# DIKTAT

## Panduan Penggunaan Dev C++ dan OpenGL

**Praktikum Grafika Komputer**

# KATA PENGANTAR

Segala puji dan syukur kami panjatkan kehadirat Allah subhanahu wa taala, karena berkat rahmat dan hidayah-Nya penulis dapat melaksanakan dan menyelesaiakan penulisan dan Pembuatan **Diktat Panduan Penggunaan Aplikasi Dev C++ v.5.11 sebagai media praktikum mata kuliah Grafika Komputer Fakultas Sains dan Teknologi UIN Sumatera Utara Medan.**

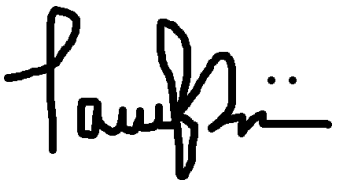
Shalawat berangkaian salam semoga selalu tercurahkan kepada Baginda Nabi Muhammad *Shallallahu 'alaihi wa sallam* beserta keluarga, yang telah membawa kita dari zaman jahiliyah ke zaman yang terang benderang dengan ilmu pengetahuan seperti saat sekarang ini, dan syafaatnya sangat kita harapkan di *yaumil akhir* kelak. Aamiin.

Pada pelaksanaan proses aktualisasi ini, penulis banyak mendapat dukungan moril dan usulan perbaikan serta penyempurnaan dari berbagai pihak. Untuk itu pada kesempatan ini, penulis ingin menyampaikan ucapan terima kasih kepada:

1. Ayah dan Ibu tercinta, serta istri yang telah memberikan doa terbaik tulus dan ikhlas dan memotivasi penulis.
2. Dekan Fakultas Sains dan Teknologi UIN Sumatera Utara, Bapak Dr. Zulham, M.Hum telah memberikan izin dan dukungan luar biasa terhadap penulis.
3. Wakil Dekan Bidang Akademik dan Kelembagaan Fakultas Sains dan Teknologi UIN Sumatera Utara, Bapak Dr. M. Ridwan, M.Ag selaku *mentor* yang selalu membantu proses pembuatan diktat .
4. Ketua Program Studi Ilmu Komputer, Bapak Ilka Zufria, S.Kom.,M.Kom, yang selalu memberikan masukan dan saran proses aktualisasi ini.
5. Dosen Penanggungjawab Keilmuan Mata Kuliah Grafika Komputer, Ibu Sriani, M.Kom.
6. Bapak/Ibu dosen Program Studi Ilmu Komputer UIN Sumatera Utara yang telah banyak membantu penulis berdiskusi dan menyelesaikan diktat ini.

Akhir kata penulis menyadari bahwa tulisan ini masih jauh dari kesempurnaan. Oleh karena itu, sangat diharapkan saran dan masukan yang konstruktif sehingga berguna bagi pembaca. Semoga dapat memberikan manfaat dan memberikan wawasan tambahan bagi para pembaca dan khususnya bagi penulis sendiri.

Medan, 2024



**Ahmad Taufik Al Afkari Siahaan, S.Pd, M.Kom**
**NIP. 199110012022031003**

**LEMBAR PENGESAHAN DIKTAT**

MATA KULIAH : GRAFIKA KOMPUTER
SEMESTER : VI (ENAM)
PROGRAM STUDI : ILMU KOMPUTER
FAKULTAS : SAINS DAN TEKNOLOGI UIN SUMATERA UTARA

DISAHKAN OLEH
TANGGAL : 14 JUNI 2024
DI : MEDAN

MEDAN, 14 JUNI 2024

Mengetahui
Dekan Fakultas Sains dan Teknologi
UIN Sumatera Utara

**Dr. Zulham, M.Hum**
NIP. 197703212009011008

## DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I
# BAHASA PEMROGRAMAN

## I.1. Deskripsi Bahasa Pemrograman

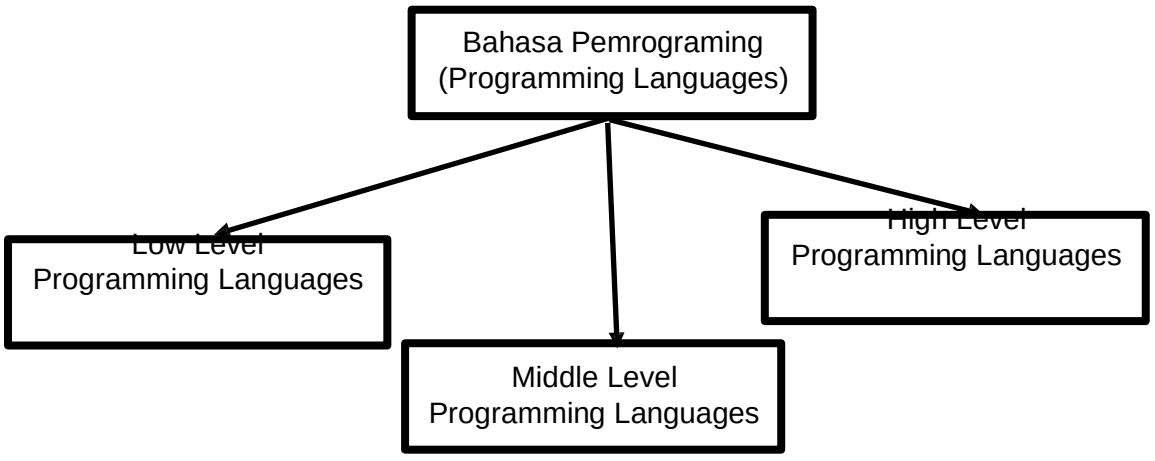
Kehidupan manusia dipenuhi dengan berbagai interaksi satu dengan yang lain, sehingga setiap individu atau kelompok manusia akan melakukan suatu respon atau reaksi yang sesuai dengan aksi yang diberikan. Atas dasar hal tersebut, maka interaksi manusia dengan yang lainnya akan berjalan secara efektif jika informasi yang diberikan oleh manusia satu ke manusia yang lain dapat dimengerti. Sehingga respon yang dihasilkan akan bersifat linier dengan aksi yang diberikan. Hal tersebut dapat didefinisikan sebagai komunikasi. Komunikasi merupakan cara bertukar informasi dalam konteks interaksi yang dilakukan seseorang atau kelompok dengan individu atau kelompok orang lain dengan syarat, informasi yang diberikan dapat dipahami dan dimengerti satu sama lain. Dalam komunikasi terdapat ketentuan dan standar yang digunakan, misalnya saja kesamaan Bahasa yang digunakan, struktur kalimat atau kata serta ucapan yang terdengar jelas dan dapat dipahami serta kemampuan memahami setiap ucapan dan kata yang dapat diterjemahkan secara tepat jika menggunakan bahasa yang berbeda.

Dalam perkembangan zaman, interaksi manusia tidak hanya terjadi dengan manusia yang lainnya, banyak hal yang telah dibuat dan ditemukan untuk mempermudah melakukan pekerjaan, sehingga dalam proses perkembangannya manusia mulai melakukan inovasi untuk memecahkan masalahnya secara cepat, tepat dan efisien. Langkah ini menjadi modal dasar seseorang atau kelompok orang membuat suatu terobosan atau solusi dengan memanfaatkan teknologi. Berawal dari ditemukannya *processor* generasi pertama hingga mengalami perkembangan dan proses transformasi menjadi perangkat yang dikenal dengan istilah komputer dengan *processor* yang memiliki berbagai *brand* seperti Intel, AMD Ryzen dan sebagainya. Maka selama itu juga seseorang atau kelompok-kelompok industri berusaha dan berlomba-lomba mengembangkan dan membuat sebuah media interaksi antara manusia dengan perangkat komputer yang akan digunakan untuk berbagai jenis keperluan.

Media interaksi inilah yang disebut sebagai bahasa pemrograman komputer. Dengan adanya media interkasi ini, manusia dapat berkreasi untuk menemukan solusi dari masalah yang sedang dihadapi atau mempermudah pekerjaannya. Dewasa ini, penggunaan bahasa pemrograman, menjadi hal yang sangat penting. segala hal menjadi sangat bergantung pada penggunaan teknologi yang dihasilkan dengan bantuan bahasa

pemrograman ini. Misalnya saja teknologi yang yang berhubungan dengan sistem keamanan ruang dan perangkat elektronik seperti gawai, komputer dapat menggunakan teknologi *face recognition*, dimana teknologi ini merupakan teknologi yang digunakan untuk mengidentifikasi kesamaan wajah pengguna gawai atau suatu ruangan dengan menggunakan beberapa parameter yang digunakan dalam pengenalan yang secara garis besar dapat dikelompokkan dengan istilah ekstarksi ciri warna, ekstraksi ciri bentuk dan ekstraksi ciri tekstur. Sehingga jika ditemukan kesamaan warna kulit, bentuk wajah dan anggota wajah, serta tekstur dari wajah seseorang cocok dengan sample yang telah disimpan dalam memori perangkat maka gawai dapat melakukan fungsinya seperti biasanya. Namun jika kondisi sebaliknya terjadi, maka gawai akan merespon untuk tidak membuka fungsinya seperti biasanya. Contoh tersebut merupakan salah satu produk hasil dari penggunaan dan implementasi bahasa pemrograman terhadap suatu perangkat dengan fungsi tertentu. Program komputer dapat didefinisikan sebagai kumpulan perintah yang tujukan ke komputer untuk melakukan sesuatu hal sesuai denga napa yang diinginkan oleh operator komputer dalam hal ini disebut *user/programmer.* Hal ini memiliki alas an yang sederhana, bahwa komputer hanya mampu melaksanakan perintah yang diberikan, dan tidak mampu melaksanakan sesuatu tanpa perintah yang tidak dimengerti.

Berbagai jenis bahasa pemrograman yang kita kenal sampai dengan saat ini, memiliki fungsi yang sangat berbeda satu dengan yang lainnya, mulai dari jenis *syntax* dan perintah yang di-input-kan, hingga *brand* dari setiap bahasa pemrograman yang digunakan. Terdapat beberapa jenis bahasa pemrograman yang dikenal sampai saat ini, mulai dari bahasa mesin, *assembly language*, hingga bahasa-bahasa pemrograman yang hampir mendekati bahasa manusia. Klasifikasi tipe dan jenis bahasa pemrograman dapat dibagi dalam 3 (tiga) klasifikasi yang dijabarkan pada Gambar.




Gambar 1. Klasifikasi Jenis Bahasa Pemrograman

Berdasarkan klasifikasi bahasa pemrograman pada Gambar. Maka setiap level memiliki definisi, fungsi dan orientasi penggunaan yang berbeda satu dengan yang lainnya. Jabaran masing-masing level bahasa dapat diuraikan sebagai berikut :

1. Low Level Programming Language

   Sesuai dengan levelnya, bahasa kategori ini memiliki level yang paling rendah, salah satu jenis bahasa jenis ini biasa disebut juga dengan bahasa mesin, dengan karakteristik *syntax* perintah yang digunakan cukup sulit dimengerti oleh *user*. Jabaran penulisan *syntax* jenis bahasa jenis ini hanya berupa 2 (dua) kode bilangan yaitu “0” dan “1” yang dikenal dengan istilah bilangan biner. Dengan definisi bilangan biner 0 mendeskripsikan kondisi *low signal* dan sementara bilangan biner 1 merepresentasikan kondisi *high signal*. Jenis *syntax* ini sering juga disebut dengan istilah *microcode*. Selain itu, *assembly language* juga termasuk kategori jenis bahasa *low level* dengan perintah-perintah dan *syntax* menggunakan istilah-istilah singkat seperti MOV, JUM, SUM, CMP.

2. Middle Level Programming Language

   Bahasa jenis ini, memiliki perintah dan *syntax* yang lebih mudah dipahami, biasanya bahasa pemrograman jenis ini memiliki fitur pemrograman yang cukup mudah dipahami oleh pengguna. Fitur dari bahasa pemrogramannya dapat mengakomodir kebutuhan bahasa pemrograman level tinggi dan level rendah. Atas dasar hal tersebut penggunaan bahasa ini masih populer hingga saat ini. Salah satu contoh dan jenis dari *middle level programming language* adalah bahasa C.

3. High Level Programming Language

   Jenis bahasa yang masuk kategori ini dapat dengan mudah dipahami dan bahasanya mendekati bahasa manusia. Bahasa jenis ini sebenarnya tidak dipahami oleh komputer secara langsung, namun bahasa pada level ini memiliki penerjemah bahasa yang disebut dengan *compiler* atau *interpreter*. Penerjamah ini dapat mentransformasikan bahasa yang *user* input-kan ke dalam bahasa mesin yang memiliki instruksi yang setara. Kemudahan penggunaan bahasa jenis ini, juga didukung oleh *library* serta *class* dengan fungsi tertentu yang dapat dimanfaatkan dalam membangun suatu program di komputer. Contoh dari jenis bahasa high level yang popular adalah Java, .Net, Pascal, Cobol, C++, C# dan lain sebagainya.

## I.2. Perkembangan Bahasa C dan C++

Bahasa C digunakan sebagai media komunikasi antara *user* dengan perangkat komputer dengan memasukkan perintah tertentu, selanjutnya perintah tersebut akan diterjemahkan *compile*r yang , untuk mengetahui apakah perintah yang telah dituliskan sesuai atau tidak didapat didefinisikan oleh komputer. Sehingga hal tersebut dapat diperbaiki sesuai dengan perintah yang dipahami oleh komputer. Bahasa C memiliki kemampuan yang dapat mengakomodir kebutuhan pemrograman level rendah dan level tinggi, sehingga bahasa C dapat diklasifikasikan pada kelas *middle programming language.* Bahasa C dikembangkan oleh seorang ilmuwan yang bernama Dennis Richie pada sekitar tahun 1972. Diawal perkembangannya bahasa C digunakan untuk mendesain system aplikasi dan memprogram jaringan komputer. Selain itu bahasa C digunakan untuk berbagai jenis *operating system* dan arsitektur komputer dengan kata lain, bahasa C dapat diaplikasikan untuk mengontrol seluruh *hardware* yang terinstalasi pada komputer. Dengan perannya yang sangat penting, bahasa C menjadi pelopor dan mampu menjadi *sample* dari bahasa-bahasa pemrograman lainnya terutama bahasa C++ yang merupakan salah satu versi yang berhasil dikembangkan setelah bahasa C.

Bahasa C++ dikembangkan oleh Bjaner Stroustrup pada era tahun 1980-an di laboratorium Bell. Bahasa C++ merupakan bahasa yang masih memiliki hubungan kerabat dengan bahasa C. Sama halnya dengan bahasa C, berbagai fitur yang ada di bahasa C++ dapat digunakan untuk mendukung jalannya komputasi untuk bahasa yang masuk dalam klasifikasi level bahasa rendah ( *low programming language*). Hal ini disebabkan, didalam bahasa C++ memiliki sistem yang efisien untuk men-*support* komputasi bahasa pemrograman dengan level bahasa yang lebih rendah.

## I.3. Persamaan Bahasa C dan C++

Bahasa C yang masih memiliki kedekatan dengan bahasa C++ yang memiliki beberapa kesamaan yang dapat dilihat jelas. Sehingga perintah bahasa C dapat juga digunakan dan diaplikasikan dalam bahasa C++. Kesamaan kedua bahasa ini juga dapat dilihat dari sisi penulisan perintahnya, dimana kedua bahasa ini memiliki sifat *case sensitive*. Istilah ini dapat dideskripsikan dengan berbagai kesalahan yang terjadi dalam penulisan perintah yang secara definsi disebutkan penulisan dan penggunaan huruf kapital dan huruf kecil dalam suatu perintah yang input oleh *user* memiliki arti dan makna perintah yang berbeda. Oleh karena itu, hal tersebut dapat dinilai sebagai suatu kesalahan dalam penulisan program sebab *compiler* mendeteksi hal tersebut tidak masuk dalam kosa kata perintah yang tidak dikenali. Biasanya hal tersebut diawali dengan proses *scanning* yang

dilakukan oleh *compiler*, jika ditemui kesalahan dalam bahasa pemrograman maka selanjutkan *compiler* akan memberikan tanda pada *line* perintah yang salah disertai dengan munculnya notifikasi *warning* pada *script* perintah yang dianggap salah.

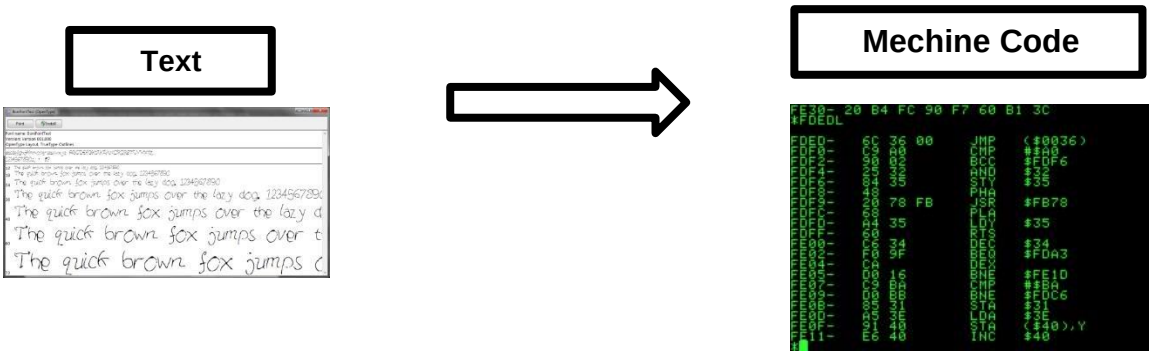
### I.4. Perbedaan Bahasa C dan Bahasa C++

Perbedaan mendasar antara bahasa C dan Bahasa C++ berada pada konsep dan kerangka pemrograman, atau dapat diartikan sebagai perbedaan penulisan kode dan perintah program. Kerangka penulisan bahasa C masuk dalam kategori kompleks, hal ini disebabkan dalam kerangka perograman bahasa C kode perintah yang dituliskan akan dipisahkan ke dalam beberapa bagian fungsi, yang selanjutnya akan digabungkan dalam *main* program. Deskripsi komputasi kerja bahasa C tersebut dikenal dengan istilah *procedural programming.* Konsep *procedural programming* sebenarnya mudah untuk dipahami, namun untuk melakukan komputasi program yang berorientasi dengan objek misalnya saja dalam pembuatan atau membangun sebuah aplikasi, maka hal ini akan memicu munculnya kompleksitas pada penulisan program. Misalnya dalam penulisan kode program untuk membangun sebuah aplikasi tertentu, maka hal tersebut akan membutuhkan kode program yang cukup banyak, sehingga penggunaan variable yang sama pada saat proses inisialisasi variable tidak dapat digunakan dalam struktur kode program pada *line* kode pada fungsi program yang lain.

Dengan konsep kompleksitas yang ditemui pada bahasa C tersebut, maka bahasa C++ muncul untuk menutupi dan mengatasi masalah kekurangan tersebut. Pada bahasa C++ kode program yang dituliskan akan dipisah ke dalam bagian-bagian yang disebut dengan *class* dan *object .* Keunggulan lain dari bahasa C++, *class* dan *object* tersebut tidak akan saling *overlapping*, walaupun memiliki inisialisasi variable yang sama dengan yang lain. Deskripsi kerangka pemrograman bahasa C++ ini dikenal dengan istilah Object Oriented Programming (OOP). Namun proses penulisan pada bahasa C dan C++ harus memiliki ketelitian, hal ini disebabkan kedua bahasa tersebut memiliki tingkat kesensitifan yang tinggi. Biasanya sering kali *user* kurang teliti dalam menyertakan tanda baca “.” (titik), “,” (koma), dan “;” (titik koma) serta tanda operasi matemaika lainnya yang menyebabkan instruksi yang dituliskan tidak sesuai dengan kaidah penulisan instruksi. Hingga akhirnya sesaat setelah dilakukan proses *compile and run*, *compiler* bahasa C atau C++ akan mendeteksi adanya *error* yang ditandai dengan adanya *warning error notification* pada pada IDE yang digunakan. Sehingga akhirnya, instruksi tidak dapat dieksekusi dan dijalankan sesuai dengan rencana yang telah disusun oleh *user.*

## I.5. Penerjemahan Bahasa

Pemrograman komputer memiliki 2 (dua) bagian utama, yaitu data dan instruksi komputer hanya dapat memahami kode yang sesuai dengan instruksi yang dikenalnya, sehingga hal ini akan menjadi kendala utama pada *user* dalam memberikan instruksi dan perintah pada komputer. Akhirnya *user* dengan sangat terpaksa harus menuliskan program dan instruksi yang dipahami oleh komputer, walaupun hal tersebut sangat sulit untuk dipahami oleh *user.* Dengan kata lain, secara tidak langsung komputer memaksa *user* memberikan perintah dan instruksi sesuai dengan instruksi yang dipahami oleh komputer. Hal tersebut memiliki makna yang tidak linier terhadap fungsi komputer yang dapat membantu menyelesaikan, mengorganisir, mempermudah pekerjaan manusia. Atas alasan tersebut, maka bahasa level tinggi akan mengambil peran sebagai jembatan yang mengakomodir interaksi antara *user* dan komputer. Konsep penerjemahan bahasa dapat dilihat pada Gambar.

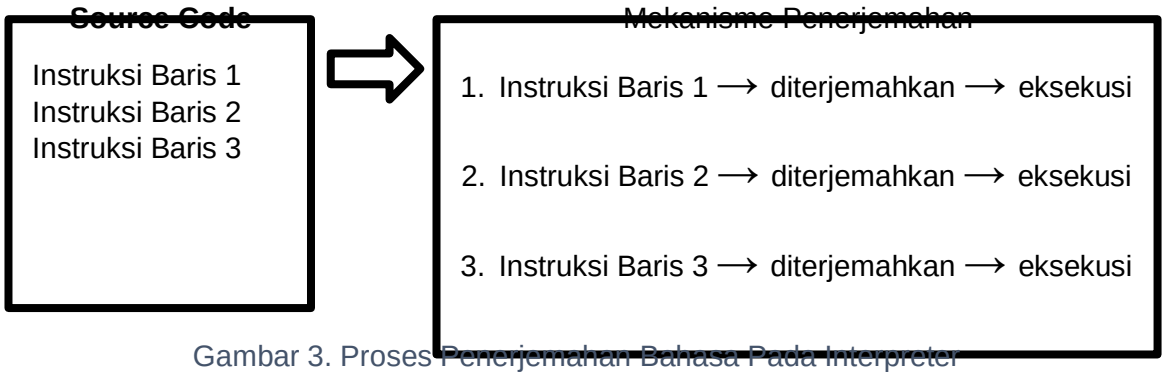



Gambar 2. Proses Penerjemahan Bahasa Instruksi ke Kode Mesin

Penerjemah pada bahasa pemrograman ini, biasanya disebut juga dengan translator. Fungsi utama dari translator ini adalah menerjemahkan setiap baris instruksi yang dituliskan oleh *user* yang menggunakan IDE (Integrated Development Environment) untuk memberikan instruksi tertentu pada komputer. Oleh karena bahasa yang digunakan oleh *user* tidak dipahami oleh komputer, maka translator akan menerjemahkan setiap baris instruksi yang ditulis *user* menjadi level bahasa yang memiliki makna yang dipahami oleh komputer. Terdapat dua jenis translator yaitu

1. *Interpreter*
2. *Compiler*

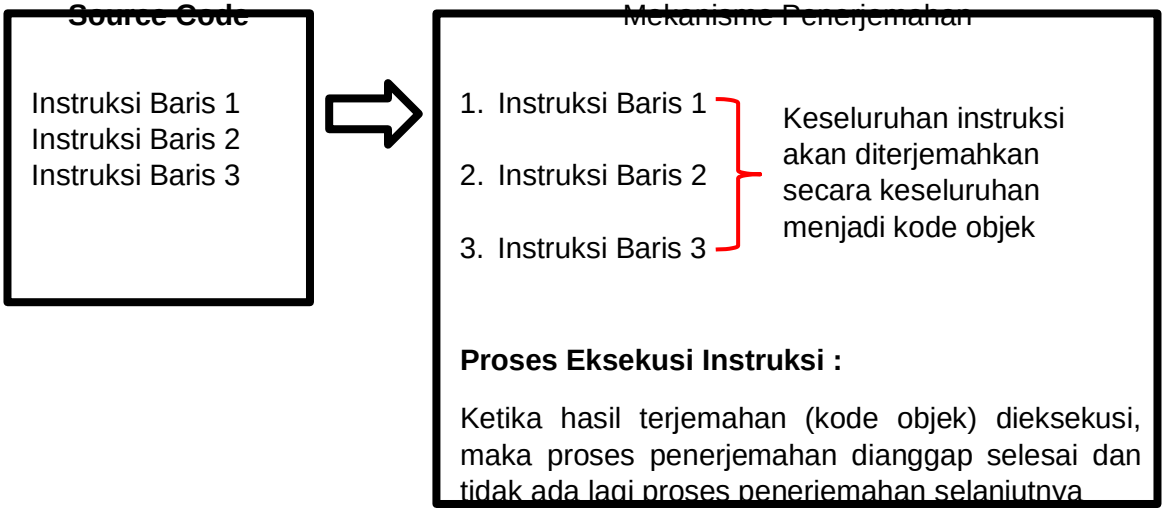
Perbedaan dua jenis translator ini didasari pada mekanisme kerja yang berbeda. Proses penerjemahan bahasa dengan menggunakan *interpreter* dapat dilihat pada Gambar 3.




