LAPORAN DOKUMENTASI FINAL PROJECT GRAFIKA KOMPUTER C

oleh

Putu Adhi Purwanto	5112100049
Fadrian Merdianto	5112100059
Eric Ivander Jeadi	5112100115
Ratih Ayu Indraswari	5112100122

ITS
Institut
Teknologi
Sepuluh Nopember

Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya
2014/2015

DAFTAR ISI

0. Daftar Isi 2
1. Objek 3
1.1 Membuat Objek 3
1.2 Daftar Objek yang Dibuat 10
2. OpenGL 11
2.1 Library OpenGL 11
2.2 Menampilkan Objek 11
2.3 Translasi dan Rotasi 11
2.4 Tekstur 11
2.5 Potongan Kode 11
3. Aplikasi 25
3.1 Tampilan Aplikasi 25
3.2 Penggunaan Aplikasi 29
4. Daftar Pustaka 30

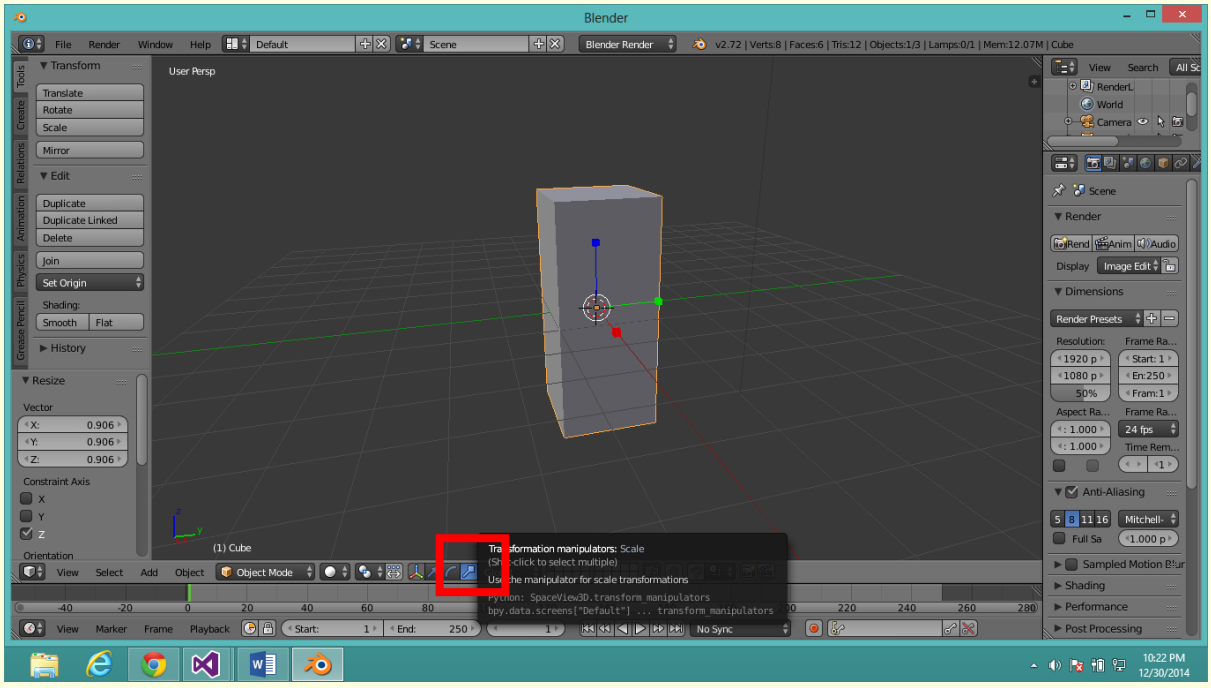
OBJEK

1.1 Membuat Objek

Objek dibuat menggunakan aplikasi Blender dan SketchUp, Blender merupakan aplikasi *freeware* dan dapat didownload di www.blender.org .

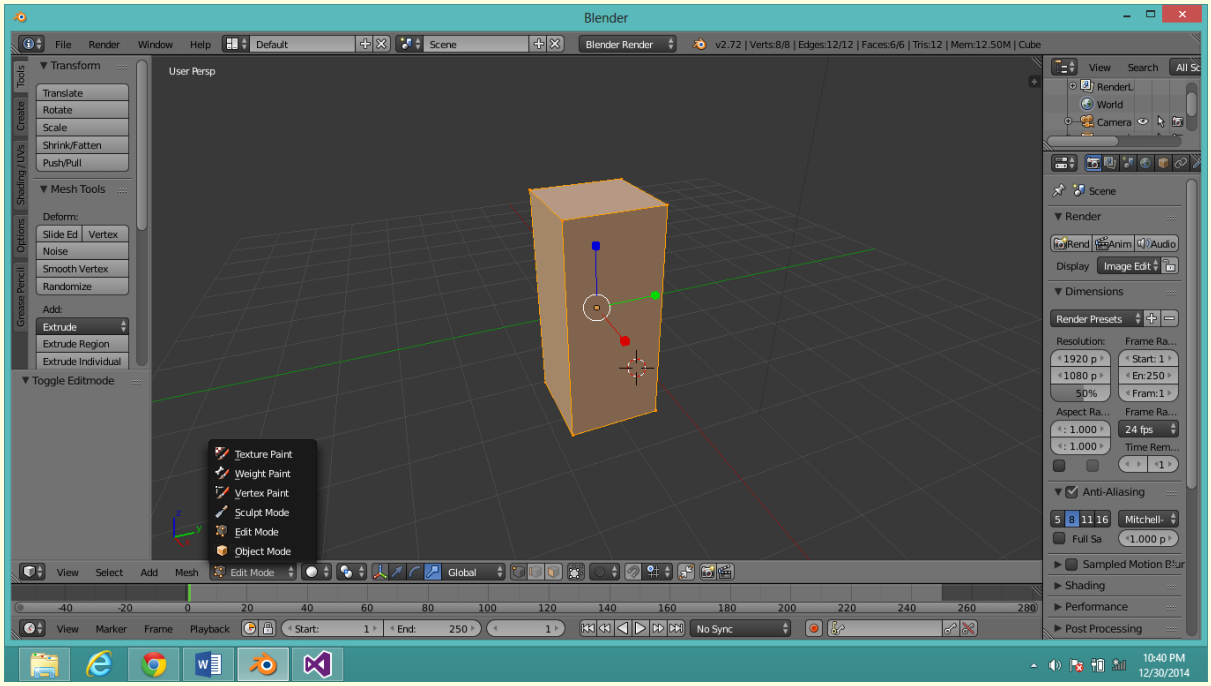
Berikut langkah-langkah membuat objek sederhana dari aplikasi Blender. Contohnya membuat dispenser.

1. Objek awal (kubus) ditarik dengan "Transformation Manipulators: Scale", atau dengan menekan tombol "S" pada keyboard.



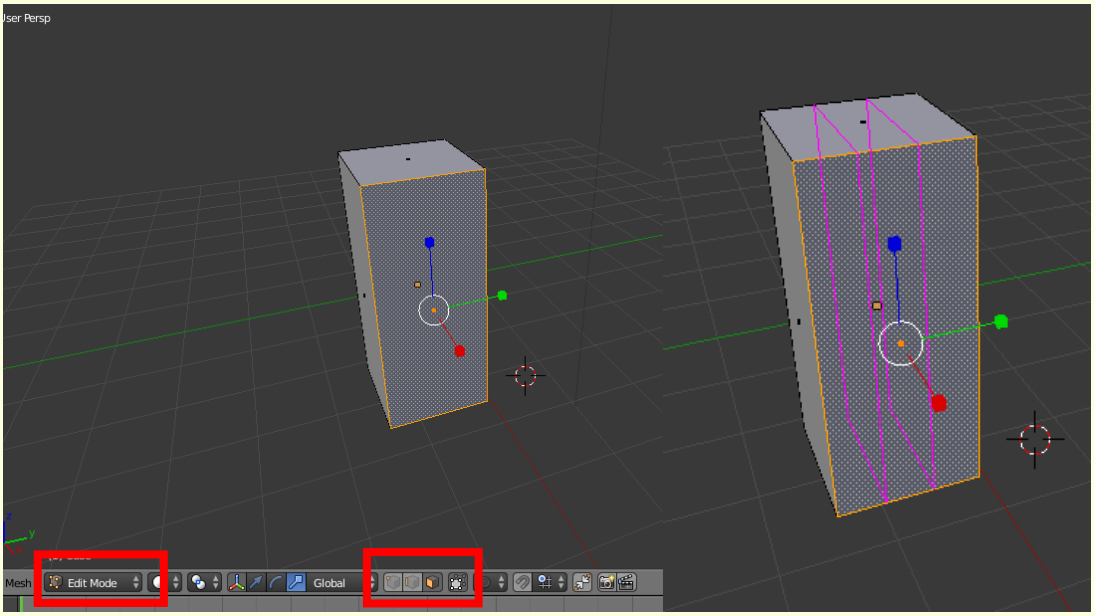
Gambar 1.1 *Scaling* objek dengan Transformation Manipulators: Scale

2. Editing objek kubus untuk merubah ke bentuk lainnya dengan memilih "edit mode".



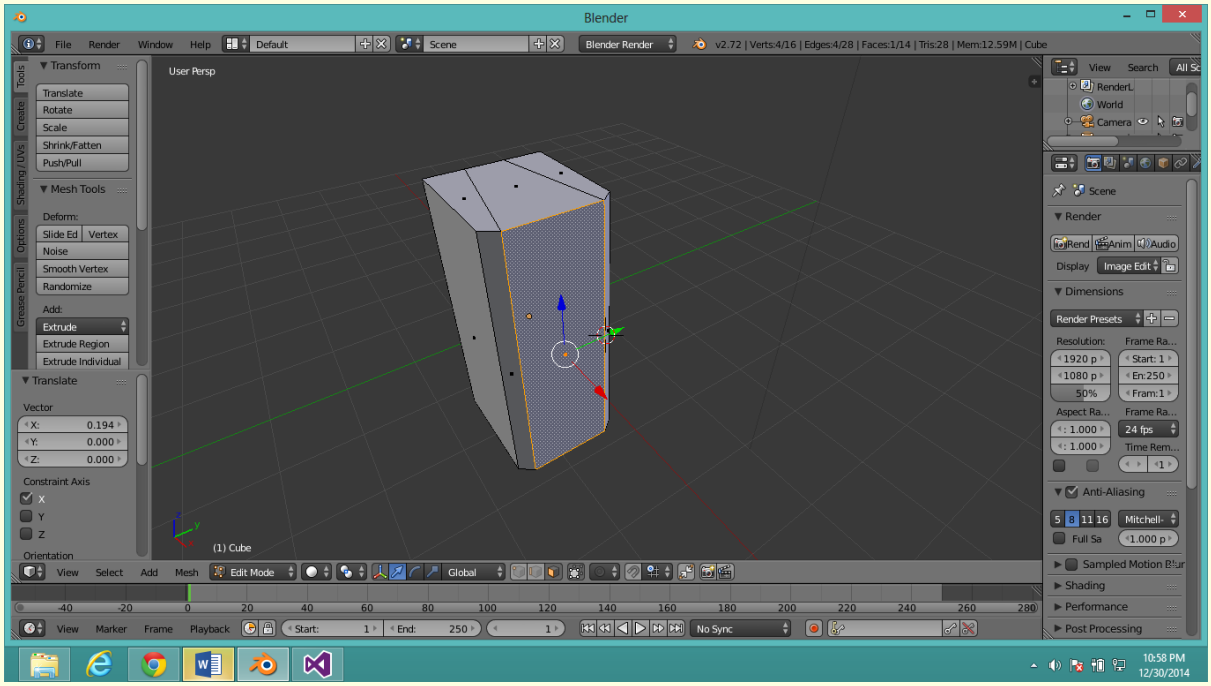
Gambar 1.2 Mulai mengedit gambar dengan edit mode

3. Menambah edge pada face objek dengan memilih surface yang ingin dirubah. Pertama-tama klik kanan di surface yang ingin dirubah dengan , kemudian tekan Ctrl+R lalu scroll atas untuk menambah jumlah edge yang ingin dtimabahkan pada face objek.



