**LAPORAN**

**MEMBANGUN ANIMASI 3D PERPUTARAN PLANET GALAKSI BIMA SAKTI**

**TERHADAP MATAHARI DENGAN MEMANFAATAN**

**GRAFIKA KOMPUTER**

**MATA KULIAH : KOMPUTER GRAFIK**

**DOSEN : Hendri Karisma, S.Kom.**

Oleh :

**Qori Nurqolbi (10108424)**

**Nur Mufmin (10108447)**

**Idham Herawan Putra (10108450)**

**Vian Arwanda (10108467)**

**PROGRAM STUDI S1**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK DAN ILMU KOMPUTER**

**UNIVERSITAS KOMPUTER INDONESIA**

**2012**

**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang Masalah**

Perkembangan teknologi yang sangat pesat sekarang ini tidak dapat kita hindari. Di sini komputer merupakan salah satu hasil dari perkembangan IPTEK yang semakin maju. Penggunaan pekerjaan dengan komputer dapat menyelesaikan pekerjaan dengan lebih efektif dan efisien. Penciptaan akan sebuah teknologi merupakan sesuatu yang sangat berharga. Inovasi dan kreatifitas yang setiap orang ciptakan menjadi sebuah karya dimana yang nantinya bisa menjadi sebuah nilai jual.

Terdapat banyak bintang, nebula, dan gugus bintang yang bisa diamati di langit setiap malamnya. Semua objek tersebut berada di dalam galaksi kita. Di beberapa bagian bintang nampak padat sehingga ketika langit cerah, bersih dari awan, dan kondisi sekitar yang gelap, kita bisa melihat pita berwarna putih yang memanjang dan melintasi beberapa rasi seperti Sagittarius (arah pusat Galaksi), Scorpius, Ophiucus, Aquila, Cassiopeia, Auriga, Crux, dan Centaurus. Sementara di bagian yang lain tampak celah-celah gelap yang menunjukkan adanya materi antar bintang yang tebal.

Bintang-bintang itu mempunyai cahaya sendiri, matahari adalah bintang karena matahari mempunyai cahaya sendiri dan merupakan bintang terdekat dengan bumi walaupun jaraknya masih sekitar 149.500.000 kilometer dan matahari sebagai salah satu bintang yang terdapat pada Galaxi Bima Sakti. Kadang-kadang kita melihat benda-benda langit yang berpindah-pindah diantara bintang-bintang, benda-benda langit itu adalah planet. Berbeda dengan bintang planet tidak mempunyai cahaya sendiri. Sampai sekarang orang mengenal sembilan buah planet pada tata surya kita dan bumi merupakan salah satu planet diantaranya. Kesembilan planet itu beredar mengelilingi matahari, garis edar atau lintasan itu disebut orbit. Bidang edar planet yaitu bidang tempat planet-planet beredar, setiap planet mempunyai bidang edar sendiri-sendiri. Satu sama lain hampir berhimpitan dan bidang edar planet bumi dinamakan bidang ekliptika. Telah disebutkan diatas bahwa setiap planet tidak mempunyai cahaya sendiri, cahaya yang dikeluarkan oleh planet-planet adalah pantulan cahaya matahari yang mengenai planet-planet tersebut. Cahaya matahari yang mengenai planet-planet hanya separuh dari bola planet. Planet yang mendapatkan cahaya matahari mengalami siang dan bagian yang tidak mendapat cahaya matahari mengalami malam.

Dilihat dari cerita diatas dapat disimulasikan suatu perputaran planet terhadap matahari dengan suatu animasi. Saat ini dengan kenyataan yang dapat dirasakan adalah kemudahan dalam proses pembuatan animasi. Dengan bantuan komputer dan grafika komputer, pembuatan animasi menjadi sangat mudah dan cepat. Bahkan akhir-akhir ini banyak bermunculan animasi 3 dimensi untuk mensimulasikan suatu keadaan seperti yang akan kami buat yaitu perputaran planet terhadap matahari. Dengan adanya simulasi dengan menggunakan komputer dan grafika komputer ini maka manusia tidak akan lagi hanya membayangkannya saja tapi akan melihat suatu animasi yang menarik.

Simulasi yang kami buat ini tentunya akan ada matahari yang menyilaukan dan menyinari planet-planetnya. Akan ada banyak bulatan-bulatan untuk planet dan matahari yang bercahaya. Cahaya matahari ini akan menyinari planet yang mendekatinya. Agar perputaran planetnya terlihat rapi dan jelas, dalam simulasi ini akan dibantu oleh orbit sehingga perputarannya tidak keluar dari batas orbit. Walaupun pembuatan simulasi ini tidak akan sama seperti aslinya, namun setidaknya kita akan berimajinasi tentang bagaimana putaran planet terhadap matahari.

