1. **Object Primitif**

Object primitive adalah sebuah bentuk dasar untuk membentuk sebuah object grafis, dikatakan object primitive karena merupakan bentuk paling dasar sebagai object dan juga merupakan sebuah dasar untuk semua object grafis. Dengan adanya object primitive, nantinya dapat digunakan untuk membuat sesuatu object / grafis yang lebih kompleks. Object primitive yang paling dasar adalah sebuah titik, object titik ini nantinya akan dapat digunakan untuk memebentuk sebuah object baru seperti garis dan object lain yang lebih kompleks seperti bangun 2D.

1. **OpenGL Primitif**

Pada openGL primitive, terdapat sebuah fungsi untuk membuat sebuah object baru. Fungsi tersebut diantaranya adalah: GL Points, GL Lines, GL Line Strip, dan GL Line Loop. Untuk menggunakan fungsi tersebut fungsi ini harus dideklarasikan ke dalam fungsi GL begin terlebih dahulu, kemudian menentukan titik koordinat yang akan digambar dengan glVertex dan mengakhirinya dengan glEnd seperti pada contoh berikut.

Code:

glBegin(fungsi object yang akan digambar);

glVertex3f (x, y, z);

glEnd();

Fungsi glBegin pada code diatas digunakan untuk memberikan perintah pada openGL bahwa akan digambar suatu object menggunakan fungsi yang diinginkan, dimana fungsi tersebut dimasukkan ke dalam fungsi glBegin dan digunakan untuk parameter pada glBegin. Seperti pada contoh berikut, menggunakan fungsi GL Points untuk dijadikan parameter pada fungsi glBegin.

Code:

glBegin(GL\_POINTS);

glVertex3f (x, y, z);

glEnd();

Kemudian glVertex3f digunakan untuk menentukan titik koordinat x, y, z. titik koordinat ini nantinya akan menentukan letak posisi object tersebut akan digambar. Ada beberapa fungsi yang dapat digunakan untuk menetukan letak titik koordinat, fungsi tersebut memiliki kegunaan yang sama, akan tetapi memiliki perbedaan pada parameter yang akan digunakan. Berikut merupakan beberapa fungsi untuk menentukan titik koordinat pada sebuah object:

* glVertex3i(x, y, z):

Merupakan fungsi untuk menggambar object pada titik koordinat x, y, dan z, dengan nilai satuan berupa integer. Contoh: glVertex3i(2, 2, 2);

* glVertex3f(x, y, z):

Merupakan fungsi untuk menggambar object pada titik koordinat x, y, dan z, dengan nilai satuan berupa float. Contoh: glVertex3f(2.5, 3.5, 0.0);

* glVertex3d(x, y, z):

Merupakan fungsi untuk menggambar object pada titik koordinat x, y, dan z, dengan nilai satuan berupa double. Contoh: glVertex3d(10.0, 10.0, 10.0);

Angka 3 pada glVertex3i(x, y, z) digunakan untuk menggambar sebuah object tiga dimensi, apabila ingin menggambar sebuah object 2 dimensi, dapat mengganti angka 3 dengan angka 2. Kemudian fungsi glEnd digunakan untuk memberi perintah pada openGL bahwa itu merupakan akhir dari proses penggamabrn object tersebut, sehingga object yang sudah diatur menggunakan glVertex dapat dijalankan dan digambar sehingga akan membentuk sebuah object.

Dari contoh code dan penjelasan yang sudah dijelaskan di atas, selanjutnya adalah mencoba membuat masing – masing object menggunakan fungsi yang ada openGL primitive. Fungsi yang akan digunakan selanjutnya adalah fungsi GL Points, GL Lines, GL Line Strip, dan GL Line Loop.

1. GL Points

GL Points merupakan salah satu object primitive yang paling dasar, fungsi ini akan menghasilkan sebuah object titik. Titik merupkan sebuah object yang paling mendasar untuk menggambar, dan untuk memperbesar ukuran dari object titik dapat menggunakan fungsi glPointSize seperti contoh berikut:

Code:

glPointSize(5.0);

glBegin(GL\_POINTS);

glVertex3f (0.0, 0.0, 0.0);

glVertex3f (0.0, 8.0, 0.0);

glVertex3f (8.0, 0.0, 0.0);

