UAS
KOMPUTER GRAFIK (A)
PROJEK BOLA MEMANTUL

DISUSUN OLEH:

Rezki anwar - 4520210033

Teknik Informatika Universitas Pancasila

Tahun ajaran 2022/2023

SourceCode:

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glut.h>


#include <math.h>

#define GL_CLAMP_TO_EDGE 0x812F

//speed
static GLfloat move = -100, height = 200, position=height, v=1, a=-2; //v: velocity, a:
acceleration


#define NUM_TEXTURES    2
GLuint  textureObjects[NUM_TEXTURES];

const char *szTextureFiles[] = {"Wood.tga", "worldmap.tga"};

#pragma pack(1)
typedef struct
{
   GLbyte identsize;              // Size of ID field that follows header (0)
   GLbyte colorMapType;           // 0 = None, 1 = paletted
   GLbyte imageType;              // 0 = none, 1 = indexed, 2 = rgb, 3 = grey, +8=rle
   unsigned short colorMapStart;          // First colour map entry
   unsigned short colorMapLength;         // Number of colors
   unsigned char  colorMapBits;   // bits per palette entry
   unsigned short xstart;                 // image x origin
   unsigned short ystart;                 // image y origin
   unsigned short width;                  // width in pixels
   unsigned short height;                 // height in pixels
   GLbyte bits;                   // bits per pixel (8 16, 24, 32)
   GLbyte descriptor;             // image descriptor
} TGAHEADER;
#pragma pack(8)
```

```
////////////////////////////////////////////////////////////////////
// Capture the current viewport and save it as a targa file.
// Be sure and call SwapBuffers for double buffered contexts or
// glFinish for single buffered contexts before calling this function.
// Returns 0 if an error occurs, or 1 on success.
GLint gltWriteTGA(const char *szFileName)
 {
   FILE *pFile;               // File pointer
   TGAHEADER tgaHeader;  // TGA file header
   unsigned long lImageSize;   // Size in bytes of image
   GLbyte *pBits = NULL;      // Pointer to bits
   GLint iViewport[4];         // Viewport in pixels
   GLenum lastBuffer;          // Storage for the current read buffer setting

 // Get the viewport dimensions
 glGetIntegerv(GL_VIEWPORT, iViewport);

   // How big is the image going to be (targas are tightly packed)
 lImageSize = iViewport[2] * 3 * iViewport[3];

   // Allocate block. If this doesn't work, go home
   pBits = (GLbyte *)malloc(lImageSize);
   if(pBits == NULL)
      return 0;

   // Read bits from color buffer
   glPixelStorei(GL_PACK_ALIGNMENT, 1);
 glPixelStorei(GL_PACK_ROW_LENGTH, 0);
 glPixelStorei(GL_PACK_SKIP_ROWS, 0);
 glPixelStorei(GL_PACK_SKIP_PIXELS, 0);

   // Get the current read buffer setting and save it. Switch to
   // the front buffer and do the read operation. Finally, restore
   // the read buffer state
   glGetIntegerv(GL_READ_BUFFER, (GLint *)&lastBuffer);
   glReadBuffer(GL_FRONT);

   glReadBuffer(lastBuffer);

   // Initialize the Targa header
   tgaHeader.identsize = 0;
   tgaHeader.colorMapType = 0;
   tgaHeader.imageType = 2;
```

```
    tgaHeader.colorMapStart = 0;
    tgaHeader.colorMapLength = 0;
    tgaHeader.colorMapBits = 0;
    tgaHeader.xstart = 0;
    tgaHeader.ystart = 0;
    tgaHeader.width = iViewport[2];
    tgaHeader.height = iViewport[3];
    tgaHeader.bits = 24;
    tgaHeader.descriptor = 0;

    // Attempt to open the file
    pFile = fopen(szFileName, "wb");
    if(pFile == NULL)
  {
        free(pBits);    // Free buffer and return error
        return 0;
  }