* 1. **Rumusan Masalah**

Perumusan masalah pada tugas grafika komputer ini yaitu:

1. Bagaimana mensimulasikan grafis OpenGL yang didukung pemrograman Visual C++.
2. Bagaimana menunjukkan perputaran planet terhadap matahari dalam bentuk grafik sehingga posisi perputarannya dapat diketahui.
   1. **Maksud dan Tujuan**

Berdasarkan permasalahan yang diteliti, maka maksud dari penulisan tugas ini adalah untuk **Membangun Animasi 3D Perputaran planet galaksi bima sakti terhadap matahari dengan memanfaatan grafika komputer.**

Sedangkan tujuan yang akan dicapai dalam penelitian ini adalah :

1. Dapat mensimulasikan grafis OpenGL yang didukung pemrograman Visual C++
2. Dapat membantu manusia untuk menunjukkan perputaran planet terhadap matahari dalam bentuk grafik
   1. **Batasan Masalah**

Pembatasan masalah dilakukan agar penulisan tugas ini dapat memberikan pemahaman yang terarah dan sesuai dengan yang diharapkan. Agar pembahasan tidak menyimpang dari pokok perumusan masalah yang ada, maka penulis membatasi permasalahan pada :

1. Pemrograman hanya menggunakan bahasa C++
2. Menunjukan perputaran planet terhadap matahari dengan animasi 3D
3. Di animasi ini hanya ada 9 perputaran planet dan ditengah-tengahnya ada berupa matahari berwarna kuning
   1. **Metodologi Penelitian**

Metodologi yang digunakan dalam penulisan tugas ini adalah sebagai berikut :

1. Tahap pengumpulan data

Metode pengumpulan data yang digunakan dalam penelitian ini adalah sebagai berikut :

1. Studi Literatur.

Pengumpulan data dengan cara mengumpulkan literatur, jurnal, *paper* dan bacaan-bacaan yang ada kaitannya dengan judul penelitian.

1. Observasi.

Teknik pengumpulan data dengan mengadakan penelitian dan peninjauan langsung terhadap permasalahan yang diambil.

1. Interview.

Teknik pengumpulan data dengan mengadakan tanya jawab secara langsung yang ada kaitannya dengan topik yang diambil.

1. Tahap pembuatan perangkat lunak dengan SDLC (Systems Development Life Cycle)

Metode penelitian dalam tugas ini menggunakan SDLC (System Development Life Cycle) yaitu proses logis yang digunakan oleh analis sistem untuk menggambarkan sebuah sistem informasi, termasuk di dalamnya persyaratan, validasi, pelatihan dan kepemilikan. SDLC merupakan siklus pengembangan sistem. Pengembangan sistem teknik (engineering system development). Meliputi langkah berikut:

1. Planning / perencanaan

Untuk menghasilkan perangkat lunak (*software*) yang berkualitas perlu dilakukan perencanaan yang matang dengan melakukan studi kelayakan. Studi kelayakan yang dilakukan meliputi : ekonomi, operasional, dan teknis.

1. Analisa

Tujuan dari analisa sistem adalah untuk menentukan masalah upaya untuk memperbaiki sistem. Sehingga diharapkan dengan dilakukannya analisa sitem, maka permasalahan yang ada akan dapat teratasi.

1. Desain

System design menguraikan layar layout, aturan bisnis, proses diagram dan dokumentasi lainnya. Hasil dari tahap ini akan menjelaskan sistem baru sebagai kumpulan modul atau subsistem.

1. Implementasi (Build and Coding)

Pada tahapan ini dilakukan implementasi dari perancangan dan desain yang telah dilakukan. Sehingga pada tahap ini menghasilkan suatu perangkat lunak (*software*).

1. Testing / Pengujian

Setelah perangkat lunak dibangun, maka dilakukan pengujian untuk menguji tingkat kehandalan perangkat lunak yang telah dibangun. Hal ini dilakukan untuk memastikan kehandalan software.

1. Pemeliharaan / Mintenance

Pemeliharaan merupakan tahap penting dalam SDLC. Tahap ini dilakukan untuk memperbaiki sistem yang telah dibangun. Selain itu tahapan ini juga untuk penambahan dan perubahan sistem.

* 1. **Sistematika Penelitian**

Sistematika penulisan proposal penelitian ini disusun untuk memberikan gambaran umum tentang penelitian yang dijalankan. Sistematika secara umum adalah sebagai berikut :

BAB I PENDAHULUAN

Menguraikan tentang dasar-dasar pemikiran yang berisi tentang latar belakang masalah, perumusan masalah, maksud dan tujuan, batasan masalah, metode penelitian, serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini memaparkan tentang teori-teori yang berkaitan dengan topik dan menunjang dalam perancangan sistem yang digunakan sebagai acuan dalam pembuatan perangkat lunak.

BAB III PEMBAHASAN

Pada bab ini dijelaskan langkah-langkah perancangan dan pembuatan animasi yang dibuat

BAB IV KESIMPULAN DAN SARAN

Berisi kesimpulan dan saran yang sudah diperoleh dari hasil penulisan tugas.

**BAB II**

**TINJAUAN PUSTAKA**

1. **Pengertian Grafika Komputer**

Komputer Grafis adalah cabang dari Ilmu Komputer dan berkaitan dengan manipulasi visual content dan proses sistesisnya secara digital. Walaupun istilah ini sering mengacu kepada komputer grafik 3 dimensi, tetapi sebenarnya juga mencakup grafik 2 dimensi dan pengolahan citra.

Grafik meliputi gambar dan pencitraan lain yang dihasilkan oleh komputer berbentuk garis, lengkungan, kurva dan sebagainya. Komputer dapat menghasilkan pencitraan dalam sejumlah pixel, dan printer dot matrix akan mencetak citra/gambar tersebut dalam sejumlah titik.