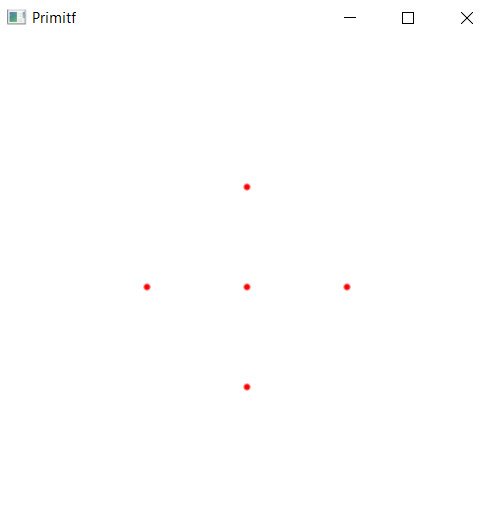
glVertex3f (0.0, -8.0, 0.0);

glVertex3f (-8.0, 0.0, 0.0);

glEnd();

Pada fungsi glPointSize terdapat sebuah argument bernilai 5.0 dengan type data float, fungsi ini memiliki artian yaitu untuk memperbesar ukuran dari GL\_POINTS dengan nilai ukuran asli sebesar 1.0 menjadi 5.0. Kemudian selanjutnya akan digambar 5 titik berdasarkan glVertex yang sudah dideklarasikan, titik koordinat pada glVertex harus berbeda agar titik yang digambar tidak menumpuk satu sama lain.

Output:



1. GL Lines

GL Lines merupakan fungsi yang ada pada openGL untuk membuat sebuah object primitive berupa garis, garis ini dapat terbentuk dengan cara menyambungkan dua titik koordinat yang telah dibuat pada glVertex. Pada GL Lines, untuk merubah ukuran dapat menggunakan glLineWidth seperti contoh code berikut:

Code:

glLineWidth(3.0);

glBegin(GL\_LINES);

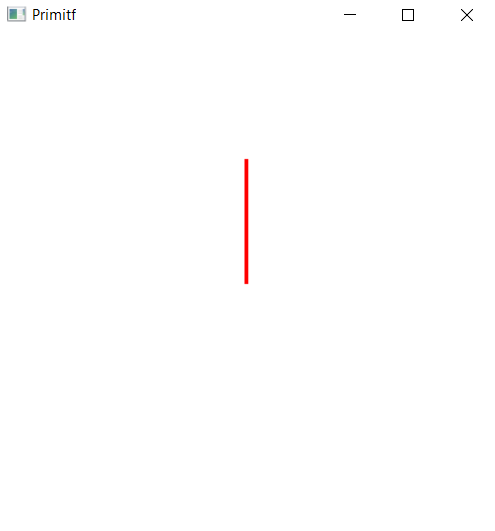
glVertex3f (0.0, 0.0, 0.0);

glVertex3f (0.0, 10.0, 0.0);

glEnd();

Pada fungsi glLineWidth terdapat sebuah argument bernilai 3.0 dengan type data float, fungsi ini memiliki artian yaitu untuk memperbesar ukuran GL\_LINES menajadi 3.0. Kemudian selanjutnya akan digambar sebuah object garis berdasarkan titik koordinat yang ada pada glVertex, glVertex yang pertama merupakan titik koordinat awal garis yang akan digambar, kemudian untuk glVertex yang kedua merupakan titik koordinat akhir garis tersebut.

Output:



1. GL Line Strip

Sama dengan GL Lines, GL Line Strip juga merupakan fungsi untuk menggambar sebuah object garis, akan tetapi terdapat perbedaan garis yang akan dibuat. Pada fungsi GL Line Strip titik kordinat pada masing – masing glVertex akan otomatis terhubung, sehingga akan membentuk sebuah segmen garis dari titik vertex awal hingga akhir seperti pada contoh code berikut:

Code:

glBegin(GL\_LINE\_STRIP);

glVertex3f (0.0, 0.0, 0.0);

glVertex3f (0.0, 8.0, 0.0);

glVertex3f (8.0, 0.0, 0.0);

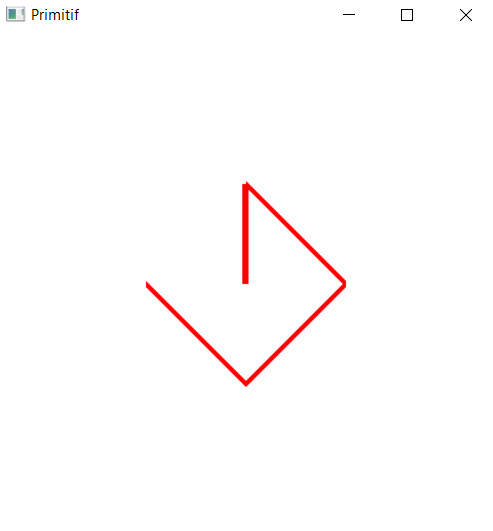
glVertex3f (0.0, -8.0, 0.0);

glVertex3f (-8.0, 0.0, 0.0);

glEnd();

Terdapat 5 titik koordinat pada glVertex yang sudah dideklarasikan didalam fungsi GL\_LINE\_STRIP, masing – masing titik koordinat ini nantinya akan otomatis terhubung satu sama lain sehingga akan terbentuk sebuah object garis yang saling terhubung berdasarkan titik koordinat yang sudah ditetapkan.

