**LAPORAN PRAKTIKUM**

**“Pertemuan ke-2: Algoritma Garis”**

Diajukan untuk memenuhi salah satu praktikum Mata kuliah Praktikum Grafika Komputer yang di ampu oleh:

 Dr., Murinto, S.Si., M.Kom.

Disusun Oleh:

Mohammad Farid Hendianto 2200018401

A / Jum’at 10.00 – 11.30 Lab. Jaringan

**PROGRAM STUDI INFORMATIKA**

**UNIVERSITAS AHMAD DAHLAN**

**FAKULTAS TEKNOLOGI INDUSTRI**

**TAHUN 2024**

**DAFTAR ISI**

[**DAFTAR ISI** 2](#_Toc162619163)

[**LANGKAH PRAKTIKUM** 3](#_Toc162619164)

[**POST TEST** 24](#_Toc162619165)

# **LANGKAH PRAKTIKUM**

**Persiapan**

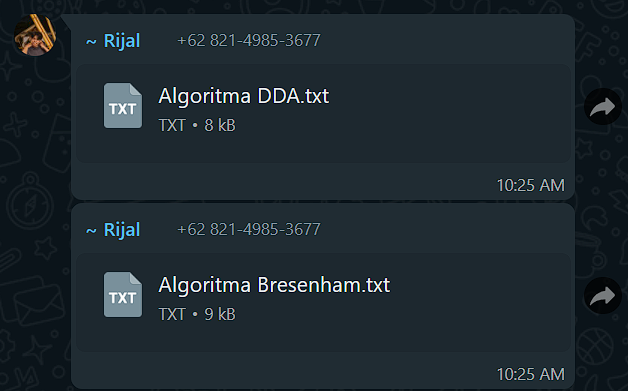
1. Buka Visual Studio C++ dan buat project baru dengan nama praktikum02.

Saya sudah menggunakan Clion, dan tidak menggunakn Visual studio Code.

1. Download kode dasar praktikum Grafika Komputer dan Library OpenGL seperti pada Praktikum 1.

Saya sudah menginstall library OpenGL melalui MSYS2 yang disediakan oleh Microsoft

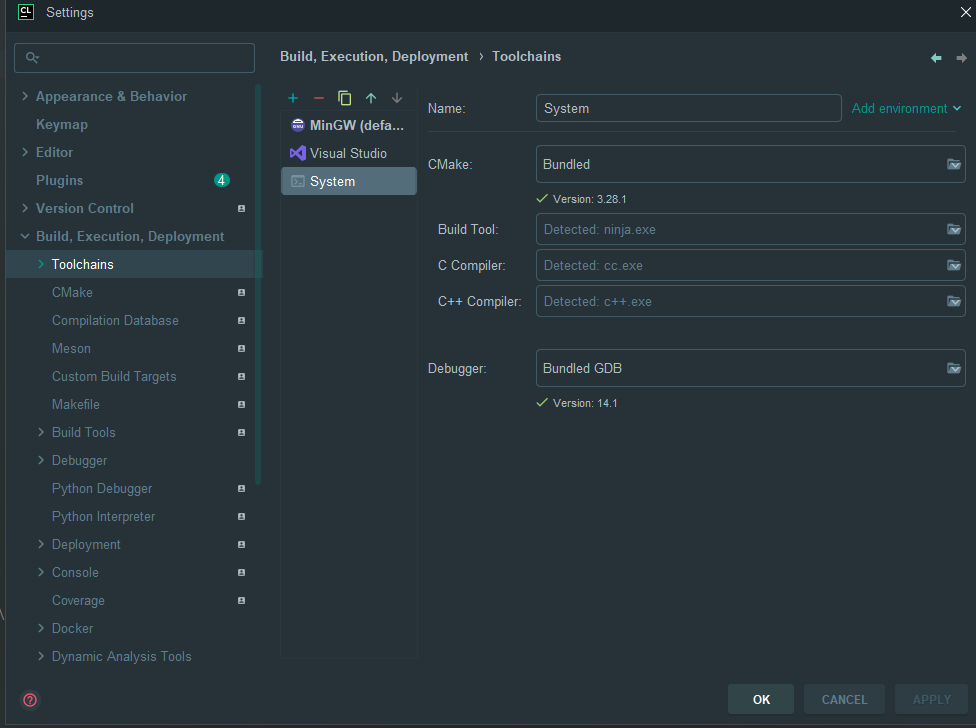
1. Ubah nama dari kode dasar “praktikum00.cpp” menjadi “praktikum02.cpp” dan copy-kan ke Source Files di project yang anda buat.



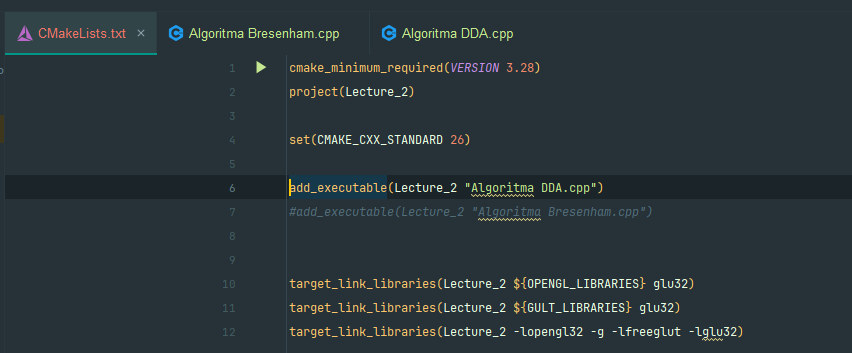
Mendownload source code yang diberikan oleh mas Rijal

1. Setting OpenGL library pada Visual Studio C/C++ seperti pada Praktikum 1.

Mengatur executable compiler ke system (Karena menggunakan MSYS2).



Mengatur cMakeLists.txt untuk linking library OpenGL



Jika ingin mengeksekusikan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **CPL** | **CPMK** | **Pertanyaan** | **Dokumen**  **Pendukung** | **Skor** |
| 1. | CPL 06-  P01 | CPMK-01 | Selesaikan langkah praktikum 1 – 9 | Hasil praktikum  langkah 1 – 9 | 50 |

Langkah-Langkah:

*Karena ini tidak ada praktikum00 format dari Pak Adhi Prahara tetapi langsung diberikan semua source code untuk pertemuan praktikum 2 ini, maka disini kita hanya mengecek apakah sudah di ubah sesuai langkah praktikum atau belum.*

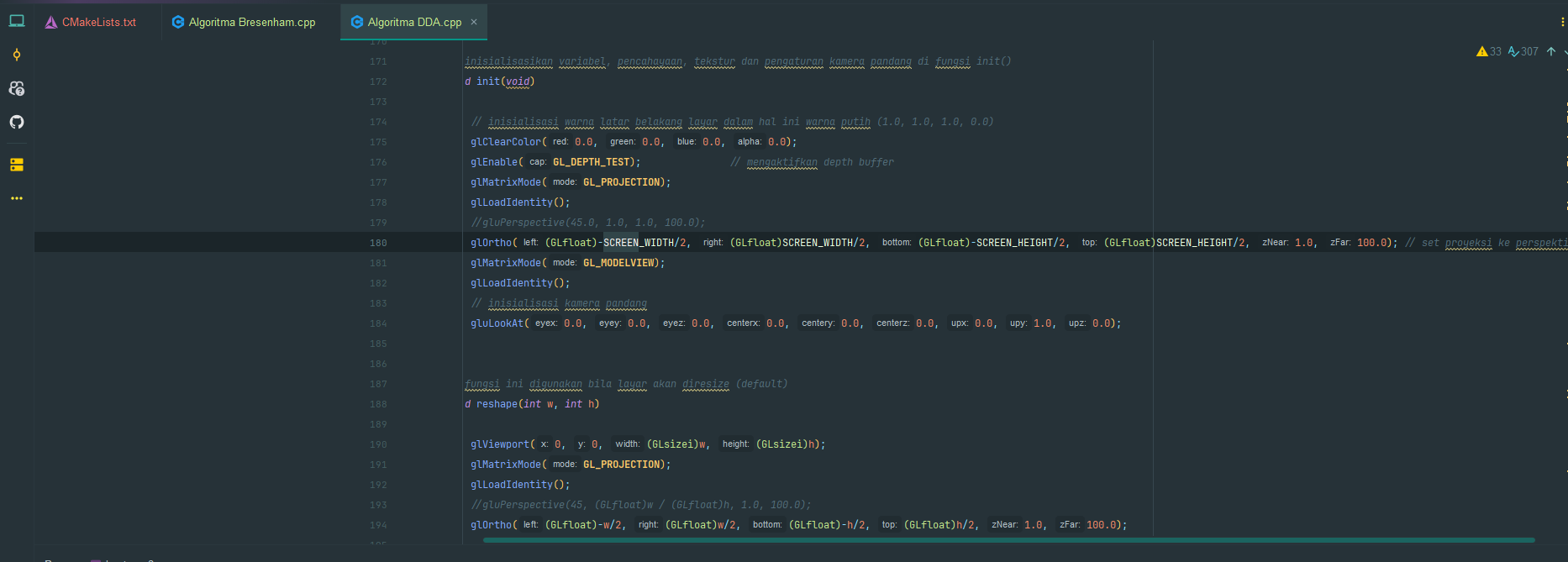
1. Untuk menggambar garis yang berupa obyek 2D, proyeksi dari kamera perlu di ubah ke proyeksi Orthogonal.
2. Di fungsi init() dalam praktikum02.cpp, ubah baris kode berikut :

gluPerspective(45.0, 1.0, 1.0, 100.0);