Perkembangan grafika komputer menuntut para pengembang sistem aplikasi grafika komputer untuk dapat mengembangkan suatu informasi yang dilengkapi dengan visualisasi dan animasi, agar dapat lebih mudah dipahami oleh pihak yang menggunakan sistem tersebut. Grafika komputer telah menunjukkan kemajuan yang pesat dalam pengembangan berbagai aplikasi untuk menghasilkan gambar.

Grafika komputer digunakan untuk menunjang berbagai bidang dengan teknologi grafika berbasis komputer. Penggunaan grafika komputer telah lama digunakan dalam beberapa macam aplikasi, diantaranya pendidikan, kedokteran, fisika, matematika, multimedia, dan lain-lain. Pada saat ini grafika komputer sudah digunakan pada bidang sains, engineering, kedokteran, bisnis, industri, pemerintahan, seni, hiburan, iklan, pendidikan, dan lain-lain. Oleh karena itu, sudah semakin banyak pula bahasa pemrograman yang dilengkapi dengan tools/library pembuatan grafik Salah satu tools/library pembuatan aplikasi grafik adalah OpenGL (Open Graphics Library). OpenGL (Open Graphics Library) adalah suatu spefikasi grafik yang low-level yang menyediakan fungsi untuk pembuatan grafik primitif termasuk titik, garis, dan lingkaran. OpenGL digunakan untuk mendefinisikan suatu objek, baik objek 2 dimensi maupun objek 3 dimensi.

OpenGL juga merupakan suatu antarmuka pemrograman aplikasi (applicationprogramming interface (API) yang tidak tergantung pada piranti dan platform yangdigunakan, sehingga OpenGL dapat berjalan pada sistem operasi Windows, UNIX dansistem operasi lainnya. OpenGL pada awalnya didesain untuk digunakan pada bahasapemrograman C/C++, namun dalam perkembangannya OpenGL dapat juga digunakandalam bahasa pemrograman yang lain seperti Java, Tcl, Ada, Visual Basic, Delphi,maupun Fortran. Namun OpenGL di-package secara berbeda-beda sesuai dengan bahasapemrograman yang digunakan.

Grafika komputer pada dasarnya adalah suatu bidang ilmu komputer yang mempelajaricara-cara untuk meningkatkan dan memudahkan komunikasi antara manusia denganmesin (komputer) dengan jalan membangkitkan, menyimpan, dan memanipulasi gambar model suatu objek menggunakan komputer. Grafika komputer memungkinkan kita untuk berkomunikasi lewat gambar-gambar, bagan-bagan, dan diagram-diagram. Grafikakomputer dikembangkan melalui suatu sistem operasi yang berbasis GUI (Graphical UserInterface). Graphical User Interface (GUI), dalam hubungannya dengan sains komputer,adalah suatu antarmuka berbentuk tampilan yang memungkinkan seorang user untuk memilih perintah, menjalankan program, melihat serangkaian file dan memilih opsi laindengan menunjukkan representasi gambar (icon) ataupun melalui sejumlah menu padalayar komputer. Beberapa kemajuan utama dalam computer grafik 3D:

1. Flat shading 🡪 suatu teknik shades masing-masing polygon dari suatu objek berdasarkan pada polygon “normal” dan posisi serta intensitas sumber cahaya.
2. Gouraud shading 🡪 ditemukan oleh Henri Gouraud pada tahun 1971 dengan teknik resource-conscious yang digunakan untuk menirukan shade dengan permukaanlembut dan penyisipan warna puncak antarpermukaan polygon.
3. Texture mapping 🡪 suatu teknik untuk menirukan detail permukaan denganpemetaan gambar (tekstur) menjadi polygons.
4. Phong shading 🡪 ditemukan oleh Bui Tuong Phong; suatu teknik shading yanglembut penyisipan yang puncak mendekati normal pencahayaan dari polygoncurved-surface dengan antarpermukaan; model pencahayaan meliputi glossyreflection dengan suatu tingkatan permukaan yang halus.
5. Bump mapping 🡪 ditemukan oleh Jim Blinn, suatu teknik normal-perturbationyang digunakan untuk menirukan permukaan yang tidak rata atau mengerut.
6. Ray Tracing 🡪 suatu metode berdasarkan pada prinsip fisik dari ilmu opticgeometris yang bisa menirukan pantulan berulang dan transparan.
7. Radiosity 🡪 suatu teknik untuk global illumination yang menggunakan teoriperpindahan radiatif untuk menirukan iluminasi secara tidak langsung (yangdicerminkan)
8. **Cabang ilmu Grafik**

Cabang ilmu Grafik yaitu :