Output:



1. GL Line Loop

Sama dengan GL Lines dan GL Line Strip, GL Line Loop juga merupakan fungsi untuk menggambar sebuah object garis, akan tetapi terdapat perbedaan garis yang akan dibuat. Pada fungsi GL Line Loop titik kordinat pada masing – masing glVertex akan otomatis terhubung, sehingga akan membentuk sebuah segmen garis kemudian pada titik koordinat garis terakhir akan langsung dihubungkan kembali ke titik koordinat awal, sehingga garis object garis yang terbentuk akan menjadi terhubung seperti pada contoh code berikut:

Code:

glBegin(GL\_LINE\_LOOP);

glVertex3f (0.0, 0.0, 0.0);

glVertex3f (0.0, 8.0, 0.0);

glVertex3f (8.0, 0.0, 0.0);

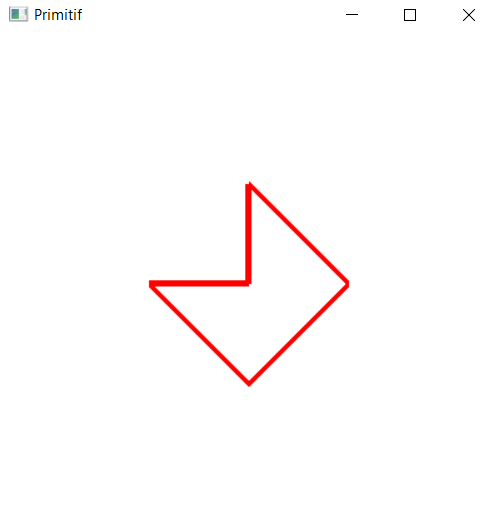
glVertex3f (0.0, -8.0, 0.0);

glVertex3f (-8.0, 0.0, 0.0);

glEnd();

Terdapat 5 titik koordinat pada glVertex yang sudah dideklarasikan didalam fungsi GL\_LINE\_LOOP, masing – masing titik koordinat ini nantinya akan otomatis terhubung satu sama lain sehingga akan terbentuk sebuah object garis yang saling terhubung berdasarkan titik koordinat yang sudah ditetapkan. Kemudian pada titik koordinat terakhir akan disambungkan ke titik koordinat awal, sehingga garis yang dibentuk menjadi terhubung.

Output:



1. **OpenGL Bangun Datar**

Pada tahap selanjutnya yaitu akan membuat sebuah object yang lebih kompleks, berbeda dengan OpenGL primitive pada bab sebelumnya, kali ini akan dibuat sebuah object bangun datar menggunakan fungsi GL Polygon, GL Triangles, GL Triangle Strip, GL Triangle Fan.

1. GL Polygon

GL Polygon merupakan sebuah fungsi yang ada di openGL untuk membuat sebuah bangun ruang yang terhubung dan didalamnya sudah terblok oleh warna / fill. Bangun yang dibentuk dapat dibuat dengan menyambungkan antara titik – titik koordinat pada glVertex, nantinya titik – titik koordinat tersebut akan dieksekusi oleh fungsi GL Polygon menjadi sebuah object bangun ruang 2 dimensi. Contoh program GL Polygon dapat dilihat pada baris code berikut:

Code:

glBegin(GL\_POLYGON);

glVertex3f (0.0, 0.0, 0.0);

glVertex3f (0.0, 8.0, 0.0);

glVertex3f (8.0, 0.0, 0.0);

glVertex3f (0.0, -8.0, 0.0);

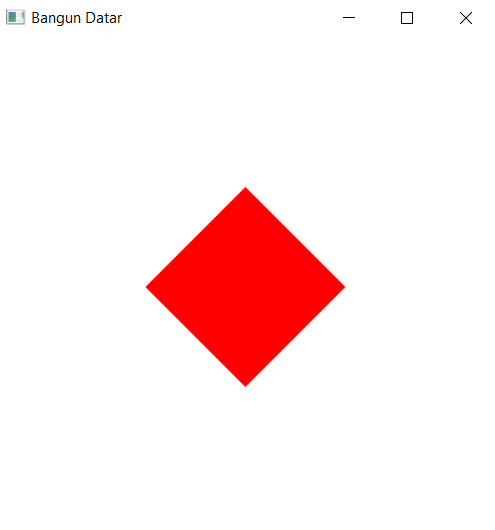
glVertex3f (-8.0, 0.0, 0.0);

glVertex3f (0.0, 8.0, 0.0);

glEnd();

Pada masing – masing titik koordina tersebut nantinya akan saling terhubung satu sama lain, kemudian akan menyambungkan titik koordinat awal hingga titik koordinat akhir yang didalamnya akan langsung terblok/terfill sehingga akan ikut berwarna dan menjadi sebuah object 2 dimensi seperti output berikut.

