1.0, 0.0);

menjadi

glClearColor(0.0, 0.0, 0.0, 0.0);
```

5. Tambahkan di praktikum03.cpp, pilihan untuk memilih interpolasi mana yang akan digunakan.

```
// enumerate untuk tipe interpolation
enum INTERP_TYPE
{
    INTERP_POINTS = 0,
    INTERP_LINES = 1,
    INTERP_LINEAR = 2,
    INTERP_COSINE = 3,
    INTERP_CUBIC = 4
};
```

6. Tambahkan fungsi **linearInterpolate()** untuk melakukan interpolasi linear berikut ke dalam praktikum03.cpp.

```
// fungsi untuk melakukan interpolasi linear dari dua titik
float linearInterpolate(float y0, float y1, float u)
{
    return (y0 * (1 - u) + y1 * u);
}
```

7. Tambahkan fungsi **cosineInterpolate()** untuk melakukan interpolasi cosine berikut ke dalam praktikum03.cpp.

```
// fungsi untuk melakukan interpolasi cosine dari dua titik
float cosineInterpolate(float y0, float y1, float u)
{
    float cosineU = (1 - cos(u * PHI)) / 2;
    return (y0 * (1 - cosineU) + y1 * cosineU);
}
```

8. Tambahkan fungsi **cubicInterpolate()** untuk melakukan interpolasi cubic berikut ke dalam praktikum03.cpp.

```
// fungsi untuk melakukan interpolasi cubic dari dua titik
float cubicInterpolate(float y0, float y1, float y2, float y3, float u)
{
    float a = y3 - y2 - y0 + y1;
    float b = 2 * y0 - 2 * y1 - y3 + y2;
    float c = y2 - y0;
    float d = y1;

    return (a*u*u*u + b*u*u + c*u + d);
}
```

9. Tambahkan fungsi **drawInterpolation()** pada praktikum03.cpp untuk menggambar kurva antara 2 titik dengan interpolasi linear.

```
// gambar garis hasil interpolasi
// point1 adalah titik awal
// point2 adalah titik akhir
// n adalah jumlah titik yang dibuat
// type adalah tipe interpolasi yang digunakan
void drawInterpolation(
    Vec3 point0,
    Vec3 point1,
    Vec3 point2,
    Vec3 point3,
    int n,
    INTERP_TYPE type1,
    INTERP_TYPE type2)
{
    float u = 0;
    float stepU = 1.0f / n; // kenaikan u
    float stepX = fabs(point2.X - point1.X) / n; // kenaikan x
    float pX = point1.X, pY = point1.Y, pZ = point1.Z; // titik awal

    // mulai menggambar titik-titik
    glPointSize(5);
    // bila menggambar titik
    if (type1 == INTERP_POINTS)
        glBegin(GL_POINTS);
    // bila menggambar garis
```

```

else if (type1 == INTERP_LINES)
    glBegin(GL_LINES);
for (int i = 0; i < n; i++)
{
    glVertex3f(pX, pY, pZ);
    pX = pX + stepX;
    u = u + stepU;
    // bila interpolasi linear
    if (type2 == INTERP_LINEAR)
        pY = linearInterpolate(point1.Y, point2.Y, u);
    // bila interpolasi cosine
    else if (type2 == INTERP_COSINE)
        pY = cosineInterpolate(point1.Y, point2.Y, u);
    // bila interpolasi cubic
    else if (type2 == INTERP_CUBIC)
        pY = cubicInterpolate(point0.Y, point1.Y, point2.Y,
                               point3.Y, u);
    glVertex3f(pX, pY, pZ);
}
glEnd();
}

```

10. Ubah fungsi **drawObject()** untuk menggambar garis dengan interpolasi linear sebagai berikut.

```

// fungsi untuk menggambar obyek kubus
void drawObject()
{
    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah kanan-kiri
    glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah atas-bawah
    glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

    // set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)
    glColor3f(0.0f, 1.0f, 0.0f);

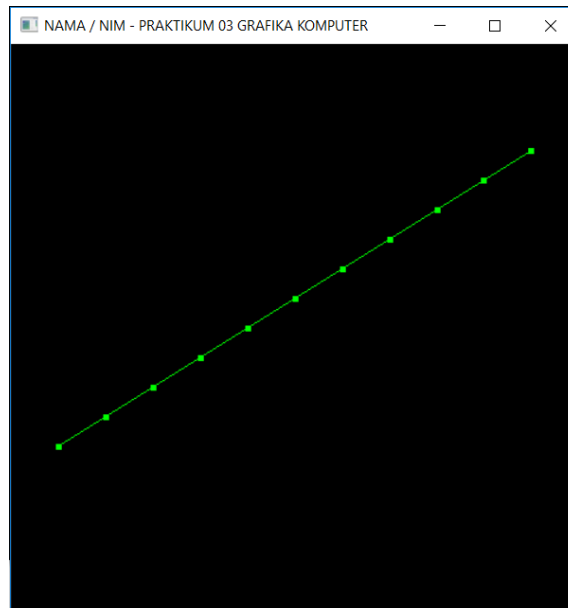
    // kuadran 1
    Vec3 point0 = Vec3(-300.0f, -200.0f, 0.0f);
    Vec3 point1 = Vec3(-200.0f, -100.0f, 0.0f);
    Vec3 point2 = Vec3( 200.0f,  150.0f, 0.0f);
    Vec3 point3 = Vec3( 300.0f,  250.0f, 0.0f);

    drawInterpolation(point0, point1, point2, point3, 10, INTERP_POINTS,
                      INTERP_LINEAR);
    drawInterpolation(point0, point1, point2, point3, 10, INTERP_LINES,
                      INTERP_LINEAR);

    glPopMatrix();
    glPopMatrix();
}

```

11. Jalankan program untuk mendapatkan hasil seperti yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Hasil penerapan interpolasi linear.

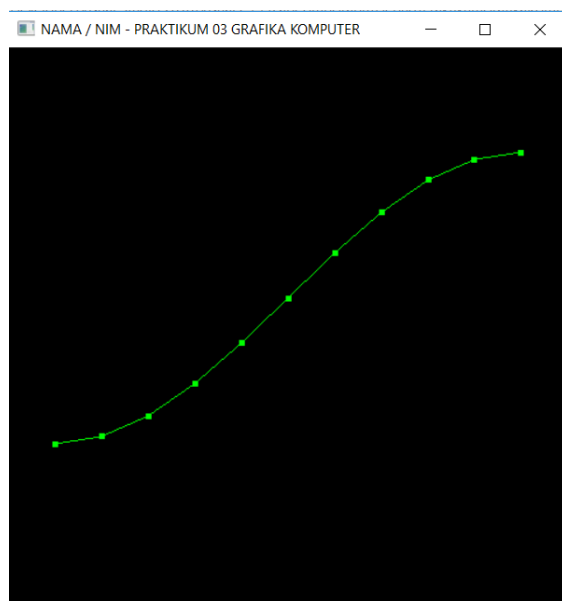
12. Ubah fungsi di **drawObject()** di baris kode berikut untuk menerapkan interpolasi cosine.

```
drawInterpolation(point0, point1, point2, point3, 10, INTERP_POINTS,
    INTERP_LINEAR);
drawInterpolation(point0, point1, point2, point3, 10, INTERP_LINES,
    INTERP_LINEAR);
```

menjadi

```
drawInterpolation(point0, point1, point2, point3, 10, INTERP_POINTS,
    INTERP_COSINE);
drawInterpolation(point0, point1, point2, point3, 10, INTERP_LINES,
    INTERP_COSINE);
```

13. Jalankan program untuk mendapatkan hasil seperti yang ditunjukkan pada Gambar 3.2.



Gambar 3.2 Hasil penerapan interpolasi cosine.

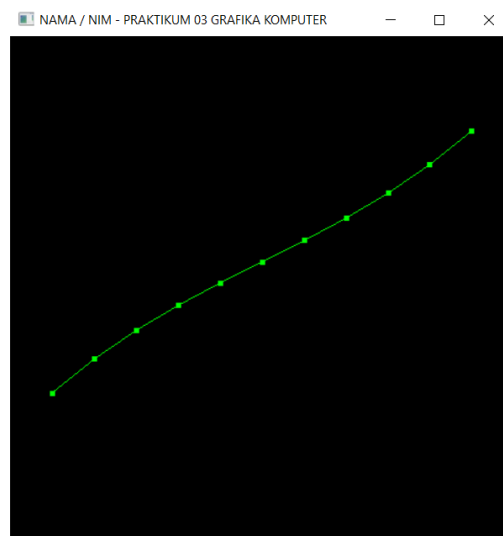
14. Ubah fungsi di **drawObject()** di baris kode berikut untuk menerapkan interpolasi cubic.

```
drawInterpolation(point0, point1, point2, point3, 10, INTERP_POINTS,
    INTERP_COSINE);
drawInterpolation(point0, point1, point2, point3, 10, INTERP_LINES,
    INTERP_COSINE);
```

menjadi

```
drawInterpolation(point0, point1, point2, point3, 10, INTERP_POINTS,
    INTERP_CUBIC);
drawInterpolation(point0, point1, point2, point3, 10, INTERP_LINES,
    INTERP_CUBIC);
```

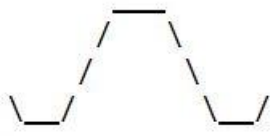
15. Jalankan program untuk mendapatkan hasil seperti yang ditunjukkan pada Gambar 3.3.



Gambar 3.3 Hasil penerapan interpolasi cubic.

### 3.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 06-P01	CPMK-01	Buatlah bentuk lembah dengan interpolasi cubic dengan jumlah titik n=10 seperti gambar dibawah ini! 	100

### 3.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 06-P01	CPMK-01	20%		
2.	Praktik	CPL 06-P01	CPMK-01	30%		
3.	Post-Test	CPL 06-P01	CPMK-01	50%		
<b>Total Nilai</b>						

## PRAKTIKUM 4: TRANSFORMASI 2D DAN 3D

Pertemuan ke : 4

Total Alokasi Waktu : 90 menit

- Materi : 10 menit
- Pre-Test : 10 menit
- Praktikum : 40 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 08-KK01	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPMK-02	Mampu menjelaskan tentang transformasi sistem koordinat, transformasi 2D dan 3D, viewing, clipping dan proyeksi serta menerapkannya menggunakan OpenGL

### 4.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan:

1. Mahasiswa mampu menjelaskan tentang konsep transformasi.
2. Mahasiswa mampu menjelaskan tentang perhitungan matriks transformasi 2D.
3. Mahasiswa mampu menjelaskan tentang perhitungan matriks transformasi 3D.
4. Mahasiswa mampu menerapkan transformasi 2D dan 3D dengan OpenGL.

### 4.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 08-KK01	CPMK-02	Kemampuan mahasiswa dalam menerapkan transformasi 2D dan 3D pada objek grafis menggunakan library OpenGL.
-------------	---------	---

### 4.3. TEORI PENDUKUNG

Transformasi berarti mengubah posisi. Transformasi dasar dalam komputer grafis diantaranya translasi, scaling, rotasi, shear untuk 2D dan 3D.

#### Translasi

Translasi berarti merubah posisi obyek dari koordinat yang satu ke koordinat yang lain. Dilakukan dengan menambahkan jarak translasi ( $t_x, t_y, t_z$ ) pada posisi awal ( $x, y, z$ ) untuk memindahkan benda ke posisi baru ( $x', y', z'$ ). Jarak translasi ( $t_x, t_y, t_z$ ) disebut juga vektor translasi atau vektor perpindahan. Rumus Translasi 3D:

$$x' = x + t_x$$

$$y' = y + t_y$$

$$z' = z + t_z$$

Atau bila direpresentasikan dengan matriks, translasi 3D dapat dirumuskan:

$$P' = T \cdot P$$

Apabila dituliskan dalam koordinat homogen:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

### Scaling

Scaling berarti mengubah ukuran obyek. Dilakukan dengan mengalikan factor skala ( $s_x, s_y, s_z$ ) pada posisi awal ( $x, y, z$ ) untuk menghasilkan ukuran baru di koordinat ( $x', y', z'$ ). Rumus Scaling 3D:

$$x' = x \cdot s_x$$

$$y' = y \cdot s_y$$

$$z' = z \cdot s_z$$

Bila direpresentasikan dengan matriks, scaling 3D dapat dirumuskan:

$$P' = S \cdot P$$

Apabila dituliskan dalam koordinat homogen:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

### Rotasi

Rotasi berarti mengubah posisi terhadap jalur melingkar pada bidang x-y-z. Dilakukan dengan menentukan sudut rotasi  $\theta$  dan titik rotasi (rotation point / pivot point) ( $x, y, z$ ) untuk menghasilkan posisi baru pada koordinat ( $x', y', z'$ ). Bila  $> 0$  : rotasi berlawanan jarum jam. Bila  $< 0$  : rotasi searah jarum jam.

Rumus rotasi terhadap sumbu-Z:

$$x' = x \cos \theta - y \sin \theta$$

$$y' = x \sin \theta + y \cos \theta$$

$$z' = z$$

Atau  $P' = R_z(\theta) \cdot P$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rumus rotasi terhadap sumbu-X:

$$x' = x$$

$$y' = y \cos \theta - z \sin \theta$$

$$z' = y \sin \theta + z \cos \theta$$

Atau  $P' = R_x(\theta) \cdot P$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rumus rotasi terhadap sumbu-Y:

$$x' = z \sin \theta + x \cos \theta$$

$$y' = y$$

$$z' = z \cos \theta - x \sin \theta$$

Atau  $P' = R_y(\theta) \cdot P$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

#### 4.4. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 08-KK01	CPMK-02	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? 1. Garis P ditranslasi sejauh T (2, 2)! 2. Garis P discaling sebesar S (2, 1)! 3. Garis P dirotasi sejauh 30 derajat!	100

#### 4.5. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. Library OpenGL

#### 4.6. LANGKAH PRAKTIKUM

##### PERSIAPAN

1. Buka Visual Studio C++ dan buat project baru dengan nama **praktikum04**.
2. Download kode dasar praktikum Grafika Komputer dan Library OpenGL seperti pada Praktikum 1.
3. Ubah nama dari kode dasar "**praktikum00.cpp**" menjadi "**praktikum04.cpp**" dan copy-kan ke **Source Files** di project yang anda buat.
4. Setting OpenGL library pada Visual Studio C/C++ seperti pada Praktikum 1.

##### PRAKTIKUM

Aturan Penilaian (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 08-KK01	CPMK-02	Selesaikan langkah praktikum 1 – 5	Hasil praktikum langkah 1 – 5	50
2.	CPL 08-KK01	CPMK-02	Selesaikan langkah praktikum 6 – 7	Hasil praktikum langkah 6 - 7	50

##### Langkah-Langkah:

1. Tambahkan variable berikut di dalam praktikum04.cpp untuk translasi, rotasi dan scaling (hanya tambahkan variable yang belum tertulis di kode dasar).

