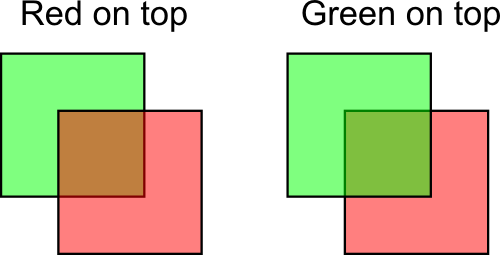


| **TITLE**: Write a program to Implement Transparency |
| --- |

**AIM:**

Write a program in OpenGL to Implement Transparency

Sample example



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

Understand the computer Input & interaction, Curves and Computer Animation

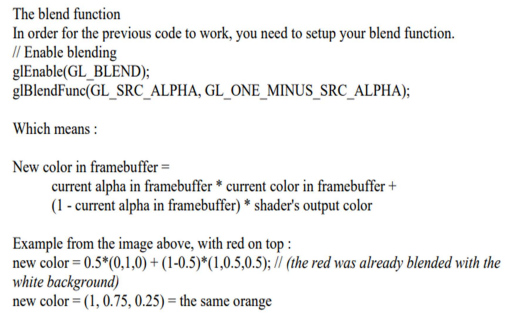
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-10-transparency/

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Algorithm/ Pseudocode for each process:**

****

**Implementation details:**

#include <GL/gl.h>

#include <GL/glu.h>

#include <GL/glut.h>

#include <stdlib.h>

static int leftFirst = 1; // Use 1 (true) instead of GL\_TRUE

static void init(void) {

glEnable(GL\_BLEND);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

glShadeModel(GL\_FLAT);

glClearColor(0.0, 0.0, 0.0, 0.0);

}

static void drawLeftTriangle(void) {

glBegin(GL\_TRIANGLES);

glColor4f(0.0, 1.0, 0.0, 0.5);

glVertex3f(0.1, 0.9, 0.0);

glVertex3f(0.1, 0.1, 0.0);

glVertex3f(0.7, 0.5, 0.0);

glEnd();

}

static void drawRightTriangle(void) {

glBegin(GL\_TRIANGLES);

glColor4f(1.0, 0.0, 0.0, 0.5);

glVertex3f(0.9, 0.9, 0.0);

glVertex3f(0.3, 0.5, 0.0);

glVertex3f(0.9, 0.1, 0.0);

glEnd();

}

void display(void) {

glClear(GL\_COLOR\_BUFFER\_BIT);

if (leftFirst) {

drawLeftTriangle();

drawRightTriangle();

} else {

drawRightTriangle();

drawLeftTriangle();

}

glFlush();

}

void reshape(int w, int h) {

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (w <= h) {

gluOrtho2D(0.0, 1.0, 0.0, 1.0 \* (GLfloat)h / (GLfloat)w);

} else {

gluOrtho2D(0.0, 1.0 \* (GLfloat)w / (GLfloat)h, 0.0, 1.0);

}

}

void keyboard(unsigned char key, int x, int y) {

switch (key) {

case 't':

case 'T':

leftFirst = !leftFirst;

glutPostRedisplay();

break;

case 27: // Escape key

exit(0);

break;

default:

break;

}

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutCreateWindow("Transparency");

init();

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

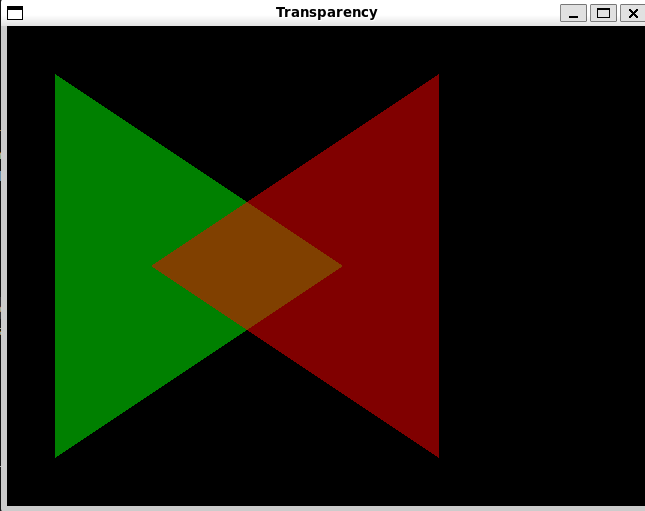
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Output(s) (Screen Shot):**



**Conclusion and discussion:**

**We have learnt to implement transparency in OpenGL.**

**Date: 29/9/24**

**Signature of faculty in-charge**

**Post lab Question**

**Improvise the code and take user input for transparency percentage**

**(Make it interactive with user )**

**#include <GL/glu.h>**

**#include <GL/glut.h>**

**#include <stdlib.h>**

**#include <stdio.h>**

**static int leftFirst = 1; // Use 1 (true) instead of GL\_TRUE**

**static float transparency = 0.5f; // Default transparency**

**static void init(void) {**

**glEnable(GL\_BLEND);**

**glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);**

**glShadeModel(GL\_FLAT);**

**glClearColor(0.0, 0.0, 0.0, 0.0);**

**}**

**static void drawLeftTriangle(void) {**

**glBegin(GL\_TRIANGLES);**

**glColor4f(0.0, 1.0, 0.0, transparency); // Use user-defined transparency**

**glVertex3f(0.1, 0.9, 0.0);**

**glVertex3f(0.1, 0.1, 0.0);**

**glVertex3f(0.7, 0.5, 0.0);**

**glEnd();**

**}**

**static void drawRightTriangle(void) {**

**glBegin(GL\_TRIANGLES);**

**glColor4f(1.0, 0.0, 0.0, transparency); // Use user-defined transparency**

**glVertex3f(0.9, 0.9, 0.0);**

**glVertex3f(0.3, 0.5, 0.0);**

**glVertex3f(0.9, 0.1, 0.0);**

**glEnd();**

**}**

**void display(void) {**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**if (leftFirst) {**

**drawLeftTriangle();**

**drawRightTriangle();**

**} else {**

**drawRightTriangle();**

**drawLeftTriangle();**

**}**

**glutSwapBuffers(); // Use double buffering**

**}**

**void reshape(int w, int h) {**

**glViewport(0, 0, (GLsizei)w, (GLsizei)h);**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**if (w <= h) {**

**gluOrtho2D(0.0, 1.0, 0.0, 1.0 \* (GLfloat)h / (GLfloat)w);**

**} else {**

**gluOrtho2D(0.0, 1.0 \* (GLfloat)w / (GLfloat)h, 0.0, 1.0);**

**}**

**}**

**void keyboard(unsigned char key, int x, int y) {**

**printf("Key pressed: %c\n", key); // Debug output**

**switch (key) {**

**case 't':**

**case 'T':**

**leftFirst = !leftFirst;**

**glutPostRedisplay();**

**break;**

**case '1':**

**transparency = 0.1f; // Set transparency to 10%**

**glutPostRedisplay();**

**break;**

**case '2':**

**transparency = 0.5f; // Set transparency to 50%**

**glutPostRedisplay();**

**break;**

**case '3':**

**transparency = 0.9f; // Set transparency to 90%**

**glutPostRedisplay();**

**break;**

**case 27: // Escape key**

**exit(0);**

**break;**

**default:**

**break;**

**}**

**}**

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

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

**glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB); // Use double buffering**

**glutInitWindowSize(640, 480);**

**glutCreateWindow("Transparency");**

**init();**

**glutReshapeFunc(reshape);**

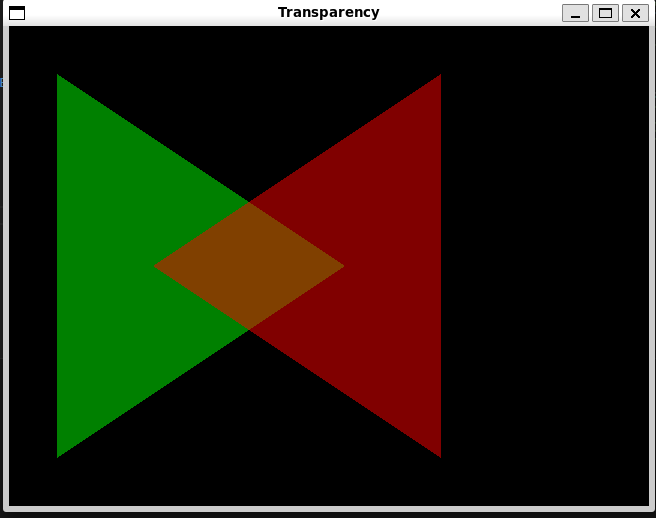
**glutKeyboardFunc(keyboard);**

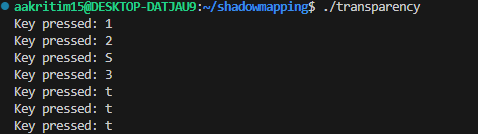
**glutDisplayFunc(display);**

**glutMainLoop();**

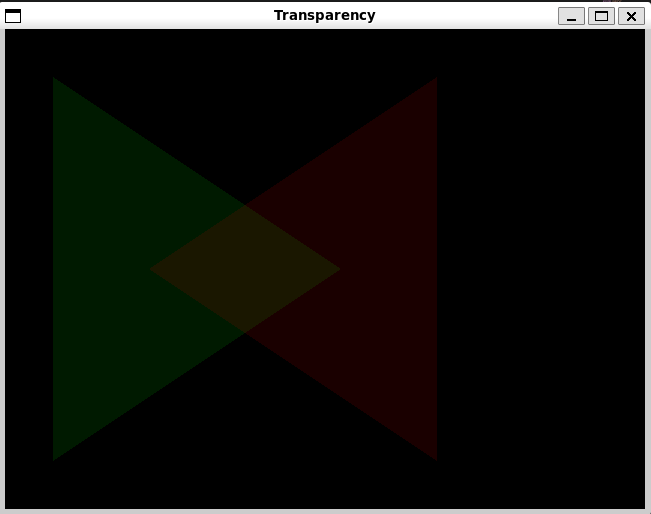
**return 0;**

**}**

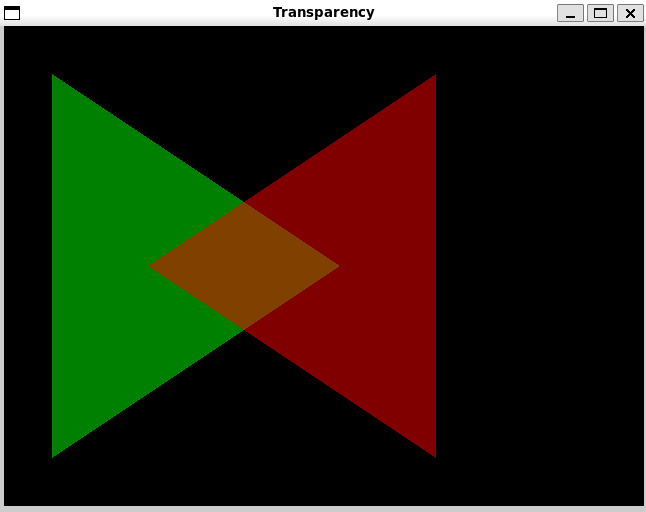
****

****

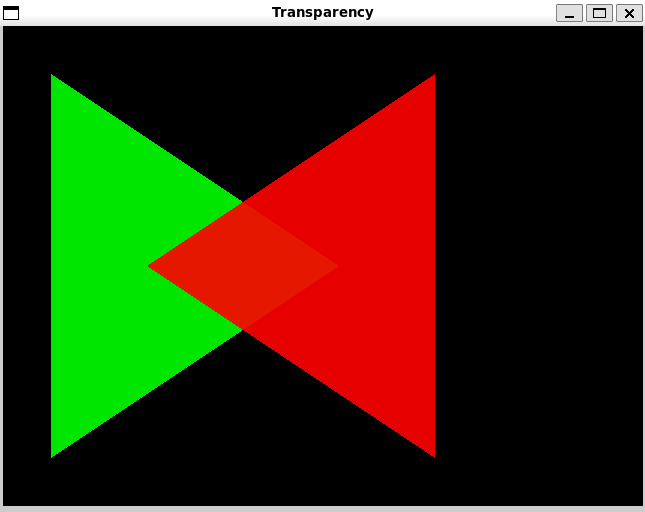
**transparency for 1**

****

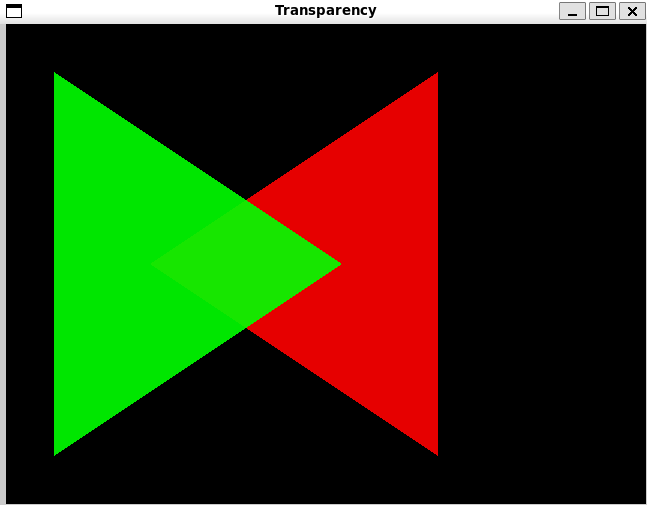
**transparency for 2**

****

**transparency for 3**

****

**transparency for t / T**

****