Gambar 3. Proses Penerjemahan Bahasa Pada Interpreter

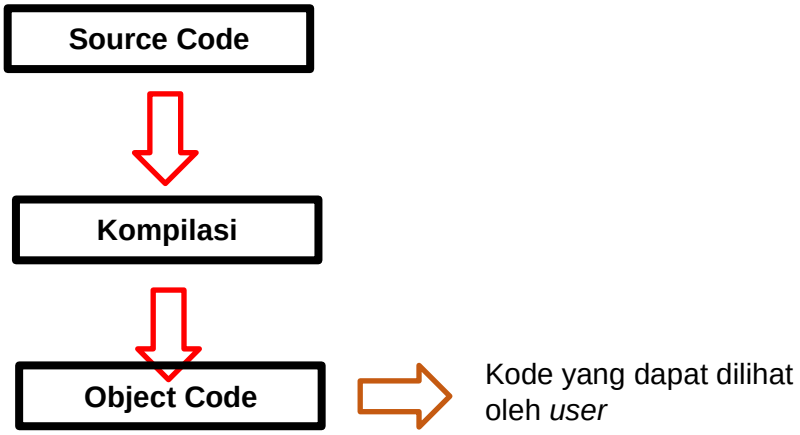
Translator *interpreter* melakukan mekanisme penerjemahan secara bertahap, dimana setiap baris instruksi akan diterjemahkan satu per satu. Proses penerjemahan akan terus berlanjut selama adanya permintaan untuk mengeksekusi instruksi yang diberikan *user.* Saat *user* menghendaki untuk menjalankan instruksi, maka *source code* instruksi yang telah ditulis *user* akan diterjemahkan baris demi baris. Dimulai dari baris pertama instruksi diterjemahkan menjadi kode mesin, setelah kode mesin hasil terjemahan intruksi baris pertama dikenali oleh komputer maka selanjutnya instruksi tersebut dijalankan. Setelah serangkaian proses terjemahan sampai dengan proses eksekusi baris pertama selesai, selanjutnya *interpreter* akan memulai menerjemahkan instruksi baris kedua dengan tahapan yang sama sampai dengan baris terakhir *source code* terakhir yang dituliskan oleh *user*. jika *interpreter* menerjemahkan instruksi baris demi baris dan merubahnya ke kode mesin, mekanisme berbeda ditunjukkan oleh translator *compiler.*

Mekanisme kerja translator *compiler* dapat dilihat pada Gambar 4.




Gambar 4. Proses Penerjemahan Bahasa Pada Compiler

*Compiler* menerjemahkan seluruh instruksi pada *source code* dan mengubahnya ke dalam kode objek. Sesuai dengan penamaannya, *compiler* mekanisme kerjanya menggunakan metode kompilasi. Dimana seluruh kumpulan instruksi akan diubah menjadi kode objek secara keseluruhan dan disusun secara terstruktur. Sesaat setelah proses kompilasi dan proses identifkasi instruksi telah dikenali oleh komputer, *compiler* tidak lagi menjalankan tugas dan perannya. Hal ini disebabkan instruksi yang telah diubah, akan menghasilkan program yang bersifat *excutable* dengan format .exe, yang didefinisikan sebagai program yang dapat dieksekusi secara langsung tanpa melalui translator). Mekanisme proses kompilasi *source code* menjadi *object code* dapat dilihat pada Gambar 5.




Gambar 5.Proses Kompilasi Source Code menjadi Object Code

Agar hasil kompilasi *source code* menjadi *object code* dan dilanjutkan dengan hasil output yang dideskripsikan dalam program *excutable* yang terstruktur, maka pada proses kompilasi terdapat sebuah proses yang disebut dengan *linking*. *Linking* memiliki peran mengumpulkan hasil kompilasi dengan beberapa *library* yang tersedia pada *compiler*. Dengan kata lainnya *linker* akan memanggil *library* yang telah disertakan pada instruksi. Bisasnya *linker* akan memanggil *library* yang telah diinisialisasi pada bagian sisi *header* instruksi yang dituliskan pada IDE. Hal ini menjadikan *object code* sebagai output hasil proses kompilasi yang akan ditampilkan sebagai program aplikasi .exe. Program yang bersifat *excutble* (.exe) dapat menampilkan bentuk pola titik, garis, polygon, bangun datar (dua dimensi), bangun ruang (tiga dimensi) dan beberapa tambahakan efek animasi yang yang dituliskan pada instruksi di IDE yang digunakan. Tampilan program *excutable* inilah yang akan ditampilkan komputer guna memberikan informasi *feedback* kepada *user* sebagai respon instruksi yang dituliskan oleh *user* pada IDE.

# BAB II
# Dev C++

## II.1. Deskripsi Dev C++

Bahasa C++ yang merupakan salah satu contoh bahasa pemrograman yang masuk dalam kategori bahasa level tinggi (*high level programming language)*, memungkinkan seorang *user* dapat berinteraksi dan berkomunikasi dengan komputer secara efisien, hal ini bertujuan agar *user* dapat membuat suatu rancangan sistem dengan fungsi tertentu yang sesuai dengan keinginannya. Bahasa C++ akan memaksakan kehendak dan perintah *user* terhadap komputer untuk mengeksekusi perintah yang diberikan. Konsep pemrograman tersebut dapat berjalan dengan baik pada bahasa C++, dimana bahasa C++ memiliki fitur *compile* yang berfungsi untuk menerjemahkan perintah yang diberikan oleh *user* kedalam bahasa mesin yang dipahami oleh komputer. *Software* komputer yang memfasilitasi *user* dalam menuliskan instruksi yang akan diberikan kepada komputer dikenal dengan istilah Integrated Development Environment (IDE). IDE akan memiliki peran penting dalam menyediakan *utility*, *library*, yang akan sangat membantu *user* dalam menjalankan instruksi yang diinginkan terhadap komputer. Idealnya sebuah IDE setidaknya memiliki 3 (tiga) bagian penting yaitu :

1. Editor : berfungsi sebagai tempat menuliskan *source code* program atau instruksi yang akan diberikan ke komputer.
2. Compiler : berfungsi sebagai penerjemah bahasa C/C++ ke bahasa yang dipahami oleh komputer.
3. Debugger : berfungsi sebagai pengoreksi instruksi yang dituliskan oleh *user*

Ada banyak IDE yang tersedia untuk mengeksekusi instruksi yang dituliskan dalam bahasa C++, diantarnya adalah : Anjuta, Code::Blocks, CodeLite, Eclipse, Geany, Gnat Proggramming Studio, GNOME Bulider, KDevelop, Kuzya, Dev C++, CodeWarrior, MyEclipse, CLion, Visual Studio, Borland C++, Oracle Developer Studio serta masih banyak brand lainnya yang mengeluarkan IDE bahasa pemrograman C++.

Dari sekian banyak brand tersebut, salah satu jenis dari IDE yang digunakan untuk memfasilitasi *user* untuk memberikan instruksi ke komputer adalah Dev C++. Dev C++ merupakan salah satu IDE yang dapat memfasilitasi *user* sebagai operator *software* untuk memberikan perintah dan instruksi ke komputer dengan menggunakan bahasa pemrograman C++ yang dalam prosesnya bahasa yang dituliskan oleh *user* akan

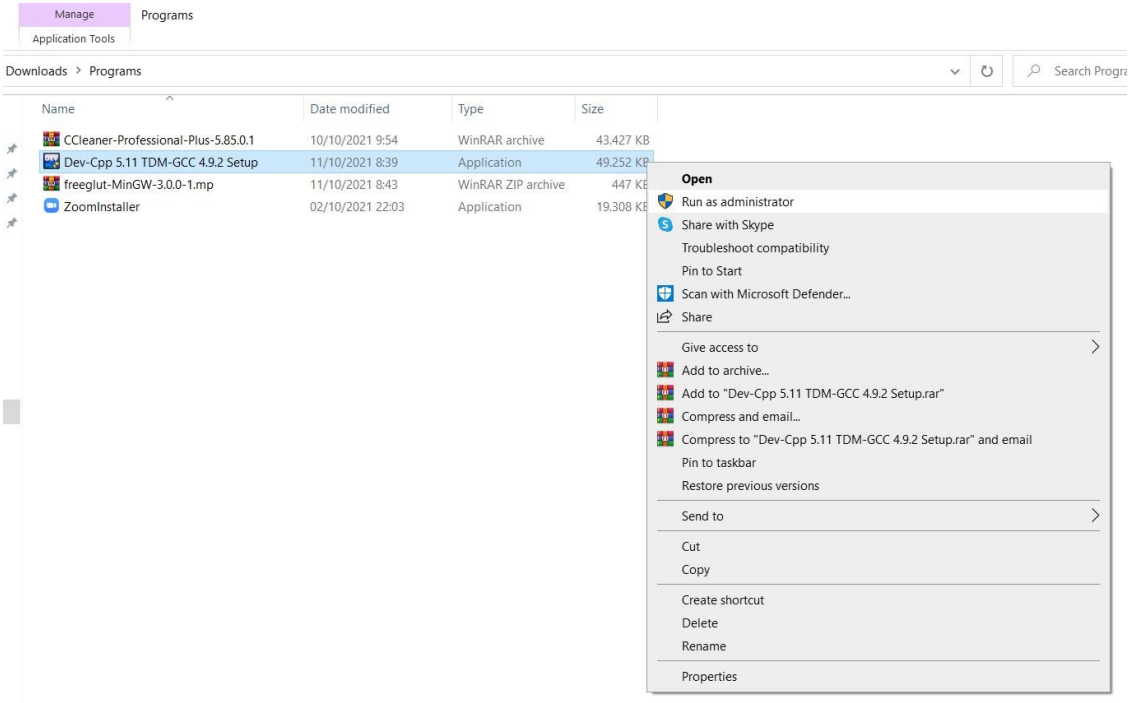
diterjemahkan menjadi bahasa atau *code mechine* yang setara, sehingga dapat dipahami oleh komputer. Agar dapat menyelesaikan berbagai macam materi pada praktikum mata kuliah grafika komputer, maka kinerja dari Dev C++ akan di-*support* oleh OpenGL. OpengL bertindak sebagai penyedia tampilan program *excutable* yang akan merespon instruksi yang diberikan *user* ke komputer. Simbol *shortcut* Dev C++ dapat dilihat pada Gambar 6.


Gambar 6. Simbol Shortcut Dev C++

## II.2. Instalasi *software* Dev C++

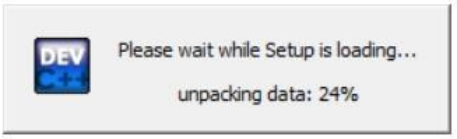
Untuk memulai instalasi Dev C++ maka dapat dilakukan dengan cara berikut :

1. Klik kanan **Run Administrator** / *double clik* pada application Dev C++, visualisasi dapat dilihat pada Gambar 7.



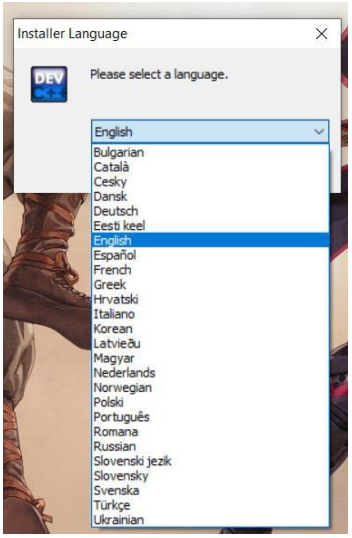

Gambar 7. Persiapan Instalasi Software Dev C++

2. Tunggu proses *unpacking data selesai*



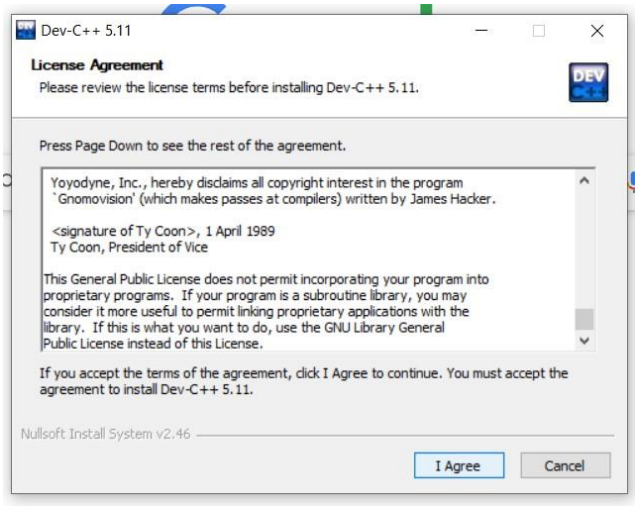

Gambar 8. Proses Unpacking Data pada Proses Instalasi Dev C++

3. Pilih bahasa operasi pada dev C++, pilih “English” lalu tekan Oke



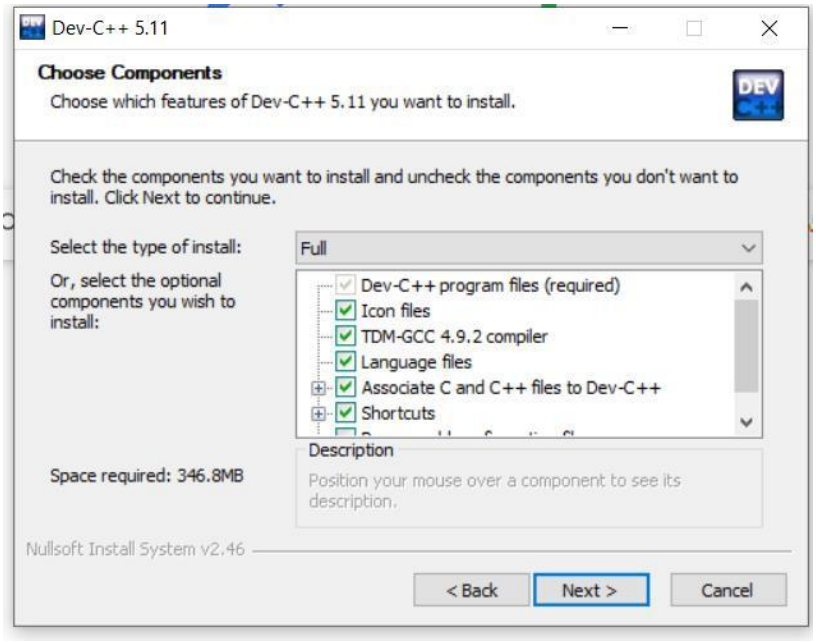

Gambar 9. Tampilan Setting Bahasa Dev C++

4. Klik “ I Agree” pada tahapan License Agreement



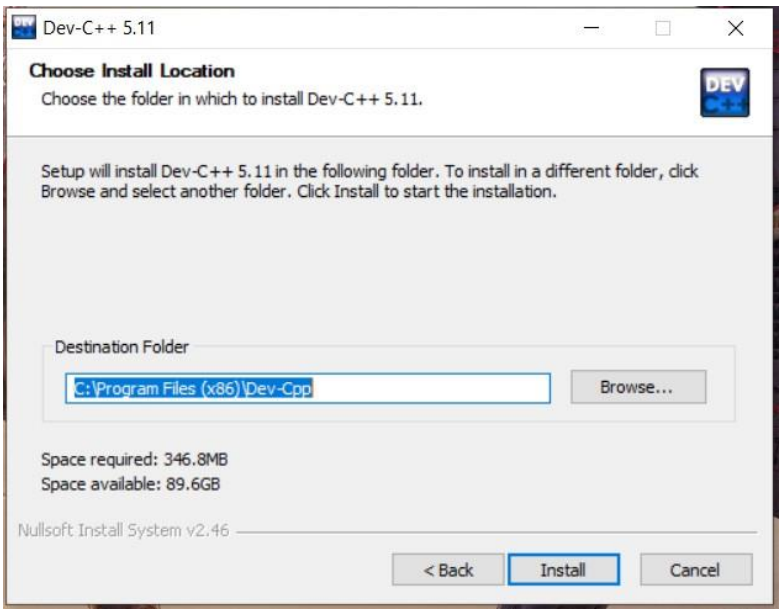

Gambar 10. License Agreement

5. Pilih “ Next >” pada Choose Components



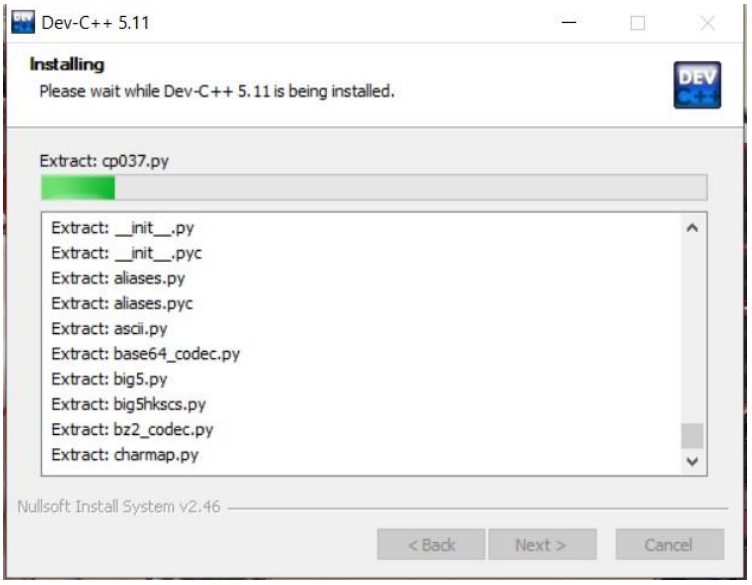

Gambar 11. Pemilihan Type Instalasi Dev C++

6. Pilih destinasi folder instalasi, disarankan mengikuti opsi otomatis yang tertulis, kemudian klik install




Gambar 12. Pemilihan Opsi Folder Instalasi Dev C++

7. Tunggu proses instalasi sampai dengan selesai

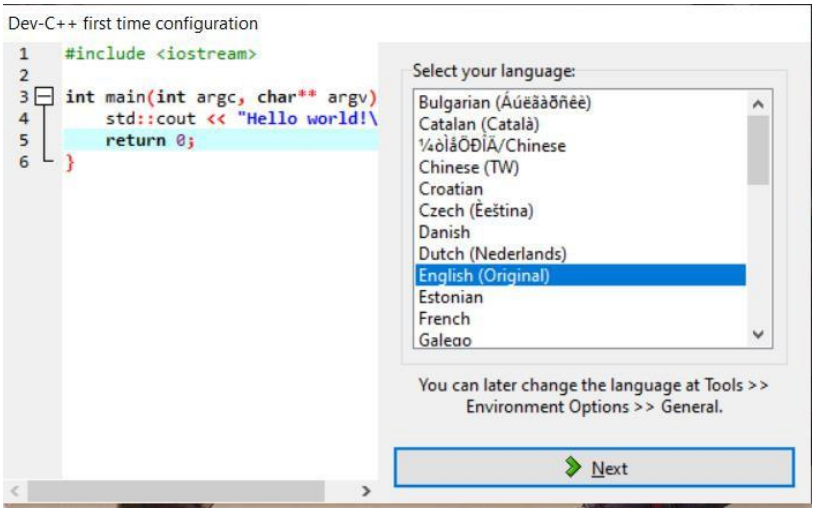



Gambar 13. Proses Ekstrasi File pada Proses Instalasi Dev C++

8. Setelah proses instalasi selesai, klik finish

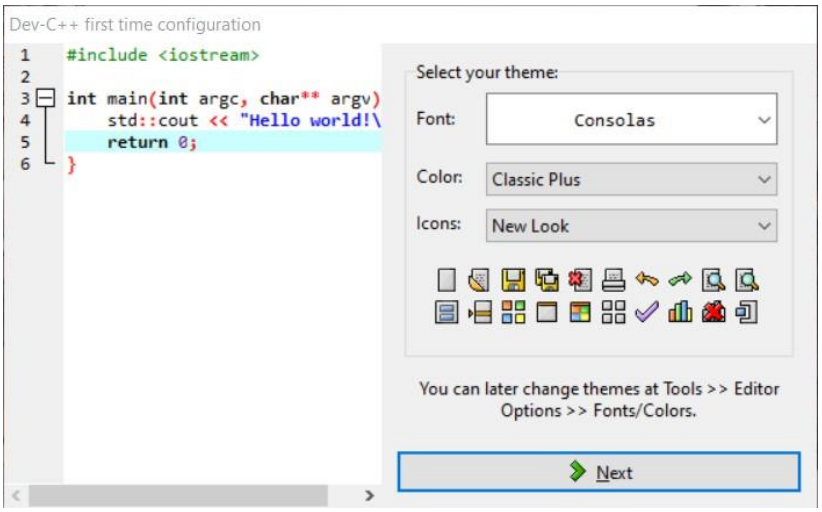

Gambar 14. Notifikasi Penyelesaian Instalasi Dev C++

9. Sesaat setelah proses instalasi selesai, maka akan muncul opsi pemilihan bahasa konfigurasi Dev C++, silahkan pilih “English” kemudian *Next*



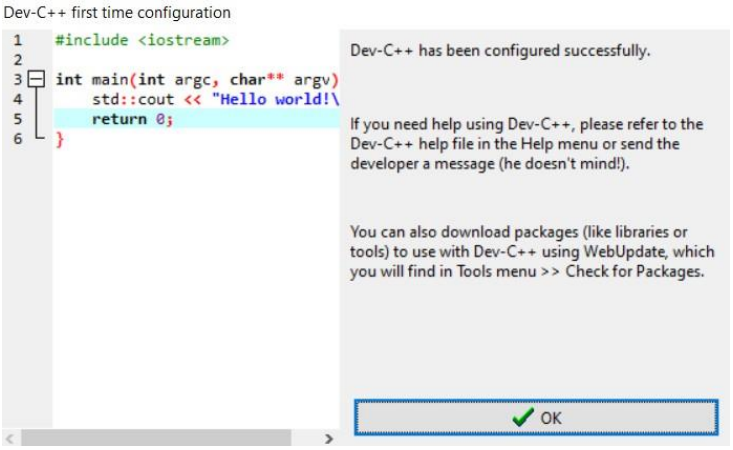

Gambar 15. Pemilihan Opsi Bahasa Operasional Dev C++

10. Proses dilanjutkan dengan melakukan *setting* pada theme display Dev C++, bisa disesuaikan dengan keinginan



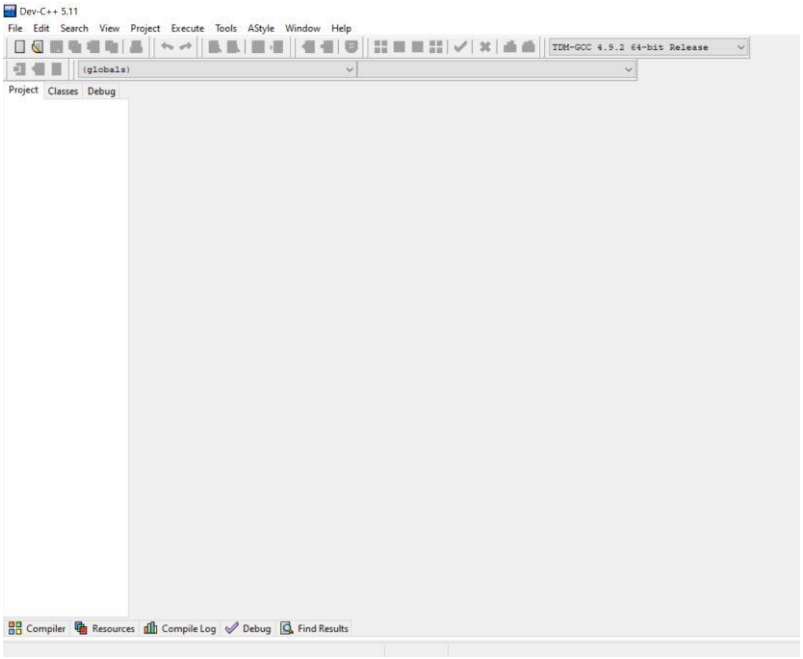

Gambar 16. Pemilihan Opsi Penerapan Tema Tampilan Dev C++

11. Selanjutnya Klik Ok untuk menyelesaiakan tahapan setting Dev C++




Gambar 17. Penyelsaian Setup Software Dev C++

12. Setelah seluruh proses selesai, maka akan muncul tampilan awal dari Dev C ++

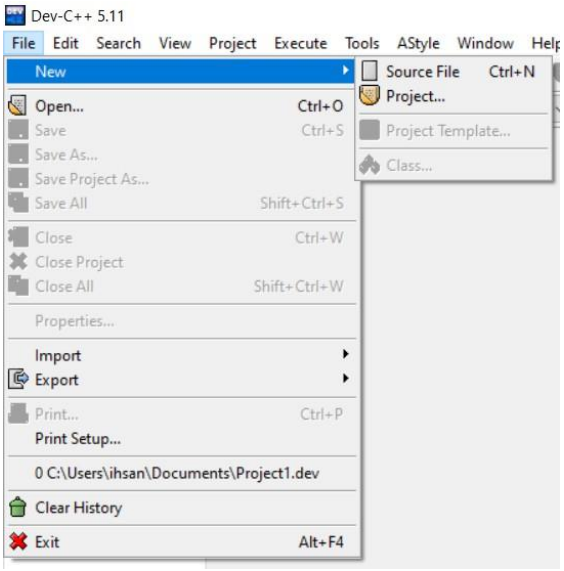



Gambar 18. Tampilan Awal Dev C++

## II.3. Pengenalan Fungsi Fitur Dev C++

Setelah proses instalasi *software* selesai dilakukan, maka tahap selanjutnya adalah mengenal fungsi fitur yang ada pada Dev C++. Sebagian fungsi dari fitur Dev C++ merupakan fungsi yang tidak asing dan dekat dengan keseharian kita misalnya fungsi fitur “File” pada Dev C++ sering ditemui pada fitur Microsoft Office yang digunakan sehari-hari. Fungsi seluruh fitur pada Dev C++ secara umum bertujuan untuk mempermudah pengoperasian Dev C++ dan mendukung kinerja dari Dev C++ dalam melakukan komputasi

program yang dituliskan oleh *user.* Berikut beberapa fitur yang diperlukan dalam pengoperasian Dev C++ yang sering digunakan pada mata kuliah grafika komputer. Menu File dan fungsinya dapat dilihat pada Gambar 19. dan Tabel 1.