    // Write the header
    fwrite(&tgaHeader, sizeof(TGAHEADER), 1, pFile);

    // Write the image data
    fwrite(pBits, lImageSize, 1, pFile);

    // Free temporary buffer and close the file
    free(pBits);
    fclose(pFile);

    // Success!
    return 1;
  }


////////////////////////////////////////////////////////////////////
// Allocate memory and load targa bits. Returns pointer to new buffer,
// height, and width of texture, and the OpenGL format of data.
// Call free() on buffer when finished!
// This only works on pretty vanilla targas... 8, 24, or 32 bit color
// only, no palettes, no RLE encoding.
GLbyte *gltLoadTGA(const char *szFileName, GLint *iWidth, GLint *iHeight, GLint
*iComponents, GLenum *eFormat)
  {
    FILE *pFile;   // File pointer
    TGAHEADER tgaHeader;  // TGA file header
```

```
 unsigned long lImageSize;  // Size in bytes of image
 short sDepth;   // Pixel depth;
 GLbyte *pBits = NULL;          // Pointer to bits

 // Default/Failed values
 *iWidth = 0;
 *iHeight = 0;

 *iComponents = GL_RGB8;

 // Attempt to open the file
 pFile = fopen(szFileName, "rb");
 if(pFile == NULL)
    return NULL;

 // Read in header (binary)
 fread(&tgaHeader, 18/* sizeof(TGAHEADER)*/, 1, pFile);

 // Get width, height, and depth of texture
 *iWidth = tgaHeader.width;
 *iHeight = tgaHeader.height;
 sDepth = tgaHeader.bits / 8;

 // Put some validity checks here. Very simply, I only understand
 // or care about 8, 24, or 32 bit targa's.
 if(tgaHeader.bits != 8 && tgaHeader.bits != 24 && tgaHeader.bits != 32)
    return NULL;

 // Calculate size of image buffer
 lImageSize = tgaHeader.width * tgaHeader.height * sDepth;

 // Allocate memory and check for success
 pBits = (GLbyte*)malloc(lImageSize * sizeof(GLbyte));
 if(pBits == NULL)
    return NULL;

 // Read in the bits
 // Check for read error. This should catch RLE or other
 // weird formats that I don't want to recognize
 if(fread(pBits, lImageSize, 1, pFile) != 1)
{
    free(pBits);
    return NULL;
}
```

```
    // Set OpenGL format expected
    switch(sDepth)
  {
      case 3:     // Most likely case

          *iComponents = GL_RGB8;
          break;
      case 4:

          *iComponents = GL_RGBA8;
          break;
      case 1:
          *eFormat = GL_LUMINANCE;
          *iComponents = GL_LUMINANCE8;
          break;
  };


    // Done with File
    fclose(pFile);

    // Return pointer to image data
    return pBits;
  }

// For best results, put this in a display list
// Draw a sphere at the origin
void gltDrawSphere(GLfloat fRadius, GLint iSlices, GLint iStacks)
    {
    GLfloat drho = (GLfloat)(3.141592653589) / (GLfloat) iStacks;
    GLfloat dtheta = 2.0f * (GLfloat)(3.141592653589) / (GLfloat) iSlices;
 GLfloat ds = 1.0f / (GLfloat) iSlices;
 GLfloat dt = 1.0f / (GLfloat) iStacks;
 GLfloat t = 1.0f;
 GLfloat s = 0.0f;
    GLint i, j;     // Looping variables

 for (i = 0; i < iStacks; i++)
  {
  GLfloat rho = (GLfloat)i * drho;
  GLfloat srho = (GLfloat)(sin(rho));
  GLfloat crho = (GLfloat)(cos(rho));
  GLfloat srhodrho = (GLfloat)(sin(rho + drho));
```

```
 GLfloat crhodrho = (GLfloat)(cos(rho + drho));

      // Many sources of OpenGL sphere drawing code uses a triangle fan
      // for the caps of the sphere. This however introduces texturing


