**Aplikasi Aljabar Lanjar pada Metode Numerik**

**Aljabar Geometri**

**Dibuat Untuk Melengkapi Tugas Besar IF-2123**

**Oleh**

**Yonas Adiel Wiguna K-03 13516030**

**Senapati Sang Diwangkara K-02 13516107**

****

**Institut Teknologi Bandung**

**Sekolah Teknik Elektro dan Informatika**

**Bandung**

**2017**

BAB I

DESKRIPSI MASALAH

Tugas besar ini adalah membuat program yang mensimulasikan transformasi linier untuk melakukan operasi translasi, refleksi, dilatasi, rotasi, dan sebagainya pada sebuah bidang 2D. Bidang dibuat dengan mendefinisikan sekumpulan titik sudut lalu membuat bidang dari titik-titik tersebut.

Program akan memiliki dua buah window, window pertama (*command prompt*) berfungsi untuk

menerima input dari user, sedangkan window kedua (*GUI*) berfungsi untuk menampilkan output berdasarkan input dari user. Kedua window ini muncul ketika user membuka file *executable*.

Saat program baru mulai dijalankan, program akan menerima input N, yaitu jumlah titik yang akan diterima. Berikutnya, program akan menerima input N buah titik tersebut (pasangan nilai x dan y). Setelah itu program akan menampilkan output sebuah bidang yang dibangkitkan dari titik-titik tersebut. Selain itu juga ditampilkan dua buah garis, yaitu sumbu x dan sumbu y. Nilai x dan y memiliki rentang minimal - 500 pixel dan maksikum 500 pixel.

Berikutnya, program dapat menerima input yang mentransformasi bidang tersebut:

* translasi bidang
* diltasi bidang
* rotasi bidang terhadap sembarang titik
* refleksi bidang terhadap garis
* operasi *shear* pada bidang
* operasi *stretch* pada bidang
* perkalian matriks terhadap titik-titik sudut bidang
* beberapa operasi di atas yang dijalankan sekaligus
* me*reset* bidang
* keluar dari program

BAB II

TEORI DASAR

Persoalan - persoalan transformasi poligon dapat diselesaikan dengan mentransformasikan semua titik sudut poligon dengan matriks transformasi yang sama.

Dengan demikian, semua operasi dapat disederhanakan menjadi perkalian vektor (Objek Titik) dan matriks (Objek Matriks). Objek Poligon sendiri merupakan sekuens dari Objek Titik. Disebut sekuens karena masing-masing sisi hanya ada pada titik i dan j jika dan hanya jika atau keduanya titik awal dan akhir dalam sekuens.

Transformasi matriks didefinisikan sebagai matriks yang memetakkan semua titik di bidang kartesian ke tepat satu titik di bidang kartesian juga dan memenuhi persamaan

Selanjutnya untuk tiap oeprasi dibutuhkan matriks tranformasinya. Diperoleh matriks transformasi[3] untuk beberapa operasi:

* dilatasi:
* rotasi terhadap (0,0) sebesar derajat:
* refleksi terhadap vektor :
* operasi *shear* sejajar sumbu x:
* operasi *shear* sejajar sumbu y:
* operasi *stretch* sejajar sumbu x:
* operasi *stretch* sejajar sumbu y:

Matriks transformasi refleksi titik terhadap sumbu x, sumbu y, garis x=y, dan garis x=-y dapat diturunkan menggunakan matriks refleksi di atas. Untuk sumbu x, vecktornya adalah , jadi matriksnya

Untuk sumbu y, vecktornya adalah , jadi matriksnya

Untuk garis x=y, vecktornya adalah , jadi matriksnya

Untuk garis x=-y, vecktornya adalah , jadi matriksnya

Sedangkan, untuk translasi dapat dilakukan penambahan titik terhadap titik lain, sebagai contoh bila titik (x,y) digeser sejauh dx di sumbu x dan dy di sumbu y, dapat dilakukan operasi[4]:

Dengan adanya matriks translasi, kita dapat merotasi titik dengan pusat titik (a,b) dengan mentranslasikan terlebih dahulu dengan (-a,-b), lalu rotasi terhadap pusat, dan akhirnya translasi kembali terhadap(a,b).

OpenGL adalah library yang tersedia untuk C++ dan Python supaya programmer dapat menggambar sebuah bidang di window baru.

Pada dasarnya, OpenGL menerima input bidang apa yang ingin digambar (dalam kasus ini sebuah poligon) lalu meminta letak titik-titik sudutnya. Proses ini dimasukkan ke dalam fungsi render yang akan dipanggiloleh program utama. Ketika berjalan, program utama akan terus mengulang fungsi render untuk dapat menampilkan animasi.

BAB III

IMPLEMENTASI DALAM C++

Berikut implementasi program dalam bahasa C++ dengan menggunakan *library* freeGLUT. Program dibuat modular dengan menggunakan class Titik, Matriks, dan Polygon.

Class Matriks:

/\*\*

\* Matriks2 Data Type

\*

\*/

#include "matriks2.h"

Matriks2**::**Matriks2**(**int row**)** **{**

setRow**(**row**);**

m\_col **=** 2**;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GETTER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Matriks2**::**getRow**()** **{**

**return** m\_row**;**

**}**

int Matriks2**::**getCol**()** **{**

**return** m\_col**;**

**}**

float Matriks2**::**getElmt**(**int row**,** int col**)** **{**

**return** m\_mat**[**row**][**col**];**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SETTER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Matriks2**::**setRow**(**int row**)** **{**

m\_row **=** row**;**

**}**

void Matriks2**::**setElmt**(**int row**,** int col**,** float elmt**)**

**{**

m\_mat**[**row**][**col**]** **=** elmt**;**

**}**

void Matriks2**::**setAll**(**float mat**[**2**][**2**])** **{**

**for** **(**int i**=**0**;** i**<**2**;** i**++)** **{**

**for** **(**int j**=**0**;** j**<**2**;** j**++)** **{**

setElmt**(**i**,**j**,**mat**[**i**][**j**]);**

**}**

**}**

**}**

Class Titik:

/\*\*

\* Titik2 Data Type

\*

\*/

#include "titik2.h"

