

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:

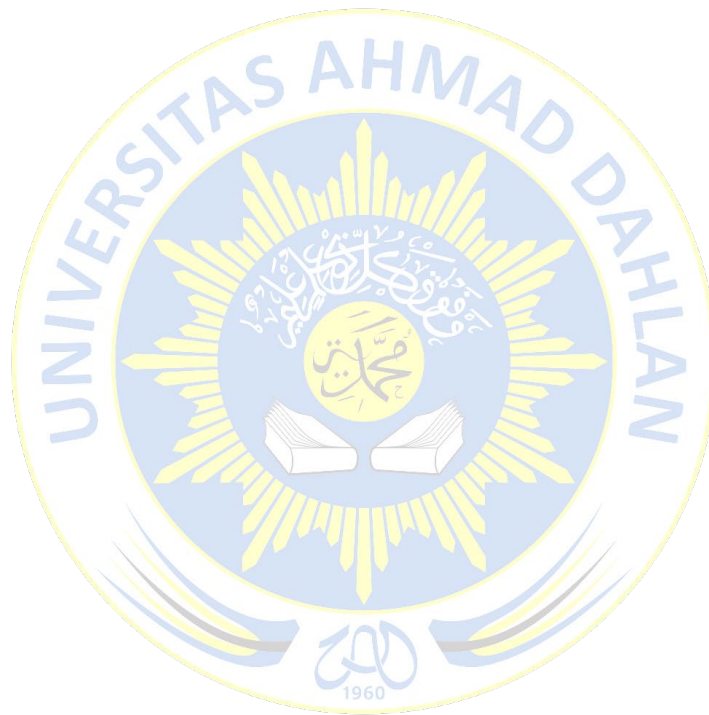
Mohammad Farid Hendianto 2200018401

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
LANGKAH PRAKTIKUM.....	3
POST TEST.....	24



LANGKAH PRAKTIKUM

Persiapan

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

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

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

Saya sudah menginstall library OpenGL melalui MSYS2 yang disediakan oleh Microsoft

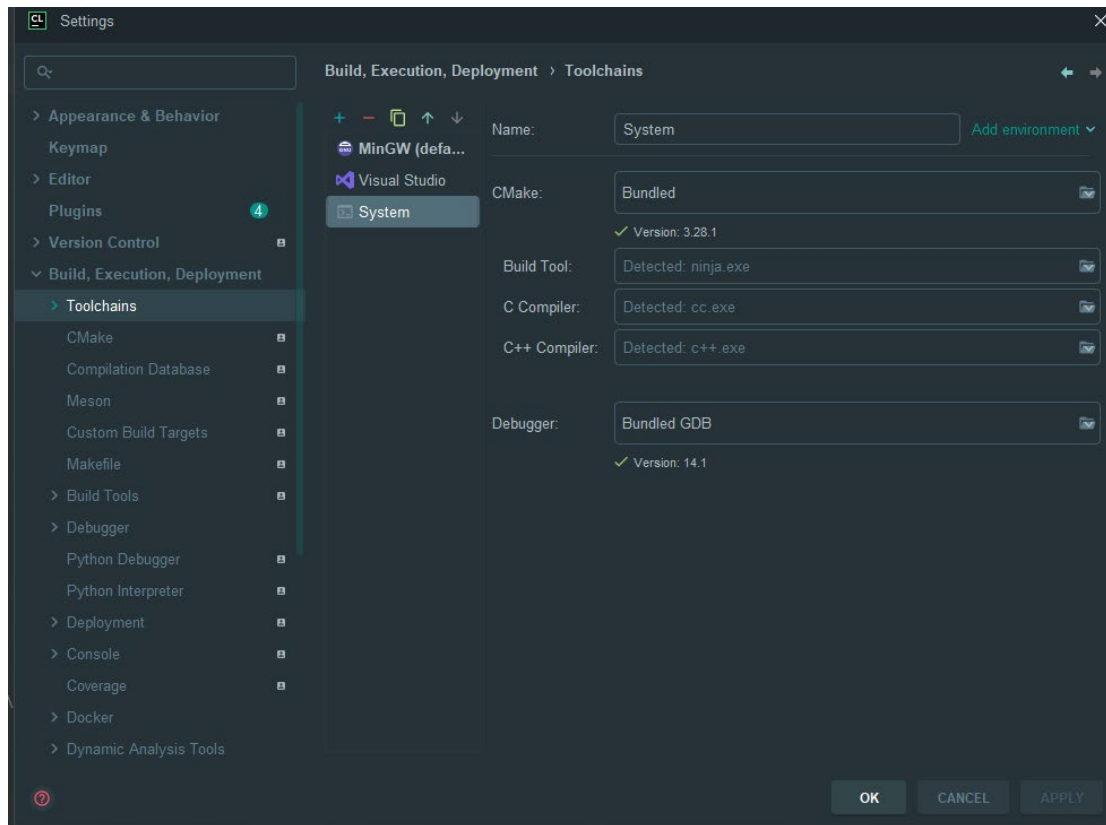
3. Ubah nama dari kode dasar “praktikum00.cpp” menjadi “praktikum02.cpp” dan copy-kan ke Source Files di project yang anda buat.