      // artifacts at the poles on some OpenGL implementations
 glBegin(GL_TRIANGLE_STRIP);
      s = 0.0f;
 for ( j = 0; j <= iSlices; j++)
  {
  GLfloat theta = (j == iSlices) ? 0.0f : j * dtheta;
  GLfloat stheta = (GLfloat)(-sin(theta));
  GLfloat ctheta = (GLfloat)(cos(theta));

  GLfloat x = stheta * srho;
  GLfloat y = ctheta * srho;
  GLfloat z = crho;

          glTexCoord2f(s, t);
          glNormal3f(x, y, z);
          glVertex3f(x * fRadius, y * fRadius, z * fRadius);

          x = stheta * srhodrho;
   y = ctheta * srhodrho;
   z = crhodrho;
   glTexCoord2f(s, t - dt);
          s += ds;
          glNormal3f(x, y, z);
          glVertex3f(x * fRadius, y * fRadius, z * fRadius);
          }
       glEnd();

       t -= dt;
       }
    }

// Rotation amounts
static GLfloat xRot = 0.0f;
static GLfloat yRot = 0.0f;

/*********************************************************************************************/
void drawball(void)
 {
```

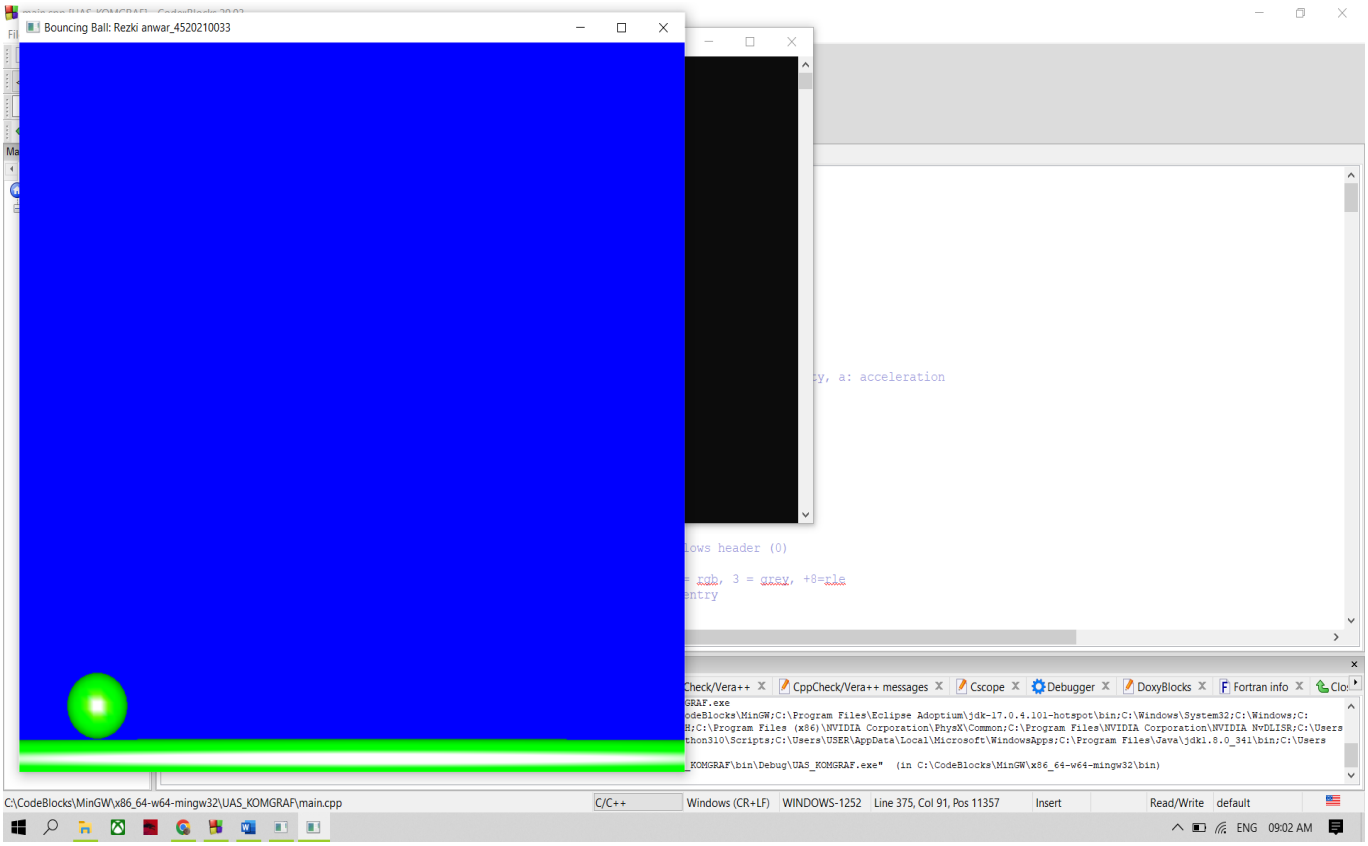
```
 glPushMatrix();
 glBindTexture(GL_TEXTURE_2D, textureObjects[1]);
 glRotatef(-90, 0.0f, 1.0f, 0.0f);
 gltDrawSphere(10, 26, 13);
 glPopMatrix();
 }

