VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANA SANGAMA, BELGAUM - 590014

"Phonics Play"

Submitted By :

Astha Rani	**1SI17CS016**
Athmiya HB	**1SI17CS017**
Ayesha Sultana	**1SI17CS019**

Under the guidance of :
Prof. Nousheen Taj

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

(An Autonomous Institute, Affiliated to Visvesvaraya Technological University, Belgaum, Recognized by AICTE and Accredited by NBA, New Delhi)

2019-2020

TABLE OF CONTENTS

ABSTRACT

- Main aim of the Project is to implement Phonics Play in OpenGL.
- There has been a huge shift in the past few years in how we teach reading in schools. This is having a big impact and helping many children learn to read and spell. Phonics is recommended as the first strategy that children should be taught in helping them learn to read. It runs alongside other teaching methods such as Guided Reading and Shared Reading to help children develop all the other vital reading skills and hopefully give them a real love of learning.
- So what exactly is Phonics? Words are made up from small units of sound called phonemes. Phonics teaches children to be able to listen carefully and identify the phonemes that make up each word. This helps children to learn to read words and to spell words
- We have used input devices like mouse and key board to interact with program

Introduction to openGL

As a software interface for graphics hardware, OpenGL's main purpose is to render two- and three-dimensional objects into a frame buffer.

These objects are described as sequences of vertices or pixels.

OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer.

OpenGL Fundamentals

This section explains some of the concepts inherent in OpenGL.

Primitives and Commands

OpenGL draws primitives—points, line segments, or polygons—subject to several selectable modes.

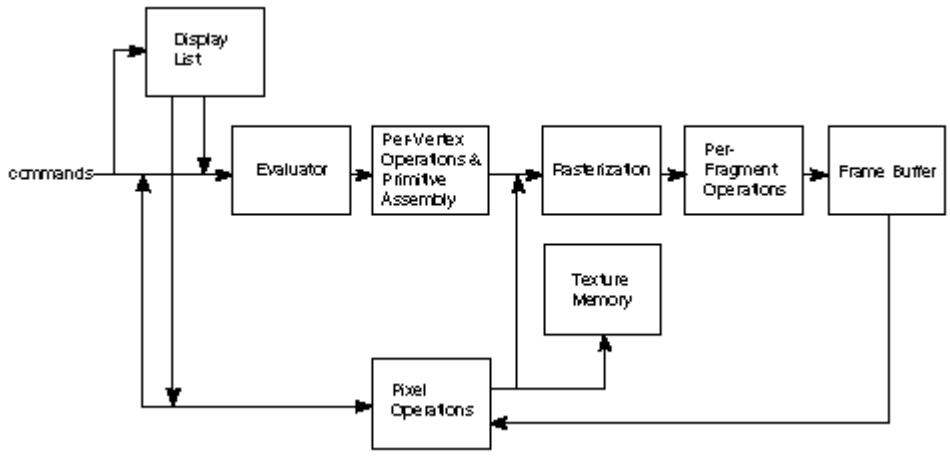
You can control modes independently of each other; that is, setting one mode doesn't affect whether other modes are set .Primitives are specified, modes are set, and other OpenGL operations are described by issuing commands in the form of function calls.

Primitives are defined by a group of one or more vertices. A vertex defines a point, an endpoint of a line, or a corner of a polygon where two edges meet. Data is associated with a vertex, and each vertex and its associated data are processed independently, in order, and in the same way. The type of clipping depends on which primitive the group of vertices represents.

Commands are always processed in the order in which they are received, although there may be an indeterminate delay before a command takes effect. This means that each primitive is drawn completely before any subsequent command takes effect. It also means that state-querying commands return data that's consistent with complete execution of all previously issued OpenGL commands.

Basic OpenGL Operation

The figure shown below gives an abstract, high-level block diagram of how OpenGL processes data. In the diagram, commands enter from the left and proceed through what can be thought of as a processing pipeline. Some commands specify geometric objects to be drawn, and others control how the objects are handled during the various processing stages.



As shown by the first block in the diagram, rather than having all commands proceed immediately through the pipeline, you can choose to accumulate some of them in a display list for processing at a later time.

Rasterization produces a series of frame buffer addresses and associated values using a two-dimensional description of a point, line segment, or polygon.

Each fragment so produced is fed into the last stage,

per-fragment operations, which performs the final operations on the data before it's stored as pixels in the frame buffer. These operations include conditional updates to the frame buffer based on incoming and previously stored z-value s (for z-buffering) and blending of incoming pixel colors with stored colors, as well as masking and other logical operations on pixel values.

All elements of OpenGL state, including the contents of the texture memory and even of the frame buffer, can be obtained by an OpenGL application.

IMPLEMENTATION

This program is implemented using various openGL functions which are shown below.

Various functions used in this program.