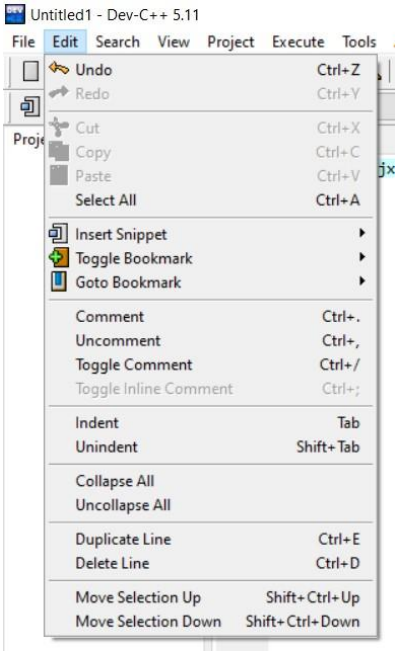

Gambar 19. Menu File

Tabel 1. Fungsi Menu File

| **Fitur Menu File** | | |
|---|---|---|
| **No** | **Nama Menu** | **Definisi dan Fungsi** |
| 1 | **New** | Menambahkan atau membuat lembar kerja Baru<br>Isi Sub Menu :<br>**1. Source File : membuat lembar kerja baru**<br>**2. Project : membuat project baru** |
| 2 | Open | Membuka lembar kerja / text editor yang pernah disimpan sebelumnya |
| 3 | Save | Menyimpan baris program pada lembar kerja yang sedang dibuka ke *default* folder yang telah ditetapkan pada saat instalasi |
| 4 | Save As | Menyimpan lembar kerja dan perangkatnya ke lokasi folder baru dan dapat di-*Rename* dengan *filename* yang baru |
| 5 | Save Project As | Menyimpan project beserta settingan projectnya ke lokasi folder baru dan dapat di-*Rename* dengan *filename* yang baru |
| 6 | Save All | Menyimpan seluruh lembar kerja yang dibuka pada tab Dev C++ |
| 7 | Close | Menutup window lembar kerja |

| | | |
|---|---|---|
| 8 | Close Project | Menutup project yang tidak digunakan lagi |
| 9 | Close All | Menutup seluruh Tab source file yang digunakan |
| 10 | Properties | Menambahkan kelengkapan tambahan pada spurce file yang digunakan |
| 11 | Import | Mengambil source file atau project file pada MS Visual C++ |
| 12 | Eksport | Mengubah dan menyimpan source file dalam format HTML, RTF, TeX |
| 13 | Print | Melakukan pencetakan source file dengan menggunakan printer |
| 14 | Print Setup | Melakukan setting persiapan pencetakan dengan printer |
| 15 | Close | Menutup window Dev C++ |

File Menu pada Dev C++ berisikan fitur-fitur yang memiliki orientasi *set up* awal sebagai persiapan untuk memulai operasi software Dev C++, seperti menyiapkan *text editor, open file,* dan melakukan penyimpanan file hasil kerja serta melakukan operasi perubahan format file serta *setting print* untuk mencetak *source code* yang telah dituliskan. Selain itu, terdapat juga operasi untuk menutup *source code window* yang dibuka secara bersamaan serta operasi *closing software* Dev C++. Seperti yang ditunjukkan pada Gambar 30.




Gambar 20. Menu Edit

Pada Gambar. mendeskripsikan Menu *Edit* berisikan arahan dan perintah untuk mengembalikan *source code* yang di-*edit* ke satu tahap sebelum atau sesudahnya, kemudian terdapat fitur lainnya seperti melakukan duplikasi *line* instruski, menghapus *line*

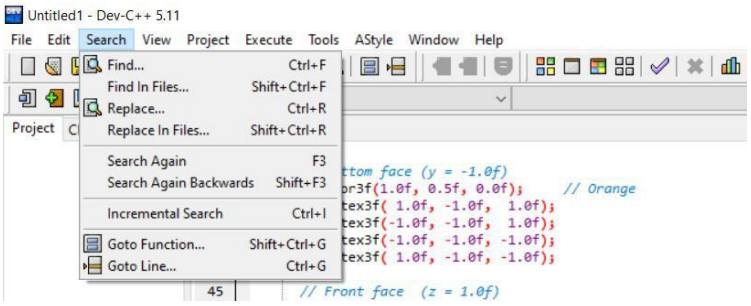
instruksi yang akan salah atau tidak dibutuhkan, kemudian terdapat juga fitur untuk mengaktifkan *command* pada *line* instruksi sebagai tanda meletakkan keterangan pada instruksi yang dituliskan, selanjutnya ada fitur untuk menggeser / *tap line* beberapa kolom ke sisi kanan biasanya digunakan untuk instruksi “if, while, for,” atau beberapa instruksi yang dituliskan untuk menjalankan instruksi iterasi / *looping* serta instruksi pengambilan keputusan. Berdasarkan deskripsi fungsi tersebut, secara pokok menu *edit* mengatur segala keperluan yang dibutuhkan dalam melakukan perubahan instruksi yang dituliskan pada *text editor* Dev C++. Sehingga fungsinya terkadang masih bisa diselesaikan dengan menggunakan fitur *hotkey* pada perangkat PC dan laptop yang digunakan *user.* Selain itu, fungsi dari menu ini juga masih dapat digantikan dengan fitur manual yang bisa dilakukan dengan menggunakan perangkat tambahan seperti *mouse.* Namun, fungsi menu *edit* masih dapat menjadi salah satu opsi yang bisa dipilih oleh *user* dalam hal kenyamanan untuk menuliskan instruksi dan dalam hal mengoperasikan IDE Dev C++. Fungsi dan jabaran menu *edit* dapat dilihat pada Tabel 2.

Tabel 2.Menu Edit

| **Fitur Menu Edit** | | |
|---|---|---|
| **No** | **Nama Menu** | **Definisi dan Fungsi** |
| 1 | **Undo** | Mengembalikan kondisi lembar kerja satu tahap sebelumnya. (Ctrl + Z)<br>**Dapat digunakan berkali-kali sesuai kebutuhan |
| 2 | Redo | Menu invers/kebalikan dari Undo |
| 3 | Cut | *Cutting text* / memindahkan baris coding program yang telah di-*block* sebelumnya. |
| 4 | Copy | Menduplikasi *coding* program ke lebih dari lokasi lembar kerja |
| 5 | Paste | Menempatkan *coding* program yang telah di-*copy* sebelumnya |
| 6 | Select All | Melakukan Blocking area atau proses seleksi pada source file dilembar kerja |
| 7 | Insert Snippet | Menambahkan atau menyisipkan variable waktu, tanggal dsb serta dapat digunakan menyisipkan penambahan perintah while, for dan sebagainya |
| 8 | Toggle Bookmark | Digunakan untuk mengidentifikasi dan melakukan perpindahan ke lokasi tertentu pada source file |
| 9 | Goto Bookmark | Berfungsi untuk masuk ke Bookmark yang telah ditambahkan |

| | | |
|---|---|---|
| 10 | Comment | Membuat baris program tertentu menjadi tidak aktif biasanya diawali dengan tanda baca “//”. |
| 11 | Uncomment | menonaktifkan fitur ekstensi comment |
| 12 | toogle comment | menonaktifkan fitur ekstensi comment |
| 13 | Indent | Membuat teks instruksi pada lembar kerja menjorok kekanan (Fungsinya hamper sama dengan Tab keyboard yang dioperasikan pada Microsoft Word) |
| 14 | Unindent | Membatalkan instruksi indent |
| 15 | Collapse All | Menciutkan/mempersingkat isi instruksi pada lembar kerja |
| 16 | Uncollapse All | Membatalkan perintah Collapse All |
| 17 | Duplicate Line | Melakukan duplikasi instruksi pada satu line tertentu dengan melakukan seleksi / block pada line yang ingin diduuplikasi |
| 18 | Delete Line | Menghapus satu line instruksi |
| 19 | Move Selection Up | Memindahkan satu atau lebih line instruksi pada posisi satu tingkat diatasnya |
| 20 | Move Selection Down | Memindahkan satu atau lebih line instruksi pada posisi satu tingkat dibawahnya |

Deskripsi selanjutnya adalah fungsi menu *search*, menu ini dapat digunakan juga untuk mempermudah *user* dalam melakukan pencarian kata kunci inisialisasi variabel dan instruksi yang dituliskan pada lembar kerja terdiri dari beberapa fitur yang mempermudah *user.* Sesuai dengan namanya, fitur menu ini dapat melakukan *replacing* kata dengan memasukkan *keyword* yang ingin diubah oleh *user*. Sehingga hasilnya *keyword* yang baru akan menimpah dan mengganti instruksi dengan penulisan yang salah atau mengganti penamaan variabel yang telah dituliskan secara berulang. Hal ini sangat mempermudah *user,* khususnya dalam menghemat waktu pencarian dan memiliki efisiensi dalam mengganti instruksi yang ingin di-*replace* secara keseluruhan. Deskripsi fungsi dan tampilan menu *search* dapat diuraikan pada Gambar 21. dan Tabel 3.

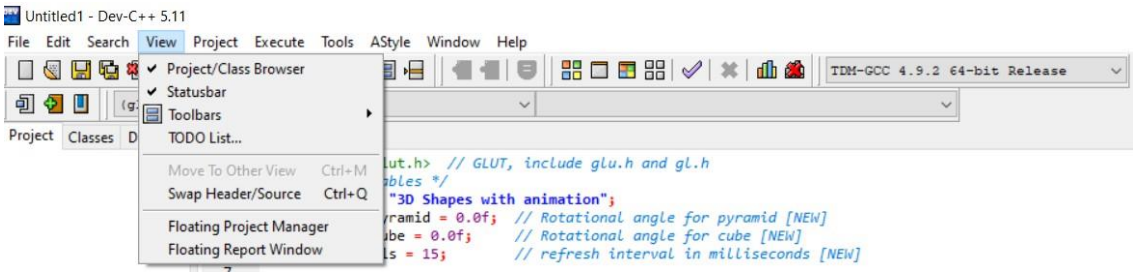



Gambar 21. Menu Search

Tabel 3.Fungsi Menu Search

| Fitur Menu Search | | |
|---|---|---|
| **No** | **Nama Menu** | **Definisi dan Fungsi** |
| 1 | Find | Mencari text *coding* program pada lembar kerja. Biasanya digunakan untuk menemukan variable inisialisasi dan coding program pada lembar kerja. |
| 2 | Fine in Files | Menemukan file dengan menggunakan kata kunci |
| 3 | Replace | Menimpa atau mengganti kata instruksi dengan kata instruksi yang baru |
| 4 | Incremental Search | Memungkian *user* mencari kata dengan tidak melakukan pengetikan secara lengkap |
| 5 | Goto Function | Memungkinkan *user* langsung masuk ke dalam instruksi yang bersifat fungsi |
| 6 | Goto Line | Memungkinkan *user* langsung ke line dilokasi tertentu |

Menu *view* merupakan salah satu fitur yang digunakan untuk melakukan *setting* tampilan menu *bar* pada Dev C++. Hal ini bertujuan untuk memberikan tampilan menu yang sesuai dengan kenyaman *user* dalam mengoperasikan Dev C++. Tampilan menu *bar* dan tampilan ketersediaan menu *editing project* dapat di *hidden*. Hal ini dilakukan dengan tujuan meminimalisir banyaknya menu yang muncul, serta dapat juga mengurasi munculnya menu pada *bar* yang mungkin tidak dibutuhkan oelh *user.* Penyesuaian tampilan menu *bar* dapat mempengaruhi efektifitas *user* dalam memaksimalkan tampilan menu yang hanya dibutuhkan oleh *user* dalam mengoperasikan Dev C++. Disamping itu, menu *view* juga dapat memberikan kesan visual yang baik bagi *user* akibat berkurangnya menu tampilan yang banyak dan bertumpuk dibagian *bar* menu. Serta sebagai langkah antisipasi mengurasi kesalahan dalam memilih opsi perintah yang akan diambil oleh *user.* Uraian deskripsi tampilan dan fungsi menu *view* dapat dilihat pada Gambar 22. dan Tabel 4.

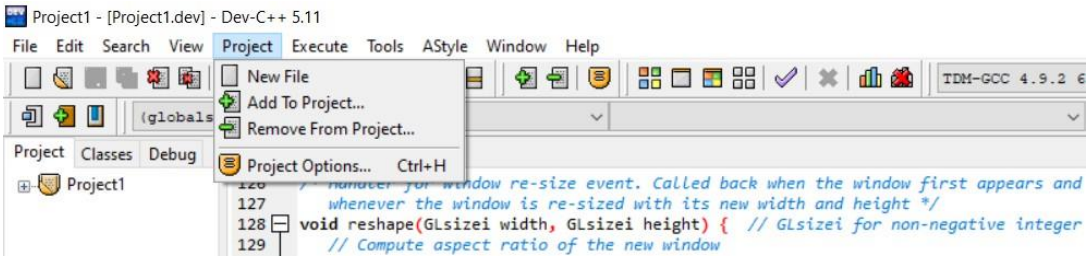



Gambar 22. Menu View

Tabel 4. Menu View

| Fitur Menu View | | |
|---|---|---|
| **No** | **Nama Menu** | **Definisi dan Fungsi** |
| 1 | Project/Class Browser | Memunculkan tampilan project pada sisi kanan lembar kerja (*jika diaktifkan) |
| 2 | Status Bar | Memunculkan tampilan yang status bar pada sisi bawah lembar kerja yang berisi informasi lokasi baris, panjang instruksi, kolom instruksi dan sebagainya |
| 3 | Toolbar List | Memberikan opsi aktif atau tidaknya tampilan menu, project dan sebagainya |
| 4 | TODO List | Memberikan opsi filter pada *user* |
| 5 | Globals | Melakukan proses ascending dan descending berdasarkan urutan abjad dari seluruh variable |

Menu *Project* memiliki deskripsi sebagai penyedia opsi untuk menambahkan atau menghapus fitur *editing* khusus pada bagian *project* yang sedang digunakan. Dapat juga digunakan sebagai opsi bantuan untuk memanggil *project* yang disimpan pada lokasi *folder* penyimpanan tertentu. Pada sisi *setting project*, menu ini dapat digunakan untuk melakukan penyesuaian terhadap kebutuhan *project* yang ingin digunakan *user* untuk mendukung kinerja dari Dev C++. *Setting* parameter pada menu *project* dapat digunakan untuk memanggil *library* dari perangkat tambahan, seperti GLU yang identik dengan OpenGL untuk menghasilkan program yang bersifat *excutable*. Tampilan menu *project* dapat dilihat pada Gambar 23. dan Tabel 5.




Gambar 23. Menu Project

Tabel 5. Menu Project

| Fitur Menu Project | | |
|---|---|---|
| **No** | **Nama Menu** | **Definisi dan Fungsi** |
| 1 | **New File** | Membuat lembar kerja baru |
| 2 | Add to Project | Menambahkan lembar kerja pada project yang telah tersimpan sebelumnya |

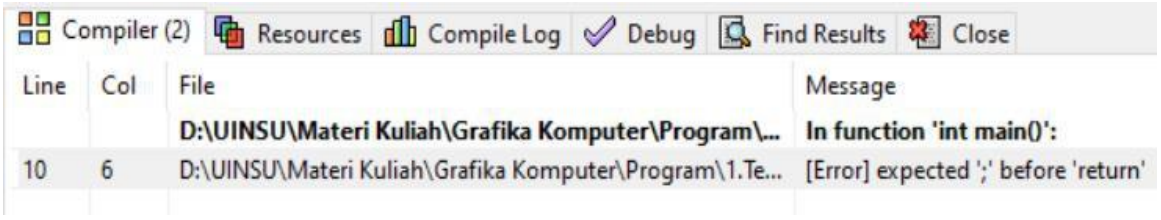
| | | |
|---|---|---|
| 3 | Remove From Project | Menghilangkan atau tidak menautkan source file pada project tertentu |
| 4 | Project Options | Memungkinkan *user* melakukan setting project dengan mengkaitkannya dengan beberapa *library* dari sebuah perangkat tambahan |
| 5 | Make all | Membuat target dari project saat ini |
| 6 | Build all | Membangkitkan kembali seluruh project |
| 7 | Process | Menampilkan sejumlah proses yang sedang dijalankan |
| 8 | Watch | Membuka *window* Watch yang masih merupakan bagian dari proses *debug* |
| 9 | Generate Makefil | Menghasilkan make file dari project yang sedang aktif |

Menu *execute* memberikan opsi kepada *user* untuk melakukan *checking error, cleaning hystori checking error,* serta pilihan opsi dalam proses eksekusi program. Terkait dengan hal tersebut menu *execute* memiliki tiga komponen utama yaitu :

1. *Compile*
2. Run
3. *Compile and Run*

Perbedaan mendasar dari ketiga komponen terletak pada cara kerjanya. Fitur *compile* bekerja hanya dengan melakukan *checking error* saja, tidak termasuk proses eksekusi program. Sementara pada fitur *run* instruksi akan dieksekusi secara langsung dan tidak melalui proses *checking error.* Sementara untuk *compile and run* melakukan pemeriksaan *checking error* terlebih dahulu sebelum instruksi dieksekusi.

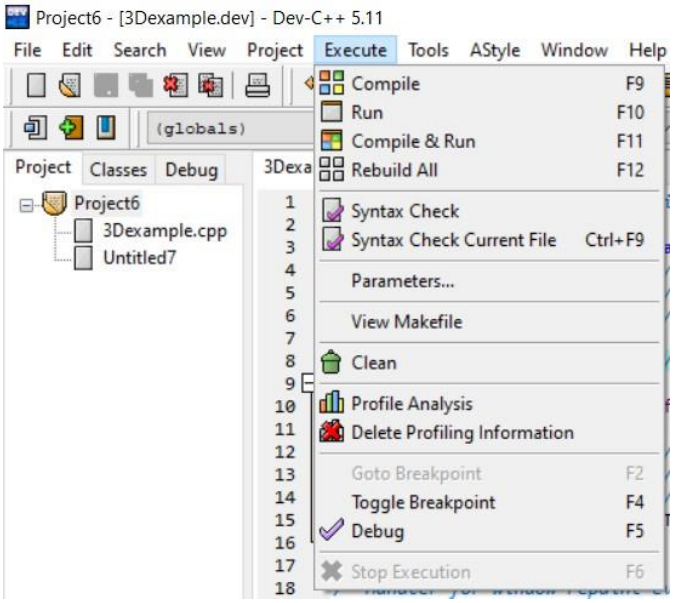
Ketiga fitur tersebut, memiliki mekanisme kerja yang berbeda, namun tetap akan menghasilkan *warning error notification*, jika terdapat kesalahan pada instruksi yang dituliskan oleh *user*. Contoh *warning error notification* dan petunjuk *line* yang dianggap memiliki kesalahan dalam penulisan instruksi dapat dilihat pada Gambar 24. dan Gambar 25.




Gambar 24. Contoh Warning Error Notification


Gambar 25. Contoh line Error Notification

Pada Gambar. dapat dilihat bahwa setelah melakukan proses *compile and run* ditemukan kesalahan pada *line* Sembilan dan keterangan untuk menambahkan tanda baca ";" (titik koma) sebelum baris kesepuluh. Tampilan menu dan fungsi fitur dari menu *executei* dapat dilihat pada Gambar 26. dan Tabel 6.



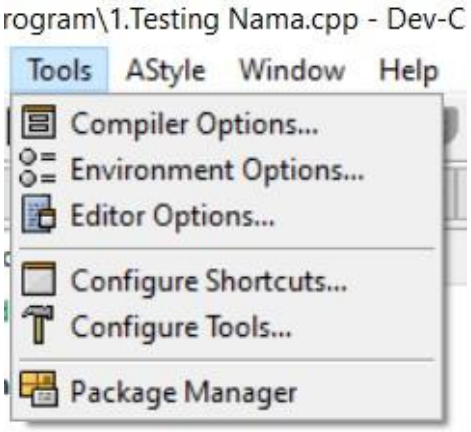

Gambar 26. Menu Execute

Tabel 6. Fungsi Menu Execute

| Fitur Menu Execute | | |
|---|---|---|
| **No** | **Nama Menu** | **Definisi dan Fungsi** |
| 1 | Compile | Berfungsi membaca dan mengubah instruksi yang ditulis pada lembar kerja agar dapat dijalankan menjadi program yang bersifat excutable |

| | | |
|---|---|---|
| 2 | Run | Berfungsi menjalankan instruksi yang telah dituliskan pada lembar kerja |
| 3 | Compile & Run | Kombinasi fungsi kerja *compile and run* |
| 4 | Rebuild All | Melakukan *rebuild* pada program yang sedang digunakan atau sudah tersimpan sebelumnya secara keseluruhan. |
| 5 | Syntax Check | Melakukan *Checking source file* yang sedang digunakan |
| 6 | Syntax Check Current File | Melakukan *Checking source file* yang terakhir kali digunakan |
| 7 | Parameter | Memungkinkan *user* memberikan kelengkapan tambahan dari perangkat tambahan yang digunakan |
| 8 | View Makefile | Memungkin *user* untuk melihat seperangkat identitas kelengkapan yang digunakan |
| 9 | Debug | Melakukan *checking error* dengan memberikan ruang *break* pada saat melakukan proses *run* |
| 10 | Delete Profiling Informastion | Berfungsi menghapus profil pada instruksi |
| 11 | Clean | Menghapus histori pengecekan kesalahan |

Menu *tools* pada Dev C++ berfungsi sebagai pengaturan tambahan yang dapat digunakan untuk menyesuaikan opsi tampilan kepada *user*. Menu ini berisikan pengaturan terkait penyesuaianb tema sesuai dengan keingginan *user*. Serta berisikan informasi tentang penggunaan *hotkey* untuk melakukan operasi tertentu. Uraian tampilan dan menu tools pada Dev C++ dapat dilihat pada Gambar 27. dan Tabel 7.