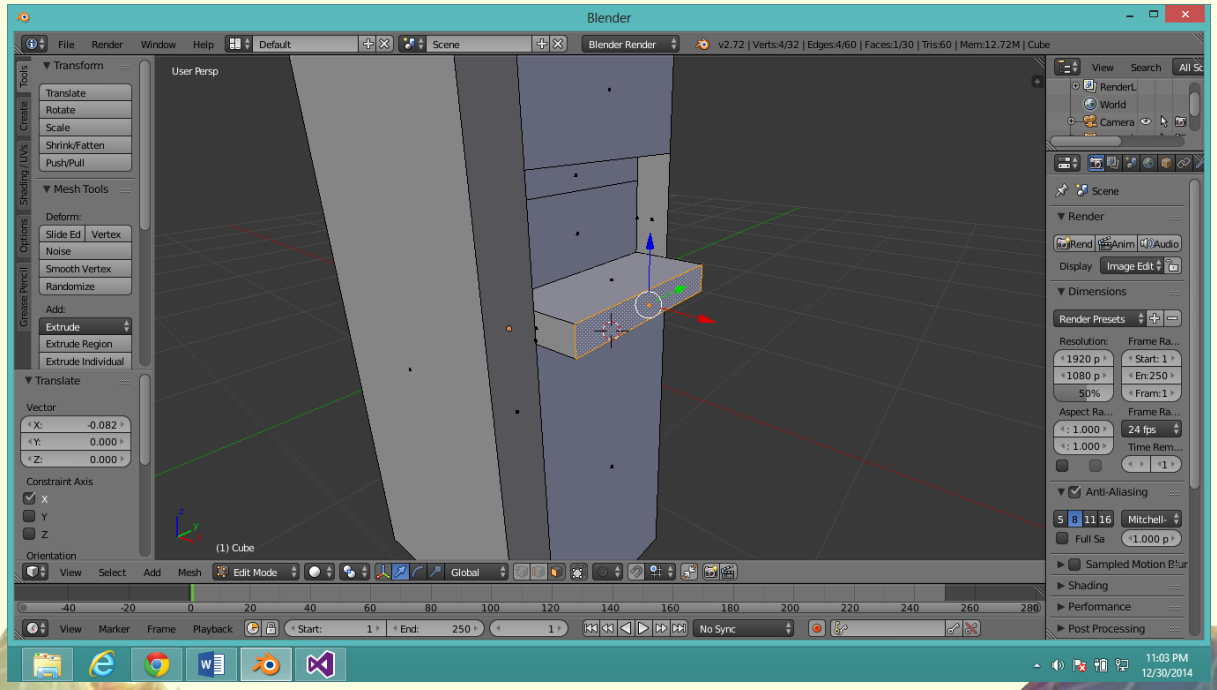
Gambar 1.3 Menambahkan edge pada face objek

4. Merubah bentuk dengan memilih “edge” (klik kanan) kemudian digeser sesuai keinginan, kemudian face yang ingin di rubah dengan klik kanan “face” lalu menariknya dengan "Transformation Manipulators: Translate”. (*dapat memilih face, edge, atau vertex dengan menekan “Ctrl+Klik Kanan” atau “Shift+Klik Kanan”).



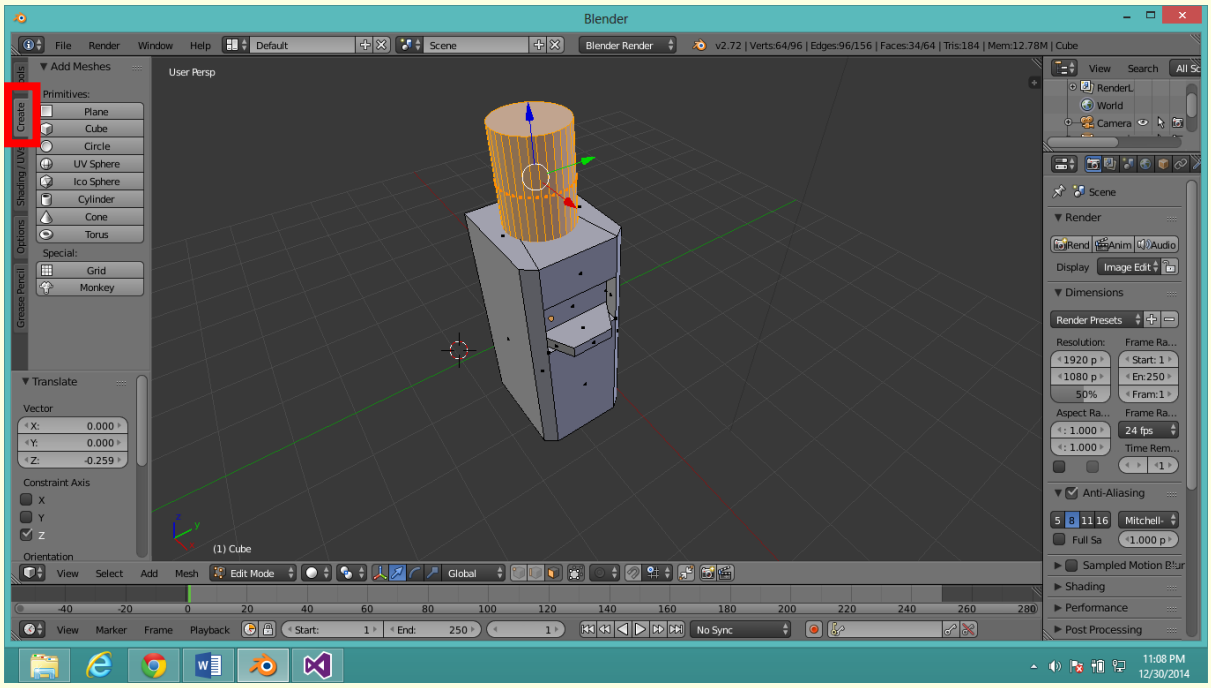
Gambar 1.4 Merubah bentuk objek

5. Menambahkan edge baru dengan menekan “E” pada keyboard dan scaling face dengan menekan “S” pada keyboard. (*scaling secara vertical ataupun horizontal dengan menekan tombol “S” kemudian “Menekan Scroll” pada mouse sambil menggesernya).



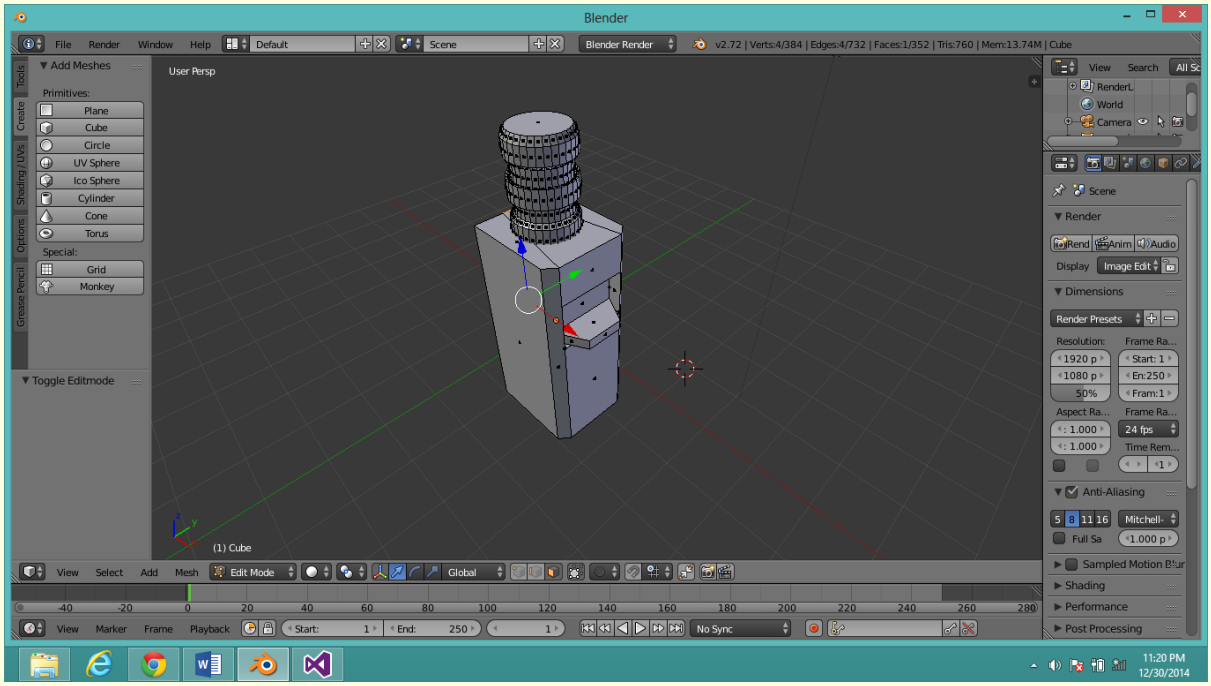
Gambar 1.5 Menambah edge baru

6. Menambahkan objek lain ke objek sebelumnya (dalam hal ini pembuatan gallon air untuk dispenser). Pertama-tama di "edit mode" klik "create" lalu pilih objek baru yang ingin ditambahkan.



