ANIMASI PROSES METAMORFOSIS DENGAN OPENGL

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika

Nama : Noppa

NIM : 07523071

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2014

ii

LEMBAR PENGESAHAN PEMBIMBING

ANIMASI PROSES METAMORFOSIS DENGAN OPENGL

TUGAS AKHIR

Oleh :

Nama : Noppa

NIM : 07523071

Yogyakarta, Juni 2014

Pembimbing

Izzati Muhimmah, ST, MSc., PhD.

iii

LEMBAR PENGESAHAN PENGUJI

ANIMASI PROSES METAMORFOSIS DENGAN OPENGL

Oleh :

Nama : Noppa
NIM : 07523071

Telah dipertahankan di Depan Sidang Penguji Sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 10 Juli 2014

Tim Penguji,

Izzati Muhimmah, ST, MSc., PhD.
Ketua

Affan Mahtarami, S.Kom., M.T.
Anggota I

Arwan Ahmad Khoiruddin, S.Kom., M.Cs
Anggota II

Mengetahui
Ketua Jurusan Teknik Informatika
Universitas Islam Indonesia

Yudi Prayudi, S.Si., M.Kom

LEMBAR PERNYATAAN KEASLIAN

HASIL TUGAS AKHIR

Saya yang bertanda tangan dibawah ini,

Nama : Noppa

NIM : 07523071

Menyatakan bahwa seluruh komponen dan isi dalam Laporan Tugas Akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti bahwa ada beberapa bagian dari karya ini adalah bukan hasil karya saya sendiri, maka saya siap menanggung resiko dan konsekuensi apapun.

Demikian pernyataan ini saya buat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, Juni 2014

Noppa

HALAMAN PERSEMBAHAN

Alhamdulillahirobil'alamiin,

Segala puji syukur kita panjatkan atas nikmat dan karunia yang telah ALLAH berikan. Terima kasih atas segala kemudahan yang Engkau berikan sehingga hamba dapat menyelesaikan skripsi ini.

Ucapan terima kasih untuk :

Dosen-dosen Teknik Informatika FTI UII, terima kasih atas semua ilmu yang sudah diajarkan kepada saya.

Ibu Izzy atas perhatian dan waktu yang selalu sabar dalam membimbing saya dalam mengerjakan dari awal hingga selesainyaskripsi ini dan memberikan saya ilmu yang berguna, termasuk membantu mendapatkan buku yang tidak beredar di Indonesia.

Bapak dan ibu serta keluarga yang telah sabar dan tetap mendukung saya dalam semangat dan materi dan kepercayaannya untuk menyelesaikan kuliah walaupun harus memakan waktu yang cukup lama.

Ibu Marsih atas perhatian dan dukungan untuk saya disaat sedang down dan kepercayaannya untuk menyelesaikan skripsi.

Teman-teman INCLUDE dan informatika, terimakasih atas perjuangan dan ilmu yang telah dibagi secara gratis.

MOTTO

“And why do we fall, Bruce? So we can learn to pick ourselves up.” - Batman Begins

"Knowledge cannot replace friendship. I'd rather be stupid than to lose you." - Patrick Star

“Nobody has a perfect life. Everybody has their own problems. Some people just know how to deal with it in a perfect way.”

If “Plan A” didn’t work. The alphabet has 25 more letters! Trust ALLAH

KATA PENGANTAR

Assalamualaikum Warrahmatullahi Wabarakatuh.

Puji syukur kehadirat ALLAH SWT yang telah melimpahkan rahmat dan hidayah-Nya sehingga saya dapat menyelesaikan tugas akhir ini. Shalawat dan salam semoga selalu tercurah kepada junjungan kita Nabi Muhammad SAW, rahmat lil alamiin.

Tugas akhir yang berjudul "Animasi Proses Metamorfosis dengan OpenGL" ini disusun sebagai salah satu syarat untuk mendapatkan gelar sarjana S1 pada jurusan Teknik Informatka Fakultas Teknik Industri Universitas Islam Indonesia.

Ucapan terima kasih saya persembahkan kepada berbagai pihak yang telah membantu menyelesaikan tugas akhir ini kepada :

1. ALLAH SWT yang telah memberikan rahmat dan hidayah-Nya
2. Bapak Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Yudi Prayudi,S.Si.,M,Kom selaku Ketua Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Ibu Izzati Muhimmah, ST, MSc., PhD. selaku Dosen Pembimbing. Terima kasih atas perhatian dan waktu yang selalu sabar dalam membimbing saya dan dalam mengerjakan tugas akhir dari awal hingga selesainyaskripsi ini dan memberikan saya ilmu yang berguna.
5. Ibu Marsih atas perhatian dan dukungan untuk menyelesaikan skripsi.
6. Bapak Adi Sugito dan Ibu Satiyem serta Juli Ratna Ningsih, orang tua dan saudara yang tetap sabar dan memberi semangat serta dukungan setiap hari.
7. Yogie Aditya, Septian Aditya, Wahyu Prasetyo, Sistha, Mentari, Marista, Tiardi Marjuki yang telah member semangat dan mengingatkan untuk menyelesaikan tugas akhir ini.

8. Teman-teman yang tidak mungkin saya sebutkan satu persatu disini, terima kasih banyak atas dukungan dan ilmunya selama ini.

Dalam pelaksanaan dan pembuatan program serta laporan tugas akhir ini saya menyadari bahwa masih banyak kekurangan-kekurangan baik yang disadari maupun tidak disadari, oleh karena itu saya mengharapkan saran dan kritik dari para pembaca.

Semoga laporan tugas akhir ini member manfaat bagi para pembaca maupun bagi kepustakaan ilmu baik pada Jurusan Teknik Informatika Universitas Islam Indonesia maupun bagi yang membutuhkan ilmu pengetahuan Teknologi Informasi di Indonesia.

Wassalamualaikum Warrahmatullahi Wabarakatuh.

Yogyakarta, Juni 2014

Noppa

SARI

Animasi adalah salah satu cara dalam memberikan sebuah informasi kepada orang lain. Karena itu dibuatlah animasi proses metamorfosis kupu-kupu untuk mempermudah kita mempelajari. Banyaknya perangkat lunak untuk membuat animasi pun semakin beragam, salah satunya adalah OpenGL. Tujuan animasi adalah untuk mengaplikasikan rumus bangun datar, bangun ruang dan kurva bezier pada matematika untuk memodelkan dan teksturing objek 3dimensi dan menganimasikannya dengan bantuan bahasa pemrograman C++ dan OpenGL. Dengan demikian maka kita selain dapat memanfaatkan rumus matematika dan bahasa pemrograman yang sudah kita pelajari, kita juga dapat membuat objek 3dimensi yang kita inginkan. Setelah animasi ini dibuat dan diuji, hasil yang didapatkan bahwa animasi ini lebih ringan dan cepat dalam proses rendering objek 3dimensi daripada perangkat lunak yang sudah tersedia.

Kata Kunci : animasi, matematika, modeling, OpenGL, teksturing

TAKARIR

Bitmapped (Raster)	Representasi citra grafis yang terdiri susunan titik yang tersimpan di memori komputer.
Callback	Fungsi yang dipanggil setelah suatu fungsi dijalankan.
Clamping	Fungsi untuk menahan posisi tekstur agar tidak berubah posisi.
Frame	Bingkai untuk menampilkan gambar pada animasi.
Filtering	Kumpulan perintah untuk menyaring data.
Input	Masukan.
Keyboard	Papan ketik untuk memasukkan data dan perintah pada komputer.
Library	Sebuah class yang memiliki tugas spesifik yang fungsinya bisa digunakan pada sebuah pemrograman.
Looping	Proses yang dilakukan secara berulang-ulang sampai batas yang ditentukan.
Mouse	Alat yang digunakan untuk memasukkan data dan perintah pada komputer.
Pseudocode	Kode pemrograman yang ditulis dengan bahasa manusia.
Redraw	Proses untuk menggambar ulang.
Repeating	Fungsi untuk mengulang.
Stroke (Vector)	Gambar yang dibuat dari titik-titik berbasis persamaan matematika.

DAFTAR ISI

HALAMAN JUDUL i
LEMBAR PENGESAHAN PEMBIMBING ii
LEMBAR PENGESAHAN PENGUJI iii
LEMBAR PERNYATAAN KEASLIAN HASIL TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
MOTTO vi
KATA PENGANTAR vii
SARI ix
TAKARIR x
DAFTAR ISI xi
DAFTAR GAMBAR xv
DAFTAR TABEL xvii
DAFTAR LAMPIRAN xviii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan Penelitian 2
1.5 Manfaat Penelitian 2
1.6 Metodologi penelitian 2
1.7 Sistematika Penulisan 3
BAB II LANDASAN TEORI 4
2.1 Sistem Koordinat 4
2.2 Konsep Bangun Datar dan Bangun Ruang 5
2.2.1 Trigonometri Segitiga 5
2.2.2 Tabung 5
2.2.3 Kerucut 6
2.2.4 Bola 6
2.3 Kurva Bezier 7
2.3.1 Kurva Bezier Linier 7
2.3.2 Kurva Bezier Kuadrat 7

2.3.3 Kurva Bezier Kubik 8
2.4 Transformasi Geometri 9
2.4.1 Translasi (Pergeseran) 9
2.4.2 Rotasi (Perputaran) 10
2.4.3 Dilatasi (Perkalian) 11
2.4.4 Refleksi (Pencerminan) 12
2.5 Teori Anatomi Kupu 13
2.5.1 Kepala (Head) 14
2.5.2 Antena (Antennae) 14
2.5.3 Mata Kompon (Compund Eye) 14
2.5.4 Probosis (Proboscis) 14
2.5.5 Palp Labial (Labial Palps) 14
2.5.6 Dada (Thorax) 14
2.5.7 Sayap Depan (Forewing) 15
2.5.8 Sayap Belakang (Hindwing) 15
2.5.9 Kaki (Legs) 15
2.5.10 Perut (Abdomen) 15
2.6 OpenGL 15
2.6.1 GLUT (GL Utility Toolkit) 16
2.6.2 Teksturing 18
2.6.3 Animasi 20
BAB III ANALISIS KEBUTUHAN & PERANCANGAN PERANGKAT LUNAK 21
3.1 Analisis Kebutuhan Sistem 21
3.1.1 Metode Analisis 21
3.1.2 Hasil Analisis 21
a. Analisis Kebutuhan Proses 21
b. Analisis Keluaran Sistem 21
3.2 Perancangan Perangkat Lunak 21
3.2.1 Metode Perancangan 21
3.2.2 Hasil Perancangan 22
1) Tabung 23
2) Kerucut 25
3) Ulat 26

4) Teksturing Tabung dan Kerucut 27
a. Tutup / alas Tabung dan Kerucut 27
b. Selimut Tabung 28
c. Selimut kerucut 28
5) Daun 29
6) Teksturing Daun 32
7) Bola 32
8) Sayap 33
9) Antena 33
10) Huruf 34
11) Papan Alur Cerita (*Storyboard*) 34
12) Animasi 35
a. Gerakan Sayap 35
b. Gerakan Geser 36
3.3 Analisis Pengujian Perangkat Lunak 36
3.3.1 Pengujian Objek 36
3.3.2 Pengujian Animasi 36
BAB IV HASIL DAN PEMBAHASAN 37
4.1 Perangkat Keras dan Lunak yang Digunakan 37
4.1.1 Perangkat Keras 37
4.1.2 Perangkat Lunak 37
4.2 Hasil 37
4.2.1 Pohon 37
4.2.2 Teksturing Tabung 38
4.2.3 Daun 39
4.2.4 Teksturing Daun 40
4.2.5 Ulat 41
4.2.6 Teksturing Ulat 42
4.2.7 Kepompong 43
4.2.8 Badan Kupu-kupu 44
4.2.9 Sayap Kupu-kupu 48
4.2.10 Antena 49
4.2.11 Teksturing Bezier 49
4.2.12 Duri 49

4.2.13 Bola 50
4.2.14 Penggabungan Objek Pohon 50
4.2.15 Mata Kupu-kupu 51
4.2.16 Penggabungan Objek Ulat 51
4.2.17 Penggabungan Objek Kupu-kupu 54
4.2.18 Animasi Sayap 54
4.2.19 Animasi Metamorfosis 56
4.2.20 Menampilkan huruf 60
4.3 Pembahasan 60
4.3.1 Perbandingan Objek Pohon 60
4.3.2 Perbandingan Objek Daun 61
4.3.3 Perbandingan Objek Ulat 61
4.3.4 Perbandingan Objek Kepompong 61
4.3.5 Perbandingan Badan Kupu-kupu 62
4.3.6 Perbandingan Sayap 62
4.3.7 Tampilan Hasil Penggabungan Objek 62
a. Pohon dan Ulat 63
b. Pohon dan Kepompong 63
c. Kupu-kupu 64
4.3.8 Tabel Perbandingan Rancangan Objek 64
4.4 Kelebihan Aplikasi 64
4.5 Kekurangan Aplikasi 65
BAB V KESIMPULAN DAN SARAN 66
5.1 Kesimpulan 66
5.2 Saran 66
DAFTAR PUSTAKA 67
LAMPIRAN

DAFTAR GAMBAR

Gambar 2. 1 Koordinat Kartesius 2 Dimensi 4
Gambar 2. 2 Koordinat Kartesius 3 Dimensi 5
Gambar 2. 3 Trigonometri 5
Gambar 2. 4 Tabung 6
Gambar 2. 5 Kerucut 6
Gambar 2. 6 Bola 7
Gambar 2. 7 Kurva Bezier Linier 7
Gambar 2. 8 Kurva Bezier Kuadrat 8
Gambar 2. 9 Kurva Bezier Kubik 9
Gambar 2. 10 Kurva Bezier Kubik dengan Titik Poin 5 9
Gambar 2. 11 Translasi (Pergeseran) 10
Gambar 2. 12 Rotasi (Perputaran) 11
Gambar 2. 13 Dilatasi (Perkalian) 11
Gambar 2. 14 Refleksi (Pencerminan) 13
Gambar 2. 15 Teori Anatomi Kupu-kupu 13
Gambar 3. 1 Kupu-kupu Sayap Burung 22
Gambar 3. 2 SketsaPohon 22
Gambar 3. 3 Sketsa Ulat 22
Gambar 3. 4 Sketsa Kepompong 23
Gambar 3. 5 Sketsa Badan Kupu-kupu 23
Gambar 3. 6 Trigonometri Segitiga pada Diagram Kartesius 24
Gambar 3. 7 Lingkaran pada Diagram Kartesius 24
Gambar 3. 8 Tabung 12 sisi 25
Gambar 3. 9 Kerucut 26
Gambar 3. 10 Polygon Tabung untuk Ulat 26
Gambar 3. 11 Tutup/alas Tabung dan kerucut 27
Gambar 3. 12 Selimut Tabung 28
Gambar 3. 13 Selimut kerucut 28
Gambar 3. 14 Garis Bengkok 29
Gambar 3. 15 Sketsa Daun 30
Gambar 3. 16 Objek Daun 3 Dimensi 31
Gambar 3. 17 Tekstur Daun 32

Gambar 3. 18 Sketsa Sayap Kupu-kupu 33
Gambar 3. 19 Papan Alur Cerita.....34
Gambar 4. 1 Perbandingan Objek Pohon.....600
Gambar 4. 2 Perbandingan Objek Daun 611
Gambar 4. 3 Perbandingan Objek Ulat 611
Gambar 4. 4 Perbandingan Objek Kepompong 611
Gambar 4. 5 Perbandingan Objek Kupu-kupu.....622
Gambar 4. 6 Perbandingan Sayap 622
Gambar 4. 7 Penggabungan Pohon dan Ulat 633
Gambar 4. 8 Penggabungan Pohon dan Kepompong 633
Gambar 4. 9 Penggabungan Objek Kupu-kupu 644

DAFTAR TABEL

Tabel 3. 1 Perhitungan Nilai Vertex Tabung 25
Tabel 3. 2 Perhitungan Nilai Vertex Kerucut 26
Tabel 3. 3 Perhitungan Nilai Vertex Garis Bengkok 29
Tabel 3. 4 Perhitungan Nilai Vertex Daun.....31
Tabel 3. 5 Perhitungan Nilai Vertex Tekstur Daun 32
Tabel 3. 6 Titik Awal dan Akhir Gerakan Sayap.....36

DAFTAR LAMPIRAN

No Lampiran

1 Tabel L-1 Array Pohon L-1
2 Syntak Pemrograman Fungsi Pohon L-2
3 Tabel L-2 Array Teksturing L-3
4 Syntak Pemrograman Fungsi Teksturing L-4
5 Syntak Pemrograman Fungsi Daun L-5
6 Syntak Pemrograman Fungsi Teksturing Daun L-6
7 Tabel L-3 Array Ulat L-7
8 Syntak Pemrograman Fungsi Ulat L-8
9 Tabel L-4 Teksturing Ulat L-10
10 Syntak Pemrograman Fungsi Teksturing Ulat L-11
11 Tabel L-5 Array Kepompong L-12
12 Syntak Pemrograman Fungsi Kepompong L-14
13 Tabel L-6 Array Kupu-kupu L-16
14 Syntak Pemrograman Fungsi Kupu-kupu L-20
15 Syntak Pemrograman Bezier L-24
16 Syntak Pemrograman Duri L-26
17 Syntak Pemrograman Bola L-27
18 Syntak Pemrograman Penggabungan Objek Pohon L-28
19 Syntak Pemrograman Penggabungan Objek Ulat L-29
20 Syntak Pemrograman Penggabungan Objek Kupu-kupu L-32
21 Syntak Pemrograman Animasi Kupu-kupu L-33
22 Syntak Pemrograman Animasi Proses Metamorfosis L-36

BAB I
PENDAHULUAN

1.1 Latar Belakang

Penggunaan gambar untuk komunikasi antara sesama manusia sudah digunakan sejak dulu. Terbukti dengan adanya peninggalan-peninggalan jaman dahulu yang tertera pada dinding batu dan goa. Pada jaman modern setelah ditemukannya komputer grafik maka perkembangan gambar untuk media komunikasi sudah mulai beranjak ke media animasi atau gambar bergerak.

Metamorfosis adalah suatu proses perkembangan biologi pada hewan yang melibatkan perubahan penampilan fisik (akibat pertumbuhan dan differensiasi sel yang secara radikal berbeda) dan/atau struktur setelah kelahiran atau penetasan. Pada serangga terdapat dua jenis metamorfosis, yaitu hemimetabola (metamorfosis tidak sempurna) dan holometabola (metamorfosis sempurna).