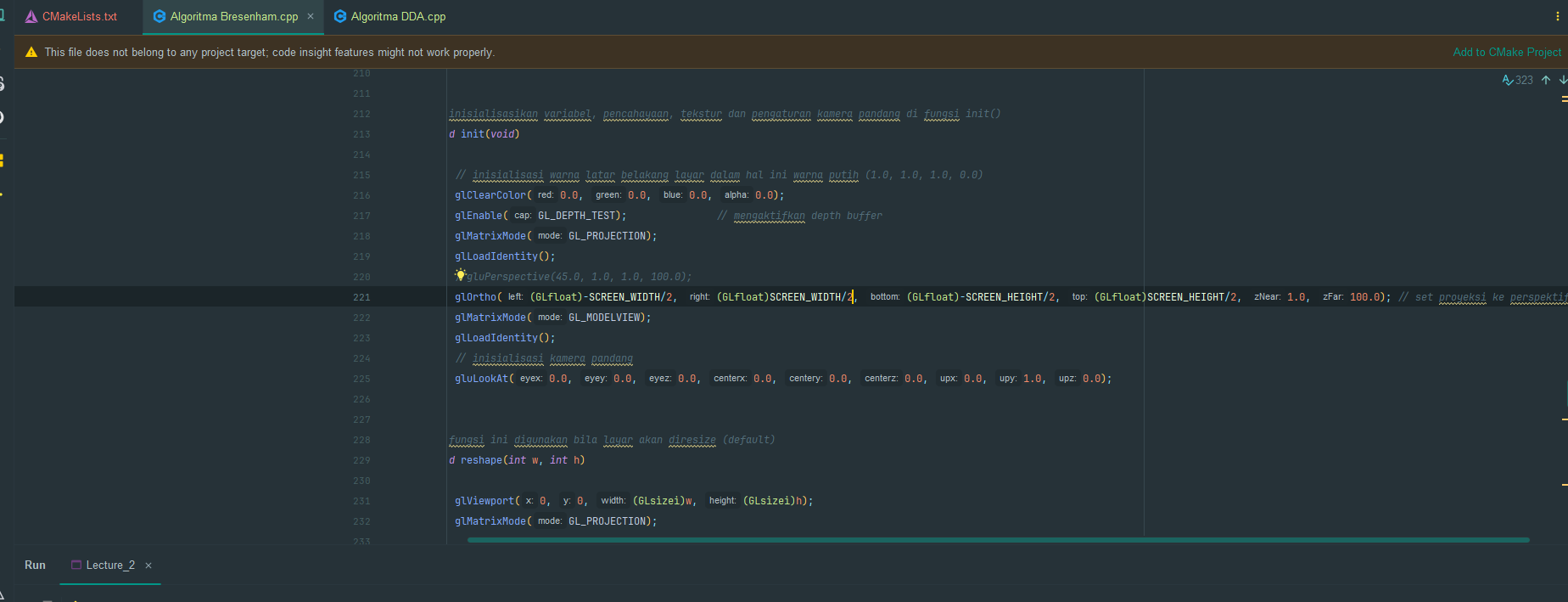
menjadi

glOrtho((GLfloat)-SCREEN\_WIDTH/2, (GLfloat)SCREEN\_WIDTH/2,

(GLfloat)-SCREEN\_HEIGHT/2, (GLfloat)SCREEN\_HEIGHT/2, 1.0, 100.0);



Algoritma DDA



Algoritma Bresenham

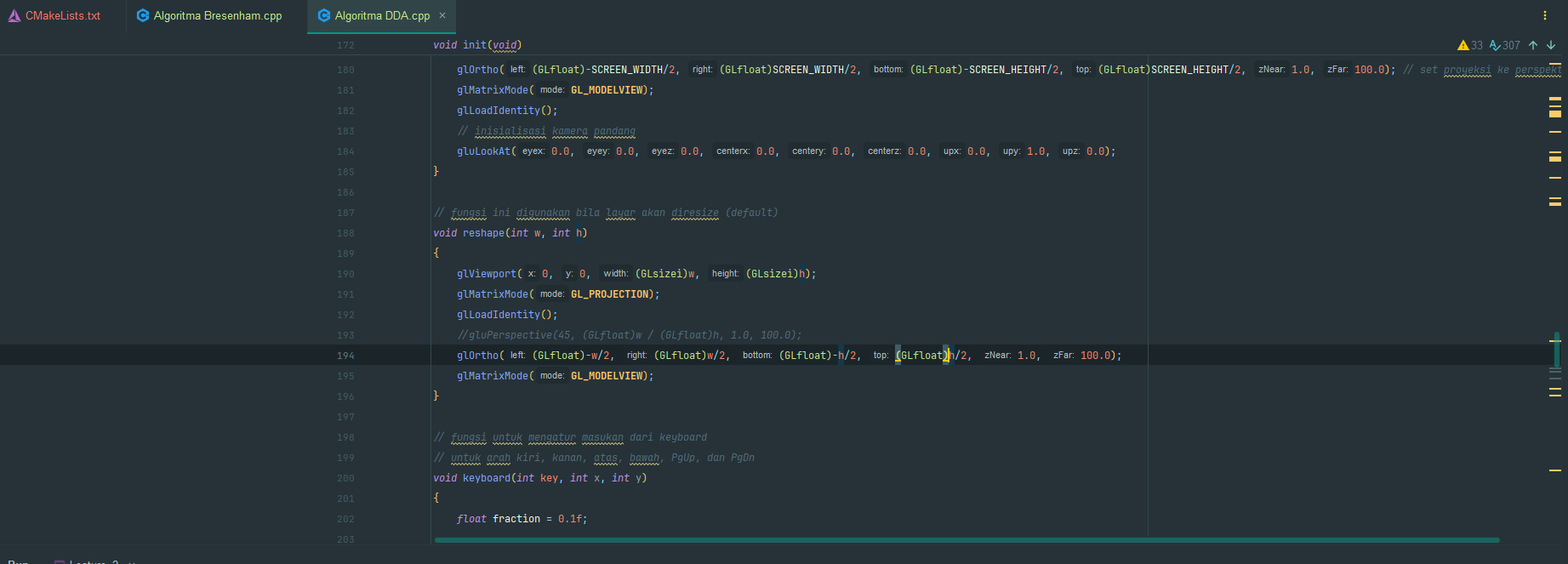
Setelah di cek, kedua algoritma sudah di ubah dari fungsi glPerspecitve() menjadi glOrtho() di bagian fungsi init().