1. Pemrosesan citra (image processing) adalah Proses ini mempunyai ciri data masukan dan informasi keluaran yang berbentuk citra. Istilah pengolahan citra digital secara umum didefinisikan sebagai pemrosesan citra dua dimensi dengan komputer.Umumnya citra digital berbentuk persegi panjang atau bujur sangkar (pada beberapa sistem pencitraan ada pula yang berbentuk segienam) yang memiliki lebar dan tinggi tertentu. Ukuran ini biasanya dinyatakan dalam banyaknya titik atau piksel sehingga ukuran citra selalu bernilai bulat. Setiap titik memiliki koordinat sesuai posisinya dalam citra. Koordinat ini biasanya dinyatakan dalam bilangan bulat positif, yang dapat dimulai dari 0 atau 1 tergantung pada sistem yang digunakan. Setiap titik juga memiliki nilai berupa angka digital yang merepresentasikan informasi yang diwakili oleh titik tersebut. Format data citra digital berhubungan erat dengan warna.
2. Pengenalan pola (pattern recognition) dapat dikatakan sebagai kemampuan manusia mengenali objek-objek berdasarkan ciri-ciri dan pengetahuan yang pernah diamatinya dari objek-objek tersebut. Tujuan dari pengenalan Pola ini adalah mengklasifikasi dan mendeskripsikan pola atau objek kompleks melalui pengetahuan sifat-sifat atau ciri-ciri objek tersebut. Apakah pola itu, pola dapat dikatakan sebagai identitas yang terdefinis dan dapat diberi suatu identifikasi atau nama. Pendekatan pengenalan pola ada 3 yaitu secara sintaks, statistik serta melalui jaringan saraf tiruan. Pendekatan secara sintaks adalah pendekatan dengan menggunakan aturan aturan tertentu. Pendekatan metoda statistik adalah pendektan dengan menggunakan data-data yang berasal dari statisik seperti pasar saham. Pendekatan dengan pola jaringan saraf tiruan adalah pendekatan melalui pola-pola ini meniru cara kerja otak manusia, Pada pola ini sistem membuat rule-rule tertentu disertai dengan menggunakan data statistik sebagai dasar untuk pengambilan keputusan.
3. Bagian dari grafik komputer meliputi:
4. Geometri: mempelajari cara menggambarkan permukaan bidang
5. Animasi: mempelajari cara menggambarkan dan memanipulasi gerakan
6. Rendering: mempelajari algoritma untuk menampilkan efek cahaya
7. Citra (Imaging): mempelajari cara pengambilan dan penyuntingan gambar.

Dalam Pembuatan sebuah desain kita juga perlu memperhatikan bentuk desain yang Anda inginkan. Tentunya supaya desain Anda dapat dilihat bagus (sesuai maksud dan tujuan Anda membuatnya), maka unsur-unsur pembuatan desain yang perlu diperhatikan adalah :

1. Garis (Line) Sebuah garis adalah unsur desain yang menghubungkan antara satu titik poin dengan titik poin yang lain sehingga bisa berbentuk gambar garis lengkung (curve) atau lurus (straight). Garis adalah unsur dasar untuk membangun bentuk atau konstruksi desain.
2. Bentuk (Shape) Bentuk adalah segala hal yang memiliki diameter tinggi dan lebar. Bentuk dasar yang dikenal orang adalah kotak (rectangle), lingkaran (circle), dan segitiga (triangle). Sementara pada kategori sifatnya, bentuk dapat dikategorikan menjadi tiga, yaitu:
3. Huruf (Character) : yang direpresentasikan dalam bentuk visual yang dapat digunakan untuk membentuk tulisan sebagai wakil dari bahasa verbal dengan bentuk visual langsung, seperti A, B, C, dsb.
4. Simbol (Symbol) : yang direpresentasikan dalam bentuk visual yang mewakili bentuk benda secara sederhana dan dapat dipahami secara umum sebagai simbol atau lambang untuk menggambarkan suatu bentuk benda nyata, misalnya gambar orang, bintang, matahari dalam bentuk sederhana (simbol), bukan dalam bentuk nyata (dengan detail).
5. Bentuk Nyata (Form) : bentuk ini betul-betul mencerminkan kondisi fisik dari suatu obyek. Seperti gambar manusia secara detil, hewan atau benda lainnya.
6. Tekstur (Texture) Tekstur adalah tampilan permukaan (corak) dari suatu benda yang dapat dinilai dengan cara dilihat atau diraba. Yang pada prakteknya, tekstur sering dikategorikan sebagai corak dari suatu permukaan benda, misalnya permukaan karpet, baju, kulit kayu, dan lain sebagainya.
7. Ruang (Space) Ruang merupakan jarak antara suatu bentuk dengan bentuk lainnya yang pada praktek desain dapat dijadikan unsur untuk memberi efek estetika desain. Sebagai contoh, tanpa ruang Anda tidak akan tahu mana kata dan mana kalimat atau paragraf. Tanpa ruang Anda tidak tahu mana yang harus dilihat terlebih dahulu, kapan harus membaca dan kapan harus berhenti sebentar. Dalam bentuk fisiknya pengidentifikasian ruang digolongkan menjadi dua unsur, yaitu obyek (figure) dan latar belakang (background).
8. Ukuran (Size) Ukuran adalah unsur lain dalam desain yang mendefinisikan besar kecilnya suatu obyek. Dengan menggunakan unsur ini Anda dapat menciptakan kontras dan penekanan (emphasis) pada obyek desain anda sehingga orang akan tahu mana yang akan dilihat atau dibaca terlebih dahulu.
9. Warna (Color) Warna merupakan unsur penting dalam obyek desain. Karena dengan warna orang bisa menampilkan identitas, menyampaikan pesan atau membedakan sifat dari bentuk-bentuk bentuk visual secara jelas. Dalam prakteknya warna dibedakan menjadi dua: yaitu warna yang ditimbulkan karena sinar (Additive color/RGB) yang biasanya digunakan pada warna lampu, monitor, TV dan sebagainya, dan warna yang dibuat dengan unsur-unsur tinta atau cat (Substractive color/CMYK) yang biasanya digunakan dalam proses pencetakan gambar ke permukaan benda padat seperti kertas, logam, kain atau plastik.