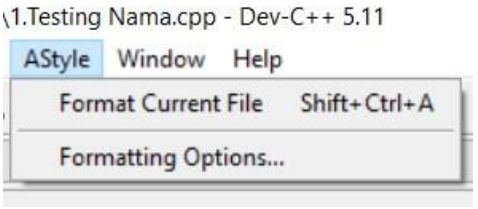



Gambar 27. Menu Tools

Tabel 7. Fungsi Menu Tools

| Fitur Menu Tools | | |
|---|---|---|
| **No** | **Nama Menu** | **Definisi dan Fungsi** |
| 1 | Compiler Options | Memberikan opsi pada *user* untuk mengganti *setting compiler* yang digunakan dan memberikan informasi *library* yang digunakan pada saat ini |
| 2 | Environment Options | berfungsi untuk menetapkan opsi tertentu IDE secara umum |
| 3 | Editor Options | Salah satu fungsi untuk melakukan penyesuaian tema tampilan Dev C++ (*setting font, size font,* dan opsi aktivasi fitur *autosave*) |
| 4 | Configure Tools | Menampilkan informasi konfigurasi dengan menggunakan *hotkey* pada *keyboard*. |
| 5 | Package Manager | Memberikan opsi tambahan kepada *user* terkait adanya *package* lainnya yang akan diinstal |

Menu *Astyle* merupakan tools tambahan yang memberikan opsi kepada *user* untuk melakukan *setting* format penulisan sesuai dengan yang diinginkan. Fungsi dan menu utama Astyle dapat dilihat pada Gambar 28. dan Tabel 8.




Gambar 28. Menu Astyle

Tabel 8. Fungsi Menu Astyle

| Fitur Menu Astyle | | |
|---|---|---|
| **No** | **Nama Menu** | **Definisi dan Fungsi** |
| 1 | Format Current File | Memberikan opsi pada *user* untuk menyesuaikan penulisan symbol dan beberapa komponen lainnya sesuai dengan format yang sudah di-*setting* pada Formatting Options |
| 2 | Fomatting Options | berfungsi untuk menetapkan opsi penulisan instuksi dengan format tertentu |

Menu Window memiliki fungsi pokok terkait pengelolaan atau menajemen *tab window* yang sedang dibuka, baik digunakan untuk berpindah tampilan dari satu *window source file* yang satu menuju *window source file* yang lain dan digunakan sebagai jalan pintas untuk menutup seluruh *window source file* yang sedang dibuka. Atau melakukan *setting* ukuran *window* yang sedang dibuka. Jabaran visual dan fungsi menu *window* dapat dilihat pada Gambar 29. dan Tabel 9.


Gambar 29. Menu Window

Tabel 9. Fungsi Menu Window

| **Fitur Menu Astyle** | | |
|---|---|---|
| **No** | **Nama Menu** | **Definisi dan Fungsi** |
| 1 | Close All | Menutup seluruh *window source file* yang terbuka saat ini |
| 2 | Full Screen Mode | Memungkinkan untuk melakukan setting ukuran *window* dengan tapilan yang penuh |
| 3 | Next | Berfungsi untuk memindahkan tampilan *window source file* pada tab selanjutnya |
| 4 | Previous | Berfungsi untuk memindahkan tampilan *window source file* pada tab sebelumnya. |

## II.3. Dasar – Dasar Penggunaan Dev C++

1. Membuat lembar Kerja Baru

   Dalam penggunaannya Dev C++ memiliki beberapa tahap persiapan, diantaranya adalah membuat lembar kerja. Lembar kerja yang dimaksud adalah ruang yang disediakan untuk menuliskan perintah yang *user* ingin instruksikan ke komputer. Istilah lembar kerja sering juga disebut dengan *text editor*. Berdasarkan fungsi yang setiap menu dan sub menu yang telah dijelaskan

sebelumnya. Maka ada beberapa cara yang dapat digunakan untuk membuat lembar kerja baru. Langkahnya dapat dilihat pada Tabel 10. berikut :

Tabel 10. Langkah Pembuatan Lembar Kerja Baru

| **Membuat Lembar Kerja Baru ( New Text Editor)** | |
|---|---|
| **No** | **Langkah** |
| 1 | Arahkan kursor pada **menu Bar**, Klik **File** → klik **New** → pilih **Source File** |
| 2 | CTRL + N |
| 3 | Klik *icon* **New** pada Tools Menu Bar |

Setelah lembar kerja telah terbuka maka *user* dapat menuliskan instruksi sesuai dengan kaidah dan aturan penulisan pada pemrograman C++ seperti yang ditunjukan pada Gambar.

2. Menyimpan lembar kerja yang berisikan instruksi

   Lembar kerja yang telah dituliskan perintah, dapat disimpan oleh *user* untuk dapat digunakan pada masa mendatang pada *directory file* yang dikehendaki oleh *user.* Hal ini memberikan opsi pada *user* untuk melanjutkan pengerjaan program yang belum selesai dilakukan atau melakuakn pengembangan lebih lanjut pada file yang telah dikerjakan sebelumnya. Langkah dalam melakukan penyimpanan lembar kerja dapat dilihat pada Tabel 11.

Tabel 11. Langkah Penyimpanan Lembar Kerja

| **Menyimpan Lembar Kerja ( save Text Editor)** | |
|---|---|
| **No** | **Langkah** |
| 1 | Arahkan kursor pada **menu Bar**, Klik **Save** → **File Name** (dapat disi sesuai dengan keinginan *user*) |
| 2 | CTRL + S |
| 3 | Klik *icon* **Save** pada Tools Menu Bar |
| 4 | Arahkan kursor pada **menu Bar**, Klik **Save As**→ **File Name** (dapat disi sesuai dengan keinginan *user*) |

Sesaat sebelum menyelesaikan proses penyimpanan, *user* harus memastikan file lembar kerja yang disimpan harus memiliki tipe file C++ sebagai *source file*-nya. Jika hal ini tidak dilakukan, maka file lembar kerja yang akan disimpan akan memiliki

format yang berbeda. Sehingga file lembar kerja yang akan disimpan tidak bisa dibuka atau mengalami kegagalan saat akan dibuka kembali sesaat setelahnya. Pada proses penyimpanan terdapat 2 (dua) sub menu yang digunakan dalam proses penyimpanan file lembar kerja, yaitu **Save dan Save As.** Perbedaan cara kerja kedua Submenu dapat dideskripsikan dengan, submenu **Save** dapat digunakan untuk meng-*update* konten lembar kerja yang berisikan instruksi yang yang telah dituliskan oleh *user* termasuk didalamnya perubahan instruksi berupa penambahan atau pengurangan konten instruksi, perubahan inisialisasi variabel pemrograman, dan lain sebagainya tanpa mengubah *file name* lembar kerja dan lokasi penyimpanan lembar kerja pada *folder* tertentu. Sementara itu, pada submenu **Save As,** *user* dapat melakukan penyimpanan *file* lembar kerja dengan *file name* dan lokasi *folder* penyimpanan yang baru.

3. Membuka file lembar kerja yang telah disimpan

   Lembar kerja yang disimpan sebelumnya dapat dibuka dan digunakan kembali oleh *user* dengan tujuan untuk melakukan perubahan pada instruksi lembar kerja yang telah dibuat sebelumnya. Langkah untuk membuka lembar kerja yang telah disimpan sebelumnya dapat dilihat pada Tabel 12.

Tabel 12. Langkah Membuka Lembar Kerja

| **Membuka Lembar Kerja (Open Text Editor File)** | |
|---|---|
| **No** | **Langkah** |
| 1 | Arahkan kursor pada **menu Bar**, Klik **File** → klik **Open** → pilih file yang ingin dibuka |
| 2 | CTRL + O (kemudian pilih file yang akan dibuka) |
| 3 | Klik *icon* **Open** pada Tools Menu Bar |

4. Melakukan Compile Instruksi (proses penerjemahan bahasa C++)

   Compile dideskripsikan sebagai proses penterjemahan bahasa C++ ke bahasa yang dipahami oleh komputer. Fitur yang melakukan proses penterjemahan pada Dev C++ dapat disebut juga dengan *compiler.* Proses penterjemahan ini juga berfungsi sebagai *checker* kesalahan penulisan instruksi/program yang tidak sesuai dengan kaidah dan aturan penulisan instruksi dalam bahasa C++, serta dapat juga digunakan untuk meilhat variabel instruksi yang tidak terinisialisasi dengan benar. Langkah melakukan proses *compile* instruksi dapat diuraikan pada Tabel 13.

Tabel 13. Langkah Compile Instruksi

| **Compile Instruksi (Compile Program)** | |
|---|---|
| **No** | **Langkah** |
| 1 | Arahkan kursor pada **menu Bar**, Klik **Execute** → klik **Compile** |
| 2 | F9 ( Fn + F9) **tergantung tipe laptop/PC yang digunakan |
| 3 | Klik *icon* **Compile** pada Tools Menu Bar |

5. Menjalankan Instruksi (Running Program)

   *Running* program adalah sebuah mekanisme untuk melihat hasil instruksi yang telah *user* tuliskan sebelumnya. Dengan kata lain *Running* program memfasilitasi *user* untuk melihat hasil kerjanya dengan menampilkan output berupa tampilan seperti hasil komputasi instruksi, bentuk grafik, diagram, bangun datar (2D), bangun ruang (3D) sesuai dengan instruksi yang telah *user* rancang pada lembar kerja. Proses submenu **Run** akan dilakukan setelah instruksi / program yang ditulis oleh *user* telah benar dan sesuai, sehingga perlu dilakukan proses *compile* terlebih dahulu sebelum melakukan proses *running* program. Hal ini disebabkan submenu run bekerja akan langsung melakukan eksekusi terhadap instruksi yang ada di lembar kerja. Hasil dari proses *running program* pada praktikum grafika komputer berupa tampilan file dengan format .exe. Langkah untuk memulai proses **Run** pada Dev C++ dapat dilihat pada Tabel 14.

Tabel 14. Langkah Running Instruksi

| **Run Instruksi (Run Program)** | |
|---|---|
| **No** | **Langkah** |
| 1 | Arahkan kursor pada **menu Bar**, Klik **Execute** → klik **Run** |
| 2 | F10 ( Fn + F10) **tergantung tipe laptop/PC yang digunakan |
| 3 | Klik *icon* **Run** pada Tools Menu Bar |

6. Run dan Compile

   Submenu *run and compile* mendeskripsikan langkah kerja yang merupakan kombinasi submenu *run and compile*, artinya instruksi yang telah dibuat oleh *user* akan melalui proses *compile* terlebih dahulu untuk diperiksa kesesuaian instruksi dan perintah yang telah dituliskan oleh *user*. Jika tidak ditemukan kesalahan maka proses akan dilanjutkan ke proses run instruksi. Kondisi sebaliknya terjadi, jika masih terjadi kesalahan dalam penulisan instruksi, maka

fitur ini akan merespon dengan *warning notification* dan memberi informasi kepada *user* lokasi letak kesalahannya. Langkah untuk menggunakan submenu *run and compile* dapat dilakukan dengan mengikuti petunjuk pada Tabel 15.

Tabel 15. Langkah Compile and Run

| **Run and Compile Instruksi (Run dan Compile Program)** | |
|---|---|
| **No** | **Langkah** |
| 1 | Arahkan kursor pada **menu Bar**, Klik **Execute** → klik **Compile & Run** |
| 2 | F11 ( Fn + F11) **tergantung tipe laptop/PC yang digunakan |
| 3 | Klik *icon* **Compile & Run** pada Tools Menu Bar |

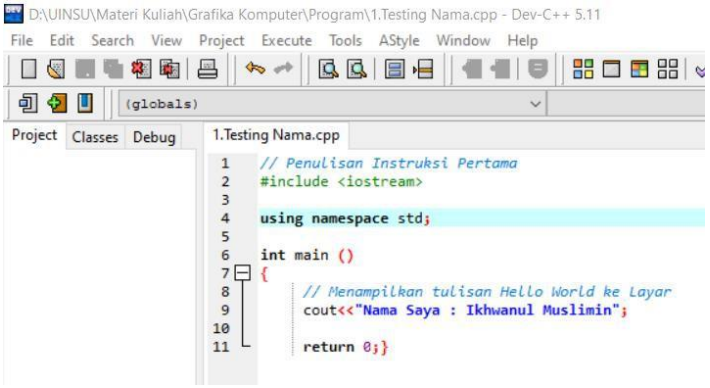
## II.4. Struktur Program Dev C++

Terdapat beberapa bagian utama yang penting dalam struktur pemrograman, hal ini berkaitan dengan identitas yang ingin *user* cantumkan dalam instruksi yang dituliskan.

Beberapa bagian penting tersebut dapat diklasisfikasikan sebagai berikut :

1. Pengarah *compiler*
2. Deklarasi dan Definisi
3. Komentar (Command)

Uraian bagian utama pada *souce file* pada IDE Dev C++ dapat dilihat pada Gambar 30.




Gambar 30. Jabaran Bagian Utama Pemrograman IDE Dev C++

**1. Pengarah Compiler**

Pada Gambar. baris kedua instruksi berisikan “#include <iostream>”, yang dapat dibagi kedalam dua bagian yaitu : “#include” dan “<iostream>” dimana kedua bagian ini termasuk bagian *statement* atau pernyataan. *Statement* “#include” merupakan sebuah *preprocessor directive* yang memiliki fungsi sebagai sebuah perintah untuk melakukan panggilan *file header* yang akan digunakan saat proses *compile and run* berlangsung. *File header* yang dimaksud dalam hal ini adalah “iostream”. *File header* merupakan suatu pernyataan yang untuk penginisialisasi utama dalam suatu program berbasis bahasa C++ yang digunakan untuk mengeksekusi dan memanggil fungsi-fungsi yang terdapat pada *library*. Dengan begitu, saat program dieksekusi maka tujuan pemanggilan fungsi dan *library* file header akan menjadi jelas, sehingga tujuan *user* menuliskan instruksi tersebut dapat dipahami oleh komputer. Ada beberapa contoh yang biasa digunakan sebagai *file header* pada pemrograman C++ diantaranya adalah

1. “#include <iostream>
   Statement ini berfungsi memanggil perintah input dan output seperti cin dan cout
2. “#include <studio>
   Statement ini berfungsi memanggil perintah input dan output seperti prinf dan scanf
3. “#include <math>
   Statement ini berfungsi memanggil perintah operasi matematika, contohnya min, max dan operasi trigonometri (sin, cos, tan)

Proses inisialisasi dalam suatu penulisan instruksi sangat dibutuhkan, hal ini bertujuan untuk mengenalkan *compiler* dengan instruksi yang dituliskan oleh *user.* Jika tidak diinisalisasi terlebih dahulu maka kemungkinan besar akan terjadi *error* pada saat proses *compile and run* berlangsung. Namun hal tersebut tidak berlaku pada *file header.* Setiap perintah input yang dipanggil seperti cout, prinf, atau operasi matematika (sin, cos, tan) tetap dapat dikenali oleh *compiler*. Hal tersebut menunjukkan setiap perintah input tersebut telah diinisialisasi pada *file header* yang digunakan.

**2. Deklarasi dan Definisi**

Secara keseluruhan program C++ memiliki susunan dan tahapan pemanggilan fungsi yang bekerja pada sekelompok data. Selain perintah input dari *file header,* terdapat *statement* lain dari bahasa C++ yang dikelompokkan sebagai berikut :

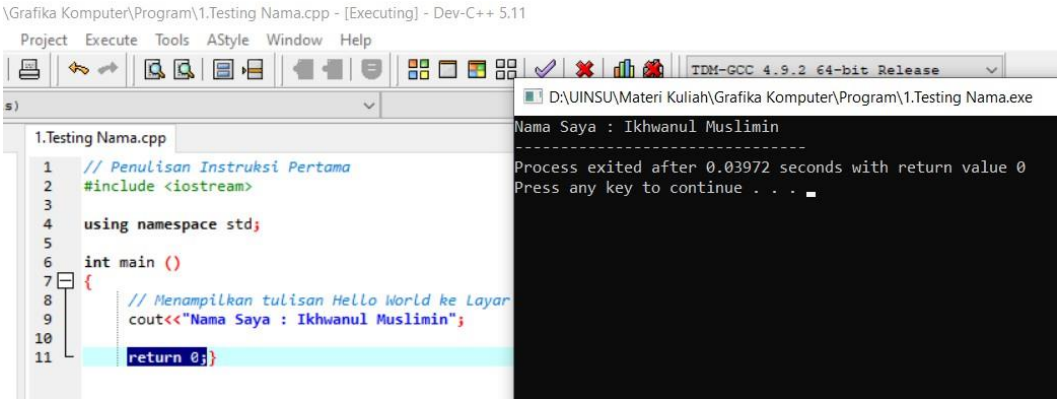
a. Statement yang tidak dapat dieksekusi (*non-excutable*)

*Statement* ini jika dikompilasi tidak akan menghasilkan kode objek sehingga visualisasi pada layar kerja *user* tidak terlihat. Statement jenis ini biasanya digunakan dalam mengatur alur program.

b. *Statement* yang dapat dieksekusi (*excutable*)

*Statement* yang hasil kompilasi menghasilkan kode objek sehingga visualisasi pada layar kerja *user* dapat terlihat sebagai suatu respon komputer terhadap instruksi yang diberikan oleh *user*.

Pada Gambar. *line* instruksi kesebelas yaitu “return 0;”, merupakan salah satu contoh statement yang bersifat *executable* yang mengilustrasikan instruksi respon dari suatu fungsi yang dijalankan untuk mengirim nilai secara visual. Jika instruksi pada *text editor* pada Gambar. dijalankan akan menghasilkan output berupa program *executable* yang ditunjukkan pada Gambar 31.




Gambar 31. Output Hasil Compile and Run Program Executable

Sementara int Main () merupakan salah satu contoh fungsi yang diinisialisasikan agar *compiler* dapat menyesuaikan jumlah *byte* memori yang diperlukan untuk data yang dihasilkan, sementara fungsi perintah selain dari *file header* butub diinisialisasi agar *compiler* dapat dengan tepat melakukan *checking* pemanggilan dari fungsi yang dieksekusi. Disamping itu, fungsi “int main ()”

### 3. Komentar (Command)

Dalam proses pembelajaran sangat diperlukan petunjuk sebagai panduan yang dapat dipahami oleh *user* pemula. Oleh sebab itu pentinganya memberikan komentar untuk setiap *line* yang dituliskan *user* pada *text editor.* Komentar juga dapat digunakan sebagai penanda batas suatu proses pada serangkaian *listing program* yang dituliskan. Umumnya tanda komentar diawali dengan tanda “//” (dua

garis miring). Terdapat dua jenis penulisan komentar yang dikelompokan sebagai berikut :

a. Komentar ( /*…komentar…*/)
   Penulisan komentar jenis ini bertujuan untuk menuliskan komentar yang memiliki panjang lebih dari satu baris.

b. Komentar (//…komentar…//)
   Penulisan komentar jenis ini digunakan untuk menuliskan komentar yang memiliki panjang hanya satu baris.

## II.5. Struktur Pemrograman Dev C++

Pemrograman dengan menggunakan bahasa C++ memiliki struktur yang tersusun. Sehingga alur pemrograman dapat dilaksanakan secara bertahap. Beberapa aturan dalam penulisan seperti peletakan simbol-simbol tanda baca menjadi hal wajib yang perlu ditulis secara teliti sesuai dengan tempatnya. Jika kesalahan terjadi, maka hal tersebut akan mempengaruhi pembacaan struktur pemrograman yang disusun. Penjabaran struktur pemrograman C++ dapat dilihat sebagai berikut :

```
main ()
{
Statement_1;
Statement_2;
……
….
Statement_n;
}
fungsi_lain ()
{
Statement_statement;
}
```

Keterangan :

1. `main ()`dapat disebut juga dengan `tubuh` fungsi, dimulai dari tanda “{“ hingga “}” tanda () digunakan untuk mengapit statement fungsi.
2. `Statement_1-n :`salah satu unsur dasar dalam pembentukan susunan program. Suatu program memiliki beberapa *statement,* dengan fungsi tertentu.

Setelah sususan program tersebut di-*compile and run*, maka komputer akan membaca satu per satu *statement* seusia dengan *list* urutannya.

3. `fungsi_lain () :` Digunakan dalam pemanggilan fungsi, tanda"()" digunakan sebagai tempat penulisan parameter.

Contoh lain dapat dilihat pada skema program berikut ini :

```
include <iostream>
using namespace std;
int main ()
{
}

//Deklarasi
Tipe kembalian, nama fungsi (daftar parameter)
// Badan Fungsi
{
}
```

Keterangan :

1. `include <iostream>` merupakan pernyataan untuk memanggil / import suatu fungsi yang dibutuhkan dalam mengkomputasi
2. `using namespace std;` jenis penyataan yang digunakan untuk seluruh standar penamaan bahasa C++
3. `int main ()` merupakan nama fungsi dengan tipe data integer (bilangan bulat)
4. `Tipe kembalian, nama fungsi (parameter)` dijabarkan sebagai nama fungsi digunakan dalam pemanggilan fungsi, dimana pada proses pemanggilan fungsi harus disesuaikan dengan daftar fungsi. Sebuah fungsi dapat saja memilki daftar parameter yang lebih dari satu, parameter merupakan variabel yang berisikan nilai yang akan disertakan yang akan dalam proses yang dilakuakan dibadan fungsi.
   Tipe kembalian terbagi atas :
   a. Fungsi dengan kembalian
      Fungsi jenis ini memungkinkan *user* mengetahui hasil komputasi dari instruksi yang dituliskan, misalnya dalam operasi matematika penjumlahan, pengurangan, perkalian dan pembagian. Hasil perhitungan operasi matematika

yang diproses komputer melalui fungsi kembalian akan dikirim ke *user* melalui sebuah tampilan. Hal ini ditandai dengan adanya penambahan *statement Return* pada akhir program.

b. Fungsi tanpa kembalian

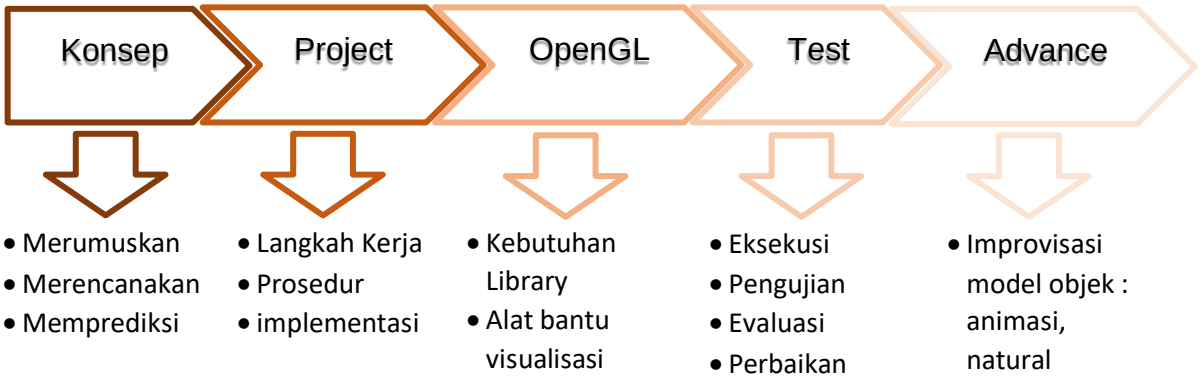
Fungsi jenis ini, tidak beroientasi dengan hasil operasi, melainkan hanya instruksi yang bersifat umum, salah satu contohnya adalah `void, dengan` contoh instruksi `prinf,` yang digunakan untuk mencetak suatu text yang ingin ditampilkan oleh *user.*

5. Badan fungsi merupakan kode atau statement yang akan dijalankan didalam fungsi, yang digunakan untuk menyelesaikan masalah yang bersifat spesifik

# BAB III PEMROGRAMAN GRAFIS

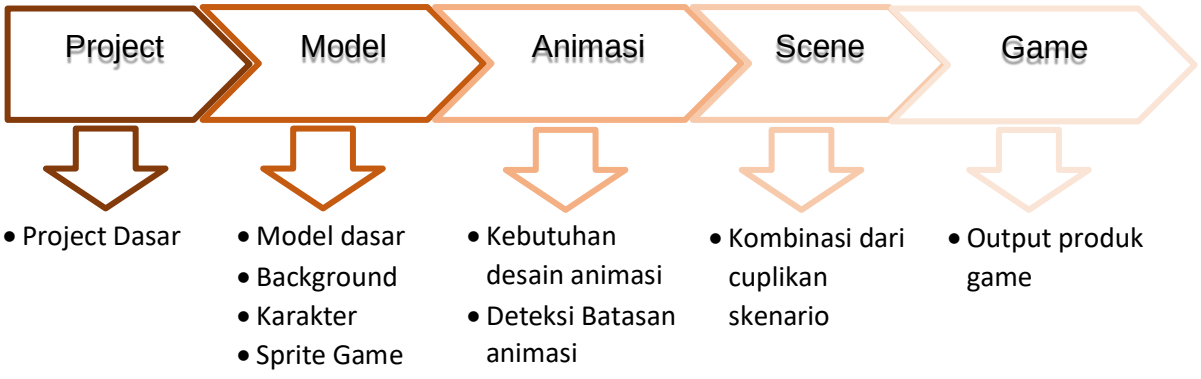
## III.1. Definisi Pemrograman Grafis

Pemrograman grafis merupakan salah satu jenis pemrograman komputer yang memiliki orientasi pada mekanisme pembentukan grafis dengan memanipulasi bagian-bagian tertentu dari sebuah grafik sesuai dengan kebutuhan *user.* Pemrograman grafis banyak dimanfaat untuk kebutuhan simulasi, desain animasi, game, serta berbagai jenis kebutuhan terkait aplikasi visualisasi lainnya. Jenis pemrigraman ini memiliki kebutuhan khusus pada *library* yang dapat membantu menampilkan visualisasi sebagai bentuk respon dari hasil eksekusi instruksi yang telah diberikan sebelumnya. Beberapa *library* grafis yang popular digunakan adalah OpenGL dan DirectX. Langkah-langkah yang dibutuhkan dalam pemrograman grafis adalah sebagai dapat dilihat pada Gambar 32.