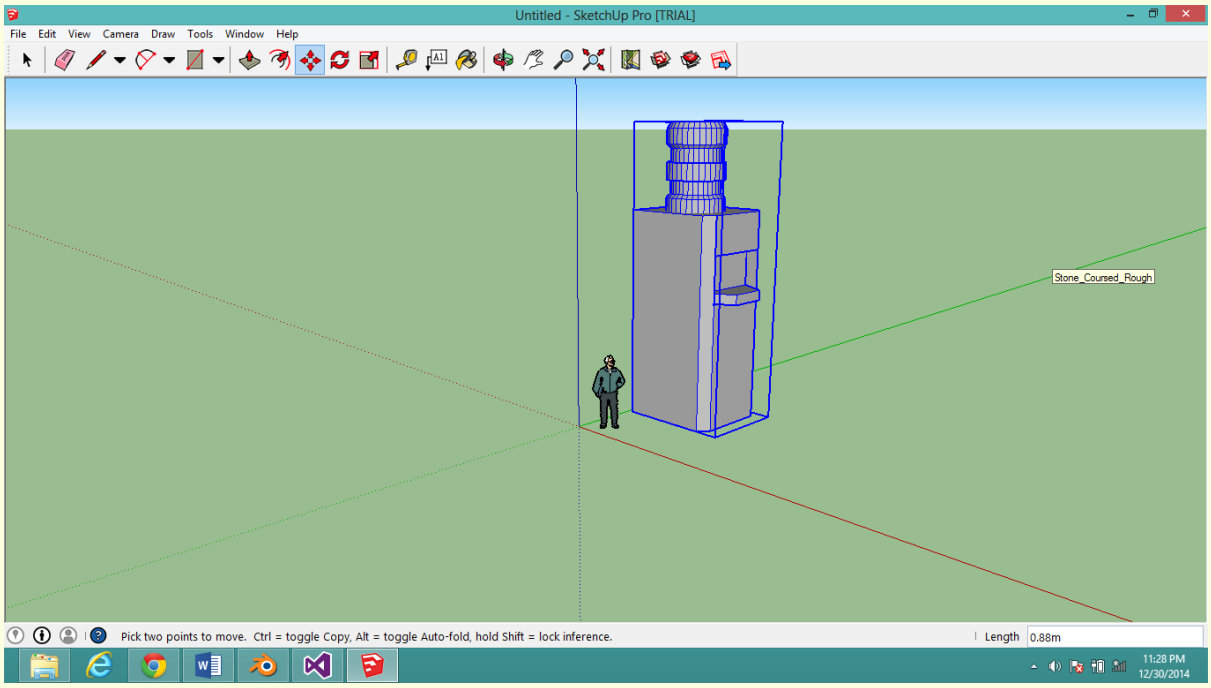
Gambar 1.6 Menambah objek baru

7. Hasil akhir pembuatan objek (dispenser) pada aplikasi Blender.



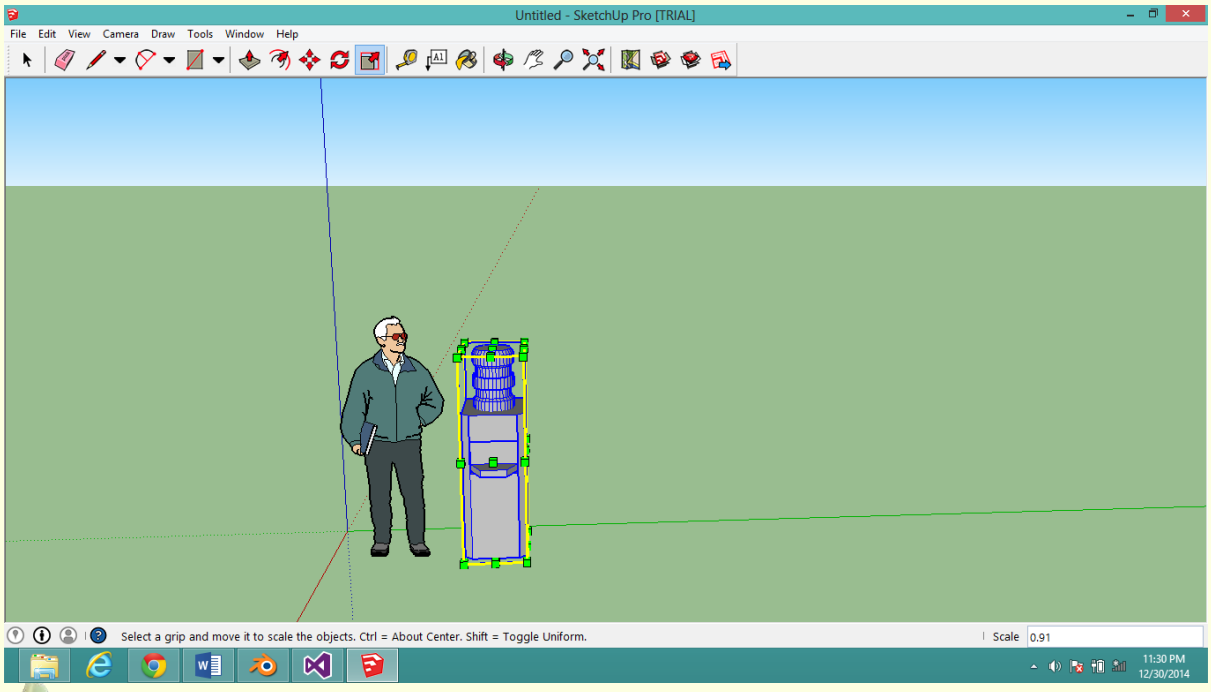
Gambar 1.7 Hasil akhir pembuatan objek

8. Untuk memberi tekstur kami menggunakan aplikasi SketchUp. Pertama-tama objek dalam aplikasi Blender "di-export" (File -> Export -> ke dalam bentuk Collada (*.dae). File collada tersebut kemudian "di-import" ke aplikasi SketchUp.



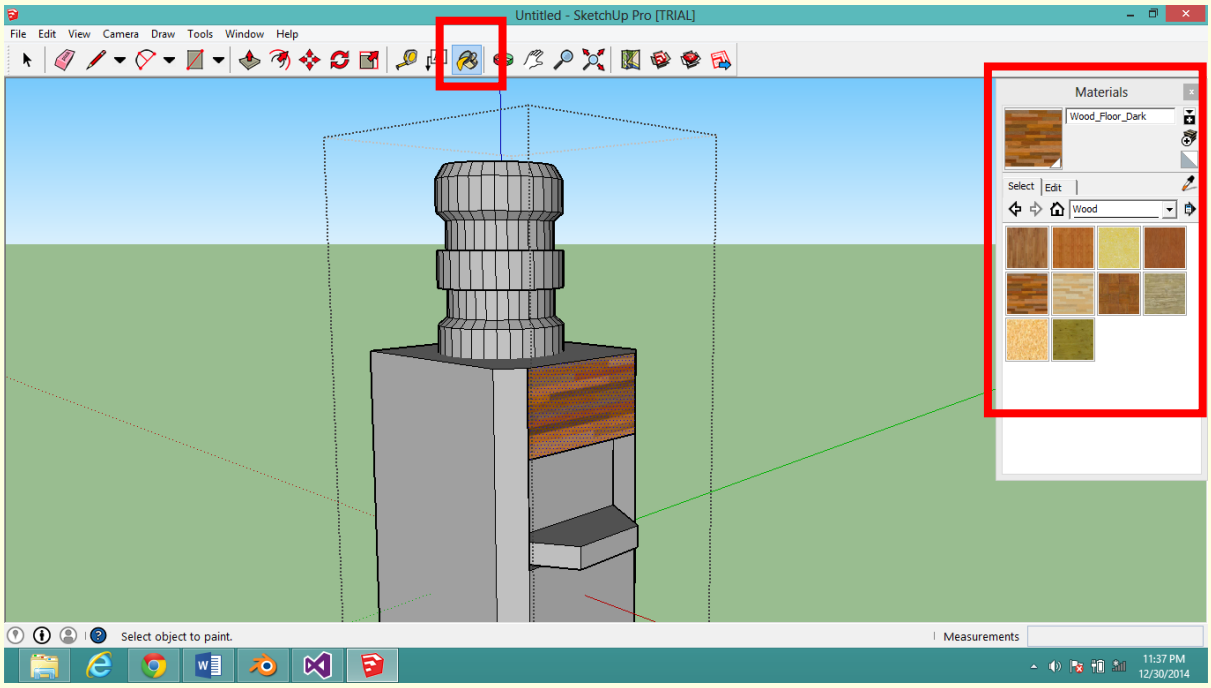
Gambar 1.8 Import file ke SketchUp

9. Rubah skala objek dengan menekan tombol "S" kemudian disesuaikan dengan ukuran sebenarnya (gambar bapak tua sebagai acuan skala).



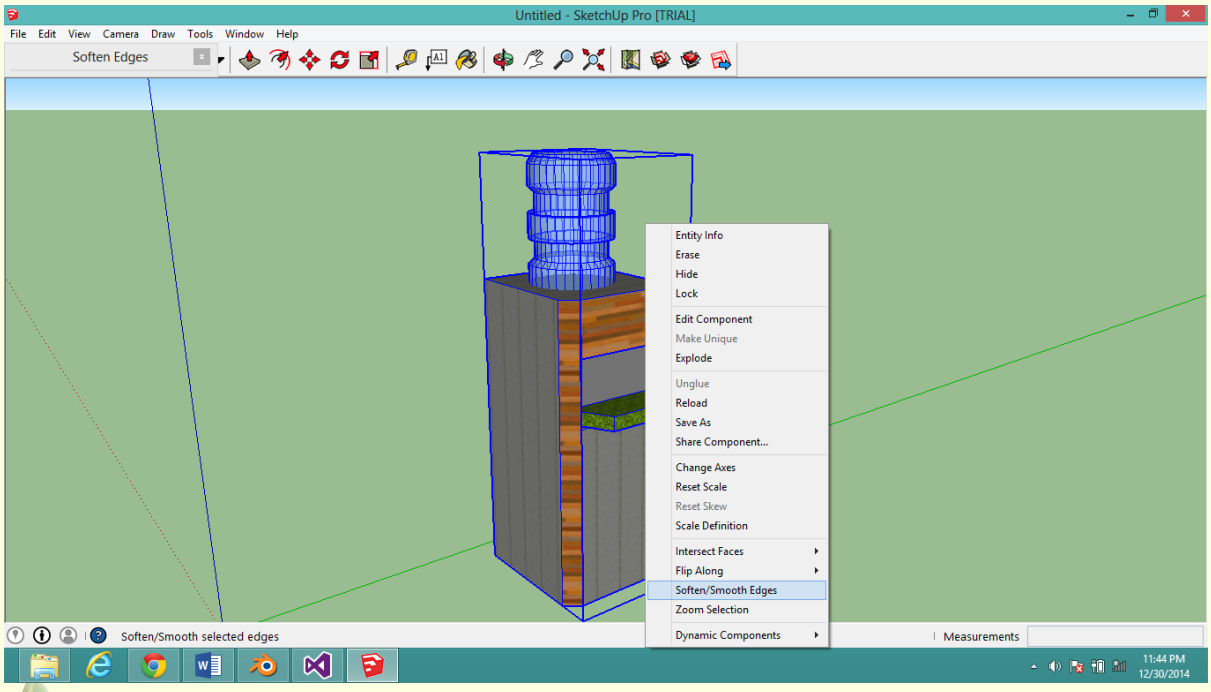
Gambar 1.9 Menyesuaikan skala objek

10. Memberi tekstur pada face tertentu dengan melakukan dua kali *double-click* pada face atau *triple-click* untuk memilih seluruh face pada objek, kemudia pilih "paint bucket" untuk memberi tekstur pada face.