- glutInit() : interaction between the windowing system and OPENGL is initiated
- glutInitDisplayMode() : used when double buffering is required and depth information is required
- glutCreateWindow() : this opens the OPENGL window and displays the title at top of the window
- glutInitWindowSize() : specifies the size of the window
- glutInitWindowPosition() : specifies the position of the window in screen co-ordinates
- glutKeyboardFunc() : handles normal ascii symbols
- glutSpecialFunc() : handles special keyboard keys
- glutReshapeFunc() : sets up the callback function for reshaping the window
- glutIdleFunc() : this handles the processing of the background
- glutDisplayFunc() : this handles redrawing of the window
- glutMainLoop() : this starts the main loop, it never returns
- glViewport() : used to set up the viewport
- glVertex3fv() : used to set up the points or vertices in three dimensions
- glColor3fv() : used to render color to faces
- glFlush() : used to flush the pipeline
- glutPostRedisplay() : used to trigger an automatic redrawal of the object
- glMatrixMode() : used to set up the required mode of the matrix
- glLoadIdentity() : used to load or initialize to the identity matrix
- glTranslatef() : used to translate or move the rotation centre from one point to another in three dimensions
- glRotatef() : used to rotate an object through a specified rotation angle

```
#include<GL/gl.h>
#include<GL/glut.h>
#include<GL/glu.h>
#include<string.h>
#include<windows.h>
#include<mmsystem.h>
void MyTimerFunc(int value);

void apple()
{
   //text
   glPushMatrix();
   glColor3f(0, 0, 0);
   glTranslatef(9,0,0);
   glRotatef(0, 0.0, 0.0, 0);
   char message[]="apple";
   glRasterPos3f(-5, -4, 2);
   for (char* c = message; *c != '\0'; c++)
   {
      glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
   }
     glPopMatrix();
   //box
   glPushMatrix();
   glColor3f(1.0, 0.3, 0.3);
   glTranslated(-5.2, 0, 2);
   glutSolidCube(8);
   glPopMatrix();
   //letter A
   glColor3f(1,1,0.5);
   glBegin(GL_LINES);
   glVertex2f(-5.3,3.5);
   glVertex2f(-7.5,-3.5);
   glVertex2f(-5.3,3.5);
   glVertex2f(-3.2,-3.5);
   glVertex2f(-6.3,0);
   glVertex2f(-4.3,0);
   glEnd();

   //apple
   glPushMatrix();
   glColor3f(1.0, 0.1, 0.1);
   glTranslated(5.5, 1.0, -5);
   glScalef(2, 2, 2);
   glutSolidSphere(1.5, 40, 50);
   glPopMatrix();

   glPushMatrix();
   glColor3f(0.5, 0.1, 0.1);
   glTranslated(5.5, 2, -5);
```

```
    glScalef(3, 2, 2);
    glutSolidSphere(0.3, 40, 50);
    glPopMatrix();

    glPushMatrix();  //leaf
    glColor3f(0.5, 0.7, 0.1);
    glTranslated(6.5, 4, -5);
    glScalef(3.2, 2, 2);
    glutSolidSphere(0.3, 40, 50);
    glPopMatrix();

    glColor3f(0,0,0);  //stem
    glBegin(GL_LINES);
    glVertex2f(5.5,2.5);
    glVertex2f(5.5,5.0);
    glEnd();

    glutSwapBuffers();
}

void ball()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(9,0,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="ball";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
       glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter B
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-6,3.5);
    glVertex2f(-6,-3.5);
    glEnd();
    glPushMatrix();
    glColor3f(1,1,0.5);
    glTranslated(-6, 1.75, 2);
    gluPartialDisk(gluNewQuadric(),0,1.75,25,25,360,180);
```

```
    glPopMatrix();
    glPushMatrix();
    glColor3f(1,1,0.5);
    glTranslated(-6, -1.75, 2);
    gluPartialDisk(gluNewQuadric(),0,1.75,25,25,360,180);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1,0.3,0.3);
    glTranslated(-6, 1.75, 2);
    gluPartialDisk(gluNewQuadric(),0,1.6,25,25,360,180);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1,0.3,0.3);
    glTranslated(-6, -1.75, 2);
    gluPartialDisk(gluNewQuadric(),0,1.6,25,25,360,180);
    glPopMatrix();

    //ball
    glPushMatrix();
    glColor3f(0.9, 0.8, 0);
    glTranslated(5.5, 1.0, -5);
    glScalef(2, 2, 2);
    glutSolidSphere(1.5, 40, 50);
    glPopMatrix();

    glPushMatrix();
    glColor3f(0.9, 0.5, 0.4);
    glTranslated(5.5, 2, -5);
    glScalef(3, 2, 2);
    glutSolidSphere(0.3, 40, 50);
    glPopMatrix();

    glColor3f(1,1,1);
    glBegin(GL_LINES);
    glVertex2f(5.5,1.5);
    glVertex2f(5.5,-2);
    glVertex2f(5.5,2.5);
    glVertex2f(5.5,4.0);
    glVertex2f(2.7,2);
    glVertex2f(4.6,2);
    glVertex2f(6.4,2);
    glVertex2f(8.3,2);
    glEnd();

    glutSwapBuffers();
}

void cone()
{
    //text
```

```
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(8,0,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="cone";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
       glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter C
    glPushMatrix();
    glColor3f(1,1,0.5);
    glTranslated(-5, 0.15, 2);
    gluPartialDisk(gluNewQuadric(),0,3.0,25,25,45,-270);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1,0.3,0.3);
    glTranslated(-4, 0.15, 2);
    gluPartialDisk(gluNewQuadric(),0,2.2,25,25,45,-270);
    glPopMatrix();
    //cone
    glPushMatrix();
    glColor3f(0.2, 0.5, 0);
    glTranslated(4, -3, 0);
    glRotatef(270, 1.0 ,0.0, 0.0);
    glutWireCone(3.5, 7, 50, 50);
    glPopMatrix();

    glutSwapBuffers();
}

void dots()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(9.5,0,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="dots";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
```

```
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter D
    glPushMatrix();
    glColor3f(1,1,0.5);
    glTranslated(-6, 0.0, 2);
    gluPartialDisk(gluNewQuadric(),0,3.2,25,25,360,180);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1,0.3,0.3);
    glTranslated(-6, 0, 2);
    gluPartialDisk(gluNewQuadric(),0,3.0,25,25,360,180);
    glPopMatrix();
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-6,3.5);
    glVertex2f(-6,-3.5);
    glVertex2f(-5.95,3.5);
    glVertex2f(-5.95,-3.5);
    glVertex2f(-5.975,3.5);
    glVertex2f(-5.975,-3.5);
    glEnd();
    //dots
    glPushMatrix();
    glColor3f(0.5, 0.5, 0);
    glTranslated(5.5, 1.0, -5);
    glScalef(2, 2, 2);
    glutSolidSphere(0.3, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.5, 0.5, 0);
    glTranslated(4.5, -1.0, -5);
    glScalef(2, 2, 2);
    glutSolidSphere(0.3, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.5, 0.5, 0);
    glTranslated(6.5, -1.0, -5);
    glScalef(2, 2, 2);
    glutSolidSphere(0.3, 40, 50);
    glPopMatrix();
    glutSwapBuffers();
}
```

```
void egg()
{
   //text
   glPushMatrix();
   glColor3f(0, 0, 0);
   glTranslatef(9.5,0,0);
   glRotatef(0, 0.0, 0.0, 0);
   char message[]="egg";
   glRasterPos3f(-5, -4, 2);
   for (char* c = message; *c != '\0'; c++)
   {
      glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
   }
    glPopMatrix();
   //box
   glPushMatrix();
   glColor3f(1.0, 0.3, 0.3);
   glTranslated(-5, 0, 2);
   glutSolidCube(8);
   glPopMatrix();
   //letter E
   glColor3f(1,1,0.5);
   glBegin(GL_LINES);
   glVertex2f(-7.5,3.5);
   glVertex2f(-7.5,-3.5);
   glVertex2f(-7.3,3.5);
   glVertex2f(-7.3,-3.5);
   glVertex2f(-7.5,3.5);
   glVertex2f(-3,3.5);
   glVertex2f(-7.5,3.3);
   glVertex2f(-3,3.3);
   glVertex2f(-7.5,0.1);
   glVertex2f(-4.5,0.1);
   glVertex2f(-7.5,-0.1);
   glVertex2f(-4.5,-0.1);
   glVertex2f(-7.5,-3.5);
   glVertex2f(-3,-3.5);
   glVertex2f(-7.5,-3.3);
   glVertex2f(-3,-3.3);
   glEnd();

   //egg
   glPushMatrix();
   glColor3f(1, 1, 1);
   glTranslated(5.2, 1.0, -5);
   glScalef(2, 4, 2);
   glutSolidSphere(.7, 40, 50);
   glPopMatrix();
```

```
    glutSwapBuffers();
}

void fish()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(9.5,0,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="fish";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter F
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-7.5,3.5);
    glVertex2f(-7.5,-3.5);
    glVertex2f(-7.3,3.5);
    glVertex2f(-7.3,-3.5);
    glVertex2f(-7.5,3.5);
    glVertex2f(-3,3.5);
    glVertex2f(-7.5,3.3);
    glVertex2f(-3,3.3);
    glVertex2f(-7.5,0.1);
    glVertex2f(-4.5,0.1);
    glVertex2f(-7.5,-0.1);
    glVertex2f(-4.5,-0.1);
    glEnd();

    //fish
    glPushMatrix();
    glColor3f(0.1, 0.5, 0.5);
    glTranslated(4.5, 1.0, -5);
    glScalef(4, 2, 2);
    glutSolidSphere(.7, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.7, 0.2, 0.1);
    glTranslated(3.0, 1.0, -5);
```

```
    glScalef(3, 2, 2);
    glutSolidSphere(.4, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1, 1, 1);
    glTranslated(2.7, 1.0, -5);
    glScalef(2, 2, 2);
    glutSolidSphere(.1, 40, 50);
    glPopMatrix();
    glBegin(GL_TRIANGLES);
    glColor3f(0.5,0.2,0.1);
    glVertex2f(7.0,1);
    glVertex2f(9.0,3);
    glVertex2f(9.0,-1);
    glEnd();
    glutSwapBuffers();
}

void grass()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(8,0,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="grass";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter G
    glPushMatrix();//big
    glColor3f(1,1,0.5);
    glTranslated(-5, 0.15, 2);
    gluPartialDisk(gluNewQuadric(),0,3.0,25,25,45,-270);
    glPopMatrix();
    glPushMatrix();//small
    glColor3f(1,0.3,0.3);
    glTranslated(-4, 0.15, 2);
    gluPartialDisk(gluNewQuadric(),0,2.2,25,25,45,-270);
    glPopMatrix();
    glColor3f(1,1,0.5);
```

```
    glBegin(GL_LINES);
    glVertex2f(-2.95,0);
    glVertex2f(-2.95,-2.1);
    glVertex2f(-2.9,0);
    glVertex2f(-2.9,-2.1);
    glVertex2f(-2.95,0.15);
    glVertex2f(-4.5,0.15);
    glVertex2f(-2.95,0.1);
    glVertex2f(-4.5,0.1);
    glEnd();
    //grass
    glBegin(GL_POLYGON);
    glColor3f(0.5,0.9,0);
    glVertex2f(1.5,-2.0);
    glVertex2f(2.5,3.5);
    glVertex2f(3.5,0.5);
    glVertex2f(4.5,4.0);
    glVertex2f(5.5,0.75);
    glVertex2f(6.5,3.3);
    glVertex2f(7.5,.3);
    glVertex2f(8,-2.0);
    glEnd();
    glBegin(GL_POINTS);
    glColor3f(0.1,0.1,0.1);
    glVertex2f(2.7,-0.5);
    glVertex2f(3.3,-0.5);
    glVertex2f(3.9,-0.5);
    glVertex2f(4.2,-0.5);
    glVertex2f(4.8,-0.5);
    glVertex2f(5.4,-0.5);
    glVertex2f(6.0,-0.5);
    glEnd();

    glutSwapBuffers();
}

void house()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(9.5,-0.5,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="house";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
      glPopMatrix();
```

```
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter H
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-7.5,3.5);
    glVertex2f(-7.5,-3.5);
    glVertex2f(-7.3,3.5);
    glVertex2f(-7.3,-3.5);
    glVertex2f(-7.5,0.1);
    glVertex2f(-3,0.1);
    glVertex2f(-7.5,-0.1);
    glVertex2f(-3,-0.1);
    glVertex2f(-3,3.5);
    glVertex2f(-3,-3.5);
    glVertex2f(-3.2,3.5);
    glVertex2f(-3.2,-3.5);
    glEnd();

    //house
    glPushMatrix();
    glColor3d(0.9, 0.8, 0.1);
    glTranslated(5.6, -1, 2);
    glutSolidCube(4.2);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.5, 0.3, 0.0);
    glTranslated(5.6, 1, 2);
    glRotatef(270, 1.0, 0.0, 0.0);
    glutSolidCone(3.3, 3.1, 50, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(.2, 0.1, 0.1);
    glTranslated(5.6, -2.4, 1);
    glScaled(0.45, 0.8, 1);
    glutSolidCube(1.7);
    glPopMatrix();

    glutSwapBuffers();
}

void icecream()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
```

```
glTranslatef(8.8,-1.0,0);
glRotatef(0, 0.0, 0.0, 0);
char message[]="icecream";
glRasterPos3f(-5, -4, 2);
for (char* c = message; *c != '\0'; c++)
{
   glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
}
  glPopMatrix();
//box
glPushMatrix();
glColor3f(1.0, 0.3, 0.3);
glTranslated(-5, 0, 2);
glutSolidCube(8);
glPopMatrix();
//letter I
glColor3f(1,1,0.5);
glBegin(GL_LINES);
glVertex2f(-5,3.5);
glVertex2f(-5,-3.5);
glVertex2f(-5.2,3.5);
glVertex2f(-5.2,-3.5);
glVertex2f(-7.5,3.5);
glVertex2f(-3,3.5);
glVertex2f(-7.5,3.3);
glVertex2f(-3,3.3);
glVertex2f(-7.5,-3.5);
glVertex2f(-3,-3.5);
glVertex2f(-7.5,-3.3);
glVertex2f(-3,-3.3);
glEnd();

//icecream
glPushMatrix();
glColor3f(1, 0.4, 0.5);
glTranslated(5.5, 2.5, -5);
glScalef(2, 2, 2);
glutSolidSphere(1.3, 40, 50);
glPopMatrix();
glPushMatrix();
glColor3f(1, 1, 1);
glTranslated(5.5, 4, -5);
glScalef(2, 2, 2);
glutSolidSphere(0.8, 40, 50);
glPopMatrix();
glPushMatrix();
glColor3f(0.4, 0.2, 0.0);
glTranslated(5.5, 1.2, 0);
glRotatef(90, 1.0 ,0.0, 0.0);
glutSolidCone(3.0, 5, 50, 50);
```

```
    glPopMatrix();

    glutSwapBuffers();
}

void jug()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(9.7,-1.0,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="jug";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter J
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-5,3.5);
    glVertex2f(-5,-3.5);
    glVertex2f(-5.2,3.5);
    glVertex2f(-5.2,-3.5);
    glVertex2f(-7.5,3.5);
    glVertex2f(-3,3.5);
    glVertex2f(-7.5,3.3);
    glVertex2f(-3,3.3);
    glVertex2f(-5,-3.6);
    glVertex2f(-7.3,0);
    glVertex2f(-5.2,-3.4);
    glVertex2f(-7.5,0);
    glEnd();

    //jug
    glPushMatrix();
    glColor3f(0.1, 0, 0.3);
    glTranslated(5.5, 0, -5);
    glScaled(0.45, 1, 1);
    glutSolidCube(6.5);
    glPopMatrix();
```

```
    glPushMatrix();
    glColor3f(0.4, 0.2, 0.0);
    glTranslated(7, 1.2, 0);
    glRotatef(90, 0.0 ,1.0, 0.0);
    glutSolidCone(2.0, 2, 50, 50);
    glPopMatrix();

    glPushMatrix();
    glColor3f(0.6, 0.6, 1);
    glTranslated(7, 1.2, 0);
    glRotatef(90, 0.0 ,1.0, 0.0);
    glutSolidCone(1.0, 1, 50, 50);
    glPopMatrix();

    glPushMatrix();
    glColor3f(0.1, 0, 0.3);
    glTranslated(4.1, 2.2, 0);
    glRotatef(-90, 0.0 ,1.0, 0.0);
    glutSolidCone(1.0, 1, 50, 50);
    glPopMatrix();

    glutSwapBuffers();
}

void kite()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(9.75,-0.5,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="kite";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5.5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter K
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-7.5,3.5);
    glVertex2f(-7.5,-3.5);
    glVertex2f(-7.35,3.5);
```

```
    glVertex2f(-7.35,-3.5);
    glEnd();
    glBegin(GL_LINE_STRIP);
    glVertex2f(-4.5,3.5);
    glVertex2f(-7.5,0);
    glVertex2f(-4.5,-3.5);
    glEnd();
    glBegin(GL_LINE_STRIP);
    glVertex2f(-4.3,3.5);
    glVertex2f(-7.3,0);
    glVertex2f(-4.3,-3.5);
    glEnd();

    //kite
    glColor3d(0.1, 1, 0.5);
    glBegin(GL_POLYGON);
    glVertex2f(5.4,4);
    glVertex2f(2.4,1);
    glVertex2f(5.4,-2);
    glVertex2f(8.4,1);
    glEnd();
    glPushMatrix();
    glColor3f(1, 0, 1);
    glTranslated(5.365, -3.0, 2);
    glRotatef(-90, 1.0, 0.0, 0.0);
    glutSolidCone(1, 1, 50, 50);
    glPopMatrix();
    glBegin(GL_LINES);
    glColor3f(1, 0, 1);
    glVertex2f(2.4,1);
    glVertex2f(8.4,1);
    glVertex2f(5.4,4);
    glVertex2f(5.4,-2);
    glEnd();

    glutSwapBuffers();
}

void leaf()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(9.5,-0.5,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="leaf";
    glRasterPos3f(-5, -3, 2);
    for (char* c = message; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
```

```
    }
      glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5.5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter L
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-6.5,3.5);
    glVertex2f(-6.5,-3.5);
    glVertex2f(-6.35,3.5);
    glVertex2f(-6.35,-3.5);
    glEnd();
    glBegin(GL_LINES);
    glVertex2f(-6.5,-3.5);
    glVertex2f(-3.5,-3.5);
    glVertex2f(-6.5,-3.3);
    glVertex2f(-3.5,-3.3);
    glEnd();

    //leaf
    glPushMatrix();
    glColor3f(0, 0.7, 0.3);
    glTranslated(5.5, 1.0, -5);
    glScalef(4, 2, 2);
    glutSolidSphere(.7, 40, 50);
    glPopMatrix();
    glBegin(GL_LINES);
    glColor3f(0.5, 0, 0.5);
    glVertex2f(1.4,1);
    glVertex2f(8.0,1);
    glEnd();

    glutSwapBuffers();
}

void moon()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(8.2,-0.5,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="moon";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
```

```
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-4.75, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter M
    glColor3f(1,1,0.5);
    glBegin(GL_LINE_STRIP);
    glVertex2f(-6.95,-3);
    glVertex2f(-6.95,3);
    glVertex2f(-4.9,0);
    glVertex2f(-2.7,3);
    glVertex2f(-2.7,-3);
    glEnd();
    //moon
    glPushMatrix();//big
    glColor3f(0.3,0.3,0.2);
    glTranslated(5, 0.15, 2);
    gluPartialDisk(gluNewQuadric(),0,3.0,25,25,45,-270);
    glPopMatrix();
    glPushMatrix();//small
    glColor3f(0.6,0.6,1);
    glTranslated(6, 0.15, 2);
    gluPartialDisk(gluNewQuadric(),0,2.25,25,25,45,-270);
    glPopMatrix();

    glutSwapBuffers();
}

void nest()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(8.8,-0.5,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="nest";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
```

```
    glTranslated(-4.75, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter N
    glColor3f(1,1,0.5);
    glBegin(GL_LINE_STRIP);
    glVertex2f(-6.95,-3);
    glVertex2f(-6.95,3);
    glVertex2f(-2.7,-3);
    glVertex2f(-2.7,3);
    glEnd();
    //nest
    glPushMatrix();
    glColor3f(0.5, 0.3, 0.1);
    glTranslated(4.5, -2.0, -5);
    glScalef(4, 2, 2);
    glutSolidSphere(.7, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1, 1, 1);
    glTranslated(3.0, -1.0, -5);
    glScalef(2, 4, 2);
    glutSolidSphere(.2, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1, 1, 1);
    glTranslated(4.75, -1.5, -5);
    glScalef(2, 4, 2);
    glutSolidSphere(.2, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1, 1, 1);
    glTranslated(6.5, -1.0, -5);
    glScalef(2, 4, 2);
    glutSolidSphere(.2, 40, 50);
    glPopMatrix();
    glutSwapBuffers();
}

void orange()
{
     //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(8,-1,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="orange";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
```

```
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter O
    glPushMatrix();
    glColor3f(1,1,0.5);
    glTranslated(-5, 0, -5);
    glScalef(2, 2.3, 2);
    glutSolidSphere(1.5, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1,0.3,0.3);
    glTranslated(-5, 0, -5);
    glScalef(2, 2.3, 2);
    glutSolidSphere(1.3, 40, 50);
    glPopMatrix();
    //orange
    glBegin(GL_LINES);
    glColor3f(0,0,0.2);
    glVertex2f(4,3);
    glVertex2f(4,5);
    glEnd();
    glPushMatrix();//orange
    glColor3f(1, 0.3, 0);
    glTranslated(4, 0, 0);
    glRotatef(0, 1.0 ,0.0, 0.0);
    glutSolidCone(3.5, 7, 50, 50);
    glPopMatrix();
    glPushMatrix();  //leaf
    glColor3f(0.5, 0.7, 0.1);
    glTranslated(5, 4, -5);
    glScalef(3.2, 2, 2);
    glutSolidSphere(0.3, 40, 50);
    glPopMatrix();

    glutSwapBuffers();
}

void pot()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(9.3,-1,0);
```

```
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="pot";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
       glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
      glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter P
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-6,3.5);
    glVertex2f(-6,-3.5);
    glEnd();
    glPushMatrix();
    glColor3f(1,1,0.5);
    glTranslated(-6, 1.75, 2);
    gluPartialDisk(gluNewQuadric(),0,1.75,25,25,360,180);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1,0.3,0.3);
    glTranslated(-6, 1.75, 2);
    gluPartialDisk(gluNewQuadric(),0,1.6,25,25,360,180);
    glPopMatrix();
    //pot
    glPushMatrix();
    glColor3f(0.7, 0.4, 0.1);
    glTranslated(5, 2, 2);
    glutSolidCube(2.5);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.7, 0.4, 0.1);
    glTranslated(5.0, -0.2, -5);
    glScalef(2, 2, 2);
    glutSolidSphere(1.5, 40, 50);
    glPopMatrix();

    glutSwapBuffers();
}

void quilt()
{
     //text
    glPushMatrix();
```

```
glColor3f(0, 0, 0);
glTranslatef(8.5,-1,0);
glRotatef(0, 0.0, 0.0, 0);
char message[]="quilt";
glRasterPos3f(-5, -4, 2);
for (char* c = message; *c != '\0'; c++)
{
   glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
}
 glPopMatrix();
//box
glPushMatrix();
glColor3f(1.0, 0.3, 0.3);
glTranslated(-5, 0, 2);
glutSolidCube(8);
glPopMatrix();
//letter Q
glPushMatrix();
glColor3f(1,1,0.5);
glTranslated(-5, 0, -5);
glScalef(2, 2.3, 2);
glutSolidSphere(1.5, 40, 50);
glPopMatrix();
glPushMatrix();
glColor3f(1,0.3,0.3);
glTranslated(-5, 0, -5);
glScalef(2, 2.3, 2);
glutSolidSphere(1.3, 40, 50);
glPopMatrix();
glBegin(GL_LINES);
glColor3f(1,1,0.5);
glVertex2f(-4,-1);
glVertex2f(-2,-3);
glVertex2f(-3.9,-1);
glVertex2f(-1.9,-3);
glVertex2f(-3.95,-1);
glVertex2f(-1.95,-3);

glEnd();
//quilt
glBegin(GL_POLYGON);
glColor3f(0.9,0.1,0.2);
glVertex2f(2,3);
glVertex2f(2,-3);
glVertex2f(7,-3);
glVertex2f(7,3);
glEnd();
glBegin(GL_LINES);
glColor3f(1,1,1);
glVertex2f(3,3);
```

```
    glVertex2f(3,-3);
    glVertex2f(4,3);
    glVertex2f(4,-3);
    glVertex2f(5,3);
    glVertex2f(5,-3);
    glVertex2f(6,3);
    glVertex2f(6,-3);
    glVertex2f(2,3);
    glVertex2f(7,-3);
    glVertex2f(7,3);
    glVertex2f(2,-3);
    glVertex2f(2,0);
    glVertex2f(7,0);
    glEnd();
    glutSwapBuffers();
}

void rainbow()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(8.5,1,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="rainbow";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
      glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter R
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-6,3.5);
    glVertex2f(-6,-3.5);
    glVertex2f(-6,0);
    glVertex2f(-3.5,-3.5);
    glVertex2f(-5.95,-0);
    glVertex2f(-3.55,-3.5);
    glEnd();
    glPushMatrix();
    glColor3f(1,1,0.5);
    glTranslated(-6, 1.75, 2);
```

```
gluPartialDisk(gluNewQuadric(),0,1.75,25,25,360,180);
glPopMatrix();
glPushMatrix();
glColor3f(1,0.3,0.3);
glTranslated(-6, 1.75, 2);
gluPartialDisk(gluNewQuadric(),0,1.6,25,25,360,180);
glPopMatrix();

//rainbow
glPushMatrix();
glColor3f(1, 0, 0);
glTranslated(5,-1, 2);
gluPartialDisk(gluNewQuadric(),0,3.3,25,25,270,180);
glPopMatrix();

glPushMatrix();
glColor3f(0.9,0.3,0);
glTranslated(5, -1, 2);
gluPartialDisk(gluNewQuadric(),0,3.1,25,25,270,180);
glPopMatrix();

glPushMatrix();
glColor3f(0.8,0.7,0);
glTranslated(5,-1, 2);
gluPartialDisk(gluNewQuadric(),0,2.9,25,25,270,180);
glPopMatrix();

glPushMatrix();
glColor3f(0.3,0.8,0);
glTranslated(5,-1, 2);
gluPartialDisk(gluNewQuadric(),0,2.7,25,25,270,180);
glPopMatrix();

glPushMatrix();
glColor3f(0.3,0.6,1);
glTranslated(5,-1, 2);
gluPartialDisk(gluNewQuadric(),0,2.5,25,25,270,180);
glPopMatrix();

glPushMatrix();
glColor3f(0.3,0.1,1);
glTranslated(5,-1, 2);
gluPartialDisk(gluNewQuadric(),0,2.3,25,25,270,180);
glPopMatrix();

glPushMatrix();
glColor3f(0.3,0,0.4);
glTranslated(5,-1, 2);
gluPartialDisk(gluNewQuadric(),0,2.1,25,25,270,180);
glPopMatrix();
```

```
    glPushMatrix();
    glColor3f(0.6,0.6,1);
    glTranslated(5,-1, 2);
    gluPartialDisk(gluNewQuadric(),0,1.9,25,25,270,180);
    glPopMatrix();
    glutSwapBuffers();
}

void sun()
{
     //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(8.4,-1,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="sun";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
      glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter S
    glPushMatrix();
    glColor3f(1,1,0.5);
    glTranslated(-5, 1.5, -5);
    glScalef(3, 2, 2);
    glutSolidSphere(0.75, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1,0.3,0.3);
    glTranslated(-4.5, 1.5, -5);
    glScalef(3, 2, 2);
    glutSolidSphere(0.73, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1,1,0.5);
    glTranslated(-5, -1.5, -5);
    glScalef(3, 2, 2);
    glutSolidSphere(0.75, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(1,0.3,0.3);
```

```
    glTranslated(-5.5, -1.5, -5);
    glScalef(3, 2, 2);
    glutSolidSphere(0.73, 40, 50);
    glPopMatrix();
    glBegin(GL_LINES);
    glColor3f(1,1,0.5);
    glVertex2f(-3.5,3);
    glVertex2f(-5,3);
    glVertex2f(-5,-3);
    glVertex2f(-6.5,-3);
    glEnd();
    //sun
    glBegin(GL_LINES);
    glColor3f(1,1,0.1);
    glVertex2f(4,4);
    glVertex2f(4,-4);
    glVertex2f(0,0);
    glVertex2f(8,0);
    glEnd();
    glPushMatrix();//orange
    glColor3f(1, 1, 0.1);
    glTranslated(4, 0, 0);
    glRotatef(0, 1.0 ,0.0, 0.0);
    glutSolidCone(2.5, 7, 50, 50);
    glPopMatrix();

    glutSwapBuffers();
}

void train()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(8.5,-1.0,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="train";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
       glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter T
```

```
glColor3f(1,1,0.5);
glBegin(GL_LINES);
glVertex2f(-5,3.5);
glVertex2f(-5,-3.5);
glVertex2f(-5.2,3.5);
glVertex2f(-5.2,-3.5);
glVertex2f(-7.5,3.5);
glVertex2f(-3,3.5);
glVertex2f(-7.5,3.3);
glVertex2f(-3,3.3);

glEnd();

//train
glPushMatrix();//wheels
glColor3f(0, 0.1, 0.1);
glTranslated(7.3, -1.55, -5);
glScalef(2, 2, 2);
glutSolidSphere(.2, 40, 50);
glPopMatrix();
glPushMatrix();//wheels
glColor3f(0, 0.1, 0.1);
glTranslated(5.3, -1.55, -5);
glScalef(2, 2, 2);
glutSolidSphere(.2, 40, 50);
glPopMatrix();
glPushMatrix();//wheels
glColor3f(0, 0.1, 0.1);
glTranslated(3.85, -1.55, -5);
glScalef(2, 2, 2);
glutSolidSphere(.2, 40, 50);
glPopMatrix();
glPushMatrix();//wheels
glColor3f(0, 0.1, 0.1);
glTranslated(1.85, -1.55, -5);
glScalef(2, 2, 2);
glutSolidSphere(.2, 40, 50);
glPopMatrix();
glPushMatrix();//big box
glColor3f(0.9, 0.5, 0.3);
glTranslated(4.5, 0, -5);
glScaled(1, 0.4, 1);
glutSolidCube(6.5);
glPopMatrix();
glPushMatrix();//window
glColor3f(0, 0, 0);
glTranslated(4.5, 0.5, -5);
glScaled(1, 0.1, 1);
glutSolidCube(6.5);
glPopMatrix();
```

```
    glPushMatrix();//middle space
    glColor3f(0.6, 0.6, 1);
    glTranslated(4.5, 0, -5);
    glScaled(0.05, 1, 1);
    glutSolidCube(6.5);
    glPopMatrix();
    glColor3f(0.5,0,0.5);//join
    glBegin(GL_LINES);
    glVertex2f(4,0);
    glVertex2f(5,0);
    glVertex2f(1.0,-2);
    glVertex2f(8.0,-2);
    glVertex2f(1.0,-2.3);
    glVertex2f(8.0,-2.3);
    glEnd();


    glutSwapBuffers();
}

void umbrella()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(8.0,-0.5,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="umbrella";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
       glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter U
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-7.5,3.5);
    glVertex2f(-7.5,-3.5);
    glVertex2f(-7.3,3.5);
    glVertex2f(-7.3,-3.5);

    glVertex2f(-3,3.5);
    glVertex2f(-3,-3.5);
```

```
    glVertex2f(-3.2,3.5);
    glVertex2f(-3.2,-3.5);

    glVertex2f(-7.5,-3.5);
    glVertex2f(-3,-3.5);
    glVertex2f(-7.5,-3.3);
    glVertex2f(-3,-3.3);
    glEnd();

    //umbrella
    glPushMatrix();
    glColor3f(0,0,0.5);
    glTranslated(4.35, 5.55, 5);
    glScalef(2, 3, 2);
    glutSolidSphere(0.2, 40, 50);
    glPopMatrix();
    glColor3f(0,0,0.5);
    glBegin(GL_POLYGON);
    glVertex2f(4.25,2.0);
    glVertex2f(4.25,-3.5);
    glVertex2f(4.5,-3.5);
    glVertex2f(4.5,2.0);
    glEnd();
    glPushMatrix();
    glColor3f(0.9,0.3,0);
    glTranslated(4.5, 1.75, 2);
    gluPartialDisk(gluNewQuadric(),0,3.75,25,25,270,180);
    glPopMatrix();

    glColor3f(0,0,0.5);
    glBegin(GL_POLYGON);
    glVertex2f(4.25,-3.5);
    glVertex2f(3.25,-2.0);
    glVertex2f(4.5,-3.5);
    glVertex2f(3.5,-2.0);
    glEnd();
    glColor3f(0,0,0.5);
    glBegin(GL_LINES);
    glVertex2f(4.35,5.5);
    glVertex2f(4.35,-1.0);
    glVertex2f(4.35,5.5);
    glVertex2f(2.35,1.7);
    glVertex2f(4.35,5.5);
    glVertex2f(6.35,1.7);
    glEnd();
    glutSwapBuffers();
}

void van()
{
```

```
//text
glPushMatrix();
glColor3f(0, 0, 0);
glTranslatef(8.5,-1.0,0);
glRotatef(0, 0.0, 0.0, 0);
char message[]="van";
glRasterPos3f(-5, -4, 2);
for (char* c = message; *c != '\0'; c++)
{
   glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
}
  glPopMatrix();
//box
glPushMatrix();
glColor3f(1.0, 0.3, 0.3);
glTranslated(-5, 0, 2);
glutSolidCube(8);
glPopMatrix();
//letter V
glColor3f(1,1,0.5);
glBegin(GL_LINES);
glVertex2f(-7.25,3.5);
glVertex2f(-5.2,-3.5);
glVertex2f(-7.2,3.5);
glVertex2f(-5.15,-3.5);
glVertex2f(-5.2,-3.5);
glVertex2f(-3.25,3.5);
glVertex2f(-5.25,-3.5);
glVertex2f(-3.3,3.5);
glEnd();

//van
glPushMatrix();//window3
glColor3f(1, 0.1, 0.1);
glTranslated(7, 1.25, -5);
glScaled(1, 2, 1);
glutSolidCube(0.6);
glPopMatrix();
glPushMatrix();//wheels
glColor3f(0, 0.1, 0.1);
glTranslated(7.3, -1.55, -5);
glScalef(2, 2, 2);
glutSolidSphere(.2, 40, 50);
glPopMatrix();
glPushMatrix();//wheels
glColor3f(0, 0.1, 0.1);
glTranslated(5.3, -1.55, -5);
glScalef(2, 2, 2);
glutSolidSphere(.2, 40, 50);
glPopMatrix();
```

```
glPushMatrix();//wheels
glColor3f(0, 0.1, 0.1);
glTranslated(3.85, -1.55, -5);
glScalef(2, 2, 2);
glutSolidSphere(.2, 40, 50);
glPopMatrix();
glPushMatrix();//wheels
glColor3f(0, 0.1, 0.1);
glTranslated(1.85, -1.55, -5);
glScalef(2, 2, 2);
glutSolidSphere(.2, 40, 50);
glPopMatrix();
glPushMatrix();//big box
glColor3f(1, 1, 0.5);
glTranslated(4.5, 0, -5);
glScaled(1, 0.4, 1);
glutSolidCube(6.5);
glPopMatrix();
glPushMatrix();//small box
glColor3f(1, 1, 0.5);
glTranslated(4.5, -0.45, -5);
glScaled(1.2, 0.25, 1);
glutSolidCube(6.5);
glPopMatrix();
glPushMatrix();//window1
glColor3f(0, 0, 0.7);
glTranslated(4.0, 0.75, -5);
glScaled(1.3, 1, 1);
glutSolidCube(0.6);
glPopMatrix();
glPushMatrix();//window2
glColor3f(0, 0, 0.7);
glTranslated(2.5, 0.75, -5);
glScaled(1.3, 1, 1);
glutSolidCube(0.6);
glPopMatrix();
glPushMatrix();//window3
glColor3f(0, 0, 0.7);
glTranslated(5.5, 0.75, -5);
glScaled(1.3, 1, 1);
glutSolidCube(0.6);
glPopMatrix();
glPushMatrix();//window3
glColor3f(0, 0, 0.7);
glTranslated(7, 0.75, -5);
glScaled(1.3, 1, 1);
glutSolidCube(0.6);
glPopMatrix();

glutSwapBuffers();
```

```
}

void watch()
{
   //text
   glPushMatrix();
   glColor3f(0, 0, 0);
   glTranslatef(8.2,-0.5,0);
   glRotatef(0, 0.0, 0.0, 0);
   char message[]="watch";
   glRasterPos3f(-5, -4, 2);
   for (char* c = message; *c != '\0'; c++)
   {
      glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
   }
     glPopMatrix();
   //box
   glPushMatrix();
   glColor3f(1.0, 0.3, 0.3);
   glTranslated(-4.75, 0, 2);
   glutSolidCube(8);
   glPopMatrix();
   //letter W
   glColor3f(1,1,0.5);
   glBegin(GL_LINE_STRIP);
   glVertex2f(-6.95,3);
   glVertex2f(-6.95,-3);
   glVertex2f(-4.9,0);
   glVertex2f(-2.7,-3);
   glVertex2f(-2.7,3);
   glEnd();
   //watch
   glPushMatrix();//strip
   glColor3f(0.7, 0.2, 0);
   glTranslated(4.5, 1.45, -5);
   glScaled(0.25, 1.25, 1);
   glutSolidCube(6.5);
   glPopMatrix();
   glPushMatrix();//dial
   glColor3f(1, 1, 0.1);
   glTranslated(4.5, 1.25, 0);
   glRotatef(0, 1.0 ,0.0, 0.0);
   glutSolidCone(1.5, 7, 50, 50);
   glPopMatrix();
   glPushMatrix();//dial
   glColor3f(1, 1, 1);
   glTranslated(4.5, 1.25, 0);
   glRotatef(0, 1.0 ,0.0, 0.0);
   glutSolidCone(0.75, 7, 50, 50);
   glPopMatrix();
```

```
    glutSwapBuffers();
}

void xmastree()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(7.5,-1,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="x-mas tree";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5.2, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter X
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-7,3);
    glVertex2f(-3,-3);
    glVertex2f(-7.15,3);
    glVertex2f(-3.15,-3);

    glVertex2f(-3,3);
    glVertex2f(-7,-3);
    glVertex2f(-3.15,3);
    glVertex2f(-7.15,-3);
    glEnd();

    //xmastree
    // tree
    glPushMatrix();
    glColor3f(1, 0.5, 0);
    glTranslated(5, -2.7, 0);
    glRotatef(90, 1.0, 0.0, 0.0);
    glutSolidTorus(0.4, 0.7, 25, 25);
    glPopMatrix();

    glPushMatrix();
    glColor3f(1, 0.5, 0);
    glTranslated(5, -2.1, 0);
    glRotatef(90, 1.0, 0.0, 0.0);
```

```
    glutSolidTorus(0.4, 0.7, 25, 25);
    glPopMatrix();

    glPushMatrix();
    glColor3f(1, 0.5, 0);
    glTranslated(5, -1.5, 0);
    glRotatef(90, 1.0, 0.0, 0.0);
    glutSolidTorus(0.4, 0.7, 25, 25);
    glPopMatrix();

    glPushMatrix();
    glColor3f(1, 0.5, 0);
    glTranslated(5, -0.9, 0);
    glRotatef(90, 1.0, 0.0, 0.0);
    glutSolidTorus(0.4, 0.7, 25, 25);
    glPopMatrix();
    //tree leaves
    glPushMatrix();
    glColor3f(0.5, 1, 0.5);
    glTranslated(5, -1, 0);
    glRotatef(270, 1.0 ,0.0, 0.0);
    glutSolidCone(3, 5, 50, 50);
    glPopMatrix();

    glutSwapBuffers();
}

void yolk()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(9.3,-1,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="yolk";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
       glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
     glPopMatrix();
    //box
    glPushMatrix();
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5.2, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter Y
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
```

```
    glVertex2f(-7,3);
    glVertex2f(-5,0);
    glVertex2f(-7.15,3);
    glVertex2f(-5.15,0);

    glVertex2f(-5,0);
    glVertex2f(-3,3);
    glVertex2f(-5.15,0);
    glVertex2f(-3.15,3);

    glVertex2f(-5.15,0);
    glVertex2f(-5.15,-3);
    glVertex2f(-5,0);
    glVertex2f(-5,-3);
    glEnd();

    //yolk
    glPushMatrix();
    glColor3f(1, 1, 1);
    glTranslated(5.2, 1.0, -5);
    glScalef(2, 3, 2);
    glutSolidSphere(1, 40, 50);
    glPopMatrix();

    glPushMatrix();
    glColor3f(1, 1, 0);
    glTranslated(5.2, 0, -5);
    glScalef(2, 2, 2);
    glutSolidSphere(0.7, 40, 50);
    glPopMatrix();

    glutSwapBuffers();
}

void zero()
{
    //text
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(9.2,-1.0,0);
    glRotatef(0, 0.0, 0.0, 0);
    char message[]="zero";
    glRasterPos3f(-5, -4, 2);
    for (char* c = message; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
      glPopMatrix();
    //box
    glPushMatrix();
```

```
    glColor3f(1.0, 0.3, 0.3);
    glTranslated(-5, 0, 2);
    glutSolidCube(8);
    glPopMatrix();
    //letter Z
    glColor3f(1,1,0.5);
    glBegin(GL_LINES);
    glVertex2f(-3.2,3.5);
    glVertex2f(-7.5,-3.5);
    glVertex2f(-3.0,3.5);
    glVertex2f(-7.3,-3.5);
    glVertex2f(-7.5,3.5);
    glVertex2f(-3,3.5);
    glVertex2f(-7.5,3.3);
    glVertex2f(-3,3.3);
    glVertex2f(-7.5,-3.5);
    glVertex2f(-3,-3.5);
    glVertex2f(-7.5,-3.3);
    glVertex2f(-3,-3.3);
    glEnd();

    //icecream
    glPushMatrix();
    glColor3f(0.9, 0.5, 0.3);
    glTranslated(5.5, 1.75, -5);
    glScalef(2, 3, 2);
    glutSolidSphere(1.5, 40, 50);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.6, 0.6, 1);
    glTranslated(5.5,1.75, -5);
    glScalef(2, 3, 2);
    glutSolidSphere(0.8, 40, 50);
    glPopMatrix();


    glutSwapBuffers();
}

void backg()
{
    glBegin(GL_POLYGON);
    glColor3f(0.6,0.6,1);
    glVertex2f(-10,-10);
    glVertex2f(10,-10);
    glVertex2f(10,10);
    glVertex2f(-10,10);
    glEnd();
    glutSwapBuffers();
}
```

```
void display()
{
   glFlush();
}
   //apple();
   //ball();
   //cone();
   //dots();
   //egg();
   //fish();
   //grass();
   //house();
   //icecream();
   //jug();
   //kite();
   //leaf();
   //moon();
   //nest();
   //orange();
   //pot();
   //quilt();
   //rainbow();
   //sun();
   //train();
   //umbrella();
   //van();
   //watch();
   //xmastree();
   //yolk();
   //zero();

void init(void)
{
   glClearColor(0.6,0.6,1,1);
   glClear(GL_COLOR_BUFFER_BIT);
   glFlush();
}
void MyTimerFunc(int value)
{

   if(value==2)
   {
      glutDisplayFunc(apple);
      glutIdleFunc(apple);
      glutTimerFunc(3600,MyTimerFunc,3);
   }
   else if(value==3)
   {
      glutDisplayFunc(backg);
      glutIdleFunc(backg);
```

```
    glutTimerFunc(300,MyTimerFunc,4);
}
else if(value==4)
{
    glutDisplayFunc(ball);
    glutIdleFunc(ball);
    glutTimerFunc(3600,MyTimerFunc,5);
}
else if(value==5)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,6);
}
else if(value==6)
{
    glutDisplayFunc(cone);
    glutIdleFunc(cone);
    glutTimerFunc(3600,MyTimerFunc,7);
}
else if(value==7)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,8);
}
else if(value==8)
{
    glutDisplayFunc(dots);
    glutIdleFunc(dots);
    glutTimerFunc(3600,MyTimerFunc,9);
}
else if(value==9)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,10);
}
else if(value==10)
{
    glutDisplayFunc(egg);
    glutIdleFunc(egg);
    glutTimerFunc(3600,MyTimerFunc,11);
}
else if(value==11)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,12);
}
```

```
else if(value==12)
{
   glutDisplayFunc(fish);
   glutIdleFunc(fish);
   glutTimerFunc(3600,MyTimerFunc,13);
}
else if(value==13)
{
   glutDisplayFunc(backg);
   glutIdleFunc(backg);
   glutTimerFunc(300,MyTimerFunc,14);
}
else if(value==14)
{
   glutDisplayFunc(grass);
   glutIdleFunc(grass);
   glutTimerFunc(3600,MyTimerFunc,15);
}
else if(value==15)
{
   glutDisplayFunc(backg);
   glutIdleFunc(backg);
   glutTimerFunc(300,MyTimerFunc,16);
}
else if(value==16)
{
   glutDisplayFunc(house);
   glutIdleFunc(house);
   glutTimerFunc(3600,MyTimerFunc,17);
}
else if(value==17)
{
   glutDisplayFunc(backg);
   glutIdleFunc(backg);
   glutTimerFunc(300,MyTimerFunc,18);
}
else if(value==18)
{
   glutDisplayFunc(icecream);
   glutIdleFunc(icecream);
   glutTimerFunc(3600,MyTimerFunc,19);
}
else if(value==19)
{
   glutDisplayFunc(backg);
   glutIdleFunc(backg);
   glutTimerFunc(300,MyTimerFunc,20);
}
else if(value==20)
{
```

```
    glutDisplayFunc(jug);
    glutIdleFunc(jug);
    glutTimerFunc(3600,MyTimerFunc,21);
}
else if(value==21)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,22);
}
else if(value==22)
{
    glutDisplayFunc(kite);
    glutIdleFunc(kite);
    glutTimerFunc(3600,MyTimerFunc,23);
}
else if(value==23)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,24);
}
else if(value==24)
{
    glutDisplayFunc(leaf);
    glutIdleFunc(leaf);
    glutTimerFunc(3600,MyTimerFunc,25);
}
else if(value==25)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,26);
}
else if(value==26)
{
    glutDisplayFunc(moon);
    glutIdleFunc(moon);
    glutTimerFunc(3600,MyTimerFunc,27);
}
else if(value==27)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,0);
}
else if(value==0)
{
    glutDisplayFunc(nest);
    glutIdleFunc(nest);
```

