**Laporan Grafika Komputer**

**Oleh :**

Marsellino Vlyzer Sanfranstama / C14200076

Aprilio Navigan P / C14220327

Rafael Alexander V H / C14220310

**PROGRAM STUDI INFORMATIKA**

****

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS KRISTEN PETRA**

**SURABAYA**

**2025**

Aplikasi ini adalah **simulator evolusi Pokemon berbasis WebGL 3D** yang menampilkan transformasi karakter PokemAplikasi ini adalah **simulator evolusi Pokemon berbasis WebGL 3D** yang menampilkan transformasi karakter Pokemon melalui tiga tahap evolusi (Abra → Kadabra → Alakazam). Pengguna dapat melihat animasi evolusi yang dramatis dengan efek visual yang kompleks, serta berinteraksi dengan kamera untuk melihat model dari berbagai sudut.

**Fitur Utama:**

* 3 model Pokemon 3D yang dapat berevolusi secara real-time
* Animasi evolusi dengan efek cahaya psikis (psychic glow)
* Kontrol kamera 360° interaktif (drag mouse + WASD)
* Transformasi geometri lengkap (5 jenis transformasi)
* Model hierarchical dengan animasi bagian tubuh independen

## **1. APA YANG DILAKUKAN APLIKASI INI?**

### **1.1 Konsep Aplikasi**

Aplikasi ini mensimulasikan **proses evolusi Pokemon** dalam tampilan 3D interaktif. Ketika pengguna menekan tombol evolusi, Pokemon akan mengalami transformasi visual yang dramatis:

1. **Pokemon melayang ke atas** dengan efek cahaya berputar
2. **Tubuh berputar-putar** di berbagai sumbu rotasi
3. **Ukuran membesar-mengecil** secara dinamis
4. **Bagian tubuh bergerak independen** (tangan melambai, kaki menekuk, kepala miring)
5. **Efek cahaya psikis** mengelilingi tubuh Pokemon
6. **Morphing bentuk** dari satu tahap evolusi ke tahap berikutnya

### **1.2 Tiga Tahap Pokemon**

#### **ABRA (Tahap 1)**

* Pokemon psikis dasar dengan bentuk sederhana
* Warna kuning/coklat keemasan
* Tidak memiliki sendok (spoon)
* Postur tidur/meditasi

#### **KADABRA (Tahap 2)**

* Evolusi dari Abra dengan tubuh lebih tinggi
* Memiliki **1 sendok** di tangan kanan
* Warna lebih gelap dengan aksen merah
* Postur lebih aktif dan waspada
* Ekor lebih panjang dengan zigzag

#### **ALAKAZAM (Tahap 3)**

* Bentuk final dengan kekuatan psikis maksimal
* Memiliki **2 sendok** (kedua tangan)
* Kumis/janggut panjang
* Otot lebih berkembang
* Aura psikis paling kuat

### **1.3 Interaksi Pengguna**

**Tombol Kontrol:**

* 3 tombol di bawah layar untuk memilih evolusi
* Klik tombol = Pokemon berubah dengan animasi dramatis

**Kontrol Kamera:**

* **Mouse Drag** = Putar kamera mengelilingi Pokemon
* **Scroll Mouse** = Zoom in/out
* **WASD** = Gerakkan titik fokus kamera
* **Q/E** = Naik/turun vertikal

## **2. IDE DAN KONSEP YANG DIIMPLEMENTASIKAN**

### **2.1 TRANSFORMASI GEOMETRI 3D (5 Jenis)**

Ini adalah **inti dari pembelajaran grafika komputer**. Semua transformasi diterapkan secara bersamaan saat evolusi.

#### **A. TRANSLASI (Perpindahan Posisi)**

**Konsep:** Menggerakkan objek dari satu titik ke titik lain dalam ruang 3D.

**Implementasi dalam aplikasi:**

* Pokemon **naik ke atas** (sumbu Y) saat mulai evolusi
* **Wobble effect** (goyang kiri-kanan pada sumbu X)
* **Turun perlahan** dengan efek bouncing setelah evolusi selesai

**Cara kerja:**

Posisi baru = Posisi lama + Vector perpindahan

Contoh: [x, y, z] + [0, 2, 0] = [x, y+2, z] (naik 2 unit)

**Visualisasi:**

↑ (Translasi Y = naik)

|

[Pokemon] → (Translasi X = kiri-kanan)