```
// inisialisasi variabel untuk transformasi seperti translasi, rotasi
atau scaling
float angle = 0.0f;           // sudut transformasi kamera
float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X
```



```

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y
float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f;      // sudut transformasi obyek di sumbu X
float objectAngleY = 0.0f;      // sudut transformasi obyek di sumbu Y
float objectAngleZ = 0.0f;      // sudut transformasi obyek di sumbu Z

float objectScaleX = 1.0f; // skala perbesaran obyek ke arah X
float objectScaleY = 1.0f; // skala perbesaran obyek ke arah Y
float objectScaleZ = 1.0f; // skala perbesaran obyek ke arah Z

float objectPositionX = 0.0f; // posisi obyek di sumbu X
float objectPositionY = 0.0f; // posisi obyek di sumbu Y
float objectPositionZ = 0.0f; // posisi obyek di sumbu Z

```

2. Ubah fungsi **drawObject()** pada praktikum04.cpp seperti dibawah ini.

```

// fungsi untuk menggambar obyek kubus
void drawObject()
{
    glPushMatrix();

    // operasi transformasi translasi obyek
    // ke arah sumbu X, Y atau Z
    glTranslatef(objectPositionX, objectPositionY, objectPositionZ);

    // operasi transformasi scaling obyek
    // memperbesar atau mengecilkan obyek
    // ke arah sumbu X, Y atau Z
    glScalef(objectScaleX, objectScaleY, objectScaleZ);

    // operasi transformasi rotasi obyek ke arah kanan-kiri
    glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah atas-bawah
    glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

    // set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)
    glColor3f(0.0f, 1.0f, 0.0f);

    glutSolidCube(1.0f); // menggambar obyek kubus

    glPopMatrix();

    glPopMatrix();
}

```

3. Tambahkan fungsi **keyboard1()** berikut ke dalam praktikum04.cpp untuk menerapkan operasi translasi dan scaling pada obyek.

```

// fungsi untuk mengatur masukan dari keyboard
void keyboard1(unsigned char key, int x, int y)
{
    float fraction = 0.5f;

    switch (key)
    {
        case 'w': // bila tombol 'w' pada keyboard ditekan
            // translasi ke atas
            objectPositionY += fraction;
    }
}

```

```

        glutPostRedisplay();
        break;
    case 's': // bila tombol 's' pada keyboard ditekan
        // translasi ke bawah
        objectPositionY -= fraction;
        glutPostRedisplay();
        break;
    case 'a': // bila tombol 'a' pada keyboard ditekan
        // translasi ke kiri
        objectPositionX -= fraction;
        glutPostRedisplay();
        break;
    case 'd': // bila tombol 'd' pada keyboard ditekan
        // translasi ke kanan
        objectPositionX += fraction;
        glutPostRedisplay();
        break;
    case 'q': // bila tombol 'q' pada keyboard ditekan
        // translasi ke depan
        objectPositionZ += fraction;
        glutPostRedisplay();
        break;
    case 'e': // bila tombol 'e' pada keyboard ditekan
        // translasi ke belakang
        objectPositionZ -= fraction;
        glutPostRedisplay();
        break;
    case 't': // bila tombol 't' pada keyboard ditekan
        // perbesar ke arah sumbu Y
        objectScaleY += 0.1f;
        glutPostRedisplay();
        break;
    case 'g': // bila tombol 'g' pada keyboard ditekan
        // perkecil ke arah sumbu Y
        objectScaleY = max(objectScaleY - 0.1f, 1.0f);
        glutPostRedisplay();
        break;
    case 'f': // bila tombol 'f' pada keyboard ditekan
        // perbesar ke arah sumbu X
        objectScaleX += 0.1f;
        glutPostRedisplay();
        break;
    case 'h': // bila tombol 'h' pada keyboard ditekan
        // perkecil ke arah sumbu X
        objectScaleX = max(objectScaleX - 0.1f, 1.0f);
        glutPostRedisplay();
        break;
    case 'r': // bila tombol 'r' pada keyboard ditekan
        // perbesar ke arah sumbu Z
        objectScaleZ += 0.1f;
        glutPostRedisplay();
        break;
    case 'y': // bila tombol 'y' pada keyboard ditekan
        // perkecil ke arah sumbu Z
        objectScaleZ = max(objectScaleZ - 0.1f, 1.0f);
        glutPostRedisplay();
        break;
    case 27: // bila tombol 'esc' pada keyboard ditekan
        // keluar program
        exit(0);
        break;
}
}

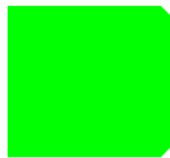
```

4. Tambahkan baris kode berikut ke fungsi **main()** tepat dibawah baris kode **glutSpecialFunc()**.

```
glutKeyboardFunc(keyboard); // keyboard
```

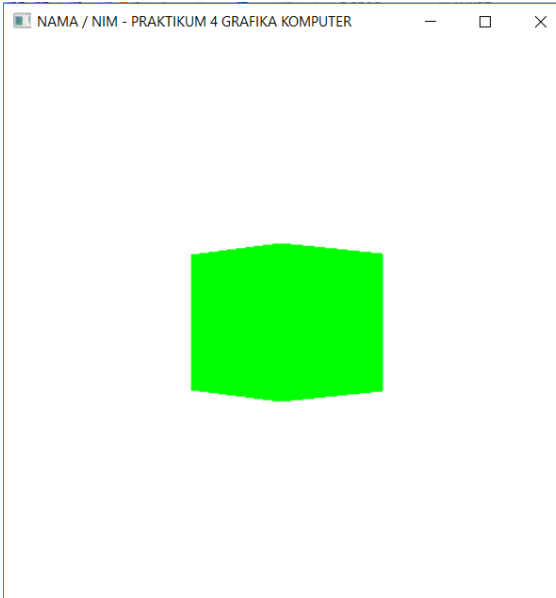
5. Jalankan program untuk melakukan translasi pada obyek dengan menekan tombol w, s, d, a serta tombol q dan e untuk translasi depan dan belakang. Contoh pada Gambar 4.1 merupakan translasi ke kiri.

NAMA / NIM - PRAKTIKUM 4 GRAFIKA KOMPUTER



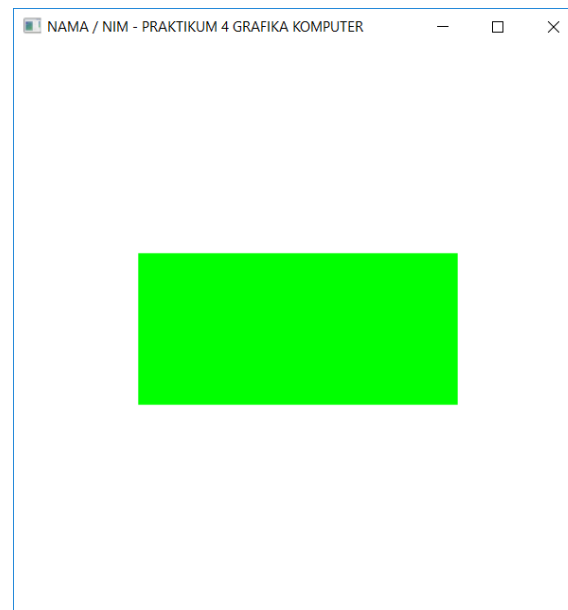
Gambar 4.1 Hasil penerapan translasi pada obyek.

6. Gunakan tombol arah untuk melakukan rotasi. Contoh Gambar 4.2 merupakan rotasi ke arah sumbu Y.



Gambar 4.2 Hasil penerapan rotasi terhadap obyek.

7. Gunakan tombol t, g, f, h, r dan y untuk melakukan scaling. Contoh Gambar 4.3 merupakan scaling ke sumbu X.



Gambar 4.3 Hasil penerapan scaling pada obyek.

#### 4.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 08-KK01	CPMK-02	Buatlah obyek persegi berwarna cyan kemudian terapkan transformasi translasi, rotasi dan scaling! Screenshot hasilnya!	100

#### 4.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 08-KK01	CPMK-02	20%		
2.	Praktik	CPL 08-KK01	CPMK-02	30%		
3.	Post-Test	CPL 08-KK01	CPMK-02	50%		
<b>Total Nilai</b>						

## PRAKTIKUM 5: PROYEKSI 3D

Pertemuan ke : 5

Total Alokasi Waktu : 90 menit

- Materi : 10 menit
- Pre-Test : 10 menit
- Praktikum : 40 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 08-KK01	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPMK-02	Mampu menjelaskan tentang transformasi sistem koordinat, transformasi 2D dan 3D, viewing, clipping dan proyeksi serta menerapkannya menggunakan OpenGL

### 5.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan:

1. Mahasiswa mampu menjelaskan tentang 3D viewing.
2. Mahasiswa mampu menjelaskan tentang proyeksi parallel.
3. Mahasiswa mampu menjelaskan tentang proyeksi perspektif.
4. Mahasiswa mampu menerapkan 3D viewing dan proyeksi dengan OpenGL.

### 5.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 08-KK01	CPMK-02	Kemampuan mahasiswa dalam menerapkan proyeksi orthogonal dan perspektif pada objek grafis menggunakan library OpenGL
-------------	---------	--

### 5.3. TEORI PENDUKUNG

Proyeksi dilakukan setelah konversi dari world coordinates ke viewing coordinates kemudian dipetakan ke koordinat proyeksi. Proyeksi obyek ditentukan dari perpotongan garis-garis proyeksi dengan bidang pandang. Jenis-jenis proyeksi diantaranya:

#### Proyeksi paralel

Proyeksi bila posisi koordinat obyek ditransformasikan ke bidang pandang dengan garis-garis paralel. Proyeksi paralel menggunakan vektor proyeksi sebagai arah dari garis-garis proyeksi. Bila proyeksi paralel tegak lurus terhadap bidang pandang disebut proyeksi paralel orthografik. Bila tidak tegak lurus disebut proyeksi paralel miring (oblique parallel projection).

Pada proyeksi orthografik bila bidang pandang ada di posisi  $z_{vp}$  sejajar dengan sumbu  $z_v$  maka setiap titik  $(x, y, z)$  pada koordinat pandang (viewing coordinates) ditransformasi ke koordinat proyeksi (projection coordinates) dengan rumus:

$$x_p = x, \text{ dan } y_p = y$$

Sedangkan koordinat  $z$  tetap sebagai kesan kedalaman.

Pada proyeksi miring bila ada dua sudut  $\alpha$  dan  $\varphi$  dan panjang garis yang ditarik dari  $(x, y, z)$  ke  $(x_p, y_p)$  adalah  $L$  maka :

$$\begin{aligned} x_p &= x + L \cos \varphi \\ y_p &= y + L \sin \varphi \end{aligned}$$

Garis  $L$  tergantung dari sudut  $\alpha$  dan sumbu  $Z$

$$\begin{aligned} \tan \alpha &= \frac{z}{L} \\ L &= \frac{z}{\tan \alpha} = zL_1 \end{aligned}$$

Dimana  $L_1$  adalah  $\tan^{-1} \alpha$ . Sehingga dapat dituliskan:

$$\begin{aligned} x_p &= x + z(L_1 \cos \varphi) \\ y_p &= y + z(L_1 \sin \varphi) \end{aligned}$$

Matriks proyeksi parallel adalah sebagai berikut:

$$M_{parallel} = \begin{bmatrix} 1 & 0 & L_1 \cos \varphi & 0 \\ 0 & 1 & L_1 \sin \varphi & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Untuk proyeksi orthografik :  $L_1 = 0$

Untuk proyeksi miring (oblique projection) :  $L_1 > 0$

### Proyeksi perspektif

Proyeksi bila posisi koordinat obyek ditransformasikan ke bidang pandang dengan garis-garis yang memusat di sebuah titik yang disebut titik referensi proyeksi atau center of projection (COP). Dibuat dengan menarik garis proyeksi yang bertemu pada titik referensi. Bila titik referensi ada di posisi  $z_{prp}$  pada sumbu  $z_v$  dan letak bidang pandang pada  $z_{vp}$  maka garis proyeksi :

$$\begin{aligned} x' &= x - xu \\ y' &= y - yu \\ z' &= z - (z - z_{prp})u \end{aligned}$$

Dengan  $u = 0 \dots 1$  dan koordinat  $(x', y', z')$  adalah koordinat titik di garis proyeksi

Saat  $u = 0$  maka ada di titik awal di  $P = (x, y, z)$

Saat  $u = 1$  maka ada di titik referensi di  $P = (0, 0, z_{prp})$

Bila matriks proyeksi perspektif adalah:

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{vp}/d_p & z_{vp}(z_{prp}/d_p) \\ 0 & 0 & -1/d_p & z_{prp}/d_p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Dan faktor homogenya:

$$h = \frac{z_{prp} - z}{d_p}$$

Maka koordinat proyeksi di bidang pandang:

$$\begin{aligned} x_p &= x_h/h \\ y_p &= y_h/h \end{aligned}$$

## 5.4. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 08-KK01	CPMK-02	Jelaskan yang dimaksud dengan proyeksi orthogonal!	30
2.	CPL 08-KK01	CPMK-02	Jelaskan yang dimaksud dengan proyeksi perspektif!	30
3.	CPL 08-KK01	CPMK-02	Jelaskan perbedaan antara proyeksi orthogonal dengan perspektif!	40

## 5.5. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. Library OpenGL

## 5.6. LANGKAH PRAKTIKUM

### PERSIAPAN

1. Buka Visual Studio C++ dan buat project baru dengan nama **praktikum05**.
2. Download kode dasar praktikum Grafika Komputer dan Library OpenGL seperti pada Praktikum 1.
3. Ubah nama dari kode dasar "**praktikum00.cpp**" menjadi "**praktikum05.cpp**" dan copy-kan ke **Source Files** di project yang anda buat.
4. Setting OpenGL library pada Visual Studio C/C++ seperti pada Praktikum 1.

### PRAKTIKUM

Aturan Penilaian (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 08-KK01	CPMK-02	Selesaikan langkah praktikum 1 – 5	Hasil praktikum langkah 1 – 5	50
2.	CPL 08-KK01	CPMK-02	Selesaikan langkah praktikum 6 – 12	Hasil praktikum langkah 6 - 12	50

### Langkah-Langkah:

1. Kode dasar praktikum tersebut sudah diatur untuk memakai proyeksi perspektif yang dapat anda lihat dari fungsi **gluPerspective()** di fungsi **init()** dan fungsi **reshape()**.
2. Tambahkan variable untuk pencahayaan di praktikum05.cpp.