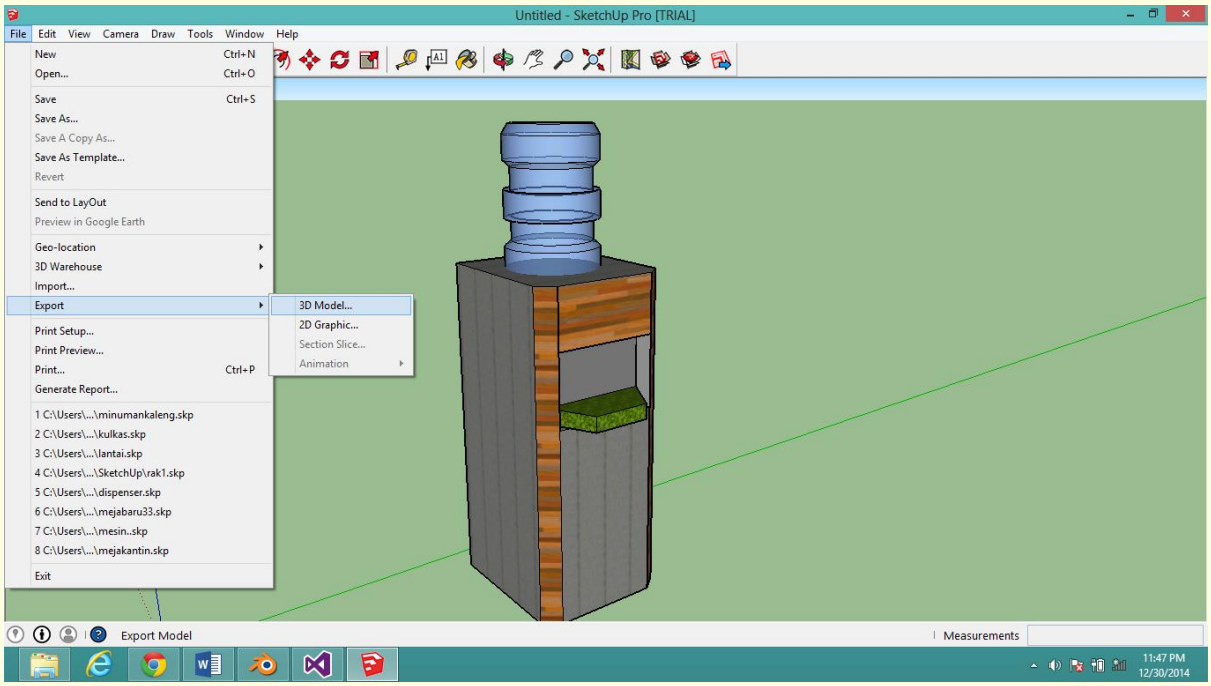
Gambar 1.10 Memberi tekstur pada face/objek

11. Memperhalus edge objek dengan klik kanan pada objek kemudian pilih "Soften/Smooth Edges".



Gambar 1.11 Memperhalus objek

12. Setelah objek selesai diberi tekstur, export objek menjadi file *.obj dengan memilih File -> Export -> 3D Model.



Gambar 1.12 Export file dari SketchUp

13. File akan tersimpan dalam bentuk file *.obj, *.mtl, dan file tekstur.

1.2 Daftar Objek yang Dibuat

1. Dispenser.
2. Kulkas.
3. Meja Bundar.
4. Meja Kantin.
5. Lampu.
6. Rak.
7. Meja Lesehan
8. Mesin Pembayaran
9. Balon.
10. Air Mineral.
11. Minuman Kaleng.
12. Popmi.
13. Roti.
14. Es Krim *Cone*.

Gambar 1.13 Kumpulan Objek yang sudah dibuat

OPENGL

2.1 Library OpenGL

Library yang digunakan dalam pembuatan *final project* ini adalah :

1. OpenGL.
2. FreeGLUT (Free OpenGL Utility Toolkit).
3. GLM (OpenGL Mathematics).
4. DevIL (Developer's Image Library).

2.2 Menampilkan Objek

Menampilkan objek menggunakan library GLM dengan memanggil fungsi "glmReadOBJ". Parameter yang diberikan adalah path menuju file obj yang telah dibuat sebelumnya.

2.3 Translasi dan Rotasi

Mentranslasi objek dengan fungsi OpenGL "glTranslatef" dan rotasi dengan "glRotatef".

2.4 Tekstur

Perubahan tekstur dilakukan dengan cara menyembunyikan objek yang lama dan menampilkan objek yang sama dengan tekstur yang sudah dipilih. Jumlah tekstur bervariasi tergantung objeknya.

2.5 Potongan Kode

1. camera.h

```
#include <gl\glut.h>
#define PI 3.1415265359
#define PIdiv180 3.1415265359/180.0

//Note: All angles in degrees

//Float 3d-vect, normally used
struct SF3dVector
{
   GLfloat x,y,z;
};
struct SF2dVector
{
   GLfloat x,y;
};

class CCamera
{
```

```
private:
   SF3dVector Position;
   /*Not used for rendering the camera, but for "moveforwards"
   So it is not necessary to "actualize" it always. It is only
   actualized when ViewDirChanged is true and moveforwards is
called*/
   SF3dVector ViewDir;
   bool ViewDirChanged;
   GLfloat RotatedX, RotatedY, RotatedZ;
   void GetViewDir ( void );
public:
   //inits the values (Position: (0|0|0) Target: (0|0|-1) )
   CCamera();
   //executes some glRotates and a glTranslate command
   //Note: You should call glLoadIdentity before using Render
   void Render ( void );
   void Move ( SF3dVector Direction );
   void RotateX ( GLfloat Angle );
   void RotateY ( GLfloat Angle );
   void RotateZ ( GLfloat Angle );
   void RotateXYZ ( SF3dVector Angles );
   void MoveForwards ( GLfloat Distance );
   void StrafeRight ( GLfloat Distance );
   void TurnRight ( GLfloat Angle );
   GLfloat getPositionX();
   GLfloat getPositionY();
   GLfloat getPositionZ();
   GLfloat getRotatedX();
   GLfloat getRotatedY();
   GLfloat getRotatedZ();
};


SF3dVector F3dVector ( GLfloat x, GLfloat y, GLfloat z );
SF3dVector AddF3dVectors ( SF3dVector * u, SF3dVector * v);
void AddF3dVectorToVector ( SF3dVector * Dst, SF3dVector * V2);
```

2. camera.cpp

```
#include "camera.h"
#include "math.h"
#include <iostream>
#include <stdio.h>
#include "windows.h"

SF3dVector F3dVector( GLfloat x, GLfloat y, GLfloat z )
{
   SF3dVector tmp;
   tmp.x = x;
   tmp.y = y;
   tmp.z = z;
   return tmp;
}
SF3dVector AddF3dVectors (SF3dVector* u, SF3dVector* v)
{
   SF3dVector result;
   result.x = u->x + v->x;
   result.y = u->y + v->y;
   result.z = u->z + v->z;
```

```
   return result;
}
void AddF3dVectorToVector ( SF3dVector * Dst, SF3dVector * V2)
{
   Dst->x += V2->x;
   Dst->y += V2->y;
   Dst->z += V2->z;
}

CCamera::CCamera()
{
   //Init with standard OGL values:
   Position = F3dVector (  0.0,
                  0.0,
                  0.0);
   ViewDir = F3dVector(  0.0,
                  0.0,
                  -1.0);
   ViewDirChanged = false;
   //Only to be sure:
   RotatedX = RotatedY = RotatedZ = 0.0;
}

void CCamera::GetViewDir( void )
{
   SF3dVector Step1, Step2;
   //Rotate around Y-axis:
   Step1.x = cos( (RotatedY + 90.0) * PIdiv180);
   Step1.z = -sin( (RotatedY + 90.0) * PIdiv180);
   //Rotate around X-axis:
   double cosX = cos (RotatedX * PIdiv180);
   Step2.x = Step1.x * cosX;
   Step2.z = Step1.z * cosX;
   Step2.y = sin(RotatedX * PIdiv180);
   //Rotation around Z-axis not yet implemented, so:
   ViewDir = Step2;
}
void CCamera::Move (SF3dVector Direction)
{
   AddF3dVectorToVector(&Position, &Direction);
}

void CCamera::RotateY (GLfloat Angle)
{
   RotatedY += Angle;
   ViewDirChanged = true;
}

void CCamera::RotateX (GLfloat Angle)
{
   RotatedX += Angle;
   ViewDirChanged = true;
}

void CCamera::Render( void )
{
   glRotatef(-RotatedX , 1.0, 0.0, 0.0);
   glRotatef(-RotatedY , 0.0, 1.0, 0.0);
   glRotatef(-RotatedZ , 0.0, 0.0, 1.0);
   glTranslatef( -Position.x, -Position.y, -Position.z );
}
```

```
void CCamera::MoveForwards( GLfloat Distance )
{
   if (ViewDirChanged) GetViewDir();
   SF3dVector MoveVector;
   MoveVector.x = ViewDir.x * -Distance;
   MoveVector.y = ViewDir.y * -Distance;
   MoveVector.z = ViewDir.z * -Distance;
   AddF3dVectorToVector(&Position, &MoveVector );
}

void CCamera::StrafeRight ( GLfloat Distance )
{
   if (ViewDirChanged) GetViewDir();
   SF3dVector MoveVector;
   MoveVector.z = -ViewDir.x * -Distance;
   MoveVector.y = 0.0;
   MoveVector.x = ViewDir.z * -Distance;
   AddF3dVectorToVector(&Position, &MoveVector );
}

void CCamera::TurnRight(GLfloat Angle)
{
   if (ViewDirChanged) GetViewDir();
}

GLfloat CCamera::getPositionX()
{
   return Position.x;
}

GLfloat CCamera::getPositionY()
{
   return Position.y;
}

GLfloat CCamera::getPositionZ()
{
   return Position.z;
}

GLfloat CCamera::getRotatedX()
{
   return RotatedX;
}

GLfloat CCamera::getRotatedY()
{
   return RotatedY;
}

GLfloat CCamera::getRotatedZ()
{
   return RotatedZ;
}
```