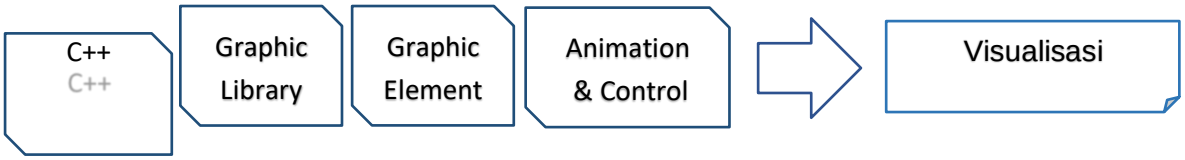

Gambar 32. Konsep Pemrograman Grafis

Sementara jika, *user* mengimplementasikan pemrograman grafis untuk keperluan desain *game* maka contoh scenario pemrograman grafisnya dapat dilihat pada Gambar 33.




*Gambar 33. Konsep Pemrograman Grafis*

Skenario pemrograman grafis dapat dikatakan memiliki peran yang sangat penting dalam dunia desain animasi terutama pada produk media memiliki aspek visualisasi. Skenario pemrograman grafis dapat dilihat pada Gambar 34.




Gambar 34. Skenario Pemrograman Grafis

Deskripsi Gambar. dapat dilihat bahwa seluruh desain objek yang akan dibuat *user* terlebih dahulu dituliskan dalam serangkaian instruksi pada IDE. Selanjutnya fitur *compile* pada IDE akan memerikas kesesuaian dan ketersediaan seluruh instruksi baik fungsi dan *library* yang akan digunakan. Selanjutnya *library* grafis akan dipanggil untuk menyesuaikam isi instruksi yang diberikan dengan mulai merubah instruksi menjadi bentuk-bentuk kode objek serta membaca seluruh elemen grafis melalui kode objek yang tersedia. Tahapan selanjutnya komputer mulai memahai isi instruksi yang ada pada kode objek, yang kemudian mulai melakukan beberapa penyesuaian termasuk didalamnya *setting* efek animasi, durasi animasi dan berbagai komponen grafis lainnya. Hal ini akan dilanjutkan dengan munculnya output *feedback* informasi sebagai respon dari komputer, yang akan memunculkan sebuah aplikasi *executable* dengan bantuan *library* yang digunakan.

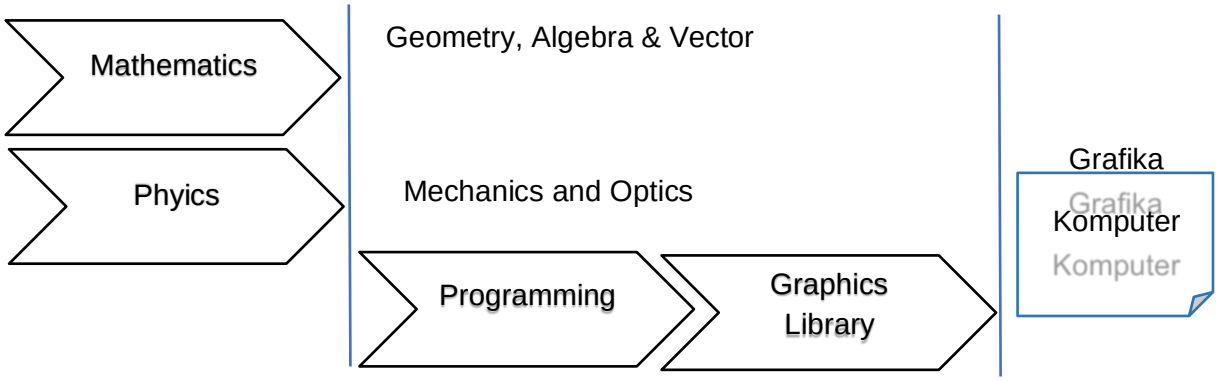
Sebuah *library* grafis idealnya harus memiliki kemampuan untuk memberikan nilai standar grafis yang diperlukan untuk keperluan grafis. Selain itu ilibrary yang digunakan harus memiliki *support* dalam hal ini dapat digunakan ke berbagai jenis *platform*, atau memiliki sifat yang *open source* yang mendukung semua bahasa pemrograman dan IDE. Hal ini akan sangat membantu *user* untuk dapat menggunakan *library* grafis tanpa mengganti *platform* atau IDE yang biasa digunakan.

### III.2. Opengl

Opengl ( Open Graphics Library) merupakan salah satu perangkat tambahan yang digunakan dalam membangun desain sebuah atau lebih bangun datar (2 Dimensi), bangun ruang (3Dimensi) dengan dukungan *software* bahasa pemrograman tertentu. Untuk menjalankan fungsi opengl secara luas, opengl harus diintegrasikan dengan Bahasa pemrograman berbasis C atau C++. Opengl termasuk salah satu jenis dari Application Programming Interface (API), hal ini disebabkan opengl tidak dapat bekerja sendiri tanpa dukungan *software* atau aplikasi Bahasa pemrograman. Selain itu, OpenGL juga dirancang sebagai *interface* yang sederhana yang tidak terikat atau bergantung dengan perangkat

keras dalam pengimplementasiannya, namun OpenGL dapat diimplementasikan ke berbagai *platform hardware*. Dengan demikian, OpenGL harus dioperasikan dalam sebuah *window* yang bekerja melalui sistem *windowing* yang mengontrol *hardware* yang digunakan. Disamping itu, OpenGL tidak memfasilitasi ketersediaan perintah bahasa level tinggi untuk menggambarkan objek tiga dimensi. Namun dengan perintah yang digunakan dalam *window yang terintegrasi* dengan *hardware* akan membantu untuk menggambarkan model tiga dimensi, namun dengan langkah yang cukup kompleks.

Untuk membangun sebuah model sesuai dengan keinginan, *user* dapat membangun satu per satu set kecil seperti : titik, garis, paligon yang memungkinkan *user* membentuk suatu objek yang diinginkan. Disamping itu, *library* juga menyediakan fitur-fitur tertentu yang dapat dibangun diatas OpenGL. OpenGL Utility Library (GLU) merupakan salah satu bagian yang memenuhi standar untuk setiap implementasi OpenGL. Selain itu, di dalam GLU juga tersedia *toolkit* yang berorientasi pada implementasi objek dengan tingkat yang lebih tinggi. GLU juga menyediakan banyak fitur pemodelan yang mendukung implementasi persamaan matematika seperti kuadrat, trigonometri atau kurva serta permukaan bidang tertentu. Desain kebutuhan dasar berdasarkan penjelasan tersebut dapat dilihat pada Gambar 35.



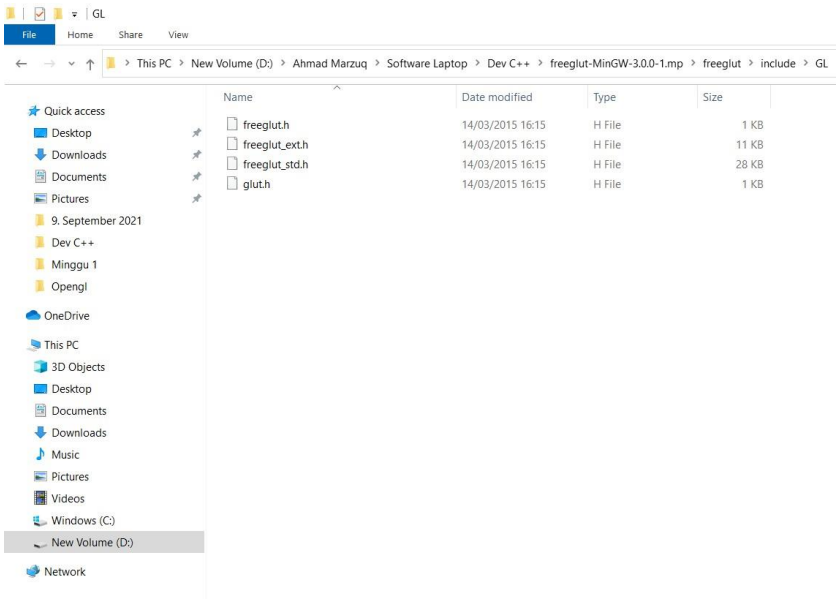

Gambar 35. Kebutuhan Dasar Library

Berdasarkan Gambar. terdapat beberapa penyeseuaian terkait konfigurasi yang perlu dilakukan sebelum menggunakan kombinasi fungsi IDE dan OpenGL sebagai *library* secara langsung. Penyesuaian ini terjadi akibat belum *include*-nya *library OpenGL* dengan IDE yang *user* gunakan. Operasi-operasi matematika khusus yang dijabarkan diatas dapat diperoleh pada *library* OpenGL. Mengingat sistem pembentukan gambar yang dilakukan masih secara primitif, penggambaran objek masih harus disusun satu per satu yang nantinya akan melewati proses kompilasi oleh *compiler* yang kemudian akan digabungkan

Bersama *library* yang dipanggil oleh *linker* pada IDE, dalam hal ini Dev C++.Proses *linker* akan dimulai dengan melakukan *setting* pada parameter *project* IDE dengan menambahkan *library* yang sesuai dengan kebutuhan *user*. Opengl memiliki beberapa bagian yang perlu diinstal atau ditambahkan pada software aplikasi Dev C++ diantaranya adalah :

1. OpenGL Utility Toolkit (GLUT) : Berfungsi sebagai penyedia *interface*, dan digunakan untuk mempermudah *setting* dan mengatur jendela opengl serta menangani *syntax* atau perintah sederhana yang diinputkan oleh *user* aplikasi.
2. File header dengan format (gl.h) : digunakan untuk sebagai penginisialisasi *syntax* untuk memanggil *library* opengl
3. OpenGL Utility Library (GLU) : salah satu bagian dari *library* untuk penerapan dan proses eksekusi dari OpenGl.

Hasil ekstraksi dari file OpenGL dapat dilihat pada Gambar 36.




Gambar 36. Hasil ekstraksi file OpenGL

## III.3. Perintah dalam OpenGL

Penulisan perintah pada OpenGL memiliki kaidah dan aturan yang harus dipenuhi agar seluruh instruksi yang ditulis *user* dapat diekesekusi dengan sempurna. Dalam matakuliah praktikum grafika komputer terdapat beberapa konsep yang paling sering digunakan terutama dalam dalam proses menggambar objek, seperti peletakaan titik koordinat pada sumbu *x* dan *y* yang erat kaitannya dengan objek bangun datar (2D) atau

bahkan tiga titik koordinat yang sering disebut juga dengan sumbu *x, y* dan *z* yang merupakan wadah untuk menggambar objek tiga dimensi (3D). Sehingga penggunaan instruksi yang diberikan harus memiliki kesesuaian dengan kaidah penulisan program OpenGL. Tugasnya adalah menginisiasi status tertentu yang mengontrol OpenGL untuk melakukan menentukan objek yang akan dirender. Render adalah suatu proses yang dijalankan dengan tujuan membangun gambar dari sebuah model yang secara kolektif dengan menyatukan beberapa atau banyak bagian komponen penyusun suatu *image* yang merepresentasikan suatu bentuk objek yang disusun dengan suatu instruksi dari sebuah IDE atau program komputer. Akhir dari proses render terdiri dari piksel yang digambarkan pada layer *display*. Piksel merupakan bagian terkecil penyusun suatu *image* yang dapat dilihat pada *hardware* monitor.

Setiap proses visualisasi hasil *compile* dan *run* IDE membutuhkan proses *linking* yang berfungsi sebagai pengumpul hasil dari proses kompilasi. Maka dari itu, perlu menyertakan *file header* gl.h untuk memanggil GLU untuk dapat menampilkan hasil *running* dari instruksi yang sebelumnya telah dituliskan pada IDE. *File header* akan membantu memanggil *library* yang disertakan untuk memproses eksekusi instruksi yang dituliskan. Sehingga dengan dipanggilnya *library* yang bersangkutan, maka hasil komputasi akan menghasilkan output yang sesuai dengan yang diharapkan. Pada OpenGL penulisan instruksi didominasi oleh kode “gl” yang berarti kode ini harus berada pada *library* GLU OpenGL. Dalam proses *compile* kode “gl” tidak perlu melalui proses deklarasi *statement* atau variabel, hal ini disebabkan seluruh instruksi kode “gl” terdapat pada *file header* “#include <GL/glut.h>”. Namun yang tidak boleh diabaikan adalah mengisi nilai-nilai variabel yang ada pada setiap instruksi kode “gl”. Pengisian nilai tersebut harus dilakukan sebagai langkah penetapan beberapa parameter seperti lokasi menggambar, *setting* ukuran *window* output, penamaan *window* hasil kerja, ketebalan objek yang akan digambar, kode warna objek serta berbagai jenis konfigirasi lainnya. Beberapa instruksi dan definisi instruksi dapat dilihat pada Tabel 16.

Tabel 16. Perintah Dasar OpenGL

| No | Instruksi/Perintah | Definisi | Tipe Data |
|---|---|---|---|
| **1** | glVertex2i(x,y); | Menentukan titik koordinat pada sumbu (*x,y*) | **Integer** |
| **2** | glVertex2f(x,y); | Menentukan titik koordinat pada sumbu (*x,y*) | **Float** |

| | | | |
|---|---|---|---|
| **3** | glVertex3i(x,y,z); | Menentukan titik koordinat pada sumbu (*x,y,z)* | **Integer** |
| **4** | glVertex3f(x,y,z); | Menentukan titik koordinat pada sumbu (*x,y,z)* | **Float** |
| **5** | glClearColour(R,G,B,$\alpha$); | Membentuk warna background objek dengan kombinasi 4 (empat) komponen warna (RGBA) | |
| **6** | glColor3f(R,G,B); | Membentuk warna depan objek dengan kombinasi 3 (tiga) komponen warna (RGB) | |
| **7** | glColor4f(R,G,B,$\alpha$); | Membentuk warna depan objek dengan kombinasi 4 (empat) komponen warna (RGBA) | |
| **8** | glBegin(GL_POINTS); | Titik | |
| **9** | glBegin(GL_LINES); | Garis | |
| **10** | glBegin(GL_LINE_STRIP); | Poligaris | |
| **11** | glBegin(GL_LINE_LOOP); | Poligaris tertutup | |
| **12** | glBegin(GL_TRIANGLES); | Segitiga | |
| **13** | glBegin(GL_TRIANGLE_STRIP); | Segitiga | |
| **14** | glBegin(GL_TRIANGLE_FAN); | Segitiga | |
| **15** | glBegin(GL_QUADS); | Segiempat | |
| **16** | glBegin(GL_QUADS_STRIP); | Segiempat | |
| **17** | glBegin(GL_LINE_STIPPLE); | Garis putus-putus | |
| **18** | glBegin(GL_POLY_STIPPLE); | Poligon dengan pola tertentu | |

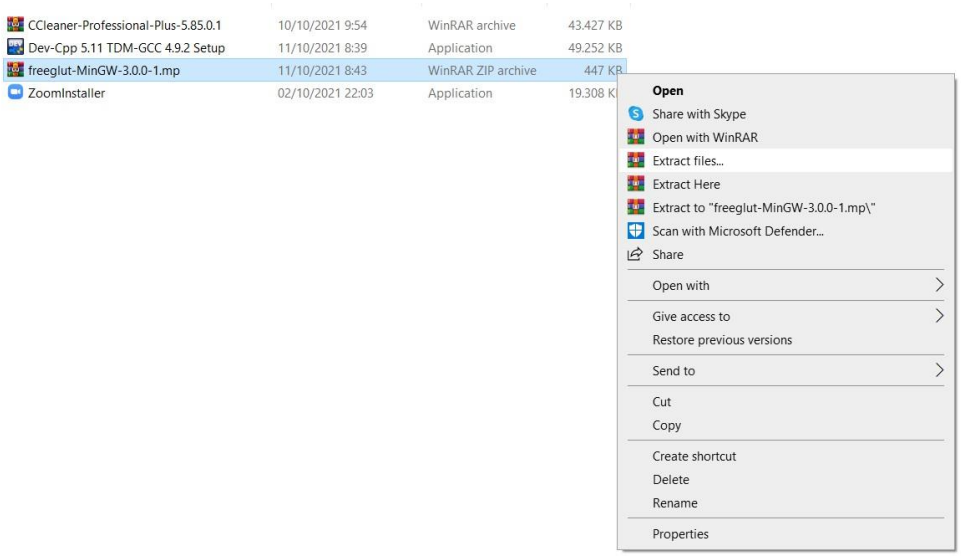
Beberapa perintah Dev C++ dan OpenGL memiliki jabaran dan uraian tipe data yang merepresentasikan jenis nilai. Beberapa jenis tipe data pada bahasa C++ dapat diuraikan sebagai berikut :

1. Tipe data Integer : Tipe data yang digunakan untuk merepresentasikan suatu nilai dengan besaran nilai bulat (contoh : 1, 2, 3,…10….dst)
2. Tipe data Float : Tipe data yang digunakan untuk merepresentasikan suatu nilai dengan besaran nilai desimal ( 3.14, 7.24, dst)

3. Tipe data Boolean : Tipe data yang berorientasi pada keputusan benar atau salahnya suatu *statement*. Jabaran nilai jenis ini berisikan nilai *true* atau *false*
4. Tipe data Char : Tipe data yang digunakan untuk menuliskan paling sedikit satu karakter huruf. (A – Z, %)

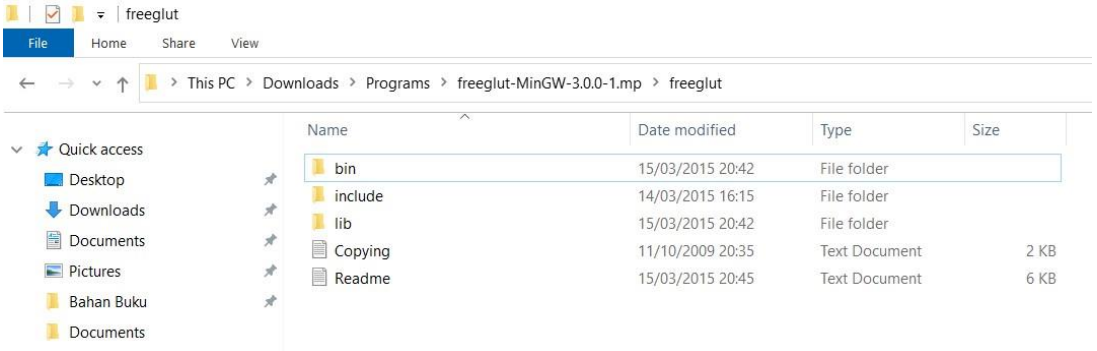
## III.4. Memulai Tahapan Konfigurasi Dev C++ dan OpenGL

1. Close window Dev C++ yang telah terbuka,
2. Lakukan ekstraksi pada file OpenGL (freeglut-MinGW-3.0.0-1.mp)



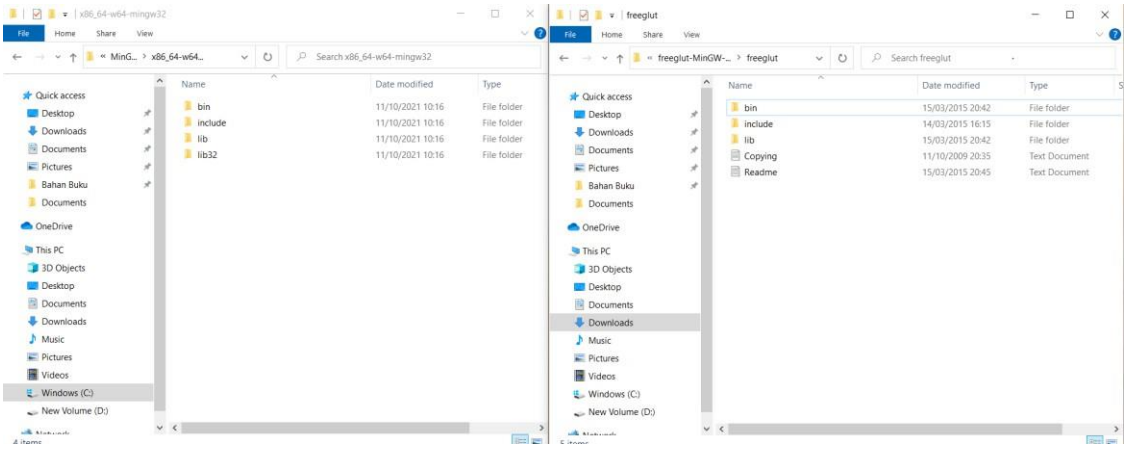

Gambar 37. Proses Ekstraksi File OpenGL

3. Buka directory file : \freeglut-MinGW-3.0.0-1.mp\freeglut



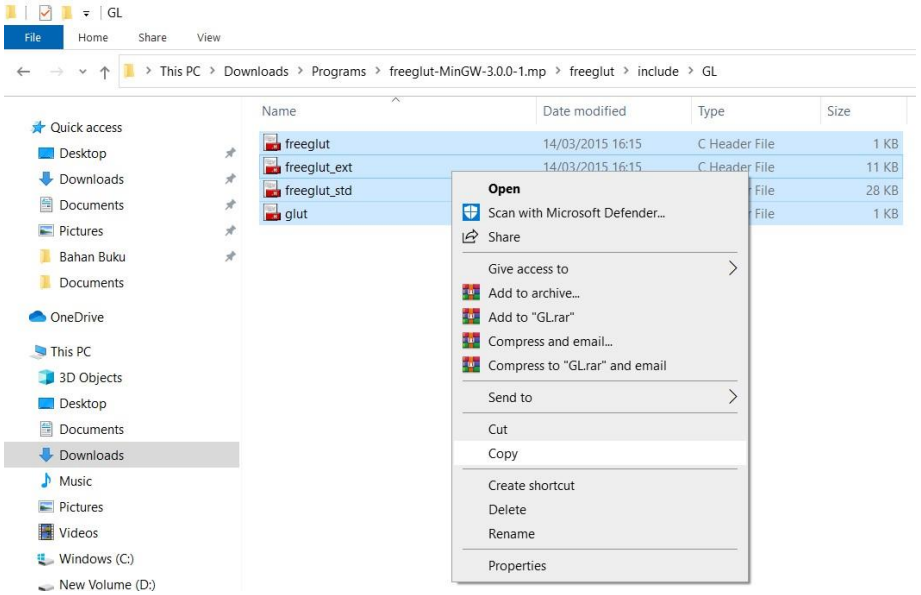

*Gambar 38. Hasil Ekstrasi File OpenGL*

4. Buka widows untuk masuk ke directory instalasi Dev C++ dengan mengakses : C:\Program Files (x86)\Dev-Cpp\MinGW64\x86_64-w64-mingw32