```
// posisi sumber cahaya
// posisi sumber cahaya utk perspektif
float position[] = {0.0f, 5.0f, 5.0f, 1.0f};
// posisi sumber cahaya utk orthogonal
//float position[] = {0.0f, 100.0f, -100.0f, 1.0f};
```

3. Tambahkan kode berikut di fungsi **init()** untuk mengaktifkan pencahayaan.

```
// aktifkan pencahayaan
glEnable(GL_LIGHTING);
```

```
glEnable(GL_COLOR_MATERIAL);
glEnable(GL_LIGHT0);
```

4. Jalankan program untuk mendapatkan tampilan proyeksi perspektif pada kubus seperti yang ditunjukkan pada Gambar 5.1.
5. Coba lakukan rotasi dengan tombol arah.

NAMA / NIM - PRAKTIKUM 05 GRAFIKA KOMPUTER



Gambar 5.1 Hasil proyeksi perspektif pada obyek kubus.

6. Ubah variable pencahayaan di langkah 6 menjadi.

```
// posisi sumber cahaya
// posisi sumber cahaya utk perspektif
//float position[] = {0.0f,5.0f,5.0f,1.0f};
// posisi sumber cahaya utk orthogonal
float position[] = {0.0f,100.0f,-100.0f,1.0f};
```

7. Di fungsi **init()** dalam praktikum05.cpp, ubah baris kode berikut :

```
gluPerspective(45.0, 1.0, 1.0, 100.0);
```

menjadi

```
glOrtho((GLfloat)-SCREEN_WIDTH/2, (GLfloat)SCREEN_WIDTH/2,
        (GLfloat)-SCREEN_HEIGHT/2, (GLfloat)SCREEN_HEIGHT/2, 1.0, 100.0);
```

8. Di fungsi **reshape()** dalam praktikum05.cpp, ubah baris kode berikut :

```
gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);
```

menjadi

```
glOrtho((GLfloat)-w/2, (GLfloat)w/2, (GLfloat)-h/2, (GLfloat)h/2, 1.0,
        100.0);
```

9. Ubah kode di bawah ini di fungsi **drawObject()** untuk menampilkan obyek kubus di proyeksi orthogonal. Besar kubus perlu diubah karena konversi unit di proyeksi perspektif dan orthogonal berbeda.



```
glutSolidCube(1.0f); // menggambar obyek kubus di proyeksi perspektif
menjadi
glutSolidCube(50.0f); // menggambar obyek kubus di proyeksi orthogonal
```

10. Jalankan program untuk mendapatkan tampilan proyeksi perspektif pada kubus seperti yang ditunjukkan pada Gambar 5.2.
11. Coba lakukan rotasi dengan tombol arah.
12. Bedakan hasilnya dengan proyeksi perspektif.

NAMA / NIM - PRAKTIKUM 05 GRAFIKA KOMPUTER



Gambar 5.2 Hasil proyeksi orthogonal terhadap obyek kubus.

### 5.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 08-KK01	CPMK-02	Buatlah 3 obyek teapot berwarna merah, hijau dan biru lalu dengan menggunakan proyeksi orthogonal ubah masing-masing teapot tersebut ke tampilan sisi depan (merah), samping (hijau), dan atas (biru)!!	100

### 5.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 08-KK01	CPMK-02	20%		
2.	Praktik	CPL 08-KK01	CPMK-02	30%		
3.	Post-Test	CPL 08-KK01	CPMK-02	50%		
<b>Total Nilai</b>						



## PRAKTIKUM 6: REPRESENTASI OBYEK 3D

Pertemuan ke : 6

Total Alokasi Waktu : 90 menit

- Materi : 10 menit
- Pre-Test : 10 menit
- Praktikum : 40 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 08-KK01	Mampu memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah
CPMK-03	Mampu menjelaskan konsep representasi objek 3D, representasi kurva spline, representasi permukaan, dan menerapkannya menggunakan OpenGL

### 6.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Mahasiswa mampu menjelaskan tentang konsep representasi obyek 3D.
2. Mahasiswa mampu menjelaskan tentang teknik representasi obyek 3D.
3. Mahasiswa mampu menerapkan teknik representasi obyek 3D dengan OpenGL.

### 6.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

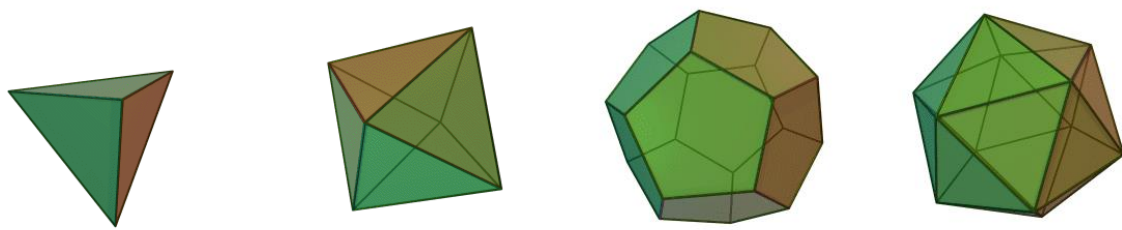
Indikator ketercapaian diukur dengan:

CPL 08-KK01	CPMK-03	Kemampuan mahasiswa dalam menerapkan konsep representasi obyek 3D dengan OpenGL.
-------------	---------	--

### 6.3. TEORI PENDUKUNG

Representasi obyek 3D digunakan untuk memodelkan bentuk-bentuk obyek di dunia nyata. Beberapa teknik representasi obyek 3D yaitu model wireframe, sweep representation, boundary representation, spatial partitioning representation, constructive solid geometry, dan sebagainya. Pada boundary representation obyek dideskripsikan dari batasan permukaannya yang membedakan obyek bagian dalam dan bagian luar. Hal ini dilakukan dengan menambahkan sisi pada jaring-jaring model obyek.

Sisi dapat berbentuk datar atau melengkung. Sisi biasanya berbentuk polygon atau quadric. Contoh obyek dengan sisi polygon adalah kubus dan polyhedron sedangkan contoh obyek dengan sisi quadric adalah tabung dan bola. Jenis polyhedron diantaranya ditunjukkan pada Gambar 6.1.



Tetrahedron

Octahedron

Dodecahedron

Icosahedron

Gambar 6.1 Bentuk polyhedron

#### 6.4. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 08-KK01	CPMK-03	Sebutkan metode representasi obyek 3D yang anda ketahui!	30
2.	CPL 08-KK01	CPMK-03	Jelaskan setiap metode representasi obyek 3D yang anda sebutkan di soal nomor 1!	70

#### 6.5. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.

#### 6.6. LANGKAH PRAKTIKUM

##### PERSIAPAN

1. Buka Visual Studio dan buat project baru dengan nama **praktikum06**.
2. Download kode dasar praktikum Grafika Komputer dan Library OpenGL seperti pada Praktikum 1.
3. Ubah nama dari kode dasar "**praktikum00.cpp**" menjadi "**praktikum06.cpp**" dan copy-kan ke Source Files di project yang anda buat.
4. Setting OpenGL library pada Visual Studio C/C++ seperti pada Praktikum 1.

##### PRAKTIKUM

Aturan Penilaian (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 08-KK01	CPMK-03	Selesaikan langkah praktikum 1 – 6	Hasil praktikum langkah 1 – 6	50
2.	CPL 08-KK01	CPMK-03	Selesaikan langkah praktikum 7 – 13	Hasil praktikum langkah 7 - 13	50

##### Langkah-Langkah:

1. Tambahkan kode berikut di fungsi **init()** untuk mengaktifkan pencahayaan.

```
// aktifkan pencahayaan
glEnable(GL_LIGHTING);
```

```
glEnable(GL_COLOR_MATERIAL);
glEnable(GL_LIGHT0);
```

2. Tambahkan fungsi **drawCube()** pada praktikum06.cpp untuk menggambar kubus.

```
// fungsi untuk menggambar kubus
void drawCube()
{
    glBegin(GL_QUADS);
    // beri warna merah di sisi depan
    glColor3f(1.0f, 0.0f, 0.0f);
    // buat sisi depan
    glVertex3f(-1.0f, -1.0f, 1.0f);
    glVertex3f( 1.0f, -1.0f, 1.0f);
    glVertex3f( 1.0f,  1.0f, 1.0f);
    glVertex3f(-1.0f,  1.0f, 1.0f);
    // beri warna hijau di sisi belakang
    glColor3f(0.0f, 1.0f, 0.0f);
    // buat sisi belakang
    glVertex3f(-1.0f, -1.0f,-1.0f);
    glVertex3f( 1.0f, -1.0f,-1.0f);
    glVertex3f( 1.0f,  1.0f,-1.0f);
    glVertex3f(-1.0f,  1.0f,-1.0f);
    // beri warna biru di sisi kiri
    glColor3f(0.0f, 0.0f, 1.0f);
    // buat sisi kiri
    glVertex3f(-1.0f, -1.0f, 1.0f);
    glVertex3f(-1.0f, -1.0f,-1.0f);
    glVertex3f(-1.0f,  1.0f,-1.0f);
    glVertex3f(-1.0f,  1.0f, 1.0f);
    // beri warna cyan di sisi kanan
    glColor3f(0.0f, 1.0f, 1.0f);
    // buat sisi kanan
    glVertex3f( 1.0f, -1.0f, 1.0f);
    glVertex3f( 1.0f, -1.0f,-1.0f);
    glVertex3f( 1.0f,  1.0f,-1.0f);
    glVertex3f( 1.0f,  1.0f, 1.0f);
    // beri warna kuning di sisi atas
    glColor3f(1.0f, 1.0f, 0.0f);
    // buat sisi atas
    glVertex3f(-1.0f,  1.0f, 1.0f);
    glVertex3f( 1.0f,  1.0f, 1.0f);
    glVertex3f( 1.0f,  1.0f,-1.0f);
    glVertex3f(-1.0f,  1.0f,-1.0f);
    // beri warna magenta di sisi bawah
    glColor3f(1.0f, 0.0f, 1.0f);
    // buat sisi bawah
    glVertex3f(-1.0f, -1.0f, 1.0f);
    glVertex3f( 1.0f, -1.0f, 1.0f);
    glVertex3f( 1.0f, -1.0f,-1.0f);
    glVertex3f(-1.0f, -1.0f,-1.0f);
    glEnd();
}
```

3. Tambahkan fungsi **drawCylinder()** pada praktikum06.cpp untuk menggambar tabung.

```
// fungsi untuk menggambar silinder
void drawCylinder(float radius, float height, int slices, int stacks)
{
    glPushMatrix();

    GLUQuadricObj* cyl = gluNewQuadric();
    gluQuadricDrawStyle(cyl, GLU_FILL);
```

```

gluQuadricNormals(cyl, GLU_SMOOTH);
gluQuadricOrientation(cyl, GLU_INSIDE);

// buat tutup atas silinder
glTranslatef(0.0f, -height/2, 0.0f);
glRotatef(-90, 1.0f, 0.0f, 0.0f);
glColor3f(1.0f, 1.0f, 0.0f); // warna kuning
gluDisk(cyl, 0.0f, radius, slices, stacks);
// buat badan silinder
glColor3f(1.0f, 0.0f, 0.0f); // warna merah
gluCylinder(cyl, radius, radius, height, slices, stacks);
// buat tutup bawah silinder
glColor3f(1.0f, 1.0f, 0.0f); // warna kuning
glTranslatef(0.0f, 0.0f, height);
gluDisk(cyl, 0.0f, radius, slices, stacks);

glPopMatrix();
}

```

4. Tambahkan fungsi **drawSphere()** pada praktikum06.cpp untuk menggambar bola.

```

// fungsi untuk menggambar bola
void drawSphere(float radius, int slices, int stacks)
{
    glPushMatrix();

    glColor3f(1.0f, 0.0f, 0.0f); // warna merah
    GLUQuadric *sphere = gluNewQuadric();
    gluQuadricDrawStyle(sphere, GLU_FILL);
    gluQuadricNormals(sphere, GLU_SMOOTH);
    gluSphere(sphere, radius, slices, stacks);

    glPopMatrix();
}

```

5. Ubah kode di fungsi **drawObject()** untuk menampilkan obyek kubus seperti berikut.