1. Di fungsi reshape() dalam praktikum02.cpp, ubah baris kode berikut :

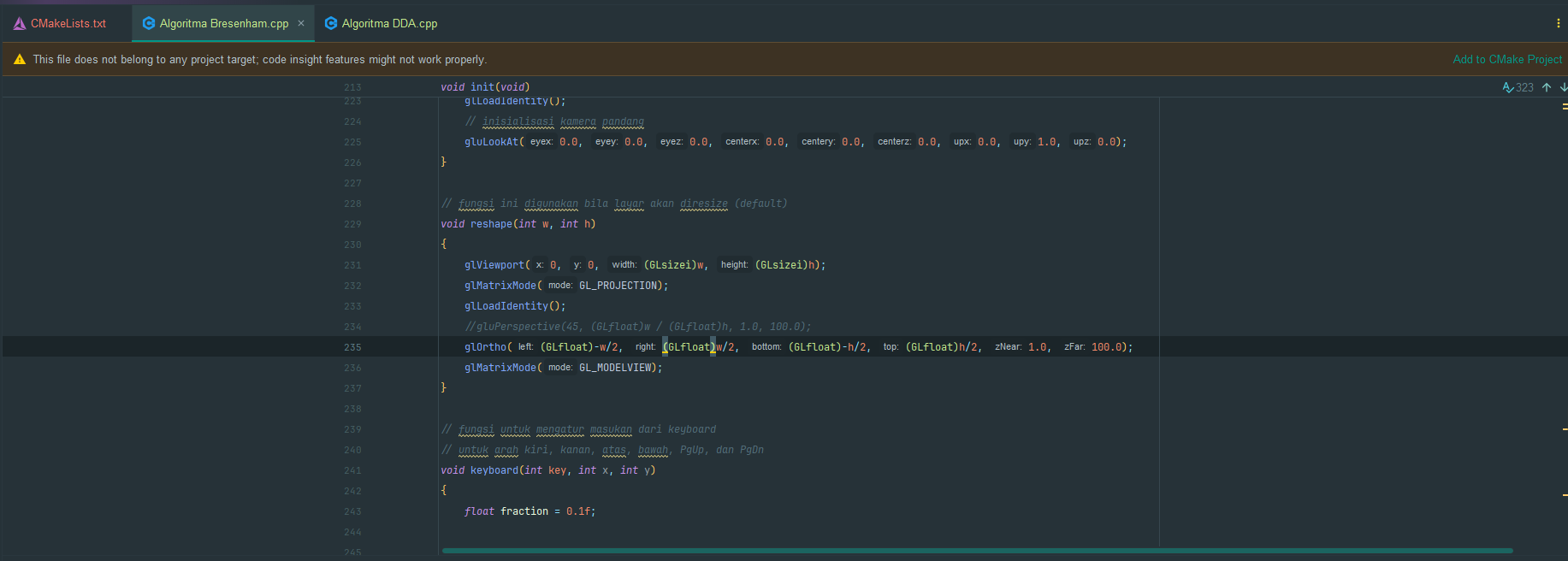
gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

menjadi

glOrtho((GLfloat)-w/2, (GLfloat)w/2, (GLfloat)-h/2, (GLfloat)h/2, 1.0, 100.0);



Algoritma DDA



Algoritma Bresenham

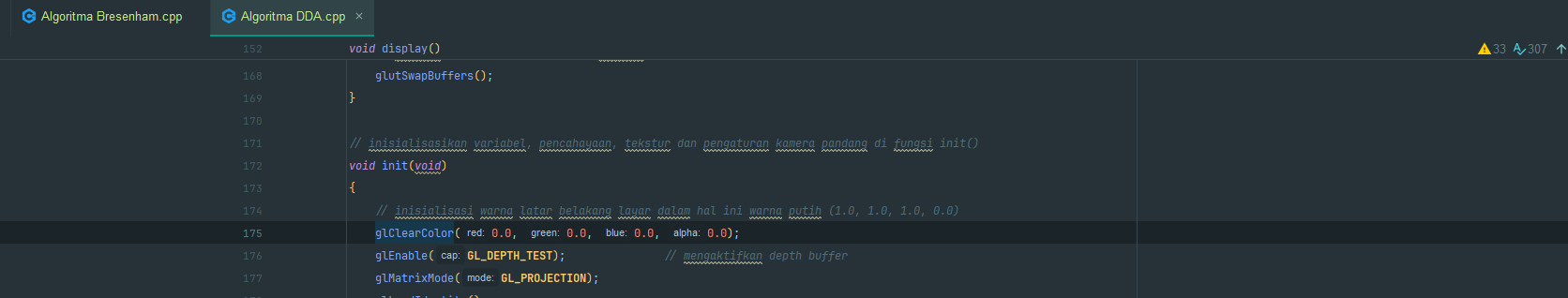
Setelah di cek, kedua algoritma sudah di ubah dari fungsi glPerspecitve() menjadi glOrtho() di bagian fungsi reshape().

1. Di fungsi init() dalam praktikum02.cpp, ubah baris kode berikut untuk mengubah warna latar belakang pada layar menjadi warna hitam.

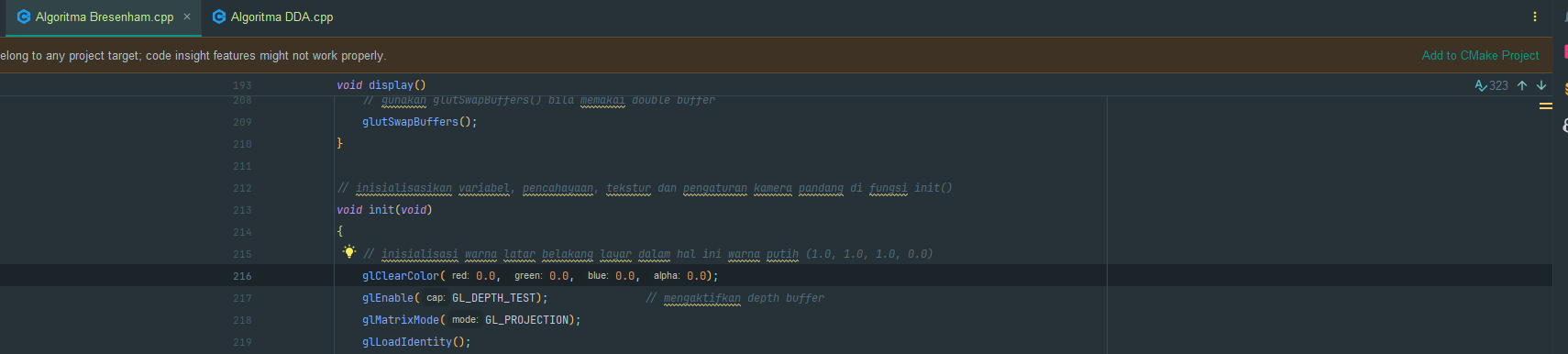
glClearColor(1.0, 1.0, 1.0, 0.0);

menjadi

glClearColor(0.0, 0.0, 0.0, 0.0);



Algoritma DDA



Algoritma Bresenham

1. Tambahkan fungsi lineDDAX() di file praktikum02.cpp anda untuk menggambar garis dengan algoritma DDA bila kenaikan terhadap X.

// fungsi untuk menggambar garis dengan algoritma DDA bila terhadap X void lineDDAX(Vec3 point1, Vec3 point2)

{

// hitung gradient garis m int dY = point2.Y - point1.Y; int dX = point2.X - point1.X;

float m = (float)dY / dX; float im = 1.0f/m;

// mulai menggambar titik-titik glBegin(GL\_POINTS);

// koordinat titik awal glVertex3f(point1.X, point1.Y, point1.Z);

float pX = point1.X, pY = point1.Y, pZ = point1.Z;

// kenaikan terhadap X

for (int i = point1.X; i < point2.X; i++)

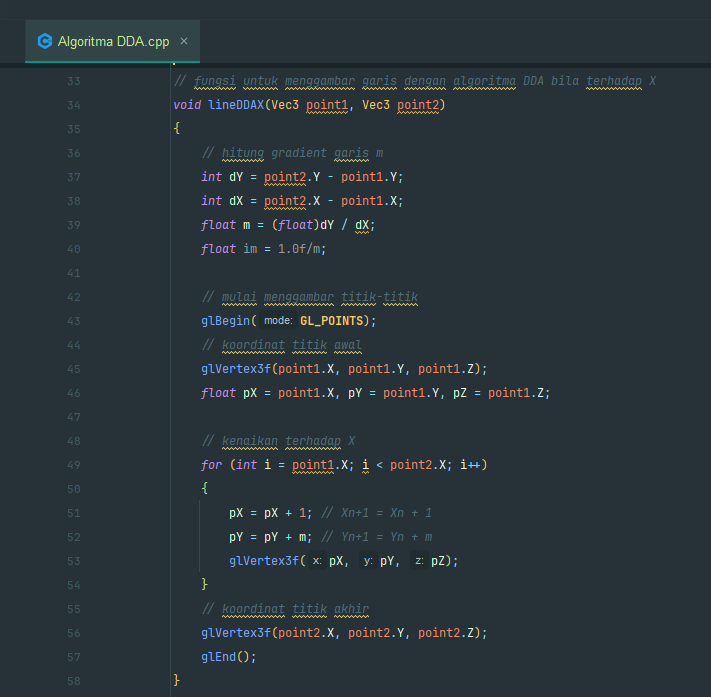
{

pX = pX + 1; // Xn+1 = Xn + 1 pY = pY + m; // Yn+1 = Yn + m glVertex3f(pX, pY, pZ);

}

// koordinat titik akhir glVertex3f(point2.X, point2.Y, point2.Z); glEnd();

}



Algoritma DDA

1. Tambahkan fungsi lineDDAY() di file praktikum02.cpp anda untuk menggambar garis dengan algoritma DDA bila kenaikan terhadap Y (copy paste dari fungsi lineDDAX() di langkah 9 dan ubah seperti fungsi dibawah ini).