Fase yang belum dewasa pada metamorfosis biasanya disebut larva/nimfa. Tapi pada metamorfosis kompleks pada kebanyakan spesies serangga, hanya fase pertama yang disebut larva/nimfa. Perkembangan nimfa berlangsung pada fase pertumbuhan berulang dan ekdisis (pergantian kulit), fase ini disebut instar.

Pada holometabola, larva sangat berbeda dengan dewasanya. Serangga yang melakukan holometabola melalui fase larva, kemudian memasuki fase tidak aktif yang disebut pupa, atau chrysalis, dan akhirnya menjadi dewasa (imago). Sementara di dalam pupa, serangga akan mengeluarkan cairan pencernaan, untuk menghancurkan tubuh larva, menyisakan sebagian sel saja. Sebagian sel itu kemudian akan tumbuh menjadi dewasa menggunakan nutrisi dari hancuran tubuh larva. Proses kematian sel disebut histolisis, dan pertumbuhan sel lagi disebut histogenesis.

Untuk membuat sebuah animasi ada banyak pilihan software yang sudah tersedia salah satunya OpenGL. OpenGL adalah suatu library yang bersifat open source, dan dapat digunakan pada berbagai jenis bahasa pemrograman salah satunya bahasa C++.

1.2 Rumusan Masalah

Bagaimana memodelkan objek ulat, pohon, kepompong dan kupu-kupu dengan menggunakan rumus matematika geometri dan kurva bezier. Kemudian objek tersebut digerakkan untuk dibuat animasinya, dengan durasi animasi selama kurang lebih 1 menit dan dibuat menggunakan bahasa pemrograman C++ dengan library OpenGL.

1.3 Batasan Masalah

1. Menampilkan sebuah animasi proses metamorfosis dari objek ulat yang bergerak di batang pohon hingga bertambah besar dan menjadi kepompong kemudian berubah menjadi kupu-kupu yang cantik.
2. Durasi animasi selama kurang lebih 1 menit.
3. Menggunakan software Microsoft Visual C++ dengan paket library OpenGL.

1.4 Tujuan Penelitian

Mengaplikasikan rumus matematika untuk membuat objek.

1.5 Manfaat Penelitian

1. Memudahkan memahami proses metamorfosis kupu-kupu.
2. Penerapan rumus matematika dalam pembuatan objek 3dimensi.

1.6 Metodologi penelitian

Metode penenilitan adalah suatu langkah yang digunakan untuk menyelesaikan masalah dalam penelitian tersebut.

1. Memilih salah satu dari banyaknya jenis kupu-kupu yang ada di Indonesia.
2. Menganalisa dan membuat rancangan / sketsa objek 3dimensi.
3. Menentukan rumus matematika yang akan digunakan untuk membuat objek 3dimensi.
4. Memodelkan sketsa menjadi objek 3dimensi menggunakan aplikasi dan bahasa pemrograman C++ dan OpenGL.
5. Pengujian dilakukan terhadap bentuk objek, teksturing dan proses animasi apakah sudah sesuai dengan rancangan.

1.7 Sistematika Penulisan

Dalam penyusunan laporan tugas akhir ini, sistematika dibagi menjadi beberapa bagian sebagai berikut :

BAB I PENDAHULUAN

Membahas tentang Latar Belakang, Rumusan Masalah, Batasan Masalah, Tujuan Penelitian, Manfaat Penelitian, Metodologi Penelitian dan Sistematika Penulisan.

BAB II LANDASAN TEORI

Membahas teori-teori yang mendukung dalam perancangan animasi proses metamorfosis dengan OpenGL. Teori tersebut meliputi sistem koordinat, bangun ruang, kurva bezier, transformasi geometri, anatomi kupu-kupu dan prinsip dasar pemrograman C++ dan OpenGL.

BAB III ANALISIS KEBUTUHAN & PERANCANGAN PERANGKAT LUNAK

Bab ini memuat tentang analisis proses dan analisis keluaran sistem. Tahapan perancangan sistem terdiri dari sketsa, rancangan objek, papan alur cerita, dan rancangan animasi.

BAB IV HASIL DAN PEMBAHASAN

Menjabarkan mengenai hasil rancangan dan proses pembuatan objek dan pengujian animasi.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan dari analisis kinerja pada bagian sebelumnya. Saran yang perlu diperhatikan kedepannya berdasarkan keterbatasan dan error yang ditemukan selama proses pembuatan animasi ini.

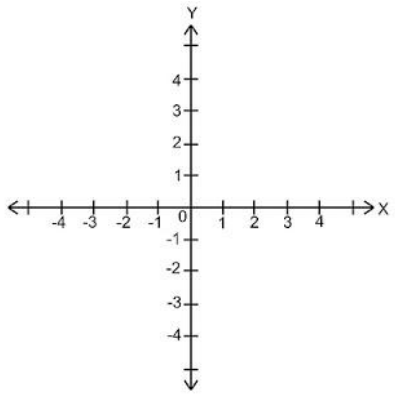
BAB II
LANDASAN TEORI

Pada bab ini akan diuraikan beberapa landasan teori yang terbagi dalam enam sub-bab yang membahas mengenai sistem koordinat, konsep bangun datar dan bangun ruang, kurva bezier, transformasi geometri, teori anatomi kupu dan OpenGL.

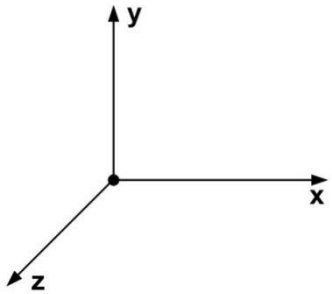
2.1 Sistem Koordinat

Digunakan untuk menggambarkan tiap titik dalam bangun datar dua dimensi dan bangun ruang tiga dimensi. Untuk bangun dua dimensi pada koordinat kartesius terdapat 2 garis sumbu yaitu sumbu X dan sumbu Y, sedang untuk bangun tiga dimensi terdapat 3 garis sumbu yaitu sumbu X, sumbu Y dan sumbu Z (Descartes, 1637).

Sumbu X digambarkan sebagai garis horisontal, sedang garis vertikal digambarkan dengan sumbu Y. Sumbu Z digunakan untuk menggambarkan kedalaman suatu bangun ruang tiga dimens. Perpotongan antara sumbu X, sumbu Y dan sumbu Z dititik 0 (nol) disebut pusat koordinat.



Gambar 2. 1 Koordinat Kartesius 2 Dimensi

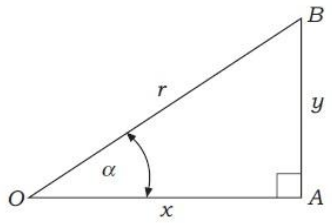


Gambar 2. 2 Koordinat Kartesius 3 Dimensi

2.2 Konsep Bangun Datar dan Bangun Ruang

2.2.1 Trigonometri Segitiga

$$sin\,\alpha^0 = \frac{y}{r} \qquad cos\,\alpha^0 = \frac{x}{r} \qquad tan\,\alpha^0 = \frac{y}{x} \tag{2.1}$$



Gambar 2. 3 Trigonometri

Dimana y adalah sisi segitiga didepan sudut α^0, r sisi terpanjang dan x adalah sisi segitiga yang lain (Lengyel, 2004).

2.2.2 Tabung

Tabung atau silinder adalah bangun ruang tiga dimensi yang dibentuk oleh dua buah lingkaran yang identik dan sejajar dan persegi panjang yang mengelilingi kedua lingkaran tersebut. Kedua lingkaran tersebut sebagai alas dan tutup tabung serta persegi panjang yang menyelimutinya disebut selimut tabung.

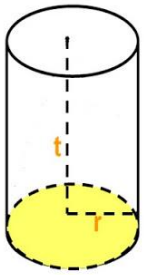
Rumus tabung :

Luas alas : $$L = \pi r^2 \tag{2.2}$$

Luas selimut : $$L = 2\pi r \cdot t \tag{2.3}$$

Luas permukaan : $\mathrm{L} = 2 \cdot \text{luas alas} + \text{luas selimut}$

$$L = 2 \cdot \pi r^2 + 2\pi r \cdot t \tag{2.4}$$



Gambar 2. 4 Tabung

2.2.3 Kerucut

Kerucut adalah sebuah limas yang beralas lingkaran. Sisi tegak kerucut tidak berupa segitiga tetapi berupa bidang lengkung yang disebut selimut kerucut.

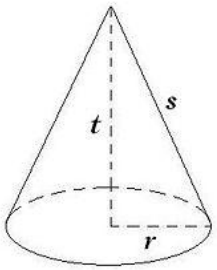
Rumus kerucut :

Luas alas : $L = \pi r^2$ (2.5)

Luas selimut : $L = \pi \cdot r \cdot s$ (2.6)

Luas permukaan : $\mathrm{L} = \text{luas alas} + \text{luas selimut}$

$$L = \pi r^2 + \pi \cdot r \cdot s \quad (2.7)$$



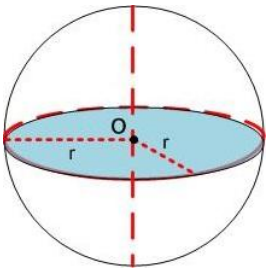
Gambar 2. 5 Kerucut

2.2.4 Bola

Bola adalah bangun ruang tiga dimensi yang dibentuk oleh lingkaran tak hingga, berjari-jari sama panjang dan berpusat pada satu titik yang sama.

Rumus bola

Luas Permukaan : $L = 4\pi r^2$ (2.8)



Gambar 2. 6 Bola

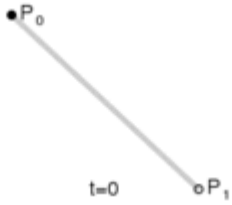
2.3 Kurva Bezier

Kurva Bezier digunakan untuk memodelkan kurva halus diantara dua atau lebih titik pembentuk kurva yang dapat diperbesar atau diperkecil sampai tak terbatas (Bezier, 1962). Didefinisikan oleh satu set titik kontrol P_0 sampai dengan P_n, dimana n disebut order (n = 1 untuk linier, 2 untuk kuadrat dll). Titik kontrol pertama dan terakhir selalu titik akhir kurva, dan titik kontrol menengah tidak terletak pada kurva.

2.3.1 Kurva Bezier Linier

Kurva Bezier linier adalah sebuah garis lurus yang dibentuk dari P_0 dan P_1. Persamaan didapat dari

$$\begin{aligned} B(t) &= P_0 + t(P_0 - P_1) \\ &= (1-t)P_0 + tP_1 \, , t \in [0,1] \end{aligned} \tag{2.9}$$



Gambar 2. 7 Kurva Bezier Linier

2.3.2 Kurva Bezier Kuadrat

Kurva Bezier kuadrat didapatkan dari titik P_0, P_1, dan P_2.

$$B(t) = (1-t)[(1-t)P_0 + tP_1] + t[(1-t)P_1 + tP_2 \, , t \in [0,1] \tag{2.10}$$

Dari persamaan (2.10) maka kurva Bezier kuadrat dibentuk dari dua kurva Bezier linier dari P_0 ke P_1 dan P_1 ke P_2. Sehingga didapat persamaan baru

$$B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2 \, , t \in [0,1] \tag{2.11}$$

Turunan dari kurva Bezier dari t adalah :

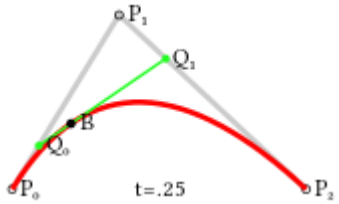
$$B'(t) = 2(1-t)(P_1 - P_0) + 2t(P_2 - P_1) \quad (2.12)$$

Sehingga garis singgung kurva di P_0 dan P_2 berpotongan di P_1. Nilai t meningkat dari 0 ke 1 sehingga membentuk kurva dari P_0 ke P_1, kemudian melengkung sampai P_2 dari P_1.

Turunan kedua dari kurva Bezier dari t adalah :

$$B''(t) = 2(P_2 - 2P_1 + P_0) \quad (2.13)$$

Kurva Bezier kuadrat disebut juga busur berbentuk kerucut.



Gambar 2. 8 Kurva Bezier Kuadrat

Titik Q_0 bervariasi dari P_0 ke P_1 dan menggambarkan kurva Bezier linier.
Titik Q_1 bervariasi dari P_1 ke P_2 dan menggambarkan kurva Bezier linier.
Titik B(t) bervariasi dari Q_0 ke Q_1 dan menggambarkan kurva Bezier kuadrat.

2.3.3 Kurva Bezier Kubik

Kurva Bezier kubik didefinisikan dengan minimal empat titik poin P_0, P_1, P_2, dan P_3. Kurva terbentuk dari P_0 ke P_3 tetapi tidak melewati titik P_1 dan P_2. Titik P_1 dan P_2 hanya digunakan untuk menentukan arah kurva dan berapa lama kurva bergerak ke titik selanjutnya.

Kurva Bezier kubik didefinisikan sebagai kombinasi linier dari dua kurva Bezier kuadrat :

$$B(t) = (1-t)B_{P_0,P_1,P_2}(t) + tB_{P_1,P_2,P_3}(t)\ , t \in [0,1] \quad (2.14)$$

Dapat dijabarkan menjadi :

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3, t \in [0,1] \quad (2.15)$$

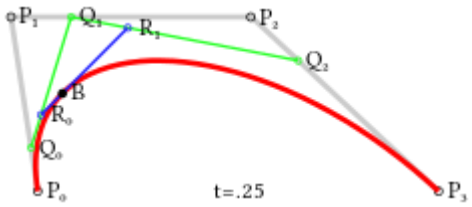
Turunan kurva Bezier kubik dari t adalah :

$$B'(t) = 3(1-t)^2(P_1 - P_0) + 6(1-t)t(P_2 - P_1) + 3t^2(P_3 - P_2) \quad (2.16)$$

Turunan kedua kurva bezier kubik dari t adalah :

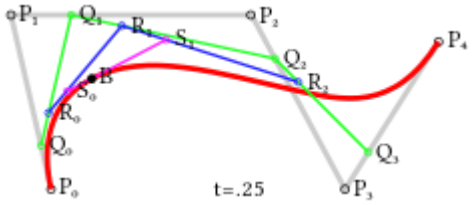
$$B''(t) = 6(1-t)(P_2 - 2P_1 + P_0) + 6t(P_3 - P_2 + P_1) \quad (2.17)$$

Titik poin tengah kurva kubik Q_0, Q_1, dan Q_2 menggambarkan kurva Bezier kuadrat :



Gambar 2. 9 Kurva Bezier Kubik

Untuk kurva Bezier kubik dengan titik poin 5 atau lebih, maka didapat poin tengah Q_0, Q_1, Q_2, dan Q_3 yang menggambarkan kurva linier, poin R_0, R_1, dan R_2 yang menggambarkan kurva Bezier kuadrat, dan poin S_0 dan S_1 yang menggambarkan kurva bezier kubik.



Gambar 2. 10 Kurva Bezier Kubik dengan Titik Poin 5

2.4 Transformasi Geometri

Menurut ashokib (2011) transformasi adalah proses perpindahan objek sehingga objek tersebut memiliki bentuk atau posisi yang baru.

Transformasi pada bidang, yaitu:

2.4.1 Translasi (Pergeseran)

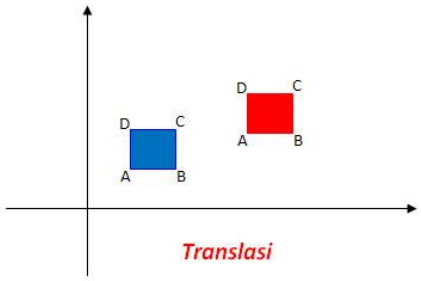
Perpindahan titik-titik pada bidang dengan cara menggeser titik-titik objek berdasar jarak dan arah tertentu.

Translasi

$T = \begin{pmatrix} a \\ b \end{pmatrix}$

memetakan titik $P(x, y)$ ke titik $P'(x', y')$ maka $x' = x + a$ dan $y' = y + b$ atau $P'(x + a, y + b)$ ditulis dalam bentuk:

$$T = \begin{pmatrix} a \\ b \end{pmatrix} : P(x, y) \rightarrow P'(x + a, y + b) \qquad (2.18)$$



Gambar 2. 11 Translasi (Pergeseran)

2.4.2 Rotasi (Perputaran)

Perpindahan titik-titik objek dengan cara memutar titik-titik objek berdasarkan besar sudut θ dan titik pusat rotasi. Rotasi terhadap titik pusat $O(0,0)$ (dilambangkan $R(O,\theta)$).

Jika titik $P(x,y)$ diputar sebesar θ berlawanan arah jarum jam terhadap $O(0,0)$, maka diperoleh $P'(x',y')$

$$R(O,\theta): P(x,y) \rightarrow P'(x',y') = P'(x\ cos\theta - y\ sin\theta, x\ sin\theta + y\ cos\theta) \tag{2.19}$$

Persamaan matriknya :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.20}$$

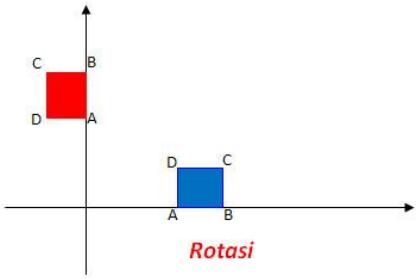
Rotasi terhadap titik pusat $P(a,b)$ (dilambangkan $R(O,\theta)$)

Jika titik $P(x,y)$ diputar sejauh θ berlawanan jaum arah jam terhadap titik $A(a,b)$ maka $P'(x',y')$

$$\begin{aligned} x' - a &= (x-a)\cos\theta - (y-b)\sin\theta \\ y' - b &= (x-a)\sin\theta + (y-b)\cos\theta \end{aligned} \tag{2.21}$$

Persamaan matriknya :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x-a \\ y-b \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix} \tag{2.22}$$



Gambar 2. 12 Rotasi (Perputaran)

2.4.3 Dilatasi (Perkalian)

Perubahan jarak titik-titik dengan faktor skala tertentu terhadap suatu titik tertentu dan titik pusat dilatasi.