3. main.cpp

```
#include <Windows.h>
#include <gl\GL.h>
#include <gl\GLU.h>
#include <gl\glut.h>
#include "glm\glm.h"
#include <stdio.h>
#include <math.h>
#include <vector>
#include <utility>
#include <string>
#include <string.h>
#include <map>
#include <stdlib.h>
#include <algorithm>
#include "camera.h"

using namespace std;

#define MAX_Balon 2
#define MAX_Dispenser 2
#define MAX_Kulkas 2
#define MAX_Lampu 1
#define MAX_MejaBundar 3
#define MAX_MejaKantin 3
#define MAX_MejaMakan 2
#define MAX_MinumanKaleng 3
#define MAX_Mug 2
#define MAX_Rak 3

#define mp make_pair

GLMmodel *sample;
float ratio;
int choose = 1;

struct attrib
{
   CCamera camObj;
   GLfloat start_x = 0.0;
   GLfloat start_y = 0.0;
   GLfloat start_z = 0.0;
};

vector<pair<string, attrib> > myObject;
map<string, int> myMap;
string selected = "";

CCamera Camera;
int win_h = 1000;
int win_w = 1000;

void Init();
void InitObj();
void mymenu();
void menu();
void display();
void Load(char *);
void keyDown(unsigned char, int, int);
void specialKey(int, int, int);
```

```
void mouseDown(int, int, int, int);
void reshape(int, int);
void displayObject(string, attrib);
void displayLantai();
void displayBalon();
void displayDispenser();
void displayKulkas();
void displayLampu();
void displayMejaBundar();
void displayMejaKantin();
void displayMejaMakan();
void displayMinumanKaleng();
void displayMug();
void displayRak();
int maxTexture(string);

int main(int argc, char** argv)
{
   InitObj();

   glutInit(&argc, argv);
   //inisialisasi
   glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
   //mode detail, 3 dimensi dan RGB
   glutInitWindowPosition(300, 1);
   glutInitWindowSize(1000, 1000);
   glutCreateWindow("Kantin");

   Camera.Move(F3dVector(0.0, 0.0, 0.0));
   Camera.MoveForwards(0.0);

   menu();

   glutDisplayFunc(display);
   glutSpecialFunc(specialKey);
   glutKeyboardFunc(keyDown);
   glutMouseFunc(mouseDown);
   glutIdleFunc(display);
   glutReshapeFunc(reshape);
   Init(); //init
   glutMainLoop();
}

void Init()
{
   //inisialisasi mode smoot dan texture dari gambar
   glEnable(GL_TEXTURE_2D);
   glEnable(GL_POINT_SMOOTH);
   glHint(GL_POINT_SMOOTH_HINT, GL_DONT_CARE);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_LIGHT1);
   //Perspektif View
   glEnable(GL_DEPTH_TEST);
   glDepthFunc(GL_LESS);
   //blend warna untuk texture dan warna
   glEnable(GL_COLOR_MATERIAL);
   glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
}

void InitObj()
```

```
{
   myObject.clear();
   struct attrib objLantai;
   objLantai.camObj = Camera;
   myObject.push_back(mp("lantai", objLantai));
   struct attrib objKulkas;
   objKulkas.camObj = Camera;
   objKulkas.start_x = -2;
   objKulkas.start_y = 0.25;
   myObject.push_back(mp("kulkas", objKulkas));
   struct attrib objDispenser;
   objDispenser.camObj = Camera;
   objDispenser.start_x = -2;
   objDispenser.start_y = 0.2;
   objDispenser.start_z = 0.9;
   myObject.push_back(mp("dispenser", objDispenser));
   struct attrib objLampu;
   objLampu.camObj = Camera;
   objLampu.start_x = -2.6;
   objLampu.start_y = 0.2;
   objLampu.start_z = -2.6;
   myObject.push_back(mp("lampu", objLampu));
   struct attrib objMejaBundar;
   objMejaBundar.camObj = Camera;
   objMejaBundar.start_y = -0.35;
   myObject.push_back(mp("mejabundar", objMejaBundar));
   struct attrib objMejaKantin;
   objMejaKantin.camObj = Camera;
   objMejaKantin.start_x = -1.5;
   objMejaKantin.start_y = -0.4;
   objMejaKantin.start_z = 2.1;
   myObject.push_back(mp("mejakantin", objMejaKantin));
   struct attrib objRak;
   objRak.camObj = Camera;
   objRak.camObj.RotateY(-90);
   objRak.start_y = -0.15;
   objRak.start_z = -2.2;
   myObject.push_back(mp("rak", objRak));
}

void mymenu(int id)
{
   if (id == 0) selected = "";
   if (id == 1) selected = "dispenser";
   if (id == 2) selected = "kulkas";
   if (id == 3) selected = "lampu";
   if (id == 4) selected = "mejabundar";
   if (id == 5) selected = "mejakantin";
   if (id == 6) selected = "rak";
   if (id == 99) exit(0);
}

void menu()
{
   int menu_id = glutCreateMenu(mymenu);
   glutAddMenuEntry("World", 0);
   glutAddMenuEntry("Dispenser", 1);
   glutAddMenuEntry("Kulkas", 2);
   glutAddMenuEntry("Lampu", 3);
   glutAddMenuEntry("Meja Bundar", 4);
   glutAddMenuEntry("Meja Bentuk L", 5);
```

```
   glutAddMenuEntry("Rak", 6);
   glutAddMenuEntry("Exit", 99);
   glutAttachMenu(GLUT_RIGHT_BUTTON);
}

void load(char *s)
{
   //load object jika belum di load
   sample = glmReadOBJ(s);
   glmUnitize(sample);
   //draw object
   glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
   glColor3ub(255, 255, 255);
   glmDraw(sample, GLM_SMOOTH | GLM_TEXTURE);
}

//keyboard
void keyDown(unsigned char key, int x, int y)
{
   //key for camera
   switch (key)
   {
   //ESC
   case 27:
      PostQuitMessage(0);
      break;
   //strafe left
   case 'a':
      if (selected.compare("") == 0) Camera.StrafeRight(-0.1);
      else
      {
         for (int i = 0; i < myObject.size(); i++)
         {
            if (myObject[i].first.compare(selected) == 0)
            {
               myObject[i].second.camObj.StrafeRight(0.1);
               i = myObject.size();
            }
         }
      }
      display();
      break;
   //strafe right
   case 'd':
      if (selected.compare("") == 0) Camera.StrafeRight(0.1);
      else
      {
         for (int i = 0; i < myObject.size(); i++)
         {
            if (myObject[i].first.compare(selected) == 0)
            {
               myObject[i].second.camObj.StrafeRight(-0.1);
               i = myObject.size();
            }
         }
      }
      display();
      break;
   //forward
   case 'w':
      if (selected.compare("") == 0) Camera.MoveForwards(-0.1);
```

```
      else
      {
         for (int i = 0; i < myObject.size(); i++)
         {
            if (myObject[i].first.compare(selected) == 0)
            {
               myObject[i].second.camObj.MoveForwards(0.1);
               i = myObject.size();
            }
         }
      }
      display();
      break;
   //backward
   case 's':
      if (selected.compare("") == 0) Camera.MoveForwards(0.1);
      else
      {
         for (int i = 0; i < myObject.size(); i++)
         {
            if (myObject[i].first.compare(selected) == 0)
            {
               myObject[i].second.camObj.MoveForwards(-0.1);
               i = myObject.size();
            }
         }
      }
      display();
      break;
   }

   //key for choose object
   switch (key)
   {
   case '1':
      if (selected.compare("") == 0 || maxTexture(selected) < 1)
break;
      else
      {
         myMap[selected] = 1;
         display();
         break;
      }
   case '2':
      if (selected.compare("") == 0 || maxTexture(selected) < 2)
break;
      else
      {
         myMap[selected] = 2;
         display();
         break;
      }
   case '3':
      if (selected.compare("") == 0 || maxTexture(selected) < 3)
break;
      else
      {
         myMap[selected] = 3;
         display();
         break;
      }
```

```
   }
}

void specialKey(int key, int x, int y)
{
   if (key == GLUT_KEY_UP)
   {
      if (selected.compare("") == 0) Camera.Move(F3dVector(0.0, 0.3,
0.0));
      else
      {
         for (int i = 0; i < myObject.size(); i++)
         {
            if (myObject[i].first.compare(selected) == 0)
            {
               myObject[i].second.camObj.Move(F3dVector(0.0, -0.3,
0.0));
               i = myObject.size();
            }
         }
      }
      display();
   }
   else if (key == GLUT_KEY_DOWN)
   {
      if (selected.compare("") == 0) Camera.Move(F3dVector(0.0, -
0.3, 0.0));
      else
      {
         for (int i = 0; i < myObject.size(); i++)
         {
            if (myObject[i].first.compare(selected) == 0)
            {
               myObject[i].second.camObj.Move(F3dVector(0.0, 0.3,
0.0));
               i = myObject.size();
            }
         }
      }
      display();
   }
   else if (key == GLUT_KEY_LEFT)
   {
      if (selected.compare("") == 0) Camera.RotateY(5.0);
      else
      {
         for (int i = 0; i < myObject.size(); i++)
         {
            if (myObject[i].first.compare(selected) == 0)
            {
               myObject[i].second.camObj.RotateY(-5.0);
               i = myObject.size();
            }
         }
      }
      display();
   }
   else if (key == GLUT_KEY_RIGHT)
   {
      if (selected.compare("") == 0) Camera.RotateY(-5.0);
      else
```

```
      {
        for (int i = 0; i < myObject.size(); i++)
        {
          if (myObject[i].first.compare(selected) == 0)
          {
            myObject[i].second.camObj.RotateY(5.0);
            i = myObject.size();
          }
        }
      }
      display();
   }
}

void mouseDown(int button, int state, int x, int y)
{
   int px = x;
   int py = win_h - y;
}

//object window tetap proposional
void reshape(int w, int h)
{
   win_w = w;
   win_h = h;
   if (h == 0) h = 1;
   ratio = 1.0 * w / h;
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glViewport(0, 0, w, h);
   gluPerspective(80, ratio, 1, 300);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void display()
{
   // set the light source properties
   GLfloat lightIntensity[] = { 1.0f, 1.0f, 1.0f, 1.0f };
   GLfloat light_position[] = { 2.0f, 10.0f, 2.0f, 0.0f };
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);
   glLightfv(GL_LIGHT0, GL_DIFFUSE, lightIntensity);

   // set the light source properties
   GLfloat lightIntensity2[] = { 1.0f, 1.0f, 1.0f, 1.0f };
   GLfloat light_position2[] = { -2.6f, 1.2f, -2.6f, 0.0f };
   glLightfv(GL_LIGHT1, GL_POSITION, light_position2);
   glLightfv(GL_LIGHT1, GL_DIFFUSE, lightIntensity2);

   //mode buffer warna dan 3 dimensi
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   //warna latar
   glClearColor(0, 0, 0, 0);

   //inisialisasi identity
   glLoadIdentity();

   gluLookAt(0.0, 1.0, 3.0, 0, 0, 0, 0, 1, 0);
   Camera.Render();

   glPushMatrix();
```

```
   glColor3f(1.0, 0.0, 0.0);
   glutSolidSphere(0.1, 20, 20);
   glPopMatrix();

   for (int i = 0; i < myObject.size(); i++)
   {
      glPushMatrix();
      displayObject(myObject[i].first, myObject[i].second);
      glPopMatrix();
   }

   //kirim ke display dan swap buffer
   glFlush();
   glutSwapBuffers();
}

void displayObject(string s, attrib prop)
{
   glTranslatef(prop.start_x, prop.start_y, prop.start_z);
   for (int i = 0; i < myObject.size(); i++)
   {
      if (myObject[i].first.compare(s) == 0)
      {
         glRotatef(myObject[i].second.camObj.getRotatedX(), 1, 0, 0);

         glRotatef(myObject[i].second.camObj.getRotatedY(), 0, 1, 0);
         glRotatef(myObject[i].second.camObj.getRotatedZ(), 0, 0, 1);
         glTranslatef(-myObject[i].second.camObj.getPositionX(), -
myObject[i].second.camObj.getPositionY(), -
myObject[i].second.camObj.getPositionZ());
         i = myObject.size();
      }
   }
   if (s.compare("lantai") == 0) displayLantai();
   if (s.compare("balon") == 0) displayBalon();
   if (s.compare("dispenser") == 0) displayDispenser();
   if (s.compare("kulkas") == 0) displayKulkas();
   if (s.compare("lampu") == 0) displayLampu();
   if (s.compare("mejabundar") == 0) displayMejaBundar();
   if (s.compare("mejakantin") == 0) displayMejaKantin();
   if (s.compare("mejamakan") == 0) displayMejaMakan();
   if (s.compare("minumankaleng") == 0) displayMinumanKaleng();
   if (s.compare("mug") == 0) displayMug();
   if (s.compare("rak") == 0) displayRak();
}

void displayLantai()
{
   //credit to : Adhipur & Ratih
   glScaled(3, 3, 3);
   if (myMap["lantai"] == 0) myMap["lantai"] = 1;
   if (myMap["lantai"] == 1)
load("./object/lantaidantembok/lantai&tembok.obj");
}

void displayBalon()
{
   //credit to : Fadrian
   if (myMap["balon"] == 0) myMap["balon"] = 1;
   if (myMap["balon"] == 1) load("./object/balon/balon2.obj");
   else if (myMap["balon"] == 2) load("./object/balon/balon3.obj");
```