Bentuk sederhana dari grafik komputer ada 2 adalah yaitu :

1. Grafik komputer 2 dimensi biasa disebut dengan 2D atau bidang adalah bentuk dari benda yang memiliki panjang dan lebar. Grafik 2 Dimensi merupakan teknik penggambaran yang berpatokan pada titik koordinat sumbu x (datar) dan sumbu y (tegak). Agar dapat tampil dengan sempurna, gambar yang akan ditampilkan dengan teknik ini harus memiliki nilai koordinat x dan y minimum 0 dan maksimum sebesar resolusi yang digunakan. Karakteristik dari grafik 2D antara lain:
2. Memiliki tampilan gambar yang datar (flat).
3. Perspektif.
4. Hanya memiliki warna dasar.
5. Berupa struk garis.
6. Menggunakan outline.
7. Memiliki frame layar yang terbatas.
8. Tidak menggunakan lighting.
9. Biasanya menggunakan shading.
10. Objek berada pada koordinat XY.
11. Grafik komputer 3 dimensi biasa disebut 3D atau ruang adalah bentuk dari benda yang memiliki panjang, lebar, dan tinggi. Grafik 3 Dimensi merupakan teknik penggambaran yg berpatokan pada titik koordinat sumbu x(datar), sumbu y(tegak), dan sumbu z(miring).Representasi dari data geometrik 3 dimensi sebagai hasil dari pemrosesan dan pemberian efek cahaya terhadap grafika komputer 2D. Tiga Dimensi, biasanya digunakan dalam penanganan grafis. 3D secara umum merujuk pada kemampuan dari sebuah video card (link). Saat ini video card menggunakan variasi dari instruksi-instruksi yang ditanamkan dalam video card itu sendiri (bukan berasal dari software) untuk mencapai hasil grafik yang lebih realistis dalam memainkan game komputer. Karakteristik dari grafik 3D, antara lain:
12. Memiliki variasi warna yang lebih banyak.
13. Memiliki layar tampilan yang lebih luas.
14. Objek berada dalam bidang XYZ.
15. Dapat menggunakan teknik lighting.
16. Dapat menimbulkan gambar yang timbul (tidak flat).
17. Memiliki tampilan yang lebih dramatis

**BAB III**

**PEMBAHASAN**

1. **Komponen Pembangun Objek**

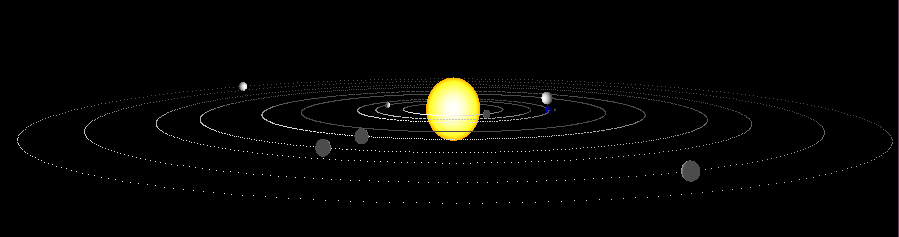
Komponen pada pembangunan simulasi perputaran planet terhadap matahari ini kebanyakan menggunakan *gl\_points* berupa banyak titik-tik yang saling berhubungan untuk menggambar orbit dan *glutSolidSphere* yang merupakan pembuatan gambar untuk kepadatan suatu planet yang berupa bola.