#### **B. ROTASI SUMBU STANDAR (X, Y, Z)**

**Konsep:** Memutar objek mengelilingi sumbu koordinat standar.

**Implementasi:**

**Rotasi Y (Yaw - Berputar Horizontal):**

* Pokemon **berputar seperti gasing** selama evolusi
* Kecepatan putaran meningkat di puncak evolusi
* Digunakan untuk efek "charging energy"

**Rotasi X (Pitch - Miring Depan-Belakang):**

* Pokemon **menunduk dan mendongak**
* Memberikan efek "gathering power"
* Angle: ±30 derajat

**Rotasi Z (Roll - Miring Kiri-Kanan):**

* Pokemon **bergoyang samping**
* Menciptakan gerakan dinamis
* Wobble effect sinkron dengan translasi X

**Visualisasi:**

Rotasi Y (Yaw)

↻

|

Rotasi Z ←[Pokemon]→ Rotasi X

(Roll) (Pitch)

#### **C. SCALING (Perubahan Ukuran)**

**Konsep:** Memperbesar atau memperkecil objek secara proporsional.

**Implementasi:**

1. **Shrink Phase:** Mengecil ke 85% (0.85x) - seperti "menarik napas"
2. **Expansion Phase:** Membesar hingga 125% (1.25x) - "ledakan energi"
3. **Normalize Phase:** Kembali ke ukuran normal (1.0x) dengan bouncing
4. **Final Size:** Setiap evolusi memiliki ukuran berbeda

**Timeline:**

1.0x → 0.85x → 1.25x → 1.0x

Normal Shrink Expand Settle

#### **D. ROTASI ARBITRARY AXIS (Sumbu Sembarang)**

**Konsep:** Memutar objek di sekitar sumbu yang bukan X/Y/Z standar, melainkan vektor bebas dalam ruang 3D.

**Implementasi:**

* Sumbu rotasi: **Vektor diagonal [1, 1, 0]**
* Menciptakan gerakan **spiral kompleks**
* Menggunakan **Rumus Rodrigues** untuk perhitungan matematis
* Pokemon berputar hingga **16 putaran penuh** (5760 derajat)

**Rumus Rodrigues:**

R = I + sin(θ)K + (1-cos(θ))K²

Dimana:

* I = Identity matrix
* θ = Sudut rotasi
* K = Skew-symmetric matrix dari axis vector

**Efek Visual:**

* Kombinasi rotasi diagonal X+Y
* Gerakan "corkscrew" (seperti membuka botol)
* Lebih kompleks dari rotasi biasa

#### **E. TRANSFORMASI HIERARCHICAL (Parent-Child)**

**Konsep:** Objek terdiri dari bagian-bagian yang saling terhubung. Ketika parent bergerak, child ikut bergerak, tapi child juga bisa bergerak sendiri relatif terhadap parent.

**Struktur Hierarki Pokemon:**

BODY (Parent Utama)

|

┌──────────────┼──────────────┐

| | |

HEAD ARMS LEGS

/ \ / \

LEFT RIGHT LEFT RIGHT

| | | |

HAND HAND KNEE KNEE

| |

SPOON SPOON (Kadabra/Alakazam)

**Implementasi Animasi per Bagian:**

1. **BODY (Tubuh):**
   * Tilt maju-mundur ±30°
   * Semua bagian lain mengikuti
2. **HEAD (Kepala):**
   * Miring kiri-kanan ±25°
   * Independen dari body tilt
3. **ARMS (Lengan):**
   * Rotasi keluar hingga 180° saat puncak evolusi
   * Lengan kiri dan kanan bergerak berbeda
4. **HANDS (Tangan):**
   * Gelombang (wave) ±35°
   * Frekuensi tinggi untuk efek "vibrating energy"
5. **SPOONS (Sendok):**
   * **Hanya ada di Kadabra/Alakazam**
   * Berputar hingga 16 putaran penuh
   * Independen dari gerakan tangan
6. **LEGS (Kaki):**
   * Stance lebar saat evolusi
7. **KNEES (Lutut):**
   * Menekuk ±40°
   * Simulasi gerakan menari/gathering energy