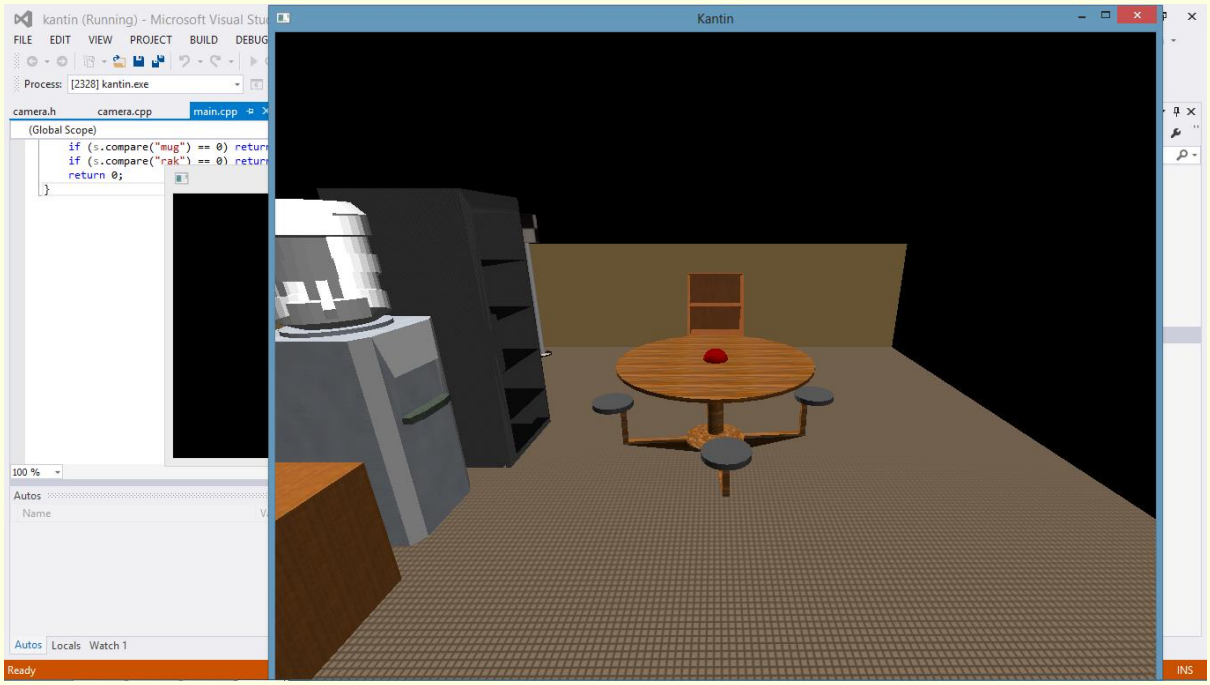
```
}

void displayDispenser()
{
   //credit to : Fadrian & Adhipur
   if (myMap["dispenser"] == 0) myMap["dispenser"] = 1;
   if (myMap["dispenser"] == 1)
load("./object/dispenser/dispenser1.obj");
   else if (myMap["dispenser"] == 2)
load("./object/dispenser/dispenser3.obj");
}

void displayKulkas()
{
   //credit to : Adhipur
   if (myMap["kulkas"] == 0) myMap["kulkas"] = 1;
   if (myMap["kulkas"] == 1) load("./object/kulkas/kulkas1.obj");
   else if (myMap["kulkas"] == 2)
load("./object/kulkas/kulkas2.obj");
}

void displayLampu()
{
   //credit to : Ratih
   if (myMap["lampu"] == 0) myMap["lampu"] = 1;
   if (myMap["lampu"] == 1) load("./object/lampu/lampe.obj");
}

void displayMejaBundar()
{
   //credit to : Adhipur
   if (myMap["mejabundar"] == 0) myMap["mejabundar"] = 1;
   if (myMap["mejabundar"] == 1)
load("./object/mejabundar/mejabundar1.obj");
   else if (myMap["mejabundar"] == 2)
load("./object/mejabundar/mejabundar2.obj");
   else if (myMap["mejabundar"] == 3)
load("./object/mejabundar/mejabundar3.obj");
}

void displayMejaKantin()
{
   //credit to : Ratih
   if (myMap["mejakantin"] == 0) myMap["mejakantin"] = 1;
   if (myMap["mejakantin"] == 1)
load("./object/mejakantin/mejakantin1.obj");
   else if (myMap["mejakantin"] == 2)
load("./object/mejakantin/mejakantin2.obj");
   else if (myMap["mejakantin"] == 3)
load("./object/mejakantin/mejakantin3.obj");
}

void displayMejaMakan()
{
   //credit to : Fadrian
   if (myMap["mejamakan"] == 0) myMap["mejamakan"] = 1;
   if (myMap["mejamakan"] == 1)
load("./object/mejamakan/mejamakan1.obj");
   else if (myMap["mejamakan"] == 2)
load("./object/mejamakan/mejamakan2.obj");
}
```

```
void displayMinumanKaleng()
{
   //credit to : Adhipur
   if (myMap["minumankaleng"] == 0) myMap["minumankaleng"] = 1;
   if (myMap["minumankaleng"] == 1)
load("./object/minumankaleng/minumankaleng1.obj");
   else if (myMap["minumankaleng"] == 2)
load("./object/minumankaleng/minumankaleng2.obj");
   else if (myMap["minumankaleng"] == 3)
load("./object/minumankaleng/minumankaleng3.obj");
}

void displayMug()
{
   //credit to : Fadrian
   if (myMap["mug"] == 0) myMap["mug"] = 1;
   if (myMap["mug"] == 1) load("./object/mug/mug1.obj");
   else if (myMap["mug"] == 2) load("./object/mug/mug2.obj");
}

void displayRak()
{
   //credit to : Adhipur
   glScalef(0.6, 0.6, 0.6);
   if (myMap["rak"] == 0) myMap["rak"] = 1;
   if (myMap["rak"] == 1) load("./object/rak/rak1.obj");
   else if (myMap["rak"] == 2) load("./object/rak/rak2.obj");
   else if (myMap["rak"] == 3) load("./object/rak/rak3.obj");
}

int maxTexture(string s)
{
   if (s.compare("balon") == 0) return MAX_Balon;
   if (s.compare("dispenser") == 0) return MAX_Dispenser;
   if (s.compare("kulkas") == 0) return MAX_Kulkas;
   if (s.compare("lampu") == 0) return MAX_Lampu;
   if (s.compare("mejabundar") == 0) return MAX_MejaBundar;
   if (s.compare("mejakantin") == 0) return MAX_MejaKantin;
   if (s.compare("mejamakan") == 0) return MAX_MejaMakan;
   if (s.compare("minumankaleng") == 0) return MAX_MinumanKaleng;
   if (s.compare("mug") == 0) return MAX_Mug;
   if (s.compare("rak") == 0) return MAX_Rak;
   return 0;
}
```