Dilatasi terhadap titik pusat $O(0,0)$

Pemetaanya :

$$[O,k] : P(x,y) \rightarrow P'(kx,ky) \tag{2.23}$$

Persamaan matriksnya :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \tag{2.24}$$

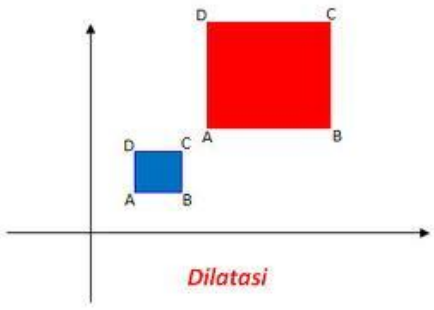
Dilatasi terhadap titik pusat $A(a,b)$

Titik $P(x,y)$ dilatasi terhadap titik $A(a,b)$ dengan faktor skala k, didapat bayangan $P'(x',y')$ dengan

$$x' - a = k(x-a) \text{ dan } y' - b = k(y-b) \tag{2.25}$$

Persamaan matriksnya :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} x-a \\ y-b \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix} \tag{2.26}$$



Gambar 2. 13 Dilatasi (Perkalian)

2.4.4 Refleksi (Pencerminan)

Memindahkan titik-titik dengan menggunakan sifat bayangan oleh suatu cermin terhadap garis sumbu tertentu.

Pencerminan terhadap sumbu X (dilambangkan M_x)

$$M_x : P(x,y) \rightarrow P'(x',y') = P'(x,-y) \tag{2.27}$$

Persamaan matriksnya :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.28}$$

Pencerminan terhadap sumbu Y (dilambangkan M_y)

$$M_y : P(x,y) \rightarrow P'(x',y') = P'(-x,y) \tag{2.29}$$

Persamaan matriksnya :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.30}$$

Pencerminan terhadap titik asal $O(0,0)$ (dilambangkan M_0)

$$M_0 : P(x,y) \rightarrow P'(x',y') = P'(-x,-y) \tag{2.31}$$

Persamaan matriksnya :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.32}$$

Pencerminan terhadap garis $y = x$ (dilambangkan $M_{y=x}$)

$$M_{y=x} : P(x,y) \rightarrow P'(x',y') = P'(x,y) \tag{2.33}$$

Persamaan matriksnya :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.34}$$

Pencerminan terhadap garis $y = -x$ (dilambangkan $M_{y=-x}$)

$$M_{y=-x} : P(x,y) \rightarrow P'(x',y') = P'(-x,-y) \tag{2.35}$$

Persamaan matriksnya :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.36}$$

Pencerminan terhadap garis $x = h$ (dilambangkan $M_{x=h}$)

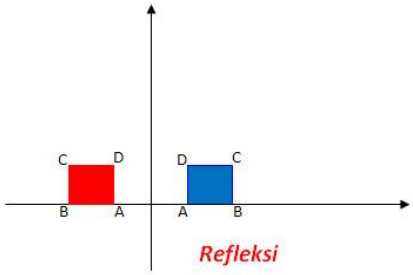
$$M_{x=h} : P(x,y) \rightarrow P'(x',y') = P'(2h - x,y) \tag{2.37}$$

Pencerminan terhadap garis y = k (dilambangkan $M_{y=k}$)

$$M_{y=k} : P(x,y) \rightarrow P'(x',y') = P'(x, 2k - y) \quad (2.38)$$

Pencerminan terhadap titik (a, b) (dilambangkan $M_{(a,b)}$)

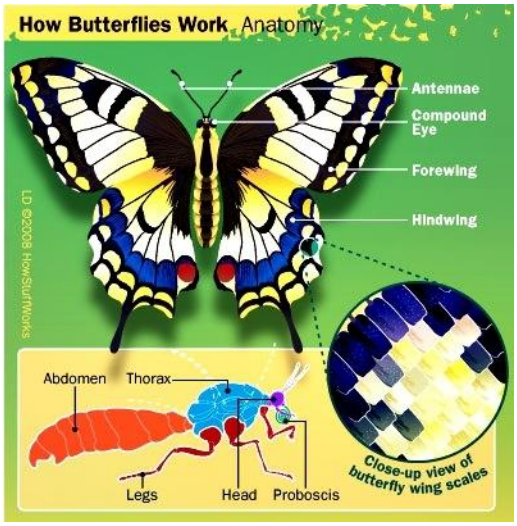
$$M_{(a,b)} : P(x,y) \rightarrow P'(x',y') = P'(2a - x, 2b - y) \quad (2.39)$$



Gambar 2. 14 Refleksi (Pencerminan)

2.5 Teori Anatomi Kupu

Menurut Estiara (2011) kupu-kupu merupakan serangga yang masuk dalam ordo Lepidoptera. Lepidoptera artinya adalah scaled wings atau bersayap sisik (lepis, sisik dan pteron, sayap). Sisik ini yang nantinya akan membuat sayap kupu-kupu mempunyai warna yang cerah. Tubuh kupu-kupu dewasa terdiri dari 3 bagian, kepala (head), dada (thorax) dan perut (abdomen).



Gambar 2. 15 Teori Anatomi Kupu-kupu

2.5.1 Kepala (Head)

Kepala adalah bagian dari serangga yang berisi otak, 2 mata kompon, probosis, dan faring (tenggorokan, dimana merupakan awal dari system pencernaan), dan 2 antena yang terpasang di kepala.

Bentuk geometri dari kepala adalah bola.

2.5.2 Antena (Antennae)

Antenna adalah alat sensor yang terdapat dikepala serangga dewasa. Antenna ini digunakan untuk mencium dan keseimbangan. Kupu-kupu mempunyai 2 antena dengan uhung yang sedikit membulat yang disebut sebagai antennal club.

Bentuk geometri antenna adalah kurva Bezier.

2.5.3 Mata Kompon (Compund Eye)

Mata kompon kupu-kupu terdiri dari banyak lensa hexagonal seperti halnya pada mata kompon serangga lainnya. Kupu-kupu hanya dapat melihat warna merah, hijau dan kuning saja.

Bentuk geometri mata kompon adalah bola.

2.5.4 Probosis (Proboscis)

Kupu-kupu dewasa menghisap nectar bunga dan cairan lainnya dengan mengguanakan probosis atau mulut penghisap yang seperti sedotan spiral. Ketika tidak digunakan, maka probosis ini akan digulung melingkar seperti selang air.

Bentuk geometri probosis adalah bola.

2.5.5 Palp Labial (Labial Palps)

Palp labial mempunyai fungsi untuk menentukan apakah sesuatu itu merupakan makanan atau bukan.

2.5.6 Dada (Thorax)

Dada adalah bagian diantara kepala dan perut dimana kaki dan sayap terpasang.

Bentuk geometri dada adalah tabung.

2.5.7 Sayap Depan (Forewing)

Forewing adalah sepasang sayap yang berada di depan atas.

Bentuk geometri sayap depan adalah kurva Bezier.

2.5.8 Sayap Belakang (Hindwing)

Hindwing adalah sepasang sayap yang berada di belakang bawah.

Bentuk geometri sayap belakang adalah kurva.

2.5.9 Kaki (Legs)

Kupu-kupu mempunyai sepasang kaki pendek yang berada di depan, dan 2 pasang kaki yang lebih panjang di belakangnya. Sepasang kaki ditengah dilengkapi sensor penciuman yang membuat kupu-kupu dapat merasakan kandungan kimia pada tempatnya hinggap.

2.5.10 Perut (Abdomen)

Perut merupakan bagian ekor serangga yang mempunyai segmentasi yang memiliki organ vital seperti jantung, tubulus atau pembuluh Malphigi untuk alat ekresi (pembuangan sisa metabolism dan benda tidak berguna lainnya), organ reproduksi dan sebagian besar system pencernaan.

Bentuk geometri perut adalah tabung.

2.6 OpenGL

OpenGL (Open Graphics Library) (dalam Silicon Graphics Inc, 1992) adalah spesifikasi grafik low-level yang menyediakan fungsi untuk mempermudah pekerjaan atau untuk keperluan pemrograman grafis, termasuk grafik primitif (titik, garis, dan lingkaran). OpenGL adalah sebuah library terdiri dari berbagai macam fungsi dan biasanya digunakan untuk menggambar sebuah objek 2D atau 3D.

OpenGL bersifat Open-Source, multi-platform dan multi-language. OpenGL dapat berjalan pada sistem operasi Windows, UNIX, SGI, LINUX dan lainnya. OpenGL pada awalnya didesain untuk digunakan pada bahasa pemrograman C/C++, namun dalam perkembangannya OpenGL dapan berjalan

pada bahasa pemrograman yang lain seperti Java, Tcl, Visual Basic, Delphi, Ada, maupun Fortran.

2.6.1 GLUT (GL Utility Toolkit)

Menurut Rosyidah (2013) GLUT merupakan pengembangan dari OpenGL yang didesain untuk aplikasi dengan level kecil hingga menegah dan menggunakan callback function untuk menambahkan interaksi dari user. GLUT menyediakan interface untuk manajemen window, menu, dan peralatan input (keyboard, dan mouse). GLUT juga menyediakan fungsi otomatis untuk menggambar objek primitive (titik, garis, lingkaran, persegi), objek 3 dimensi wire (kerangka) maupun yang solid seperti cube (kubus), sphere (bola), dan cone (kerucut), torus dan lain-lain.

GLUT berfungsi menciptakan fleksibilitas code antar platform yang dapat dijalankan lebih dari satu sistem operasi (Windows, Linux, Mac OS X, FreeBSD, OpenBSD, NetBSD), dan untuk lebih mudah mempelajari OpenGL, tanpa mengetahui spesifikasi sistem operasi dikarenakan OpenGL adalah sebagai mesin.

GLUT dibangun untuk menciptakan aplikasi grafis menggunakan pemrograman yang bersifat procedural. Terdapat fungsi *main loop* dan *looping* yang bertujuan untuk penanganan fungsi *callback* sebagai *input* user seperti fungsi *redraw*, *mouse*, *keyboard*, dan lain-lain.

Heriady (2007) memberitahukan untuk pemrograman OpenGL menggunakan C++ diperlukan library tambahan, yaitu :

a. Glut.h, gl.h, glu.h dan glaux.h yang dicopy ke C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\include\GL

b. Glut32.lib, glu32.lib, glut.lib, glaux.lib, dan opengl32.lib yang dicopy ke C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\lib

c. Glut32.dll, glu32.dll, dan opengl32.dll yang dicopy ke C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\bin

Untuk memulai pemrograman grafik pada OpenGL dibutuhkan fungsi sebagai berikut :

a. glutInit*Window*size(int width, int height);

Digunakan untuk membuat window untuk menampilkan objek 3D yang dibuat, width untuk menentukan lebar dari window yang akan dibuat, dan height untuk tinggi dari window dalam satuan pixel.

b. glutInit(int argc, char **argv);

Untuk menginisialisasi glut dan memproses argument command_line yang digunakan. Fungsi ini dipanggil sebelum pemanggilan fungsi yang lain.

c. glutInitDisplayMode(unsigned int mode);

Untuk menginisialisasi model dari tampilan. Variabel dapat diisi dengan GLUT_DOUBLE untuk menggunakan buffer pada window sebesar dua kali.
GLUT_RGB untuk menggunakan Red Green Blue Alpha pada pewarnaan.
GLUT_DEPTH untuk menggunakan depth buffer agar objek 3D yang ditampilkan lebih nyata.
Untuk mengkombinasikan variabel diatas digunakan garis vertikal (|).

d. glutCreateWindow(char *name);

Untuk membuat window dalam OpenGL dan menampilkan judul pada window tersebut.

Untuk mengatur tampilan dilayar monitor, dapat digunakan fungsi perspektif yang ditulis :
gluPerspective(float fovy, float aspect, float zNear, float zFar);
fovy untuk mengatur sudut pandang terhadap sumbu Y, aspect untuk rasio perbandingan antara lebar dan tinggi object, zNear adalah titik terdekat dan zFar adalah titik terjauh (keduanya harus bernilai positif).

Untuk mendukung fungsi gluPerspective digunakan fungsi glMatrixMode yang berfungsi untuk mengubah model matrix yang sedang aktif. Ditulis dengan glMatrixMode(GLenum mode);

Untuk mengaktifkan matrix proyeksi maka GLenum mode diisi dengan GL_PROJECTION, sedang untuk mengaktifkan matrix pandangan GLenum mode diisi dengan GL_MODELVIEW.

Untuk menampilkan object 3D pada layar monitor dibutuhkan fungsi :
void glutDisplayFunc(void (*func) (void));
Fungsi diatas ditulis didalam void main, kemudian membuat fungsi yang bertipe void agar dapat dipanggil oleh fungsi glutDisplayFunc. Didalam fungsi bertipe void tersebut objek 3D diletakkan.

Sebelum memulai menampilkan objek 3D pada fungsi bertipe void tersebut dilakukan permbersihan buffer lalu memanggil matrix identitas kemudian baru diisi objek 3D yang akan ditampilkan. Fungsi tersebut dituliskan dengan :
glClear(GLbitfield mask);
mask diisi dengan GL_COLOR_BUFFER_BIT untuk membersihkan buffer pada pewarnaan, GL_DEPTH_BUFFER_BIT untuk membersihkan depth buffer.
glLoadIdentity(void);
glutSwapBuffers(void);
Terdapat satu fungsi lagi yang digunakan untuk perulangan pada tampilan yang akan terus ditampilkan selama aplikasi tidak ditutup, yaitu glutMainLoop(void);

2.6.2 Teksturing

Heriady (2007) untuk memberikan tekstur pada objek tiga dimensi yang dibuat ada beberapa cara, tetapi disini kita gunakan teksturing dengan gambar berformat bitmap (*.bmp).

Untuk dapat menggunakannya terlebih dahulu kita harus memanggil file bitmap ke memori, dan mengatur koordinat tekstur dan menempelkannya pada polygon objek tiga dimensi.

Untuk dapat menjalankan teksturing pada OpenGL ditambahkan #include <GL/glaux.h> pada awal program.

Fungsi yang telah tersedia pada OpenGL untuk member tekstur bertipe file bitmap adalah :

AUX_RGBImageRec *auxDIBImageLoad(char *filename);
Untuk memanggil file bitmap, parameter filename adalah nama file bitmap yang akan dipakai.

Void glGenTextures(int n, int *textures);

Fungsi untuk menentukan tekstur yang akan dibuat sebanyak n dan berasal dari *textures, n diisi dengan nilai 1 apabila hanya satu tekstur dan *textures diisi dengan poin asal dari tekstur yang dibuat, file bitmap tersebut.

Void glBindTexture(GLenum target, GLuint texture);

Fungsi untuk mengubah tekstur ke format dua dimensi. Parameter target diisi dengan GL_TEXTURE_2D, sedang parameter texture diisi dengan tekstur yang akan dirubah ke dua dimensi. Bisa juga untuk menentukan tekstur apabila ada lebih dari satu tekstur dalam program yang akan dibuat.

Void glPixelStorei(GLenum pname, Glint param);

Fungsi untuk mengatur penyimpanan pixel dimemory.

Void glTexParameteri(GLenum target, GLenum pname, Glint param);

Fungsi untuk mengatur ***Filtering***, ***Repeating***, dan ***Clamping*** dari tekstur yang akan dibuat.

Int gluBuild2DMipmaps(GLenum target, Glint components, Glint width, Glint height, GLenum format, GLenum type, const void *data);

Fungsi untuk membuat tekstur 2dimensi agar dapat dibuat mipmap.

Void glTexImage2D(GLenum target, Glint level, Glint components, GLsizei width, GLsize height, Glint border, GLenum format, GLenum type, const GLvoid*pixels);

Fungsi untuk menggenerate tekstur 2dimensi, ini fungsi terakhir dalam pembuatan tekstur.

Void free(void *memblock);

Fungsi untuk membebaskan memory, agar memory dapat digunakan untuk proses yang lain.

Untuk membuat objek pada OpenGL kita memanggil fungsi glBegin(GLenum Mode);

GLenum mode diisi dengan primitif objek seperti titik, garis, polygon dll. Lalu dilanjutkan dengan

glVertex{234}{typedata}(typecoords);

234 adalah koordinat yang kita gunakan, typedata menentukan tipe data yang digunakan pada koordinat, typecoords diisi dengan titik koordinat yang ditentukan.

Lalu ditutup dengan glEnd();

Untuk membuat tekstur kita mengganti fungsi glVertex dengan fungsi glTextCoord2{typedata}(typecoords);

Untuk tekstur harus dibuat dalam bidang datar terlebih dahulu

Untuk menggabungkan beberapa objek menjadi satu maka kita memanggil dan menyimpan beberapa objek yang akan digabung kedalam satu matrix glPushMatrix(); dan glPopMatrix(); yang diantara tiap objek sudah dilakukan transformasi untuk menentukan posisi dan arah.

2.6.3 Animasi

Animasi adalah sekumpulan objek (gambar) yang disusun secara beraturan mengikuti alur pergerakan yang telah ditentukan. Gambar atau objek yang dimaksud dalam definisi di atas bisa berupa gambar manusia, hewan, maupun tulisan.

Menurut Guha (2011) teknik animasi pada OpenGL dibagi menjadi:

a. Interaktif, dengan menggunakan keyboard atau mouse untuk mengontol pergerakan animasi.

b. Otomatis, dengan fungsi waktu tunda yaitu saat program berjalan berhenti sesaat kemudian akan berjalan kembali setelah waktu jeda habis.

c. Otomatis, berdasarkan waktu yang ditentukan. Program akan berjalan sesuai dengan waktu yang ditentukan kemudian berhenti saat waktu habis.

BAB III
ANALISIS KEBUTUHAN & PERANCANGAN PERANGKAT LUNAK

3.1 Analisis Kebutuhan Sistem

Anasilis kebutuhan merupakan langkah awal untuk menentukan perangkat lunak yang akan dibuat.

3.1.1 Metode Analisis

Tahap ini digunakan untuk mengetahui semua kebutuhan dalam pengembangan animasi yang akan dibuat.

3.1.2 Hasil Analisis

a. Analisis Kebutuhan Proses

Berdasarkan analisis yang dilakukan penulis, proses yang akan dilakukan adalah sebagai berikut :

1) Proses pembuatan objek dengan menentukan titik-titik koordinat dari perhitungan matematika yang sudah ditentukan.
2) Proses teksturing dari kerangka objek yang telah dibuat.
3) Proses animasi

b. Analisis Keluaran Sistem

Keluaran yang dihasilkan adalah animasi dari proses metamorfosis kupu-kupu.

3.2 Perancangan Perangkat Lunak

3.2.1 Metode Perancangan

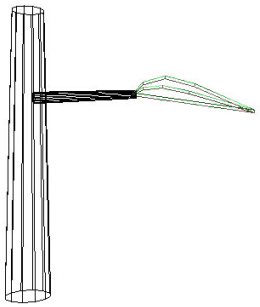
Metode perancangan animasi ini adalah sketsa objek dan papan alur cerita (*Storyboard*).