Output:



1. GL Triangles

GL Triangles merupakan sebuah fungsi untuk membuat sebuah object bangun datar segitiga, object segitiga ini dibuat dengan menghubungkan 3 titik koordinat pada glVertex. Apabila titik koordinat pada glVertex kurang dari 3, maka tidak akan terbentuk sebuah object segitiga, begitu pula sebaliknya apabila sudah 3 ada titik koordinat glVertex maka titik koordinat glVertex yang lain akan diabaikan. Artinya untuk menggunakan fungsi GL Triangles diwajibkan untuk memiliki/mendeklarasikan minimal 3 glVertex agar object segitiga dapat terbentuk. Contoh program GL Triangles dapat dilihat pada baris code berikut:

Code:

glBegin(GL\_TRIANGLES);

glVertex3f (0.0, -8.0, 0.0);

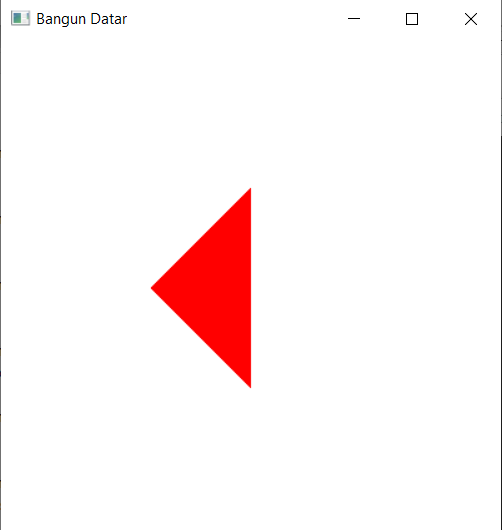
glVertex3f (-8.0, 0.0, 0.0);

glVertex3f (0.0, 8.0, 0.0);

glEnd();

Ketiga titik koordinat yang ada di dalam fungsi GL\_TRIANGLES di atas akan membentuk sebuah segitaga yang sudah diblok/difill, karena sudah sesuai memenuhi tiga titik yang diperlukan untuk membuat sebuah segitiga, apabila ditambahkan satu glVertex lagi dibawahnya, maka glVertex yang baru ini akan diabaikan dan hanya menggambar ada tiga titik awal koordinat yang sudah ditentukan.

Output:



1. GL Triangle Strip

GL Triangle Strip merupakan sebuah fungsi untuk membuat sebuah object bangun datar segitiga, object segitiga ini dibuat dengan menghubungkan 3 titik koordinat pada glVertex. Berbeda dengan GL Triangles, pada GL Triangle strip untuk membuat segitiga tambahan kita tidak perlu menambahkan 3 titik koordinat glVertex lagi, cukup dengan menambahkan 1 titik koordinat glVertex sudah dapat membentuk segitiga yang baru, dan masing – masing segitiga tersebut akan terkait. Hal ini sangat berguna untuk mengurangi banyaknya data dan waktu yang akan diproses oleh program. Contoh program GL Triangle Strip dapat dilihat pada baris code berikut:

Code:

glBegin(GL\_TRIANGLE\_STRIP);

glVertex3f (8.0, 0.0, 0.0);

glVertex3f (0.0, -8.0, 0.0);

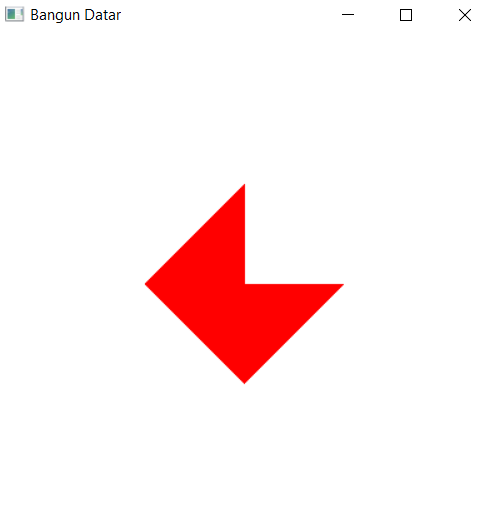
glVertex3f (-8.0, 0.0, 0.0);

glVertex3f (0.0, 8.0, 0.0);

glEnd();