// fungsi untuk menggambar garis dengan algoritma DDA bila terhadap Y void lineDDAY(Vec3 point1, Vec3 point2)

{

// hitung gradient garis m int dY = point2.Y - point1.Y; int dX = point2.X - point1.X; float m = (float)dY / dX; float im = 1.0f/m;

// mulai menggambar titik-titik glBegin(GL\_POINTS);

// koordinat titik awal glVertex3f(point1.X, point1.Y, point1.Z);

float pX = point1.X, pY = point1.Y, pZ = point1.Z;

// kenaikan terhadap Y

for (int i = point1.Y; i < point2.Y; i++)

{

pX = pX + im; // Xn+1 = Xn + 1/m pY = pY + 1; // Yn+1 = Yn + 1 glVertex3f(pX, pY, pZ);

}

// koordinat titik akhir glVertex3f(point2.X, point2.Y, point2.Z); glEnd();

}



Algoritma DDA

1. Tambahkan fungsi lineDDA() di file praktikum02.cpp anda untuk menggambar garis dengan algoritma DDA dengan memanggil fungsi yang anda buat di langkah 9 dan langkah 10.

// fungsi untuk menggambar garis dengan algoritma DDA void lineDDA(Vec3 point1, Vec3 point2)

{

// hitung selisih panjang

int dY = point2.Y - point1.Y; int dX = point2.X - point1.X;

// bila deltaY lebih pendek dari deltaX

if (abs(dY) < abs(dX))

{

if (point1.X < point2.X) // bila X1 < X2 lineDDAX(point1, point2);

else // bila X1 > X2 maka dibalik lineDDAX(point2, point1);

}

else // bila deltaY lebih panjang dari deltaX

{

if (point1.Y < point2.Y) // bila Y1 < Y2 lineDDAY(point1, point2);

else // bila Y1 > Y2 maka dibalik lineDDAY(point2, point1);

}

}

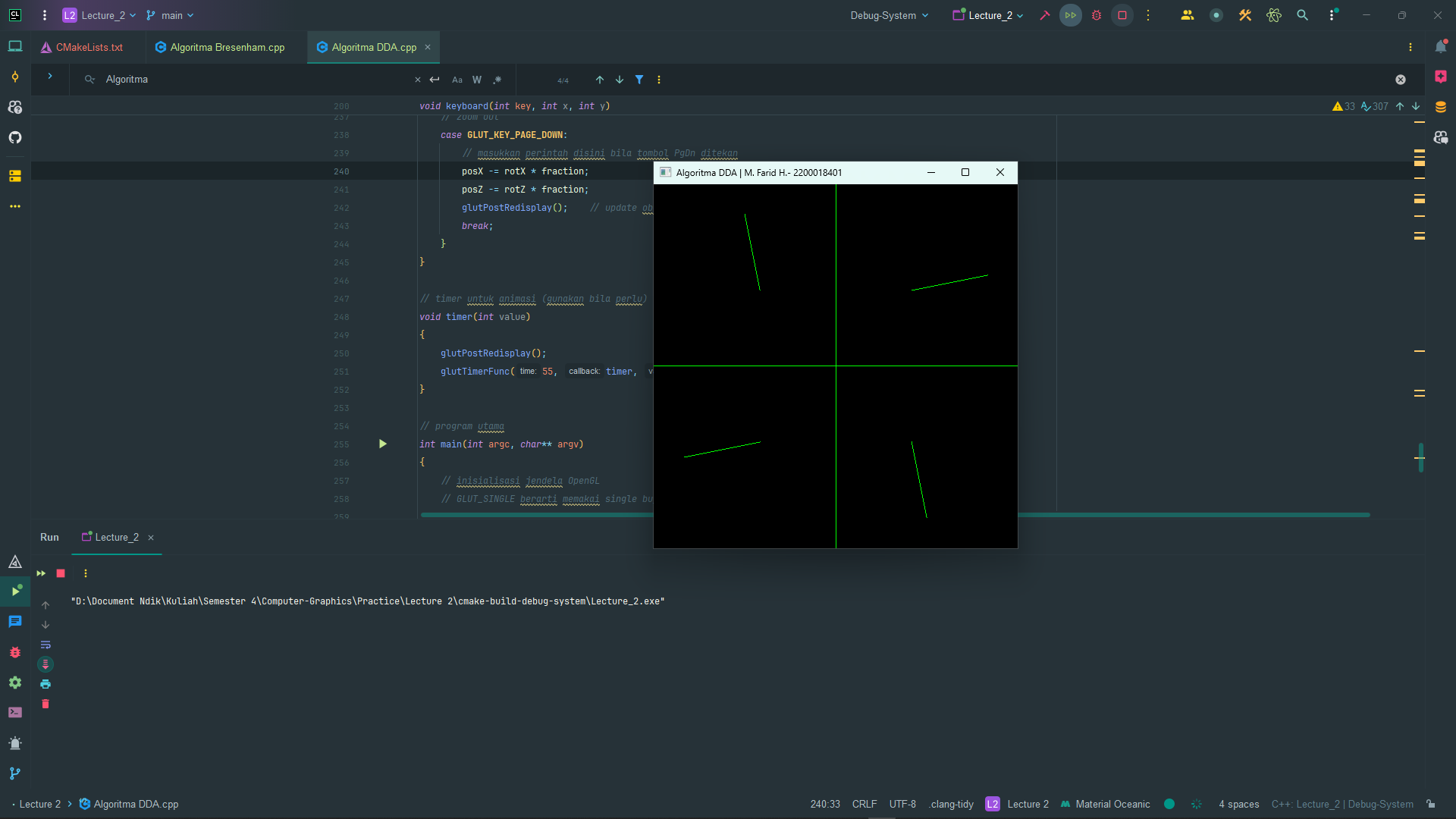


Algoritma DDA

1. Ubah fungsi drawObject() di praktikum02.cpp menjadi seperti dibawah ini.

|  |
| --- |
|  |

1. Jalankan program untuk melihat hasil dari pembuatan garis dengan algoritma DDA seperti yang ditunjukkan pada Gambar 2.1.



Hasil pembuatan garis dengan algoritma DDA.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **CPL** | **CPMK** | **Pertanyaan** | **Dokumen**  **Pendukung** | **Skor** |
| 2. | CPL 06-  P01 | CPMK-01 | Selesaikan langkah praktikum 10 –  14 | Hasil praktikum  langkah 10 - 14 | 50 |

1. Tambahkan fungsi lineBresenhamX() di file praktikum02.cpp anda untuk menggambar garis dengan algoritma Bresenham bila kenaikan terhadap X.

// fungsi untuk menggambar garis dengan algoritma Bresenham

// bila slopenya terhadap X

void lineBresenhamX(Vec3 point1, Vec3 point2)

{

// hitung selisih panjang

int dY = point2.Y - point1.Y; int dX = point2.X - point1.X;

int yi = 1; // skala penambahan

// bila delta Y kurang dari 0 if (dY < 0)

{

yi = -1; dY = -dY;

}

// mulai menggambar titik-titik glBegin(GL\_POINTS);

// koordinat titik awal glVertex3f(point1.X, point1.Y, point1.Z);

int pX = point1.X, pY = point1.Y, pZ = point1.Z; int dY2 = 2\*dY; // hitung 2\*deltaY

int dX2 = 2\*dX; // hitung 2\*deltaX int pk = dY2 - dX; // hitung p0

// kenaikan terhadap X

for (int i = point1.X; i < point2.X; i++)

{

if (pk < 0) // bila p < 0

{

pk = pk + dY2; // update pk+1 = pk + 2dY pX = pX + 1; // Xn+1 = Xn + 1

pY = pY; // Yn+1 = Yn

}

else // bila p >= 0

{

pk = pk + dY2 - dX2; pX = pX + 1;

pY = pY + yi;