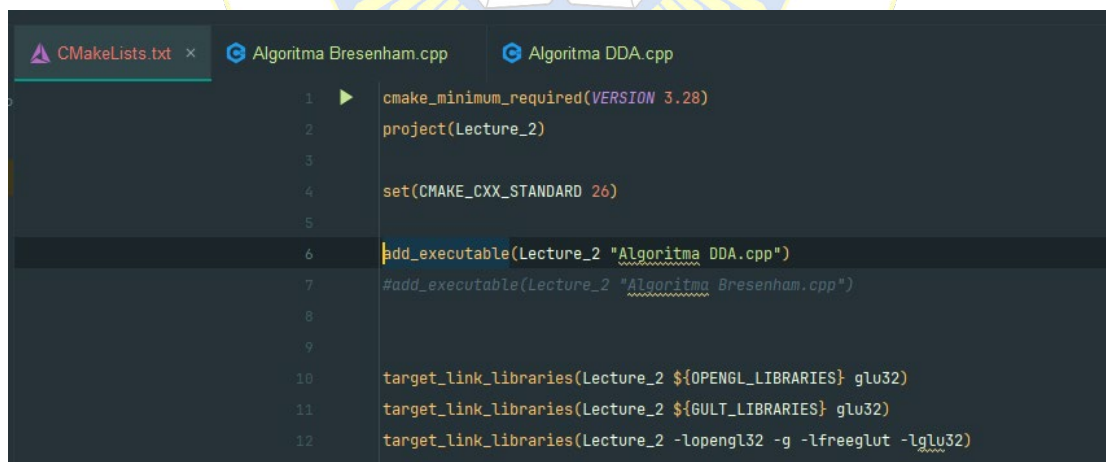
Mendownload source code yang diberikan oleh mas Rijal

4. 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:

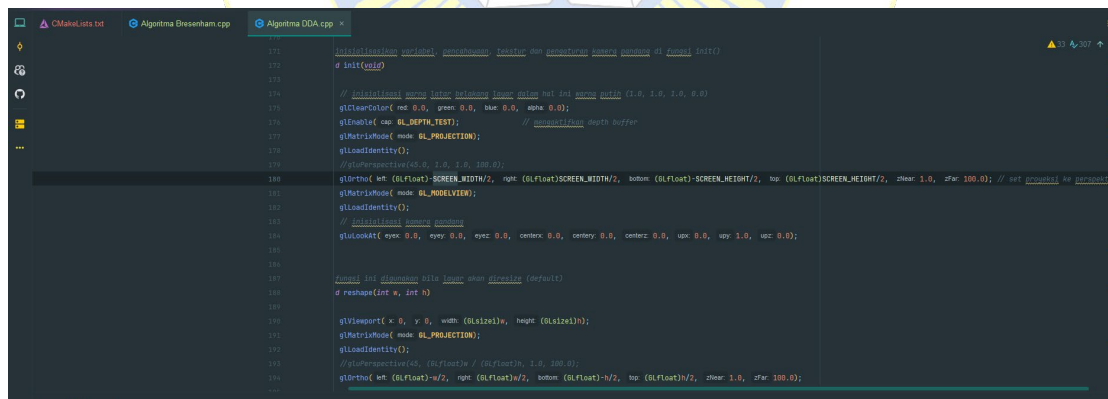
Karena ini tidak ada praktikum00 format dari Pak Adhi Prahara tetapi langsung diberikan semua source code untuk pertemuan praktikum 2 ini, maka disini kita hanya mengecek apakah sudah di ubah sesuai langkah praktikum atau belum.

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,
```



Algoritma DDA

```

119
120 //inisialisasi variabel, koneksi dan penentuan nama jendela di fungsi init()
121 d init(void)
122
123 //inisialisasi warna latar belakang menggunakan nilai rgb (1.0, 1.0, 1.0, 0.0)
124 glClearColor( red 0.0, green 0.0, blue 0.0, alpha 0.0);
125 glEnable( GL_DEPTH_TEST); // enable depth buffer
126 glMatrixMode( mode GL_PROJECTION);
127 glLoadIdentity();
128 gluPerspective(45.0, (GLfloat)w / (GLfloat)h, 1.0, 100.0);
129
130 glMatrixMode( mode GL_MODELVIEW);
131 glLoadIdentity();
132 //inisialisasi kamera standar
133 gluLookat( eyes 0.0, eyes 0.0, eyes 0.0, centerx 0.0, centery 0.0, centerz 0.0, upx 0.0, upy 1.0, upz 0.0);
134
135 //fungsi ini digunakan bila layar akan diresize (default)
136 void reshape(int w, int h)
137 {
138     glViewport( x 0, y 0, width (GLsizei)w, height (GLsizei)h);
139     glMatrixMode( mode GL_PROJECTION);
140 }
    
```

Algoritma Bresenham

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

3. 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);`

```

171 void init(void)
172 {
173     gluOrtho( w (GLfloat) SCREEN_WIDTH/2, right (GLfloat) SCREEN_WIDTH/2, bottom (GLfloat) -SCREEN_HEIGHT/2, top (GLfloat) SCREEN_HEIGHT/2, znear 1.0, zfar 100.0); // set projection ke orthographic
174     glMatrixMode( mode GL_MODELVIEW);
175     glLoadIdentity();
176     //inisialisasi kamera standar
177     gluLookat( eyes 0.0, eyes 0.0, eyes 0.0, centerx 0.0, centery 0.0, centerz 0.0, upx 0.0, upy 1.0, upz 0.0);
178 }
179
180 //fungsi ini digunakan bila layar akan diresize (default)
181 void reshape(int w, int h)
182 {
183     glViewport( x 0, y 0, width (GLsizei)w, height (GLsizei)h);
184     glMatrixMode( mode GL_PROJECTION);
185     glLoadIdentity();
186     //gluPerspective(45.0, (GLfloat)w / (GLfloat)h, 1.0, 100.0);
187     gluOrtho( w (GLfloat)-w/2, right (GLfloat)w/2, bottom (GLfloat)-h/2, top (GLfloat)h/2, znear 1.0, zfar 100.0);
188     glMatrixMode( mode GL_MODELVIEW);
189 }
190
191 //fungsi untuk menerima masukan dari keyboard
192 //fungsi untuk aksi, gerakan, atau tekan, tahan, dan tekan
193 void keyboard(int key, int x, int y)
194 {
195     float fraction = 0.1f;
196 }
    