#include "matriks2.h"

#include <cmath>

Titik2**::**Titik2**(**float x**,** float y**)** **{**

setX**(**x**);**

setY**(**y**);**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GETTER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float Titik2**::**getX**()** **{**

**return** m\_x**;**

**}**

float Titik2**::**getY**()** **{**

**return** m\_y**;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SETTER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Titik2**::**setX**(**float x**)** **{**

m\_x **=** x**;**

**}**

void Titik2**::**setY**(**float y**)** **{**

m\_y **=** y**;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\* OPERATION \*\*\*\*\*\*\*\*\*\*\*\*\*/

void Titik2**::**translate**(**float dx**,** float dy**)** **{**

setX**(**getX**()** **+** dx**);**

setY**(**getY**()** **+** dy**);**

**}**

void Titik2**::**dilate**(**float k**)** **{**

Matriks2 m**(**2**);**

float mat**[**2**][**2**]** **=** **{**

**{** k**,** 0.0f**},**

**{**0.0f**,** k**}**

**};**

m**.**setAll**(**mat**);**

transform**(**m**);**

**}**

void Titik2**::**rotateOrigin**(**float deg**)** **{**

Matriks2 m**(**2**);**

float mat**[**2**][**2**]** **=** **{**

**{**cos**(**rad**(**deg**)),** **-**sin**(**rad**(**deg**))},**

**{**sin**(**rad**(**deg**)),** cos**(**rad**(**deg**))}**

**};**

m**.**setAll**(**mat**);**

transform**(**m**);**

**}**

void Titik2**::**rotate**(**float deg**,** float c\_x**,** float c\_y**)** **{**

translate**(-**c\_x**,** **-**c\_y**);**

rotateOrigin**(**deg**);**

translate**(**c\_x**,** c\_y**);**

**}**

void Titik2**::**reflectByLine**(**int type**)** **{**

Matriks2 m**(**2**);**

float mat**[**2**][**2**]** **=** **{{**0**,**0**},{**0**,**0**}};**

**if** **(**type **==** 0**)** **{**

mat**[**0**][**0**]** **=** 1**;**

mat**[**1**][**1**]** **=** **-**1**;**

**}** **else** **if** **(**type **==** 1**)** **{**

mat**[**0**][**0**]** **=** **-**1**;**

mat**[**1**][**1**]** **=** 1**;**

**}** **else** **if** **(**type **==** 2**)** **{**

mat**[**0**][**1**]** **=** 1**;**

mat**[**1**][**0**]** **=** 1**;**

**}** **else** **if** **(**type **==** 3**)** **{**

mat**[**0**][**1**]** **=** **-**1**;**

mat**[**1**][**0**]** **=** **-**1**;**

**}**

m**.**setAll**(**mat**);**

transform**(**m**);**

**}**

void Titik2**::**reflectByPoint**(**float c\_x**,** float c\_y**)** **{**

setX**(**2**\***c\_x **-** getX**());**

setY**(**2**\***c\_y **-** getY**());**

**}**

void Titik2**::**shearByX**(**float k**)** **{**

Matriks2 m**(**2**);**

float mat**[**2**][**2**]** **=** **{**

**{**1**,** k**},**

**{**0**,** 1**}**

**};**

m**.**setAll**(**mat**);**

transform**(**m**);**

**}**

void Titik2**::**shearByY**(**float k**)** **{**

Matriks2 m**(**2**);**

float mat**[**2**][**2**]** **=** **{**

**{**1**,** 0**},**

**{**k**,** 1**}**

**};**

m**.**setAll**(**mat**);**

transform**(**m**);**

**}**

void Titik2**::**stretchByX**(**float k**)** **{**

Matriks2 m**(**2**);**

float mat**[**2**][**2**]** **=** **{**

**{**k**,** 0**},**

**{**0**,** 1**}**

**};**

m**.**setAll**(**mat**);**

transform**(**m**);**

**}**

void Titik2**::**stretchByY**(**float k**)** **{**

Matriks2 m**(**2**);**

float mat**[**2**][**2**]** **=** **{**

**{**1**,** 0**},**

**{**0**,** k**}**

**};**

m**.**setAll**(**mat**);**

transform**(**m**);**

**}**

void Titik2**::**customTransform**(**float a**,** float b**,** float c**,** float d**)** **{**

Matriks2 m**(**2**);**

float mat**[**2**][**2**]** **=** **{**

**{**a**,** b**},**

**{**c**,** d**}**

**};**

m**.**setAll**(**mat**);**

transform**(**m**);**

**}**

void Titik2**::**transform**(**Matriks2 m**)** **{**

float x **=** getX**();**

float y **=** getY**();**

setX**(**x **\*** m**.**getElmt**(**0**,**0**)** **+** y **\*** m**.**getElmt**(**0**,**1**));**

setY**(**x **\*** m**.**getElmt**(**1**,**0**)** **+** y **\*** m**.**getElmt**(**1**,**1**));**

**}**

Class Polygon:

/\*\*

\* Poly2 Data Type

\*

\*/

#include <stdlib.h>

#include "titik2.h"

#include "matriks2.h"

#include "poly2.h"

Poly2**::**Poly2**(**int n**)** **{**

m\_n\_edges **=** n**;**

m\_corners **=** **(**Titik2**\*)** malloc**(**n **\*** **sizeof(**Titik2**));**

**}**

int Poly2**::**getEdge**()** **{**

**return** m\_n\_edges**;**

**}**

Titik2 Poly2**::**getCorner**(**int i**)** **{**

**return** m\_corners**[**i**];**

**}**

void Poly2**::**setEdge**(**int n**)** **{**

m\_n\_edges **=** n**;**

Titik2**\*** m\_corners\_old **=** m\_corners**;**

Titik2**\*** m\_corners\_new **=** **(**Titik2**\*)** realloc**(**m\_corners**,** n **\*** **sizeof(**Titik2**));**

m\_corners **=** m\_corners\_new**;**

free**(**m\_corners\_old**);**

**}**

void Poly2**::**setCorner**(**int i**,** Titik2 corner**)** **{**

m\_corners**[**i**]** **=** corner**;**

**}**

void Poly2**::**translate**(**float dx**,** float dy**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**translate**(**dx**,** dy**);**

setCorner**(**i**,** corner**);**

**}**

**}**

void Poly2**::**animateTranslate**(**float dx**,** float dy**,** int step**)** **{**

translate**(**dx**\***step**/**ANIMATION\_STEP**,** dy**\***step**/**ANIMATION\_STEP**);**