3.2.2 Hasil Perancangan

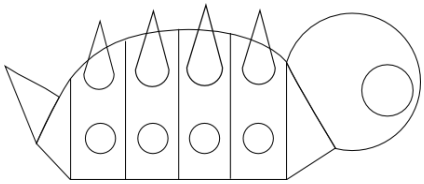
Bentuk objek kupu yang akan dibuat mengacu pada jenis kupu-kupu sayap burung (*Ornithoptera Priamus Poseidon*).

Gambar 3. 1 Kupu-kupu Sayap Burung

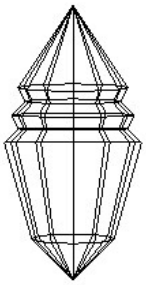
Untuk membuat objek pada animasi ini sebelumnya dibuat sketsa objek satu-persatu terlebih dahulu, objek yang akan dibuat antara lain pohon, daun, ulat, kepompong, kupu-kupu, sayap dan antena. Pohon, ulat, kepompong, dan kupu-kupu menggunakan objek dasar 3 dimensi tabung. Sedang objek daun menggunakan prinsip dasar prisma segitiga, dan sayap dan antenna dibuat dari kurva bezier. Berikut gambar sketsa objek yang akan dibuat dari objek tabung :



Gambar 3. 2 SketsaPohon



Gambar 3. 3 Sketsa Ulat



Gambar 3. 4 Sketsa Kepompong

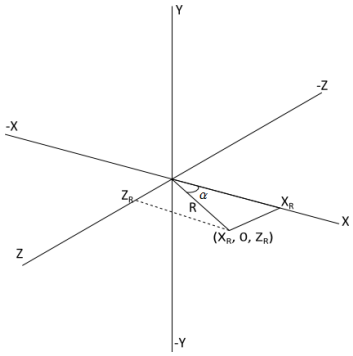
Gambar 3. 5 Sketsa Badan Kupu-kupu

Dari keempat gambar sketsa diatas terdapat sebuah kemiripan yaitu dibuat dari objek tabung yang alas dan tutup tabung memiliki jari-jari yang berbeda dan ditutup dengan objek kerucut sehingga ujung objek tabung tidak datar melingkar tetapi meruncing.

1) Tabung

Untuk membuat sebuah objek tabung pada bangun ruang tiga dimensi, pertama kali kita harus membuat alas/tutup tabung. Alas/tutup tabung disini tidak dibuat dari objek lingkaran melainkan dari objek segitiga yang dibuat melingkari sumbu Y seperti gambar dbawah.

Untuk mendapatkan bangun datar segitiga didapat dengan rumus matematika trigonometri segitiga. Dapat dilihat pada gambar dibawah :



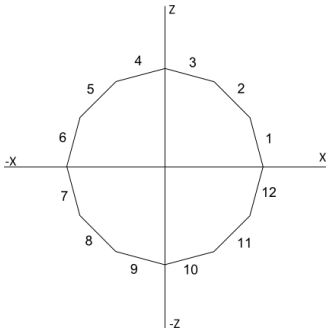
Gambar 3. 6 Trigonometri Segitiga pada Diagram Kartesius

Pada gambar diatas terdapat garis R dengan panjang 1.0 yang didapat dari dua titik vertex yaitu (0, 0, 0) dan (X_R, 0, Z_R) dan antara garis R dengan sumbu X membentuk sudut 30^0. Nilai X_R dan Z_R didapatkan dengan rumus sebelumnya, yaitu :

$$\begin{aligned} cos\,30^0 &= \frac{X_R}{R} & sin\,30^0 &= \frac{Z_R}{R} \\ X_R &= R \cdot cos\,30^0 & Z_R &= R \cdot cos\,30^0 \\ X_R &= 1.0 \cdot 0.866 & Z_R &= 1.0 \cdot 0.5 \\ X_R &= 0.866 & Z_R &= 0.5 \end{aligned} \tag{3.1}$$

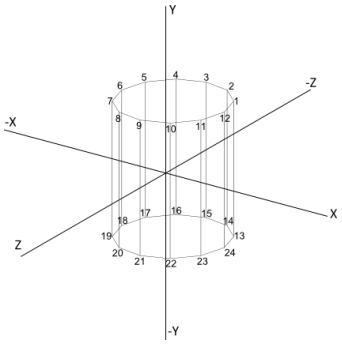
Maka hasil akhirnya (0.866, 0, 0.5)

Setelah didapatkan objek segitiga tersebut kemudian dibuat melingkar sehingga menjadi objek baru lingkaran.



Gambar 3. 7 Lingkaran pada Diagram Kartesius

Gambar diatas adalah hasil dari objek segitiga yang disusun memutar 360^0. Nilai 1 sampai 12 pada gambar di atas adalah sisi polygon yang dibuat.



Gambar 3. 8 Tabung 12 sisi

Pada gambar tabung 12 sisi diatas terdapat urutan angka yang akan dijadikan array untuk menyimpan nilai titik vertexnya. Sisi polygon 1 pada tabung memiliki 4 nilai vertex, yaitu vertex 1, 2, 13, 14, dan sisi polygon 2 nilai vertexnya 2, 3, 14, 15, dan seterusnya.

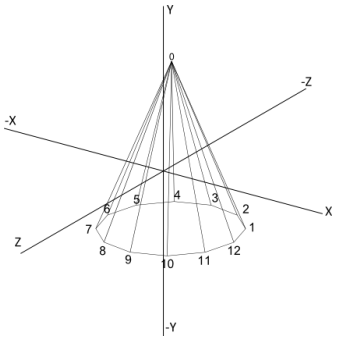
Besar sudut masing-masing tabung 12 sisi yaitu 360^0 : 12 = 30^0. Untuk membuat tabung 12 sisi menjadi fleksibel maka dibutuhkan dua variabel yaitu R sebagai jari-jari dan tinggi tabung. Nilai n sebagai penentu tinggi tabung pada koordinat Y. Sehingga untuk menentukan nilai vertex masing-masing tiap array didapat dengan persamaan :

Tabel 3. 1 Perhitungan Nilai Vertex Tabung

Vertex 1 : $X = R \cdot \cos 30^0$ $Y = n \cdot \text{tinggi}$ $Z = R \cdot \sin 30^0$	Vertex 3 : $X = R \cdot \cos 30^0$ $Y = -n \cdot \text{tinggi}$ $Z = R \cdot \sin 30^0$
Vertex 2 : $X = R \cdot \cos 30^0$ $Y = -n \cdot \text{tinggi}$ $Z = R \cdot \sin 30^0$	Vertex 4 : $X = R \cdot \cos 30^0$ $Y = n \cdot \text{tinggi}$ $Z = R \cdot \sin 30^0$

2) Kerucut

Untuk bangun ruang kerucut, hampir sama dengan tabung yaitu dengan menyimpan vertex ke dalam array. Hanya saja vertex yang disimpan setiap sisi polygon ada tiga, sedang sisi tabung 12 sisi ada empat.



Gambar 3. 9 Kerucut

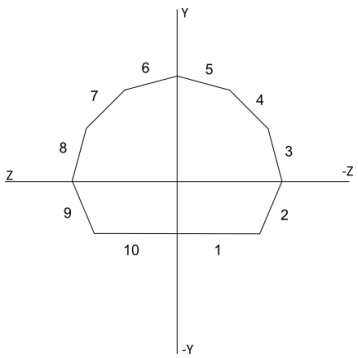
Dari gambar diatas maka dapat diketahui titik vertex yang akan disimpan pada sisi polygon pertama kerucut, yaitu vertex 0, 1, 2, dan polygon kedua 0, 2, 3 dan seterusnya. Nilai vertex tiap array yaitu :

Tabel 3. 2 Perhitungan Nilai Vertex Kerucut

Vertex 1 : $X = R \cdot \cos 30^0$ $Y = n \cdot \text{tinggi}$ $Z = R \cdot \sin 30^0$	Vertex 3 : $X = R \cdot \cos 30^0$ $Y = -n \cdot \text{tinggi}$ $Z = R \cdot \sin 30^0$
Vertex 2 : $X = R \cdot \cos 30^0$ $Y = -n \cdot \text{tinggi}$ $Z = R \cdot \sin 30^0$	

3) Ulat

Untuk objek ulat, tabung yang dibuat berbeda yaitu dibuat seperti pada gambar dibawah ini.



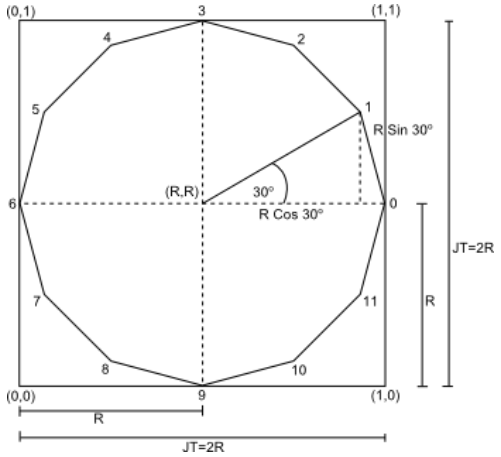
Gambar 3. 10 Polygon Tabung untuk Ulat

Perhitungan untuk mencari titik vertexnya sama dengan tabung hanya perbedaan nilai sudut pada polygon 1 dan 10 yang berbeda.

4) Teksturing Tabung dan Kerucut

Setelah objek tabung dibuat, kemudian diberikan tekstur agar sesuai objek yang akan kita buat. Menurut Heriady (2007) cara untuk memberi tekstur pada objek 3 dimensi disebut sebagai proses rendering. Untuk tutup/alas tabung dan kerucut, dapat dilihat pada gambar dibawah ini :

a. Tutup / alas Tabung dan Kerucut



Gambar 3. 11 Tutup/alas Tabung dan kerucut

Tekstur koordinatnya (0,0) , (1,0) , (1,1) , (0,1). Titik pusat ada pada (R,R), R adalah jari-jari tutup tabung, sehingga JT=2xR. JT adalah jarak terpanjang.

Untuk menghitung titik koordinat x, y maka diketahui sudut yang dibentuk pada vertex 0 dan 1 adalah 30^0. Sehingga $x = R\cos 30^0$ dan $y = R\sin 30^0$. Posisi koordinat tekstur dihitung dari titik (0,0) maka nilai x dan y ditambah dengan nilai JT, sehingga rumus yang digunakan :

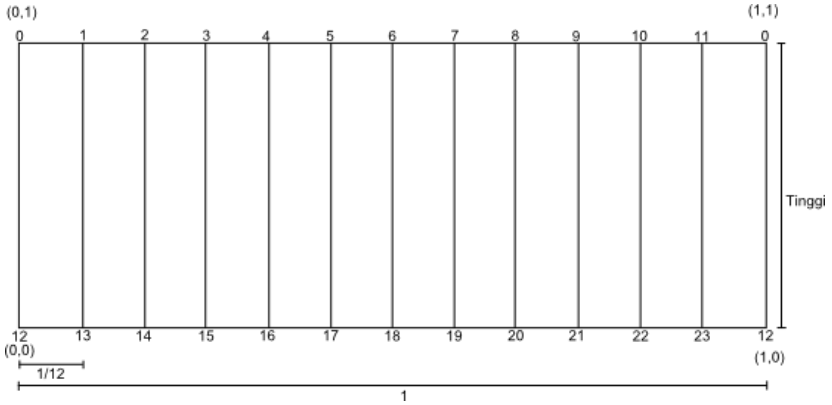
$$x = (R + R\,cos\,30^0) \div JT$$
$$= R(1 + cos\,30^0) \div JT$$

dan (3.2)

$$y = (R + R\,sin\,30^0) \div JT$$
$$= R(1 + sin\,30^0) \div JT$$

b. Selimut Tabung

untuk membuat tekstur selimut tabung maka, pada selimut tabung dibentangkan menjadi sebuah persegi panjang untuk kemudian di bagi berdasarkan sisi polygon yang dibuat.

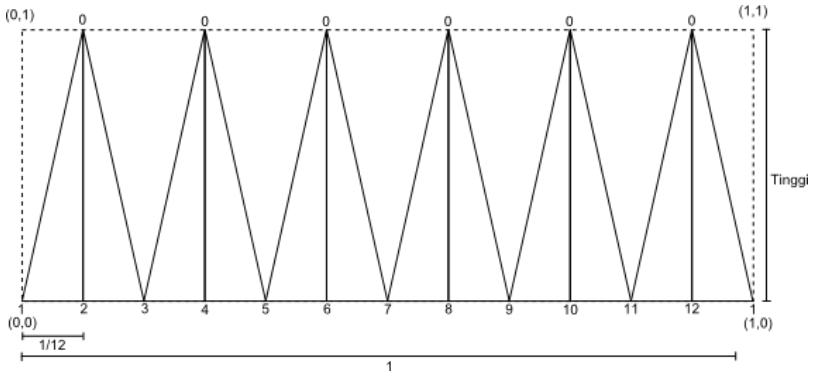


Gambar 3. 12 Selimut Tabung

Pada gambar selimut tabung diatas yang telah dibentangkan pada tekstur dengan titik koordinat (0,0) , (1,0) , (1,1) , (0,1). Angka 0 sampai 23 adalah urutan array yang akan kita buat. Nilai sumbu X sebesar 1, maka tiap sisi polygon memerlukan panjang tekstur 1/12. Sehingga polygon 1 urutan arraynya 0, 12, 13, 1 dengan tekstur koordinat (0,1) , (0,0) , (1/12,0) , (1/12,1). Polygon 2 arraynya 13, 1, 2, 14 dengan tekstur koordinat (1/12,0) , (1/12,1) , (2/12,1) , (2/12,0) dan seterusnya.

c. Selimut kerucut

Untuk merender selimut kerucut, kita gunakan cara yang sama dengan selimut tabung, tetapi yang membedakan adalah polygon selimut kerucut berbentuk segitiga sedang selimut tabung berbentuk persegi.

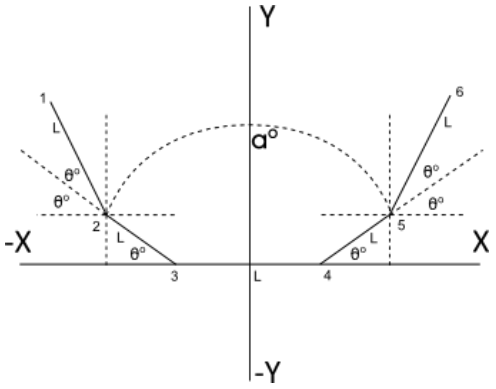


Gambar 3. 13 Selimut kerucut

Polygon 1 urutan arraynya 0,1,2 dan tekstur koordinat (1/12,1) , (0,0) , (1/12,0). Polygon 2 urutan arraynya 0,2,3 dan tekstur koordinat (1/12,1) , (1/12,0) , (2/12,0). Dan seterusnya.

5) Daun

Daun dibuat dengan metode pembuatan garis bengkok pada koordinat kartesius. Garis bengkok adalah sebuah garis yang melengkung membentuk sudut sebesar α^0.



Gambar 3. 14 Garis Bengkok

Pada gambar diatas, garis bengkok dibuat menggunakan lima buah garis dan setiap garis dibentuk oleh dua buah titik vertex (1, 2, 3, 4, 5, 6). Panjang kelima garis sama dan dinyatakan dalam L. Sudut θ^0 dihitung dengan cara :

$$\theta^0 = (180^0 - \alpha^0) \div 4 \qquad (3.3)$$

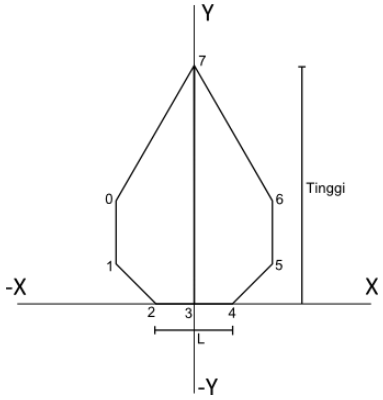
Pembagi 4 didapatkan dari kelima garis yang membentuk sudut sama besar, nilai α^0 diantara 0^0 sampai 180^0, jika nilai $\alpha^0 = 0^0$ maka garis akan membentuk seperti huruf U dan jika $\alpha^0 = 180^0$ maka garis akan membentuk garis lurus. Keenam titik vertex tersebut dapat dihitung dengan cara:

Tabel 3. 3 Perhitungan Nilai Vertex Garis Bengkok

Vertex 1 : $X = -(\text{vertex } 2X + L\cos 2\theta^0)$ $= -(0.5 \cdot L + L\cos\theta^0 + L\cos 2\theta^0)$ $Y = \text{vertex } 2Y + L\sin 2\theta^0$ $= L\ \sin\theta^0 + L\sin 2\theta^0$ $Z = 0$	Vertex 4 : $X = -0.5 \cdot L$ $Y = 0$ $Z = 0$

Vertex 2 : $X = -(\text{vertex } 3X + L\cos\theta^0)$ $= -(0.5 \cdot L + L\cos\theta^0)$ $Y = L\sin\theta^0$ $Z = 0$	Vertex 5 : $X = \text{vertex } 4X + L\cos\theta^0$ $= 0.5 \cdot L + L\cos\theta^0$ $Y = L\sin\theta^0$ $Z = 0$
Vertex 3 : $X = -0.5 \cdot L$ $Y = 0$ $Z = 0$	Vertex 6 : $X = \text{vertex } 5X + L\cos 2\theta^0)$ $= 0.5 \cdot L + L\cos\theta^0 + L\cos 2\theta^0)$ $Y = \text{vertex } 5Y + L\sin 2\theta^0$ $= L\ \sin\theta^0 + L\sin 2\theta^0$ $Z = 0$

Objek daun yang akan dibuat berbentuk seperti gambar dibawah ini :

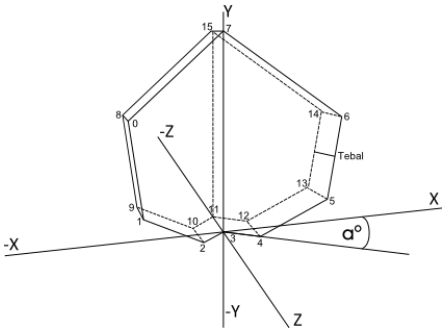


Gambar 3. 15 Sketsa Daun

Sudut yang dibentuk oleh α sebesar 0^0 sehingga :