```

Algoritma DDA

```

211 void init(void)
212 {
213     // inisialisasi kamera
214     glLoadIdentity();
215     gluLookAt(eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ);
216 }
217
218 // fungsi ini digunakan jika layar akan diresize (default)
219 void reshape(int w, int h)
220 {
221     glViewport(0, 0, w, h);
222     glMatrixMode(GL_PROJECTION);
223     glLoadIdentity();
224     // gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);
225     gluOrtho2D((GLfloat)w/2, (GLfloat)w/2, (GLfloat)h/2, (GLfloat)h/2, 1.0, 100.0);
226     glMatrixMode(GL_MODELVIEW);
227 }
228
229 // fungsi untuk mengatur mouse dari keyboard
230 // untuk mengklik, menekan, dan menggerakkan
231 void keyboard(int key, int x, int y)
232 {
233     float fraction = 0.1f;
234 }

```

Algoritma Bresenham

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

4. 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);`

```

171 void display()
172 {
173     glutSwapBuffers();
174 }
175
176 // inisialisasi variabel, parameter, tetapan dan penentuan kamera pada di fungsi init()
177 void init(void)
178 {
179     // inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)
180     glClearColor(1.0, 1.0, 1.0, 0.0);
181     glEnable(GL_DEPTH_TEST); // mengaktifkan depth buffer
182     glMatrixMode(GL_PROJECTION);
183     glLoadIdentity();

```

Algoritma DDA

```

171 void display()
172 {
173     // mengaktifkan double buffer
174     glutSwapBuffers();
175 }
176
177 // inisialisasi variabel, parameter, tetapan dan penentuan kamera pada di fungsi init()
178 void init(void)
179 {
180     // inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)
181     glClearColor(1.0, 1.0, 1.0, 0.0);
182     glEnable(GL_DEPTH_TEST); // mengaktifkan depth buffer
183     glMatrixMode(GL_PROJECTION);
184     glLoadIdentity();

```

Algoritma Bresenham

5. 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();
}
```



```

33 // fungsi untuk menggambar garis dengan algoritma DDA bila terhadap X
34 void lineDDAX(Vec3 point1, Vec3 point2)
35 {
36     // hitung gradient garis m
37     int dY = point2.Y - point1.Y;
38     int dX = point2.X - point1.X;
39     float m = (float)dY / dX;
40     float im = 1.0f/m;
41
42     // mulai menggambar titik-titik
43     glBegin( mode: GL_POINTS);
44     // koordinat titik awal
45     glVertex3f(point1.X, point1.Y, point1.Z);
46     float pX = point1.X, pY = point1.Y, pZ = point1.Z;
47
48     // kenaikan terhadap X
49     for (int i = point1.X; i < point2.X; i++)
50     {
51         pX = pX + 1; // Xn+1 = Xn + 1
52         pY = pY + m; // Yn+1 = Yn + m
53         glVertex3f( x: pX, y: pY, z: pZ);
54     }
55     // koordinat titik akhir
56     glVertex3f(point2.X, point2.Y, point2.Z);
57     glEnd();
58 }

```

Algoritma DDA

6. 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; //  $X_{n+1} = X_n + 1/m$ 
        pY = pY + 1; //  $Y_{n+1} = Y_n + 1$ 
        glVertex3f(pX, pY, pZ);
    }

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

```

60 // fungsi untuk menggambar garis dengan algoritma DDA bila terhadap Y
61 void lineDDAY(Vec3 point1, Vec3 point2)
62 {
63     // hitung gradient garis m
64     int dY = point2.Y - point1.Y;
65     int dX = point2.X - point1.X;
66     float m = (float)dY / dX;
67     float im = 1.0f/m;
68
69     // mulai menggambar titik-titik
70     glBegin( mode: GL_POINTS);
71     // koordinat titik awal
72     glVertex3f(point1.X, point1.Y, point1.Z);
73     float pX = point1.X, pY = point1.Y, pZ = point1.Z;
74
75     // kenaikan terhadap Y
76     for (int i = point1.Y; i < point2.Y; i++)
77     {
78         pX = pX + im; //  $X_{n+1} = X_n + 1/m$ 
79         pY = pY + 1; //  $Y_{n+1} = Y_n + 1$ 
80         glVertex3f( x: pX, y: pY, z: pZ);
81     }
82     // koordinat titik akhir
83     glVertex3f(point2.X, point2.Y, point2.Z);
84     glEnd();
85 }

```

1968
Algoritma DDA

7. 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);
    }
}
```

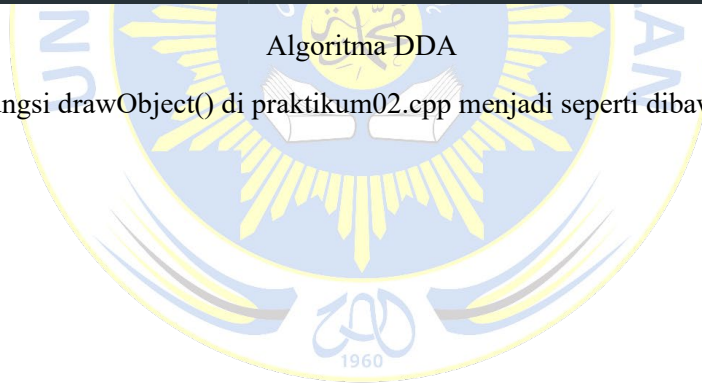
```