**}**

void Poly2**::**dilate**(**float k**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**dilate**(**k**);**

setCorner**(**i**,** corner**);**

**}**

**}**

void Poly2**::**animateDilate**(**float k**,** int step**)** **{**

dilate**(**k**\***step**/**ANIMATION\_STEP**);**

**}**

void Poly2**::**rotateOrigin**(**float deg**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**rotateOrigin**(**deg**);**

setCorner**(**i**,** corner**);**

**}**

**}**

void Poly2**::**animateRotateOrigin**(**float deg**,** int step**)** **{**

rotateOrigin**(**deg**\***step**/**ANIMATION\_STEP**);**

**}**

void Poly2**::**rotate**(**float deg**,** float c\_x**,** float c\_y**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**rotate**(**deg**,** c\_x**,** c\_y**);**

setCorner**(**i**,** corner**);**

**}**

**}**

void Poly2**::**animateRotate**(**float deg**,** float c\_x**,** float c\_y**,** int step**)** **{**

rotate**(**deg**\***step**/**ANIMATION\_STEP**,** c\_x**,** c\_y**);**

**}**

void Poly2**::**reflectByLine**(**int type**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**reflectByLine**(**type**);**

setCorner**(**i**,** corner**);**

**}**

**}**

void Poly2**::**reflectByPoint**(**float c\_x**,** float c\_y**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**reflectByPoint**(**c\_x**,** c\_y**);**

setCorner**(**i**,** corner**);**

**}**

**}**

void Poly2**::**shearByX**(**float k**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**shearByX**(**k**);**

setCorner**(**i**,** corner**);**

**}**

**}**

void Poly2**::**animateShearByX**(**float k**,** int step**)** **{**

shearByX**(**k**\***step**/**ANIMATION\_STEP**);**

**}**

void Poly2**::**shearByY**(**float k**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**shearByY**(**k**);**

setCorner**(**i**,** corner**);**

**}**

**}**

void Poly2**::**animateShearByY**(**float k**,** int step**)** **{**

shearByX**(**k**\***step**/**ANIMATION\_STEP**);**

**}**

void Poly2**::**stretchByX**(**float k**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**stretchByX**(**k**);**

setCorner**(**i**,** corner**);**

**}**

**}**

void Poly2**::**animateStretchByX**(**float k**,** int step**)** **{**

stretchByX**(**k**\***step**/**ANIMATION\_STEP**);**

**}**

void Poly2**::**stretchByY**(**float k**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**stretchByY**(**k**);**

setCorner**(**i**,** corner**);**

**}**

**}**

void Poly2**::**animateStretchByY**(**float k**,** int step**)** **{**

stretchByY**(**k**\***step**/**ANIMATION\_STEP**);**

**}**

void Poly2**::**customTransform**(**float a**,** float b**,** float c**,** float d**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**customTransform**(**a**,** b**,** c**,** d**);**

setCorner**(**i**,** corner**);**

**}**

**}**

void Poly2**::**animateCustomTransform**(**float a**,** float b**,** float c**,** float d**,** int step**)** **{**

customTransform**(**

a**\***step**/**ANIMATION\_STEP**,**

b**\***step**/**ANIMATION\_STEP**,**

c**\***step**/**ANIMATION\_STEP**,**

d**\***step**/**ANIMATION\_STEP

**);**

**}**

void Poly2**::**transform**(**Matriks2 m**)** **{**

**for** **(**int i**=**0**;** i**<**getEdge**();** i**++)** **{**

Titik2 corner **=** getCorner**(**i**);**

corner**.**transform**(**m**);**

setCorner**(**i**,** corner**);**

**}**

**}**

Program utama:

#include <GL/glut.h>

#include <math.h>

#include <iostream>

#include <windows.h>

#include <stdio.h>

#include "inc/poly2.h"

#include "inc/titik2.h"

void transform**(**std**::**string cmd**);**

// Globals

Poly2 polygon**(**1**);**

void render**(**void**)** **{** /\* function called whenever redisplay needed \*/

glMatrixMode**(**GL\_MODELVIEW**);**

glClear**(**GL\_COLOR\_BUFFER\_BIT**);** // clear the drawing buffer.

glLoadIdentity**();**

gluOrtho2D**(-**500.0**,** 500.0**,** **-**500.0**,** 500.0**);**

glBegin**(**GL\_POLYGON**);** **{**

glColor3f**(**1.0f**,** 0.0f**,** 0.0f**);**

**for** **(**int i **=** 0**;** i **<** polygon**.**getEdge**();** i**++)** **{**

glVertex2f**(**polygon**.**getCorner**(**i**).**getX**(),** polygon**.**getCorner**(**i**).**getY**());**

**}**

**}** glEnd**();**

// Garis-garis sumbu

glBegin**(**GL\_LINES**);** **{** // Sumbu X, merah

glColor3f**(**1.0f**,**0.0f**,**0.0f**);**

glVertex2f**(-**500.0**,** 0.0**);**

glVertex2f**(**500.0**,** 0.0**);**

glVertex2f**(**500.0**,** 0.0**);**

glVertex2f**(**490.0**,** 10.0**);**

glVertex2f**(**500.0**,** 0.0**);**

glVertex2f**(**490.0**,** **-**10.0**);**

// Sumbu Y, ijo

glColor3f**(**0.0f**,**1.0f**,**0.0f**);**

glVertex2f**(**0.0**,** **-**500.0**);**

glVertex2f**(**0.0**,** 500.0**);**

glVertex2f**(**0.0**,** 500.0**);**

glVertex2f**(**10.0**,** 490.0**);**

glVertex2f**(**0.0**,** 500.0**);**

glVertex2f**(-**10.0**,** 490.0**);**

**}** glEnd**();**

glFlush**();** /\* Complete any pending operations \*/

**}**

void resizeWindow**(**int width**,** int height**)** **{** // fungsi biar pas windownya diresize jadinya gak ngestretch

GLfloat aspect **=** **(**GLfloat**)**width **/** **(**GLfloat**)**height**;**

// Set the viewport to cover the new window

glViewport**(**0**,** 0**,** width**,** height**);**

// Set the aspect ratio of the clipping area to match the viewport

glMatrixMode**(**GL\_PROJECTION**);** // To operate on the Projection matrix

glLoadIdentity**();** // Reset the projection matrix

**if** **(**width **>=** height**)** **{**

// aspect >= 1, set the height from -1 to 1, with larger width

gluOrtho2D**(-**1.0 **\*** aspect**,** 1.0 **\*** aspect**,** **-**1.0**,** 1.0**);**

**}** **else** **{**

// aspect < 1, set the width to -1 to 1, with larger height

gluOrtho2D**(-**1.0**,** 1.0**,** **-**1.0 **/** aspect**,** 1.0 **/** aspect**);**

**}**

**}**

DWORD WINAPI input**(**LPVOID param**)** **{**

// Saves original polygon for reset

Poly2 original**(**polygon**.**getEdge**());**

**for** **(**int i **=** 0**;** i **<** original**.**getEdge**();** i**++)** **{**

original**.**setCorner**(**i**,** polygon**.**getCorner**(**i**));**

**}**

std**::**string cmd**;**

int n**;**

**while** **(**1 **<** 2**)** **{**

std**::**cin **>>** cmd**;**

**if** **(**cmd **==** "multiple"**)** **{**

std**::**cin **>>** n**;**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** **{**

std**::**cout **<<** "... "**;** std**::**cin **>>** cmd**;**

transform**(**cmd**);**

**}**

**}** **else** **if** **(**cmd **==** "reset"**)** **{**

**for** **(**int i **=** 0**;** i **<** polygon**.**getEdge**();** i**++)** **{**

polygon**.**setCorner**(**i**,** original**.**getCorner**(**i**));**

**}**

**}** **else** **if** **(**cmd **==** "exit"**)** **{**exit**(**0**);}**

**else** **{**transform**(**cmd**);}**

glutPostRedisplay**();**

cmd **=** ""**;**

**}**

**}**

void transform**(**std**::**string cmd**)** **{**

float cx**,** cy**,** deg**,** cz**;** // Recycleable

std**::**string parameter**;**

**if** **(**cmd **==** "translate"**)** **{**

std**::**cin **>>** cx **>>** cy**;**

polygon**.**translate**(**cx**,** cy**);**

**}** **else** **if** **(**cmd **==** "dilate"**)** **{**

std**::**cin **>>** cx**;**

polygon**.**dilate**(**cx**);**

**}** **else** **if** **(**cmd **==** "rotate"**)** **{**

std**::**cin **>>** deg **>>** cx **>>** cy**;**

polygon**.**rotate**(**deg**,** cx**,** cy**);**

**}** **else** **if** **(**cmd **==** "reflect"**)** **{**

std**::**cin **>>** parameter**;**

**if** **(**parameter **==** "x"**)** **{**polygon**.**reflectByLine**(**0**);}**

**else** **if** **(**parameter **==** "y"**)** **{**polygon**.**reflectByLine**(**1**);}**

**else** **if** **(**parameter **==** "y=x" **||** parameter **==** "x=y"**)** **{**polygon**.**reflectByLine**(**2**);}**

**else** **if** **(**parameter **==** "y=-x" **||** parameter **==** "x=-y"**)** **{**polygon**.**reflectByLine**(**3**);}**

**else** **{** // doesn't work

sscanf**(**parameter**.**c\_str**(),** "(%f,%f)"**,** **&**cx**,** **&**cy**);**

polygon**.**reflectByPoint**(**cx**,** cy**);**

**}**

**}** **else** **if** **(**cmd **==** "shear"**)** **{**

std**::**cin **>>** parameter **>>** cx**;**

**(**parameter **==** "x"**)** **?** polygon**.**shearByX**(**cx**)** **:** polygon**.**shearByY**(**cx**);**

**}** **else** **if** **(**cmd **==** "stretch"**)** **{**

std**::**cin **>>** parameter **>>** cx**;**

**(**parameter **==** "x"**)** **?** polygon**.**stretchByX**(**cx**)** **:** polygon**.**stretchByY**(**cx**);**

**}** **else** **if** **(**cmd **==** "custom"**)** **{**

std**::**cin **>>** cx **>>** cy **>>** deg **>>** cz**;**

polygon**.**customTransform**(**cx**,** cy**,** deg**,** cz**);**

**}** **else** **{**std**::**cout **<<** "Invalid command" **<<** std**::**endl**;}**

**}**

DWORD WINAPI startLoop**(**LPVOID param**)** **{**

int argc **=** 1**;**

glutInit**(&**argc**,** **NULL);**

glutInitDisplayMode**(**GLUT\_SINGLE **|** GLUT\_RGB**);**

glutInitWindowSize**(**600**,**600**);**

glutCreateWindow**(**"Algeo"**);**

glClearColor**(**1.0**,** 1.0**,** 1.0**,** 1.0**);** // background putih

glutDisplayFunc**(**render**);**

glutReshapeFunc**(**resizeWindow**);**

glutMainLoop**();**

**}**

int main**(**int argc**,** char **\***argv**[])** **{**

int Data\_Of\_Thread\_1 **=** 1**;**

int Data\_Of\_Thread\_2 **=** 2**;**

HANDLE Handle\_Of\_Thread\_1 **=** 0**;**

HANDLE Handle\_Of\_Thread\_2 **=** 0**;**

HANDLE Array\_Of\_Thread\_Handles**[**3**];**

int n**;**

std**::**cout **<<** " \_\_\_\_ \_\_\_\_\_ \_ \n"**;**

std**::**cout **<<** " / \_\_ \\ / \_\_\_\_| | \n"**;**

std**::**cout **<<** " | | | |\_ \_\_ \_\_\_ \_ \_\_ | | \_\_| | \n"**;**

std**::**cout **<<** " | | | | '\_ \\ / \_ \\ '\_ \\| | |\_ | | \n"**;**

std**::**cout **<<** " | |\_\_| | |\_) | \_\_/ | | | |\_\_| | |\_\_\_\_ \n"**;**

std**::**cout **<<** " \\\_\_\_\_/| .\_\_/ \\\_\_\_|\_| |\_|\\\_\_\_\_\_|\_\_\_\_\_\_|\n"**;**