Ketiga titik koordinat awal yang ada di dalam fungsi GL\_TRIANGLE\_STRIP di atas akan membentuk sebuah segitaga yang sudah diblok/difill, karena sudah sesuai memenuhi tiga titik yang diperlukan untuk membuat sebuah segitiga. Kemudian agar dapat membentuk segitiga lagi, ditambahkan 1 titik koordinat glVertex agar dapat terbentuk sebuah segitiga baru yang akan digambar.

Output:



1. GL Triangle Fan

Sama halnya dengan GL Triangles dan GL Triangle Strip, GL Triangle Fan juga merupakan sebuah fungsi untuk membuat sebuah segitiga. Pada GL Triangle fan juga membutuhkan 3 titik koordinat awal untuk membentuk sebuah segitiga. Dan hanya perlu tambahan 1 titik koordinat glVertex agar dapat membentuk sebuah object segitiga yang baru. Perbedannya hanya pada fungsi GL Triangle Fan akan menarik garis tambahan dari vertex awal. Contoh program GL Triangle Fan dapat dilihat pada baris code berikut:

Code:

glBegin(GL\_TRIANGLE\_FAN);

glVertex3f (0.0, 0.0, 0.0);

glVertex3f (0.0, 8.0, 0.0);

glVertex3f (8.0, 0.0, 0.0);

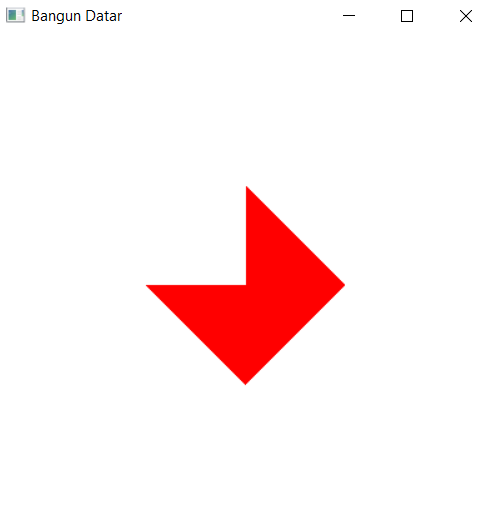
glVertex3f (0.0, -8.0, 0.0);

glVertex3f (-8.0, 0.0, 0.0);

glEnd();

Ketiga titik koordinat awal yang ada di dalam fungsi GL\_TRIANGLE\_STRIP di atas akan membentuk sebuah segitaga yang sudah diblok/difill. Kemudian untuk menambah sebuah segitiga baru, ditambahkan sebuah 1 titik koordinat glVertex dan nantinya garis tersebut akan ditarik dari titik koordinat glVertex awal.

Output:



1. **Pewarnaan Object**

Pada openGL pewarnaan dispesifikasikan sebagai gabungan intensitas komponen antara RGB (Red, Green, Blue). Berdasarkan gabungan dari ketiga komponen tersebut, terbentuklah suatu warna dari kombinasi RGB yang dapat digunakan. Perintah / code untuk mengatur warna dalam openGL secara umum menggunakan **glColor3f(red, green, blue);** dengan rentang nilai yang dapat digunakan yaitu 0 sampai dengan 1.

Berikut merupakan tabel warna dasar yang dapat digunakan dalam komposisi RGB:

|  |  |  |  |
| --- | --- | --- | --- |
| **WARNA** | **Red** | **Green** | **Blue** |
| Merah | 1 | 0 | 0 |
| Hijau | 0 | 1 | 0 |
| Biru | 0 | 0 | 1 |
| Kuning | 1 | 1 | 0 |
| Magenta | 1 | 0 | 1 |
| Cyan | 0 | 1 | 1 |
| Putih | 1 | 1 | 1 |
| Hitam | 0 | 0 | 0 |
| Abu - Abu | 0.5 | 0.5 | 0.5 |

Kombinasi RGB tersebut dapat digunakan dan dimasukkan sebagai nilai argument pada code **glColor3f(red, green, blue);**. Berikut merupakan contoh code pewarnaan object dengan warna hijau:

Code:

glColor3f(0.0, 1.0, 0.0);

glBegin(GL\_TRIANGLES);

glVertex3f (0.0, -8.0, 0.0);

