**PRAKTIKUM GRAFIKA KOMPUTER**

Pertemuan ke-9 (Kurva Spline)



**Yoga Agustiansyah**2206050

Jurusan Ilmu Komputer

Program Studi Teknik Informatika

Institut Teknologi Garut

Jl. Mayor Syamsu No. 1 Jayaraga Garut 44151 Indonesia

# **I. PENDAHULUAN**

Kurva spline adalah kurva matematis yang terbentuk dari serangkaian segmen polinomial yang digabungkan secara mulus. Biasanya, kurva ini dibuat dengan menginterpolasi sekelompok titik kontrol untuk menghasilkan kurva yang halus dan kontinu. Terdapat berbagai jenis kurva spline, seperti spline linear, spline kubik, B-spline, Catmull-Rom spline, dan Bezier spline. Spline digunakan dalam grafika komputer, desain grafis, animasi, dan berbagai bidang lainnya untuk membuat kurva yang fleksibel dan mudah dimanipulasi. Keistimewaan utama kurva spline adalah kemampuannya untuk merepresentasikan kurva yang kompleks dengan mempertahankan kelancaran dan kehalusan di antara titik kontrolnya.

Dalam praktikum ini, kita akan memahami pembuatan kurva berjenis Spline Kubik, Catmull-Rom Spline dan Bezier Spline.

# **II. PEMBAHASAN**

## **A. Kurva Spline Cubic**

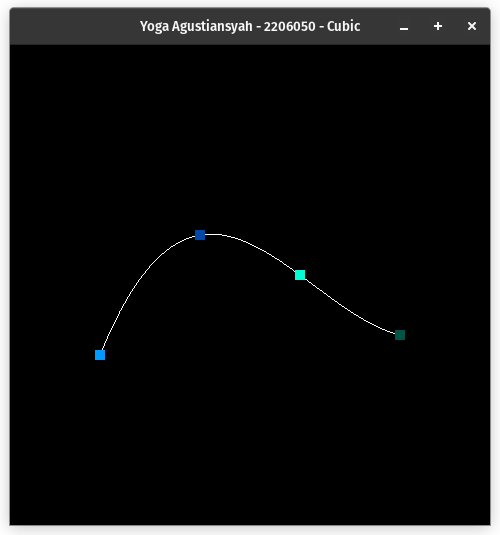
**a) Kode Pembuatan Kurva**

|  |
| --- |
| void drawSplineCubic(Vec3 point1, Vec3 point2, Vec3 point3, Vec3 point4, int nPoint) {  // hitung bobot jarak u di masing-masing titik  float utotal = (abs(point2.X - point1.X) + abs(point3.X - point2.X) + abs(point4.X - point3.X));  float u1 = 0;  float u2 = abs(point2.X - point1.X) / utotal;  float u3 = (abs(point2.X - point1.X) + abs(point3.X - point2.X)) / utotal;  float u4 = 1;  // hitung inverse matriks dari koefisien u (lihat slide kuliah)  float inverseMat[16];  float coeffMat[16] = {  1.00f, 0.00f, 0.00f, 0.00f,  1.00f, u2, pow(u2, 2), pow(u2, 3),  1.00f, u3, pow(u3, 2), pow(u3, 3),  1.00f, 1.00f, 1.00f, 1.00f};  bool status = inverse(coeffMat, inverseMat);  // hitung koefisien cubic au^3 + bu^2 + cu + d  if (status == true) {  float outMatX[4], outMatY[4], outMatZ[4];  float inMatX[4] = {point1.X, point2.X, point3.X, point4.X};  float inMatY[4] = {point1.Y, point2.Y, point3.Y, point4.Y};  float inMatZ[4] = {point1.Z, point2.Z, point3.Z, point4.Z};  DotMatrix(inverseMat, inMatX, outMatX);  DotMatrix(inverseMat, inMatY, outMatY);  DotMatrix(inverseMat, inMatZ, outMatZ);  // gambar kurva cubic spline dengan titik kontrol diatas  // hitung posisi y untuk setiap x di setiap point dengan persamaan diatas  for (int i = 0; i < nPoint; i++) {  float step = 1.0f / nPoint; // jeda setiap titik pd bobot u  // titik awal  float pX = point1.X, pY = point1.Y, pZ = point1.Z;  //  float u = 0.0f;  for (int j = 0; j < nPoint; j++) {  // bentuk segment kurva cubic spline sebanyak nPoint  u = u + step;  glVertex3f(pX, pY, pZ); // gambar titik awal  // koordinat X pada kurva  pX = outMatX[3] \* pow(u, 3) + outMatX[2] \* pow(u, 2) + outMatX[1] \* u + outMatX[0];  // koordinat Y pada kurva  pY = outMatY[3] \* pow(u, 3) + outMatY[2] \* pow(u, 2) + outMatY[1] \* u + outMatY[0];  // koordinat Z pada kurva  pZ = outMatZ[3] \* pow(u, 3) + outMatZ[2] \* pow(u, 2) + outMatZ[1] \* u + outMatZ[0];  glVertex3f(pX, pY, pZ); // gambar titik akhir  }  }  }  } |