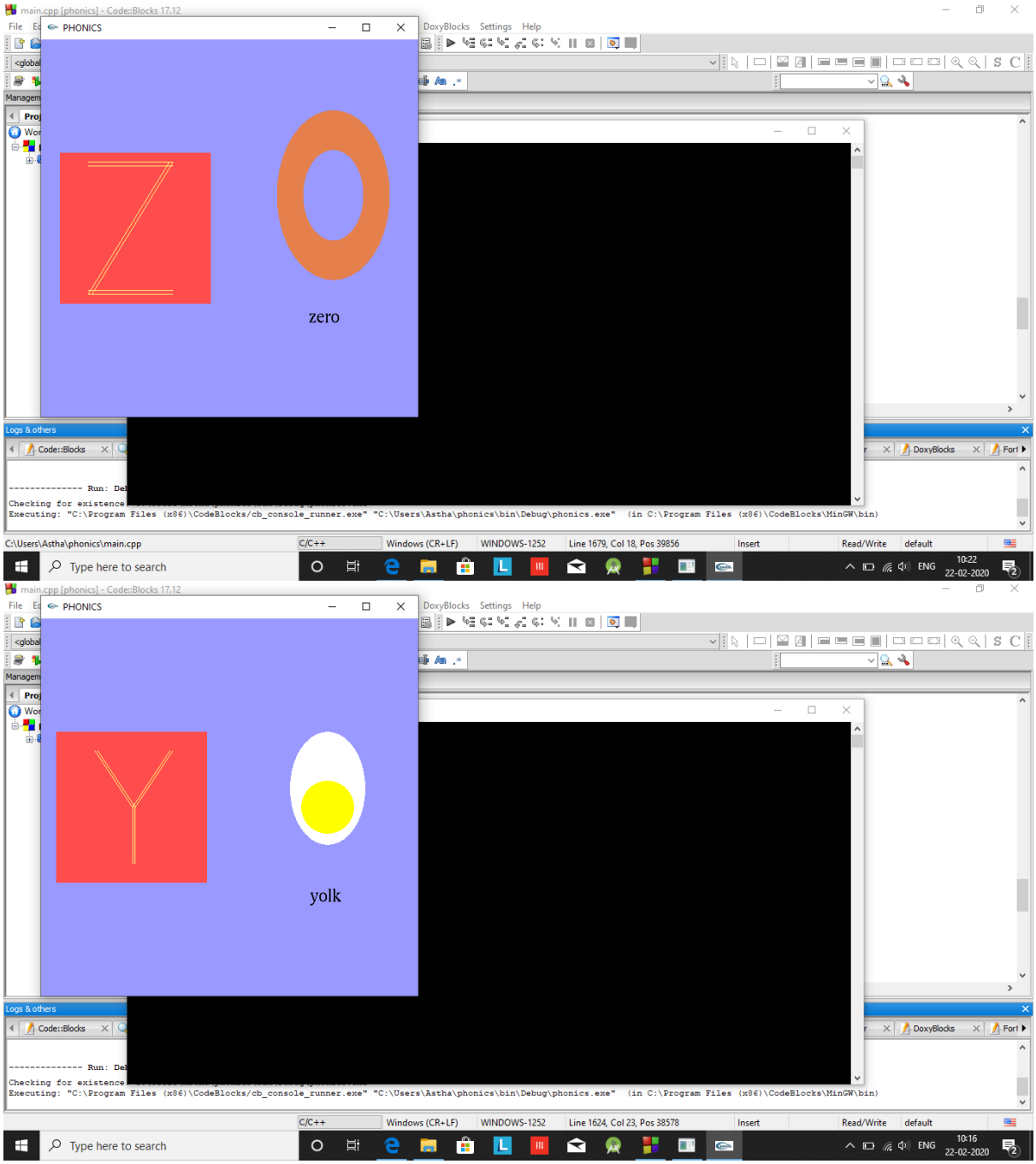
```
    glutTimerFunc(3600,MyTimerFunc,1);
}
else if(value==1)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,28);
}
else if(value==28)
{
    glutDisplayFunc(orange);
    glutIdleFunc(orange);
    glutTimerFunc(3600,MyTimerFunc,29);
}
else if(value==29)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,30);
}
else if(value==30)
{
    glutDisplayFunc(pot);
    glutIdleFunc(pot);
    glutTimerFunc(3600,MyTimerFunc,31);
}
else if(value==31)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,32);
}
else if(value==32)
{
    glutDisplayFunc(quilt);
    glutIdleFunc(quilt);
    glutTimerFunc(3600,MyTimerFunc,33);
}
else if(value==33)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,34);
}
else if(value==34)
{
    glutDisplayFunc(rainbow);
    glutIdleFunc(rainbow);
    glutTimerFunc(3600,MyTimerFunc,35);
}
```