// update pk+1 = pk + 2dY - 2dX

// Xn+1 = Xn + 1

// Yn+1 = Yn + yi

}

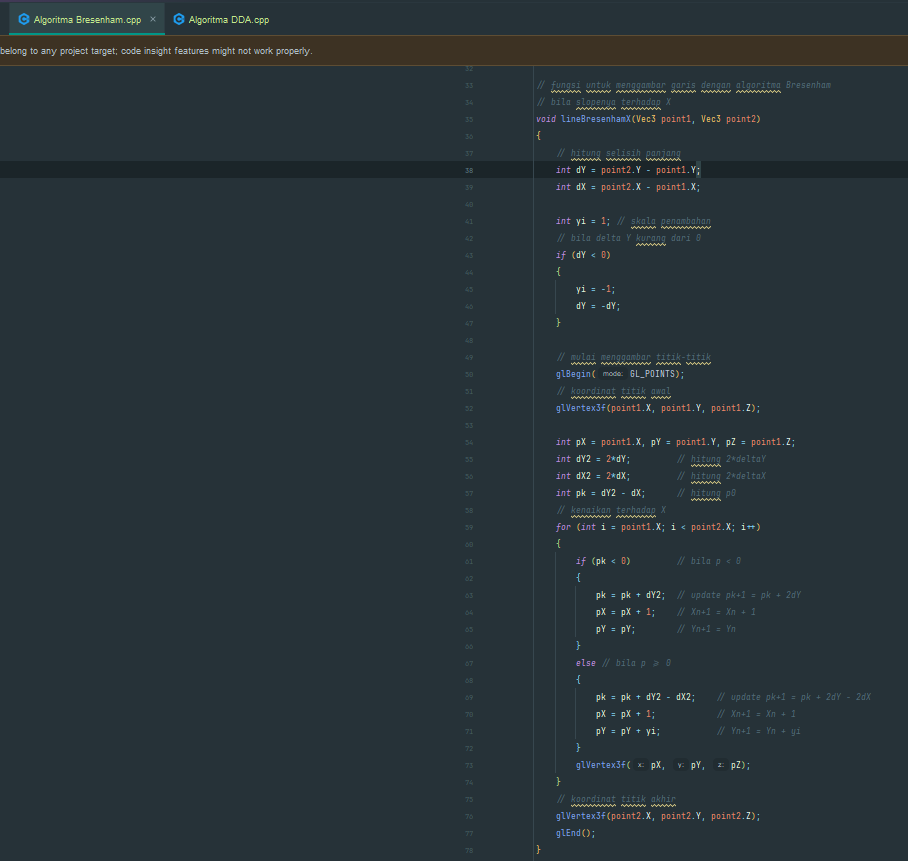
glVertex3f(pX, pY, pZ);

}

// koordinat titik akhir glVertex3f(point2.X, point2.Y, point2.Z); glEnd();

}

27



Algoritma BresenHam

1. Tambahkan fungsi lineBresenhamY() di file praktikum02.cpp anda untuk menggambar garis dengan algoritma Bresenham bila kenaikan terhadap Y (copy paste dari fungsi lineBresenhamX() di langkah 14 dan ubah seperti fungsi dibawah ini).

// fungsi untuk menggambar garis dengan algoritma Bresenham

// bila slopenya terhadap Y

void lineBresenhamY(Vec3 point1, Vec3 point2)

{

// hitung selisih panjang

int dY = point2.Y - point1.Y; int dX = point2.X - point1.X;

int xi = 1; // skala penambahan

// bila delta X kurang dari 0 if (dX < 0)

{

xi = -1; dX = -dX;

}

// mulai menggambar titik-titik glBegin(GL\_POINTS);

// koordinat titik awal glVertex3f(point1.X, point1.Y, point1.Z);

int pX = point1.X, pY = point1.Y, pZ = point1.Z; int dY2 = 2\*dY; // hitung 2\*deltaY

int dX2 = 2\*dX; // hitung 2\*deltaX int pk = dX2 - dY; // hitung p0

// kenaikan terhadap Y

for (int i = point1.Y; i < point2.Y; i++)

{

if (pk < 0) // bila p < 0

{

pk = pk + dX2; // update pk+1 = pk + 2dX pX = pX; // Xn+1 = Xn

pY = pY + 1; // Yn+1 = Yn + 1

}

else // bila p >= 0

{

pk = pk + dX2 - dY2; // update pk+1 = pk + 2dX - 2dY pX = pX + xi; // Xn+1 = Xn + xi

pY = pY + 1; // Yn+1 = Yn + 1

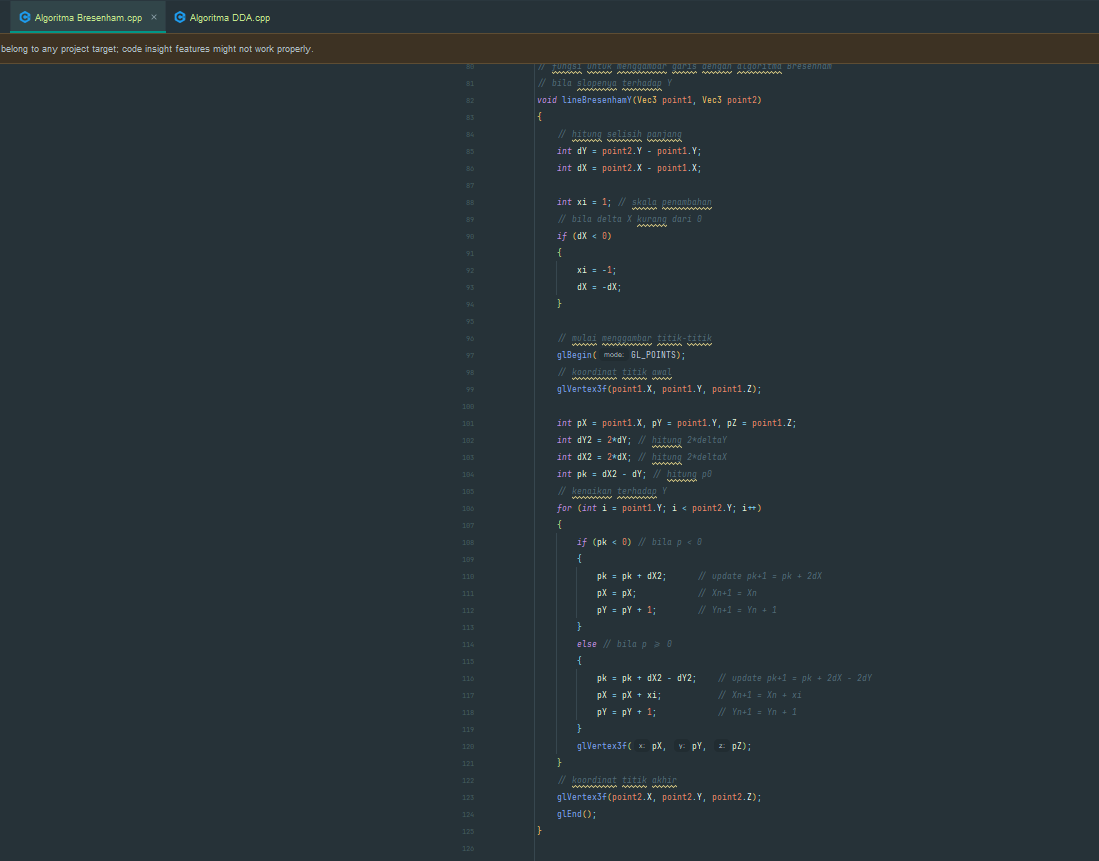
}

glVertex3f(pX, pY, pZ);

}

// koordinat titik akhir glVertex3f(point2.X, point2.Y, point2.Z); glEnd();

}



Algoritma BresenHam

1. Tambahkan fungsi lineBresenham() di file praktikum02.cpp anda untuk menggambar garis dengan algoritma Bresenham dengan memanggil fungsi di langkah 14 dan langkah 15.