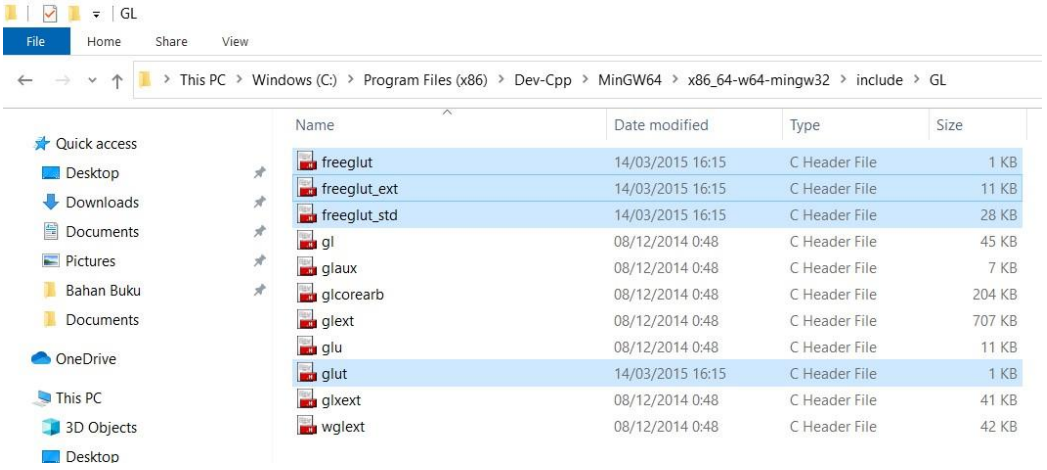
Gambar 39.Jabaran Tampilan Folder instalasi IDE Dev C++ dan OpenGL

5. Buka directory : C:\Users\ihsan\Downloads\Programs\freeglut-MinGW-3.0.0 1.mp\freeglut\include\GL copy seluruh file



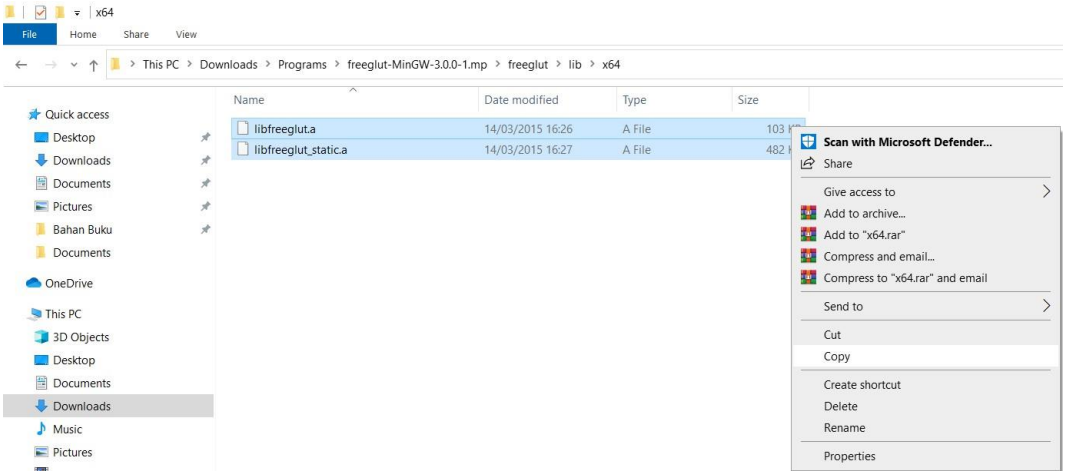

Gambar 40. Proses Duplikasi dan Replace Data OpenGL Bagian GL

6. Pastekan file tersebut ke : C:\Program Files (x86)\Dev-Cpp\MinGW64\x86_64-w64-mingw32\include\GL



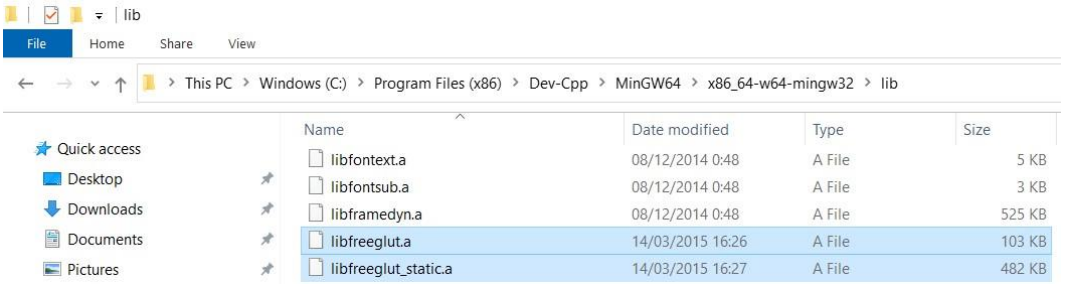

Gambar 41. Hasil Duplikasi dan Replace Data OpenGL Bagian GL

7. Selanjutnya akses : C:\Users\ihsan\Downloads\Programs\freeglut-MinGW-3.0.0-1.mp\freeglut\lib\x64 copy seluruh file



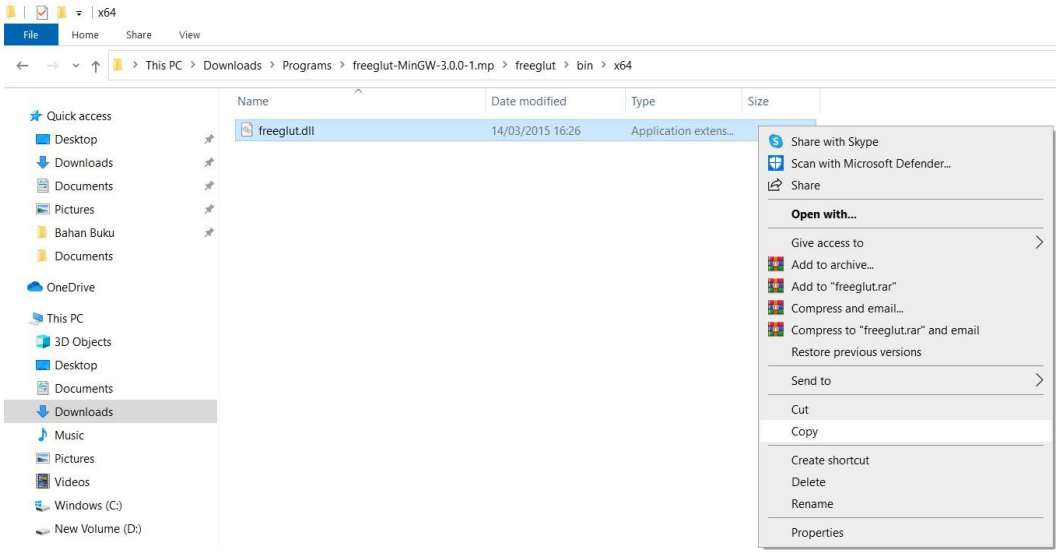

Gambar 42. Hasil Duplikasi dan Replace Data OpenGL Bagian Lib

8. Pastekan file yang di copy ke : C:\Program Files (x86)\Dev-Cpp\MinGW64\x86_64-w64-mingw32\lib



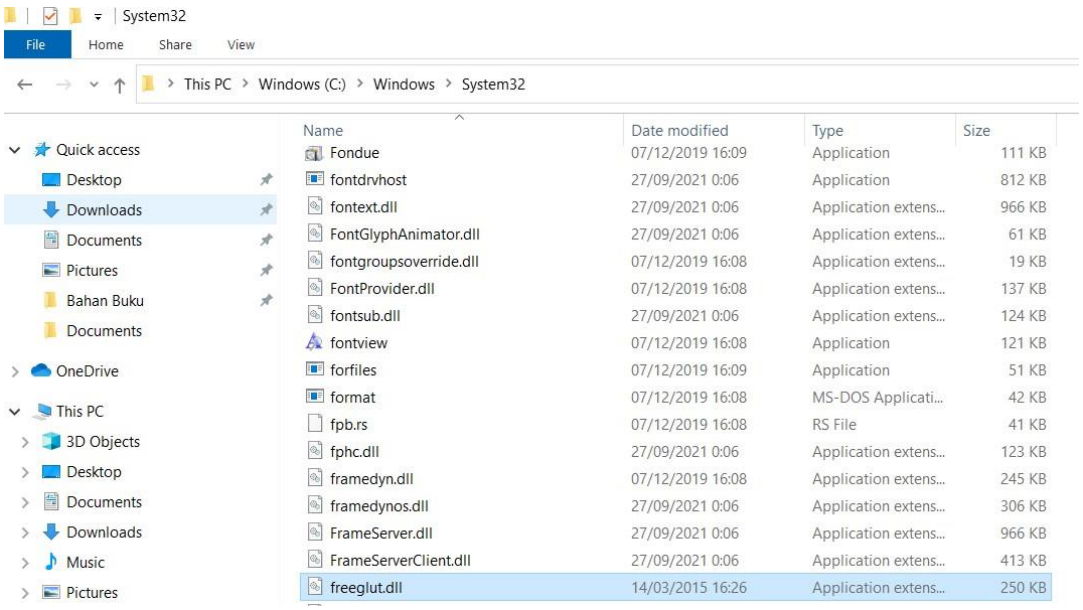

Gambar 43. Hasil Duplikasi dan Replace Data OpenGL Bagian Lib

9. Akses folder C:\Users\ihsan\Downloads\Programs\freeglut-MinGW-3.0.0-1.mp\freeglut\bin\x64 copy file “ freeglut.dll”




Gambar 44. Proses Duplikasi dan Replace Data OpenGL Bagian bin

10. Pastekan ke lokasi folder : C:\Windows\System32

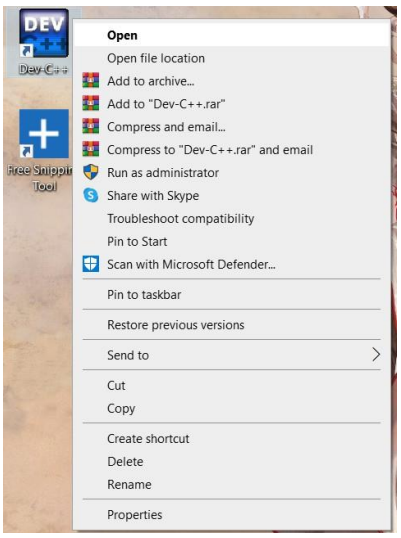



Gambar 45. Hasil Duplikasi dan Replace Data OpenGL Bagian Lib pada System 32

### III.5. Memulai Dev C++ dan OpenGL

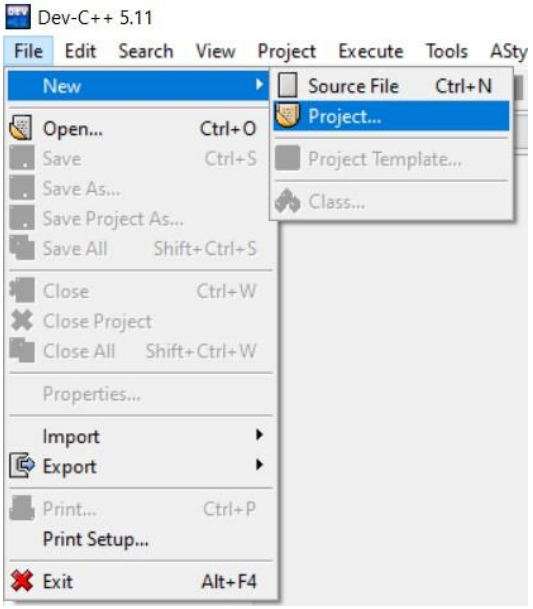
Setelah proses konfigurasi integrasi Dev C++ dan OpenGL selesai dilakukan, maka tahapan selanjutnya adalah memulai project dengan menggunakan keduanya

1. Buka Software Dev C++



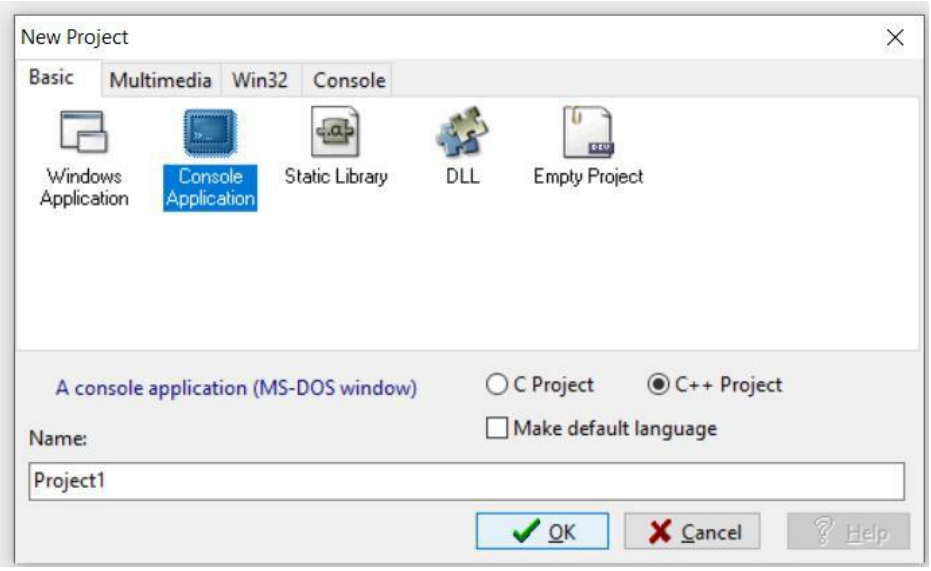

Gambar 46. Proses Open Software Dev C++

2. Pilih File, New, Project



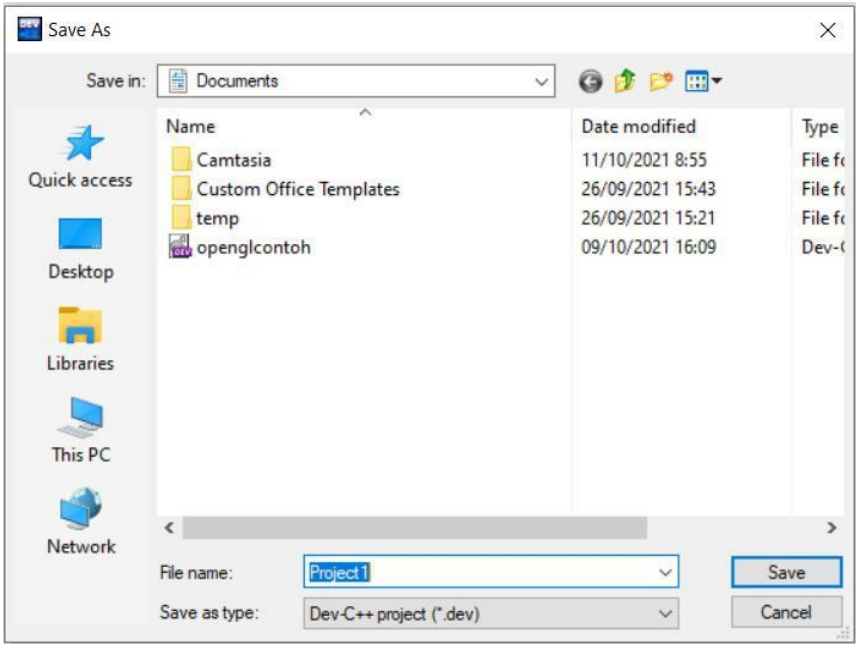

Gambar 47. Proses New Project Dev C++

3. Pilih console application, Rename file kemudian klik Oke



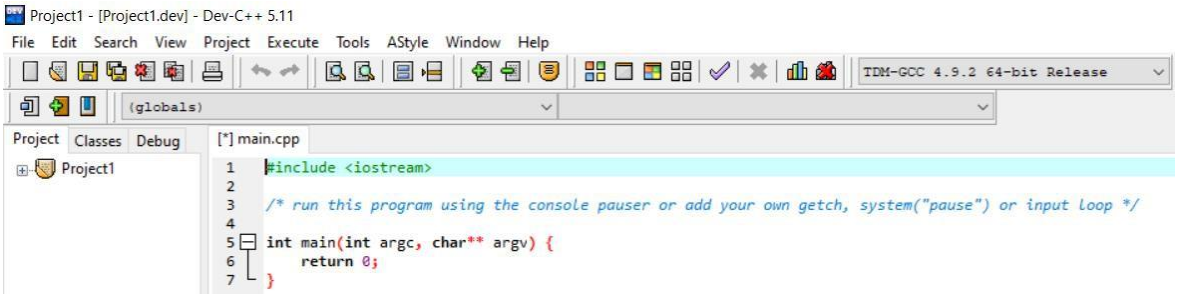

Gambar 48. Proses Save and Rename New Project Dev C++

4. Save project dilokasi file yang diinginkan, klik oke



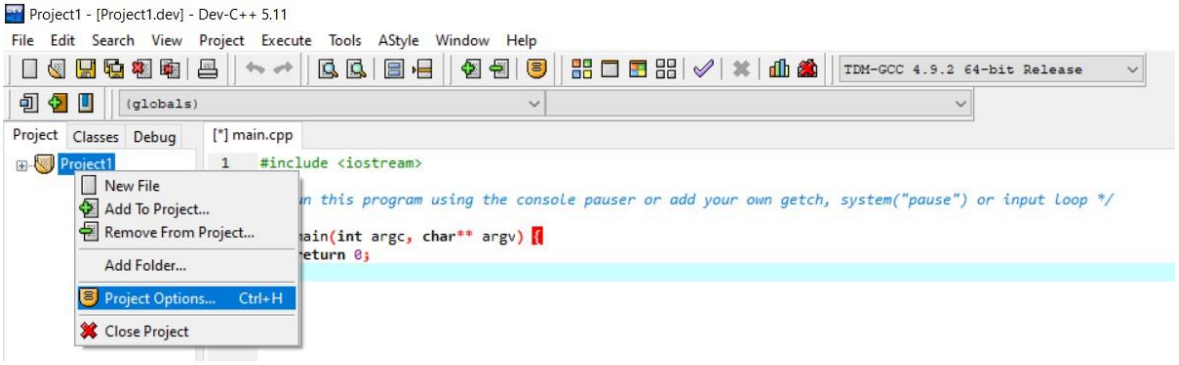

Gambar 49. Proses Penentuan Lokasi Penyimpanan Project Dev C++

5. Kemudian akan muncul tampilan



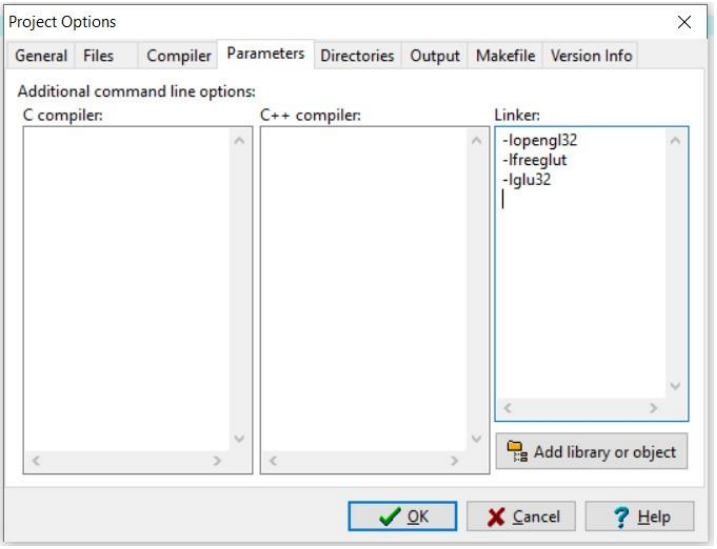

Gambar 50. Tampilan Hasil New Project Dev C++

6. Klik kanan pada project, dilanjutkan pilih project options



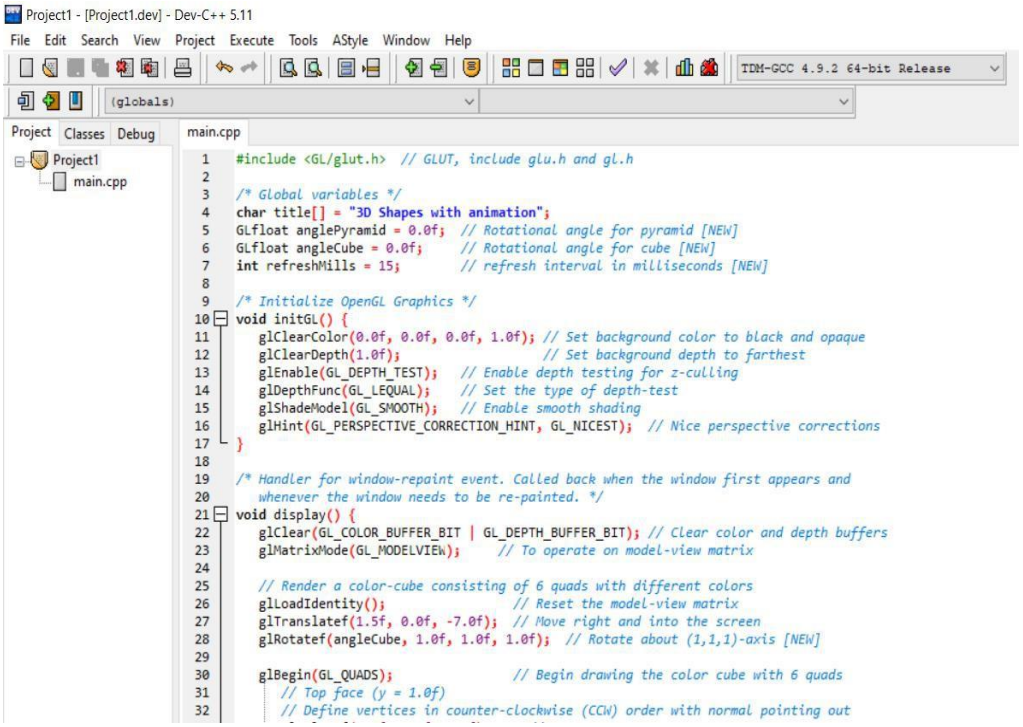

Gambar 51. Proses Setting Project Dev C++

7. Pada Project Options, pilih Parameter, dan pada bagian linker ketik perintah berikut : -lopengl32, -lfreeglut, -lglu32 (tanpa tanda baca koma) kemudian klik Oke



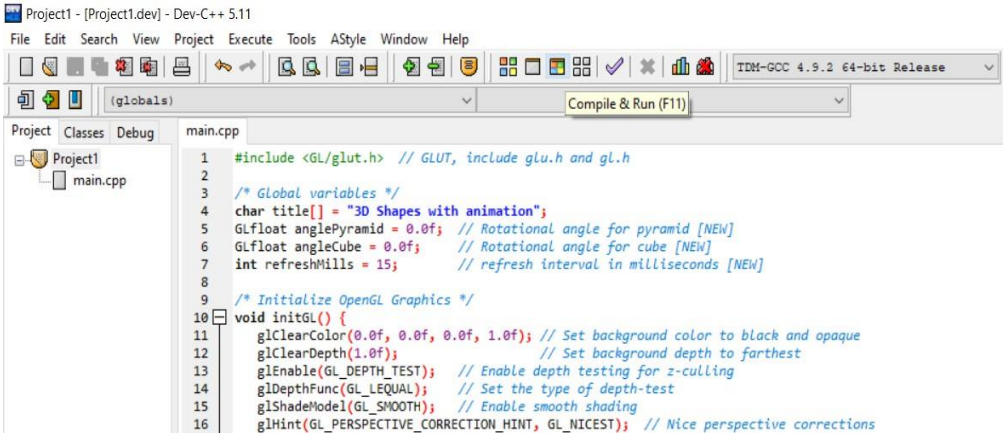

Gambar 52. Proses Setting Paramter Project Dev C++

8. Hapus coding pada poin 5 dan masukkan atau copy coding



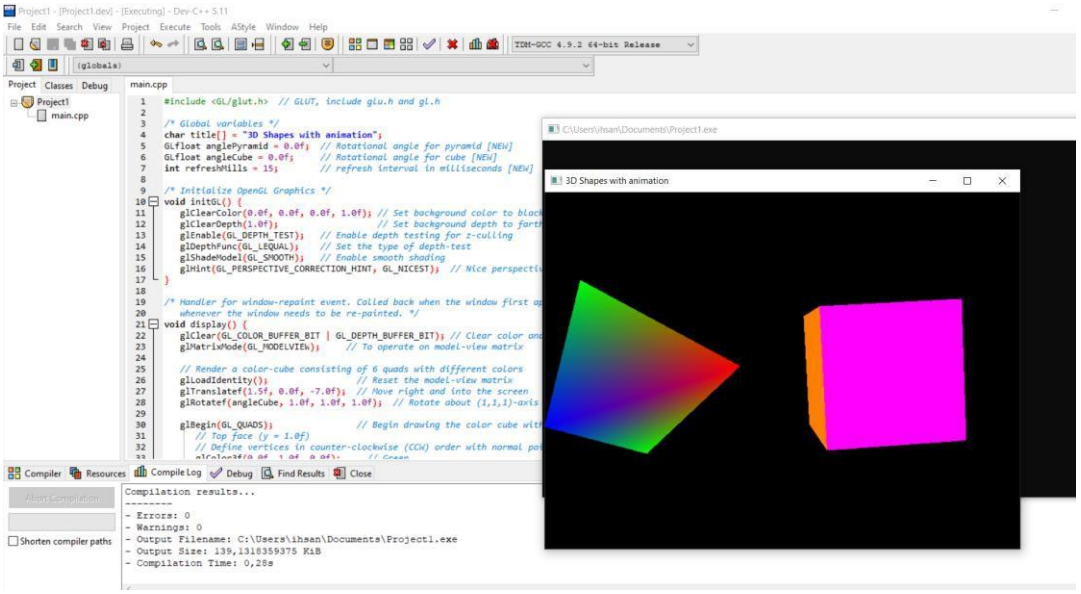

Gambar 53. Proses Testing Integrasi Dev C++ dan OpenGL

9. Setelah berhasil dicopy klik compile and run




*Gambar 54. Proses Compile and Run Dev C++ dan OpenGL*

10. Dan akan menghasilkan output seperti Gambar 55.




Gambar 55. Hasil Testing Integrasi Dev C++ dan OpenGL

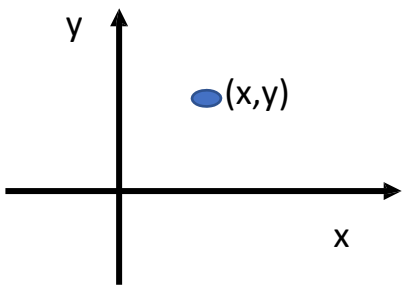
# BAB IV PRIMITIVE DRAWING

## IV.1. Definisi Primitive Drawing

*Primitive Drawing* merupakan suatu metode menggambar objek pada suatu layar / monitor dengan menggunakan langkah-langkah geometri yang bersifat sederhana. Geometri merupakan suatu cabang ilmu matematika yang mengandung tentang konsep skema bangun ruang yang berfokus pada pengukuran, bentuk posisi dan beberapa parameter bangun ruang lainnya. *Primitive drawing* membangun sebuah bangun datar atau bangun ruang menggunakan bagian-bagian objek pada objek gambarnya, misalnya titik, garis, garis dari suatu objek. Sehingga pada tahap tertentu, seluruh bagian tersebut akan digabungkan menjadi sebuah deskripsi bentuk objek bangun datar (dua dimensi) atau bangun ruang (tiga dimensi).