$\theta^0 = (180^0 - 0^0) \div 4$
$\theta^0 = 45^0$

Agar objek terlihat nyata maka dibuat melengkung sebesar α^0 dan tebal daun seperti pada gambar :



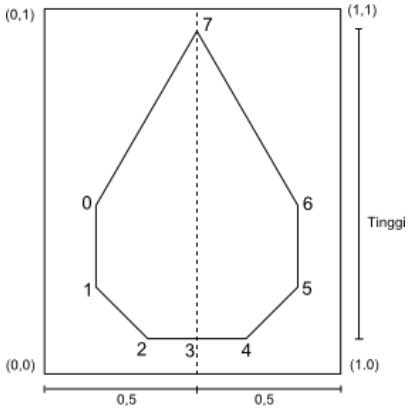
Gambar 3. 16 Objek Daun 3 Dimensi

Pada gambar diatas selain titik vertex 3, 7, 11, dan 15, ditambahkan nilai α^0. Perhitungan nilai vertexnya pada setiap array :

Tabel 3. 4 Perhitungan Nilai Vertex Daun

Array	Vertex		
	X	Y	Z
0	$-\mathrm{L}(0.5+\cos\theta^0+\cos 2\theta^0)$	$\mathrm{L}\sin\theta^0+\sin 2\theta^0$	$\mathrm{X}\tan\alpha^0$
1	$-\mathrm{L}(0.5+\cos\theta^0)$	$\mathrm{L}\sin\theta^0$	$\mathrm{X}\tan\alpha^0$
2	$-0.5\cdot\mathrm{L}$	0	$\mathrm{X}\tan\alpha^0$
3	0	0	0
4	$-0.5\cdot\mathrm{L}$	0	$\mathrm{X}\tan\alpha^0$
5	$-\mathrm{L}(0.5+\cos\theta^0)$	$\mathrm{L}\sin\theta^0$	$\mathrm{X}\tan\alpha^0$
6	$-\mathrm{L}(0.5+\cos\theta^0+\cos 2\theta^0)$	$\mathrm{L}\sin\theta^0+\sin 2\theta^0$	$\mathrm{X}\tan\alpha^0$
7	0	Tinggi	0
8	$-\mathrm{L}(0.5+\cos\theta^0+\cos 2\theta^0)$	$\mathrm{L}\sin\theta^0+\sin 2\theta^0$	$\mathrm{X}\tan\alpha^0-\mathrm{tebal}$
9	$-\mathrm{L}(0.5+\cos\theta^0)$	$\mathrm{L}\sin\theta^0$	$\mathrm{X}\tan\alpha^0-\mathrm{tebal}$
10	$-0.5\cdot\mathrm{L}$	0	$\mathrm{X}\tan\alpha^0-\mathrm{tebal}$
11	0	0	0
12	$-0.5\cdot\mathrm{L}$	0	$\mathrm{X}\tan\alpha^0-\mathrm{tebal}$
13	$-\mathrm{L}(0.5+\cos\theta^0)$	$\mathrm{L}\sin\theta^0$	$\mathrm{X}\tan\alpha^0-\mathrm{tebal}$
14	$-\mathrm{L}(0.5+\cos\theta^0+\cos 2\theta^0)$	$\mathrm{L}\sin\theta^0+\sin 2\theta^0$	$\mathrm{X}\tan\alpha^0-\mathrm{tebal}$
15	0	Tinggi	0

6) Teksturing Daun



Gambar 3. 17 Tekstur Daun

Gambar diatas adalah gambar polygon daun yang ditempelkan pada permukaan tekstur dengan koordinat (0,0) , (0,1) , (1,1) , (1,0). Untuk mencari titik koordinat tekstur dapat digunakan dalil phytagoras. Titik koordinatnya dapat dilihat pada table dibawah :

Tabel 3. 5 Perhitungan Nilai Vertex Tekstur Daun

Array	Vertex	
	X	Y
0	$0.5 - (\sqrt{(\text{vertex X}^2 + \text{vertex Z}^2)})/\text{JT}$	Vertex Y / JT
1	$0.5 - (\sqrt{(\text{vertex X}^2 + \text{vertex Z}^2)})/\text{JT}$	Vertex Y / JT
2	$0.5 - (\sqrt{(\text{vertex X}^2 + \text{vertex Z}^2)})/\text{JT}$	0
3	0.5	0
4	$0.5 - (\sqrt{(\text{vertex X}^2 + \text{vertex Z}^2)})/\text{JT}$	Tinggi / JT
5	$0.5 - (\sqrt{(\text{vertex X}^2 + \text{vertex Z}^2)})/\text{JT}$	Vertex Y / JT
6	$0.5 - (\sqrt{(\text{vertex X}^2 + \text{vertex Z}^2)})/\text{JT}$	Vertex Y / JT
7	0.5	Tinggi / JT

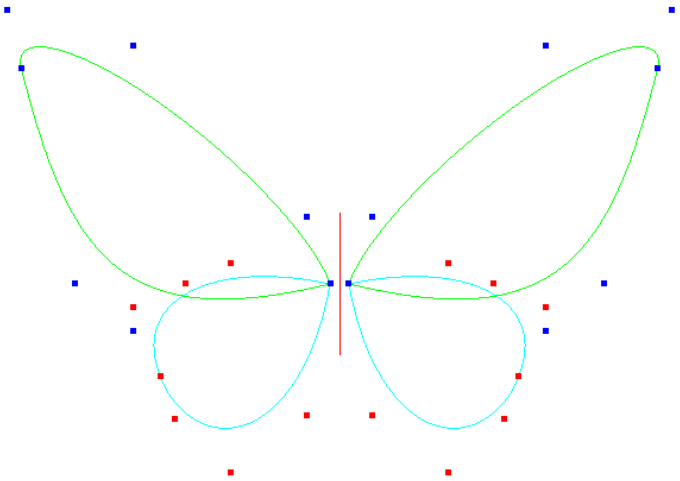
7) Bola

Untuk membuat bola digunakan persamaan quadric yang didefinisikan oleh pesamaan kuadrat umum :

$$a1x2 + a2y2 + a3z2 + a4xy + a5yz + a6xz + a7x + a8y + a9z + a10 = 0$$

8) Sayap

Untuk membuat sayap kupu-kupu menggunakan metode garis bezier kubik yang dibuat dengan 5 titik dan 4 titik yang dihubungkan menjadi satu objek.



Gambar 3. 18 Sketsa Sayap Kupu-kupu

Untuk sayap bagian atas dan bawah dibuat terlebih dahulu titik-titik koordinat yang digunakan sebagai penentu garis bezier. Sayap atas titik koordinatnya berwarna biru, sedang sayap bawah titik koordinatnya berwarna merah. Garis berwarna hijau dan biru laut adalah garis Bezier yang dibuat dari titik-titik koordinat dibawah ini.

Untuk sayap atas titik koordinatnya :

Garis bezier 1 : (0.1, 0.1, 0) , (1, 1.8, 0) , (5, 5, 0) , (7, 5, 0) , (6.8, 4, 0)

Garis bezier 2 : (6.8, 4, 0) , (6, 0, 0) , (5, -1, 0) , (0.1, 0.1, 0)

Untuk sayap bawah titik koordinatnya :

Garis bezier 3 : (0.2, -0.1, 0) , (3, 0.5, 0) , (4, 0, 0) , (5, -0.5, 0) , (4.5, -2, 0)

Garis bezier 4 : (0.2, -0.1, 0) , (1, -3.5, 0) , (3, -4.5, 0) , (4.2, -3, 0) , (4.5, -2, 0)

Untuk sayap sebelah kiri dibuat pencerminannya dari sayap sebelah kanan terhadap sumbu Y.

9) Antena

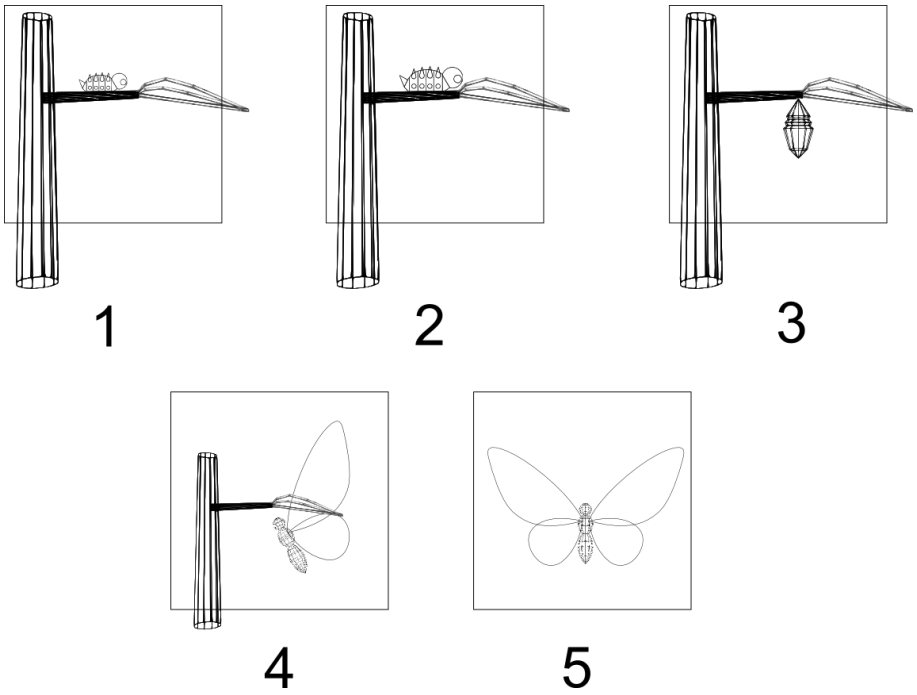
Antena dibuat dari kurva garis bezier kubik 5 titik. Dengan titik koordinatnya adalah (0.1, 0, 0), (1, 2, 0), (2, 4, 0), (1, 1.7, 0), (0.1, 0, 0).

10) Huruf

Guha (2011) mengatakan untuk menampilkan huruf, ada dua cara yaitu dengan *bitmapped(raster)* atau *stroke(vector)*. *Bitmapped* dibuat berdasarkan susunan blok persegi, sedangkan *stroke* dibuat dari objek garis. Jenis huruf yang tersedia hanya terbatas, untuk mode *bitmapped* hanya ada 3 jenis huruf dan *stroke* ada 2 jenis huruf.

11) Papan Alur Cerita (*Storyboard*)

Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan storyboard kita dapat menyampaikan ide cerita kita kepada orang lain dengan lebih mudah, karena kita dapat menggiring khayalan seseorang mengikuti gambar-gambar yang tersaji, sehingga menghasilkan persepsi yang sama pada ide cerita kita.



Gambar 3. 19 Papan Alur Cerita

Penjelasan alur cerita :

1. Setelah menetas, larva ulat akan mencari makan. Sebagian larva mengkonsumsi cangkang telur yang kosong sebagai makanan pertamanya. Kulit luar larva tidak meregang mengikuti pertumbuhannya, tetapi ketika menjadi

sangat ketat larva akan berganti kulit. Jumlah pergantian kulit larva selama hidup antara 4-6 kali, dan periode antara pergantian kulit (molting) disebut instar.

2. Ketika larva mencapai pertumbuhan maksimal, larva akan berhenti makan, berjalan mencari tempat berlindung terdekat, melekatkan diri pada ranting atau daun dengan anyaman benang. Larva telah memasuki fase prepupa dan melepaskan kulit terakhir kali untuk membentuk pupa.

3. Fase pupa jika dilihat dari luar seperti periode tidur, tetapi didalam pupa terjadi proses pembentukan serangga yang sempurna. Pupa pada umumnya keras, halus dan berupa struktur tanpa anggota tubuh. Umumnya pupa berwarna hijau, coklat atau warna sesuai sekitarnya. Pembentukan pupa biasanya berlangsung selama 7-20 hari tergantung spesiesnya.

4. Setelah keluar dari pupa, kupu-kupu akan merangkak ke atas sehingga sayapnya yang lemah, kusut dan agak basah menggantung ke bawah dan mengembang secara normal. Setelah sayap mengembang dan kuat, sayap akan membuka dan menutup beberapa kali dan percobaan terbang.

5. Fase imago atau kupu-kupu adalah fase dewasa.

12) Animasi

Teknik animasi yang digunakan adalah otomatis-interaktif. Yaitu penggabungan antara teknik otomatis dengan waktu jeda dan interaktif dengan pengguanaan tombol *keyboard*.

a. Gerakan Sayap

Untuk membuat animasi sayap kupu-kupu mengepak, maka ditentukan terlebih dahulu posisi awal dan posisi akhir sayap saat mengepak. Maka dipakailah rumus **2.9 Garis Bezier Linier** sebagai penentu gerakan sayap mengepak dan waktu jeda animasi diantara 0-1.

Karena sayap kupu-kupu ada dua di sebelah kanan, depan dan belakang, dan dua di sebelah kiri, depan dan belakang, maka total sayap ada empat. Jadi masing-masing sayap ditentukan terlebih dahulu titik awal dan titik akhir untuk animasinya dalam satuan sudut, yaitu :

Tabel 3. 6 Titik Awal dan Akhir Gerakan Sayap

Sayap	Titik Awal P_0	Titik Akhir P_1
Depan Kanan	-30^0	50^0
Belakang Kanan	-20^0	50^0
Depan Kiri	30^0	-50^0
Belakang Kiri	20^0	-50^0

Setelah dibuat titik awal dan titik akhir nya, selanjutnya menentukan alur pergerakan untuk animasinya. Untuk membuat gerakan animasi kepakan sayap menjadi lebih hidup maka kita meniru gerakan pendulum (*Osilasi*).

Gerak pendulum termasuk gerak harmonik sederhana, yaitu gerak bolak-balik benda melalui titik keseimbangan tertentu dengan banyaknya getaran dalam setiap detik secara konstan.

Animasi untuk kepak sayap dibuat dengan teknik otomatis dengan fungsi waktu tunda, sehingga sayap dibuat akan terus mengepak.

b. Gerakan Geser

Untuk membuat objek animasi berjalan menggeser, kita menggunakan prinsip transformasi geometri **2.18 translasi**. Parameter animasi ditentukan dengan nilai t, nilai waktu tiap *frame* kurang lebih 1/100-1/200 detik. Objek yang akan digerakkan di geser sejauh nilai t pada koordinat yang diinginkan. Kemudian dibuat sebuah fungsi baru untuk menentukan nilai t agar terus naik / bertambah.

3.3 Analisis Pengujian Perangkat Lunak

3.3.1 Pengujian Objek

Pengujian objek untuk mengetahui apakah objek yang dibuat sesuai dengan sketsa

3.3.2 Pengujian Animasi

Pengujian animasi untuk mengetahui apakah animasi yang dibuat sesuai dengan papan alur cerita yang dibuat atau tidak.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Perangkat Keras dan Lunak yang Digunakan

4.1.1 Perangkat Keras

Perangkat keras yang digunakan untuk membangun aplikasi animasi metamorfosis kupu-kupu dengan OpenGL ini adalah sebagai berikut :

1) Processor Intel i5
2) Ram 4GB
3) VGA ATI Radeon HD 6470

4.1.2 Perangkat Lunak

perangkat lunak yang dibutuhkan dalam pembuatan animasi metamorphosis kupu-kupu dengan OpenGL ini adalah sebagai berikut :

1) *windows 7 home premium 64-bit*

Windows merupakan sistem operasi yang dibuat oleh *Microsoft. Windows* digunakan karena sebagian besar software yang ada dapat berjalan dan digunakan di sistem operasi ini.

2) *Microsoft visual studio 2012*

Microsoft visual studio 2012 adalah *software* yang memiliki paket pemrograman yang cukup lengkap. Bahasa pemrograman yang ada pada *Microsoft visual studio 2012* antara lain C++, C# dan VB dan memiliki kompatibilitas antara C++ dengan library OpenGL.

4.2 Hasil

4.2.1 Pohon

Hasil perhitungan array untuk objek pohon dapat dilihat pada Lampiran 1. Dari nilai array yang didapatkan pada table Lampiran 1, maka kita dapat membuat fungsi untuk menggambarkan titik koordinat tersebut pada program. Berikut *pseudocode* nya

```
1.    Mulai
2.    Cek file = benar
```

```
3.    Buka file pohon.txt
4.    If file tidak ada
5.    Cetak "error buka file"
6.    While data file masih
7.    File baca pohon ke-i.X >> pohon ke-i.Y >> pohon ke-i.Z
8.    For i = 1 to 12
9.    Sudut = (pohon ke-i.X / 180) * pi
10.   pohon ke-i.X = (jari * 0.7) * cos(sudut)
11.   pohon ke-i.Y = pohon ke-i.Y * tinggi
12.   Sudut = (pohon ke-i.Z /180) * pi
13.   pohon ke-i.Z = (jari * 0.7) * sin (sudut)
14.   End for
15.   For i = 13 to 24
16.   Sudut = (pohon ke-i.X / 180) * pi
17.   pohon ke-i.X = jari * cos(sudut)
18.   pohon ke-i.Y = pohon ke-i.Y * tinggi
19.   Sudut = (pohon ke-i.Z /180) * pi
20.   pohon ke-i.Z = jari * sin (sudut)
21.   End for
22.   Selesai
```

Syntak pemrograman fungsi pohon dapat dilihat pada Lampiran 2.

4.2.2 Teksturing Tabung

Hasil perhitungan array untuk teksturing objek tabung dapat dilihat pada lampiran 3. Dari nilai array yang didapatkan pada table lampiran 3, maka kita dapat membuat fungsi untuk menggambarkan titik koordinat tersebut pada program. Berikut *pseudocode* nya

```
1.    Mulai
2.    Buat variabel JT
3.    Cek file = benar
4.    Buka file tekstur.txt
5.    If file tidak ada
6.    Cetak "error buka file"
7.    While data file masih
8.    File baca tekstur ke-i.X >> tekstur ke-i.Y
9.    For i = 0 to 25
10.   Tekstur ke-i.X = teksturke-i.X / 12
11.   End for
12.   JT = 2 * jari
```

```
13.    For i = 26 to 37
14.    Sudut = (tekstur ke-i.X / 180) * pi
15.    Tekstur ke-i.X = (jari * (1 + cos(sudut))) / JT
16.    Sudut = (tekstur ke-i.Y /180) * pi
17.    Tekstur ke-i.Z = (jari * (1 + sin (sudut))) / JT
18.    End for
19.    Selesai
```

Syntak pemrograman fungsi teksturing dapat dilihat pada lampiran 4.