```

glutSolidCube(1.0f);

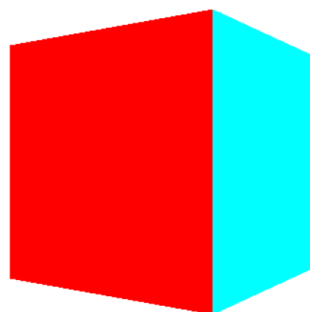
menjadi

drawCube(); // panggil fungsi untuk membuat obyek kubus

```

6. Jalankan program untuk menampilkan hasil seperti yang ditunjukkan pada Gambar 6.2.

NAMA / NIM - PRAKTIKUM 06 GRAFIKA KOMPUTER



Gambar 6.2 Hasil menggambar obyek kubus

7. Ubah fungsi **drawObject()** pada baris kode berikut untuk menampilkan obyek tabung.

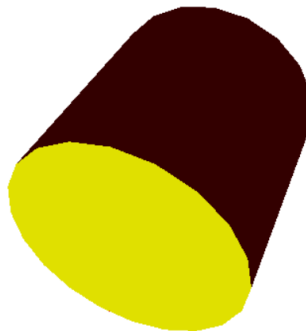
```
drawCube(); // panggil fungsi untuk membuat obyek kubus
```

Menjadi

```
drawCylinder(1.0f, 2.0f, 20, 20); // fungsi untuk membuat obyek silinder
```

8. Jalankan program untuk menampilkan hasil seperti yang ditunjukkan pada Gambar 6.3.

NAMA / NIM - PRAKTIKUM 06 GRAFIKA KOMPUTER



Gambar 6.3 Hasil menggambar obyek tabung

9. Ubah fungsi **drawObject()** pada baris kode berikut untuk menampilkan obyek bola.

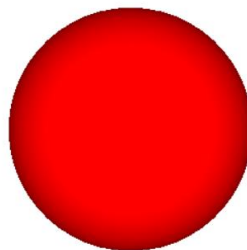
```
drawCylinder(1.0f, 2.0f, 20, 20); // fungsi untuk membuat obyek silinder
```

Menjadi

```
drawSphere(1.0f, 50, 50); // fungsi untuk membuat obyek bola
```

10. Jalankan program untuk menampilkan hasil seperti yang ditunjukkan pada Gambar 6.4.

NAMA / NIM - PRAKTIKUM 06 GRAFIKA KOMPUTER



Gambar 6.4 Hasil menggambar obyek bola.

11. Ubah fungsi **drawObject()** pada baris kode berikut untuk menampilkan obyek-obyek yang lain yang sudah ada dalam GLUT.

```
// fungsi untuk membuat obyek bola  
drawSphere(1.0f, 50, 50);
```

Menjadi salah satu obyek berikut (hilangkan tanda komentarnya di salah satu baris fungsi obyek berikut untuk menampilkan obyeknya)

```
// membuat obyek polyhedron
//glutSolidTetrahedron();
//glutSolidOctahedron();
//glutSolidDodecahedron();
//glutSolidIcosahedron();
//glutSolidCube(1.0f);
//glutSolidCone(1.0f, 1.0f, 50, 50);
//glutSolidSphere(1.0f, 50, 50);
//glutSolidTeapot(1.0f);
//glutSolidTorus(0.5f, 1.0f, 20, 20);
```

12. Jalankan program untuk menampilkan hasilnya.
13. Untuk menampilkan representasi wireframe pada obyek pada langkah 15 cukup mengganti "Solid" dengan "Wire". Misalnya glutWireTeapot(1.0f).

## 6.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 08-KK01	CPMK-03	Buatlah obyek piramid berwarna merah menggunakan gluCylinder seperti pada contoh membuat obyek silinder!	100

## 6.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 08-KK01	CPMK-03	20%		
2.	Praktik	CPL 08-KK01	CPMK-03	30%		
3.	Post-Test	CPL 08-KK01	CPMK-03	50%		
<b>Total Nilai</b>						



## PRAKTIKUM 7: KURVA SPLINE

Pertemuan ke : 7

Total Alokasi Waktu : 90 menit

- Materi : 10 menit
- Pre-Test : 10 menit
- Praktikum : 40 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 08-KK01	Mampu memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah
CPMK-03	Mampu menjelaskan konsep representasi objek 3D, representasi kurva spline, representasi permukaan, dan menerapkannya menggunakan OpenGL

### 7.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Mahasiswa mampu menjelaskan tentang konsep kurva spline.
2. Mahasiswa mampu menjelaskan tentang jenis-jenis kurva spline.
3. Mahasiswa mampu menerapkan kurva spline dengan OpenGL.

### 7.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 08-KK01	CPMK-03	Kemampuan mahasiswa dalam menerapkan kurva spline dengan OpenGL.
-------------	---------	--

### 7.3. TEORI PENDUKUNG

Kurva spline merupakan kurva yang digambar secara fleksibel untuk menghasilkan kurva yang smooth melalui titik-titik kontrolnya. Untuk menjaga agar kurva tetap smooth maka diperlukan kontinuitas di titik-titik kontrolnya. Kurva spline mempunyai ciri khas yaitu mempunyai titik control yang ditentukan user dan titik control tersebut yang akan di interpolasi menjadi kurva. Jenis-jenis kurva spline diantaranya:

#### Kubik Spline

Kubik spline menggunakan fungsi polynomial pangkat tiga dengan masukan 4 titik control. Untuk membuat kurva sebelumnya harus diketahui terlebih dahulu koefisien polynomial dari keempat titik control tadi.

$$p(u) = c_0 + c_1u + c_2u^2 + c_3u^3$$

Bila titik control adalah  $p_0, p_1, p_2, p_3$  dan misalnya diberikan  $u = 0, \frac{1}{3}, \frac{2}{3}, 1$ ,  $u$  merupakan nilai didalam interval  $[0, 1]$ . Maka kondisi untuk keempat titik menjadi:

- $p_0 = p(0) = c_0$
- $p_1 = p\left(\frac{1}{3}\right) = c_0 + \frac{1}{3}c_1 + \left(\frac{1}{3}\right)^2 c_2 + \left(\frac{1}{3}\right)^3 c_3$
- $p_2 = p\left(\frac{2}{3}\right) = c_0 + \frac{2}{3}c_1 + \left(\frac{2}{3}\right)^2 c_2 + \left(\frac{2}{3}\right)^3 c_3$
- $p_3 = p(1) = c_0 + c_1 + c_2 + c_3$

Rumus polynomial kubik dapat dituliskan

$$p = Ac \text{ dimana}$$

$$p = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}, \text{ dan } A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \frac{1}{3} & \left(\frac{1}{3}\right)^2 & \left(\frac{1}{3}\right)^3 \\ 1 & \frac{2}{3} & \left(\frac{2}{3}\right)^2 & \left(\frac{2}{3}\right)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Untuk menghitung  $c$  kita invers matriksnya misalnya  $M_I = A^{-1}$  sehingga  $c = M_I p$

### Catmull-Rom Spline

Catmull-Rom spline menginterpolasi titik tengah pada titik control. Bila diberikan 4 titik control  $p_0, p_1, p_2, p_3$  maka yang diinterpolasi titik tengahnya saja.

- $p(0) = p_1$
- $p(1) = p_2$

Hitung tangent

- $p'(0) \approx \frac{p_2 - p_0}{2}$
- $p'(1) \approx \frac{p_3 - p_1}{2}$

Rumus catmull-rom spline dapat dituliskan

$$p = Ac \text{ dimana}$$

$$p = \begin{bmatrix} p_1 \\ p_2 \\ \frac{p_2 - p_0}{2} \\ \frac{p_3 - p_1}{2} \end{bmatrix}, \text{ dan } A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix}$$

Untuk menghitung  $c$  kita invers matriksnya misalnya  $M_R = A^{-1}$  sehingga  $c = M_R p$

### Bezier Spline

Bezier spline menggunakan titik ujung dari titik control untuk interpolasi kemudian menghitung arah tangen untuk menentukan arah kurva. Bila diketahui 2 titik control  $p_0$  dan  $p_3$ .

- $p_0 = P(0) = c_0$
- $p_3 = p(1) = c_0 + c_1 + c_2 + c_3$

Bezier menyatakan bahwa titik  $p_1, p_2$  digunakan untuk memperkirakan tangen garis antara  $p_0, p_3$

- $p'(0) \approx \frac{\Delta p}{\Delta u} = \frac{p_1 - p_0}{\frac{1}{3}} = 3(p_1 - p_0) = c_1$
- $p'(1) \approx \frac{\Delta p}{\Delta u} = \frac{p_3 - p_2}{\frac{1}{3}} = 3(p_3 - p_2) = c_1 + 2c_2 + 3c_3$

Rumus kurva bezier dapat dituliskan

$$p = Ac \text{ dimana}$$

$$p = \begin{bmatrix} p_0 \\ p_3 \\ 3(p_1 - p_0) \\ 3(p_3 - p_2) \end{bmatrix}, \text{ dan } A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix}$$

Untuk menghitung  $c$  kita invers matriksnya misalnya  $M_B = A^{-1}$  sehingga  $c = M_B p$

#### 7.4. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 08-KK01	CPMK-03	Jelaskan perbedaan antara kurva spline Cubic, Catmull-Rom, Hermit, dan Bezier!	100

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

#### 7.5. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.

#### 7.6. LANGKAH PRAKTIKUM

##### PERSIAPAN

1. Buka Visual Studio C++ dan buat project baru dengan nama **praktikum07**.
2. Download kode dasar praktikum Grafika Komputer dan Library OpenGL seperti pada Praktikum 1.
3. Ubah nama dari kode dasar "**praktikum00.cpp**" menjadi "**praktikum07.cpp**" dan copy-kan ke **Source Files** di project yang anda buat.
4. Setting OpenGL library pada Visual Studio C/C++ seperti pada Praktikum 1.
5. Download file tambahan **praktikum07.h** di elearning Grafika Komputer lalu copy-kan ke folder **Source Files** project anda.

##### PRAKTIKUM

**Aturan Penilaian (Total Skor: 100):**

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 08-KK01	CPMK-03	Selesaikan langkah praktikum 1 – 7	Hasil praktikum langkah 1 – 7	50
2.	CPL 08-KK01	CPMK-03	Selesaikan langkah praktikum 8 – 11	Hasil praktikum langkah 8 - 11	50

##### Langkah-Langkah:

1. Tambahkan deklarasi header tambahan praktikum07 berikut di praktikum07.cpp.

```
// header untuk praktikum 07
#include "praktikum07.h"
```

2. Di fungsi **init()** dalam praktikum07.cpp, ubah baris kode berikut untuk mengubah warna latar belakang pada layar menjadi warna hitam.

```
glClearColor(1.0, 1.0, 1.0, 0.0);

menjadi

glClearColor(0.0, 0.0, 0.0, 0.0);
```

3. Tambahkan fungsi **drawSplineCubic()** berikut pada praktikum07.cpp untuk membuat kurva spline Cubic.

```
// fungsi untuk membuat kurva spline cubic dari 4 titik kontrol
// point1 sampai point4 = titik kontrol
// nPoint = jumlah titik interpolasi antara point1 sampai point4
void drawSplineCubic(Vec3 point1, Vec3 point2, Vec3 point3, Vec3 point4,
    int nPoint)
{
    // hitung bobot jarak u di masing-masing titik
    float utotal = (abs(point2.X - point1.X) + abs(point3.X - point2.X) +
        abs(point4.X - point3.X));
    float u1 = 0;
    float u2 = abs(point2.X - point1.X) / utotal;
    float u3 = abs(point2.X - point1.X) + abs(point3.X - point2.X) /
        utotal;
    float u4 = 1;

    // hitung inverse matriks dari koefisien u (lihat slide kuliah)
    float inverseMat[16];
    float coeffMat[16] = {
        1.00f, 0.00f, 0.00f, 0.00f,
        1.00f, u2, pow(u2, 2), pow(u2, 3),
        1.00f, u3, pow(u3, 2), pow(u3, 3),
        1.00f, 1.00f, 1.00f, 1.00f };
    bool status = inverse(coeffMat, inverseMat);

    // hitung koefisien cubic au^3 + bu^2 + cu + d
    if (status == true)
    {
        float outMatX[4], outMatY[4], outMatZ[4];
        float inMatX[4] = { point1.X, point2.X, point3.X, point4.X };
        float inMatY[4] = { point1.Y, point2.Y, point3.Y, point4.Y };
        float inMatZ[4] = { point1.Z, point2.Z, point3.Z, point4.Z };

        DotMatrix(inverseMat, inMatX, outMatX);
        DotMatrix(inverseMat, inMatY, outMatY);
        DotMatrix(inverseMat, inMatZ, outMatZ);

        // gambar kurva cubic spline dengan titik kontrol diatas hitung
        // posisi y untuk setiap x di setiap point dengan persamaan diatas
        for (int i=0; i<nPoint; i++)
        {
            // jeda setiap titik pd bobot u
            float step = 1.0f / nPoint;
            // titik awal
            float pX = point1.X, pY = point1.Y, pZ = point1.Z;
            //
            float u = 0.0f;
            for (int i = 0; i < nPoint; i++)
            {
                // segment kurva cubic spline sebanyak nPoint
                u = u + step;
                glVertex3f(pX, pY, pZ); // gambar titik awal
                // koordinat X pada kurva
                pX = outMatX[3] * pow(u, 3) + outMatX[2] *
                    pow(u, 2) + outMatX[1] * u + outMatX[0];
                // koordinat Y pada kurva
                pY = outMatY[3] * pow(u, 3) + outMatY[2] *
                    pow(u, 2) + outMatY[1] * u + outMatY[0];
                // koordinat Z pada kurva
                pZ = outMatZ[3] * pow(u, 3) + outMatZ[2] *
```

```

        pow(u, 2) + outMatZ[1] * u + outMatZ[0];
        glVertex3f(pX, pY, pZ); // gambar titik akhir
    }
}
}
}

```

4. Tambahkan fungsi **drawSplineBezier()** berikut pada praktikum07.cpp untuk membuat kurva spline Bezier.