PENGGUNAAN APLIKASI

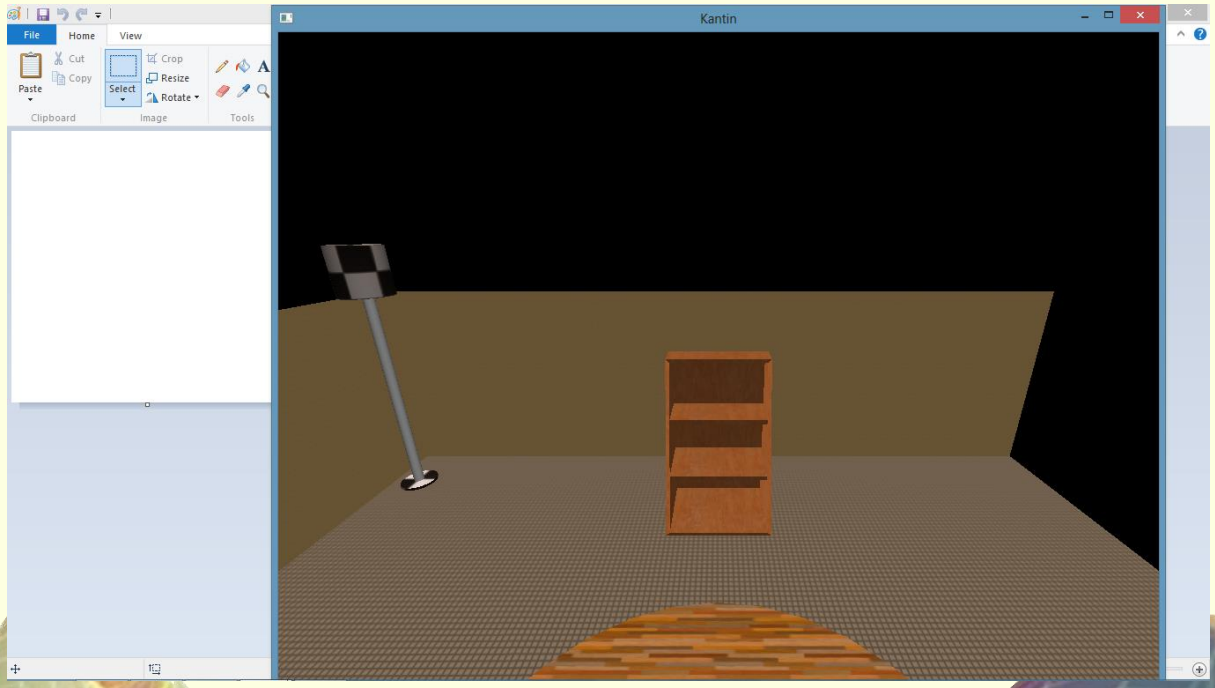
3.1 Tampilan Aplikasi

1. Tampilan Awal Aplikasi



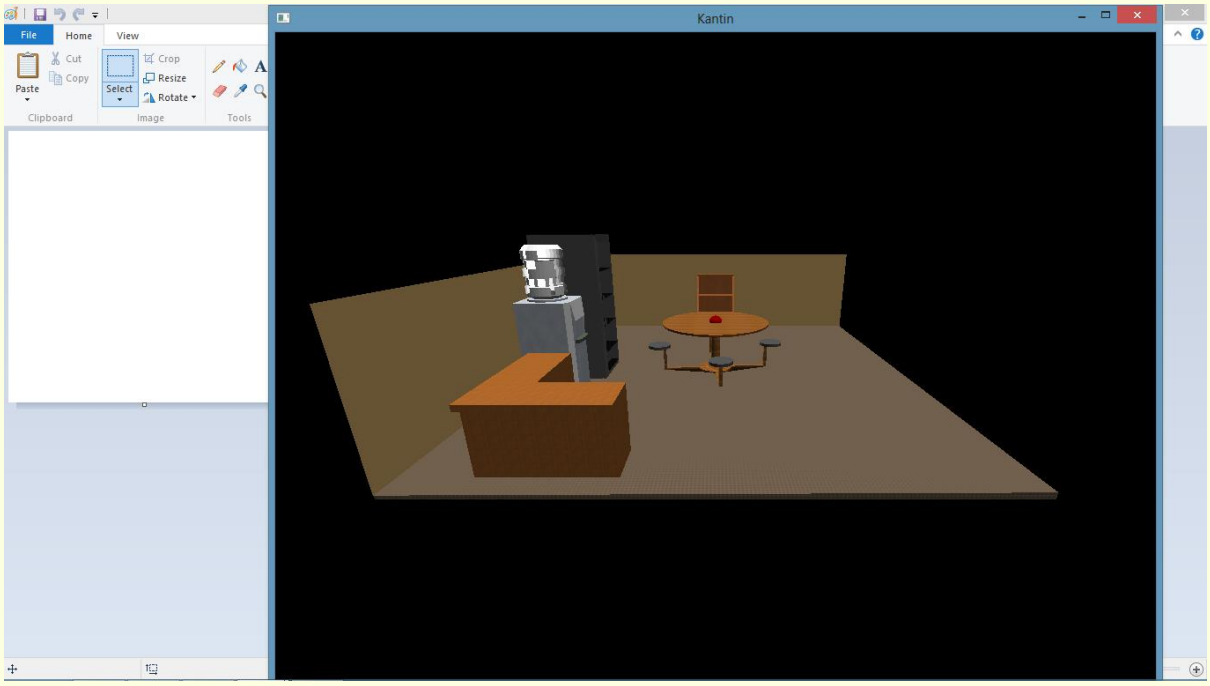
Gambar 3.1 Tampilan Awal

2. Zoom-In



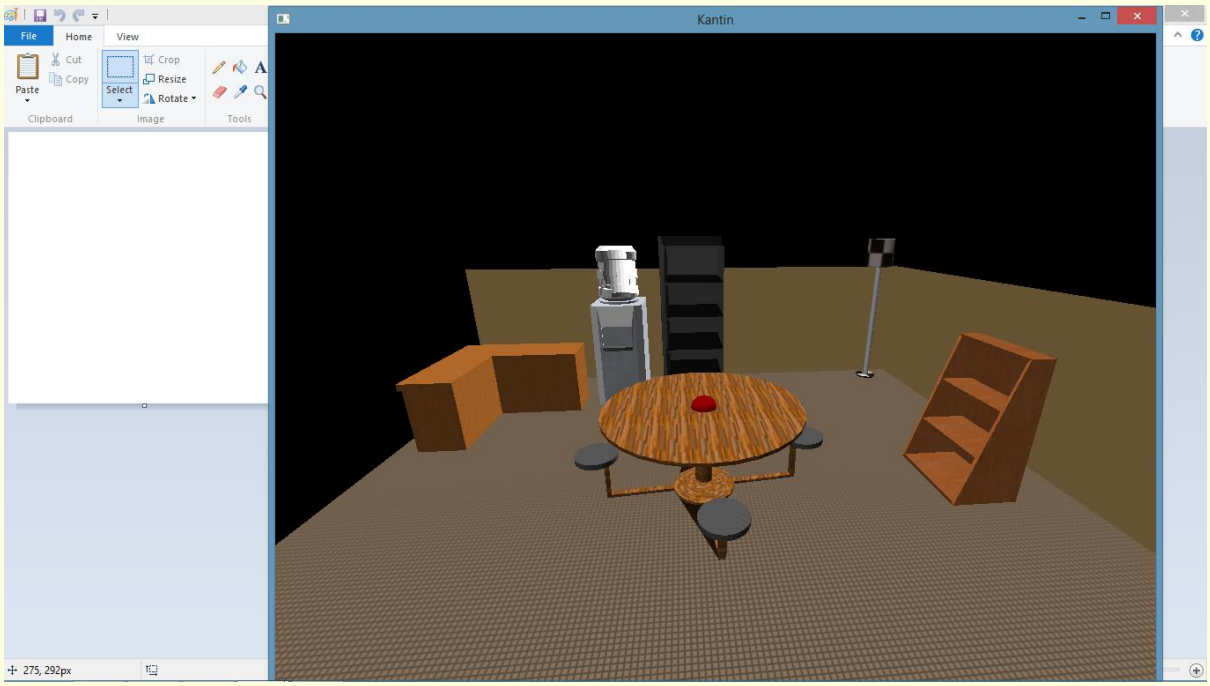
Gambar 3.2 Zoom-In

3. Zoom-Out



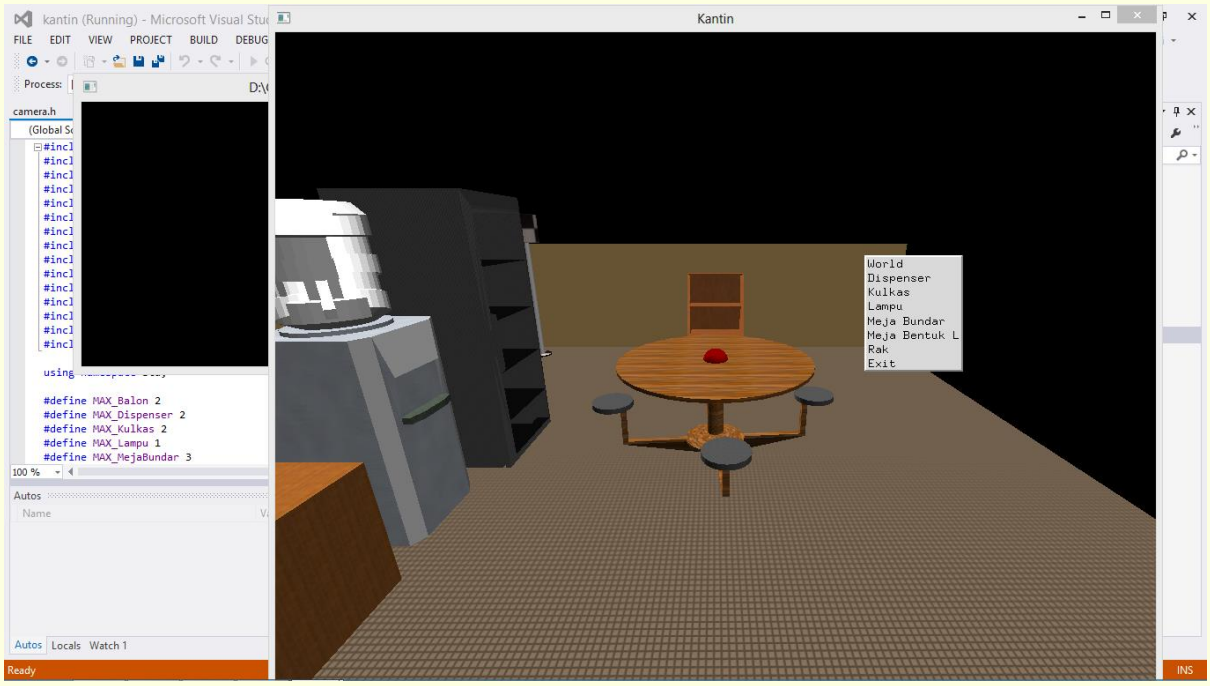
Gambar 3.3 Zoom-Out

4. Rotasi View/World



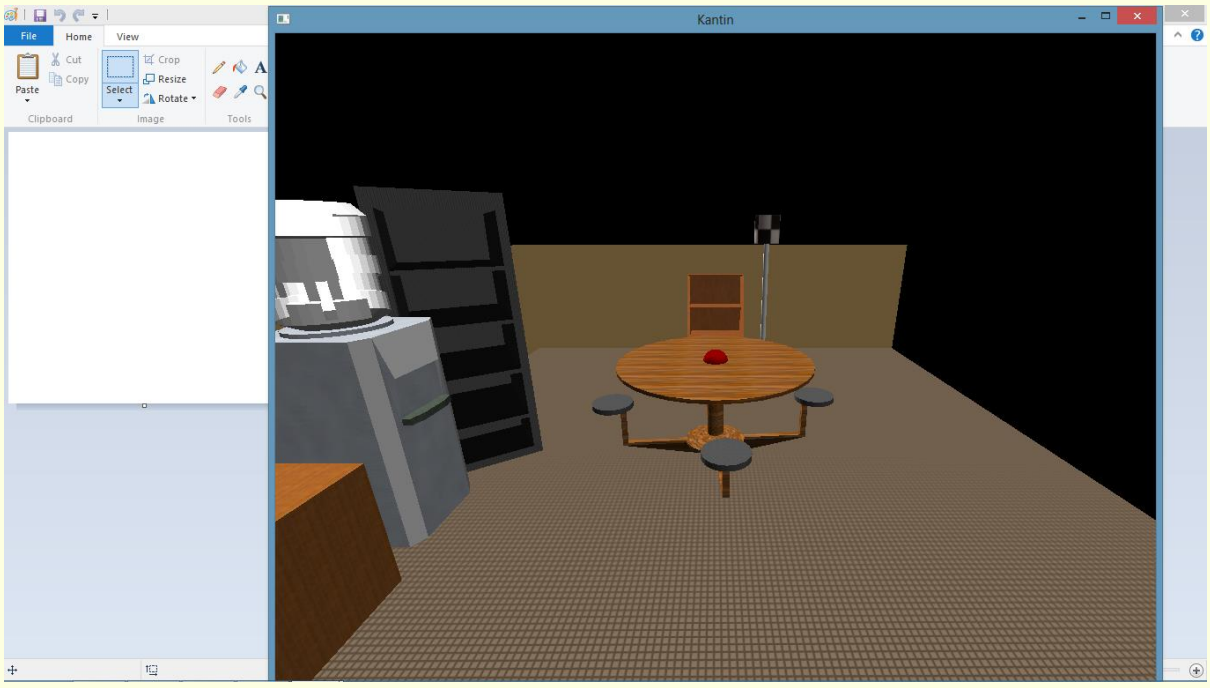
Gambar 3.4 Rotasi View

5. Menu (Klik Kanan) untuk Pilih Objek



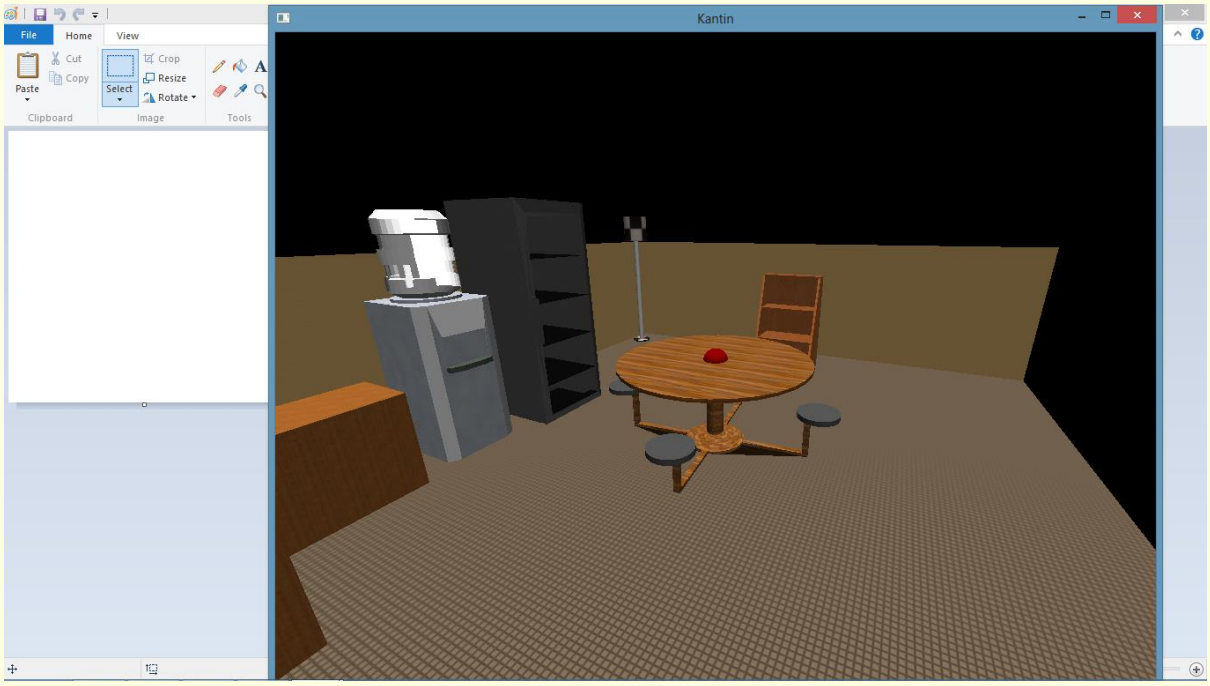
Gambar 3.5 Tampilan Menu

6. Translasi dan Rotasi Objek (Kulkas dan Lampu)



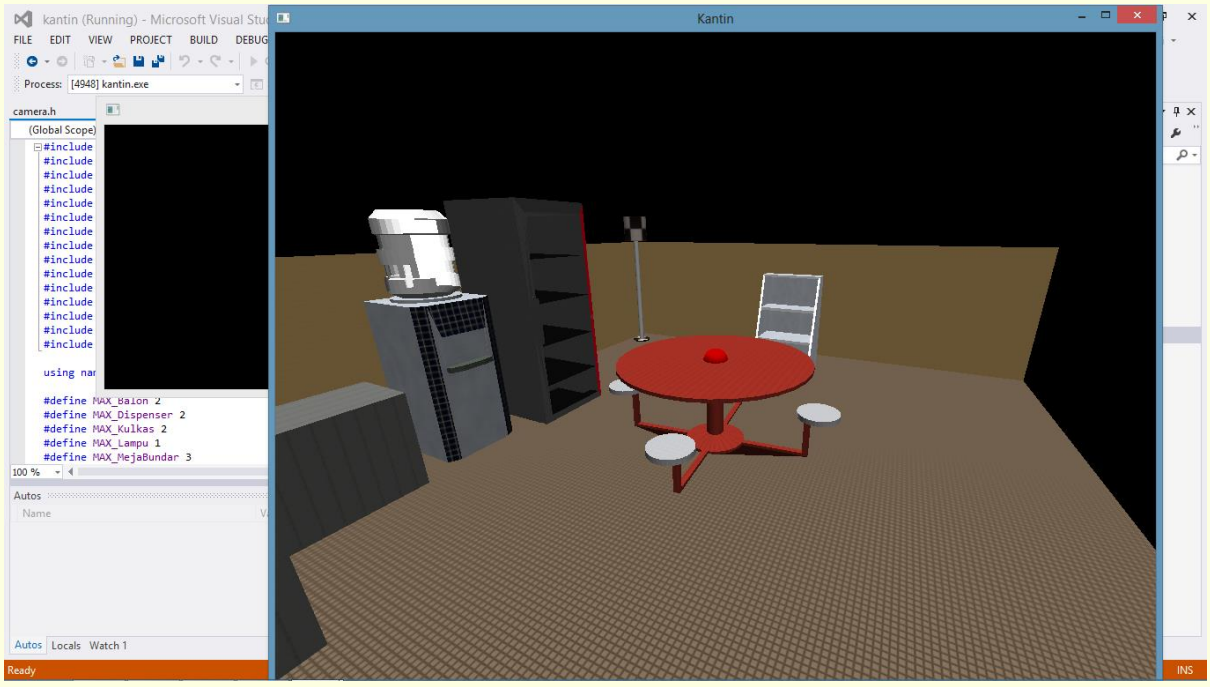
Gambar 3.6 Hasil Translasi dan Rotasi

7. Tekstur 1



Gambar 3.7 Hasil Tekstur 1

8. Tekstur 2



Gambar 3.8 Hasil Tekstur 2

3.2 Penggunaan Aplikasi

1. Kontrol untuk objek

- W = Translasi objek relatif ke depan
- A = Translasi objek relatif ke kiri
- S = Translasi objek relatif ke belakang
- D = Translasi objek relatif ke kanan