1. **Fungsi Dasar Pembentuk Objek**

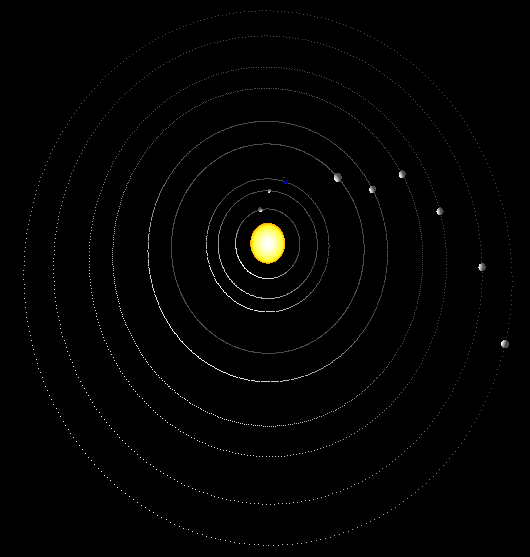
|  |  |
| --- | --- |
| Void | Keterangan |
| void Rotate(void)  {  hari += 2;  if(hari > 360)  hari = hari - 360;  tahun += .1;  if(tahun > 360)  tahun = tahun - 360;  glutPostRedisplay();  } | Rotasi pergerakan untuk planet dan bulan. Hari menunjukkan perputaran bulan dan tahun menunjukkan perputaran planet |
| void garisOrbit(float rad, float inc)  {  glPushMatrix();  float y = 0.0;  glColor3f(1.0, 1.0, 1.0);  glBegin(GL\_POINTS);  for(float sudut = 0; sudut <=  360; sudut+=inc)  {  float x =  rad\*sin((sudut)\*phi/180);  float z =  rad\*cos((sudut)\*phi/180);  glVertex3f(x, y, z);  }  glEnd();  glPopMatrix();  } | Garis orbit untuk menunjukkan perputaran planet yang beraturan dan tetap pada tempatnya |
| void merkurius(void)  {  glPushMatrix();  garisOrbit(5.5, 1);    glRotatef(tahun\*9, 0.0, 1.0, 0.0);  glTranslatef(5.5, 0.0, 0.0);    glutSolidSphere(0.4, 10, 8);  glPopMatrix();  } | Penggambaran planet dan rotasinya. Untuk pembuatan planet hampir sama fungsi yang digunakannya, yang membedakannya hanya nilai dari setiap komponen. |
| void fungsiKeyboard(void)  {  if(GetAsyncKeyState(VK\_LEFT))  {  jarakX += (cos((rot+90)\*phi/180))/2;  jarakZ += -(sin((rot+90)\*phi/180))/2;  }  if(GetAsyncKeyState(VK\_RIGHT))  {  jarakX += (cos((rot-90)\*phi/180))/2;  jarakZ += -(sin((rot-90)\*phi/180))/2;  }  if(GetAsyncKeyState(VK\_UP))  {  jarakX += (cos(rot\*phi/180)\* cos(pitch\*phi/180))/2;  jarakZ += -(sin(rot\*phi/180) \* cos(pitch\*phi/180))/2;  jarakY += sin(pitch\*phi/180);  }  if(GetAsyncKeyState(VK\_DOWN))  {  jarakX -= (cos(rot\*phi/180)\* cos(pitch\*phi/180))/2;  jarakZ -= -(sin(rot\*phi/180) \* cos(pitch\*phi/180))/2;  jarakY -= sin(pitch\*phi/180);  }  xlook = jarakX + (cos(rot\*phi/180) \* cos(pitch\*phi/180));  zlook = jarakZ - (sin(rot\*phi/180) \* cos(pitch\*phi/180));  ylook = jarakY + sin(pitch\*phi/180);  xup = cos(rot\*phi/180) \* cos((pitch+90)\*phi/180);  zup = -sin(rot\*phi/180) \* cos((pitch+90)\*phi/180);  yup = sin((pitch+90)\*phi/180);  glutPostRedisplay();  } | Void ini untuk interaksi yang dilakukan oleh keyboard yaitu   1. Dengan menekan tombol *Up* maka gambar akan menjauhi layar 2. Dengan menekan tombol *Down* maka gambar akan mendekati layar 3. Dengan menekan tombol *Right* maka gambar akan bergerak ke arah kiri 4. Dengan menekan tombol *Left* maka gambar akan bergerak ke arah kanan |
| void PassiveMouse(int x, int y)  {  if( (lastx - x) >50 || (lastx - x) <-50 ||  (lasty - y) >50 || (lasty - y) <-50 )  {  lastx = x;  lasty = y;  }  rot += ((lastx - x)/2);  lastx = x;  if(rot>360)  rot-=360;  if(rot<0)  rot+=360;  pitch += ((lasty - y))/2;  lasty = y;  if(pitch > 90)  pitch = 90.0;  if(pitch<-90)  pitch = -90.0;  glutPostRedisplay();  } | Void ini untuk interaksi yang digunakan oleh *mouse*. Pergerakannya sesuai dengan ke arah mana *mouse* bergerak |

1. **Tampilan Objek**

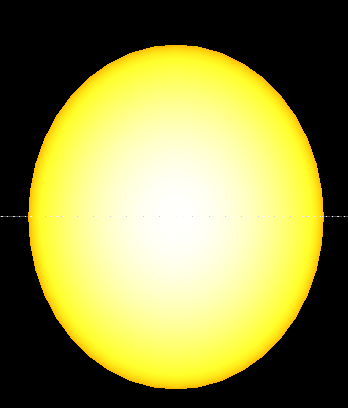
Ini adalah tampilan awal perputaran planet terhadap matahari



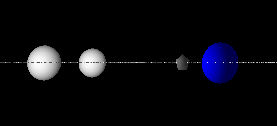
Gambar dibawah ini tampak dari atas



Gambar dibawah ini adalah gambar matahari yang diberi cahaya agar terlihat seperti matahari



Gambar dibawah ini adalah gambar beberapa planet yang mengelilingi matahari



**BAB IV**

**KESIMPULAN DAN SARAN**

1. **Kesimpulan**

Dengan menggunakan opengl ini dalam pembuatan perputaran planet terhadap matahari dapat disimpulkan:

1. Merepresentasikan gambar 3D perputaran planet terhadap matahari
2. Dengan menggunakan opengl ini akan terlihat jelas seperti apa perputaran planet terhadap matahari yang perputarannya dilakukan secara beraturan dengan menggunakan orbit
3. **Saran**

Dalam perancangan simulasi ini masih terdapat banyak kekurangan, diantaranya yaitu:

1. Background pada objek ini belum ada seperti bintang-bintang yang bertebaran seperti pada galaksi bimasakti
2. Belum menggunakan teksturing sehingga pewarnaannya masih kurang jelas

**SOURCE CODE**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Qori Nurqolbi - 10108424 \*/

/\* Idham Herawan Putra - 10108450 \*/

/\* Nur Mufmin - 10108447 \*/

/\* Vian Arwanda - 10108467 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <cstdlib>

#include <iostream>

#include <Windows.h>

#include <stdio.h>

#include <gl\GL.h>

#include <gl\GLU.h>

#include <gl\OpenGL\GL\glut.h>

#define checkImageWidth 164

#define checkImageHeight 164

#define phi 3.1415926535

using namespace std;

int w=900, h=600, z=0;

float tahun = 0, hari = 0;

float pitch = 0, rot = 90.0;

float jarakX = 0, jarakY = 0, jarakZ = 10.0;

float xlook = 0.0, ylook = 0.0, zlook = 0.0;

float xup = 0.0, yup = 0.0, zup= 0.0;

int lastx = 0, lasty = 0;

// Lighting values

static float ambientLight0[] = { 0.5f, 0.5f, 0.5f, 1.0f };

static float ambientLight1[] = { 0.1f, 0.1f, 0.1f, 1.0f };

static float diffuseLight[] = { 0.7f, 0.7f, 0.7f, 1.0f };

static float specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };

static float specref[] = { 1.0f, 1.0f, 1.0f, 1.0f };

static float lightPos[] = { 0.0f, 0.0f, 0.0f, 1.0f };

static float emissionSun[] = { 3.0f, 0.0f, 0.0f, 0.0f};

static float nullv[] = { 0.0f, 0.0f, 0.0f, 1.0f};

void fungsiKeyboard();

void merkurius(void);

void venus(void);

void bumi(void);

void mars(void);

void jupiter(void);

void saturnus(void);

void uranus(void);

void neptunus(void);

void pluto(void);

void init(void)

{

glClearColor(0.0,0.0,0.0,0.0);

glDepthFunc(GL\_LEQUAL);

glHint(GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_NICEST);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0,(GLdouble) w/(GLdouble) h, 1.0,300.0);

glMatrixMode(GL\_MODELVIEW);

glShadeModel(GL\_SMOOTH); //mengaktifkan shade model

glHint(GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_NICEST);

}

void enable(void)

{

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_LIGHTING); // mengaktifkan fungsi pencahayaan

glEnable(GL\_LIGHT0); // mengaktifkan sumber cahaya

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_NORMALIZE);

glDisable(GL\_CULL\_FACE);

}

void Rotate(void)

{

hari += 2;

if(hari > 360)

hari = hari - 360;

tahun += .1;

if(tahun > 360)

tahun = tahun - 360;

glutPostRedisplay();

}

void garisOrbit(float rad, float inc)

{

glPushMatrix();

float y = 0.0;

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_POINTS);

for(float sudut = 0; sudut <= 360; sudut+=inc)

{

float x = rad\*sin((sudut)\*phi/180);

float z = rad\*cos((sudut)\*phi/180);

glVertex3f(x, y, z);

}

glEnd();

glPopMatrix();

}

void merkurius(void)

{

glPushMatrix();

garisOrbit(5.5, 1);

glRotatef(tahun\*9, 0.0, 1.0, 0.0); // rotasi planet

glTranslatef(5.5, 0.0, 0.0);

glutSolidSphere(0.4, 10, 8); //gambar planet merkurius

glPopMatrix();

}

void venus(void)

{

glPushMatrix();

garisOrbit(8.5,1);

glRotatef(tahun\*8, 0.0, 1.0, 0.0);

glTranslatef(8.5, 0.0, 0.0);

glutSolidSphere(0.3, 20, 16);

glPopMatrix();

}

void bumi(void)

{

garisOrbit(10.5,1);

glRotatef(tahun\*7, 0.0, 1.0, 0.0);

glTranslatef(10.5, 0.0, 0.0);

glColor3f(0.0, 0.0, 1.0);

glutSolidSphere(0.37, 20, 16);

glRotatef(hari\*6, 0.0, 1.0, 0.0); //rotasi bulan

glTranslatef(0.80, 0.0, 0.0);

glColor3f(0.5, 0.5, 0.5);

glutSolidSphere(0.15, 5, 4); //gambar bulan

}

void mars(void)

{

glPushMatrix();

garisOrbit(16.5,0.3);

glRotatef(tahun\*5, 0.0, 1.0, 0.0);

glTranslatef(16.5, 0.0, 0.0);

glutSolidSphere(0.7, 20, 16);

glPopMatrix();

}

void jupiter(void)

{

glPushMatrix();

garisOrbit(20.5, 0.5);

glRotatef(tahun\*4, 0.0, 1.0, 0.0);

glTranslatef(20.5, 0.0, 0.0);

glutSolidSphere(0.6, 20, 16);

glPopMatrix();

}

void saturnus(void)

{

glPushMatrix();

garisOrbit(26.5, 0.7);

glRotatef(tahun\*4, 0.0, 1.0, 0.0);

glTranslatef(26.5, 0.0, 0.0);

glutSolidSphere(0.6, 20, 16);

glPopMatrix();

}

void uranus(void)