/*********************************************************************************************/


// Called to draw scene
void RenderScene(void)
 {

 // Clear the window with current clearing color
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 /*****************************************************************************/
 //the ball's move
 v = v+a; position = position + v; move = move - a/1.9;
 if(position < 0) v = -v;
 if(position < -10) {position = -10; v = 0; a=0;}

 //draw the ball  :)
 glPushMatrix();
 glColor3f(0.0f, 1.0f, 0.0f);//Green
 glTranslatef(move, position, 1.0f);
 glRotatef(v*3, 0.0f, 1.0f, 1.0f); //the ball rotation
 drawball();
 glPopMatrix();


  //floor
 glPushMatrix();
 glBindTexture(GL_TEXTURE_BORDER, textureObjects[0]);
 glTranslatef(0.0f,-26.0f, 0.0f);
 glScalef(20.0f, 0.30f, 1.0f);
 gltDrawSphere(20, 26, 13);
 glPopMatrix();

 // Show the image
 glutSwapBuffers();
 }

 //Trigger button x,y
```

```
     void keyboard (unsigned char key, int x, int y)
{
   switch (key) {
      //up
      case 'x':
         v = (v + 5) ;
          glutPostRedisplay();
          break;
      case 'y':
         a = (a - 2);
          glutPostRedisplay();
          break;
      default:
         break;
   }
}


// This function does any needed initialization on the rendering
// context.
void SetupRC()
 {
 int i;

 GLfloat  lightPos[] = { -50.f, 50.0f, 100.0f, 1.0f };
 //GLfloat  lightPos[] = { -100.f, 100.0f, 50.0f, 1.0f };
 GLfloat  ambientLight[] = { 0.9f, 0.3f, 0.6f, 0.9f };
 GLfloat  diffuseLight[] = { 0.12f, 0.7f, 0.10f, 1.9f };
 GLfloat  specular[] = { 50.0f, 50.0f, 1.0f, 50.0f};
 GLfloat  specref[] =  { 100.0f, 100.0f, 100.0f, 80.0f };

 glEnable(GL_DEPTH_TEST); // Hidden surface removal
 glEnable(GL_LIGHTING);  //tambahan buat lightning,

 glLightfv(GL_LIGHT0, GL_SHININESS, specref);//menambahkan GL_SHININESS
 glLightfv(GL_LIGHT0,GL_AMBIENT,ambientLight);
 glLightfv(GL_LIGHT0,GL_DIFFUSE,diffuseLight);
 glLightfv(GL_LIGHT0,GL_SPECULAR,specular);