```

// fungsi untuk membuat kurva spline bezier dari 4 titik kontrol
// point1 dan point4 = titik kontrol awal dan akhir
// point2 dan point3 = titik kontrol pembentuk kurva
// nPoint = jumlah titik interpolasi antara point1 sampai point4
void drawSplineBezier(Vec3 point1, Vec3 point2, Vec3 point3,
    Vec3 point4, int nPoint)
{
    // hitung bobot jarak u di masing-masing titik
    float utotal = (abs(point2.X - point1.X) + abs(point3.X - point2.X) +
        abs(point4.X - point3.X));
    float u1 = 0;
    float u2 = abs(point2.X - point1.X) / utotal;
    float u3 = (abs(point2.X - point1.X) + abs(point3.X - point2.X)) /
        utotal;
    float u4 = 1;

    // hitung inverse matriks dari koefisien u (lihat slide kuliah)
    float inverseMat[16];
    float coeffMat[16] = {
        1.0000f, 0.0000f, 0.0000f, 0.0000f,
        1.0000f, 1.0000f, 1.0000f, 1.0000f,
        0.0000f, 1.0000f, 0.0000f, 0.0000f,
        0.0000f, 1.0000f, 2.0000f, 3.0000f };
    bool status = inverse(coeffMat, inverseMat);

    // hitung koefisien
    if (status == true)
    {
        float outMatX[4], outMatY[4], outMatZ[4];
        float inMatX[4] = { point1.X, point4.X,
            1.0f/(u2-u1)*(point2.X - point1.X),
            1.0f/(u4-u3)*(point4.X - point3.X) };
        float inMatY[4] = { point1.Y, point4.Y,
            1.0f/(u2-u1)*(point2.Y - point1.Y),
            1.0f/(u4-u3)*(point4.Y - point3.Y) };
        float inMatZ[4] = { point1.Z, point4.Z,
            1.0f/(u2-u1)*(point2.Z - point1.Z),
            1.0f/(u4-u3)*(point4.Z - point3.Z) };

        DotMatrix(inverseMat, inMatX, outMatX);
        DotMatrix(inverseMat, inMatY, outMatY);
        DotMatrix(inverseMat, inMatZ, outMatZ);

        // gambar kurva cubic spline dengan titik kontrol diatas hitung
        // posisi y untuk setiap x di setiap point dengan persamaan diatas
        for (int i=0; i<nPoint; i++)
        {
            // jeda setiap titik pd bobot u
            float step = 1.0f / nPoint;
            // titik awal
            float pX = point1.X, pY = point1.Y, pZ = point1.Z;
            //
            float u = 0.0f;

```

```

    for (int i = 0; i < nPoint; i++)
    {
        // bentuk segment kurva spline sebanyak nPoint
        u = u + step;
        glVertex3f(pX, pY, pZ); // gambar titik awal
        // koordinat X pada kurva
        pX = outMatX[3] * pow(u, 3) + outMatX[2] *
            pow(u, 2) + outMatX[1] * u + outMatX[0];
        // koordinat Y pada kurva
        pY = outMatY[3] * pow(u, 3) + outMatY[2] *
            pow(u, 2) + outMatY[1] * u + outMatY[0];
        // koordinat Z pada kurva
        pZ = outMatZ[3] * pow(u, 3) + outMatZ[2] *
            pow(u, 2) + outMatZ[1] * u + outMatZ[0];
        glVertex3f(pX, pY, pZ); // gambar titik akhir
    }
}
}
}

```

5. Tambahkan fungsi **drawSplineCatmullRom()** berikut pada praktikum07.cpp untuk membuat kurva spline Catmull-Rom.

```

// fungsi untuk membuat kurva spline catmull-rom dari 4 titik kontrol
// point1 dan point4 = titik kontrol awal dan akhir
// point2 dan point3 = titik kontrol pembentuk kurva
// nPoint = jumlah titik interpolasi antara point1 sampai point4
void drawSplineCatmullRom(Vec3 point1, Vec3 point2, Vec3 point3,
    Vec3 point4, int nPoint)
{
    // hitung bobot jarak u di masing-masing titik
    float utotal = (abs(point2.X - point1.X) + abs(point3.X - point2.X) +
        abs(point4.X - point3.X));
    float u1 = 0;
    float u2 = abs(point2.X - point1.X) / utotal;
    float u3 = (abs(point2.X - point1.X) + abs(point3.X - point2.X)) /
        utotal;
    float u4 = 1;

    // hitung inverse matriks dari koefisien u (lihat slide kuliah)
    float inverseMat[16];
    float coeffMat[16] = {
        1.0000f, 0.0000f, 0.0000f, 0.0000f,
        1.0000f, 1.0000f, 1.0000f, 1.0000f,
        0.0000f, 1.0000f, 0.0000f, 0.0000f,
        0.0000f, 1.0000f, 2.0000f, 3.0000f };
    bool status = inverse(coeffMat, inverseMat);

    // hitung koefisien
    if (status == true)
    {
        float outMatX[4], outMatY[4], outMatZ[4];
        float inMatX[4] = { point1.X, point4.X,
            1.0f/(u3-u1)*(point3.X - point1.X),
            1.0f/(u4-u2)*(point4.X - point2.X) };
        float inMatY[4] = { point1.Y, point4.Y,
            1.0f/(u3-u1)*(point3.Y - point1.Y),
            1.0f/(u4-u2)*(point4.Y - point2.Y) };
        float inMatZ[4] = { point1.Z, point4.Z,
            1.0f/(u3-u1)*(point3.Z - point1.Z),
            1.0f/(u4-u2)*(point4.Z - point2.Z) };

        DotMatrix(inverseMat, inMatX, outMatX);
    }
}

```

```

DotMatrix(inverseMat, inMatY, outMatY);
DotMatrix(inverseMat, inMatZ, outMatZ);

// gambar kurva spline dengan titik kontrol diatas hitung posisi
// y untuk setiap x di setiap point dengan persamaan diatas
for (int i=0; i<nPoint; i++)
{
    // jeda setiap titik pd bobot u
    float step = 1.0f / nPoint;
    // titik awal
    float pX = point1.X, pY = point1.Y, pZ = point1.Z;
    //
    float u = 0.0f;
    for (int i = 0; i < nPoint; i++)
    {
        // bentuk segment kurva spline sebanyak nPoint
        u = u + step;
        glVertex3f(pX, pY, pZ); // gambar titik awal
        // koordinat X pada kurva
        pX = outMatX[3] * pow(u, 3) + outMatX[2] *
            pow(u, 2) + outMatX[1] * u + outMatX[0];
        // koordinat Y pada kurva
        pY = outMatY[3] * pow(u, 3) + outMatY[2] *
            pow(u, 2) + outMatY[1] * u + outMatY[0];
        // koordinat Z pada kurva
        pZ = outMatZ[3] * pow(u, 3) + outMatZ[2] *
            pow(u, 2) + outMatZ[1] * u + outMatZ[0];
        glVertex3f(pX, pY, pZ); // gambar titik akhir
    }
}
}
}

```

6. Ubah fungsi **drawObject()** pada praktikum07.cpp menjadi seperti dibawah ini.

```

// fungsi untuk menggambar obyek kubus
void drawObject()
{
    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah kanan-kiri
    glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah atas-bawah
    glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

    // membuat 4 titik kontrol kurva
    Vec3 point1 = Vec3(-150.0f, -70.0f, 0.0f);
    Vec3 point2 = Vec3(-50.0f, 50.0f, 0.0f);
    Vec3 point3 = Vec3(50.0f, 10.0f, 0.0f);
    Vec3 point4 = Vec3(150.0f, -50.0f, 0.0f);

    // tandai setiap titik kontrol kurva dengan warna
    markPoint(point1, Vec3(0.0f, 1.0f, 0.0f), 5.0f);
    markPoint(point2, Vec3(1.0f, 0.0f, 0.0f), 5.0f);
    markPoint(point3, Vec3(1.0f, 0.0f, 1.0f), 5.0f);
    markPoint(point4, Vec3(1.0f, 1.0f, 0.0f), 5.0f);

    // mengatur warna obyek menjadi berwarna putih
    glColor3f(1.0f, 1.0f, 1.0f);
}

```

```

glBegin(GL_LINES);

// membuat kurva spline cubic dari titik kontrol diatas
drawSplineCubic(point1, point2, point3, point4, 30);

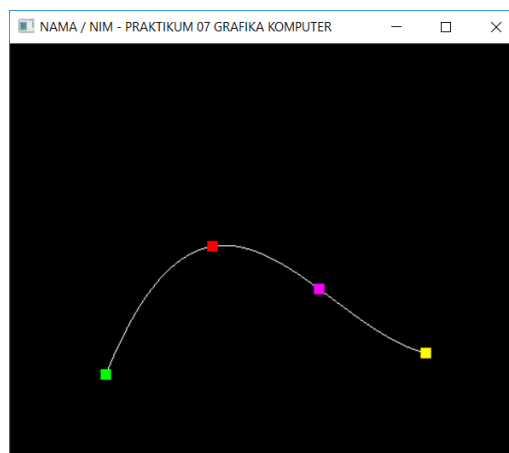
glEnd();

glPopMatrix();

glPopMatrix();
}

```

7. Jalankan program untuk menghasilkan gambar kurva spline Cubic seperti yang ditunjukkan pada Gambar 7.1.



Gambar 7.1 Hasil menggambar kurva spline Cubic.

8. Ubah kode di fungsi drawObject() untuk menggambar kurva spline Bezier seperti berikut.

```

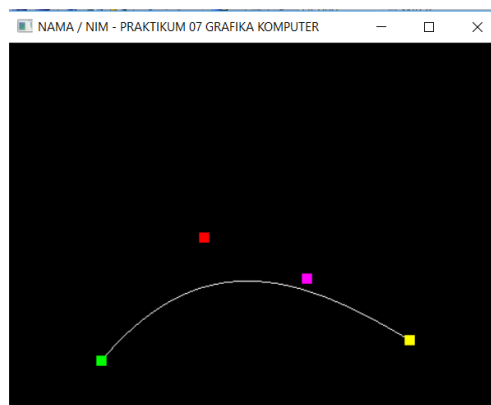
// membuat kurva spline cubic dari titik kontrol diatas
drawSplineCubic(point1, point2, point3, point4, 30);

menjadi

// membuat kurva spline bezier dari titik kontrol diatas
drawSplineBezier(point1, point2, point3, point4, 30);

```

9. Jalankan program untuk menghasilkan gambar kurva spline Bezier seperti yang ditunjukkan pada Gambar 7.2.



Gambar 7.2 Hasil menggambar kurva spline Bezier



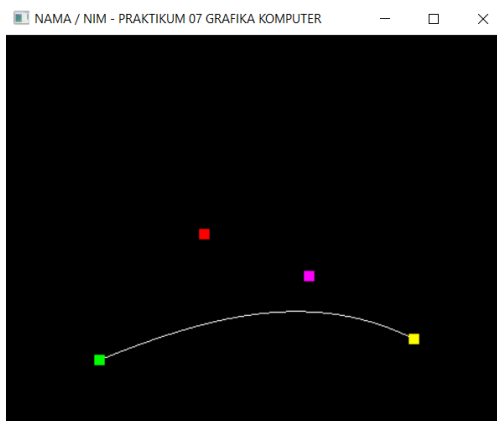
10. Ubah kode di fungsi drawObject() untuk menggambar kurva spline Bezier seperti berikut.

```
// membuat kurva spline bezier dari titik kontrol diatas
drawSplineBezier(point1, point2, point3, point4, 30);

menjadi

// membuat kurva spline catmullrom dari titik kontrol diatas
drawSplineCatmullRom(point1, point2, point3, point4, 30);
```

11. Jalankan program untuk menghasilkan gambar kurva spline Catmull-Rom seperti yang ditunjukkan pada Gambar 7.3.



Gambar 7.3 Hasil menggambar kurva spline Catmull-Rom

## 7.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 08-KK01	CPMK-03	Buatlah buatlah kurva spline cubic, bezier dan catmull rom dengan posisi titik control: 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) Screenshot hasil 3 kurva spline anda!	100

## 7.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 08-KK01	CPMK-03	20%		
2.	Praktik	CPL 08-KK01	CPMK-03	30%		
3.	Post-Test	CPL 08-KK01	CPMK-03	50%		
<b>Total Nilai</b>						

## PRAKTIKUM 8: TEKNIK REPRESENTASI PERMUKAAN

Pertemuan ke : 8

Total Alokasi Waktu : 90 menit

- Materi : 10 menit
- Pre-Test : 10 menit
- Praktikum : 40 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 08-KK01	Mampu memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah
CPMK-03	Mampu menjelaskan konsep representasi objek 3D, representasi kurva spline, representasi permukaan, dan menerapkannya menggunakan OpenGL

### 8.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Mahasiswa mampu menjelaskan tentang konsep permukaan.
2. Mahasiswa mampu menjelaskan tentang teknik-teknik permukaan.
3. Mahasiswa mampu menerapkan teknik permukaan dengan OpenGL.

### 8.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 08-KK01	CPMK-03	Kemampuan mahasiswa dalam menerapkan teknik representasi permukaan dengan OpenGL.
-------------	---------	---

### 8.3. TEORI PENDUKUNG

Dalam representasi obyek 3D menggunakan boundary representation, terdapat permukaan yang harus ada untuk memisahkan antara obyek bagian dalam dan luar. Permukaan obyek dapat dibuat menggunakan polygon, kurva, quadric, spline, dan lain-lain. Semua jenis kurva spline dapat digunakan untuk membuat permukaan obyek seperti kurva NURBS (Non-Uniform Rational B-Splines). Kurva NURBS merupakan pengembangan dari kurva B-Spline. Bila pada kurva B-Spline 3D titik kontrolnya adalah  $p_i = [x_i, y_i, z_i]$  dan bila direpresentasikan ke koordinat homogen terbobot menjadi:

$$q_i = w_i \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}$$

Bobot  $w_i$  digunakan untuk meningkatkan atau menurunkan pengaruh dari titik control terhadap pembentukan kurva. Titik terbobot tersebut bisa digunakan untuk membentuk B-Spline 4D yang ditunjukkan pada matriks berikut.

Tiga komponen pertama merupakan titik terbobot.

$$q(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \sum_{i=0}^n B_{i,d}(u) w_i p_i$$

Sedangkan komponen terakhir adalah scalar polynomial B-Spline.

$$w(u) = \sum_{i=0}^n B_{i,d}(u) w_i$$

Karena bobot  $w$  mungkin tidak bernilai 1 maka ketiga koordinat perlu disesuaikan.

$$p(u) = \frac{1}{w(u)} q(u) = \frac{\sum_{i=0}^n B_{i,d}(u) w_i p_i}{\sum_{i=0}^n B_{i,d}(u) w_i}$$

Kurva NURBS banyak digunakan dalam aplikasi computer grafis Karena mempunyai kelebihan dibanding dengan kurva B-Spline yang lain. Bila diterapkan transformasi affine, kurva NURBS akan menghasilkan kurva yang benar dari sudut pandang perspektif.

#### 8.4. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 08-KK01	CPMK-03	Sebutkan teknik representasi permukaan yang anda ketahui!	30
2.	CPL 08-KK01	CPMK-03	Jelaskan setiap teknik representasi permukaan yang anda sebutkan di soal nomor 1!	70

#### 8.5. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.

#### 8.6. LANGKAH PRAKTIKUM

##### PERSIAPAN

1. Buka Visual Studio C++ dan buat project baru dengan nama **praktikum09**.
2. Download kode dasar praktikum Grafika Komputer dan Library OpenGL seperti pada Praktikum 1.
3. Ubah nama dari kode dasar "**praktikum00.cpp**" menjadi "**praktikum09.cpp**" dan copy-kan ke **Source Files** di project yang anda buat.
4. Setting OpenGL library pada Visual Studio C/C++ seperti pada Praktikum 1.

##### PRAKTIKUM

Aturan Penilaian (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
----	-----	------	------------	-------------------	------

1.	CPL 08-KK01	CPMK-03	Selesaikan langkah praktikum 1 – 5	Hasil praktikum langkah 1 – 5	100
----	-------------	---------	------------------------------------	-------------------------------	-----

### Langkah-Langkah:

1. Tambahkan variable berikut di praktikum09.cpp untuk inialisasi pembuatan permukaan dengan kurva (tambahkan dibawah struct Vec3()).