4.2.3 Daun

Untuk objek daun, kita tidak menyimpan nilai array, tetapi kta langsung membuatnya pada program. *Pseudocode* nya

```
1.     Mulai
2.     Buat variabel sudut teta = pi / 4
3.     Buat variabel sudut 2 teta = pi / 2
4.     Buat variabel t alpha = ( sudut alpha / 180) * pi
5.     Daun ke-0.X = -L * ( 0.3 + cos (sudut teta) + cos (sudut 2 teta))
6.     Daun ke-0.Y = L * ( sin (sudut teta ) + sin (sudut 2 teta))
7.     Daun ke-0.Z = -daun ke-0.X * tan (t alpha)
8.     Daun ke-1.X = -L * ( 0.5 + cos ( sudut teta ))
9.     Daun ke-1.Y = L * sin ( sudut teta )
10.    Daun ke-1.Z = -daun ke-1.X * tan (t alpha)
11.    Daun ke-2.X = -L * 0.7
12.    Daun ke-2.Y = 0
13.    Daun ke-2.Z = -daun ke-2.X * tan (t alpha)
14.    Daun ke-3.X = 0
15.    Daun ke-3.Y = -L * 0.3
16.    Daun ke-3.Z = 0
17.    Daun ke-4.X = L * (0.7
18.    Daun ke-4.Y = 0
19.    Daun ke-4.Z = daun ke-4.X * tan (t alpha)
20.    Daun ke-5.X = L * ( 0.5 + cos ( sudut teta ))
21.    Daun ke-5.Y = L * sin ( sudut teta )
22.    Daun ke-5.Z = daun ke-5.X * tan (t alpha)
23.    Daun ke-6.X = L * ( 0.3 + cos (sudut teta) + cos (sudut 2 teta))
24.    Daun ke-6.Y = L * ( sin (sudut teta ) + sin (sudut 2 teta))
25.    Daun ke-6.Z = daun ke-6.X * tan (t alpha)
26.    Daun ke-7.X = 0
27.    Daun ke-7.Y = tinggi
```

```
28.    Daun ke-7.Z = 0
29.    For i = 0 to 7
30.    Daun ke-i+8.X = daun ke-i.X
31.    Daun ke-i+8.Y = daun ke-i.Y
32.    Daun ke-i+8.Z = daun ke-i.Z – tebal
33.    End for
34.    Selesai
```

Syntak pemrograman fungsi daun dapat dilihat pada lampiran 5.

4.2.4 Teksturing Daun

Untuk teksturing pada daun kita juga membuatnya langsung pada program, berikut *pseudocode* nya

```
1.     Mulai
2.     DaunTK ke-0.X = 0.5 – ( akar (( daun ke-0.X * daun ke-0.X )  + (
daun ke-0.Z * daun ke-0.Z ))  / JT )
3.     Daun TK ke-0.Y = daun ke-0.Y / JT
4.     DaunTK ke-1.X = 0.5 – ( akar (( daun ke-1.X * daun ke-1.X )  + (
daun ke-1.Z * daun ke-1.Z ))  / JT )
5.     Daun TK ke-1.Y = daun ke-1.Y / JT
6.     DaunTK ke-2.X = 0.5 – ( akar (( daun ke-2.X * daun ke-2.X )  + (
daun ke-2.Z * daun ke-2.Z ))  / JT )
7.     Daun TK ke-2.Y = 0
8.     DaunTK ke-3.X = 0.5
9.     Daun TK ke-3.Y = 0
10.    DaunTK ke-4.X = 0.5 – ( akar (( daun ke-4.X * daun ke-4.X )  + (
daun ke-4.Z * daun ke-4.Z ))  / JT )
11.    Daun TK ke-4.Y = 0
12.    DaunTK ke-5.X = 0.5 – ( akar (( daun ke-5.X * daun ke-5.X )  + (
daun ke-5.Z * daun ke-5.Z ))  / JT )
13.    Daun TK ke-5.Y = daun ke-5.Y / JT
14.    DaunTK ke-6.X = 0.5 – ( akar (( daun ke-6.X * daun ke-6.X )  + (
daun ke-6.Z * daun ke-6.Z ))  / JT )
15.    Daun TK ke-6.Y = daun ke-6.Y / JT
16.    DaunTK ke-7.X = 0.5
17.    Daun TK ke-7.Y = tinggi / JT
18.    End for
19.    Selesai
```

Syntak pemrograman fungsi teksturing daun dapat dilihat pada lampiran 6.

4.2.5 Ulat

Hasil perhitungan nilai array untuk ulat dapat dilihat pada lampiran 7. Dari nilai array yang didapatkan pada table lampiran 7, maka kita dapat membuat fungsi untuk menggambarkan titik koordinat tersebut pada program. Berikut *pseudocode* nya

```
1.    Mulai
2.    Cek file = benar
3.    Buka file ulat.txt
4.    If file tidak ada
5.    Cetak "error buka file"
6.    While data file masih
7.    File baca ulat ke-i.X >> ulat ke-i.Y >> ulat ke-i.Z
8.    For i = 0 dan 51
9.    Ulat ke-i.X = ulat ke-i.X * panjang
10.   Ulat ke-i.Y = ulat ke-i.Y
11.   Ulat ke-i.Z = ulat ke-i.Z
12.   End for
13.   For i = 1 to 6
14.   ulat ke-i.X = ulat ke-i.X * panjang
15.   Sudut = (ulat ke-i.Y / 180) * pi
16.   Array ke-i.Y = ( jari * 0.25 ) * cos ( sudut )
17.   Sudut = (array ke-i.Z /180) * pi
18.   Array ke-i.Z = ( jari * 0.25 ) * sin (sudut)
19.   End for
20.   For i = 6 to 15
21.   ulat ke-i.X = ulat ke-i.X * panjang
22.   Sudut = (ulat ke-i.Y / 180) * pi
23.   Array ke-i.Y = ( jari * 0.5 ) * cos ( sudut )
24.   Sudut = (array ke-i.Z /180) * pi
25.   Array ke-i.Z = ( jari * 0.5 ) * sin (sudut)
26.   End for
27.   For i = 16 to 22
28.   ulat ke-i.X = ulat ke-i.X * panjang
29.   Sudut = (ulat ke-i.Y / 180) * pi
30.   Array ke-i.Y = ( jari * 0.7 ) * cos ( sudut )
31.   Sudut = (array ke-i.Z /180) * pi
32.   Array ke-i.Z = ( jari * 0.7 ) * sin (sudut)
33.   End for
34.   For i = 23 to 29
35.   ulat ke-i.X = ulat ke-i.X * panjang
```

```
36.    Sudut = (ulat ke-i.Y / 180) * pi
37.    Array ke-i.Y = ( jari * 0.8 ) * cos ( sudut )
38.    Sudut = (array ke-i.Z /180) * pi
39.    Array ke-i.Z = ( jari * 0.8 ) * sin (sudut)
40.    End for
41.    For i = 30 to 36
42.    ulat ke-i.X = ulat ke-i.X * panjang
43.    Sudut = (ulat ke-i.Y / 180) * pi
44.    Array ke-i.Y = ( jari * 0.9 ) * cos ( sudut )
45.    Sudut = (array ke-i.Z /180) * pi
46.    Array ke-i.Z = ( jari * 0.9 ) * sin (sudut)
47.    End for
48.    For i = 37 to 43
49.    ulat ke-i.X = ulat ke-i.X * panjang
50.    Sudut = (ulat ke-i.Y / 180) * pi
51.    Array ke-i.Y = ( jari * 0.7 ) * cos ( sudut )
52.    Sudut = (array ke-i.Z /180) * pi
53.    Array ke-i.Z = ( jari * 0.7 ) * sin (sudut)
54.    End for
55.    For i = 44 to 50
56.    ulat ke-i.X = ulat ke-i.X * panjang
57.    Sudut = (ulat ke-i.Y / 180) * pi
58.    Array ke-i.Y = ( jari * 0.5 ) * cos ( sudut )
59.    Sudut = (array ke-i.Z /180) * pi
60.    Array ke-i.Z = ( jari * 0.5 ) * sin (sudut)
61.    End for
62.    Selesai
```

Syntak pemrograman fungsi ulat dapat dilihat pada lampiran 8.

4.2.6 Teksturing Ulat

Untuk teksturing ulat menggunakan metode teksturing tetapi dengan 10 sisi polygon, hasil perhitungan arraynya dapat dilihat pada lampiran 9. Dari nilai array yang didapatkan pada table lampiran 9, maka kita dapat membuat fungsi untuk menggambarkan titik koordinat tersebut pada program. Berikut *pseudocode*nya

```
1.     Mulai
2.     Cek file = benar
3.     Buka file ulatTK.txt
```

```
4.    If file tidak ada
5.    Cetak "error buka file"
6.    While data file masih
7.    File baca ulatTK ke-i.X >> ulatTK ke-i.Y
8.    For i = 0 to 21
9.    ulatTK ke-i.X = ulatTK ke-i.X / 10
10.   End for
11.   Selesai
```

Syntak pemrograman teksturing ulat dapat dilihat pada lampiran 10.

4.2.7 Kepompong

Hasil perhitungan nilai array untuk kepompong dapat dilihat pada lampiran 11. Dari nilai array yang didapatkan pada table lampiran 11, maka kita dapat membuat fungsi untuk menggambarkan titik koordinat tersebut pada program. Berikut *pseudocode* nya

```
1.    Mulai
2.    Cek file = benar
3.    Buka file kepompong.txt
4.    If file tidak ada
5.    Cetak "error buka file"
6.    While data file masih
7.    File baca kepompong ke-i.X >> kepompong ke-i.Y >> kepompong ke-i.Z
8.    For i = 0 to 73
9.    kepompong ke-i.X = kepompong ke-i.X
10.   kepompong ke-i.Y = kepompong ke-i.Y * tinggi
11.   kepompong ke-i.Z = kepompong  ke-i.Z
12.   End for
13.   For i = 1 to 12
14.   Sudut = (kepompong ke-i.X / 180) * pi
15.   kepompong ke-i.X = ( jari * 0.8 ) * cos(sudut)
16.   kepompong ke-i.Y = kepompong ke-i.Y * tinggi
17.   Sudut = (kepompong ke-i.Z /180) * pi
18.   kepompong ke-i.Z = ( jari * 0.8 ) * sin (sudut)
19.   End for
20.   For i = 13 to 24
21.   Sudut = (kepompong ke-i.X / 180) * pi
22.   kepompong ke-i.X = ( jari * 0.7 ) * cos(sudut)
23.   kepompong ke-i.Y = kepompong ke-i.Y * tinggi
24.   Sudut = (kepompong ke-i.Z /180) * pi
```

```
25.    kepompong ke-i.Z = ( jari * 0.7 ) * sin (sudut)
26.    End for
27.    For i = 25 to 36
28.    Sudut = (kepompong ke-i.X / 180) * pi
29.    kepompong ke-i.X = ( jari * 0.9 ) * cos(sudut)
30.    kepompong ke-i.Y = kepompong ke-i.Y * tinggi
31.    Sudut = (kepompong ke-i.Z /180) * pi
32.    kepompong ke-i.Z = ( jari * 0.9 ) * sin (sudut)
33.    End for
34.    For i = 37 to 48
35.    Sudut = (kepompong ke-i.X / 180) * pi
36.    kepompong ke-i.X = ( jari * 0.8 ) * cos(sudut)
37.    kepompong ke-i.Y = kepompong ke-i.Y * tinggi
38.    Sudut = (kepompong ke-i.Z /180) * pi
39.    kepompong ke-i.Z = ( jari * 0.8 ) * sin (sudut)
40.    End for
41.    For i = 49 to 60
42.    Sudut = (kepompong ke-i.X / 180) * pi
43.    kepompong ke-i.X = jari * cos(sudut)
44.    kepompong ke-i.Y = kepompong ke-i.Y * tinggi
45.    Sudut = (kepompong ke-i.Z /180) * pi
46.    kepompong ke-i.Z = jari * sin (sudut)
47.    End for
48.    For i = 61 to 72
49.    Sudut = (kepompong ke-i.X / 180) * pi
50.    kepompong ke-i.X = ( jari * 0.6 ) * cos(sudut)
51.    kepompong ke-i.Y = kepompong ke-i.Y * tinggi
52.    Sudut = (kepompong ke-i.Z /180) * pi
53.    kepompong ke-i.Z = ( jari * 0.6 ) * sin (sudut)
54.    End for
55.    Selesai
```

Syntak pemrograman fungsi kepompong dapat dilihat pada lampiran 12. Untuk teksturing kepompong, fungsi yang digunakan menggunakan fungsi **4.2.2 Teksturing Tabung.**

4.2.8 Badan Kupu-kupu

Hasil perhitungan nilai array untuk kupu-kupu dapat dilihat pada lampiran 13. Dari nilai array yang didapatkan pada table lampiran 13, maka kita dapat

membuat fungsi untuk menggambarkan titik koordinat tersebut pada program. Berikut *pseudocode* nya

```
1.    Mulai
2.    Cek file = benar
3.    Buka file kupu.txt
4.    If file tidak ada
5.    Cetak "error buka file"
6.    While data file masih
7.    File baca kupu ke-i.X >> kupu ke-i.Y >> kupu ke-i.Z
8.    For i = 0 & 193
9.    Kupu ke-i.X = kupu ke-i.X
10.   Kupu ke-i.Y = kupu ke-i.Y * tinggi
11.   Kupu ke-i.Z = kupu  ke-i.Z
12.   End for
13.   For i = 1 to 12
14.   Sudut = (kupu ke-i.X / 180) * pi
15.   kupu ke-i.X = ( jari * 0.35 ) * cos(sudut)
16.   kupu ke-i.Y = kupu ke-i.Y * tinggi
17.   Sudut = (kupu ke-i.Z /180) * pi
18.   kupu ke-i.Z = ( jari * 0.35 ) * sin (sudut)
19.   End for
20.   For i = 13 to 24
21.   Sudut = (kupu ke-i.X / 180) * pi
22.   kupu ke-i.X = ( jari * 0.7 ) * cos(sudut)
23.   kupu ke-i.Y = kupu ke-i.Y * tinggi
24.   Sudut = (kupu ke-i.Z /180) * pi
25.   kupu ke-i.Z = ( jari * 0.7 ) * sin (sudut)
26.   End for
27.   For i = 25 to 36
28.   Sudut = (kupu ke-i.X / 180) * pi
29.   kupu ke-i.X = ( jari * 0.85 ) * cos(sudut)
30.   kupu ke-i.Y = kupu ke-i.Y * tinggi
31.   Sudut = (kupu ke-i.Z /180) * pi
32.   kupu ke-i.Z = ( jari * 0.85 ) * sin (sudut)
33.   End for
34.   For i = 37 to 48
35.   Sudut = (kupu ke-i.X / 180) * pi
36.   kupu ke-i.X = ( jari * 0.7 ) * cos(sudut)
37.   kupu ke-i.Y = kupu ke-i.Y * tinggi
38.   Sudut = (kupu ke-i.Z /180) * pi
39.   kupu ke-i.Z = ( jari * 0.7 ) * sin (sudut)
```

```
40.    End for
41.    For i = 49 to 60
42.    Sudut = (kupu ke-i.X / 180) * pi
43.    kupu ke-i.X = ( jari * 0.35 ) * cos(sudut)
44.    kupu ke-i.Y = kupu ke-i.Y * tinggi
45.    Sudut = (kupu ke-i.Z /180) * pi
46.    kupu ke-i.Z = ( jari * 0.35 ) * sin (sudut)
47.    End for
48.    For i = 61 to 72
49.    Sudut = (kupu ke-i.X / 180) * pi
50.    kupu ke-i.X = ( jari * 0.8 ) * cos(sudut)
51.    kupu ke-i.Y = kupu ke-i.Y * tinggi
52.    Sudut = (kupu ke-i.Z /180) * pi
53.    kupu ke-i.Z = ( jari * 0.8 ) * sin (sudut)
54.    End for
55.    For i = 73 to 84
56.    Sudut = (kupu ke-i.X / 180) * pi
57.    kupu ke-i.X = jari * cos(sudut)
58.    kupu ke-i.Y = kupu ke-i.Y * tinggi
59.    Sudut = (kupu ke-i.Z /180) * pi
60.    kupu ke-i.Z = jari * sin (sudut)
61.    End for
62.    For i = 85 to 96
63.    Sudut = (kupu ke-i.X / 180) * pi
64.    kupu ke-i.X = jari * cos(sudut)
65.    kupu ke-i.Y = kupu ke-i.Y * tinggi
66.    Sudut = (kupu ke-i.Z /180) * pi
67.    kupu ke-i.Z = jari * sin (sudut)
68.    End for
69.    For i = 97 to 108
70.    Sudut = (kupu ke-i.X / 180) * pi
71.    kupu ke-i.X = ( jari * 0.8 ) * cos(sudut)
72.    kupu ke-i.Y = kupu ke-i.Y * tinggi
73.    Sudut = (kupu ke-i.Z /180) * pi
74.    kupu ke-i.Z = ( jari * 0.8 ) * sin (sudut)
75.    End for
76.    For i = 109 to 120
77.    Sudut = (kupu ke-i.X / 180) * pi
78.    kupu ke-i.X = ( jari * 0.4 ) * cos(sudut)
79.    kupu ke-i.Y = kupu ke-i.Y * tinggi
80.    Sudut = (kupu ke-i.Z /180) * pi
```

```
81.    kupu ke-i.Z = ( jari * 0.4 ) * sin (sudut)
82.    End for
83.    For i = 121 to 132
84.    Sudut = (kupu ke-i.X / 180) * pi
85.    kupu ke-i.X = ( jari * 0.7 ) * cos(sudut)
86.    kupu ke-i.Y = kupu ke-i.Y * tinggi
87.    Sudut = (kupu ke-i.Z /180) * pi
88.    kupu ke-i.Z = ( jari * 0.7 ) * sin (sudut)
89.    End for
90.    For i = 133 to 144
91.    Sudut = (kupu ke-i.X / 180) * pi
92.    kupu ke-i.X = ( jari * 0.9 ) * cos(sudut)
93.    kupu ke-i.Y = kupu ke-i.Y * tinggi
94.    Sudut = (kupu ke-i.Z /180) * pi
95.    kupu ke-i.Z = ( jari * 0.9 ) * sin (sudut)
96.    End for
97.    For i = 145 to 156
98.    Sudut = (kupu ke-i.X / 180) * pi
99.    kupu ke-i.X = jari * cos(sudut)
100.   kupu ke-i.Y = kupu ke-i.Y * tinggi
101.   Sudut = (kupu ke-i.Z /180) * pi
102.   kupu ke-i.Z = jari * sin (sudut)
103.   End for
104.   For i = 157 to 168
105.   Sudut = (kupu ke-i.X / 180) * pi
106.   kupu ke-i.X = ( jari * 0.9 ) * cos(sudut)
107.   kupu ke-i.Y = kupu ke-i.Y * tinggi
108.   Sudut = (kupu ke-i.Z /180) * pi
109.   kupu ke-i.Z = ( jari * 0.9 ) * sin (sudut)
110.   End for
111.   For i = 169 to 180
112.   Sudut = (kupu ke-i.X / 180) * pi
113.   kupu ke-i.X = ( jari * 0.7 ) * cos(sudut)
114.   kupu ke-i.Y = kupu ke-i.Y * tinggi
115.   Sudut = (kupu ke-i.Z /180) * pi
116.   kupu ke-i.Z = ( jari * 0.7 ) * sin (sudut)
117.   End for
118.   For i = 181 to 192
119.   Sudut = (kupu ke-i.X / 180) * pi
120.   kupu ke-i.X = ( jari * 0.4 ) * cos(sudut)
121.   kupu ke-i.Y = kupu ke-i.Y * tinggi
```

```
122.   Sudut = (kupu ke-i.Z /180) * pi
123.   kupu ke-i.Z = ( jari * 0.4 ) * sin (sudut)
124.   End for
125.   Selesai
```