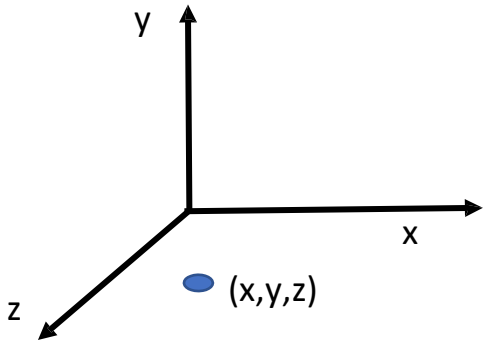
## IV.2 Konsep Dasar Titik Koordinat

Opengl dapat berperan sebagai wadah atau tempat untuk menampilkan hasil komputasi dari instruksi atau perintah yang dituliskan dalam software Bahasa pemrograman C++. Misalnya membuat suatu bangun datar, maka secara konsep bangun datar yang telah direncanakan pasti memiliki beberapa bagian yang harus dipersipakan, contohnya bangun datar memiliki titik-titik koordinat, fungsi dari titik koordinat tersebut sebagai acuan letak bangun datar yang akan dibangun, kemudian ada juga garis yang menghubungkan atar titik koordinat agar bentuk dari bangun datar tersebut dapat secara tegas dapat dibentuk. Hal ini juga didukung beberapa fitur tambahan lain, untuk mendeskripsikan bangun datar yang dibangun seperti, warna garis, warna bidang bangun datar dan lain sebagainya. Pada konsep dasar penempatan sumbu, sudah sangat popular dikenal. Terlebih lagi di masa sekolah hingga tingkat Pendidikan tinggi. Sumbu koordinat yang dimaksud adalah sumbu (x dan y). Ilustrasi bentuk titk koordinat dari sumbu (x,y) dapat dilihat pada Gambar 56.



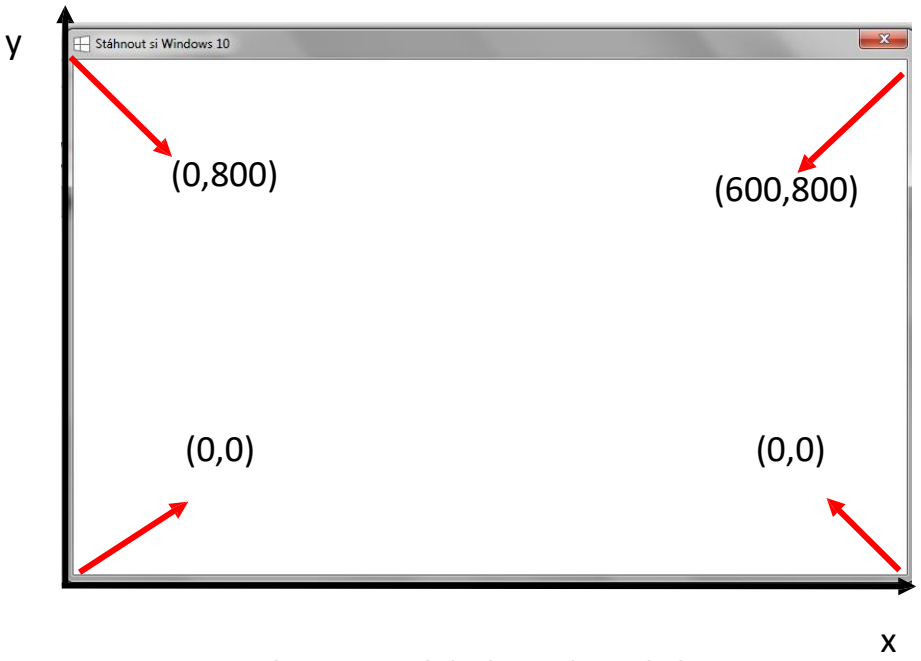

Gambar 56. Ilustrasi Sumbu x dan y

Sumbu x dan y dapat digunakan untuk menggambar objek titik, dan bangun datar yang sering juga dikenal dengan bangun datar atau dua dimensi (2D). Dengan adanya sumbu x dan y, *user* dapat menentukan tempat menggambar objek pada lokasi yang diinginkannya. Namun dalam visualisasi program *executable,* hasil dari proses *compile and Run* sumbu (x dan y) tidak dapat dilihat bentuknya. Namun *user* dapat menginisialisasi lokasi objek titik yang akan digambar dengan menggunakan instruksi yang ditulis pada *text editor*. Hal yang sama, juga dapat dilihat dengan menggunakan sumbu (x,y,z), dimana sumbu ini merepresentasikan tampilan yang memiliki orientasi bidang 3 dimensi. Deskripsi titik koordinat dengan menggunakan sumbu (x,y,z) dapat dilihat pada Gambar 57.




Gambar 57. Ilustrasi Sumbu x, y dan z

Dalam lembar kerja, *user* akan disajikan ruang atau dimensi lokasi menggambar yang biasa disebut juga dengan *drawing window.* Hal ini dapat membantu *user* untuk mengetahui seberapa besar ruang *drawing window* serta dapat memposisikan lokasi gambar secara tepat sesuai dengan ukuran *drawing window* yang telah di-*setting* sebelumnya. Jika dimensi *drawing windows* di-*setting* sebesar 600,800, maka *user* akan dapat memahami bawa nilai sumbu x maksimal adalah sebesa 600, dan nilai sumbu y maksimal adalah sebesar 800. Dengan besar dimensi tersebut, maka *user* dapat menggunakan titik koordinat (x,y) untuk menggambar objek titik dimasa saja. Dimana uraian titik koordinat terkecil dari sumbu (x,y) adalah (0,0) hingga yang terbesar adalah (600,800). Dengan titik koordinat sebanyak itu, maka *user* bebas menggambarkan titik sesuai dengan cakupan dimensi *drawing windows* yang telah diatur. Deskripsi terkait ukuran *drawing windows* dapat dilihat pada Gambar 58.

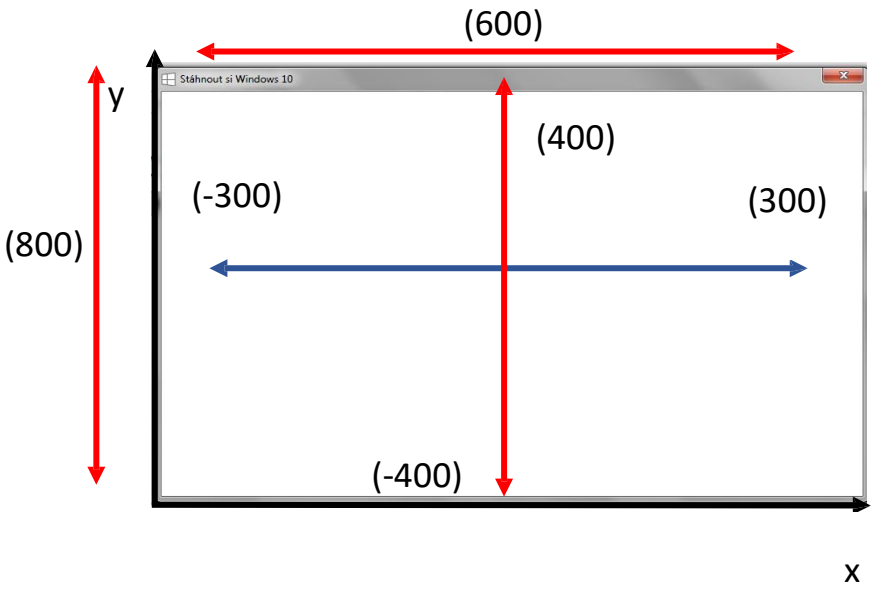



Gambar 58. Deskripsi Drawing Windows

Untuk melakukan *setting* ukuran *drawing windows*, *user* dapat menggunakan perintah atau instruksi sebagai berikut :

1. glutInitWindowSize(600,800);
2. glutInitWindowPosition(100,100);

Kedua perintah ini cukup sering digunakan untuk mengkonfigurasi serta menentukan besarnya dimensi *drawing windows.* Selain itu, *user* juga dapat menggunakan perintah glOrtho(), instruksi ini memberikan *user* opsi untuk meletakkan objek yang akan digambar. Hasil penerapan instruksi glOrtho() dapat dilihat pada Gambar 59.




Gambar 59. Implementasi Instruksi glOrtho()

## IV.3. Latihan

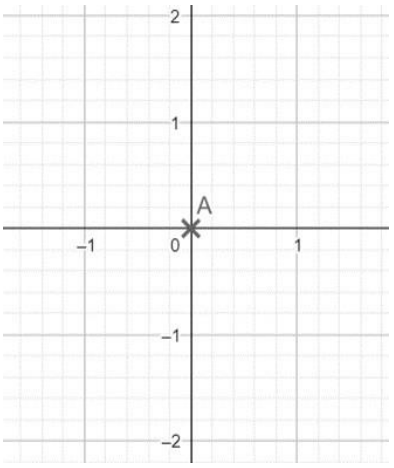
### 1. Latihan I Membuat Objek Sebuah Objek Titik

Seorang *user* ingin menggambar sebuah objek titik pada sumbu koordinat x dan y, dengan menggunakan Dev C++ dan OpenGL dengan kriteria :

a. Titik koordinat (0,0)
b. Warna Merah
c. Ukuran *drawing windows* (600, 800)
d. Daerah matriks ( disesuaikan)
e. Besar objek titik 55,5

Maka penyelesaiannya dapat dilakukan dengan langkah berikut :

1. Menetapkan titik koordinat lokasi penggambaran titik pada sumbu (x,y) yaitu (0,0), seperti yang ditunjukkan pada Gambar 60.




Gambar 60. Titik Koordinat Lokasi Penggambaran Objek Titik

2. Preproses *drawing* objek telah diperoleh, kemudian melakukan beberapa konfigurasi yaitu dengan menggunakan instruksi :
   a. `#include <GL/glut.h>`
      Instruksi ini digunakan untuk memanggil *library* OpenGL, dengan begitu hasil dari kompilasi instruksi yang berupa kode objek dapat ditampilkan oleh OpenGL melalui program yang bersifat *executable* (.exe)

   b. `main(int argc, char** argv)`
      `argc dan argv` merupakan instruksi yang akan di-*forward* ke `main()` dalam bahasa pemrograman C atau C++. `argc` merupakan kepanjangan dari

*argument for count* yang didefinisikan sebagai salah satu parameter dalam tipe int (integer) serta memiliki fungsi untuk menunjukkan banyaknya parameter yang digunakan pada proses eksekusi suatu program. Sementara `argv` merupakan singkatan dari *argument for vector* yang berfungsi untuk menyimpan seluruh parameter yang digunakan selama proses *compile and run* berlangsung.

c. `void titik ();`

`void` merupakan fungsi dengan tipe data kosong, berbeda dengan integer dan float yang memiliki nilai. Oleh karena `void` tidak memiliki nilai, maka operasi yang dieksekusi tidak akan menyertakan perintah `Return.` Penulisan parameter pada fungsi `void` bersifat opsional.

d. `glutInit(&argc,argv);` memiliki fungsi dalam melakukan proses *argument command line* yang disertakan. Fungsi ini merupakan satu dari sekian fungsi yang masuk ke dalam kategori, yang harus dipanggil pertama kali, atau masuk dalam fungsi yang harus dibuat diawal program.

e. `glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);` merupakan fungsi yang dapat digunakan untuk memberikan efek RGB pada grafik, titik atau pun garis yang ingin dibuat oleh *user.* Serta memungkinkan *user* untuk memilih menggunakan buffer windows secara *single* atau *dooble*. Perbedaan utamanya adalah jika buffer windows secara *single* kode objek yang dihasilkan proses *linking* akan langsung ditampilkan pada program yang bersifat *executable* (.exe), sehingga *user* dapat melihat secara langsung objek dalam bentuk gambar. Sementara *dooble, user* dapat membayangakan menggunakan dua *buffer windows*, dimana hanya salah satunya saja yang dapat terlihat sementara yang lainnya tidak terlihat.

f. `glutInitWindowSize(int x, int y);` dideskripsikan sebagai perintah untuk melakukan *setting* ukuran *drawing windows*, sebesar (x,y).

g. `glutInitWindowPosition();` digunakan untuk menentukan posisi sebuah *windows*

h. `glutCreateWindow("……..");` digunakan untuk membuat suatu *windows*

i. `glClearColor();` berfungsi untuk membersihkan atau me-*reset* nilai pixel.

j. `glMatrixModel();` berfungsi untuk menginisialisasi matriks

k. `glOrtho();` berfungsi sebagai penentu letak secara spesifik objek akan digambar, misalnya glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0); perintah ini akan membuat objek yang digambar berada pada interval sumbu x dengan titik koordinat [-1,1], sumbu y dengan titik koordinat [-1,1] dan sumbu z dengan titik koordinat [-1,1].

l. `glutDisplayFunc();`merupakan perintah yang digunakan untuk menampilkan hasil komputasi suatu fungsi, namun penulisan nama fungsi haru sesuai dengan fungsi yang ditulis sebelumnya.

m. `glutMainLoop();`merupakan perintah untuk mengulang atau melakukan proses *looping*

3. Selanjutnya melakukan penggambaran objek titik dengan menggunakan perintah
   a. `glClear(GL_COLOR_BUFFER_BIT);` digunakan untuk menghapus warna pada layar

   b. `glPointSize();` digunakan untuk menentukan besar objek yang akan digambar, semakin besar nilainya akan semakin besar objeknya.

   c. `glBegin();` digunakan untuk menuliskan perintah menggambar objek titik

   d. `glColor3f();` digunakan untuk memberikan warna yang baru pada objek, setelah sebelumnya warna yang ada telah dihapus.

   e. `glVertex2f();` dideskripsikan sebagai instruksi untuk menuliskan titik koordinat objek akan digambar pada *drawing windows.*

4. Mengakhiri program dengan *statement* tambahan,
   a. `glEnd();`mengakhiri perintah `glBegin()`

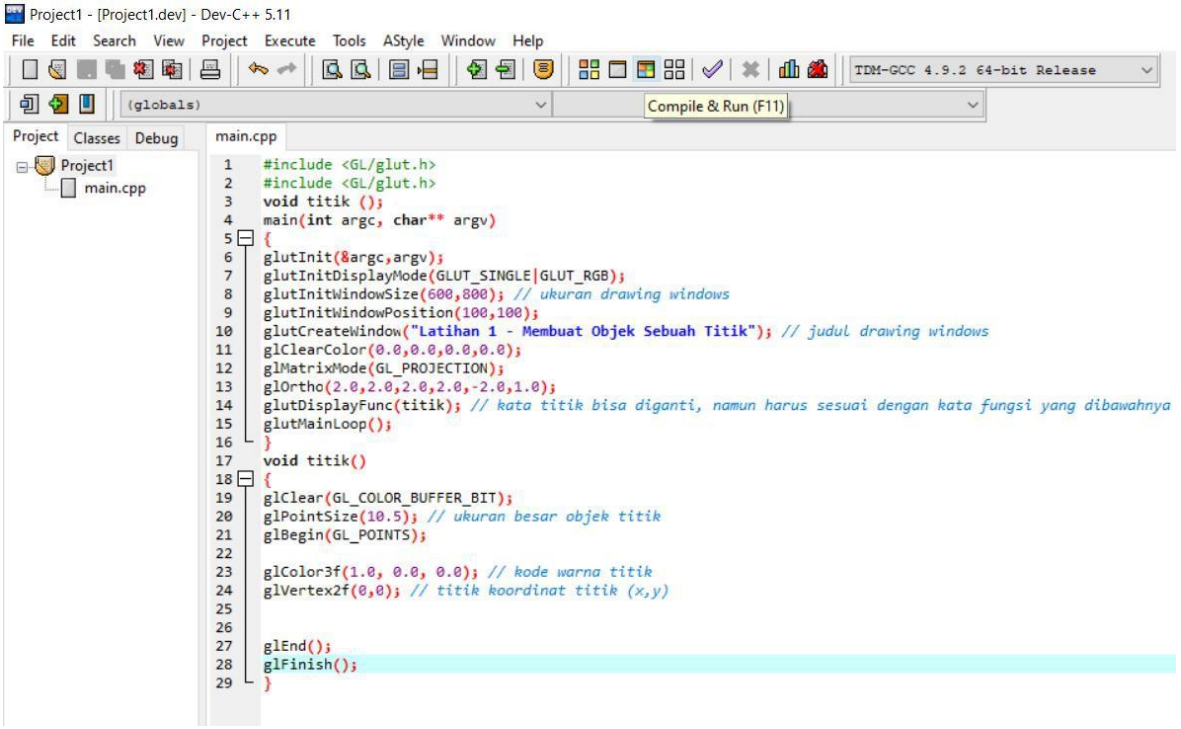
   b. `glFinish();` adalah sebagai instruksi untuk

Sehingga dari uraian diatas, maka seluruh *list* program dapat dituliskan dan digabungkan sebagai berikut :

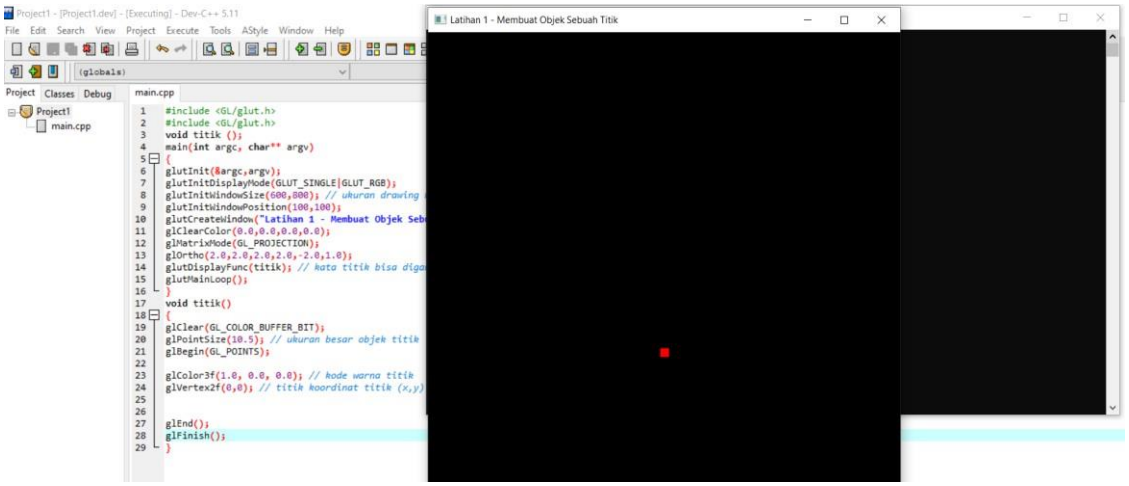
*List* Program Menggambar Objek 1

```
#include <GL/glut.h>
#include <GL/glut.h>
void titik ();
main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(600,800); // ukuran drawing windows
glutInitWindowPosition(100,100);

glutCreateWindow("Latihan 1 - Membuat Objek Sebuah Titik"); // judul
drawing windows

glClearColor(0.0,0.0,0.0,0.0);
glMatrixMode(GL_PROJECTION);
glOrtho(2.0,2.0,2.0,2.0,-2.0,1.0);

glutDisplayFunc(titik); // kata titik bisa diganti, namun harus
sesuai
                        dengan kata fungsi yang dibawahnya
glutMainLoop();
}
void titik()
{

glClear(GL_COLOR_BUFFER_BIT);
glPointSize(10.5); // ukuran besar objek titik
glBegin(GL_POINTS);

glColor3f(1.0, 0.0, 0.0); // kode warna titik
glVertex2f(0,0); // titik koordinat titik (x,y)


glEnd();
glFinish();
}
```

Tampilan dan hasil proses *compile and run* dengan program diatas dapat dilihat Gambar. dan Gambar 61.