87 // fungsi untuk menggambar garis dengan algoritma DDA
88 void lineDDA(Vec3 point1, Vec3 point2)
89 {
90     // hitung selisih panjang
91     int dY = point2.Y - point1.Y;
92     int dX = point2.X - point1.X;
93     // bila deltaY lebih pendek dari deltaX
94     if (abs(X: dY) < abs(X: dX))
95     {
96         if (point1.X < point2.X) // bila X1 < X2
97             lineDDAX(point1, point2);
98         else // bila X1 > X2 maka dibalik
99             lineDDAX( point1: point2, point2: point1);
100     }
101     else // bila deltaY lebih panjang dari deltaX
102     {
103         if (point1.Y < point2.Y) // bila Y1 < Y2
104             lineDDAY(point1, point2);
105         else // bila Y1 > Y2 maka dibalik
106             lineDDAY( point1: point2, point2: point1);
107     }
108 }
109

```

Algoritma DDA

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



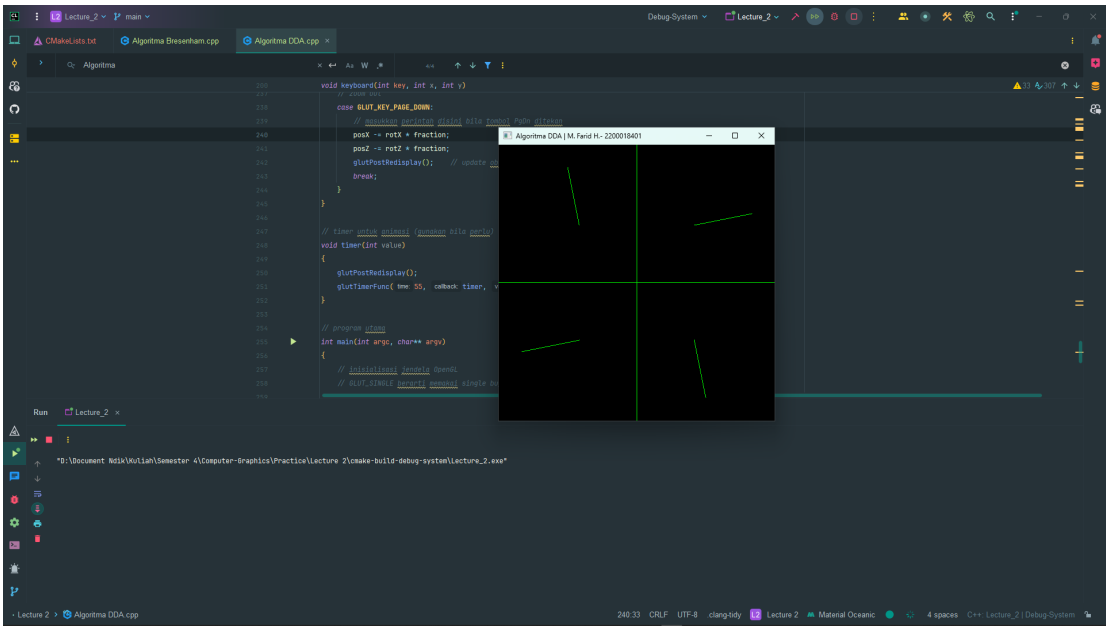
```
// fungsi untuk menggambar obyek
void drawObject()
{
    glPushMatrix();
    // operasi transformasi rotasi obyek ke arah kanan-kiri
    glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);
    glPushMatrix();
    // operasi transformasi rotasi obyek ke arah atas-bawah
    glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

    // set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)
    glColor3f(0.0f, 1.0f, 0.0f);

    // gambar sumbu
    Vec3 sbY1 = Vec3( 0.0f,-300.0f, 0.0f);
    Vec3 sbY2 = Vec3( 0.0f, 300.0f, 0.0f);
    Vec3 sbX1 = Vec3(-300.0f, 0.0f, 0.0f);
    Vec3 sbX2 = Vec3( 300.0f, 0.0f, 0.0f);
    lineDDA(sbX1, sbX2);
    lineDDA(sbY1, sbY2);
    // kuadran 1
    Vec3 point1 = Vec3( 100.0f, -100.0f, 0.0f);
    Vec3 point2 = Vec3( 200.0f, 120.0f, 0.0f);
    lineDDA(point1, point2);
    // kuadran 2
    point1 = Vec3(-100.0f, 100.0f, 0.0f);
    point2 = Vec3(-120.0f, 200.0f, 0.0f);
    lineDDA(point1, point2);
    // kuadran 3
    point1 = Vec3(-100.0f, -100.0f, 0.0f);
    point2 = Vec3(-200.0f, -120.0f, 0.0f);
    lineDDA(point1, point2);
    // kuadran 4
    point1 = Vec3( 100.0f, -100.0f, 0.0f);
    point2 = Vec3( 120.0f, -200.0f, 0.0f);
    lineDDA(point1, point2);

    glPopMatrix();
    glPopMatrix();
}
```

9. 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

10. 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; // update pk+1 = pk + 2dY - 2dX
    }
    glVertex3f(pX, pY, pZ);
}
}
}

```

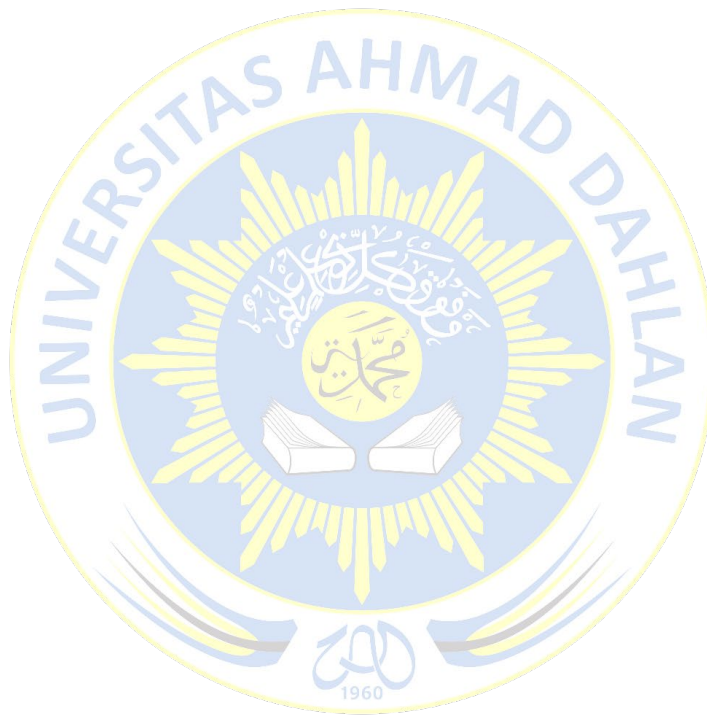
```

31 // fungsi untuk menggambar garis dengan algoritma Bresenham
32 // bila slope terhadap X
33 void lineBresenham(Vec3 point1, Vec3 point2)
34 {
35     // hitung selisih panjang
36     int dY = point2.Y - point1.Y;
37     int dX = point2.X - point1.X;
38
39     int yi = 1; // skala penambahan
40     // bila delta Y kurang dari 0
41     if (dY < 0)
42     {
43         yi = -1;
44         dY = -dY;
45     }
46
47     // mulai menambahkan titik titik
48     glBegin( GL_POINTS);
49     // koordinat titik awal
50     glVertex3f(point1.X, point1.Y, point1.Z);
51
52     int pX = point1.X, pY = point1.Y, pZ = point1.Z;
53     int dY2 = 2*dY; // hitung 2*deltaY
54     int dX2 = 2*dX; // hitung 2*deltaX
55     int pk = dY2 - dX; // hitung p0
56     // kenaikan terhadap X
57     for (int i = point1.X; i < point2.X; i++)
58     {
59         if (pk < 0) // bila p < 0
60         {
61             pk = pk + dY2; // update pk+1 = pk + 2dY
62             pX = pX + 1; // Xn+1 = Xn + 1
63             pY = pY; // Yn+1 = Yn
64         }
65         else // bila p >= 0
66         {
67             pk = pk + dY2 - dX2; // update pk+1 = pk + 2dY - 2dX
68             pX = pX + 1; // Xn+1 = Xn + 1
69             pY = pY + yi; // Yn+1 = Yn + yi
70         }
71         glVertex3f( pX, pY, pZ);
72     }
73     // koordinat titik akhir
74     glVertex3f(point2.X, point2.Y, point2.Z);
75     glEnd();
76 }