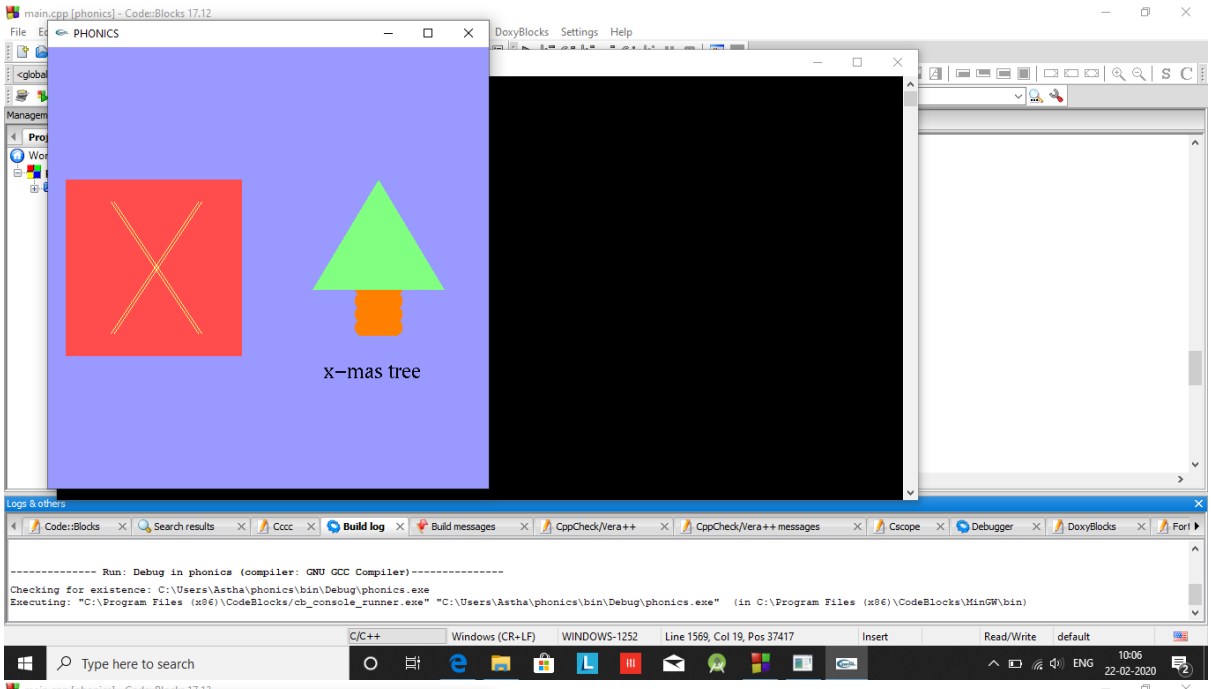
```
else if(value==35)
{
   glutDisplayFunc(backg);
   glutIdleFunc(backg);
   glutTimerFunc(300,MyTimerFunc,36);
}
else if(value==36)
{
   glutDisplayFunc(sun);
   glutIdleFunc(sun);
   glutTimerFunc(3600,MyTimerFunc,37);
}
else if(value==37)
{
   glutDisplayFunc(backg);
   glutIdleFunc(backg);
   glutTimerFunc(300,MyTimerFunc,38);
}
else if(value==38)
{
   glutDisplayFunc(train);
   glutIdleFunc(train);
   glutTimerFunc(3600,MyTimerFunc,39);
}
else if(value==39)
{
   glutDisplayFunc(backg);
   glutIdleFunc(backg);
   glutTimerFunc(300,MyTimerFunc,40);
}
else if(value==40)
{
   glutDisplayFunc(umbrella);
   glutIdleFunc(umbrella);
   glutTimerFunc(3600,MyTimerFunc,41);
}
else if(value==41)
{
   glutDisplayFunc(backg);
   glutIdleFunc(backg);
   glutTimerFunc(300,MyTimerFunc,42);
}
else if(value==42)
{
   glutDisplayFunc(van);
   glutIdleFunc(van);
   glutTimerFunc(3600,MyTimerFunc,43);
}
else if(value==43)
{
```