**Cara Kerja:**

Transform\_Child = Transform\_Parent × Transform\_Local\_Child

Contoh: Posisi spoon = Posisi body + Posisi arm + Posisi hand + Posisi spoon

### **2.2 PEMBUATAN MODEL 3D (Geometry Generation)**

**Konsep:** Membuat bentuk 3D dari formula matematika, bukan dari file 3D.

#### **Bentuk Dasar yang Dibuat:**

**A. SPHERE (Bola)**

* Untuk kepala, tubuh, mata
* Formula: Koordinat spherical

x = radius × sin(phi) × cos(theta)

y = radius × cos(phi)

z = radius × sin(phi) × sin(theta)

* Resolusi: 32 segment horizontal × 16 segment vertikal

**B. CYLINDER (Silinder)**

* Untuk lengan, kaki, ekor
* Body utama untuk Kadabra/Alakazam
* Variasi: Tapered cylinder (atas lebih kecil dari bawah)

**C. CONE (Kerucut)**

* Untuk telinga Abra
* Untuk detail ujung ekor

**D. TORUS (Donat - untuk sendok)**

* Gagang sendok
* Bentuk melingkar 3D

**E. Custom Shapes:**

* **Mustache (Kumis Alakazam):** Kurva bezier
* **Tail (Ekor):** Cylinder dengan segmentasi zigzag
* **Spoon Bowl:** Hemisphere yang dipipihkan

### **2.4 LIGHTING & SHADING (Pencahayaan)**

**Konsep:** Simulasi pencahayaan realistis menggunakan shader GLSL.

#### **Blinn-Phong Lighting Model**

Terdiri dari 3 komponen:

**1. AMBIENT (Cahaya Lingkungan)**

Ambient = Ambient\_Strength × Object\_Color

* Cahaya dasar yang merata
* Strength: 25% (+ 20% saat evolusi)
* Mencegah area terlalu gelap

**2. DIFFUSE (Cahaya Tersebar)**

Diffuse = max(dot(Normal, Light\_Direction), 0) × Light\_Color

* Cahaya yang tersebar saat mengenai permukaan
* Tergantung sudut permukaan terhadap cahaya
* Memberikan bentuk 3D yang jelas

**3. SPECULAR (Pantulan Kilap)**

Specular = pow(max(dot(View, Reflect), 0), Shininess) × Light\_Color

* Pantulan cahaya terang (highlight)
* Shininess: 32 (normal) hingga 128 (saat evolusi)
* Membuat material terlihat berkilau

**Formula Lengkap:**

Final\_Color = Ambient + Diffuse + Specular + Glow

#### **Dynamic Lighting Effects**

**Psychic Glow (Cahaya Psikis):**

* 3 layer glow dengan ukuran bertingkat
* Blend mode: ADDITIVE (cahaya bertumpuk)
* Intensitas berdenyut (pulsating)
* Warna: Pink-purple-blue gradient

**Metallic Surface:**

* Reflectivity meningkat saat evolusi
* Sendok memiliki metallic = 0.9 (sangat reflektif)
* Tubuh metallic = 0.3-0.5

### **2.5 SISTEM KAMERA INTERAKTIF**

**Konsep:** Kamera yang dapat dikontrol pengguna untuk melihat objek dari berbagai sudut.

#### **Spherical Camera System**

Kamera bergerak di permukaan bola imajiner mengelilingi Pokemon:

**Parameter:**

* **Theta (θ):** Sudut horizontal (0-360°)
* **Phi (φ):** Sudut vertikal (0-180°)
* **Distance (r):** Jarak dari target (3-30 unit)

**Konversi ke Koordinat Cartesian:**

camera\_x = target\_x + r × sin(φ) × cos(θ)

camera\_y = target\_y + r × cos(φ)

camera\_z = target\_z + r × sin(φ) × sin(θ)

**Visualisasi:**

φ (Phi - vertical)

|

| Camera

| /

|/\_\_\_\_\_ θ (Theta - horizontal)

[Pokemon]

#### **Kontrol Input:**

**Mouse Drag:**

* Gerak horizontal → ubah theta
* Gerak vertikal → ubah phi
* Sensitivity: 0.005 radian per pixel

**Mouse Wheel:**

* Scroll up → zoom in (r berkurang)
* Scroll down → zoom out (r bertambah)
* Range: 3-30 unit

**Keyboard WASD:**

* Menggerakkan target (titik fokus), bukan kamera
* W/S: Maju/mundur relatif terhadap view direction
* A/D: Strafe kiri/kanan
* Q/E: Naik/turun vertikal