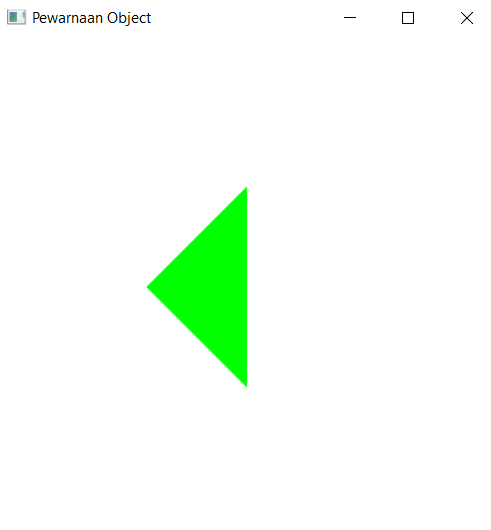
glVertex3f (-8.0, 0.0, 0.0);

glVertex3f (0.0, 8.0, 0.0);

glEnd();

Pada baris code glColor3f untuk menghasilkan warna hijau, argument yang diisi adalah (0.0, 1.0, 0.0) maka warna pada object akan berubah menjadi hijau, warna ini berlaku terus menerus, dan akan berhenti apabila terdapat glColor dengan komposisi RGB terbaru. Untuk menghasilkan warna baru dapat mengkombinasikan nilai RGB dengan rentang nilai 0 samapi 1.

Output:



Pewarnaan pada openGL tidak terbatas hanya dengan menggunakan **glColor3f(red, green, blue);** pewarnaan object pada openGL juga bisa menggunakan code lain seperti **glColor3ub(red, green, blue);** berbeda dengan code sebelumnya, pada glColor3ub nilai RGB memiliki rentang 0 – 255. Nilai tersebut dapat dikombinasikan sehingga akan menghasilkan warna baru. Berikut merupakan contoh code pewarnaan object menggunakan glColor3ub:

Code:

glColor3ub(46, 227, 229);

glBegin(GL\_TRIANGLES);

glVertex3f (0.0, -8.0, 0.0);

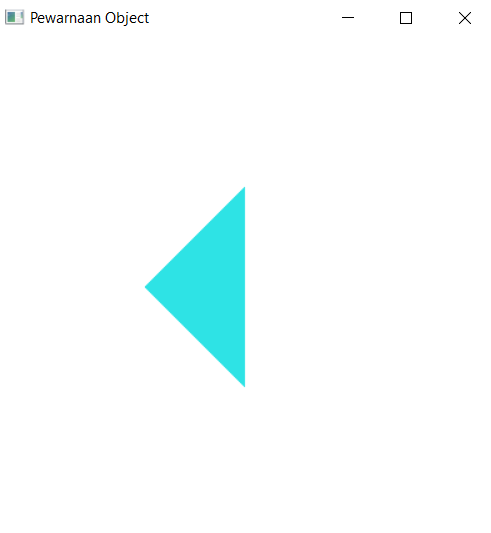
glVertex3f (-8.0, 0.0, 0.0);

glVertex3f (0.0, 8.0, 0.0);

glEnd();

Sesuai dengan penjelasan sebelumnya, pada glColor3ub rentang nilai yang dapat dijadikan sebagai argument RGB adalah 0 – 255, sehingga akan menghasilkan nilai yang baru.

Output:



Agar menhasilkan warna yang berbeda dan lebih unik pada object, proses pewarnaan juga dapat dilakukan pada setiap vertex, dengan begitu warna pada masing – masing vertex akan berbeda dan menghasilkan perpaduan warna yang unik sperti contoh code berikut:

Code:

glBegin(GL\_TRIANGLES);

glColor3f(0.0, 1.0, 0.0);

glVertex3f (0.0, -8.0, 0.0);

glColor3f(1.0, 0.0, 0.0);

glVertex3f (-8.0, 0.0, 0.0);

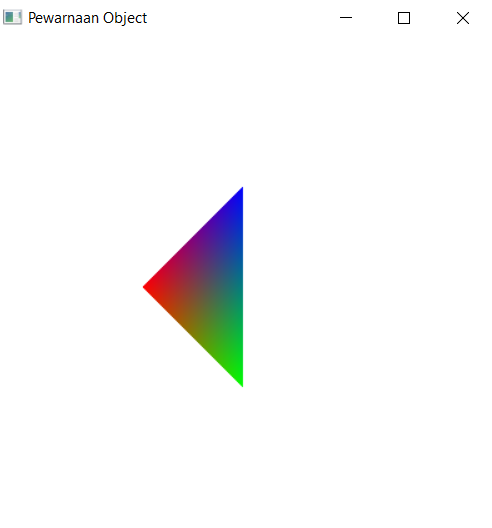
glColor3f(0.0, 0.0, 1.0);

glVertex3f (0.0, 8.0, 0.0);

glEnd();

Sesuai dengan penjelasan sebelumnya, object akan memiliki perpaduan warna yang unik jika pada setiap vertex diberikan warna yang berbeda. Seperti pada code diatas. Setiap glVertex baru yang dibuat maka akan ditambahkan glColor baru, nantinya object yang terbentuk akan memiliki perpaduan warna yang unik seperti output berikut.