```
Vec3 controlPoint[4][4];
bool showPoints = false;

GLfloat mat_diffuse[] = { 0.7, 0.7, 0.7, 1.0 };
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_shininess[] = { 100.0 };

GLUnurbsObj *theNurb;
```

2. Ubah fungsi **drawObject()** di praktikum09.cpp untuk menggambar permukaan dengan kurva NURBS.

```
// fungsi untuk menggambar obyek
void drawObject()
{
    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah kanan-kiri
    glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah atas-bawah
    glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);
    glScalef(0.5, 0.5, 0.5);

    // set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)
    glColor3f(0.0f, 1.0f, 0.0f);

    //
    GLfloat knots[8] = { 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0 };

    gluBeginSurface(theNurb);
    gluNurbsSurface(theNurb, 8, knots, 8, knots,
        4 * 3, 3, &controlPoint[0][0].X, 4, 4, GL_MAP2_VERTEX_3);
    gluEndSurface(theNurb);

    if (showPoints)
    {
        glPointSize(5.0);
        glDisable(GL_LIGHTING);
        glColor3f(1.0, 1.0, 0.0);
        glBegin(GL_POINTS);
        for (int i = 0; i < 4; i++)
        {
            for (int j = 0; j < 4; j++)
            {
                glVertex3f(controlPoint[i][j].X, controlPoint[i][j].Y,
                    controlPoint[i][j].Z);
            }
        }
        glEnd();
        glEnable(GL_LIGHTING);
    }
}
```

```

    glPopMatrix();

    glPopMatrix();
}

```

3. Ubah fungsi **init()** di praktikum09.cpp seperti di bawah ini.

```

// inisialisasi
void init(void)
{
    // inisialisasi warna latar belakang
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glEnable(GL_DEPTH_TEST); // mengaktifkan depth buffer
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    // inisialisasi kamera pandang
    gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

    //
    int u, v;
    for (u = 0; u < 4; u++)
    {
        for (v = 0; v < 4; v++)
        {
            controlPoint[u][v].X = 2.0*((GLfloat)u - 1.5);
            controlPoint[u][v].Y = 2.0*((GLfloat)v - 1.5);

            if ((u == 1 || u == 2) && (v == 1 || v == 2))
                controlPoint[u][v].Z = 3.0;
            else
                controlPoint[u][v].Z = -3.0;
        }
    }

    glClearColor(0.0, 0.0, 0.0, 0.0);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_AUTO_NORMAL);
    glEnable(GL_NORMALIZE);

    theNurb = gluNewNurbsRenderer();
    gluNurbsProperty(theNurb, GLU_SAMPLING_TOLERANCE, 25.0);
    gluNurbsProperty(theNurb, GLU_DISPLAY_MODE, GLU_FILL);
}

```

4. Tambahkan kode dibawah ini pada fungsi **keyboard()** agar tombol F1 menampilkan titik control dari NURB.

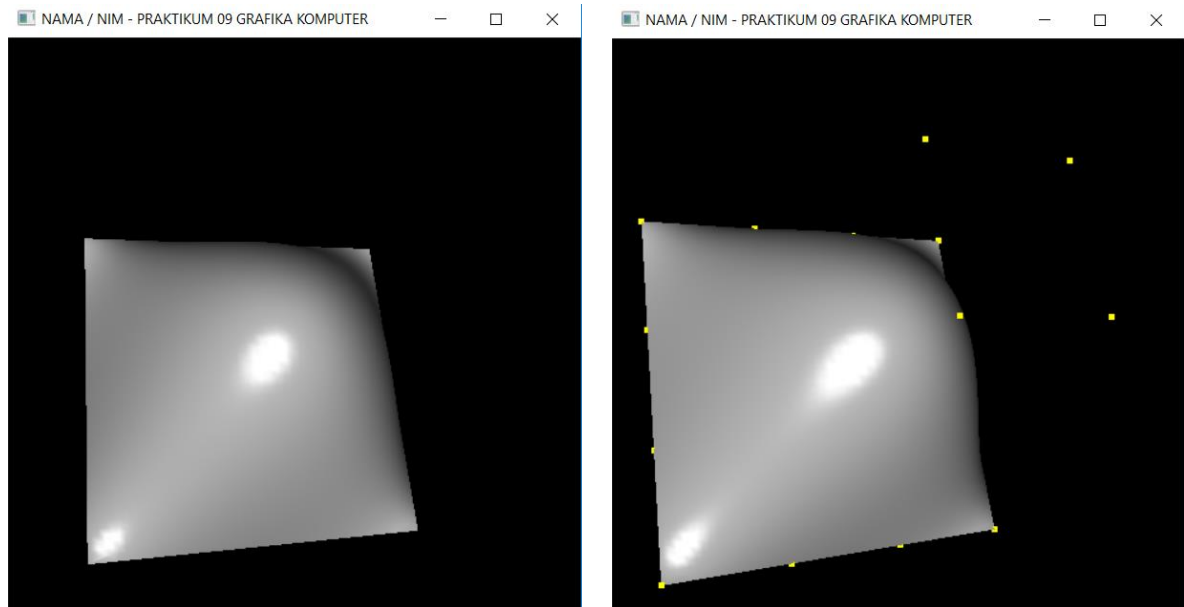
```

...
// tampilkan point
case GLUT_KEY_F1:
    showPoints = !showPoints;
    break;

```

...

5. Jalankan program untuk menghasilkan gambar permukaan dari kurva seperti ditunjukkan pada Gambar 8.1. Tekan F1 untuk menampilkan titik-titik control permukaan kurva NURBS.



Gambar 8.1 Permukaan kurva NURBS

### 8.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 08-KK01	CPMK-03	Buatlah permukaan NURB untuk memodelkan permukaan yang menjorok ke dalam!	100

### 8.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 08-KK01	CPMK-03	20%		
2.	Praktik	CPL 08-KK01	CPMK-03	30%		
3.	Post-Test	CPL 08-KK01	CPMK-03	50%		
<b>Total Nilai</b>						

## PRAKTIKUM 9: TEKNIK PEMODELAN OBYEK 3D

Pertemuan ke : 9

Total Alokasi Waktu : 90 menit

- Materi : 10 menit
- Pre-Test : 10 menit
- Praktikum : 40 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 09-KK02	Mampu memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah
CPMK-04	Mampu menjelaskan dan menerapkan teknik pemodelan 3D, teknik subdivisi, rekonstruksi 3D, dan aplikasi Augmented Reality/Virtual Reality

### 9.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Mahasiswa mampu menjelaskan tentang konsep pemodelan 3D.
2. Mahasiswa mampu menjelaskan teknik pemodelan obyek 3D.
3. Mahasiswa mampu menerapkan pemodelan obyek 3D dengan OpenGL.

### 9.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 09-KK02	CPMK-04	Kemampuan mahasiswa dalam menerapkan teknik pemodelan 3D dengan OpenGL.
-------------	---------	---

### 9.3. TEORI PENDUKUNG

Teknik pemodelan 3D menggunakan pemodelan dengan dataset polygon. Model dibangun dari kombinasi bentuk-bentuk polygon. Teknik pemodelan dengan dataset polygon diantaranya dengan melakukan triangulating polygon, subdivisi permukaan poligon, lofting, surface of revolution, beveling, Delaunay triangulation, pemodelan Boolean, pemodelan metaball dan marching cube.

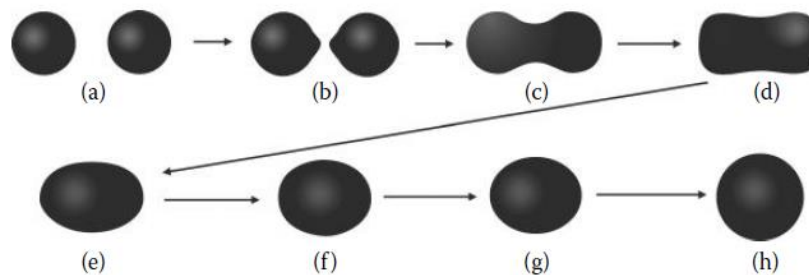
Metaball merupakan obyek blobby yang dibuat dari kombinasi beberapa bidang dan membuat permukaannya bergabung di setiap tempat dimana bidang mendapatkan nilai yang sama. Algoritma dalam membuat metaball dimulai dengan menentukan threshold ( $t$ ) dan fungsi ( $f$ ) yang mendefinisikan permukaan metaball. Berikut adalah fungsi yang merepresentasikan volume yang dilingkupi oleh permukaan metaball.

$$\sum_{i=1}^n f_i(x, y, z) \leq t$$

Fungsi yang dipilih untuk membuat metaball biasanya:

$$f(x, y, z) = \frac{1}{((x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2)}$$

Dimana  $(x_0, y_0, z_0)$  adalah titi pusat metaball. Bila metaball berdekatan maka iso-surface nya merupakan penambahan dari bidang kedua metaball tersebut seperti yang ditunjukkan pada Gambar 9.1.



Gambar 9.1 Permukaan metaball

#### 9.4. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 09-KK02	CPMK-04	Sebutkan teknik pemodelan 3D yang anda ketahui!	30
2.	CPL 09-KK02	CPMK-04	Jelaskan setiap teknik pemodelan 3D yang anda sebutkan di soal nomor 1!	70

#### 9.5. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

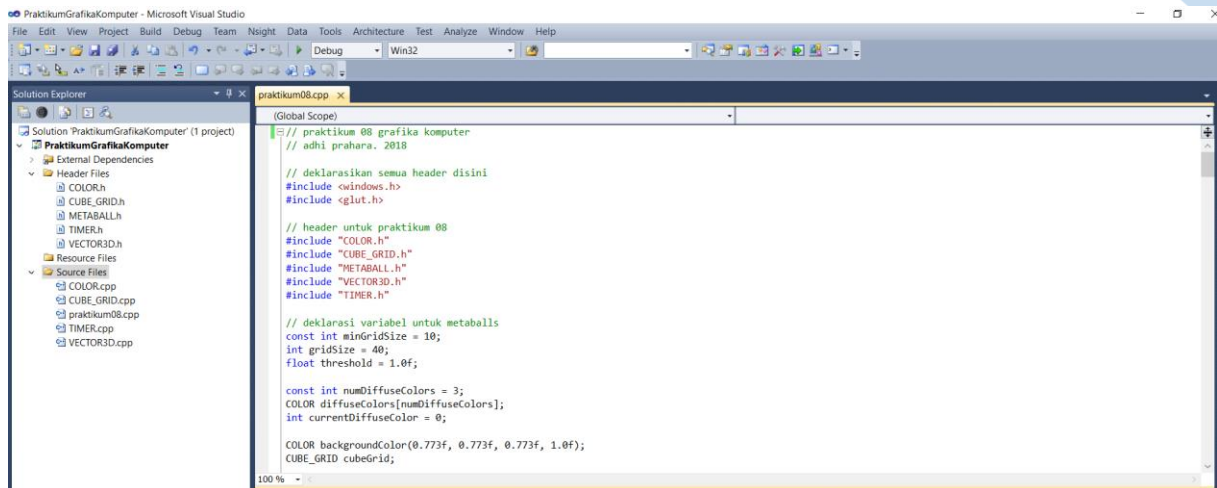
1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.

#### 9.6. LANGKAH PRAKTIKUM

##### PERSIAPAN

1. Buka Visual Studio C++ dan buat project baru dengan nama **praktikum08**.
2. Download kode dasar praktikum Grafika Komputer dan Library OpenGL seperti pada praktikum 1.
3. Ubah nama dari kode dasar "**praktikum00.cpp**" menjadi "**praktikum08.cpp**" dan copy-kan ke **Source Files** di project yang anda buat.
4. Setting OpenGL library pada Visual Studio C/C++ seperti pada Praktikum 1.
5. Download file tambahan untuk praktikum08 di e-learning dan ekstrak di lokasi dimana source files project anda berada.
6. Masukkan semua file .h dari file tambahan praktikum08 ke **Header Files** di project anda.
7. Masukkan semua file .cpp dari file tambahan praktikum08 ke **Source Files** di project anda.
8. Tampilan Visual Studio setelah pengaturan diatas selesai seperti ditunjukkan pada Gambar 9.2.





Gambar 9.2 Setting file tambahan praktikum09 di Visual Studio

## PRAKTIKUM

### Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 09-KK02	CPMK-04	Selesaikan langkah praktikum 1 – 7	Hasil praktikum langkah 1 – 7	100

### Langkah-Langkah:

1. Tambahkan deklarasi header tambahan praktikum08 berikut di praktikum08.cpp.