**Mengubah Warna Titik**

|  |
| --- |
|  |

## **Output**



**Penjelasan Program**

Fungsi drawSplineCubic ini adalah untuk menggambar kurva cubic spline 3D berdasarkan empat titik kontrol yang diberikan (point1, point2, point3, dan point4). Berikut adalah penjelasan workflow atau alur kerja dari fungsi tersebut:

1. Menghitung Bobot Jarak U:
   * Menghitung total jarak bobot u antara titik kontrol.
   * Menghitung nilai u1, u2, u3, dan u4 sebagai koefisien pembobotan jarak relatif antara titik kontrol. Ini akan digunakan dalam perhitungan koefisien kurva.
2. Menghitung Inverse Matriks:
   * Membuat matriks koefisien berdasarkan nilai u yang dihitung sebelumnya.
   * Menghitung inversenya menggunakan fungsi inverse (yang tidak tersedia di kode yang diberikan namun diasumsikan telah didefinisikan di tempat lain).
   * Jika invers matriks berhasil dihitung, lanjut ke langkah selanjutnya.
3. Menghitung Koefisien Cubic:
   * Menggunakan matriks invers yang telah dihitung, mengalikan dengan vektor masukan (X, Y, Z) dari titik kontrol untuk menghasilkan koefisien cubic spline.
4. Menggambar Kurva Cubic Spline:
   * Untuk setiap titik pada kurva, melakukan langkah-langkah berikut:
     + Mengatur jumlah langkah (step) untuk menentukan jeda antara titik pada kurva.
     + Melakukan perulangan sebanyak nPoint (jumlah titik yang ingin digambar pada kurva).
     + Menggunakan persamaan cubic spline, menghitung koordinat X, Y, dan Z untuk setiap nilai u yang diiterasikan.
     + Menggambar titik awal dan akhir untuk setiap segmen kurva cubic spline yang dihasilkan.

Fungsi ini bertanggung jawab untuk menggambar kurva cubic spline dalam ruang tiga dimensi (3D) berdasarkan titik kontrol yang diberikan dengan menggunakan algoritma interpolasi cubic spline.