```

Algoritma BresenHam

11. 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();
}
```

```

Algorithm Breshenham.cpp x Algorithm DDA.cpp
belong to any project target; code insight features might not work properly.

// Fungsi untuk menggambar garis dengan algoritma Bresenham
// bisa digunakan untuk menggambar
void lineBreshenham(Vec3 point1, Vec3 point2)
{
    // Hitung vektor arah
    int dY = point2.Y - point1.Y;
    int dX = point2.X - point1.X;

    int xi = 1; // skala normalisasi
    // bila delta X kurang dari 0
    if (dX < 0)
    {
        xi = -1;
        dX = -dX;
    }

    // hitung parameter 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 2deltaY
    int dX2 = 2*dX; // hitung 2deltaX
    int pk = dX2 - dY; // hitung p0
    // hitung langkah 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( x: pX, y: pY, z: pZ);
    }

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

```

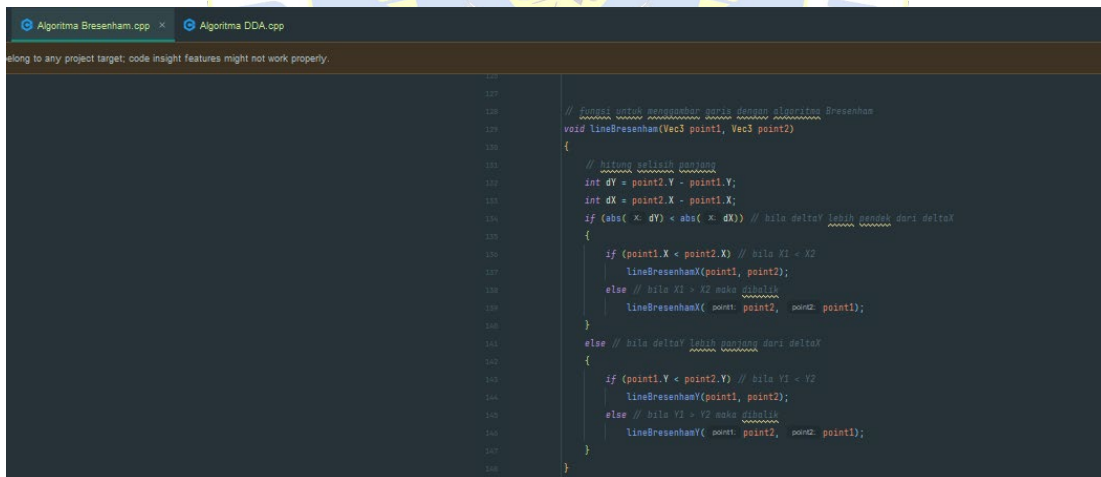


Algoritma BresenHam

12. 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
```



```
120 // fungsi untuk menggambar garis dengan algoritma Bresenham
121 void lineBresenham(Vec3 point1, Vec3 point2)
122 {
123     // hitung selisih panjang
124     int dY = point2.Y - point1.Y;
125     int dX = point2.X - point1.X;
126     if (abs(dY) < abs(dX)) // bila deltaY lebih pendek dari deltaX
127     {
128         if (point1.X < point2.X) // bila X1 < X2
129             lineBresenhamX(point1, point2);
130         else // bila X1 > X2 maka dibalik
131             lineBresenhamX(point2, point1);
132     }
133     else // bila deltaY lebih panjang dari deltaX
134     {
135         if (point1.Y < point2.Y) // bila Y1 < Y2
136             lineBresenhamY(point1, point2);
137         else // bila Y1 > Y2 maka dibalik
138             lineBresenhamY(point2, point1);
139     }
140 }
```