// fungsi untuk menggambar garis dengan algoritma Bresenham void lineBresenham(Vec3 point1, Vec3 point2)

{

// hitung selisih panjang

int dY = point2.Y - point1.Y; int dX = point2.X - point1.X;

if (abs(dY) < abs(dX)) // bila deltaY lebih pendek dari deltaX

{

if (point1.X < point2.X) // bila X1 < X2 lineBresenhamX(point1, point2);

else // bila X1 > X2 maka dibalik lineBresenhamX(point2, point1);

}

else // bila deltaY lebih panjang dari deltaX

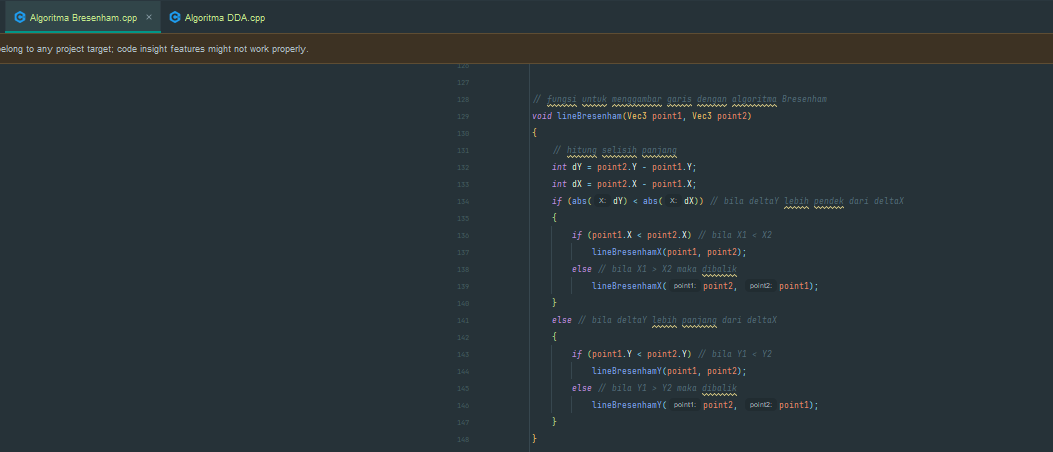
{

if (point1.Y < point2.Y) // bila Y1 < Y2 lineBresenhamY(point1, point2);

else // bila Y1 > Y2 maka dibalik lineBresenhamY(point2, point1);

}

}



1. Ubah fungsi drawObject() di praktikum02.cpp di baris kode berikut untuk menerapkan algoritma bresenham.

lineDDA(sbX1, sbX2); lineDDA(sbY1, sbY2);

menjadi

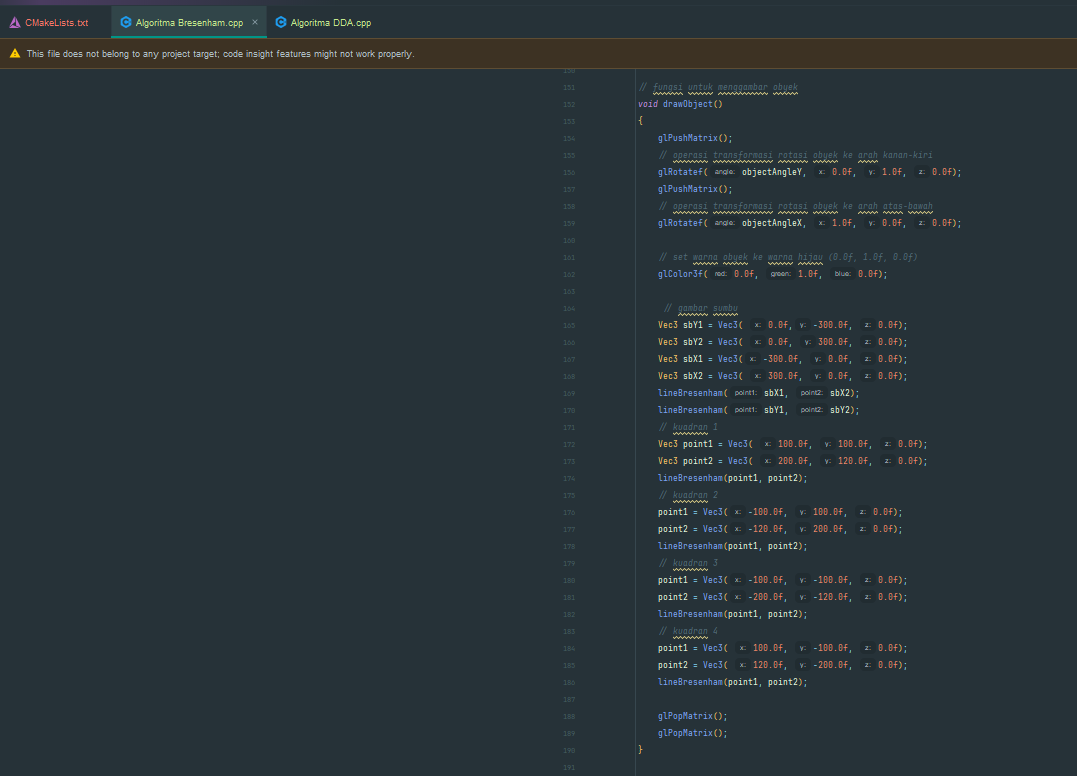
lineBresenham(sbX1, sbX2); lineBresenham(sbY1, sbY2);

dan

lineDDA(point1, point2);

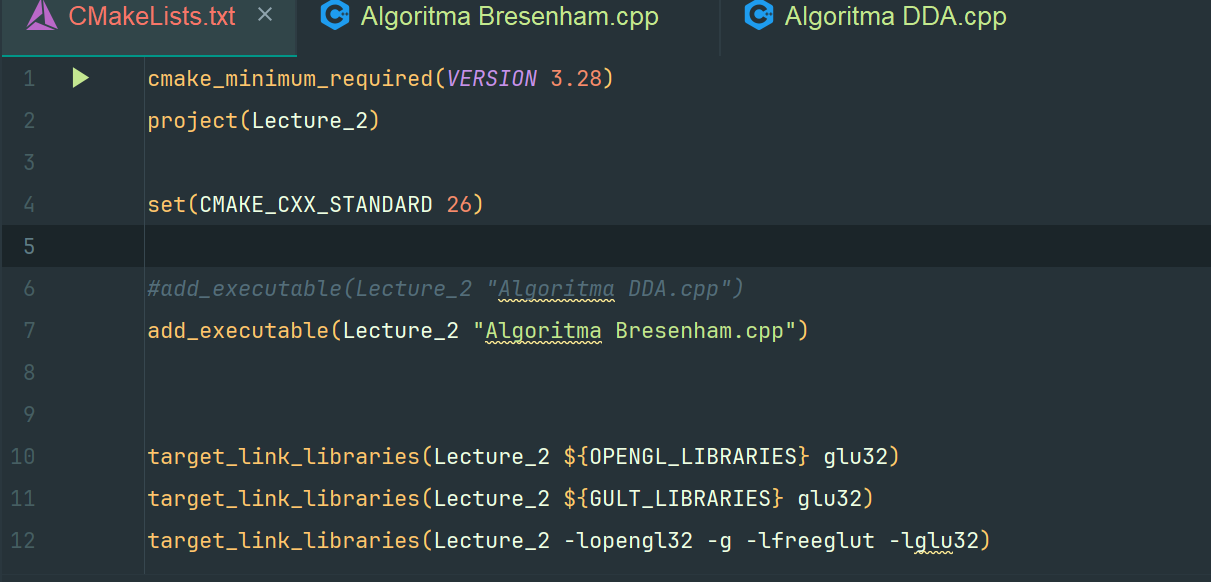
menjadi

lineBresenham(point1, point2);