```
// header untuk praktikum 08
#include "COLOR.h"
#include "CUBE_GRID.h"
#include "METABALL.h"
#include "VECTOR3D.h"
#include "TIMER.h"
```

2. Tambahkan variable berikut di praktikum08.cpp untuk inisialisasi metaball dan pencahayaan.

```
const int minGridSize = 10;
int gridSize = 40;
float threshold = 1.0f;

const int numDiffuseColors = 3;
COLOR diffuseColors[numDiffuseColors];
int currentDiffuseColor = 0;

COLOR backgroundColor(0.773f, 0.773f, 0.773f, 1.0f);
CUBE_GRID cubeGrid;

const int numMetaballs = 3;
METABALL metaballs[numMetaballs];
TIMER timer;

//set up lighting
float shininess = 32.0f;
float ambient[] = { 0.0f, 0.0f, 0.2f, 1.0f };
float position[] = { -1.0f, 1.0f, 1.0f, 0.0f };
float specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
```

3. Ubah fungsi **drawObject()** di praktikum08.cpp untuk menggambar metaball seperti di bawah ini.

```
// fungsi ini digunakan untuk menggambar obyek
void drawObject()
{
    glPushMatrix();

    glRotatef(objectAngle, objectRotation.X, objectRotation.Y,
        objectRotation.Z);

    // dinormalisasi dulu
    glEnable(GL_NORMALIZE);
    glEnable(GL_CULL_FACE);

    glLightfv(GL_LIGHT1, GL_AMBIENT, ambient);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, diffuseColors[currentDiffuseColor]);
    glLightfv(GL_LIGHT1, GL_POSITION, position);
    glLightfv(GL_LIGHT1, GL_SPECULAR, specular);
    glEnable(GL_LIGHT1);

    // set pencahayaan
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, white);
    glMaterialfv(GL_FRONT, GL_SPECULAR, white);
    glMaterialfv(GL_FRONT, GL_SHININESS, &shininess);

    // update posisi metaball
    float c = 2.0f*(float)cos(timer.GetTime() / 600);

    metaballs[0].position.x = -4.0f*(float)cos(timer.GetTime()/700) - c;
    metaballs[0].position.y = 4.0f*(float)sin(timer.GetTime()/600) - c;

    metaballs[1].position.x = 5.0f*(float)sin(timer.GetTime()/400) + c;
    metaballs[1].position.y = 5.0f*(float)cos(timer.GetTime()/400) - c;

    metaballs[2].position.x = -5.0f*(float)cos(timer.GetTime()/400) -
        0.2f*(float)sin(timer.GetTime() / 600);
    metaballs[2].position.y = 5.0f*(float)sin(timer.GetTime()/500) -
        0.2f*(float)sin(timer.GetTime() / 400);

    // bersihkan layar
    for (int i = 0; i<cubeGrid.numVertices; i++)
    {
        cubeGrid.vertices[i].value = 0.0f;
        cubeGrid.vertices[i].normal.LoadZero();
    }

    // hitung bidang skalar disetiap titik
    VECTOR3D ballToPoint;
    float squaredRadius;
    VECTOR3D ballPosition;
    float normalScale;
    for (int i = 0; i<numMetaballs; i++)
    {
        squaredRadius = metaballs[i].squaredRadius;
        ballPosition = metaballs[i].position;
        for (int j = 0; j<cubeGrid.numVertices; j++)
        {
            ballToPoint.x = cubeGrid.vertices[j].position.x -
                ballPosition.x;
            ballToPoint.y = cubeGrid.vertices[j].position.y -
                ballPosition.y;
            ballToPoint.z = cubeGrid.vertices[j].position.z -
                ballPosition.z;
```

```

        // hitung jarak bola ke titik
        float squaredDistance = ballToPoint.x*ballToPoint.x +
            ballToPoint.y*ballToPoint.y + ballToPoint.z*ballToPoint.z;
        if (squaredDistance == 0.0f)
            squaredDistance = 0.0001f;
        // value = r^2/d^2
        cubeGrid.vertices[j].value += squaredRadius / squaredDistance;
        // normal = (r^2 * v)/d^4
        normalScale = squaredRadius / (squaredDistance*squaredDistance);
        cubeGrid.vertices[j].normal.x += ballToPoint.x*normalScale;
        cubeGrid.vertices[j].normal.y += ballToPoint.y*normalScale;
        cubeGrid.vertices[j].normal.z += ballToPoint.z*normalScale;
    }
}

glPopMatrix();
}

```

4. Ubah fungsi `display()` di `praktikum08.cpp` seperti di bawah ini.

```

// taruh semua fungsi obyek yang akan digambar di fungsi display()
void display()
{
    // bersihkan dan reset layar dan buffer
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    gluLookAt(camPosition.X, camPosition.Y, camPosition.Z,
        camPosition.X + camLookAt.X,
        camPosition.Y + camLookAt.Y,
        camPosition.Z + camLookAt.Z,
        camUp.X, camUp.Y, camUp.Z);

    glShadeModel(GL_SMOOTH);

    // panggil fungsi untuk menggambar obyek
    drawObject();

    glEnable(GL_LIGHTING);
    glTranslatef(0.0f, 0.0f, -30.0f);
    glRotatef((float)timer.GetTime() / 30, 1.0f, 0.0f, 1.0f);
    cubeGrid.DrawSurface(threshold);
    glDisable(GL_LIGHTING);

    cubeGrid.DrawSurface(threshold);

    glutSwapBuffers();
}

```

5. Ubah fungsi `init()` pada `praktikum08.cpp` menjadi seperti dibawah ini.

```

// inisialisasikan variabel, pencahayaan, tekstur,
// pengaturan pandangan kamera dan sebagainya di fungsi init()
void init(void)
{
    // inisialisasi warna latar belakang layar
    // dalam hal ini warna putih warna putih (1.0, 1.0, 1.0, 0.0)
    glClearColor(1.0, 1.0, 1.0, 0.0);
    // mengaktifkan depth buffer
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    // set proyeksi ke proyeksi perspektif
    gluPerspective(fov, 1.0, 1.0, 100.0);
}

```

```

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
// inisialisasi kamera pandang
// kamera berada di posisi (0.0f, 0.0f, 0.0f)
gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

//set up grid
if (!cubeGrid.CreateMemory())
    return;
if (!cubeGrid.Init(gridSize))
    return;

//set up metaballs
for (int i = 0; i<numMetaballs; i++)
    metaballs[i].Init(VECTOR3D(0.0f, 0.0f, 0.0f), 5.0f + float(i));

//Set Up Colors
diffuseColors[0].Set(0.345f, 0.843f, 0.902f, 1.0f);
diffuseColors[1].Set(0.047f, 0.839f, 0.271f, 1.0f);
diffuseColors[2].Set(0.976f, 0.213f, 0.847f, 1.0f);

timer.Reset();
}

```

6. Jalankan program untuk menghasilkan obyek metaball seperti yang ditunjukkan pada Gambar 9.3.
7. Gerakkan dengan menggunakan tombol arah pada keyboard.



Gambar 9.3 Hasil menggambar metaball

## 9.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 09-KK02	CPMK-04	Buatlah 4 buah metaballs dari kode di praktikum 8!	100

## 9.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 09-KK02	CPMK-04	20%		
2.	Praktik	CPL 09-KK02	CPMK-04	30%		
3.	Post-Test	CPL 09-KK02	CPMK-04	50%		
<b>Total Nilai</b>						

## PRAKTIKUM 10: TEKNIK SUBDIVISI

Pertemuan ke : 10

Total Alokasi Waktu : 90 menit

- Materi : 10 menit
- Pre-Test : 10 menit
- Praktikum : 40 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 09-KK02	Mampu memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah
CPMK-04	Mampu menjelaskan dan menerapkan teknik pemodelan 3D, teknik subdivisi, rekonstruksi 3D, dan aplikasi Augmented Reality/Virtual Reality

### 10.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Mahasiswa mampu menjelaskan tentang konsep subdivisi.
2. Mahasiswa mampu menjelaskan tentang teknik-teknik subdivisi.
3. Mahasiswa mampu menerapkan teknik subdivisi dengan OpenGL.

### 10.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

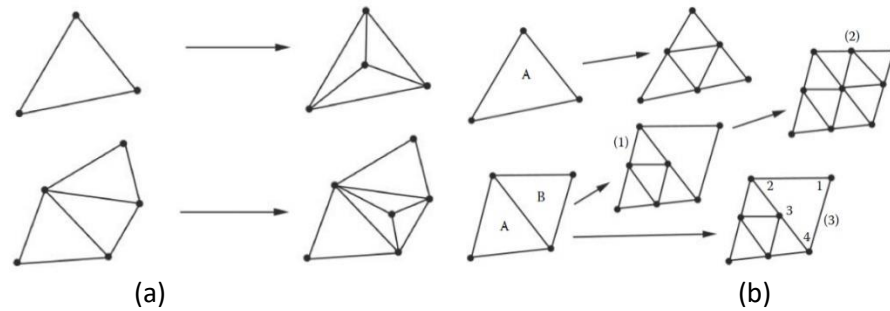
CPL 09-KK02	CPMK-04	Kemampuan mahasiswa dalam menerapkan teknik subdivisi dengan OpenGL.
-------------	---------	--

### 10.3. TEORI PENDUKUNG

Permukaan polygon merupakan tipe permukaan yang paling banyak digunakan di dalam aplikasi computer grafis. Salah satu kelebihanannya yaitu mudah untuk diterapkan subdivisi terhadap poligon. Subdivisi digunakan untuk membuat mesh permukaan menjadi lebih detil dan bagus. Sebagai contoh sebuah bola dapat dimodelkan menggunakan 20 poligon segitiga akan tetapi hasil pemodelan bola menjadi sangat kasar dan sulit dikenali sebagai bola. Apabila diterapkan subdivisi tepi dari 20 poligon tersebut maka akan didapatkan 80 poligon segitiga untuk memodelkan bola. Model bola akan semakin detil dan bagus apabila dimodelkan dengan lebih banyak polygon dipermukaannya.

Permukaan polygon segitiga apabila diterapkan subdivisi akan menghasilkan tiga segitiga baru seperti diilustrasikan pada Gambar 10.1a dan Gambar 10.1b. Subdivisi segitiga bisa diterapkan melalui titik pusat segitiga atau tepian segitiga. Pada tahapan subdivisi polygon segitiga, vertex akan otomatis

ditambahkan dalam struktur data pembentukan sisi permukaan karena tidak ada pengaruh terhadap sisi lain yang berkaitan dengan sisi tersebut.



Gambar 10.1 Subdivisi permukaan segitiga

#### 10.4. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 09-KK02	CPMK-04	Bagaimana cara menerapkan subdivisi 2x pada permukaan segitiga?	100

#### 10.5. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.

#### 10.6. LANGKAH PRAKTIKUM

##### PERSIAPAN

1. Buka Visual Studio C++ dan buat project baru dengan nama **praktikum10**.
2. Download kode dasar praktikum Grafika Komputer dan Library OpenGL seperti pada Praktikum 1.
3. Ubah nama dari kode dasar “praktikum00.cpp” menjadi “praktikum10.cpp” dan copy-kan ke **Source Files** di project yang anda buat.
4. Setting OpenGL library pada Visual Studio C/C++ seperti pada Praktikum 1.

##### PRAKTIKUM

**Aturan Penilaian (Total Skor: 100):**

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 09-KK02	CPMK-04	Selesaikan langkah praktikum 1 – 7	Hasil praktikum langkah 1 – 7	100

##### Langkah-Langkah:

1. Tambahkan header dan variable berikut di praktikum10.cpp untuk inisialisasi pemodelan dengan multiresolusi.

```
#include <math.h>

#define vX 0.525731112119133696
#define vZ 0.850650808352039932
```

```

// vertex data array
static GLfloat vdata[12][3] =
{
    { -vX, 0.0, vZ }, { vX, 0.0, vZ }, { -vX, 0.0, -vZ }, { vX, 0.0, -vZ },
    { 0.0, vZ, vX }, { 0.0, vZ, -vX }, { 0.0, -vZ, vX }, { 0.0, -vZ, -vX },
    { vZ, vX, 0.0 }, { -vZ, vX, 0.0 }, { vZ, -vX, 0.0 }, { -vZ, -vX, 0.0 }
};

// titik-titik segitiga
static int tindices[20][3] = {
    { 1,4,0 }, { 4,9,0 }, { 4,5,9 }, { 8,5,4 }, { 1,8,4 },
    { 1,10,8 }, { 10,3,8 }, { 8,3,5 }, { 3,2,5 }, { 3,7,2 },
    { 3,10,7 }, { 10,6,7 }, { 6,11,7 }, { 6,0,11 }, { 6,1,0 },
    { 10,1,6 }, { 11,0,9 }, { 2,11,9 }, { 5,2,9 }, { 11,2,7 }
};

GLfloat mat_specular[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat mat_diffuse[] = { 0.8, 0.6, 0.4, 1.0 };
GLfloat mat_ambient[] = { 0.8, 0.6, 0.4, 1.0 };
GLfloat mat_shininess = 100.0;

GLfloat light_ambient[] = { 0.2, 0.2, 0.2, 1.0 };
GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_specular[] = { 0.0, 0.0, 0.0, 1.0 };

GLfloat light_position1[] = { 1.5, 1.0, -2.0, 0.0 };
GLfloat light_position2[] = { 1.5, 1.0, 2.0, 0.0 };

int flat = 1;           // 0 = smooth shading, 1 = flat shading
int subdiv = 0;

```

2. Tambahkan fungsi subdivisi berikut pada praktikum10.cpp untuk menerapkan teknik subdivisi pada permukaan obyek.