 glEnable(GL_LIGHT0);

 glLightfv(GL_LIGHT1,GL_POSITION,lightPos);
 glLightfv(GL_LIGHT1, GL_SHININESS, specref);
 glLightfv(GL_LIGHT1,GL_AMBIENT,ambientLight);
```

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,diffuseLight);

glFrontFace(GL_CCW);  // Counter clock-wise polygons face out
glEnable(GL_COLOR_MATERIAL); // give color to surface

glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);

glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,specref);
glMateriali(GL_FRONT_AND_BACK,GL_SHININESS,128); // give shininess effect

// White background
glClearColor(0.0,0.0,1.0,0.0);//dark blue
  // Set up texture maps
   glEnable(GL_TEXTURE_BORDER);
   glGenTextures(NUM_TEXTURES, textureObjects);
   glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);



   for(i = 0; i < NUM_TEXTURES; i++)
      {
      GLbyte *pBytes;
      GLint iWidth, iHeight, iComponents;
      GLenum eFormat;

      glBindTexture(GL_TEXTURE_BORDER, textureObjects[i]);

      // Load this texture map
      pBytes = gltLoadTGA(szTextureFiles[i], &iWidth, &iHeight, &iComponents,
&eFormat);
      gluBuild2DMipmaps(GL_TEXTURE_BORDER, iComponents, iWidth, iHeight,
eFormat, GL_UNSIGNED_BYTE, pBytes);
      free(pBytes);

      glTexParameteri(GL_TEXTURE_MATRIX, GL_TEXTURE_WRAP_S,
GL_CLAMP_TO_EDGE);  // Konfigurasi parameter tekstur baru
      glTexParameteri(GL_TEXTURE_MATRIX, GL_TEXTURE_WRAP_T,
GL_CLAMP_TO_EDGE);  // Konfigurasi parameter tekstur baru
      glTexParameteri(GL_TEXTURE_MATRIX, GL_TEXTURE_MIN_FILTER,
GL_NEAREST);  // Konfigurasi parameter tekstur baru
      glTexParameteri(GL_TEXTURE_MATRIX, GL_TEXTURE_MAG_FILTER,
GL_NEAREST);  // Konfigurasi parameter tekstur baru
      }
```

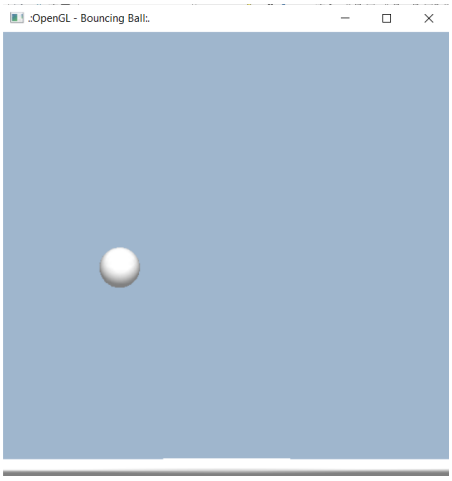
```
   }


void TimerFunc(int value)
   {
   glutPostRedisplay();
   glutTimerFunc(100, TimerFunc, 1);
   }


void ChangeSize(int w, int h)
 {
 GLfloat nRange = 110.0f;

 if(h == 0)
  h = 1;

   glViewport(0, 0, w, h);

 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();

   if (w <= h)
  glOrtho (-nRange, nRange, -nRange*h/w, nRange*h/w, -nRange, nRange);
   else
  glOrtho (-nRange*w/h, nRange*w/h, -nRange, nRange, -nRange, nRange);

 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
 glTranslatef(0.0f, -80.0f, 0.0f);

 }

int main(int argc, char* argv[])
 {
 glutInit(&argc, argv);
 glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
 glutInitWindowSize(750,750);
 glutInitWindowPosition(10,10);
 glutCreateWindow("Bouncing Ball: Rezki anwar_4520210033");
 glutReshapeFunc(ChangeSize);
 glutDisplayFunc(RenderScene);
 glutKeyboardFunc(keyboard);
 glutTimerFunc(500, TimerFunc, 1);
```

```
SetupRC();
glutMainLoop();

return 0;
}
```