- (Atas) = Translasi objek ke atas
- (Kiri) = Rotasi objek searah jarum jam terhadap sumbu-y
- (Bawah) = Translasi objek ke bawah
- (Kanan) = Rotasi objek berlawanan arah jarum jam terhadap sumbu-y

- 1 = Pilihan Tekstur 1
- 2 = Pilihan Tekstur 2
- 3 = Pilihan Tekstur 3

2. Kontrol untuk *World*

- W = Translasi objek relatif ke belakang
- A = Translasi objek relatif ke kanan
- S = Translasi objek relatif ke depan
- D = Translasi objek relatif ke kiri
- (Atas) = Translasi objek ke bawah
- (Kiri) = Rotasi objek berlawanan arah jarum jam terhadap sumbu-y
- (Bawah) = Translasi objek ke atas
- (Kanan) = Rotasi objek searah jarum jam terhadap sumbu-y

DAFTAR PUSTAKA

Crocoll, Pillipp, *Camera with OpenGL, viewed 28 Desember 2014,* < http://codecolony.de >.

Angel, Edrward & Shreiner, Dave 2012, “Interactive Computer Graphics”, Pearson, *viewed 29 Desember 2014,* < http://lp.if.its.ac.id/freeshare/Kuliah/2014-2015%20Gasal/S1/Grafika%20kelas%20E/Angel%20-%20New%20Edition.pdf >.