{

glPushMatrix();

garisOrbit(30.5, 0.9);

glRotatef(tahun\*3, 0.0, 1.0, 0.0);

glTranslatef(30.5, 0.0, 0.0);

glutSolidSphere(0.6, 20, 16);

glPopMatrix();

}

void neptunus(void)

{

glPushMatrix();

garisOrbit(36.5, 0.95);

glRotatef(tahun\*2, 0.0, 1.0, 0.0);

glTranslatef(36.5, 0.0, 0.0);

glutSolidSphere(0.6, 20, 16);

glPopMatrix();

}

void pluto(void)

{

glPushMatrix();

garisOrbit(41.5,0.990);

glRotatef(tahun, 0.0, 1.0, 0.0);

glTranslatef(41.5, 0.0, 0.0);

glutSolidSphere(0.6, 20, 16);

glPopMatrix();

}

void displayObjek(void)

{

init();

enable();

Rotate();

fungsiKeyboard();

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

//set the viewing transformation

gluLookAt(jarakX, jarakY, jarakZ, xlook, ylook, zlook, xup, yup, zup);

glPushMatrix(); //save state

glTranslatef(0.0, 0.0, -5.0); //translate out to sun

//set specular reflectivity with low shine

glColor4f(1.0, 0.8, 0.0, 1.0);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specref);

glMateriali(GL\_FRONT, GL\_SHININESS, 3);

glMaterialfv(GL\_FRONT, GL\_EMISSION, emissionSun); //make sun glow

glLightfv(GL\_LIGHT1, GL\_POSITION, lightPos); //move light 1 under sun

glutSolidSphere(3, 40, 32); //matahari

//save lighting

glPushAttrib(GL\_LIGHTING\_BIT);

glDisable(GL\_LIGHT0);

glEnable(GL\_LIGHT1); //turn on the sun

glMaterialfv(GL\_FRONT, GL\_SPECULAR, nullv);

glMaterialfv(GL\_FRONT, GL\_EMISSION, nullv);

merkurius();

venus();

glPushMatrix();

bumi();

glPopMatrix();

mars();

jupiter();

saturnus();

uranus();

neptunus();

pluto();

glPopAttrib();

glPopMatrix(); //restore state

glutSwapBuffers();

}

void SetupLighting(void)

{

// Setup and enable light0

glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambientLight0);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, diffuseLight);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, specular);

// Setup light1

glLightfv(GL\_LIGHT1, GL\_AMBIENT, ambientLight1);

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, diffuseLight);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, specular);

// Set Material properties to follow glColor values

glColorMaterial(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE);

}

void SetupRC(void)

{

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_CULL\_FACE);

SetupLighting();

glClearColor(0.0, 0.0, 0.0, 0.0);

glShadeModel(GL\_SMOOTH);

}

void Reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(40.0, (GLfloat)w/(GLfloat)h, 1.0, 400.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void fungsiKeyboard(void)

{

if(GetAsyncKeyState(VK\_LEFT))

{

jarakX += (cos((rot+90)\*phi/180))/2;

jarakZ += -(sin((rot+90)\*phi/180))/2;

}

if(GetAsyncKeyState(VK\_RIGHT))

{

jarakX += (cos((rot-90)\*phi/180))/2;

jarakZ += -(sin((rot-90)\*phi/180))/2;

}

if(GetAsyncKeyState(VK\_UP))

{

jarakX += (cos(rot\*phi/180)\* cos(pitch\*phi/180))/2;

jarakZ += -(sin(rot\*phi/180) \* cos(pitch\*phi/180))/2;

jarakY += sin(pitch\*phi/180);

}

if(GetAsyncKeyState(VK\_DOWN))

{

jarakX -= (cos(rot\*phi/180)\* cos(pitch\*phi/180))/2;

jarakZ -= -(sin(rot\*phi/180) \* cos(pitch\*phi/180))/2;

jarakY -= sin(pitch\*phi/180);

}

xlook = jarakX + (cos(rot\*phi/180) \* cos(pitch\*phi/180));

zlook = jarakZ - (sin(rot\*phi/180) \* cos(pitch\*phi/180));

ylook = jarakY + sin(pitch\*phi/180);

xup = cos(rot\*phi/180) \* cos((pitch+90)\*phi/180);

zup = -sin(rot\*phi/180) \* cos((pitch+90)\*phi/180);

yup = sin((pitch+90)\*phi/180);

glutPostRedisplay();

}

void PassiveMouse(int x, int y)

{

if( (lastx - x) >50 || (lastx - x) <-50 ||

(lasty - y) >50 || (lasty - y) <-50 )

{

lastx = x;

lasty = y;

}

rot += ((lastx - x)/2);

lastx = x;

if(rot>360)

rot-=360;

if(rot<0)

rot+=360;

pitch += ((lasty - y))/2;

lasty = y;

if(pitch > 90)

pitch = 90.0;

if(pitch<-90)

pitch = -90.0;

glutPostRedisplay();

}

int main(void)

{

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(900, 700);

glutInitWindowPosition(100, 100);

glutCreateWindow("Tata surya");

SetupRC();

glutDisplayFunc(displayObjek);

glutReshapeFunc(Reshape);

glutPassiveMotionFunc(PassiveMouse);

glutMainLoop();

}