Syntak pemrograman fungsi kupu-kupu dapat dilihat pada lampiran 14. Untuk teksturing kupu, fungsi yang digunakan menggunakan fungsi **4.2.2 Teksturing Tabung.**

4.2.9 Sayap Kupu-kupu

Nilai titik koordinat yang sudah ditentukan disimpan dalam variabel global untuk nandti dipanggil didalam fungsi untuk menggambarkan sayap. Untuk menggambar sayap pada program ini kita membuat sayap kanan depan dan belakang berdasarkan titik koordinat yang sudah ditentukan. Untuk sayap bagian kiri dilakukan pencerminan untuk menjadikannya objek baru. *Pseudocode* untuk menggambar sayap kanan depan :

```
1.     Mulai
2.     Panggil tekstur koordinat pada variabel global
3.     Aktifkan tekstur bezier
4.     Mulai simpan matrix
5.     Rotasi -30^0 pada sumbu Y
6.     Ambil tekstur file
7.     Panggil koordinat bezier 1 pada variabel global
8.     Aktifkan bezier satu dimensi
9.     Aktifkan polygon mode
10.    For i = 0 to 30
11.    Panggil fungsi untuk menggambarkan posisi
12.    Selesai polygon mode
13.    Panggil koordinat bezier 2 pada variabel global
14.    Aktifkan bezier satu dimensi
15.    Aktifkan polygon mode
16.    For i = 0 to 30
17.    Panggil fungsi untuk menggambarkan posisi
18.    Selesai polygon mode
19.    Selesai simpan matrix
20.    Selesai
```

Untuk sayap kanan belakang, *pseudocode* nya sama dengan sayap kanan depan tetapi variabel global yang dipanggil adalah bezier 3 dan bezier 4.

Sayap kiri depan dan belakang dilakukan pencerminan dari sayap kanan depan dan belakang. *Pseudocode* untuk sayap kiri depan.

```
1.    Mulai
2.    Mulai simpan matrix
3.    Dilatasi pada –X, Y, dan Z
4.    Panggil fungsi sayap kanan depan
5.    Aktifkan fungsi blend
6.    Selesai simpan matrix
7.    Selesai
```

Untuk sayap kiri belakang, *pseudocode* nya sama dengan sayap kiri depan tetapi yang diambil fungsi sayap kanan belakang.

Syntak pemrograman sayap dapat dilihat pada lampiran 15.

4.2.10 Antena

Untuk menggambar antena, sama dengan menggambar pada sayap tetapi variabel global yang diambil adalah antena. Syntak pemrograman antena dapat dilihat pada lampiran 15.

4.2.11 Teksturing Bezier

Untuk teksturing pada bezier, titik koordinat yang disimpan adalah (0, 0) , (0, 1) , (1, 1) , (1, 0). Syntak pemanggilan fungsi teksturing bezier dapat dilihat pada lampiran 15.

4.2.12 Duri

Nilai array yang digunakan untuk duri mengambil dari nilai array pada kupu-kupu pada array ke-0 dan array ke-13 sampai array ke-24. *Pseudocode* unuk menggambarkan objek duri.

```
1.    Mulai
2.    Cek file = benar
3.    Buka file kupu.txt
4.    If file tidak ada
5.    Cetak "error buka file"
6.    While data file masih
```

```
7.    File baca duri ke-i.X >> duri ke-i.Y >> duri ke-i.Z
8.    For i = 0
9.    Kupu ke-i.X = kupu ke-i.X
10.   Kupu ke-i.Y = kupu ke-i.Y * tinggi
11.   Kupu ke-i.Z = kupu  ke-i.Z
12.   End for
13.   For i = 13 to 24
14.   Sudut = (duri ke-i.X / 180) * pi
15.   duri ke-i.X = ( jari * 0.7 ) * cos(sudut)
16.   duri ke-i.Y = duri ke-i.Y * tinggi
17.   Sudut = (duri ke-i.Z /180) * pi
18.   duri ke-i.Z = ( jari * 0.7 ) * sin (sudut)
19.   End for
20.   Selesai
```

Syntak pemrograman duri dapat dilihat pada lampiran 16.

4.2.13 Bola

Untuk menggambar bola, kita dapat langsung mengambil fungsi bawaan dari openGL yaitu **GLUquadric** untuk membuat *quadric* objek. Untuk teksturingnya kita juga dapat langsung memanggil fungsi yang tersedia yaitu **gluQuadricTexture**. Kemudian kita baru memanggil objek yang akan kita buat. Syntak pemrograman bola dapat dilihat pada lampiran 17.

4.2.14 Penggabungan Objek Pohon

Setelah objek pohon dibuat, untuk menampilkan bentuk pohon agar terlihat lebih menyerupai maka kita buat objek ranting dan tambahkan objek daun. Untuk membuat ranting, objek yang digunakan sama dengan objek pohon hanya saja nilai jari-jari dan tinggi yang berbeda. Berikut *Pseudocode*nya

```
1.    Mulai
2.    Mulai simpan matrix
3.    Panggil objek pohon kedua
4.    Geser sejauh 1 sumbu Y
5.    Rotasi -90^0 sumbu Y
6.    Rotasi -10^0 sumbu X
7.    Panggil objek daun
8.    Selesai simpan matrix
9.    Selesai
```

Setelah ranting dibuat, kita satukan dengan objek pohon pertama. *Pseudocode* nya :

```
1.    Mulai
2.    Dilatasikan 3 kali sumbu X, Y, Z
3.    Mulai simpan matrix
4.    Panggil objek pohon pertama
5.    Geser 1.3 sumbu X, 1 Y
6.    Rotasi -89^0 sumbu Z
7.    Panggil ranting
8.    Selesai simpan matrix
9.    Selesai
```

Syntak pemrograman penggabungan objek pohon dapat dilihat pada lampiran 18.

4.2.15 Mata Kupu-kupu

Mata dibuat dari objek bola yang di beri tekstur. Dan dicerminkan agar menjadi sepasang mata kupu-kupu. *Pseudocode* untuk mata kanan.

```
1.    Mulai
2.    Mulai simpan matrix
3.    Panggil tekstur mata
4.    Geser 0.15 sumbu X, 1.6 sumbu Y dan 0.2 sumbu Z
5.    Rotasi 30^0 sumbu Y
6.    Panggil bola
7.    Selesai simpan matrix
8.    Selesai
```

Untuk mata kiri kita cerminkan dari mata kanan. Untuk mencerminkan mata kiri sama dengan mencerminkan pada sayap. Setelah dicerminkan kita jadikan dalam satu objek agar memudahkan dalam menyatukan objek ke kupu-kupu. Syntak pemrograman dapat dilihat pada lampiran 17.

4.2.16 Penggabungan Objek Ulat

Setelah badan ulat dibuat, kita dapat menambahkan objek duri dan bola untuk kepala. Duri diletakkan di badan ulat pada bagian tengah dan samping. Untuk menempatkan posisinya kita dapat panggil beberapa kali objek duri dan

kita geser posisinya sesuai yang kita inginkan. Pertama kali kita buat deretan duri pada bagian tengah, *pseudocode* nya.

```
1.    Mulai
2.    Mulai simpan matrix
3.    Dilatasi 1.1 sumbu X, Y, Z
4.    Geser 0.32 sumbu X, -1.54 sumbu Y
5.    Panggil objek duri
6.    Selesai simpan matrix
7.    Mulai simpan matrix
8.    Dilatasi 1.1 sumbu X, Y, Z
9.    Geser 0.13 sumbu X, -1.5 sumbu Y
10.   Panggil objek duri
11.   Selesai simpan matrix
12.   Mulai simpan matrix
13.   Dilatasi 1.1 sumbu X, Y, Z
14.   Geser -0.13 sumbu X, -1.53 sumbu Y
15.   Panggil objek duri
16.   Selesai simpan matrix
17.   Mulai simpan matrix
18.   Dilatasi 1.1 sumbu X, Y, Z
19.   Geser -0.36 sumbu X, -1.63 sumbu Y
20.   Panggil objek duri
21.   Selesai simpan matrix
22.   Mulai simpan matrix
23.   Dilatasi 1.1 sumbu X, Y, Z
24.   Geser -0.45 sumbu X, -1.75 sumbu Y
25.   Panggil objek duri
26.   Selesai simpan matrix
27.   Selesai
```

Setelah objek duri bagian tengah dibuat, kita buat deretan duri bagian kanan. *Pseudocode* nya.

```
1.    Mulai
2.    Mulai simpan matrix
3.    Rotasi 40^0 sumbu X
4.    Geser 0.39 sumbu X, -1.52 sumbu Y, 0.03 sumbu Z
5.    Panggil objek duri
6.    Selesai simpan matrix
7.    Mulai simpan matrix
8.    Rotasi 40^0 sumbu X
```

```
9.    Geser 0.15 sumbu X, -1.48 sumbu Y, 0.03 sumbu Z
10.   Panggil objek duri
11.   Selesai simpan matrix
12.   Mulai simpan matrix
13.   Rotasi 40^0 sumbu X
14.   Geser -0.15 sumbu X, -1.51 sumbu Y, 0.03 sumbu Z
15.   Panggil objek duri
16.   Selesai simpan matrix
17.   Mulai simpan matrix
18.   Rotasi 40^0 sumbu X
19.   Geser -0.39 sumbu X, -1.61 sumbu Y, 0.03 sumbu Z
20.   Panggil objek duri
21.   Selesai simpan matrix
22.   Selesai
```

Setelah dibuat, kita cerminkan duri kanan untuk menjadi duri sebelah kiri. Kemudian kita buat menjadi satu objek.

Setelah objek duri selesai kita buat, selanjutnya kita buat kepala ulat dari objek bola. *Pseudocode* nya.

```
1.    Mulai
2.    Mulai simpan matrix
3.    Panggil tekstur kepala ulat
4.    Geser 0.65 sumbu X, -0.2 sumbu Y
5.    Panggil bola
6.    Selesai simpan matrix
7.    Selesai
```

Kemudian kita gabungkan dengan objek badan ulat dan duri

```
1.    Mulai
2.    Mulai simpan matrix
3.    Rotasi 180 sumbu X
4.    Panggil objek badan ulat
5.    Panggil objek kepala ulat
6.    Selesai simpan matrix
7.    Panggil objek duri yang sudah digabung
8.    Selesai
```

Syntak pemrograman penggabungan objek ulat dapat dilihat pada lampiran 19.

4.2.17 Penggabungan Objek Kupu-kupu

Untuk membuat kupu-kupu sesuai dengan yang kita inginkan maka kita satukan objek badan kupu-kupu, dan mata. Kita tambahkan objek bola sebagai mulut. *Pseudocode* nya.

```
1.    Mulai
2.    Mulai simpan matrix
3.    Geser -0.8 sumbu Y, -0.4 sumbu Z
4.    Rotasi 90^0 sumbu Y
5.    Panggil objek badan kupu
6.    Mulai simpan matrix
7.    Panggil tekstur untuk mulut
8.    Geser 0.3 sumbu X, 1.4 sumbu  sumbu Y
9.    Rotasi 90^0 sumbu Y
10.   Dilatasi 0.6 sumbu X, Y, Z
11.   Panggil bola untuk mulut
12.   Selesai simpan matrix
13.   Mulai simpan matrix
14.   Geser 2.0 sumbu Y
15.   Panggil objek antena
16.   Selesai simpan matrix
17.   Panggil mata
18.   Selesai simpan matrix
19.   Selesai
```

Syntak pemrograman penggabungan objek kupu-kupu dapat dilihat pada lampiran 20.

4.2.18 Animasi Sayap

Setelah sayap selesai dibuat, dan titik awal dan akhir untuk animasi sudah ditentukan, maka kita buat untuk animasi sayap mengepak. *Pseudocode* nya.

```
1.    Mulai
2.    Kepak pertama = ( 1 – p ) * -30 + p * 50
3.    Kepak kedua = ( 1 – p ) * -20 + p * 50
4.    Kepak ketiga = ( 1 – p ) * 30 + p * -50
5.    Kepak keempat = ( 1 – p ) * 20 + p * -50
6.    Panggil objek kupu
7.    Mulai simpan matrix
8.    Rotasi kepak pertama sumbu Y
9.    Panggil sayap kanan depan
```

```
10.    Selesai simpan matrix
11.    Mulai simpan matrix
12.    Rotasi kepak kedua sumbu Y
13.    Panggil sayap kanan belakang
14.    Selesai simpan matrix
15.    Mulai simpan matrix
16.    Rotasi kepak ketiga sumbu Y
17.    Panggil sayap kiri depan
18.    Selesai simpan matrix
19.    Mulai simpan matrix
20.    Rotasi kepak keempat sumbu Y
21.    Panggil sayap kiri belakang
22.    Selesai simpan matrix
23.    Selesai
```

Kita juga membuat animasi sayap saat kupu-kupu keluar dari kepompong. Cara membuatnya sama dengan animasi sayap mengepak, hanya perbedaannya pada nilai P_0 dan P_1 nya dan sudut animasinya yang berbeda. Setelah fungsi animasi sayap mengepak dibuat, maka untuk menjalankannya kita buat fungsi untuk mengatur pergerakannya. *Pseudocode* nya.

```
1.     Mulai
2.     If isAnimasi
3.     T++
4.     S++
5.     If global = 0
6.     P += 0.01
7.     If p >= 1then global = 1
8.     Else if global = 1
9.     P -= 0.01
10.    If p <= 0.01 then global = 0
11.    Timer fungsi ( periode animasi, animasi, 1)
12.    Redisplay
13.    Selesai
```

Karena animasi sayap mengepak ini menggunakan fungsi papan ketik untuk menjalankan dan menghentikan animasi, maka kita buat fungsi kontrol animasinya. *Pseudocode* nya.

```
1.     Mulai
2.     Switch
```

```
3.     Case ‘spasi’
4.     If ( isAnimasi ) isAnimasi = 0
5.     Else isAnimasi =1
6.     Redisplay
7.     Break
8.     Case ‘delete’
9.     If ( isAnimasi ) isAnimasi = 0
10.    T = 0
11.    Redisplay
12.    Break
13.    Default
14.    Break
15.    Selesai
```

Syntak pemrograman animasi kupu-kupu dapat dilihat pada lampiran 21.

4.2.19 Animasi Metamorfosis

Setelah semua objek selesai dibuat maka kita buat alur animasinya. Untuk animasinya kita menggunakan parameter t sebagai penentu waktu animasi dan ditampilkan *frame by frame*. Kita membuat 8 kelompok untuk menampilkan animasinya, dengan masing-masing kelompok objek yang ditampilkan dapat berganti. *Pseudocode* nya.