## **B. Kurva Spline Catmull-Rom**

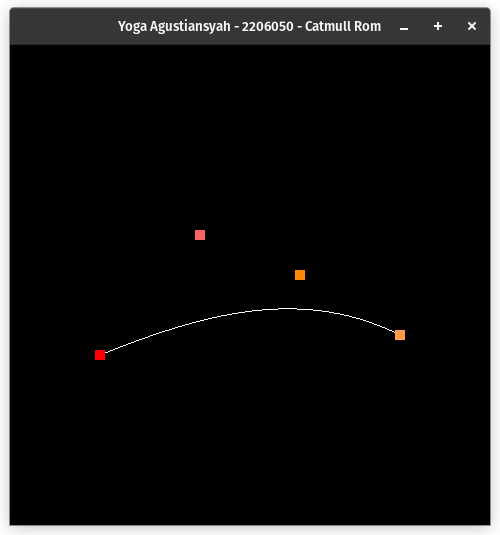
**a) Kode Pembuatan Kurva**

|  |
| --- |
| void drawSplineCatmullRom(Vec3 point1, Vec3 point2, Vec3 point3, Vec3 point4, int nPoint) {  // hitung bobot jarak u di masing-masing titik  float utotal = (abs(point2.X - point1.X) + abs(point3.X - point2.X) + abs(point4.X - point3.X));  float u1 = 0;  float u2 = abs(point2.X - point1.X) / utotal;  float u3 = (abs(point2.X - point1.X) + abs(point3.X - point2.X)) / utotal;  float u4 = 1;  // hitung inverse matriks dari koefisien u (lihat slide kuliah)  float inverseMat[16];  float coeffMat[16] = {  1.0000f, 0.0000f, 0.0000f, 0.0000f,  1.0000f, 1.0000f, 1.0000f, 1.0000f,  0.0000f, 1.0000f, 0.0000f, 0.0000f,  0.0000f, 1.0000f, 2.0000f, 3.0000f  };  bool status = inverse(coeffMat, inverseMat);  // hitung koefisien  if (status == true) {  float outMatX[4], outMatY[4], outMatZ[4];  float inMatX[4] = {point1.X, point4.X,  1.0f / (u3 - u1) \* (point3.X - point1.X),  1.0f / (u4 - u2) \* (point4.X - point2.X)};  float inMatY[4] = {point1.Y, point4.Y,  1.0f / (u3 - u1) \* (point3.Y - point1.Y),  1.0f / (u4 - u2) \* (point4.Y - point2.Y)};  float inMatZ[4] = {point1.Z, point4.Z,  1.0f / (u3 - u1) \* (point3.Z - point1.Z),  1.0f / (u4 - u2) \* (point4.Z - point2.Z)};  DotMatrix(inverseMat, inMatX, outMatX);  DotMatrix(inverseMat, inMatY, outMatY);  DotMatrix(inverseMat, inMatZ, outMatZ);  // gambar kurva spline dengan titik kontrol diatas  // hitung posisi y untuk setiap x di setiap point dengan  // persamaan diatas  for (int i = 0; i < nPoint; i++) {  // jeda setiap titik pd bobot u  float step = 1.0f / nPoint;  // titik awal  float pX = point1.X, pY = point1.Y, pZ = point1.Z;  //  float u = 0.0f;  for (int j = 0; j < nPoint; j++) {  // bentuk segment kurva spline sebanyak nPoint  u = u + step;  glVertex3f(pX, pY, pZ); // gambar titik awal  // koordinat X pada kurva  pX = outMatX[3] \* pow(u, 3) + outMatX[2] \* pow(u, 2) + outMatX[1] \* u + outMatX[0];  // koordinat Y pada kurva  pY = outMatY[3] \* pow(u, 3) + outMatY[2] \* pow(u, 2) + outMatY[1] \* u + outMatY[0];  // koordinat Z pada kurva  pZ = outMatZ[3] \* pow(u, 3) + outMatZ[2] \* pow(u, 2) + outMatZ[1] \* u + outMatZ[0];  glVertex3f(pX, pY, pZ); // gambar titik akhir  }  }  }  } |

**Mengubah Warna Titik**

|  |
| --- |
|  |

## **Output**



**Penjelasan Program**

Fungsi drawSplineCatmullRom ini digunakan untuk menggambar kurva spline Catmull-Rom dalam ruang tiga dimensi (3D) berdasarkan empat titik kontrol yang diberikan (point1, point2, point3, dan point4). Berikut adalah penjelasan workflow atau alur kerja dari fungsi tersebut:

1. Menghitung Bobot Jarak U:
   * Menghitung total jarak bobot u antara titik kontrol.
   * Menghitung nilai u1, u2, u3, dan u4 sebagai koefisien pembobotan jarak relatif antara titik kontrol. Ini akan digunakan dalam perhitungan koefisien kurva.
2. Menghitung Inverse Matriks:
   * Membuat matriks koefisien berdasarkan nilai u yang dihitung sebelumnya.
   * Menghitung inversenya menggunakan fungsi inverse (yang tidak tersedia di kode yang diberikan namun diasumsikan telah didefinisikan di tempat lain).
   * Jika invers matriks berhasil dihitung, lanjut ke langkah selanjutnya.
3. Menghitung Koefisien:
   * Berdasarkan formula Catmull-Rom spline, menghitung koefisien kurva dengan menggunakan matriks invers yang telah dihitung sebelumnya.
4. Menggambar Kurva Spline Catmull-Rom:
   * Untuk setiap titik pada kurva, melakukan langkah-langkah berikut:
     + Mengatur jumlah langkah (step) untuk menentukan jeda antara titik pada kurva.
     + Melakukan perulangan sebanyak nPoint (jumlah titik yang ingin digambar pada kurva).
     + Menggunakan persamaan spline Catmull-Rom, menghitung koordinat X, Y, dan Z untuk setiap nilai u yang diiterasikan.
     + Menggambar titik awal dan akhir untuk setiap segmen kurva Catmull-Rom yang dihasilkan.

Fungsi ini bekerja mirip dengan cubic spline namun menggunakan formula Catmull-Rom yang memberikan hasil kurva spline yang lebih halus melewati semua titik kontrol yang diberikan.

## **C. Kurva Spline Catmull-Rom**

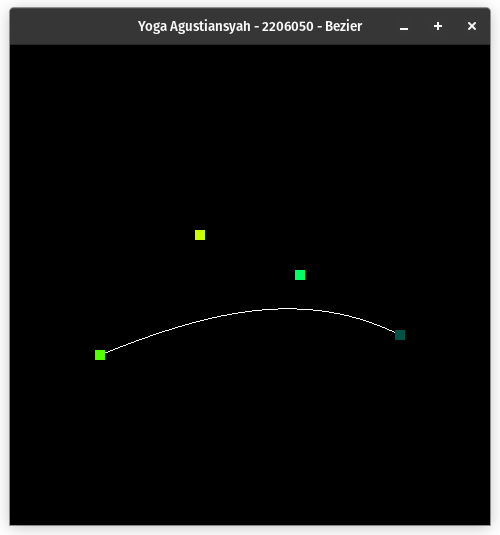
**a) Kode Pembuatan Kurva**

|  |
| --- |
| void drawSplineBezier(Vec3 point1, Vec3 point2, Vec3 point3, Vec3 point4, int nPoint) {  // hitung bobot jarak u di masing-masing titik  float utotal = (abs(point2.X - point1.X) + abs(point3.X - point2.X) + abs(point4.X - point3.X));  float u1 = 0;  float u2 = abs(point2.X - point1.X) / utotal;  float u3 = (abs(point2.X - point1.X) + abs(point3.X - point2.X)) / utotal;  float u4 = 1;  // hitung inverse matriks dari koefisien u (lihat slide kuliah)  float inverseMat[16];  float coeffMat[16] = {  1.00f, 0.00f, 0.00f, 0.00f,  1.00f, 1.00f, 1.00f, 1.00f,  0.00f, 1.00f, 0.00f, 0.00f,  0.00f, 1.00f, 2.00f, 3.00f  };  bool status = inverse(coeffMat, inverseMat);  // hitung koefisien cubic au^3 + bu^2 + cu + d  if (status == true) {  float outMatX[4], outMatY[4], outMatZ[4];  // rumus bezier  float inMatX[4] = {point1.X, point4.X,  1.0f / (u3 - u1) \* (point3.X - point1.X),  1.0f / (u4 - u2) \* (point4.X - point2.X)};  float inMatY[4] = {point1.Y, point4.Y,  1.0f / (u3 - u1) \* (point3.Y - point1.Y),  1.0f / (u4 - u2) \* (point4.Y - point2.Y)};  float inMatZ[4] = {point1.Z, point4.Z,  1.0f / (u3 - u1) \* (point3.Z - point1.Z),  1.0f / (u4 - u2) \* (point4.Z - point2.Z)};  DotMatrix(inverseMat, inMatX, outMatX);  DotMatrix(inverseMat, inMatY, outMatY);  DotMatrix(inverseMat, inMatZ, outMatZ);  // gambar kurva cubic spline dengan titik kontrol diatas  // hitung posisi y untuk setiap x di setiap point dengan persamaan diatas  for (int i = 0; i < nPoint; i++) {  float step = 1.0f / nPoint; // jeda setiap titik pd bobot u  // titik awal  float pX = point1.X, pY = point1.Y, pZ = point1.Z;  //  float u = 0.0f;  for (int j = 0; j < nPoint; j++) {  // bentuk segment kurva cubic spline sebanyak nPoint  u = u + step;  glVertex3f(pX, pY, pZ); // gambar titik awal  // koordinat X pada kurva  pX = outMatX[3] \* pow(u, 3) + outMatX[2] \* pow(u, 2) + outMatX[1] \* u + outMatX[0];  // koordinat Y pada kurva  pY = outMatY[3] \* pow(u, 3) + outMatY[2] \* pow(u, 2) + outMatY[1] \* u + outMatY[0];  // koordinat Z pada kurva  pZ = outMatZ[3] \* pow(u, 3) + outMatZ[2] \* pow(u, 2) + outMatZ[1] \* u + outMatZ[0];  glVertex3f(pX, pY, pZ); // gambar titik akhir  }  }  }  } |