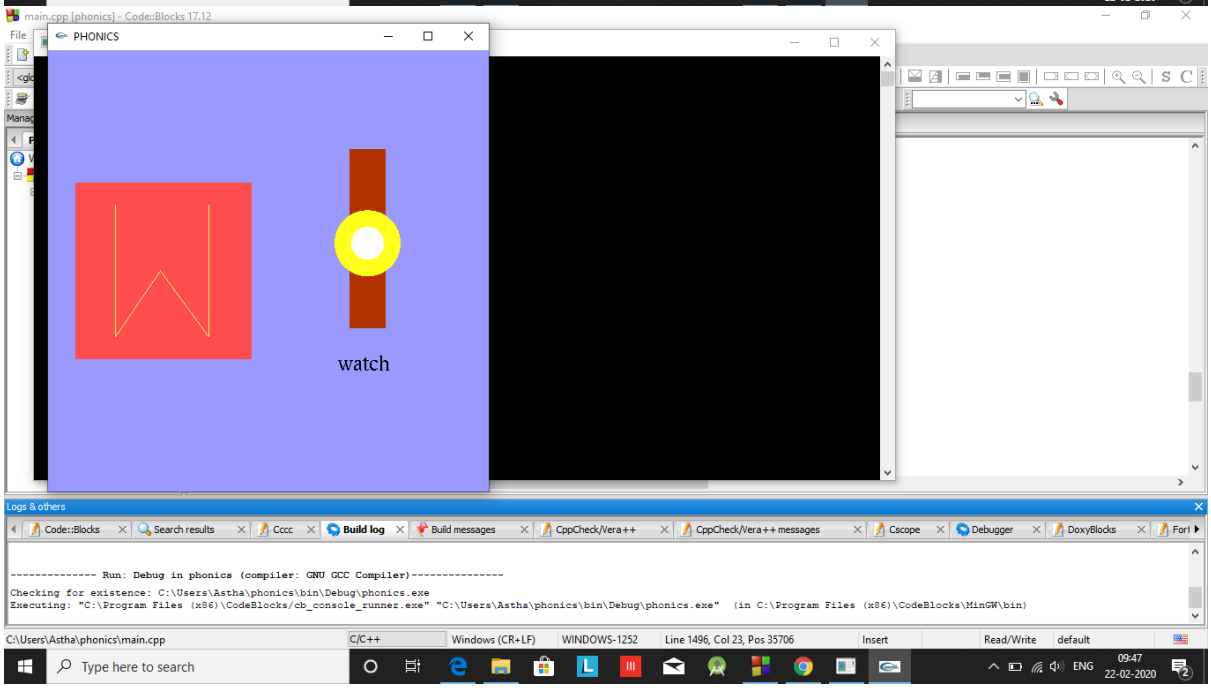
```
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,44);
}
else if(value==44)
{
    glutDisplayFunc(watch);
    glutIdleFunc(watch);
    glutTimerFunc(3600,MyTimerFunc,45);
}
else if(value==45)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,46);
}
else if(value==46)
{
    glutDisplayFunc(xmastree);
    glutIdleFunc(xmastree);
    glutTimerFunc(3600,MyTimerFunc,47);
}
else if(value==47)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,48);
}
else if(value==48)
{
    glutDisplayFunc(yolk);
    glutIdleFunc(yolk);
    glutTimerFunc(3600,MyTimerFunc,49);
}
else if(value==49)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
    glutTimerFunc(300,MyTimerFunc,50);
}
else if(value==50)
{
    glutDisplayFunc(zero);
    glutIdleFunc(zero);
    glutTimerFunc(3600,MyTimerFunc,51);
}
else if(value==51)
{
    glutDisplayFunc(backg);
    glutIdleFunc(backg);
```

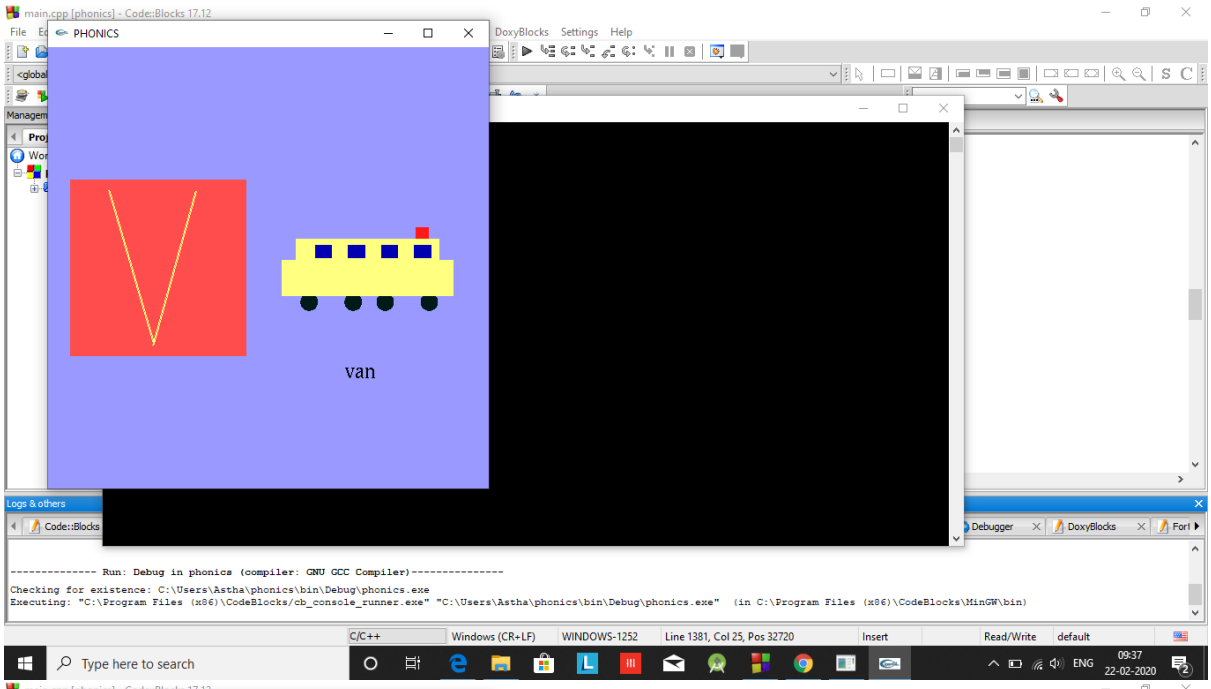
```
        //glutTimerFunc(300,MyTimerFunc,52);
    }

}

void resize(int w, int h)
{
    glViewport(0,0,w,h);
    glClearColor(0.6,0.6,1,1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-10.0, 10.0, -10.0, 10.0, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glutSwapBuffers();
}
int main(int argc, char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE|GLUT_DEPTH);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(50,50);
    glutCreateWindow("PHONICS");
    init();
    glutReshapeFunc(resize);
    //initOpenGl();
    //static int i=0;
    glutDisplayFunc(display);
    glutIdleFunc(display);
    glutTimerFunc(1000,MyTimerFunc,2);
    sndPlaySound("phon.wav",SND_ASYNC);
    //glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(0.6, 0.6, 1, 1);
    glutMainLoop();
    return 0;
}
```

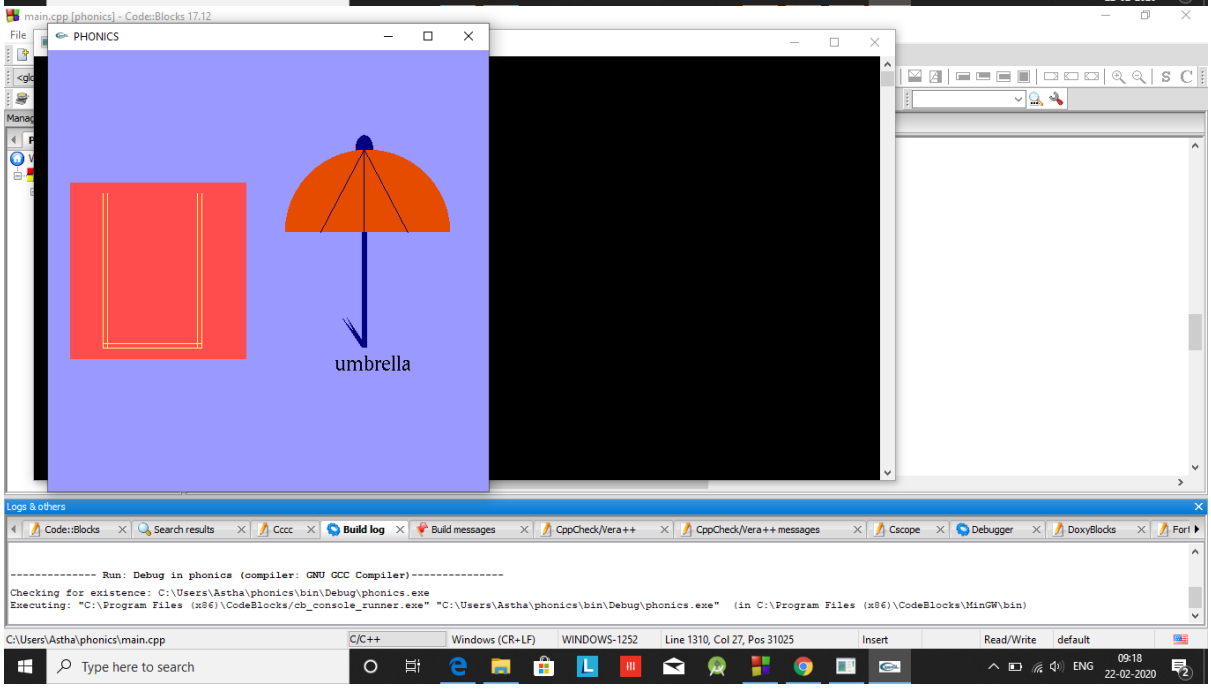
OUTPUT

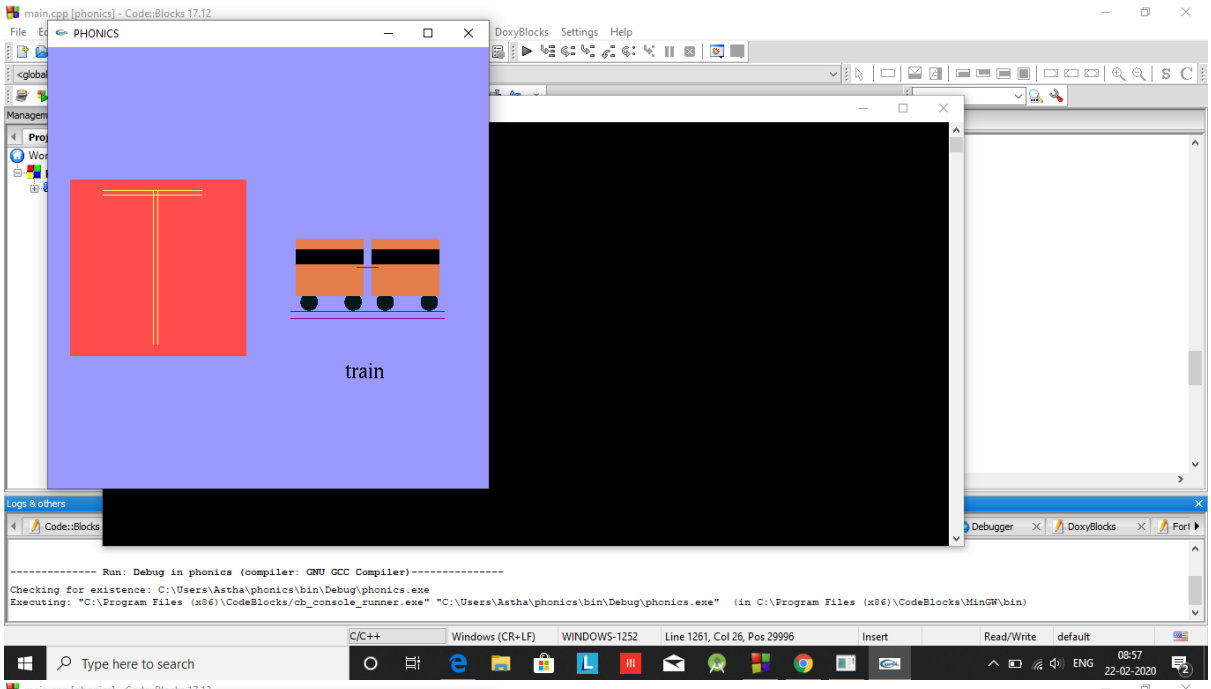


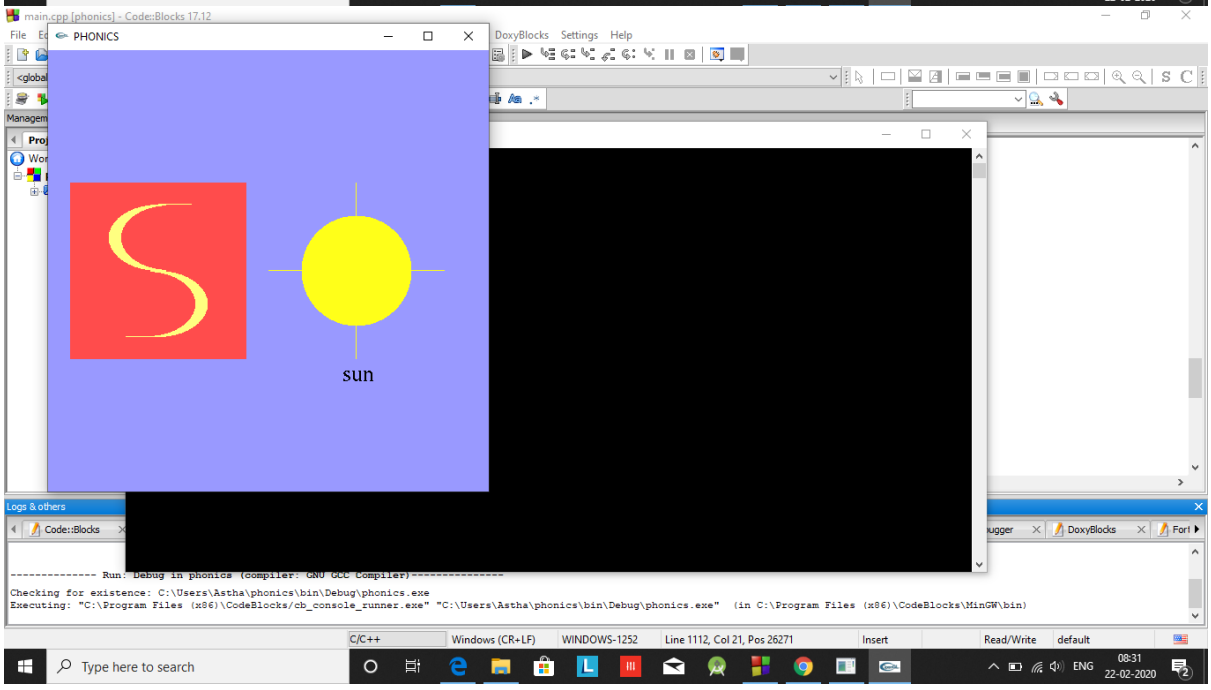

PHONICS
x-mas tree
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
DoxyBlocks
Run: Debug in phonics (compiler: GNU GCC Compiler)
Checking for existence: C:\Users\Astha\phonics\bin\Debug\phonics.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "C:\Users\Astha\phonics\bin\Debug\phonics.exe" (in C:\Program Files (x86)\CodeBlocks\MinGW\bin)
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1569, Col 19, Pos 37417
Insert
Read/Write
default
Type here to search
ENG
10:06
22-02-2020

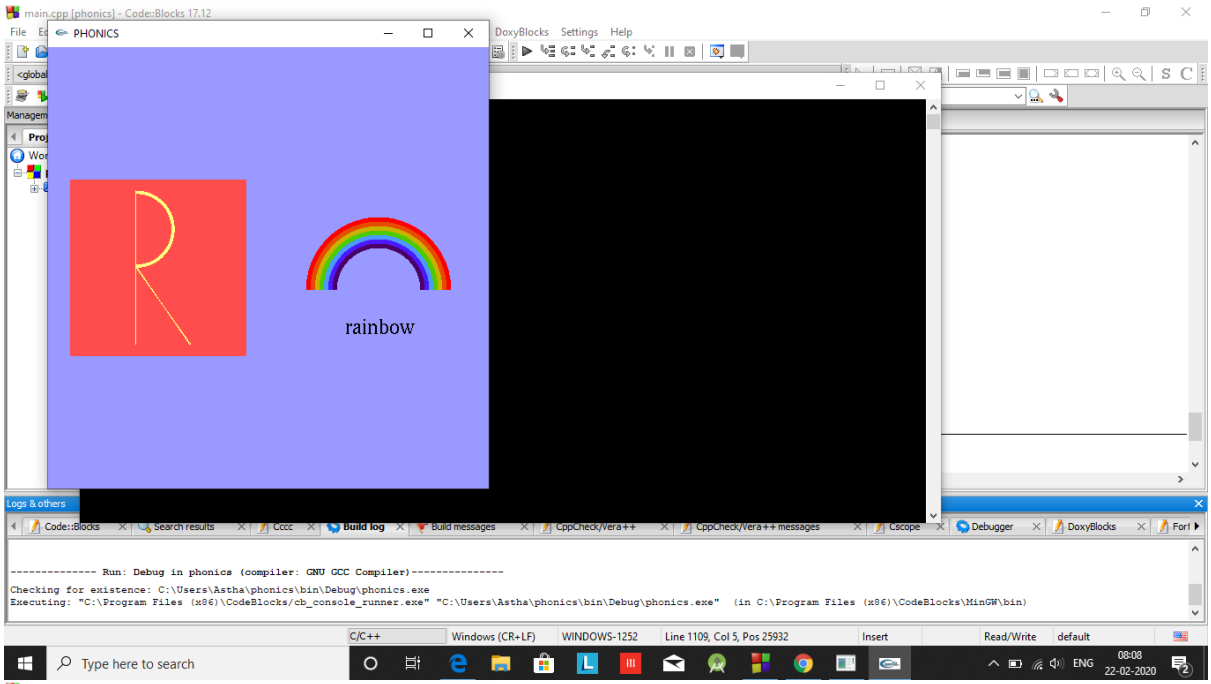

PHONICS
watch
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
DoxyBlocks
Run: Debug in phonics (compiler: GNU GCC Compiler)
Checking for existence: C:\Users\Astha\phonics\bin\Debug\phonics.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "C:\Users\Astha\phonics\bin\Debug\phonics.exe" (in C:\Program Files (x86)\CodeBlocks\MinGW\bin)
C:\Users\Astha\phonics\main.cpp
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1496, Col 23, Pos 35706
Insert
Read/Write
default
Type here to search
ENG
09:47
22-02-2020