Kelompok 1 :

```
1.     Mulai
2.     If ( t > 5 & t <= 800 )
3.     Mulai simpan matrix
4.     Geser -2 sumbu X, -3.27 sumbu Y
5.     Panggil objek pohon
6.     Selesai simpan matrix
7.     Mulai simpan matrix
8.     Geser 1*t/200 sumbu X
9.     Panggil objek ulat
10.    Selesai simpan matrix
11.    Selesai
```

Kelompok 2 :

```
1.     Mulai
2.     If ( t > 800 & t <= 1600 )
3.     Mulai simpan matrix
```

```
4.    Geser -2 sumbu X, -3.27 sumbu Y
5.    Panggil objek pohon
6.    Selesai simpan matrix
7.    S = ( t – 800 )
8.    Mulai simpan matrix
9.    Geser -s/200 sumbu X
10.   Geser 4 sumbu X
11.   Rotasi 180 sumbu Y
12.   Dilatasi 1.1 sumbu X, Y, Z
13.   Panggil objek ulat
14.   Selesai simpan matrix
15.   Selesai
```

Kelompok 3 :

```
1.    Mulai
2.    If ( t > 1600 & t <= 2400 )
3.    Mulai simpan matrix
4.    Geser -2 sumbu X, -3.27 sumbu Y
5.    Panggil objek pohon
6.    Selesai simpan matrix
7.    S = ( t – 1600 )
8.    Mulai simpan matrix
9.    Geser s/200 sumbu X
10.   Dilatasi 1.2 sumbu X, Y, Z
11.   Panggil objek ulat
12.   Selesai simpan matrix
13.   Selesai
```

Kelompok 4 :

```
1.    Mulai
2.    If ( t > 2400 & t <= 3200 )
3.    Mulai simpan matrix
4.    Geser -2 sumbu X, -3.27 sumbu Y
5.    Panggil objek pohon
6.    Selesai simpan matrix
7.    S = ( t – 2400 )
8.    Mulai simpan matrix
9.    Geser -s/200 sumbu X
10.   Geser 4 sumbu X
11.   Rotasi 180 sumbu Y
12.   Dilatasi 1.3 sumbu X, Y, Z
```

```
13.    Panggil objek ulat
14.    Selesai simpan matrix
15.    Selesai
```

Kelompok 5 :

```
1.     Mulai
2.     If ( t > 3200 & t <= 4000 )
3.     Mulai simpan matrix
4.     Geser -2 sumbu X, -3.27 sumbu Y
5.     Panggil objek pohon
6.     Selesai simpan matrix
7.     S = ( t – 3200 )
8.     Mulai simpan matrix
9.     Geser s/200 sumbu X
10.    Dilatasi 1.4 sumbu X, Y, Z
11.    Panggil objek ulat
12.    Selesai simpan matrix
13.    Selesai
```

Kelompok 6 :

```
1.     Mulai
2.     If ( t > 4000 & t <= 4800 )
3.     Mulai simpan matrix
4.     Geser -2 sumbu X, -3.27 sumbu Y
5.     Panggil objek pohon
6.     Selesai simpan matrix
7.     Mulai simpan matrix
8.     Geser 4.5 sumbu X, -1.3 sumbu Y
9.     Rotasi t sumbu Y
10.    Panggil objek kepompong
11.    Selesai simpan matrix
12.    Selesai
```

Kelompok 7 :

```
1.     Mulai
2.     If ( t > 4800 & t <= 6400 )
3.     S = ( t – 4000 )
4.     Mulai simpan matrix
5.     Geser -s/100 sumbu X, s/200 sumbu Y
6.     Geser 6 sumbu X, -7.15 sumbu Y
7.     Panggil objek pohon
```

```
8.    Selesai simpan matrix
9.    Mulai simpan matrix
10.   Geser -s/100 sumbu X, s/175 sumbu Y
11.   Geser 12.5 sumbu X, -5.75 sumbu Y
12.   Dilatasi 800/s sumbu X, Y, Z
13.   Rotasi t sumbu Y
14.   Panggil objek kepompong
15.   Selesai simpan matrix
16.   If ( t > 4800 & t <= 5000 )
17.   S = ( t – 4700 )
18.   Mulai simpan matrix
19.   Geser s/200 sumbu X, -s/200 sumbu Y
20.   Geser 3.8 sumbu X, -0.2 sumbu Y
21.   Dilatasi s/600 sumbu X, Y, Z
22.   Rotasi 75 sumbu Y
23.   Panggil animasi sayap
24.   Selesai simpan matrix
25.   If ( t > 5000 )
26.   Mulai simpan matrix
27.   Geser 5.4 sumbu X, -1.7 sumbu Y
28.   Dilatasi 0.5 sumbu X, Y, Z
29.   Rotasi 75 sumbu Y
30.   Panggil animasi kupu
31.   Selesai simpan matrix
32.   Selesai
```

Kelompok 8 :

```
1.    Mulai
2.    If ( t > 6400 & t <= 8000 )
3.    S = ( t – 6400 )
4.    Mulai simpan matrix
5.    Geser -s/150 sumbu X, s/200 sumbu Y
6.    Geser 5.5 sumbu X, -1.5 sumbu Y
7.    Dilatasi 0.5 sumbu X, Y, Z
8.    Rotasi -t/200 sumbu Y
9.    Rotasi 100 sumbu Y
10.   Panggil animasi kupu
11.   Selesai simpan matrix
12.   Else if ( t > 8001 )
13.   Tutup program
14.   Redisplay
```

```
15.    Selesai
```

Syntak pemrograman animasi metamorfosis dapat dilihat pada lampiran 22.

4.2.20 Menampilkan huruf

Terakhir kita buat kalimat untuk memberikan keterangan pada animasi. Kita menggunakan cara *stroke* denagan jenis huruf Roman. Untuk dapat menampilkan huruf kita mengambil fungsi yang tersedia dari OpenGL. Kalimat yang kita tampilkan kita bagi juga kedalam 8 kelompok animasi sehingga kalimat yang keluar akan terlihat seperti kalimat pada film. Kalimat yang ditampilkan sudah kita buat pada papan alur cerita.

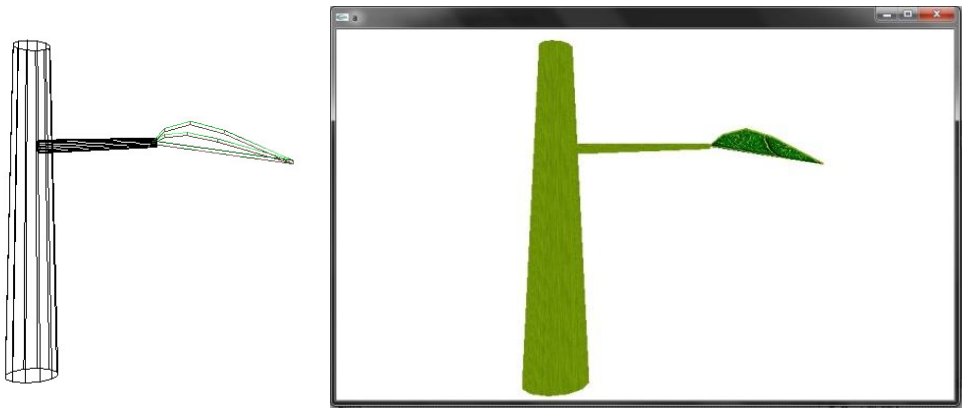
Syntak pemrograman huruf dapat dilihat pada lampiran 22.

4.3 Pembahasan

Pembahasan yang dilakukan yaitu dengan melakukan pengujian sketsa dengan hasil pembuatan objek dengan OpenGL. Perbandingan sketsa dengan hasil aplikasi:

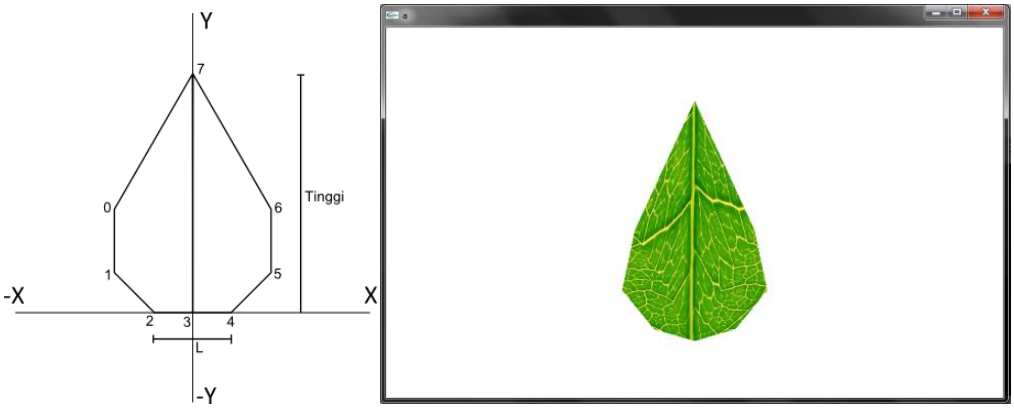
4.3.1 Perbandingan Objek Pohon

Dibawah ini akan ditampilkan ulang sketsa dan hasil dari pemodelan dengan OpenGL.



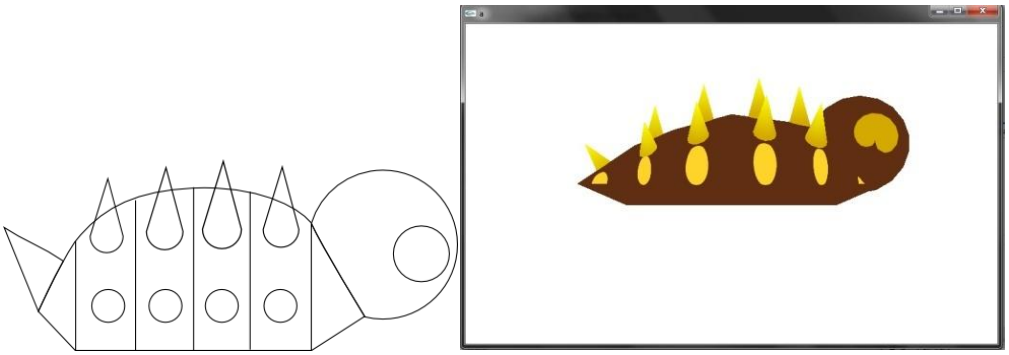
Gambar 4. 1 Perbandingan Objek Pohon

4.3.2 Perbandingan Objek Daun



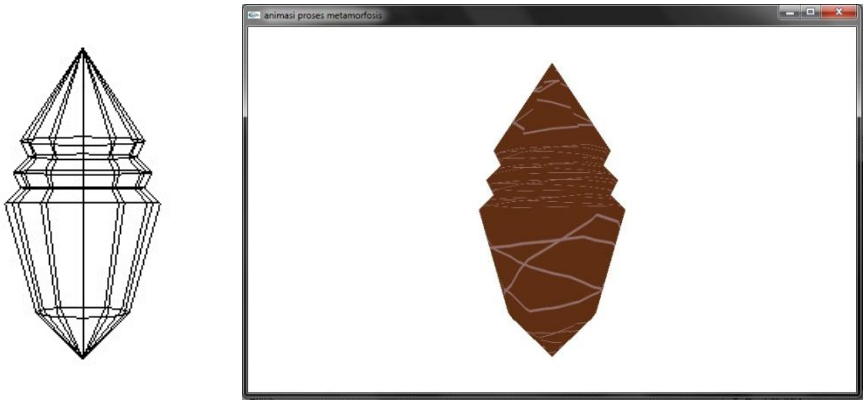
Gambar 4. 2 Perbandingan Objek Daun

4.3.3 Perbandingan Objek Ulat



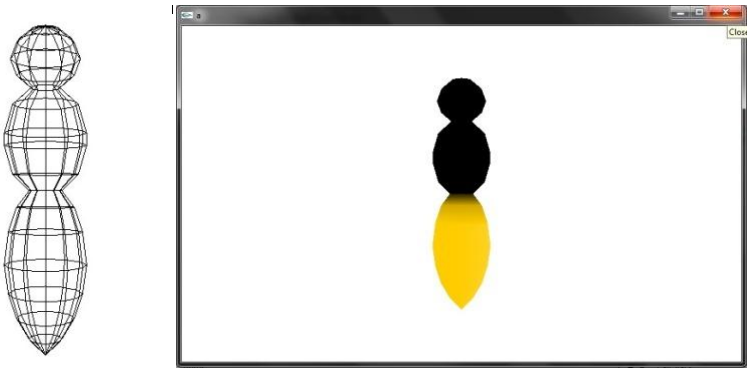
Gambar 4. 3 Perbandingan Objek Ulat

4.3.4 Perbandingan Objek Kepompong



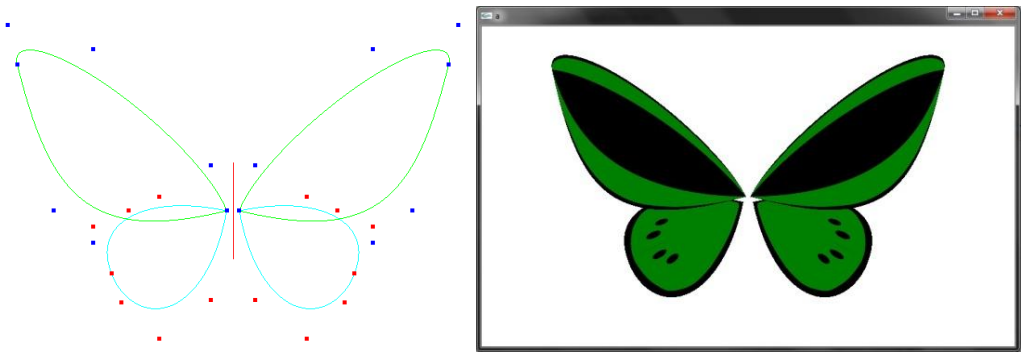
Gambar 4. 4 Perbandingan Objek Kepompong

4.3.5 Perbandingan Badan Kupu-kupu



Gambar 4. 5 Perbandingan Objek Kupu-kupu

4.3.6 Perbandingan Sayap



Gambar 4. 6 Perbandingan Sayap

4.3.7 Tampilan Hasil Penggabungan Objek

Hasil penggabungan objek dalam animasi terbagi menjadi

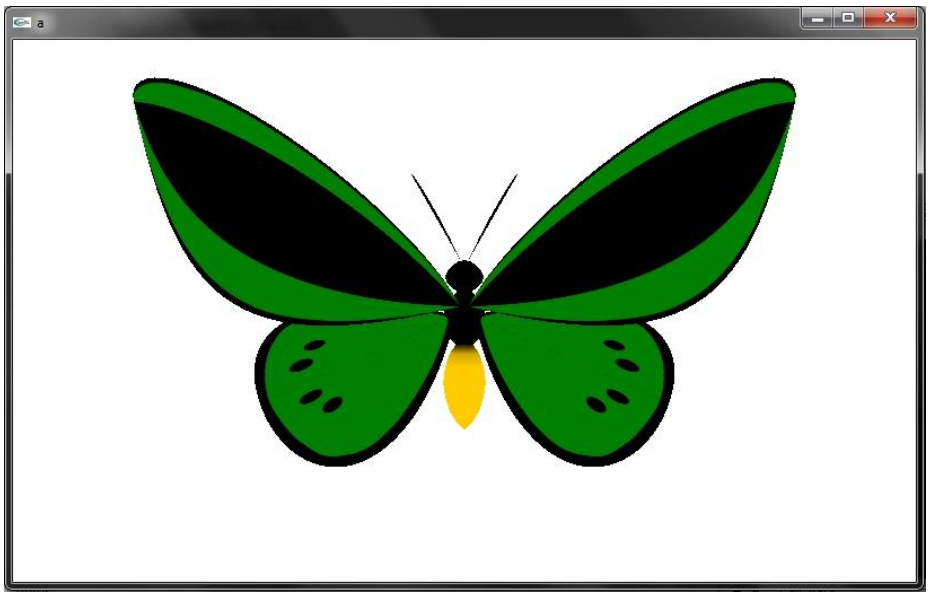
a. Pohon dan Ulat

Gambar 4. 7 Penggabungan Pohon dan Ulat

b. Pohon dan Kepompong

Gambar 4. 8 Penggabungan Pohon dan Kepompong

c. Kupu-kupu



Gambar 4. 9 Penggabungan Objek Kupu-kupu

4.3.8 Tabel Perbandingan Rancangan Objek

Tabel 4. 1 Perbandingan Rancangan Objek

Rancangan	Hasil	Waktu Rendering
Objek pohon	Sesuai dengan rancangan	6.0 detik
Objek daun	Sesuai dengan rancangan	6.0 detik
Objek ulat	Sesuai dengan rancangan	6.1 detik
Objek kepompong	Sesuai dengan rancangan	5.9 detik
Objek badan kupu-kupu	Sesuai dengan rancangan	5.9 detik
Objek sayap	Sesuai dengan rancangan	5.8 detik

4.4 Kelebihan Aplikasi

Kelebihan dari animasi ini adalah sebagai berikut :

1. Objek yang dibuat sudah sesuai dengan sketsa.
2. Objek kupu-kupu yang dibuat mengacu pada jenis kupu-kupu sayap burung asli Indonesia bagian timur.
3. Pemodelan objek menggunakan fungsi matematika dan garis Bezier.
4. Objek yang ditampilkan sudah diberikan tekstur.
5. Animasi ini sudah diberikan teks penjelasan alur metamorphosis.

6. Proses dan waktu untuk merendering aplikasi cukup ringan dan cepat kurang lebih 6 detik.

4.5 Kekurangan Aplikasi

Kekurangan dari animasi ini adalah sebagai berikut :

1. Sayap animasi dijalankan ketika objek diputar pada salah satu sumbu, tulisan masih ikut berputar.
2. Objek pada animasi ini masih menggunakan primitif objek pada OpenGL, belum menggunakan cara memanggil file objek dari perangkat lunak seperti 3DS MAX, Blender sehingga membutuhkan waktu untuk memahaminya.
3. Interaksi dengan user masih terbatas.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pengujian perbandingan objek pada animasi ini, maka kesimpulan yang didapat adalah sebagai berikut :

1. Animasi metamorfosis dibuat untuk memudahkan memahami tentang proses metamorfosis pada kupu-kupu.
2. Animasi ini dapat menampilkan polygon-polygon pembentuk objek dengan cukup baik.
3. Pemanfaatan rumus matematika sebagai dasar pembentuk objek dan penentu alur animasi dapat meminimalisir kesalahan bentuk yang diiinginkan.

5.2 Saran

Untuk mengembangkan animasi ini, dapat ditambahkan :

1. Animasi dapat dikembangkan untuk membuat animasi dengan model dan cerita yang berbeda.
2. Objek yang digunakan bisa menggunakan objek dari perangkat lunak yang tersedia dengan format file yang sudah OpenGL dukung seperti 3DS MAX dan Blender.
3. Animasi ini dapat dikembangkan kedalam proses pembuatan game.

DAFTAR PUSTAKA

Guha, Sumanta. (2011). *Computer Graphics Through OpenGL From Theory to Experiments.* New York : CRC Press.

Heriady. (2007). *Pemrograman Grafik 3D Menggunakan C dan OpenGL.* Yogyakarta : Graha Ilmu.

Lengyel, Eric. (2004). *Matematics for 3D Game Programming and Computer Graphics.* Hingham, Massachusetts : Charles River Media, INC.

Allain, Alex (1998). *Turorial C++*. Diakses tanggal 20 February 2012, dari http://www.cprogramming.com/tutorial.html

Anugrahjuni. (2011). *Metamorfosis kupu-kupu.* Diakses tanggal 20 April 2014, dari anugrahjuni.wordpress.com/biologi-in/metamorfosis-kupu-kupu/

Artawan, Rudi. (2010). *Kupu-kupu terindah di dunia yang ada di Indonesia.* Blogspot. Diakses tanggal 31 Maret 2012, dari http://biologinfo.blogspot.com/2010/05/tidak-salah-jika-ada-yang-mengatakan.html

Estiara, Doddy. (2011). *Anatomi Kupu-kupu.* Blogspot. Diakses tanggal 10 September 2012, dari http://jujujitu.blogspot.com/2011/07/anatomi-kupu-kupu.html

Silicon Graphic OpenGL. (1992). *The Official Guide to Learning OpenGL, Version 1.1.* Diakses tanggal 23 Agustus 2012, dari glprogramming.com/red/

Wikipedia. (2001). *Bezier Curve.* Diakses tanggal 24 April 2014, dari http://en.wikipedia.org/wiki/Bézier_curve (2014)

Wikipedia. (2001). *Storyboard*. Diakses tanggal 20 february 2014, dari http://en.wikipedia.org/wiki/Storyboard

http://www.scribd.com/doc/74599978/26/Bola (2006)

http://rosyidah-binti.blogspot.com/2013/04/opengl-glut.html

https://www.opengl.org/

http://www.swiftless.com/opengltuts.html