Output:



1. **Object dan Kurva**

Kurva merupakan suatu garis yang tidak harus lurus, sedangkan garis lurus merupakan kurva khusus yang tidak memiliki lengkungan. Kurva dapat dibuat dengan mengunakan pendekatan naïve, dimana membuat beberapa titik demi titik yang saling bersebelahan. Beberapa contoh bentuk kurva adalah: grafik sinus, lintasan parabola, grafik persamaan logaritma, dll.

1. Kurva Polinomial

Polynomial adalah persamaan matematika dalam bentuk persamaan linear. Jika digambar persemaan linear dapat membentuk sebuah garis lurus.

Contoh code kurva polynomial:

glBegin(GL\_LINES);

glColor3f(0.0, 0.0, 0.0);

glVertex2i(20,0);

glVertex2i(-20,0);

glVertex2i(0,20);

glVertex2i(0,-20);

glEnd();

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_LINE\_STRIP);

for(float t =-10.0; t<=10.0; t+=0.1)

{

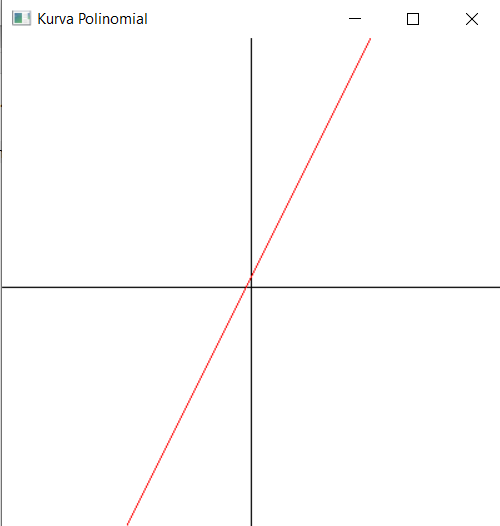
glVertex2f(t,2\*t+1);

}

glEnd();

Untuk menghasilkan kurva polynomial atau berupa kurva berbentuk garis lurus adalah dengan menghubungkan titik – titik agar menjadi sebuah garis. Pada code diatas titik – titik tersebut dbuat menggunakan perulangan for, dimana dilakukan penambahan nilai t += 0.1 pada setiap perulangan dengan kondisi awal t = -10 sampai kondisi akhir t = 10. Kemudian setiap t yang sudah melakukan perulangan dimasukkan ke dalam glVertex yang sudah didefinisikan seperti rumus persamaan linear, sehingga nantinya akan membentuk suatu kurva lurus seperti berikut.

Output:



1. Kurva Polinomial 2

Polynomial 2 adalah persamaan matematika dalam bentuk persamaan kuadrat. Jika digambar persemaan kuadrat dapat membentuk sebuah garis parabola.

Contoh code kurva polynomial:

glBegin(GL\_LINES);

glColor3f(0.0, 0.0, 0.0);

glVertex2i(20,0);

glVertex2i(-20,0);

glVertex2i(0,20);

glVertex2i(0,-20);

glEnd();

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_LINE\_STRIP);

for(float t =-10.0; t<=10.0; t+=0.1)

{

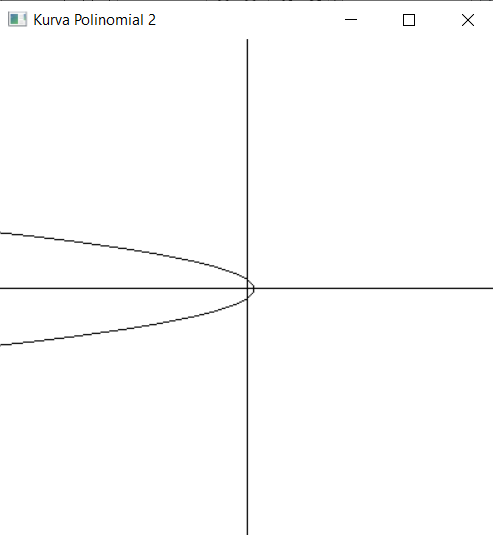
glVertex3f (0.5-t\*t, t, 0.0);