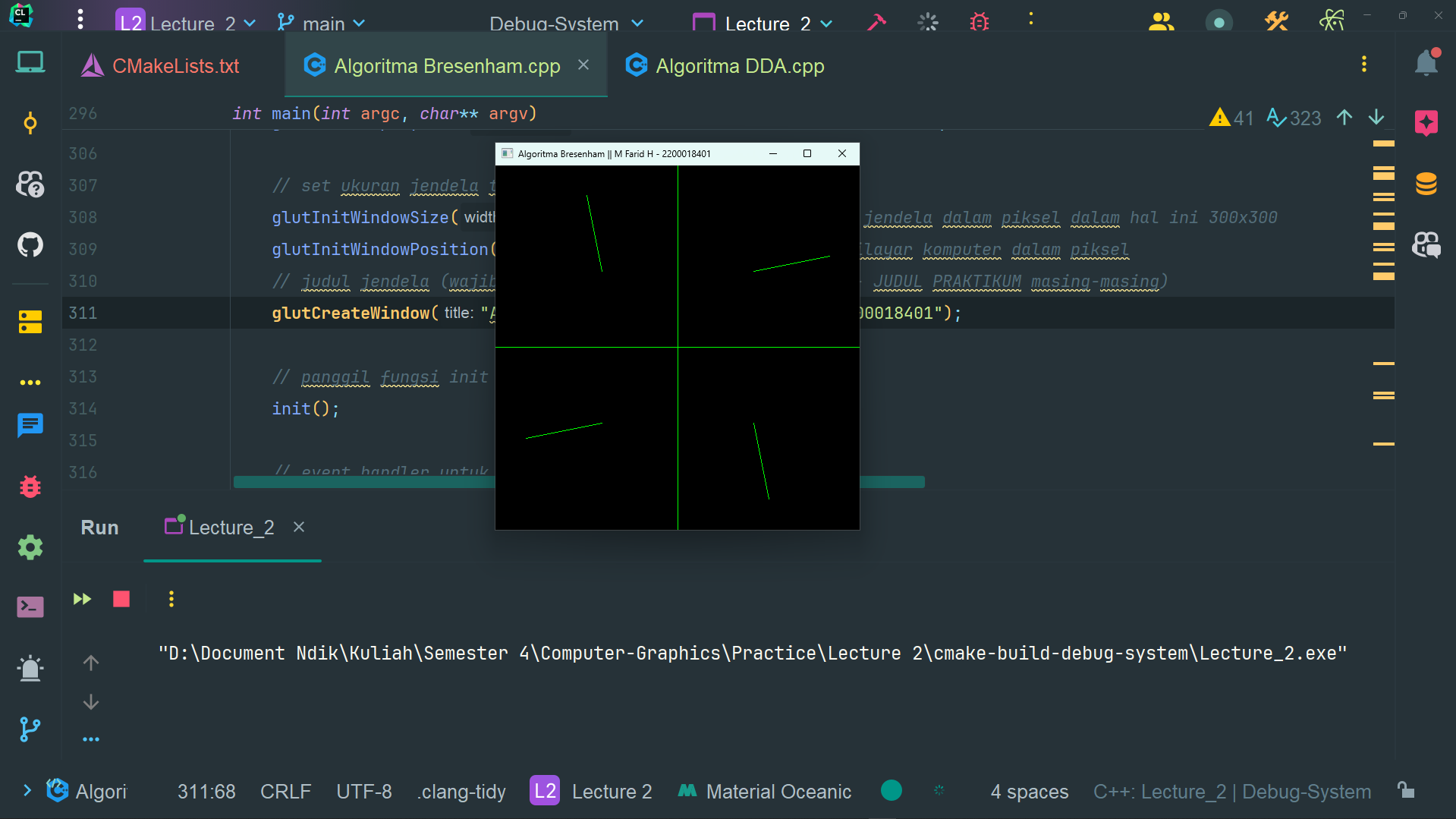


Algoritma Bresenham

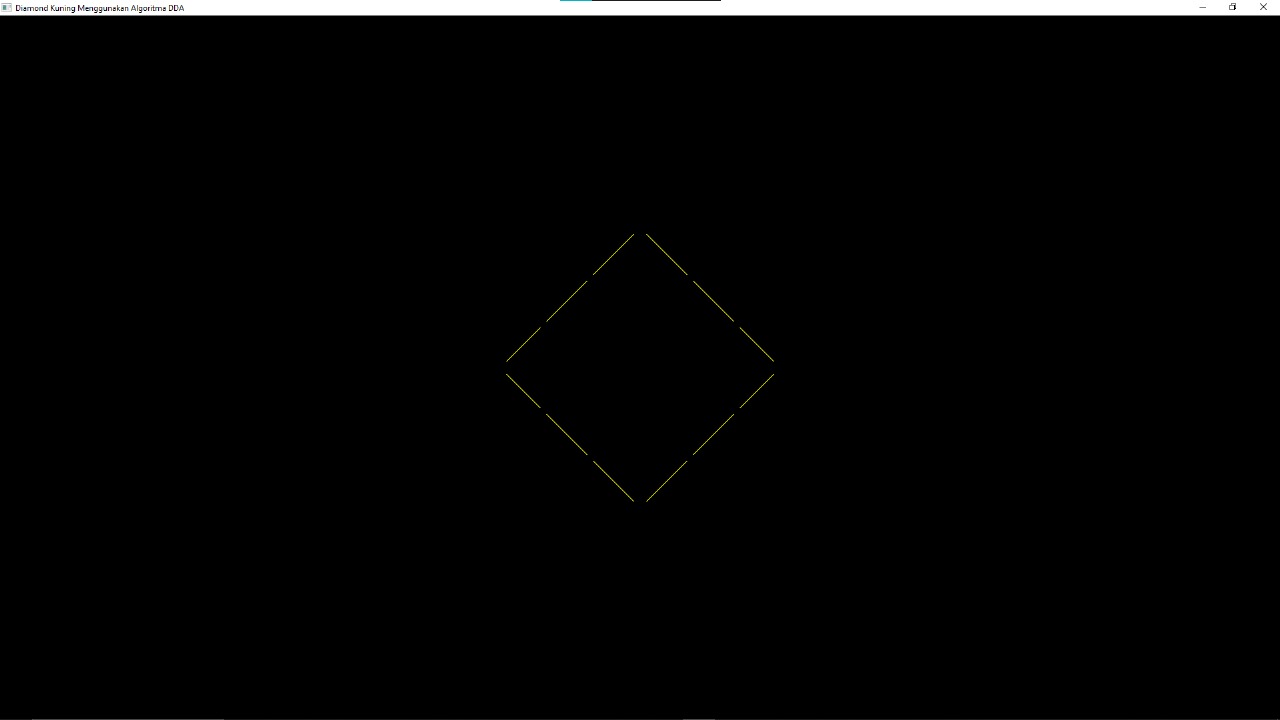
1. Jalankan program untuk melihat hasil dari pembuatan garis dengan algoritma Bresenham seperti yang ditunjukkan pada Gambar 2.2.



Komentar pada bagian Algoritma DDA, dan pilih executable bagian Algoritma Bresenham