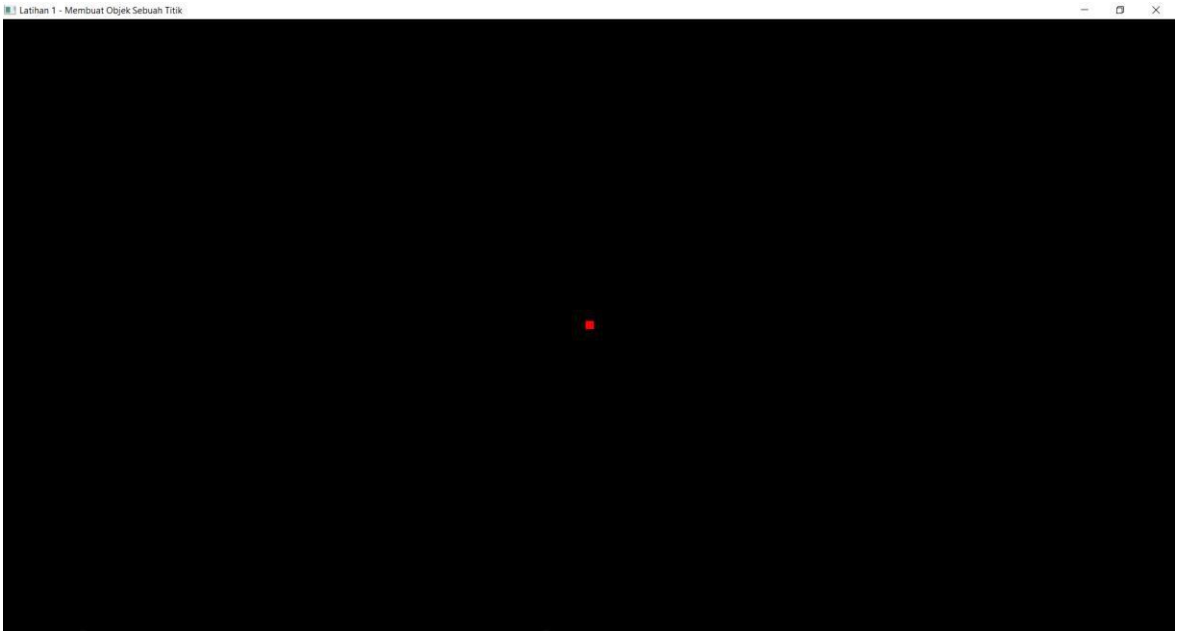



Gambar 61. Penulisan Program Menggambar sebuah Objek Titik




Gambar 62. Penulisan Program Menggambar sebuah Objek Titik

Sementara untuk memastikan bahwa objek titik yang digambar, berada tepat pada titik koordinat (0,0), *user* dapat memperbesar ukuran *drawing windows* menjadi ukuran *full screen* seperti yang ditunjukkan pada Gambar 63.




Gambar 63. Tampilan Full Screen Objek Titik

**2. Latihan II Membuat Objek Beberapa Objek Titik**

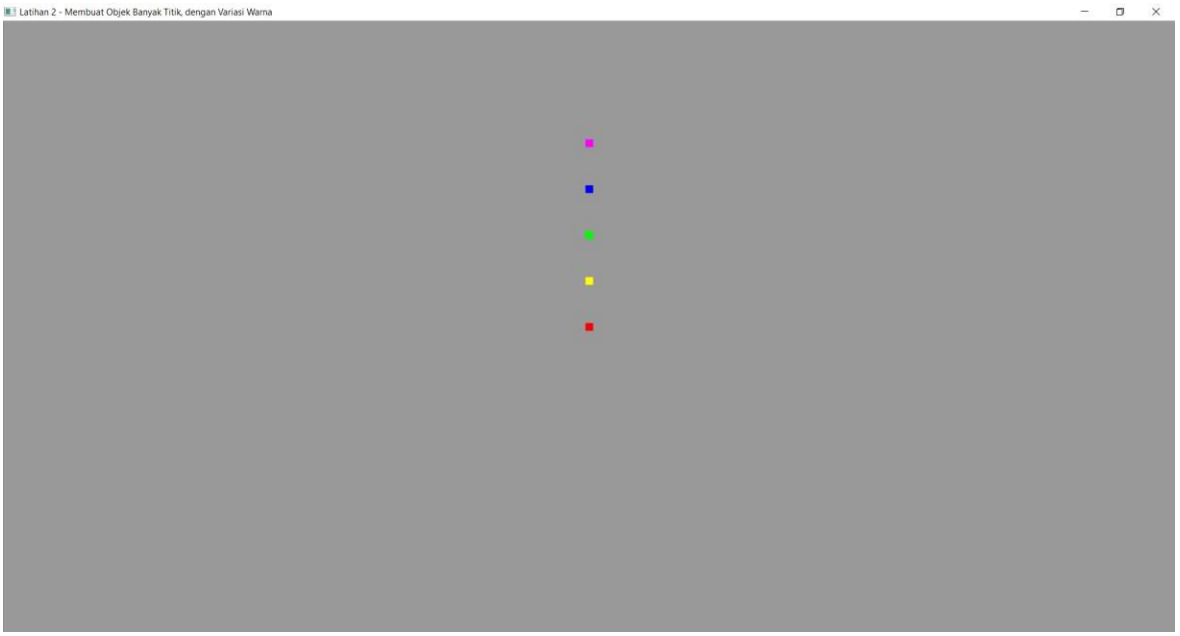
Seorang *user* ingin menggambar beberapa objek titik pada sumbu koordinat x dan y, dengan menggunakan Dev C++ dan OpenGL dengan kriteria :

a. Titik koordinat titik pertama (0,0) ; titik kedua (0,0.15) ; titik ketiga (0,0.3) ; titik keempat (0,0.45) ; titik kelima (0,0.6)
b. Warna titik pertama merah, titik kedua kuning, titik ketiga hijau, titik keempat biru, titik kelima ungu
c. Ukuran *drawing windows* (600, 800)
d. Daerah matriks ( disesuaikan)
e. Besar objek titik 10,5
f. Background hitam diganti dengan warna abu-abu

Penyelesaiannya dapat dengan membuat duplikasi pada program “Latihan-I” dengan menambahkan instruksi sesuai dengan keinginan *user*, yaitu dengan menambahkan empat titik lainnya. Modifikasi list program pertama dapat dilakukan dengan cara menduplikasi perintah pembentukan titik pertama untuk empat titik lainnya, serta melakukan penambahan instruksi untuk mengubah warna, dan titik koordinat masing-masing objek titik. Untung mengubah warna *background drawing windows* dapat dilakukan dengan mengubah kode warna pada instruksi “`glClearColor()`”.List program secara lengkap dapat dilihat dibawah ini.

```
#include <GL/glut.h>
#include <GL/glut.h>
void titik ();
main(int argc, char** argv)
{ glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(600,800); // ukuran drawing windows
glutInitWindowPosition(100,100);
glutCreateWindow("Latihan 2 - Membuat Objek Banyak Titik, dengan Variasi
Warna"); // judul drawing windows
glClearColor(0.6f, 0.6f, 0.6f, 1.0f); //Ganti warna drawing windows dari
hitam menjadi abu-abu
glMatrixMode(GL_PROJECTION);
glOrtho(2.0,2.0,2.0,2.0,-2.0,1.0);
glutDisplayFunc(titik); // kata titik bisa diganti, namun harus sesuai
dengan kata fungsi yang dibawahnya
glutMainLoop();
}
void titik()
{
glClear(GL_COLOR_BUFFER_BIT);
glPointSize(10.0); // ukuran besar objek titik
glBegin(GL_POINTS);

glColor3f(1.0, 0.0, 0.0); // kode warna titik pertama
glVertex2f(0,0); // titik koordinat titik (x,y)

glColor3f(1.0, 1.0, 0.0); // kode warna titik kedua
glVertex2f(0,0.15); // titik koordinat titik (x,y)

glColor3f(0.0, 1.0, 0.0); // kode warna titik ketiga
glVertex2f(0,0.30); // titik koordinat titik ketiga (x,y)

glColor3f(0.0, 0.0, 1.0); // kode warna titik keempat
glVertex2f(0,0.45); // titik koordinat titik keempat (x,y)

glColor3f(1.0, 0.0, 1.0); // kode warna titik kelima
glVertex2f(0,0.60); // titik koordinat titik kelima (x,y)

glEnd();
glFlush();
}
```

Hasil *compile and run* dari program diatas dapat dilihat pada Gambar 64. berikut




Gambar 64. Hasil Compile and Run Program Objek Banyak Titik dengan Variasi Warna

### 3. Latihan III Menggambar Garis Antar Titik Koordinat

Seorang *user* ingin menggambar dua buah objek titik yang terhubung satu dengan yang lainnya melalui sebuah garis pada sumbu koordinat x dan y, dengan menggunakan Dev C++ dan OpenGL dengan kriteria :

a. Titik koordinat titik pertama (0,0) ; titik kedua (300,400) dengan warna garis hitam
b. Warna *background* putih

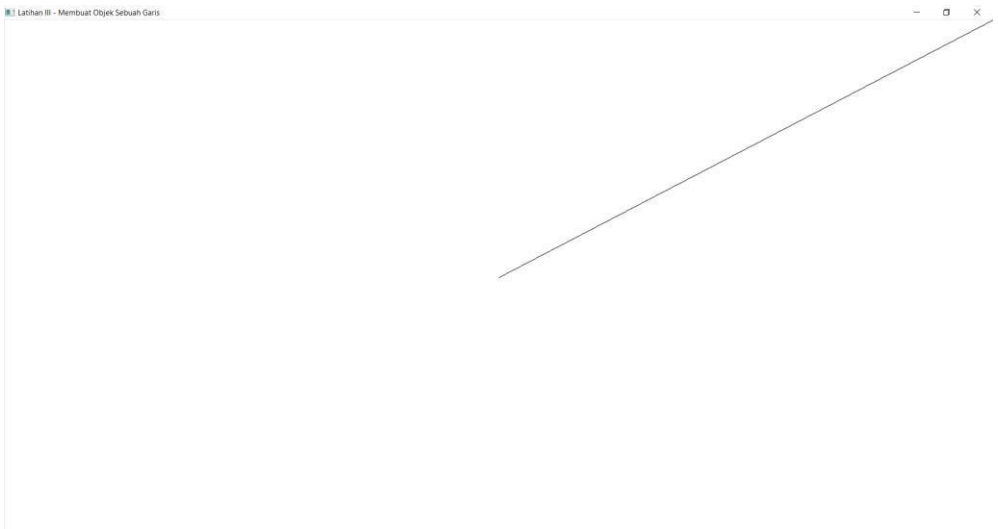
Penyelesaian dengan list program dibawah ini :

List program membuat garis yang menghubungkan dua titik

```
#include <GL/glut.h>
void Garis (); //kata garis dapat diganti sesuai keinginan
main(int argc, char** argv)
{ glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(600,800); // ukuran drawing windows
glutInitWindowPosition(100,100);
glutCreateWindow("Latihan III - Membuat Objek Sebuah Garis"); // judul drawing windows
```

```
glClearColor(1.0,1.0,1.0,0.0); //kode warna background
glMatrixMode(GL_PROJECTION);
gluOrtho2D(-300.0,300.0,-400.0,400.0); //nilainya harus masuk dalam rentang nilai drawing windows
glutIdleFunc(Garis);// kata garis mengikuti perintah void yang paling atas
glutDisplayFunc(Garis); // kata garis mengikuti perintah void yang paling atas
glutMainLoop();
}
void Garis()// kata garis mengikuti perintah void yang paling atas
{ glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_LINES); //fungsi membuat garis

glColor3f(0.0, 0.0, 0.0); // kode warna titik pertama
glVertex2d(0,0); // titik koordinat titik (x,y) pertama

glColor3f(0.0, 0.0, 0.0); // kode warna titik kedua
glVertex2d(300,400); // titik koordinat titik (x,y) kedua
glEnd();
glFinish();
}
```

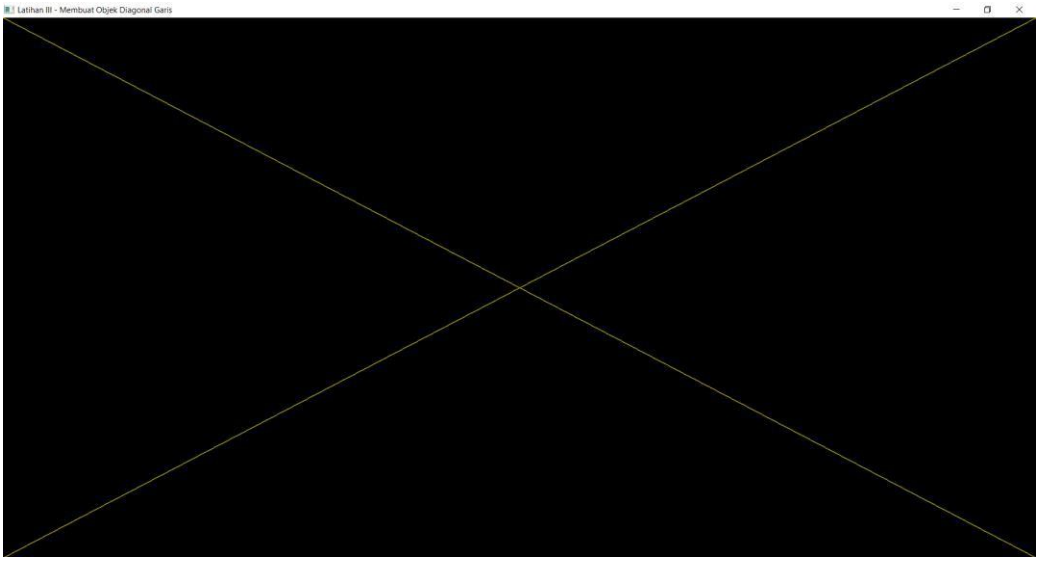
Hasil proses *compile and run* pada list program diatas, ditunjukkan pada Gambar 65.




Gambar 65. Hasil Compile and Run Program Objek Garis yang Menghubungkan Titik

## 4. Latihan IV Menggambar Garis Diagonal

Berdasarkan *list* program Latihan -III, silahkan membuat latihan untuk menggambar objek garis dengan menggunakan empat sisi, seperti yang ditujukkan Gambar 68.




Gambar 66. Hasil Proses Compile and Run Latihan IV

Berdasarkan Gambar. dapat dilihat garis yang terbentuk berwarna kuning, dan background *drawing windows* berwarna hitam.

## 5. Latihan V Menggambar Objek Bangun Datar

Dalam OpenGL, untuk mengambarkan sebuah objek bangun datar segitiga, dapat menggunakan perintah `glBegin(GL_POLYGON);.` Hal ini disebabkan bahwa bangun datar merupakan contoh penerapan polygon, Poligon didefinisikan sebagai sebuah objek yang terbentuk dari titik yang terhubung satu sama lain dengan sebuah garis atau beberapa garis yang dapat membentuk sebuah sudut, yang dapat dilihat sebagai sebuah objek bangun datar. Untuk melakukan *setting* pada *fill* pada suatu bangun datar, maka dapat dituliskan perintah `glColor3f(nila1,nilai2,nilai3);`. Contoh program yang dapat *user* gunakan adalah sebagai berikut :

```
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_POLYGON); // instruksi kode Poligon

glColor3f(1.0, 0.0, 0.0);
glVertex2f(0.,100.0);
glColor3f(0.0, 1.0, 0.0);
```

```
glVertex2f(-100.,0.);
glColor3f(0.0, 0.0, 1.0);
glVertex2f(100.,0.);
```

Hasil proses *compile and run* pada program diatas dapat dilihat pada Gambar 67.

Gambar 67. Hasil Proses Compile and Run Latihan V

## 6. Latihan VI Menggambar Objek Bangun Datar Persegi Panjang

Berdasarkan program pembuatan segitiga, silahkan *user* melakukan pembuatan bangun datar persegi panjang dengan menggunakan kriteria objek pada Gambar 68. sebagai berikut :

a. Warna *background* hitam, dan ubahlah warna objek pada Gambar. seperti dengan merah putih, seperti Bendera Negara Kesatuan Republik Indonesia.
b. Objek yang digambar harus berada pada daerah koordinat (-100, 100)
c. Gunakan `glBegin(GL_POLYGON);` untuk menghubungkan setiap titik koordinat sesuai dengan Gambar.
d. *Rename* nama *drawing windows* dengan nama : “Latihan-VI Menggambar Objek Bendera Merah Putih”

Gambar 68. Hasil Proses Compile and Run Latihan VI

## 7. Latihan VII Membuat Lingkaran dengan Menggunakan Persamaan Matematika

Lingkaran merupakan salah satu jenis bangun datar yang memiliki sudut sebesar 360 derajat. Oleh karena itu, akan sangat sulit jika melakukan penggambaran lingkaran secara manual dengan melakukan *plotting* terhadap seluruh titik secara manual. Salah satu solusi yang dapat dilakukan untuk menimimalisir pembentukan Lingkaran secara manual adalah dengan menggunakan persamaan matematik. Tentunya hal tersebut membutuhkan logika pemrograman tepat agar sebuah lingkaran utuh dan simetris dapat terbentuk. Deskrip program yang dapat digunakan dalam membetuk sebuah lingkaran adalah sebagai berikut :

```
#include <GL/glut.h>
#include<math.h>
const double PI = 3.142857143;
int i,jari_jari,banyak_titik,x_pusat,y_pusat;

void Lingkaran ();
main(int argc, char** argv)
{ glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(600,600);
glutInitWindowPosition(100,100);
glutCreateWindow("Latihan V - Membuat Lingkaran");
glClearColor(1.0,1.0,1.0,0.0);
glMatrixMode(GL_PROJECTION);
glOrtho(1.0,1.0,1.0,1.0,-1.0,1.0);
glutDisplayFunc(Lingkaran);
glutMainLoop();
return 0;
}
void Lingkaran ()
{ glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1,0,0);//untuk warna lingkaran
glBegin(GL_POLYGON);
      jari_jari = 50;
```

```
    banyak_titik = 60;
    x_pusat = 0;
    y_pusat = 0;

    for (i=0;i<=360;i++)
    {
        float sudut=i*(2*PI/banyak_titik);
        float x=x_pusat+jari_jari*cos (sudut);
        float y=y_pusat+jari_jari*sin (sudut);
        glVertex2f(x/100,y/100);
    }
glEnd();
glFlush();
}
```

Keterangan Program :

1. #include<math.h> oleh karena saat ini *user* melakukan desain dengan menggunakan persamaan matematika, maka instruksi ini perlu dituliskan pada *list* program. Tujuannya adalah memasukkan operasi matematika yang digunakan, dapat dilihat pada *list* program diatas terdapat operasi matematika sin, cos, penjumlahan (+), pembagian("/") serta perkalian (*). Fungsi operasi tersebut tidak perlu dideklarikan kembali, sebab sudah dimasukkan melalui instruksi #include<math.h>.

2. `const double PI = 3.142857143`; dan
   `int i,jari_jari,banyak_titik,x_pusat,y_pusat;`

   Kedua instruksi ini merupakan instruksi untuk mendeklarasikan nilai variable, urauan variabel nilai dapat diuraikan sebagai berikut :

   a. `const double PI = 3.142857143`; *Pi* maerupakan salah satu komponen variabel yang penting dsalam menghitung luas lingkaran, maupun keliling lingkaran. Seperti yang diketahui nilai *Pi* adalah sebesar 3,14.

   b. `int i,jari_jari,banyak_titik,x_pusat,y_pusat;`
      Pada deklarasi instruksi ini, dapat dilihat bahwa ada beberapa variabel diantaranya adalah *i, jari_jari, banyak_titik, x_pusat, y_pusat.* Seluruh

variabel tersebut nantinya kan diproses, digunakan dan dihitung dalam badan fungsi

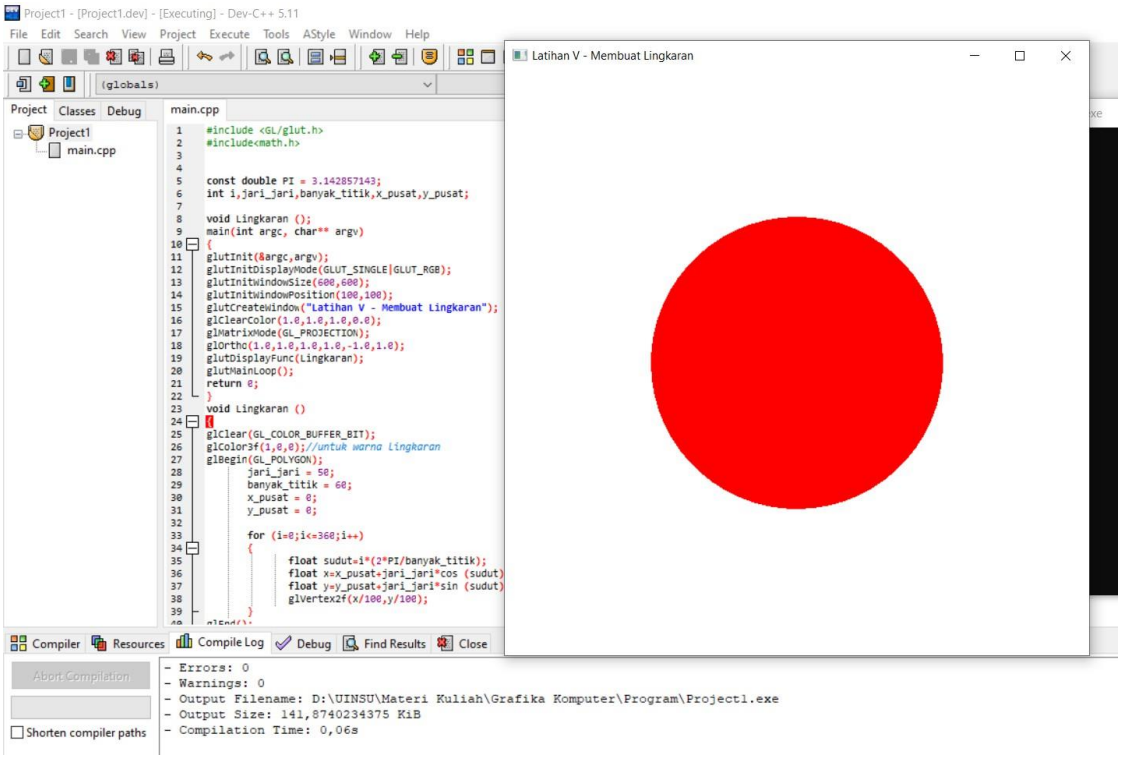
3. jari_jari = 50; , banyak_titik = 60;, x_pusat = 0;, y_pusat = 0;
   jari-jari merupakan bagian dari lingkaran yang menjadi poin penting dalam menghitung lingkaran. Diameter lingkaran adalah dua kali besar jari-jari, sementara banyak_titik merepresentasikan jumlah titik yang terbentuk sebagai unsur penyusun lingkaran yang saling terhubung satu dengan yang lainnya dengan garis.

4. for (i=0;i<=360;i++)
   variabel “I” merupakan ilustrasi sudut dari lingkaran yang dimulai dari 0 (nol) hingga 360 (tiga ratus enam puluh), variabel “I” akan terus bertambah dari nilai awal 0 hingga 360, seiring dengan proses *for* bekerja.

5. float sudut=i*(2*PI/banyak_titik);
   Instruksi ini menghitung pembentukan titik-titik pada lingkaran dengan menggunakan nilai pertambahan variabel “i” dari 0 sampai dengan 360.

6. float x=x_pusat+jari_jari*cos (sudut);  float y=y_pusat+jari_jari*sin (sudut);
   Instruksi ini menghitung perputaran titik pembentukan sudut sesuai dengan nilai sudur hasil operasi sin dan cos yang dipengaruhi oleh variabel “i”.

Berdasarkan hal tersebut, maka *list* program akan di- *compile and run* dan menghasilkan output sebagaimana yang dapat dilihat pada Gambar 69.




Gambar 69. Hasil Proses Compile and Run Latihan VII

# DAFTAR PUSTAKA


Astle Dave, Hawkins Kevin. 2004. *Beginning OpenGL Game Programming*. Premier Press. USA

Buss, Samuel R. 2003. *3D Computer GraphicsA Mathematical Introduction with OpenGL*, Addison-Wesley Publishing Company. USA

Hejlsberg Anders, Wiltamuth Scott, Golde Peter. 2004 *The C Programming Language*. Addison-Wesley). USA

Mason Woo, Jackie Neider, Tom Davis. 1997. *OpenGL Architecture Review Board - OpenGL Programming Guide*. Addison-Wesley Pub. USA.

Shreiner Dave, Woo Mason, Neider Jackie, Davis Tom. 2005. OpenGL Architecture Review Board and OpenGL Programming Guide. Addison-Wesley Professional. USA

Terry Patrick D. 1997. *Compliers and Compiler Generators An Introduction With C++.* International Thomson Computer Press. USA

Stroustrup Bjarne. 2000. *The C++ Programming Language.*Addison-Wesley Professional. USA.

Prinz Ulla Kirch, Prinz Peter. 2002. *A Complete Guide to Programming in C++*. Jones and Barlett Publishers. Canada

Wright Richard S, Lipchak Benjamin, Haemel Nicholas. 2007. *OpenGL (R) SuperBible_ Comprehensive Tutorial and Reference*. Addison-Wesley Professional. USA.