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

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

menjadi

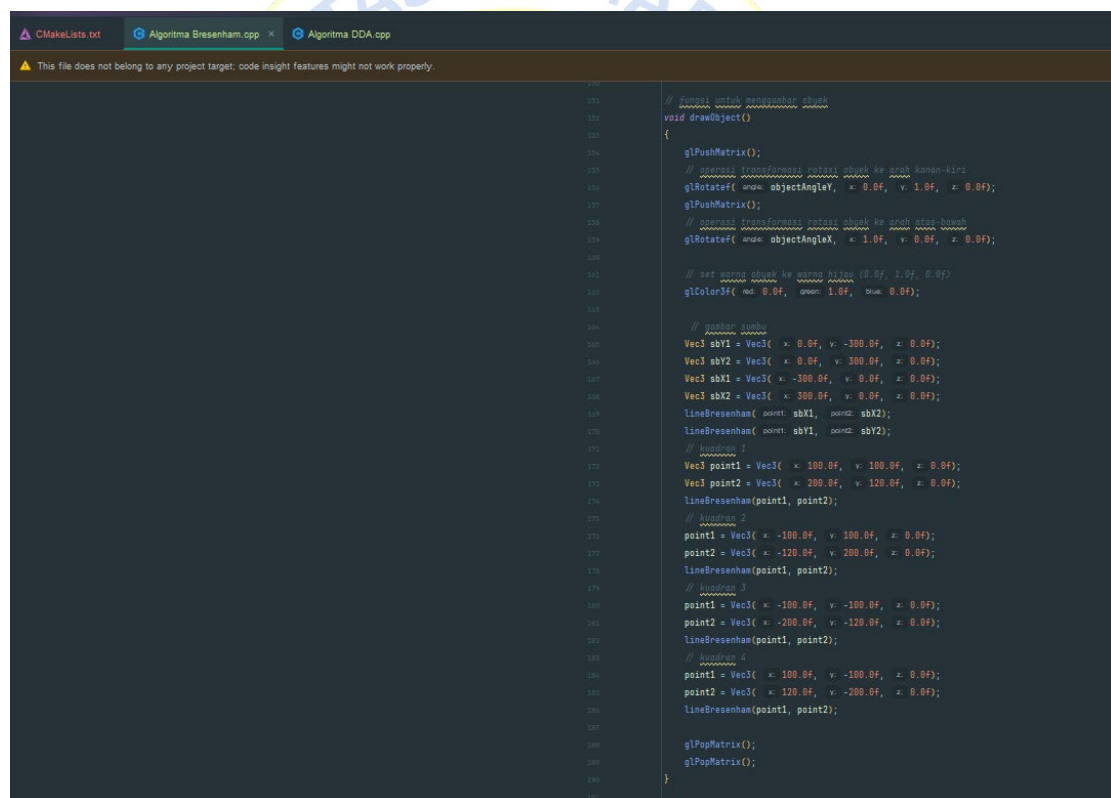
```
lineBresenham(sbX1, sbX2);
lineBresenham(sbY1, sbY2);
```

dan

```
lineDDA(point1, point2);
```

menjadi

```
lineBresenham(point1, point2);
```



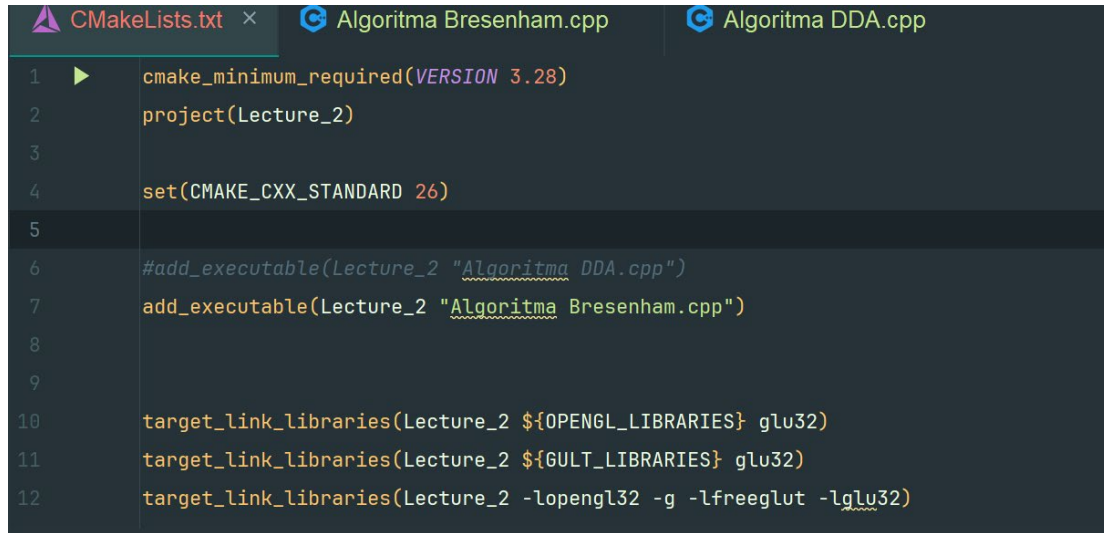
```

128
129
130 // fungsi untuk menggambar objek
131 void drawObject()
132 {
133     glPushMatrix();
134     // operasi transformasi rotasi objek ke arah kanan-kiri
135     glRotatef(www_objectAngleY, x, 0.0f, y, 1.0f, z, 0.0f);
136     glPushMatrix();
137     // operasi transformasi rotasi objek ke arah atas-bawah
138     glRotatef(www_objectAngleX, x, 1.0f, y, 0.0f, z, 0.0f);
139
140     // set warna objek ke warna hitam (0.0f, 1.0f, 0.0f)
141     glColor3f(red, 0.0f, green, 1.0f, blue, 0.0f);
142
143     // gambar kubus
144     Vec3 sbY1 = Vec3(x, 0.0f, y, -300.0f, z, 0.0f);
145     Vec3 sbY2 = Vec3(x, 0.0f, y, 300.0f, z, 0.0f);
146     Vec3 sbX1 = Vec3(x, -300.0f, y, 0.0f, z, 0.0f);
147     Vec3 sbX2 = Vec3(x, 300.0f, y, 0.0f, z, 0.0f);
148     lineBresenham(point1, sbY1, point2, sbY2);
149     lineBresenham(point1, sbY2, point2, sbY2);
150
151     // kubus 1
152     Vec3 point1 = Vec3(x, 100.0f, y, 100.0f, z, 0.0f);
153     Vec3 point2 = Vec3(x, 200.0f, y, 120.0f, z, 0.0f);
154     lineBresenham(point1, point2);
155
156     // kubus 2
157     point1 = Vec3(x, -100.0f, y, 100.0f, z, 0.0f);
158     point2 = Vec3(x, -120.0f, y, 200.0f, z, 0.0f);
159     lineBresenham(point1, point2);
160
161     // kubus 3
162     point1 = Vec3(x, -100.0f, y, -100.0f, z, 0.0f);
163     point2 = Vec3(x, -200.0f, y, -120.0f, z, 0.0f);
164     lineBresenham(point1, point2);
165
166     // kubus 4
167     point1 = Vec3(x, 100.0f, y, -100.0f, z, 0.0f);
168     point2 = Vec3(x, 120.0f, y, -200.0f, z, 0.0f);
169     lineBresenham(point1, point2);
170
171     glPopMatrix();
172     glPopMatrix();
173 }
174