}

glEnd();

Untuk menghasilkan kurva polynomial 2 atau berupa kurva berbentuk garis parabola adalah dengan menghubungkan titik – titik agar menjadi sebuah garis. Pada code diatas titik – titik tersebut dbuat menggunakan perulangan for, dimana dilakukan penambahan nilai t += 0.1 pada setiap perulangan dengan kondisi awal t = -10 sampai kondisi akhir t = 10. Kemudian setiap t yang sudah melakukan perulangan dimasukkan ke dalam glVertex yang sudah didefinisikan seperti rumus persamaan kuadrat, sehingga nantinya akan membentuk suatu kurva parabola seperti berikut.

Output:



1. Kurva Polynomial 3

Polynomial 3 hampir sama dengan polynomial 2, akan tetapi pada polynomial 3 ini memiliki bentuk yang lebih kompleks.

Contoh code kurva polynomial:

glBegin(GL\_LINES);

glColor3f(0.0, 0.0, 0.0);

glVertex2i(20,0);

glVertex2i(-20,0);

glVertex2i(0,20);

glVertex2i(0,-20);

glEnd();

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_LINE\_STRIP);

for(float t =-10.0; t<=10.0; t+=0.1)

{

//glVertex2f(t,2\*t+1);

//glVertex3f (0.5-t\*t, t, 0.0);

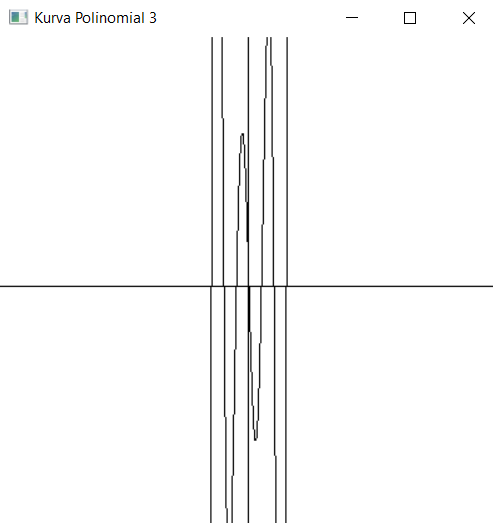
glVertex3f(t,(t-3)\*(t-2)\*(t-1)\*(t)\*(t+1)\*(t+2)\*(t+3),0.0);

}

glEnd();

Untuk menghasilkan kurva polynomial 3 adalah dengan menghubungkan titik – titik agar menjadi sebuah garis. Pada code diatas titik – titik tersebut dibuat menggunakan perulangan for, dimana dilakukan penambahan nilai t += 0.1 pada setiap perulangan dengan kondisi awal t = -10 sampai kondisi akhir t = 10. Kemudian setiap t yang sudah melakukan perulangan dimasukkan ke dalam glVertex yang sudah didefinisikan seperti rumus polynomial 3 sehingga nantinya akan membentuk suatu kurva seperti berikut.

Output:



1. Kurva Trigonometri

Kurva trigonometri adalah kurva yang dihasilkan dari fungsi-fungsi trigonometri: sinus, cosinus, dan tangen.

Contoh code kurva trigonometri:

glBegin(GL\_LINES);

glColor3f(0.0, 0.0, 0.0);

glVertex2i(20,0);

glVertex2i(-20,0);

glVertex2i(0,20);

glVertex2i(0,-20);

glEnd();

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_LINE\_STRIP);

float A=5;float B=2;float C=3;float D=1.2;

for(float t =-10.0; t<=10.0; t+=00.1)

{

//glVertex2f(t,2\*t+1);

//glVertex3f (0.5-t\*t, t, 0.0);

//glVertex3f(t, (t-3)\*(t-2)\*(t-1)\*(t)\*(t+1)\*(t+2)\*(t+3),0.0);

glVertex2f(t,A\*sin(B\*t + C) + D);

}

glEnd();

Untuk menghasilkan kurva Trigonometri sinus adalah dengan menghubungkan titik – titik agar menjadi sebuah garis. Pada code diatas titik – titik tersebut dibuat menggunakan perulangan for, dimana dilakukan penambahan nilai t += 00.1 pada setiap perulangan dengan kondisi awal t = -10 sampai kondisi akhir t = 10. Kemudian setiap t yang sudah melakukan perulangan dimasukkan ke dalam glVertex yang sudah didefinisikan seperti rumus trigonometri sinus sehingga nantinya akan membentuk suatu kurva seperti berikut.

Output:

