1- point

```
#include <glut.h>
void display(){
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0,1,1);

	glEnable(GL_POINT_SMOOTH);
	glPointSize(15);
	glBegin(GL_POINTS);
	glVertex2f(30,30);

	glEnd();
	glFlush();
}
void main (){
	glutInitWindowPosition(100,100);
	glutInitWindowSize(600,600);
	glutCreateWindow("Point");
	gluOrtho2D(0,200,0,200);
	glutDisplayFunc(display);
	glutMainLoop();
}
```

2- random Point

```
#include<stdlib.h>
#include<ctime>
#include <glut.h>
void display(){
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0,1,1);

	srand(time(NULL));
	int x=(rand()%200)+1;
	int y=(rand()%200)+1;

	glEnable(GL_POINT_SMOOTH);
	glPointSize(15);
	glBegin(GL_POINTS);
	glVertex2f(x,y);

	glEnd();
	glFlush();}
```

3- Triangle

```
void display(){

     glClearColor(1,1,1,1);
     glClear(GL_COLOR_BUFFER_BIT);
     glColor3f(0,1,1);
     glLineWidth(3);
     glBegin(GL_LINE_LOOP);

     glVertex2f(30,30);
     glVertex2f(130,30);
     glVertex2f(30,130);

     glEnd();
     glFlush();
}
```

4- Square

```
void display(){

     glClearColor(1,1,1,1);
     glClear(GL_COLOR_BUFFER_BIT);
     glColor3f(0,1,1);
     glLineWidth(3);
     glBegin(GL_LINE_LOOP);

     glVertex2f(30,30);
     glVertex2f(130,30);
     glVertex2f(130,130);
     glVertex2f(30,130);

     glEnd();
     glFlush();
}
```

5- 4 Square With Points

```
void display(){
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0,1,1);
//--------------------------<points>
	glEnable(GL_POINT_SMOOTH); // to make points a circular
	glPointSize(15);
	glBegin(GL_POINTS);
	glVertex2f(30,30);
	glVertex2f(130,30);
	glVertex2f(130,130);
	glVertex2f(30,130);
	glVertex2f(80,30);
	glVertex2f(130,80);
	glVertex2f(80,130);
	glVertex2f(30,80);
	glVertex2f(55,55);
	glVertex2f(105,55);
	glVertex2f(105,105);
	glVertex2f(55,105);
	glVertex2f(80,55);
	glVertex2f(105,80);
	glVertex2f(80,105);
	glVertex2f(55,80);
	glEnd();
//------------------------------
	glLineWidth(3);
//------------------------------<<1>>
	glBegin(GL_LINE_LOOP);
	glVertex2f(30,30);
	glVertex2f(130,30);
	glVertex2f(130,130);
	glVertex2f(30,130);
	glEnd();
//------------------------------<<2>>
	glBegin(GL_LINE_LOOP);
	glVertex2f(80,30);
	glVertex2f(130,80);
	glVertex2f(80,130);
	glVertex2f(30,80);
	glEnd();
//------------------------------<<3>>
	glBegin(GL_LINE_LOOP);
	glVertex2f(55,55);
	glVertex2f(105,55);
	glVertex2f(105,105);
	glVertex2f(55,105);
	glEnd();
//------------------------------<<4>>
	glBegin(GL_LINE_LOOP);
	glVertex2f(80,55);
	glVertex2f(105,80);
	glVertex2f(80,105);
	glVertex2f(55,80);
	glEnd();
//------------------------------END
	glFlush();}
```

6- Circle

```
#include<math.h>
void display()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
	glLineWidth(3);
	glBegin(GL_LINE_STRIP);
	glColor3f (0, 1, 1);

	float x=80,y=80,r=50;
	double a=22,b=7, pi=a/b;

	for(float i=0;i<=2*pi;i+=pi/360)
		glVertex2f(x+sin(i)*r,y+cos(i)*r);

	glEnd();
	glFlush();
}
```

7- Oval shape

```
#include<math.h>
void display()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
	glLineWidth(3);
	glBegin(GL_LINE_STRIP);
	glColor3f (0, 1, 1);

	float x=80,y=80,r=50;
	double a=22,b=7, pi=a/b;

	for(float i=0;i<=2*pi;i+=pi/360)
		glVertex2f(x+sin(i)*r,y+cos(i)*(1.4*r));

	glEnd();
	glFlush();
}
```

8- Circle 4 Color

```
#include<math.h>
void display()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
	glLineWidth(3);
	glBegin(GL_LINE_STRIP);

	float x=80,y=80,r=50;
	double pi=22.0/7.0;

	glColor3f (1, 0, 0);
	for(float i=0;i<=pi/2;i+=pi/500)
		glVertex2f(x+sin(i)*r,y+cos(i)*r);
	//=============================
	glColor3f (0, 1, 0);
	for(float i=pi/2;i<=pi;i+=pi/500)
		glVertex2f(x+sin(i)*r,y+cos(i)*r);
	//==============================
	glColor3f (0, 0, 1);
	for(float i=pi;i<=1.5*pi;i+=pi/500)
		glVertex2f(x+sin(i)*r,y+cos(i)*r);
	//==============================
	glColor3f (0.5, .5, .5);
	for(float i=1.5*pi;i<=2*pi;i+=pi/500)
		glVertex2f(x+sin(i)*r,y+cos(i)*r);
	//==============================
	glEnd();
	glFlush();
}
```

9- Transition 2 Points "Diagonal , X,Y-axis "

```
#include <glut.h>

float p1=0,p2=0,deltax=0.01,deltay=0.001,c=0;

void display(){

     glClearColor(c,c,c,0);
     glClear(GL_COLOR_BUFFER_BIT);
     glPointSize(10);
     glBegin(GL_POINTS);

     glColor3f(1,0,0);
     glVertex2f(p1,p1); /*To Move Points In Direction y-axis
                         use glVertex2f(Constant ,p1); */

     glColor3f(0,0,1);
     glVertex2f(p2,p2); /* To Move Points In Direction X-axis
                         use glVertex2f(P2 ,Constant); */
     glEnd();
     glFlush();

     p1=p1+deltax;
     if (p1>1.0||p1<0)
          deltax=-deltax;

     p2=p2+=deltay;
     if (p2>1.0||p2<0)
          deltay=-deltay;

     glutPostRedisplay();
}
void main(){

     glutInitWindowPosition(50,50);
     glutInitWindowSize(800,800);
     glutCreateWindow("Transition 2 points");
     glutDisplayFunc(display);
     glutMainLoop();
}
```

10- Transition 2 Points With Control Keys

```
#include <glut.h>

float p1=0,p2=0,deltax=0.001,deltay=0.0001,c=0;
bool stop=false;

void display(){
	glClearColor(c,c,c,0);
	glClear(GL_COLOR_BUFFER_BIT);
	glPointSize(10);
	glBegin(GL_POINTS);

	glColor3f(1,0,0);
	glVertex2f(p1,p1);

	glColor3f(0,0,1);
	glVertex2f(p2,p2);

	glEnd();
	glFlush();

	p1=p1+deltax;
	if (p1>1.0||p1<0)
		deltax=-deltax;

	p2=p2+=deltay;
	if (p2>1.0||p2<0)
		deltay=-deltay;

	if (stop==false)
		glutPostRedisplay();

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

	if (key==27) exit(0);            //Exit From Program
	else if (key=='c')  c=!c;        //Change Color
	else if (key=='a')  stop=!stop; //Stop & Resume
	else if (key=='r')  p1=0,p2=0;  //Reset
	glutPostRedisplay();
}
void main(){
	glutInitWindowPosition(50,50);
	glutInitWindowSize(800,800);
	glutCreateWindow("Transition 2 Points");
	glutKeyboardFunc(keyboard);
	glutDisplayFunc(display);
	glutMainLoop();
}
```

11- Chess Board

```
#include <glut.h>
void display(){

	glClearColor(0,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);

	bool c=0;

	for (int y=50;y<=400;y+=50)
	{
		for (int x=50;x<=400;x+=50)
		{
			glBegin(GL_POLYGON);
			glColor3f(c,c,c);

			glVertex2f(50+x,50+y);
			glVertex2f(100+x,50+y);
			glVertex2f(100+x,100+y);
			glVertex2f(50+x,100+y);
			glEnd();
			c=!c;
		}
		c=!c;
	}
	glFlush();
}
void main(){

	glutInitWindowPosition(100,0);
	glutInitWindowSize(900,900);
	glutCreateWindow("Chess Board");
	gluOrtho2D(0,600,0,600);
	glutDisplayFunc(display);
	glutMainLoop();

}
```

12- Chess Board With Pointer

```
#include <glut.h>

float p1=125,p2=125;

void display(){

	glClearColor(0,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);

	bool c=0;

	for (int y=50;y<=400;y+=50)
	{
		for (int x=50;x<=400;x+=50)
		{
			glBegin(GL_POLYGON);
			glColor3f(c,c,c);

			glVertex2f(50+x,50+y);
			glVertex2f(100+x,50+y);
			glVertex2f(100+x,100+y);
			glVertex2f(50+x,100+y);
			glEnd();
			c=!c;
		}
		c=!c;
	}

	glPointSize(65);
	glBegin(GL_POINTS);

	glColor3f(1,0.5,0.9);
	glVertex2f(p1,p2);

	glEnd();
	glFlush();

	glutPostRedisplay();
}


void move_up(){
	if (p2<450)
		p2+=50;
}
```

```
void move_down(){
      if (p2>150)
            p2-=50;
}

void move_left(){
      if (p1>150)
            p1-=50;
}

void move_right(){
      if (p1<450)
            p1+=50;
}

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

      switch(key){

      case 27:exit(0); break;
      case GLUT_KEY_UP:move_up(); break;
      case GLUT_KEY_DOWN:move_down() ;break;
      case GLUT_KEY_LEFT:move_left() ;break;
      case GLUT_KEY_RIGHT:move_right() ;break;

      }
      glutPostRedisplay();

}

void main(){

      glutInitWindowPosition(100,0);
      glutInitWindowSize(900,900);
      glutCreateWindow("Chess Board With Pointer");
      gluOrtho2D(0,600,0,600);
      glutDisplayFunc(display);
      glutSpecialFunc(keyboard);
      glutMainLoop();

}
```

13- Cube & Teapot "3D" (Rotation)

```
#include<glut.h>

float angle=1.0; //Speed Of Rotation

void display()
{
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0,0.5,1);
	glLineWidth(1.5);

	glMatrixMode(GL_MODELVIEW);

	glRotatef(angle,0,1,0); //Rotate Around Y-axis
	glutWireCube(30);

	glColor3f(1,0,1);
	glutWireTeapot(20);

	glFlush();
	glutSwapBuffers();
}
void main()
{
	glutInitWindowPosition(50, 60);
	glutInitWindowSize(800, 800);
	glutInitDisplayMode(GLUT_RGB | GL_DOUBLE);
	glutCreateWindow("Rotate Cube");

	glutDisplayFunc(display);
	glutIdleFunc(display); //the same function of
glutPostRedisplay();

	glOrtho(-44.0,44,-44.0,44,-44.0,44); //not gluOrtho
	glRotatef(1,1,0,0);

	glutMainLoop();
}
```

14- Cube & Teapot "3D" (Rotation) With Special function

```
#include<glut.h>
float angle=1.0; //Speed Of Rotation
float ROX=0,ROY=1;
void display(){
      glClearColor(1,1,1,0);
      glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(0,0.5,1);
      glLineWidth(1.5);

      glMatrixMode(GL_MODELVIEW);

      glRotatef(angle,ROX,ROY,0); //Rotate Around Y-axis
      glutWireCube(30);

      glColor3f(1,0,1);
      glutWireTeapot(20);

      glFlush();
      glutSwapBuffers();
}
void specFunc(int key,int x,int y){
      switch (key){
            case GLUT_KEY_UP: angle++;break;
            case GLUT_KEY_DOWN:if (angle>0) angle--;break;

            case GLUT_KEY_RIGHT:ROX=0;ROY=1;break;
            case GLUT_KEY_LEFT:ROY=0;ROX=1;break;
      }
}
void main(){
      glutInitWindowPosition(50, 60);
      glutInitWindowSize(800, 800);
      glutInitDisplayMode(GLUT_RGB | GL_DOUBLE);
      glutCreateWindow("Rotate Cube");

      glutDisplayFunc(display);
      glutIdleFunc(display); //the same function of
glutPostRedisplay();
      glutSpecialFunc(specFunc);

      glOrtho(-44.0,44,-44.0,44,-44.0,44); //not gluOrtho
      glRotatef(1,1,0,0);
      glutMainLoop();
}
```

15- Teapot accelerate (Rotation) With Special func & Keyboard func

```
#include< stdlib.h> //remove this if not work
#include<glut.h>
#include<math.h>
#include<stdio.h>

bool RunMode=1;
float CurrentAngle=0.0f,AnimateStep=3.0f;

void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT);

	if (RunMode==1){

		CurrentAngle+=AnimateStep;
		if (CurrentAngle>360.0)
			CurrentAngle-=360.0;
	}

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	glTranslatef(0,0,0);
	glRotatef(CurrentAngle,0,0,1);
	glutWireTeapot(2);
	glFlush();
	glutSwapBuffers();

	if (RunMode==1)
		glutPostRedisplay();

}
void keybFunction(unsigned char key ,int x,int y){

	switch (key){
	case 'r':
		RunMode=!RunMode;
		if (RunMode==1)
			glutPostRedisplay();
		break;

	case 's':
		RunMode=1; display(); RunMode=0;break;
```

```
        case 27: exit(0);
        }

}


void specFunction(int key,int x,int y){

        switch (key){

        case GLUT_KEY_UP:if (AnimateStep<1.0e3)
                                        AnimateStep*=sqrt(2.0);
break;

        case GLUT_KEY_DOWN:if (AnimateStep>1.0e-6)
                                        AnimateStep /= 9.0;
break;
        }
}

void main()
{
        glutInitDisplayMode(GL_DOUBLE | GLUT_RGB
|GLUT_DEPTH);
        glutCreateWindow("Rotate Exponentially");
        glutKeyboardFunc(keybFunction);
        glutSpecialFunc(specFunction);

        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0,5,0,5,-5,5); //not gluOrtho


        glutDisplayFunc(display);
        glutMainLoop();
}
```