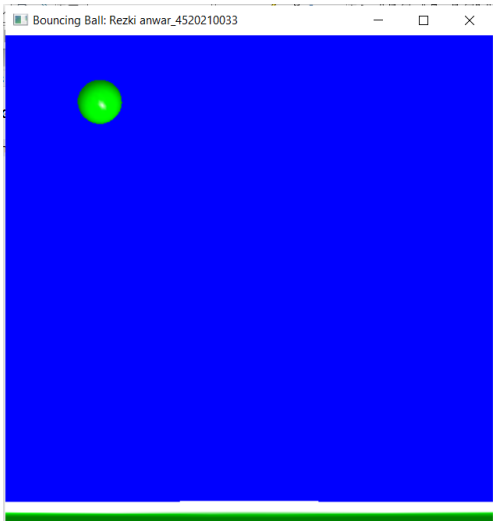


Modifikasi Warna

#warna Bola
glColor4f(1.0f, 1.0f, 1.0f, 0.0f);//white
#warna Background
glClearColor(0.6230f, 0.713f, 0.803f, 1.0f);



glColor3f(0.0f, 1.0f, 0.0f);//Green
glClearColor(0.0,0.0,1.0,0.0);//dark blue



Modifikasi Transformasi

```
//speed
static GLfloat move = -100, height = 200, position=height, v=1, a=-2; //v: velocity, a: acceleration
```

```
//draw the ball
 glPushMatrix();
 glColor3f(0.0f, 1.0f, 0.0f);//Green
 glTranslatef(move, position, 1.0f);
 glRotatef(v*3, 0.0f, 1.0f, 1.0f); //the ball rotation
 drawball();
 glPopMatrix();
```

```
/Trigger button x (up) ,y (down)
      void keyboard (unsigned char key, int x, int y)
{
   switch (key) {
      case 'x':
         v = (v + 5) ;
          glutPostRedisplay();
          break;
      case 'y':
         a = (a - 2);
          glutPostRedisplay();
          break;
      default:
         break;
   }
}
```

```
// Register keyboard callback function
glutKeyboardFunc(keyboard);
```

Jadi, tombol “x” akan membuat bola terangkat ke atas dan jika menekan tombol ”y” akan membuat bola kebawah.

Modifikasi tekstur

```
// Set up texture maps
glEnable(GL_TEXTURE_BORDER);
   glGenTextures(NUM_TEXTURES, textureObjects);
   glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
```

1. **glEnable(GL_TEXTURE_2D)** mengaktifkan rendering tekstur 2D di OpenGL.
2. **glGenTextures(NUM_TEXTURES, textureObjects)** membuat NUM_TEXTURES jumlah obyek tekstur baru yang dapat digunakan oleh aplikasi dan menyimpan ID obyek tekstur dalam array textureObjects.
3. **glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL)** mengatur mode tekstur yang akan digunakan saat rendering obyek tekstur. Mode GL_DECAL akan menggunakan tekstur sebagai lapisan teratas pada obyek yang sedang di-render dan mengabaikan warna obyek yang sedang di-render.