**Mengubah Warna Titik**

|  |
| --- |
|  |

## **Output**



**Penjelasan Program**

Fungsi drawSplineBezier ini bertujuan untuk menggambar kurva spline Bezier dalam ruang tiga dimensi (3D) berdasarkan empat titik kontrol yang diberikan (point1, point2, point3, dan point4). Berikut adalah penjelasan workflow atau alur kerja dari fungsi tersebut:

1. Menghitung Bobot Jarak U:
   * Menghitung total jarak bobot u antara titik kontrol.
   * Menghitung nilai u1, u2, u3, dan u4 sebagai koefisien pembobotan jarak relatif antara titik kontrol. Ini akan digunakan dalam perhitungan koefisien kurva.
2. Menghitung Inverse Matriks:
   * Membuat matriks koefisien berdasarkan nilai u yang dihitung sebelumnya.
   * Menghitung inversenya menggunakan fungsi inverse (yang diasumsikan telah didefinisikan di tempat lain).
   * Jika invers matriks berhasil dihitung, lanjut ke langkah selanjutnya.
3. Menghitung Koefisien Cubic Bezier:
   * Berdasarkan formula Bezier, menghitung koefisien kurva dengan menggunakan matriks invers yang telah dihitung sebelumnya.
4. Menggambar Kurva Spline Bezier:
   * Untuk setiap titik pada kurva, melakukan langkah-langkah berikut:
     + Mengatur jumlah langkah (step) untuk menentukan jeda antara titik pada kurva.
     + Melakukan perulangan sebanyak nPoint (jumlah titik yang ingin digambar pada kurva).
     + Menggunakan persamaan spline Bezier, menghitung koordinat X, Y, dan Z untuk setiap nilai u yang diiterasikan.
     + Menggambar titik awal dan akhir untuk setiap segmen kurva Bezier yang dihasilkan.

Fungsi ini menggunakan pendekatan kurva Bezier dengan menghitung koefisien menggunakan matriks invers. Kurva Bezier dapat memberikan kontrol yang lebih besar terhadap jalur kurva, terutama karena penggunaan titik kontrol yang diberikan secara langsung dalam perhitungannya.

**D. Perbedaan Setiap Metode Pembuatan Kurva**

Tiga metode pembuatan kurva spline, yaitu Spline Cubic, Spline Catmull-Rom, dan Bezier, memiliki perbedaan dalam pendekatan matematis yang digunakan untuk menghasilkan kurva.