```

Algoritma Bresenham

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

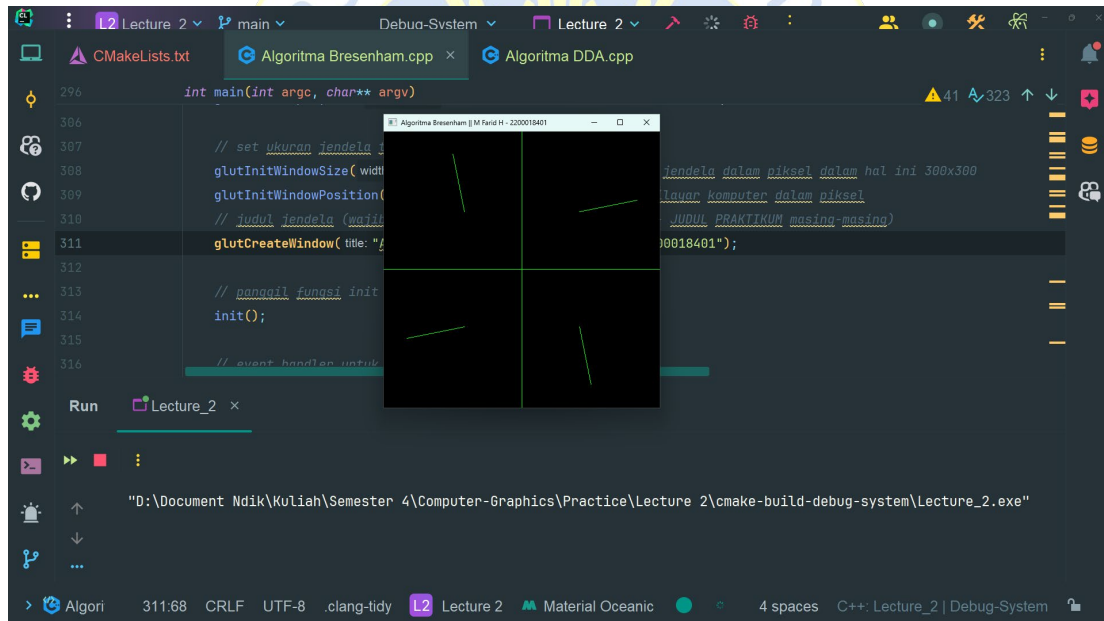


```

1 cmake_minimum_required(VERSION 3.28)
2 project(Lecture_2)
3
4 set(CMAKE_CXX_STANDARD 26)
5
6 #add_executable(Lecture_2 "Algoritma DDA.cpp")
7 add_executable(Lecture_2 "Algoritma Bresenham.cpp")
8
9
10 target_link_libraries(Lecture_2 ${OPENGL_LIBRARIES} glu32)
11 target_link_libraries(Lecture_2 ${GLUT_LIBRARIES} glu32)
12 target_link_libraries(Lecture_2 -lopengl32 -g -lfreetgl -lglu32)

```

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



```

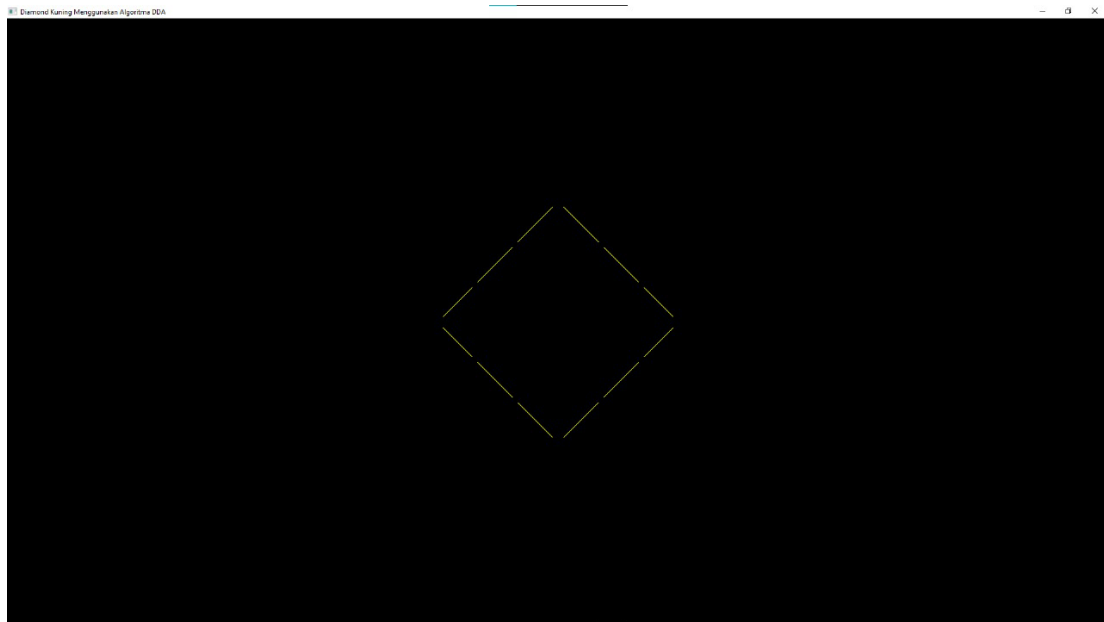
296 int main(int argc, char** argv)
306
307 // set ukuran jendela
308 glutInitWindowSize(300, 300)
309 glutInitWindowPosition(0, 0)
310 // judul jendela (wajib)
311 glutCreateWindow("JUDUL PRAKTIKUM masing-masing");
312
313 // panggil fungsi init
314 init();
315
316 // event handler untuk