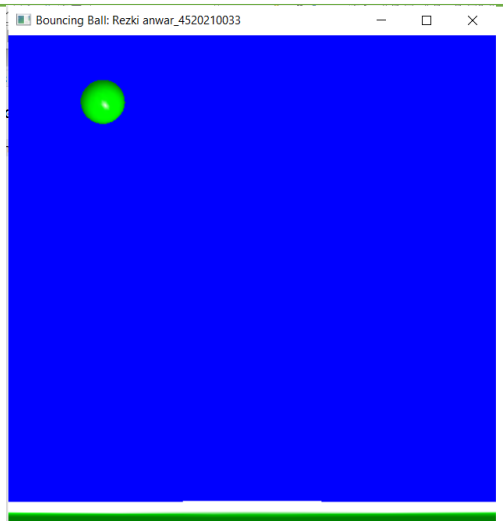
```
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
```

Diubah menjadi:

```
  glTexParameteri(GL_TEXTURE_BORDER, GL_TEXTURE_WRAP_S,
GL_CLAMP_TO_EDGE);  // Konfigurasi parameter tekstur baru
      glTexParameteri(GL_TEXTURE_BORDER, GL_TEXTURE_WRAP_T,
GL_CLAMP_TO_EDGE);  // Konfigurasi parameter tekstur baru
      glTexParameteri(GL_TEXTURE_BORDER, GL_TEXTURE_MIN_FILTER,
GL_NEAREST);  // Konfigurasi parameter tekstur baru
      glTexParameteri(GL_TEXTURE_BORDER_COLOR, GL_TEXTURE_MAG_FILTER,
GL_NEAREST);  // Konfigurasi parameter tekstur baru
```

Modifikasi Pencahayaan

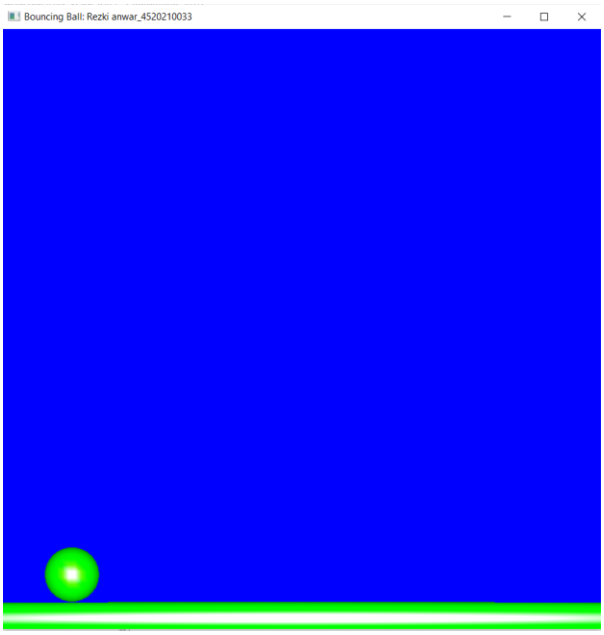
```
GLfloat  lightPos[] = { -50.f, 50.0f, 100.0f, 1.0f };
//GLfloat  lightPos[] = { -100.f, 100.0f, 50.0f, 1.0f };
GLfloat  ambientLight[] = { 0.3f, 0.3f, 0.3f, 0.0f };
GLfloat  diffuseLight[] = { 0.7f, 0.7f, 0.7f, 1.0f };
GLfloat  specular[] = { 1.0f, 1.0f, 1.0f, 1.0f};
GLfloat  specref[] =  { 1.0f, 1.0f, 1.0f, 1.0f };
```



Diubah menjadi lebih cerah dengan memberika angka yang lebih tinggi:

```
GLfloat  lightPos[] = { -50.f, 50.0f, 100.0f, 1.0f };
 //GLfloat  lightPos[] = { -100.f, 100.0f, 50.0f, 1.0f };
 GLfloat  ambientLight[] = { 0.9f, 0.3f, 0.6f, 0.9f };
 GLfloat  diffuseLight[] = { 0.12f, 0.7f, 0.10f, 1.9f };
 GLfloat  specular[] = { 50.0f, 50.0f, 1.0f, 50.0f};
 GLfloat  specref[] =  { 100.0f, 100.0f, 100.0f, 80.0f };
```

Ditambahkan juga Varibel GL_SHENINESS


Bouncing Ball: Rezki anwar_4520210033