```

// fungsi untuk melakukan normalisasi koordinat posisi
Vec3 normalize(Vec3 value)
{
    Vec3 result;
    float lengths = sqrt((value.X * value.X) + (value.Y * value.Y)
        + (value.Z * value.Z));
    result.X = value.X / lengths;
    result.Y = value.Y / lengths;
    result.Z = value.Z / lengths;

    return result;
}

// fungsi untuk melakukan operasi perkalian cross
Vec3 cross(Vec3 value1, Vec3 value2)
{
    Vec3 result;
    result.X = value1.Y * value2.Z - value2.Y * value1.Z;
    result.Y = value1.Z * value2.X - value2.Z * value1.X;
    result.Z = value1.X * value2.Y - value2.X * value1.Y;

    return result;
}

// fungsi untuk menghitung normal
void normface(Vec3 v1, Vec3 v2, Vec3 v3)
{
    Vec3 d1, d2;

```



```

    d1.X = v1.X - v2.X; d1.Y = v1.Y - v2.Y; d1.Z = v1.Z - v2.Z;
    d2.X = v2.X - v3.X; d2.Y = v2.Y - v3.Y; d2.Z = v2.Z - v3.Z;

    Vec3 tn = cross(d1, d2);
    tn = normalize(tn);
    glNormal3f(tn.X, tn.Y, tn.Z);
}

// menggambar polygon segitiga dengan sisi normal
void drawTriangleFlat(Vec3 v1, Vec3 v2, Vec3 v3)
{
    glBegin(GL_TRIANGLES);
    normface(v1, v2, v3);
    glVertex3f(v1.X, v1.Y, v1.Z);
    glVertex3f(v2.X, v2.Y, v2.Z);
    glVertex3f(v3.X, v3.Y, v3.Z);
    glEnd();
}

// menggambar polygon segitiga smooth dengan normal
void drawTriangleSmooth(Vec3 v1, Vec3 v2, Vec3 v3)
{
    glBegin(GL_TRIANGLES);
    glNormal3f(v1.X, v1.Y, v1.Z);
    glVertex3f(v1.X, v1.Y, v1.Z);
    glNormal3f(v2.X, v2.Y, v2.Z);
    glVertex3f(v2.X, v2.Y, v2.Z);
    glNormal3f(v3.X, v3.Y, v3.Z);
    glVertex3f(v3.X, v3.Y, v3.Z);
    glEnd();
}

// subdivisi permukaan secara rekursif
// gambar hasil subdivisi segitiganya
void subdivide(Vec3 &v1, Vec3 &v2, Vec3 &v3, int depth)
{
    Vec3 v12, v23, v31;

    if (depth == 0)
    {
        if (flat == 1)
            drawTriangleFlat(v1, v2, v3);
        else
            drawTriangleSmooth(v1, v2, v3);
        return;
    }

    // hitung titik tengah polygon segitiga
    v12.X = (v1.X + v2.X) / 2.0;
    v12.Y = (v1.Y + v2.Y) / 2.0;
    v12.Z = (v1.Z + v2.Z) / 2.0;
    v23.X = (v2.X + v3.X) / 2.0;
    v23.Y = (v2.Y + v3.Y) / 2.0;
    v23.Z = (v2.Z + v3.Z) / 2.0;
    v31.X = (v3.X + v1.X) / 2.0;
    v31.Y = (v3.Y + v1.Y) / 2.0;
    v31.Z = (v3.Z + v1.Z) / 2.0;

    // extrude titik tengahnya
    v12 = normalize(v12);
    v23 = normalize(v23);
    v31 = normalize(v31);

    // subdivisi polygon segitiga secara rekursif

```

```

subdivide(v1, v12, v31, depth - 1);
subdivide(v2, v23, v12, depth - 1);
subdivide(v3, v31, v23, depth - 1);
subdivide(v12, v23, v31, depth - 1);
}

```

3. Ubah fungsi **drawObject()** di praktikum10.cpp seperti dibawah ini.

```

// fungsi untuk menggambar obyek kubus
void drawObject()
{
    glPushMatrix();

    glLightfv(GL_LIGHT0, GL_POSITION, light_position1);
    glLightfv(GL_LIGHT1, GL_POSITION, light_position2);

    // operasi transformasi rotasi obyek ke arah kanan-kiri
    glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah atas-bawah
    glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

    // buat daftar vertex
    for (int i = 0; i < 20; i++)
    {
        Vec3 vdata1 = Vec3(
            vdata[tindices[i][0]][0],
            vdata[tindices[i][0]][1],
            vdata[tindices[i][0]][2]);
        Vec3 vdata2 = Vec3(
            vdata[tindices[i][1]][0],
            vdata[tindices[i][1]][1],
            vdata[tindices[i][1]][2]);
        Vec3 vdata3 = Vec3(
            vdata[tindices[i][2]][0],
            vdata[tindices[i][2]][1],
            vdata[tindices[i][2]][2]);
        subdivide(vdata1, vdata2, vdata3, subdiv);
        vdata[tindices[i][0]][0] = vdata1.X;
        vdata[tindices[i][0]][1] = vdata1.Y;
        vdata[tindices[i][0]][2] = vdata1.Z;
        vdata[tindices[i][1]][0] = vdata2.X;
        vdata[tindices[i][1]][1] = vdata2.Y;
        vdata[tindices[i][1]][2] = vdata2.Z;
        vdata[tindices[i][2]][0] = vdata3.X;
        vdata[tindices[i][2]][1] = vdata3.Y;
        vdata[tindices[i][2]][2] = vdata3.Z;
    }

    glPopMatrix();

    glPopMatrix();
}

```

4. Ubah fungsi **init()** di praktikum10.cpp seperti di bawah ini.

```

// inisialisasi
void init(void)
{
    // inisialisasi warna latar belakang
    glClearColor(1.0, 1.0, 1.0, 0.0);
}

```

```

glEnable(GL_DEPTH_TEST);          // mengaktifkan depth buffer
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
// inisialisasi kamera pandang
gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

// inisialisasi pencahayaan
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT1, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT1, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT1, GL_SPECULAR, light_specular);

glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialf(GL_LIGHT0, GL_SHININESS, mat_shininess);
glMaterialf(GL_LIGHT1, GL_SHININESS, mat_shininess);

glShadeModel(GL_SMOOTH);          // aktifkan smooth shading
glEnable(GL_LIGHTING);            // aktifkan pencahayaan
glEnable(GL_LIGHT0);              // aktifkan sumber cahaya 0
glEnable(GL_LIGHT1);              // aktifkan sumber cahaya 1
}

```

5. Ubah fungsi **keyboard ()** untuk menerapkan rotasi pada obyek dan teknik subdivisi pada obyek.

```

// fungsi untuk mengatur masukan dari keyboard
// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn
void keyboard(int key, int x, int y)
{
    float fraction = 0.1f;

    switch (key)
    {
        // masukkan perintah disini bila tombol kiri ditekan
        case GLUT_KEY_LEFT:
            // dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat
            objectAngleY -= 1.0f;
            glutPostRedisplay(); // update obyek
            break;
        // masukkan perintah disini bila tombol kanan ditekan
        case GLUT_KEY_RIGHT:
            // dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat
            objectAngleY += 1.0f;
            glutPostRedisplay(); // update obyek
            break;
        // masukkan perintah disini bila tombol atas ditekan
        case GLUT_KEY_UP:
            // dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat
            objectAngleX -= 1.0f;
            glutPostRedisplay(); // update obyek
            break;
        // masukkan perintah disini bila tombol bawah ditekan
        case GLUT_KEY_DOWN:
            // dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat
            objectAngleX += 1.0f;
            glutPostRedisplay(); // update obyek
            break;
    }
}

```

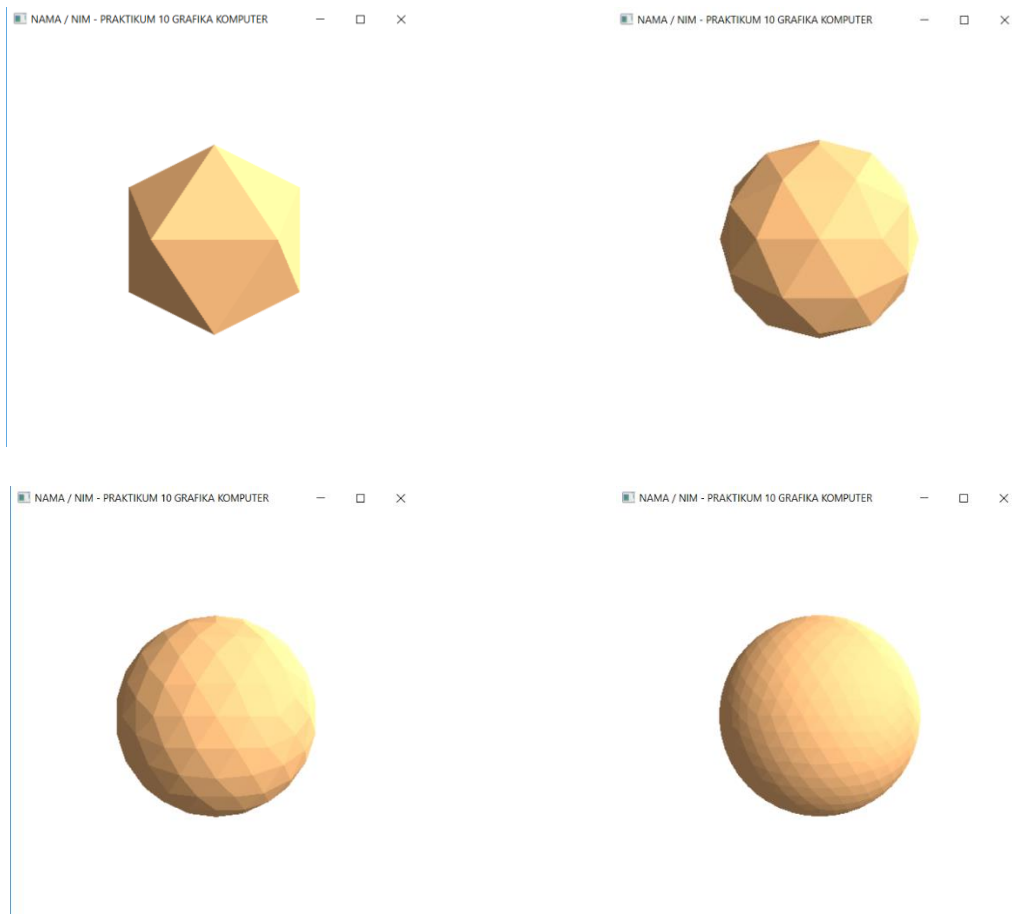
```

// masukkan perintah disini bila tombol PgUp ditekan
case GLUT_KEY_PAGE_UP:
    posX += rotX * fraction;
    posZ += rotZ * fraction;
    glutPostRedisplay(); // update obyek
    break;
// masukkan perintah disini bila tombol PgDn ditekan
case GLUT_KEY_PAGE_DOWN:
    posX -= rotX * fraction;
    posZ -= rotZ * fraction;
    glutPostRedisplay(); // update obyek
case GLUT_KEY_F1:
    subdiv++; // lakukan subdivisi
    glutPostRedisplay(); // update obyek
    break;
case GLUT_KEY_F2:
    subdiv--; // lakukan subdivisi
    glutPostRedisplay(); // update obyek
    break;
}

if (subdiv<0) subdiv = 0;
}

```

6. Jalankan program untuk mendapatkan tampilan icosahedron seperti ditunjukkan pada Gambar 10.2.
7. Gunakan tombol F1 dan F2 untuk menerapkan teknik subdivisi permukaan obyek.



Gambar 10.2 Hasil penerapan teknik subdivisi pada permukaan icosahedron menjadi bola

### 10.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 09-KK02	CPMK-04	Buatlah sebuah obyek torus (gunakan <code>glutSolidTorus()</code> ) kemudian terapkan subdivisi dengan ring 5, 10, 20, 50!	100

### 10.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 09-KK02	CPMK-04	20%		
2.	Praktik	CPL 09-KK02	CPMK-04	30%		
3.	Post-Test	CPL 09-KK02	CPMK-04	50%		
<b>Total Nilai</b>						

**LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM**

<b>Nama :</b> <b>NIM :</b>	<b>Asisten:</b> <b>Paraf Asisten:</b>	<b>Tanggal:</b> <b>Nilai:</b>
-------------------------------	--	----------------------------------

--

## DAFTAR PUSTAKA

1. Edward Angel dan Dave Shreiner, 2011, Interactive Computer Graphics – A Top-Down Approach with Shader-Based OpenGL 6th Edition, Pearson Education
2. Donald Hearn dan M. Pauline Baker, 1997, Computer Graphics C Version 2nd Edition, Pearson Education
3. John F. Hughes, Andries Van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner dan Kurt Akeley, 2013, Computer Graphics Principles and Practice 3rd Edition, Addison-Wesley Professional
4. Dave Shreiner, Graham Sellers, John Kessenich and Bill Licea-Kane, 2013, OpenGL Programming Guide 8th Edition, Pearson Education
5. Edward Angel dan Dave Shreiner, 2014, Interactive Computer Graphics – A Top-Down Approach with WebGL 7th Edition, Pearson Education
6. R. Stuart Ferguson, 2014, Practical Algorithms for 3D Computer Graphics 2nd Edition, CRC Press : Taylor & Francis Group, LLC



**LABORATORIUM  
S1 INFORMATIKA  
FAKULTAS TEKNOLOGI INDUSTRI  
UNIVERSITAS AHMAD DAHLAN**



**2024**