1. Spline Cubic:
   * Interpolasi Polinomial Derajat Tiga: Pendekatan ini menggunakan interpolasi polinomial derajat tiga (atau cubic) antara titik kontrol.
   * Basis Bobot U: Spline Cubic menggunakan bobot u yang berbeda-beda antara titik kontrol untuk memperoleh nilai u2​ dan u3​. Bobot ini mengatur pengaturan polinomial cubic antara titik kontrol yang berdekatan.
   * Perhitungan Koefisien Cubic: Dengan menggunakan invers matriks dari koefisien bobot yang dihitung, fungsi ini memperoleh koefisien cubic untuk setiap sumbu X,Y,Z. Koefisien ini digunakan untuk menggambar kurva cubic antara titik kontrol.
   * Interpolasi Lancar antara Titik: Hasil kurva mengikuti pola interpolasi yang lancar antara titik kontrol, dengan tingkat kehalusan yang bergantung pada jarak dan bobot u yang dihitung.
   * **Rumus untuk kurva spline cubic antara titik kontrol P1, P2, P3, dan P4 adalah:**

C(u) = P1u3 + P2u2 + P3u + P4

Rumus umum dari kurva spline cubic dapat dituliskan dengan menggunakan vektor parameter u dan matriks koefisien kontrol.

1. Spline Catmull-Rom:
   * Interpolasi Titik Kontrol: Menggunakan metode interpolasi yang memastikan kurva melewati setiap titik kontrolnya secara langsung.
   * Rumus Matematis Tetap: Tanpa memperhitungkan bobot u seperti pada Spline Cubic, Spline Catmull-Rom menggunakan rumus matematis tetap yang telah ditentukan sebelumnya. Rumus ini menjamin bahwa kurva akan melalui setiap titik kontrol dengan lancar.
   * Tidak Ada Penghitungan Bobot U: Tidak ada perhitungan bobot u, sehingga kurva Catmull-Rom terdiri dari segmen polinomial yang menghubungkan setiap titik kontrol.
2. Spline Bezier:
   * Rumus Bezier Khusus: Bezier menggunakan rumus khusus yang telah ditentukan sebelumnya untuk kurva.
   * Titik Kontrol Mengatur Pola Kurva: Pola kurva yang dihasilkan sangat tergantung pada posisi dan jumlah titik kontrol yang diberikan. Setiap titik kontrol memiliki pengaruh langsung pada pola kurva yang dihasilkan.
   * Interpolasi Polinomial Derajat Tiga: Mirip dengan Spline Cubic, Bezier juga menggunakan polinomial derajat tiga yang diatur oleh titik kontrolnya, namun dengan cara pendekatan matematis yang berbeda.
   * Rumus untuk kurva spline Bezier antara titik kontrol P1, P2, P3, dan P4 dapat dituliskan dalam bentuk:

B(u) = (1 – u)3 P1 + 3u(1 – u)2 P2 + 3u2(1 – u) P3 + u3P4

Rumus ini menggunakan polinomial derajat tiga yang dikontrol oleh empat titik kontrol untuk menentukan jalur kurva Bezier.

# **Untuk kode lengkap semua programnya, bisa diakses di :** [**https://github.com/yoga220802/Praktikum\_Grafkom/tree/main/Pertemuan%206/tasks**](https://github.com/yoga220802/Praktikum_Grafkom/tree/main/Pertemuan 6/tasks)

# **III. Kesimpulan**

Dalam praktikum ini, kita telah mempelajari dasar-dasar pemodelan 3D dengan jaring poligon menggunakan OpenGL dan GLUT. kita telah menjelajahi konsep dasar pemodelan 3D dengan bentuk geometris dasar seperti prisma segi lima, antiprisma segi lima, tetrahedron, octahedron, dodecahedron, dan icosahedron.

Selama praktikum, kita telah mempelajari konsep pencahayaan dan material yang memengaruhi tampilan objek 3D. kita juga telah melihat bagaimana menggambar objek 3D dengan penggunaan glBegin dan glEnd untuk mendefinisikan poligon dan permukaan. Modifikasi warna objek telah kita lakukan untuk memberikan tampilan yang lebih menarik.

Dengan menyelesaikan praktikum ini, kita memiliki pemahaman dasar tentang pemodelan 3D dan penggunaan OpenGL dalam pengembangan aplikasi grafis. kita siap untuk menjelajahi konsep yang lebih canggih dalam dunia grafik komputer dan membangun aplikasi grafis yang lebih kompleks di masa depan.