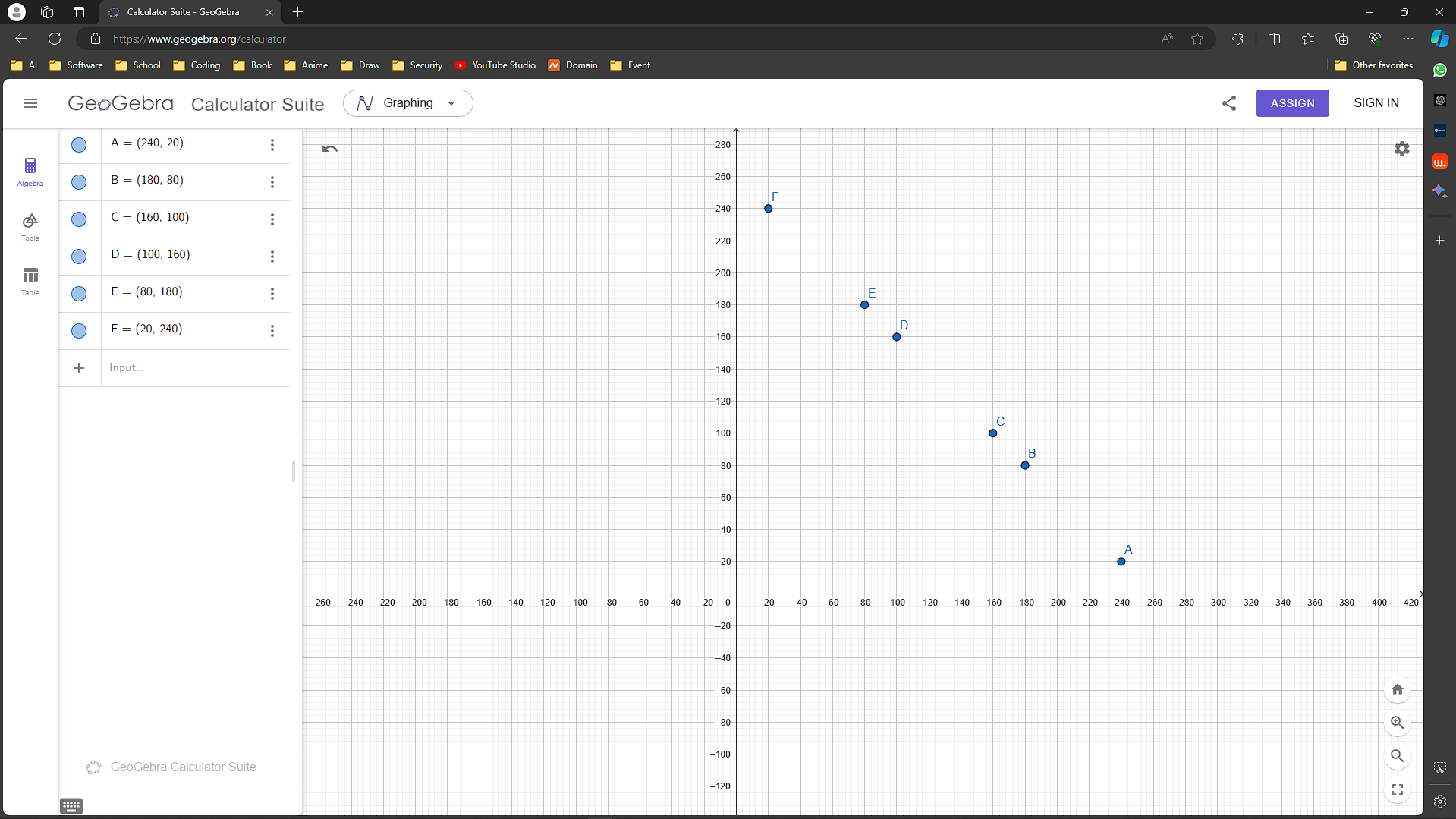
# **POST TEST**



Membuat seperti di atas

Step by Step

1. Membuat prediksi titik-titik, dapat menggunkaan Geogebra



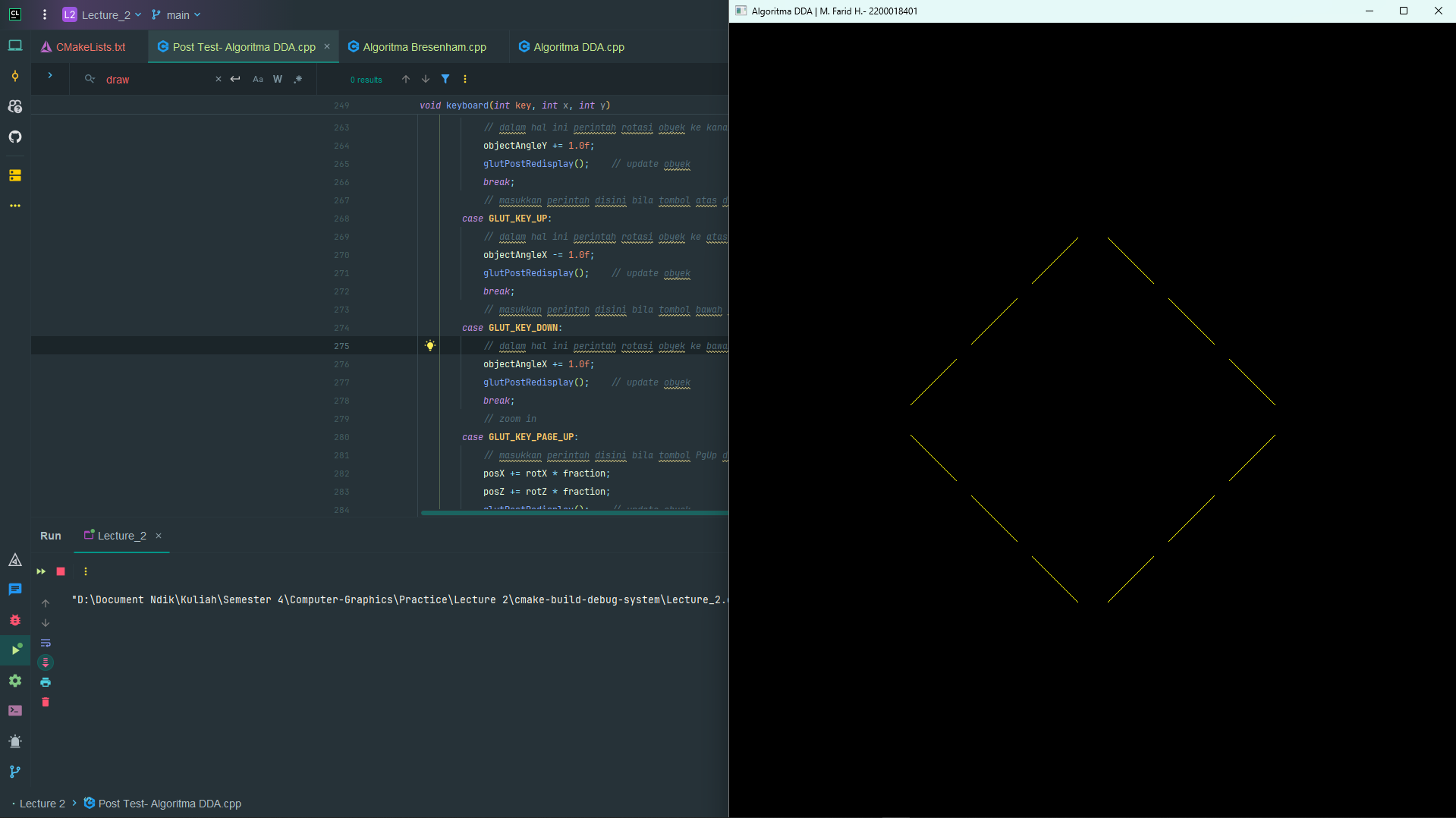
Di atas sudah membuat bagian sisi kanan atas belah ketupat, untuk melanjutkan sisi bagian lainnya, gunakan teknik mirror (pencerminan)

Jadinya Anda bisa melakukan pencerminan terhadap sumbu x, terhadap sumbu y, dll

1. Memindahkan titik ke dalam bentuk kodingan

|  |
| --- |
| *// fungsi untuk menggambar obyek void* drawObject() {  glPushMatrix();  *// operasi transformasi rotasi obyek ke arah kanan-kiri* glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);  glPushMatrix();  *// operasi transformasi rotasi obyek ke arah atas-bawah* glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);   *// warna kuning* glColor3f(1.0f, 1.0f, 0.0f);   *// belah ketupat // - SISI KANAN ATAS -  // garis 1* Vec3 point1(240,20,0);  Vec3 point2(180,80,0);   *// garis 2* Vec3 point3(160,100,0);  Vec3 point4(100,160,0);   *// garis 3* Vec3 point5(80,180,0);  Vec3 point6(20,240,0);  lineDDA(point1, point2); lineDDA(point3, point4); lineDDA(point5, point6);  *// SISI KANAN BAWAH //melakukan mirror terhadap sumbu x  // garis 1* Vec3 point7(240,-20,0);  Vec3 point8(180,-80,0);   *// garis 2* Vec3 point9(160,-100,0);  Vec3 point10(100,-160,0);   *// garis 3* Vec3 point11(80,-180,0);  Vec3 point12(20,-240,0);   lineDDA(point7, point8);  lineDDA(point9, point10);  lineDDA(point11, point12);  *// SISI KIRI ATAS // melakukan mirror terhadap sumbu y // garis 1* Vec3 point13(-240,20,0);  Vec3 point14(-180,80,0);  *// garis 2* Vec3 point15(-160,100,0);  Vec3 point16(-100,160,0);  *// garis 3* Vec3 point17(-80,180,0);  Vec3 point18(-20,240,0);   lineDDA(point13, point14);  lineDDA(point15, point16);  lineDDA(point17, point18);  *// SISI KIRI BAWAH // melakukan mirror terhadap sumbu x dan y // garis 1* Vec3 point19(-240,-20,0);  Vec3 point20(-180,-80,0);  *// garis 2* Vec3 point21(-160,-100,0);  Vec3 point22(-100,-160,0);  *// garis 3* Vec3 point23(-80,-180,0);  Vec3 point24(-20,-240,0);   lineDDA(point19, point20);  lineDDA(point21, point22);  lineDDA(point23, point24);   glPopMatrix();  glPopMatrix(); } |

1. Kemudian jalankan (menggunakan algoritma DDA)



Penjelasan ringkas kodingan:

Saya menggunakan metode merancang titik-titik untuk membangun sebuah belah ketupat, sehingga tiap 1 sisi harus memiliki 3 garis. Tiap garis disambungkan melalui 2 titik sehingga tiap sisi belah ketupat memuat 6 titik. Jadi total titik yang dimiliki belah ketupat dengan garis terputus adalah 24 titik. Lebih baik untuk menentukan jeda kosong dan garis memodifikasi pada fungsi DDA, tetapi hanya ingin menggambarkan saja, saya hanya menginisialisasikan titik-titik untuk membuat bangun tersebut

Untuk mengakses kodingan pertemuan ke-2, Anda bisa mengakses link github berikut pada folder Practice

[IRedDragonICY/Computer-Graphics (github.com)](https://github.com/IRedDragonICY/Computer-Graphics)