```

Run Lecture_2 x

"D:\Document Ndik\Kuliah\Semester 4\Computer-Graphics\Practice\Lecture 2\cmake-build-debug-system\Lecture_2.exe"

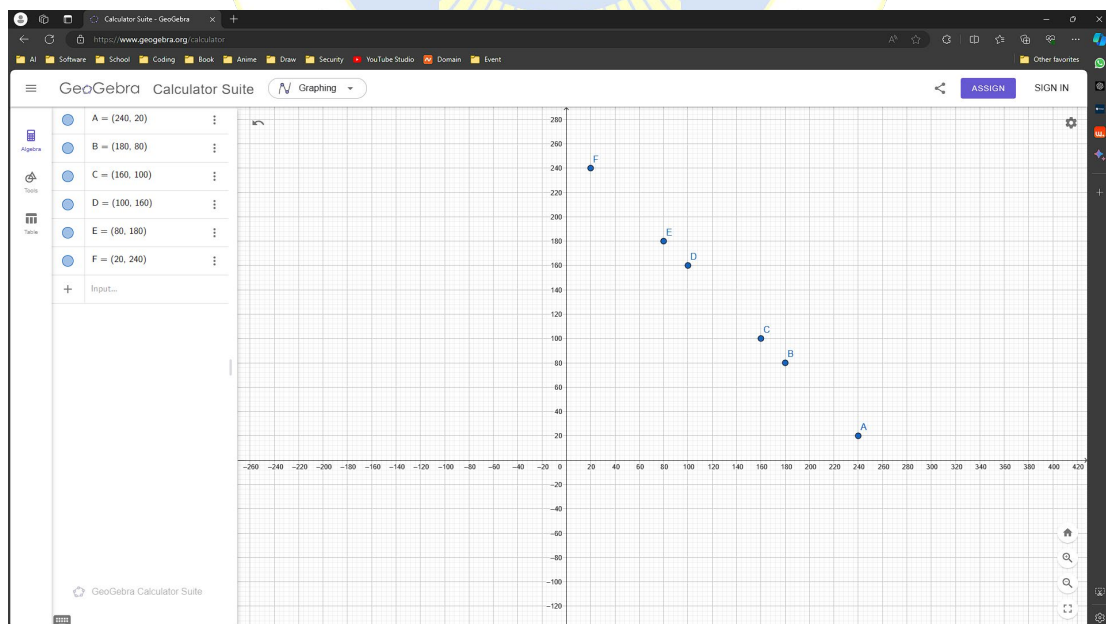
POST TEST



Membuat seperti di atas

Step by Step

1. Membuat prediksi titik-titik, dapat menggunakan 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

2. 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);

// SISTI 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);

// SISTI 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);

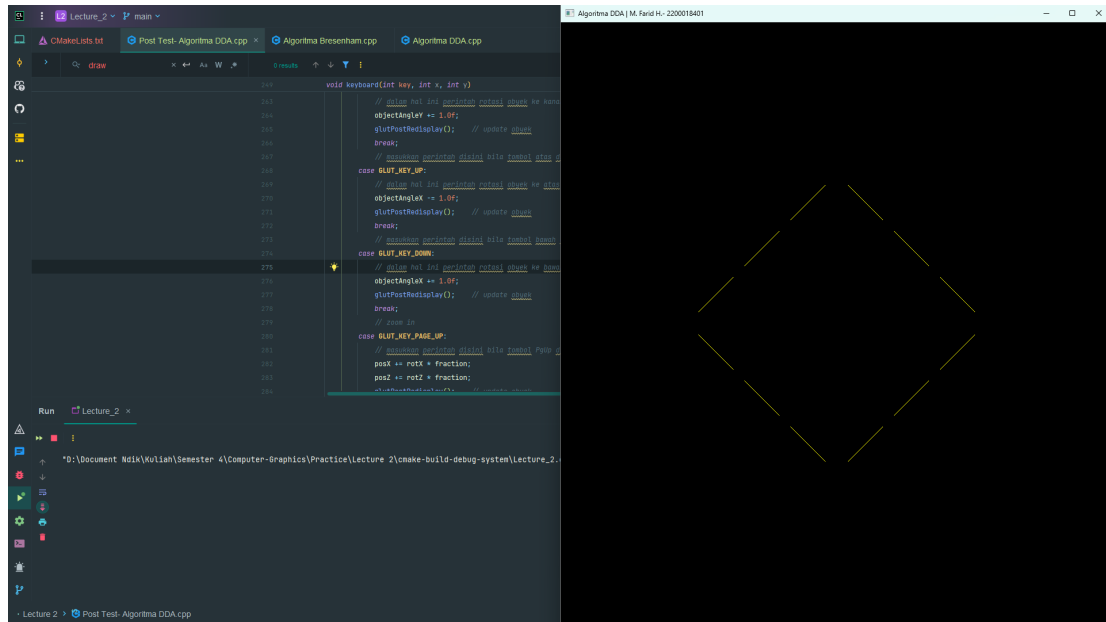
// garis 3
    Vec3 point23(-80,-180,0);
    Vec3 point24(-20,-240,0);

    lineDDA(point19, point20);
    lineDDA(point21, point22);
    lineDDA(point23, point24);

    glPopMatrix();
    glPopMatrix();
}

```

3. 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

[IRedDragon1CY/Computer-Graphics \(github.com\)](https://github.com/IRedDragon1CY/Computer-Graphics)