std**::**cout **<<** " | | \n"**;**

std**::**cout **<<** " |\_| \n\n"**;**

std**::**cout **<<** "Segi berapa? "**;** std**::**cin **>>** n**;**

polygon**.**setEdge**(**n**);**

float x**,** y**;** Titik2 titikTemp**(**0.0**,** 0.0**);**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** **{**

std**::**cin **>>** x **>>** y**;**

titikTemp**.**setX**(**x**);**

titikTemp**.**setY**(**y**);**

polygon**.**setCorner**(**i**,** titikTemp**);**

**}**

Handle\_Of\_Thread\_1 **=** CreateThread**(NULL,** 0**,** startLoop**,** **NULL,** 0**,** **NULL);**

**if** **(**Handle\_Of\_Thread\_1 **==** **NULL)** ExitProcess**(**0**);**

Handle\_Of\_Thread\_2 **=** CreateThread**(NULL,** 0**,** input**,** **&**Data\_Of\_Thread\_2**,** 0**,** **NULL);**

**if** **(**Handle\_Of\_Thread\_2 **==** **NULL)** ExitProcess**(**0**);**

Array\_Of\_Thread\_Handles**[**0**]** **=** Handle\_Of\_Thread\_1**;**

Array\_Of\_Thread\_Handles**[**1**]** **=** Handle\_Of\_Thread\_2**;**

WaitForMultipleObjects**(**2**,** Array\_Of\_Thread\_Handles**,** TRUE**,** INFINITE**);**

CloseHandle**(**Handle\_Of\_Thread\_1**);**

CloseHandle**(**Handle\_Of\_Thread\_2**);**

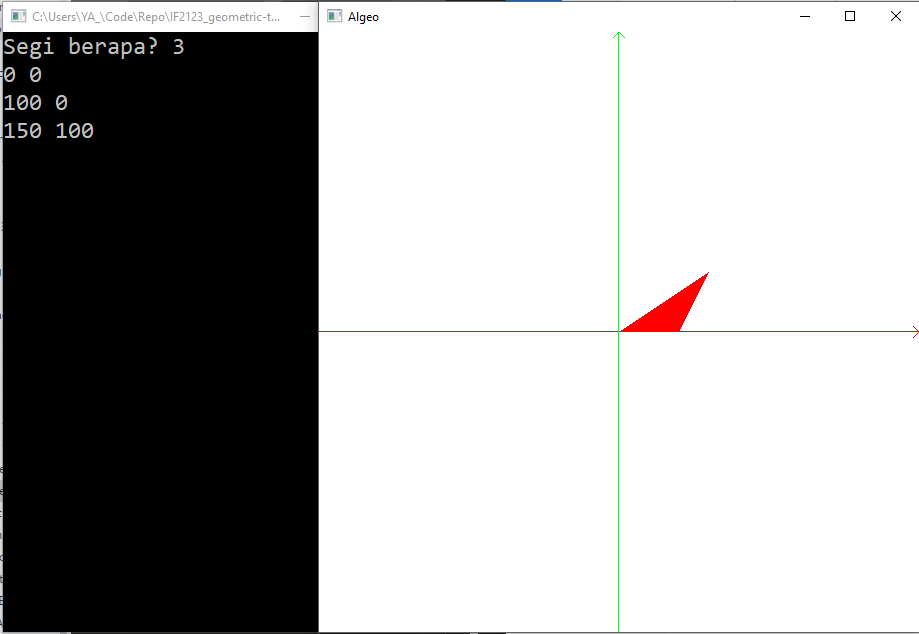
**return** 0**;** // execution never reaches this point

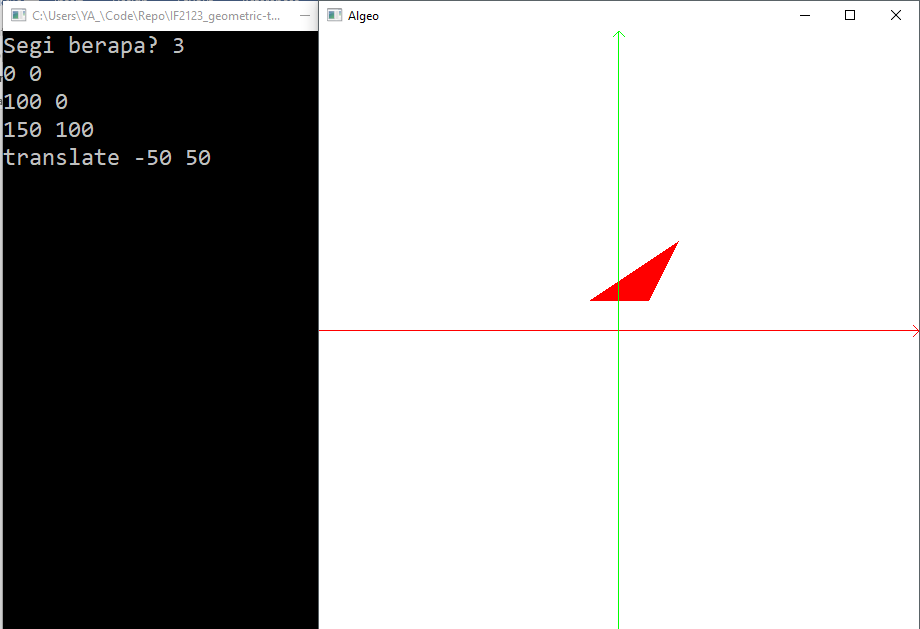
**}**

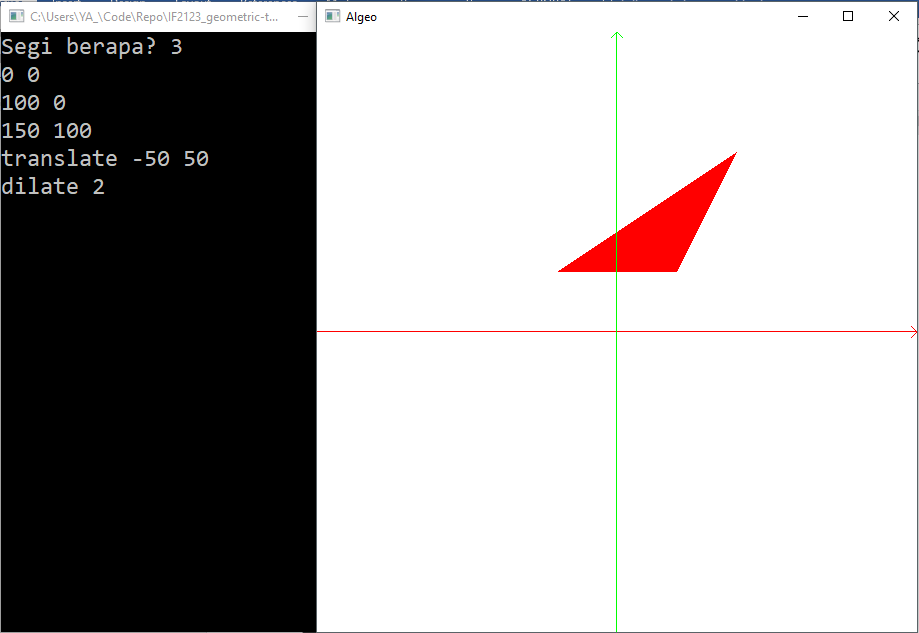
BAB IV

EKSPERIMEN

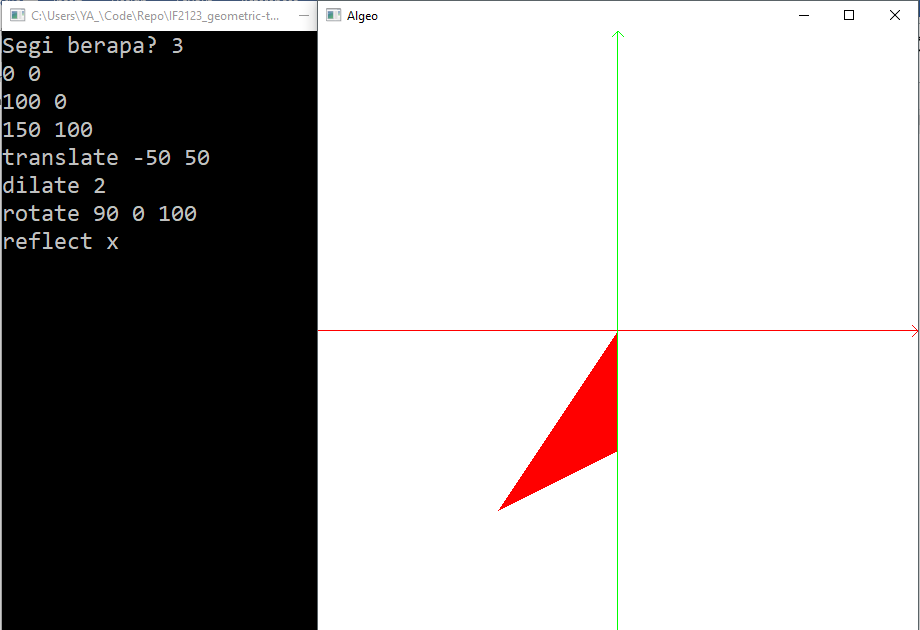
Berikut salah satu eksperimen yang dilakukan pada program beserta hasilnya:

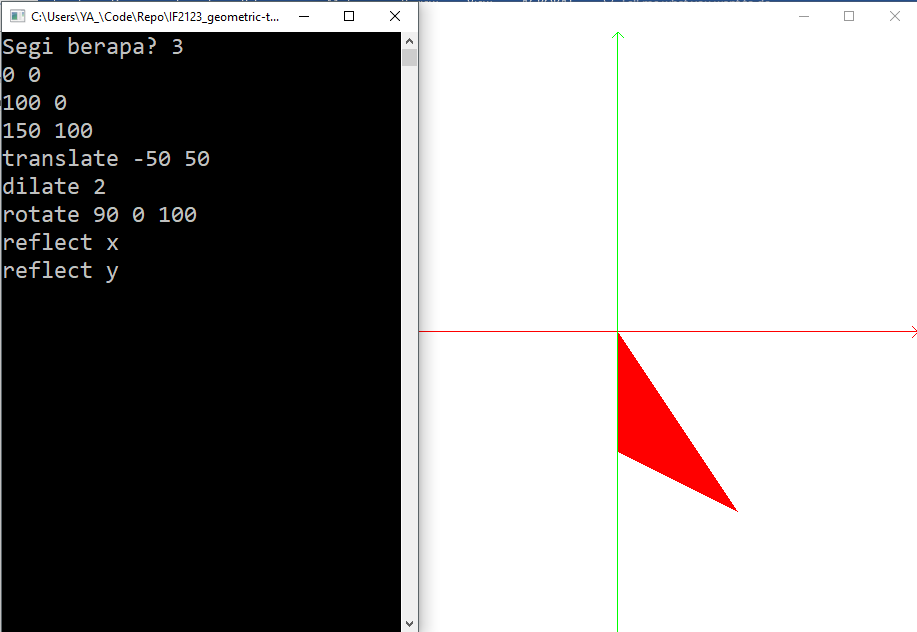


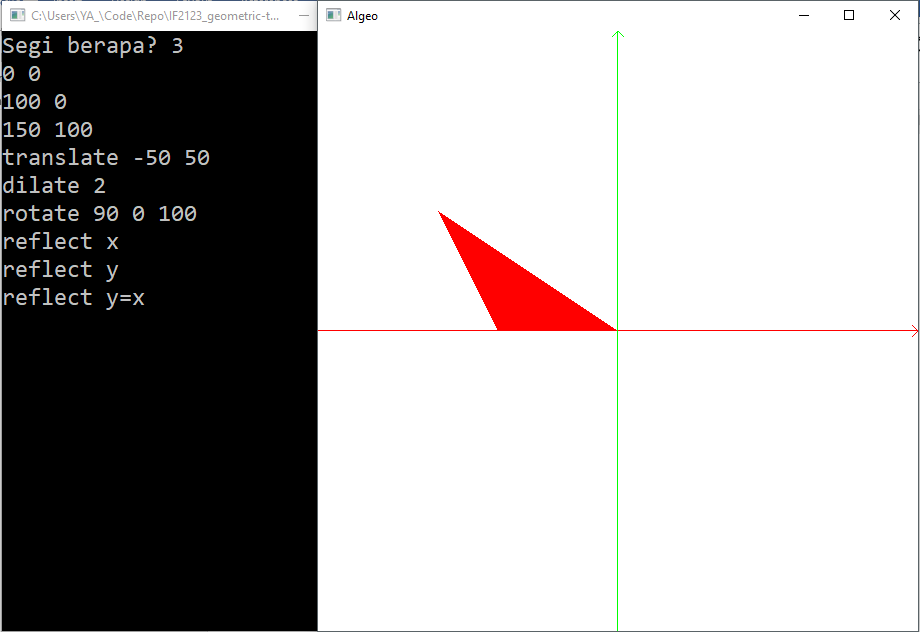


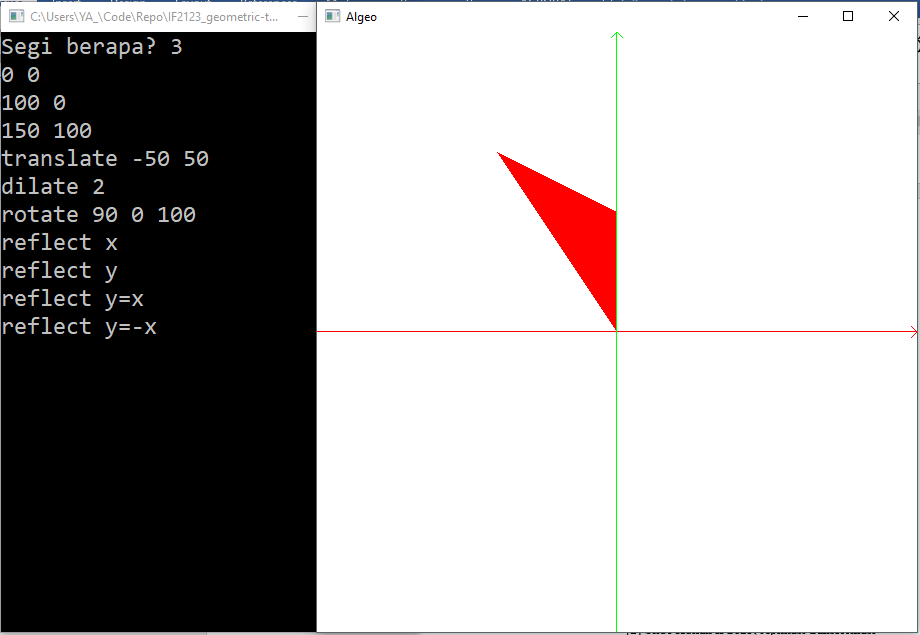






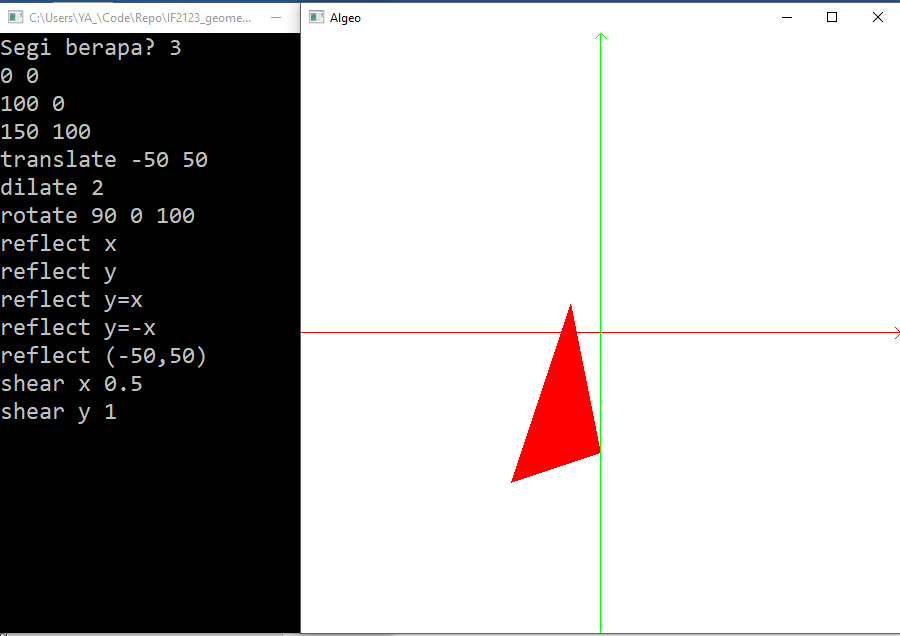


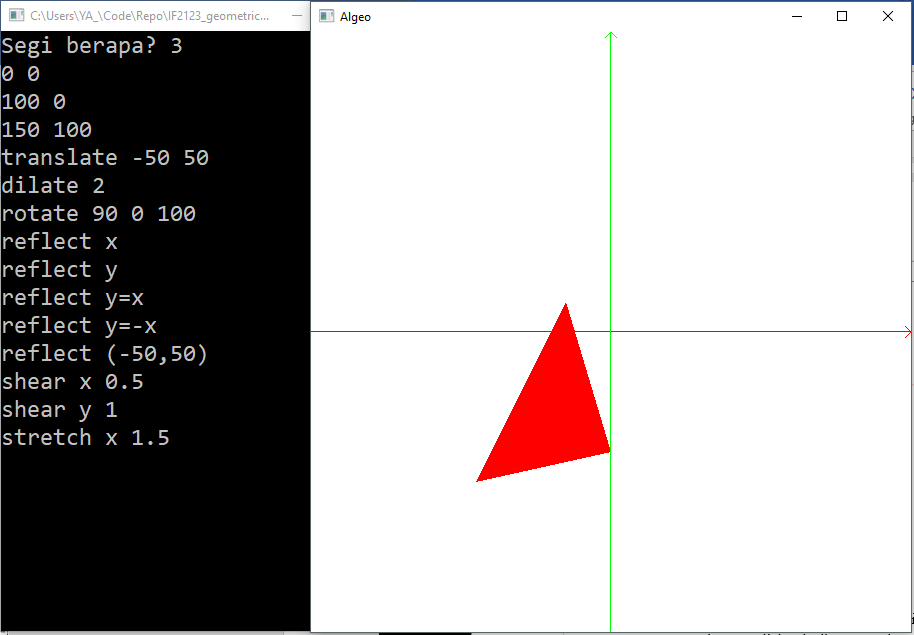


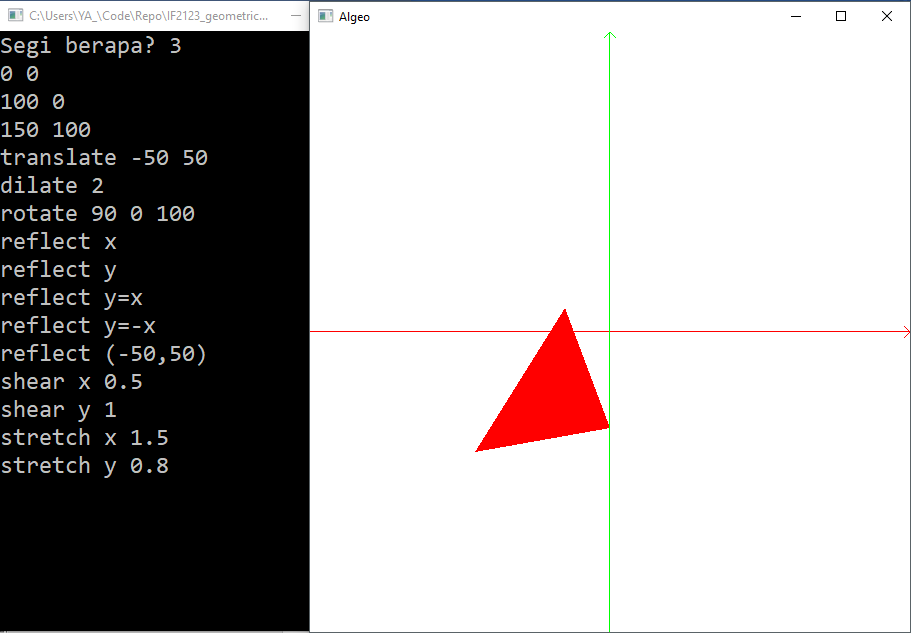






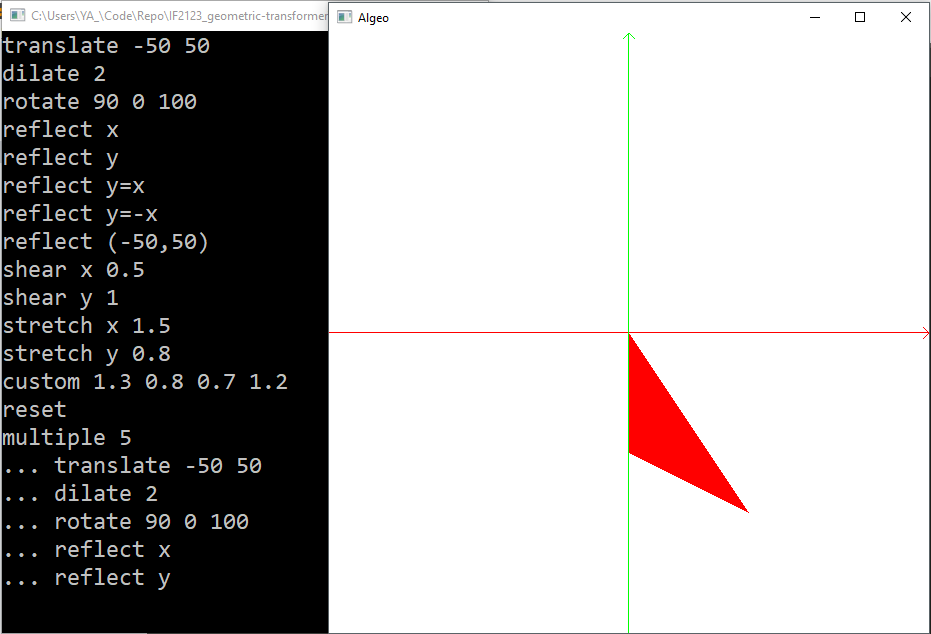












BAB V

KESIMPULAN

Transformasi pada bidang 2D dapat disimulasikan pada komputer melalui OpenGL API.

Operasi translasi dapat dilakukan dengan menambahkan vektor titik dengan vektor perpindahannya. Operasi dilatasi, rotasi, refleksi, *shear*, *stretch*, dapat dilakukan dengan menyiapkan matriks transformasi yang sesuai dengan operasi, lalu melakukan perkalian matriks dengan representasi vektor dari titik yang ingin ditransformasikan.

Operasi pada poligon akan sama dengan titik, hanya saja dilakukan pada semua titik sudut yang mendeskripsikan poligon, sehingga didapat poligon bayangan setelah dioperasikan.

REFERENSI

[1] Anton, Howard, 1997, Linear Algebra, Tenth Edition pdf.

[2] Slide Kuliah IF2123, Aplikasi transformasi linier pada grafika komputer (*computer graphics*), oleh Rinaldi Munir.

[3] Gentle, James E., 2007, "Matrix Transformations and Factorizations".

[4] OpenGL Tutorial, <http://www.opengl-tutorial.org>.