**View Matrix:**

View = lookAt(camera\_position, target\_position, up\_vector)

### **2.6 MATRIX MATHEMATICS (Matematika Matriks)**

**Konsep:** Semua transformasi 3D direpresentasikan sebagai matriks 4×4.

#### **Transformation Matrix Chain**

Urutan transformasi (dari kanan ke kiri):

Final\_Matrix = Projection × View × Model

Model = Translation × Rotation\_Y × Rotation\_X × Rotation\_Z × Arbitrary\_Rotation × Scale

**Contoh Perhitungan:**

Vertex\_Position\_Final = Projection × View × Model × Vertex\_Position\_Local

[x'] [P00 P01 P02 P03] [V00 ...] [M00 ...] [x]

[y'] = [P10 P11 P12 P13] × [V10 ...] × [M10 ...] × [y]

[z'] [P20 P21 P22 P23] [V20 ...] [M20 ...] [z]

[w'] [P30 P31 P32 P33] [V30 ...] [M30 ...] [1]

#### **Operasi Matriks yang Diimplementasikan:**

1. **Identity:** Matriks identitas (tidak ada transformasi)
2. **Multiply:** Perkalian dua matriks
3. **Translate:** Matriks translasi
4. **RotateX/Y/Z:** Matriks rotasi sumbu standar
5. **RotateArbitrary:** Rotasi sumbu bebas (Rodrigues)
6. **Scale:** Matriks scaling
7. **Perspective:** Proyeksi perspektif
8. **LookAt:** View matrix kamera
9. **Invert:** Invers matriks (untuk normal transformation)
10. **Transpose:** Transpose matriks

### **2.7 SHADER PROGRAMMING (GLSL)**

**Konsep:** Program kecil yang berjalan di GPU untuk memproses vertex dan pixel.

#### **Vertex Shader (Pemrosesan Titik)**

**Input:**

* Position vertex (aPosition)
* Normal vector (aNormal)
* Color (aColor)

**Proses:**

1. Transformasi hierarchical (jika ada)
2. Model transformation (World space)
3. View transformation (Camera space)
4. Projection transformation (Screen space)

**Output:**

* gl\_Position: Posisi akhir di layar
* vNormal: Normal untuk lighting
* vColor: Warna untuk fragment shader

**Pseudo-code:**

glsl

void main() {

vec3 pos = applyHierarchicalTransform(aPosition);

vec4 worldPos = uModelMatrix × vec4(pos, 1.0);

gl\_Position = uProjectionMatrix × uViewMatrix × worldPos;

vNormal = mat3(uNormalMatrix) × aNormal;

vColor = aColor;

}

#### **Fragment Shader (Pewarnaan Pixel)**

**Input:**

* vNormal: Normal terinterpolasi
* vPosition: Posisi world
* vColor: Warna base

**Proses:**

1. Normalisasi vectors
2. Hitung lighting (Blinn-Phong)
3. Apply glow effect
4. Combine components

**Output:**

* gl\_FragColor: Warna final pixel

**Pseudo-code:**

glsl

void main() {

vec3 normal = normalize(vNormal);

vec3 lightDir = normalize(uLightPosition - vPosition);

float diffuse = max(dot(normal, lightDir), 0.0);

float specular = pow(max(dot(halfDir, normal), 0.0), uShininess);

vec3 ambient = 0.25 × vColor;

vec3 color = ambient + diffuse × vColor + specular × vec3(1.0);

gl\_FragColor = vec4(color + uGlowIntensity, 1.0);

}

### **2.8 OPTIMISASI & PERFORMANCE**

**Teknik yang Digunakan:**

**1. Buffer Management:**

* VBO (Vertex Buffer Object) untuk data vertex
* IBO (Index Buffer Object) untuk indices
* Menghindari pengiriman data berulang ke GPU

**2. Instancing Concept:**

* Satu mesh digunakan berkali-kali
* Hanya matrix transformation yang berubah

**3. Culling:**

* Back-face culling (tidak render sisi belakang)
* Depth testing (objek di depan menutupi yang di belakang)

**4. Blending:**

* Alpha blending untuk glow effect
* Hanya aktif saat evolusi (tidak selalu)

**5. RequestAnimationFrame:**

* Sinkronisasi dengan refresh rate monitor
* Efisiensi baterai dan CPU