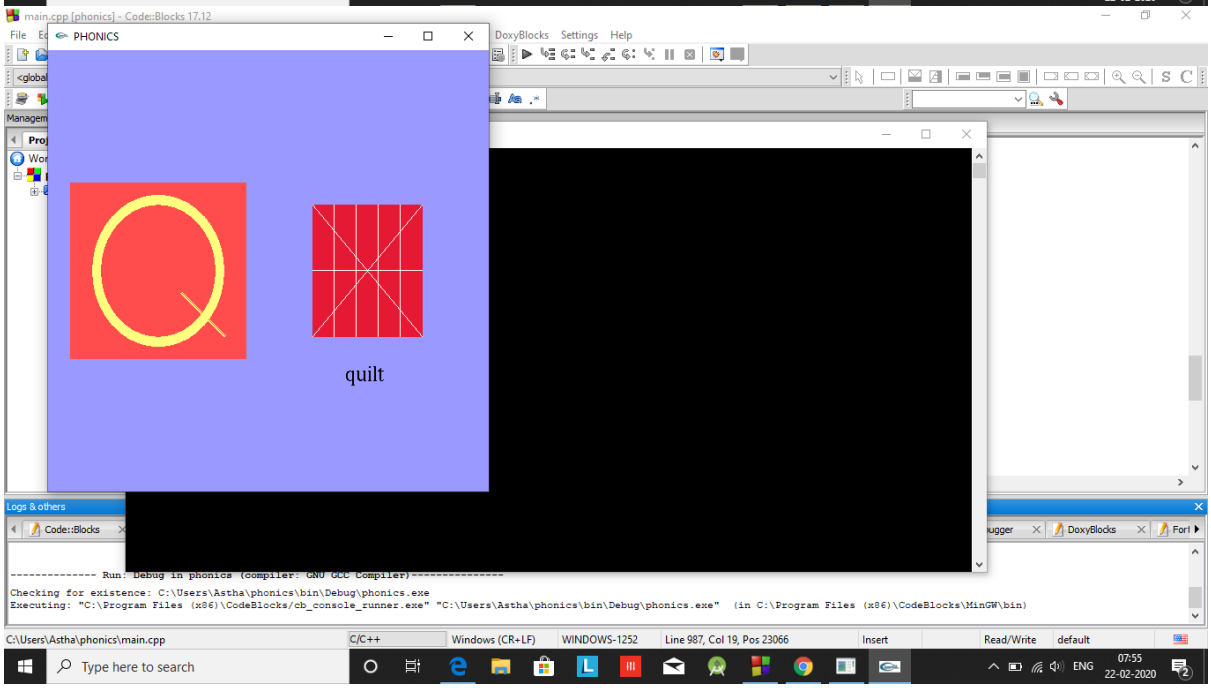

PHONICS
van

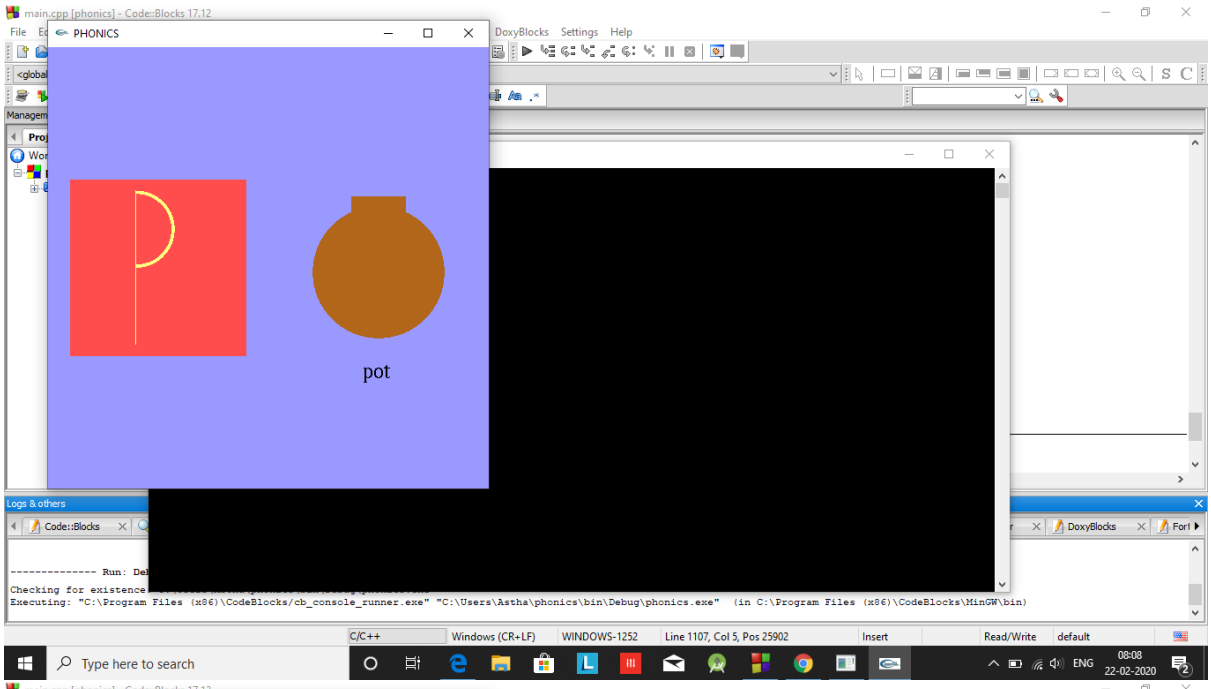

PHONICS
umbrella

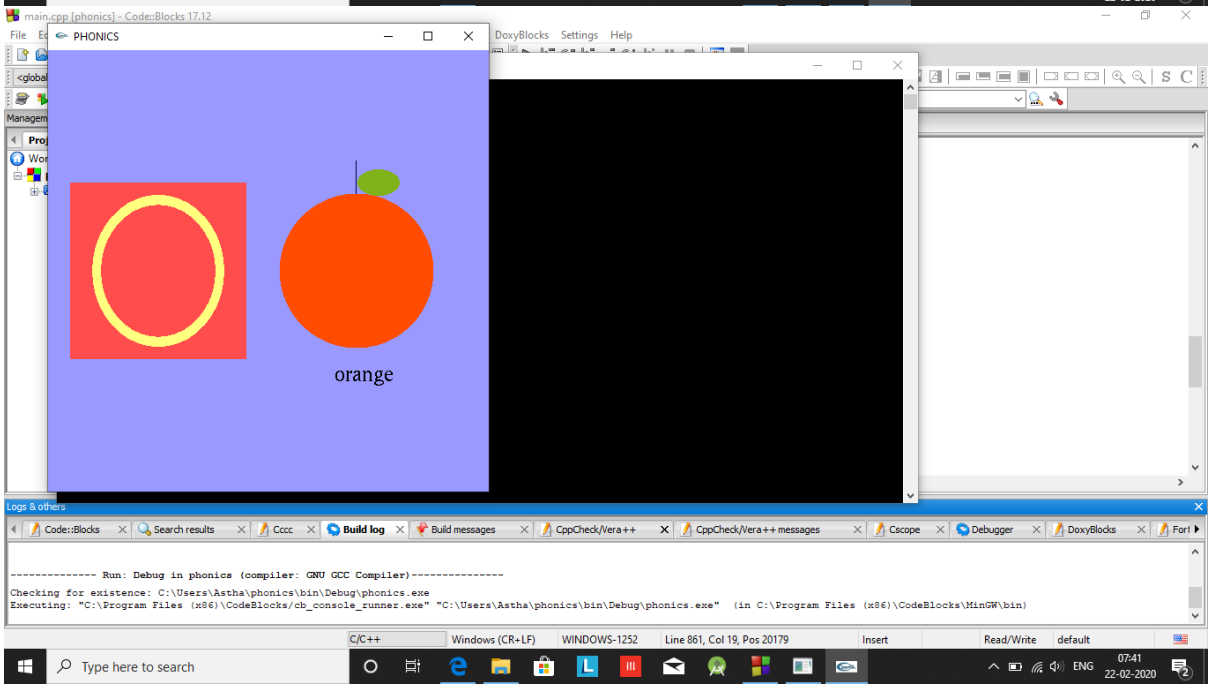

PHONICS
train

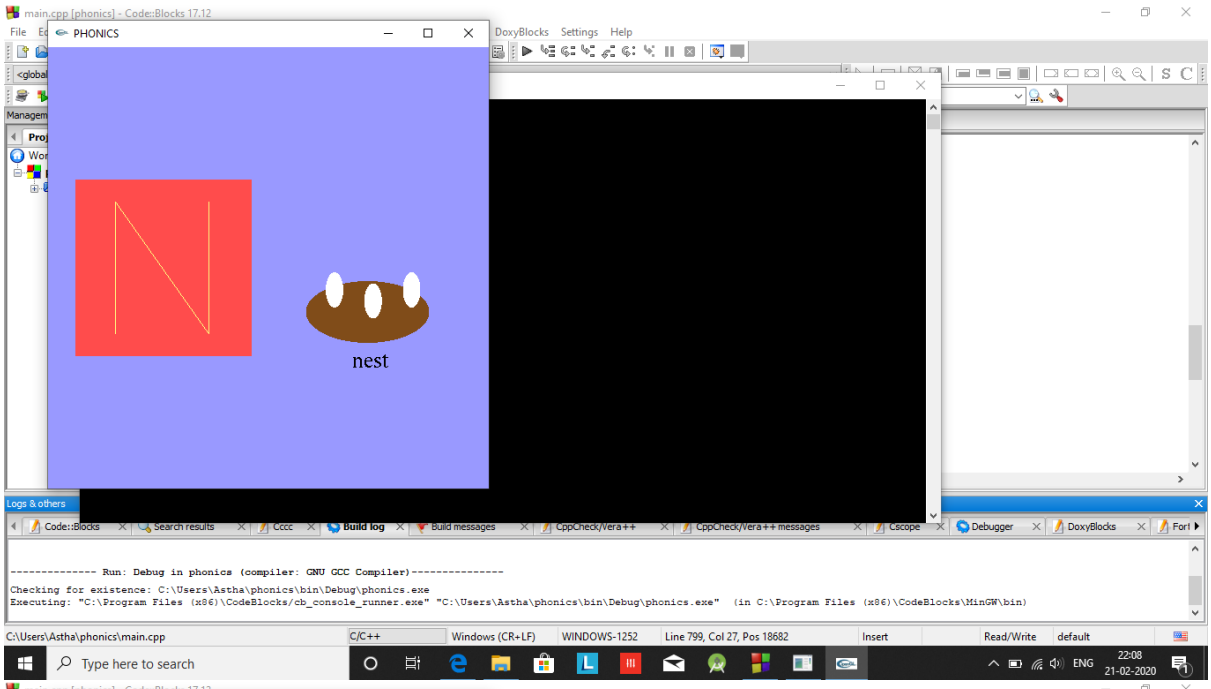

PHONICS
sun

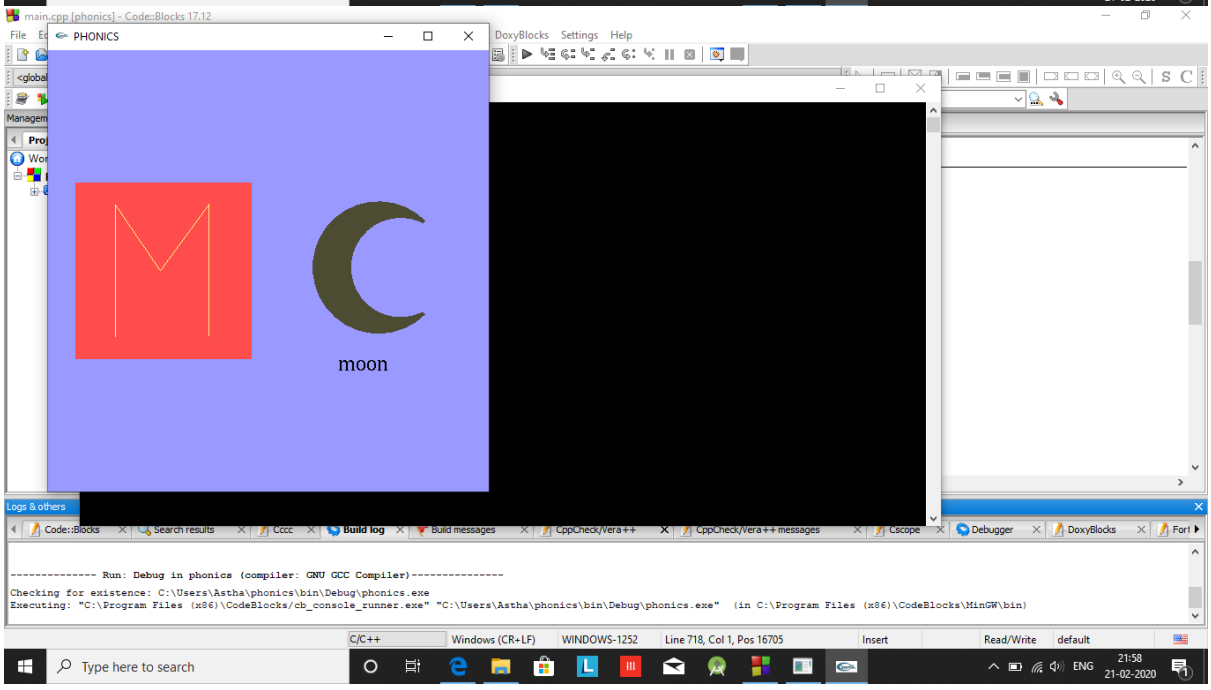

PHONICS
rainbow

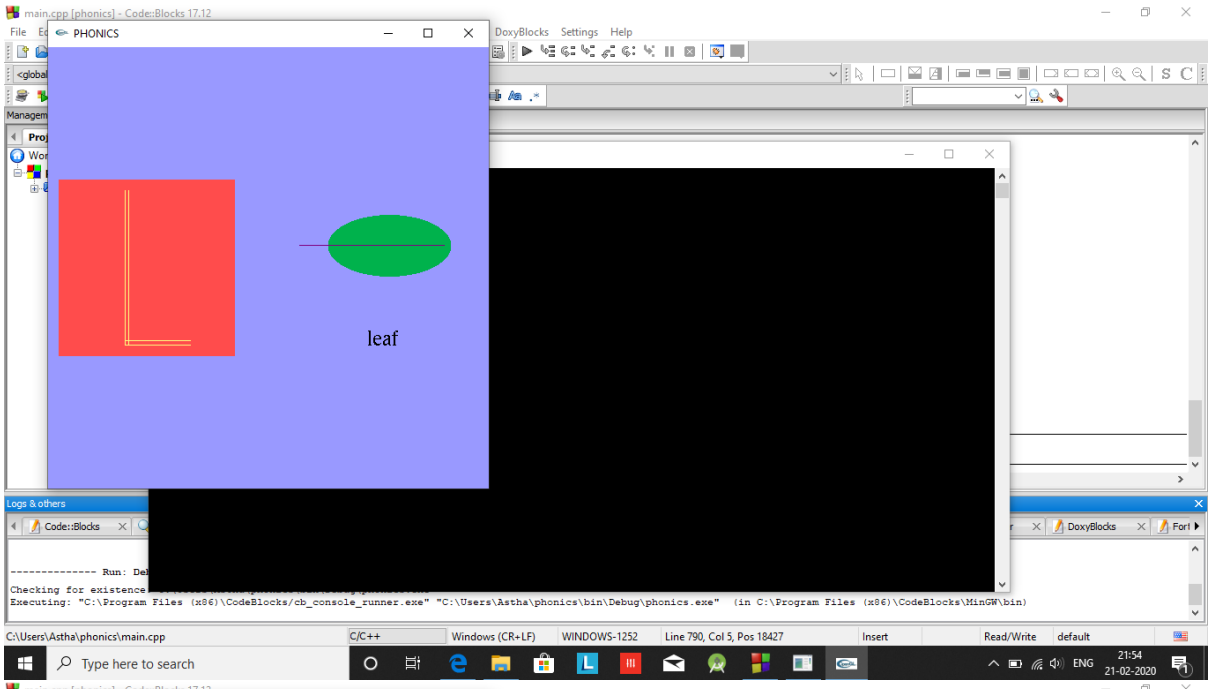

PHONICS
quilt

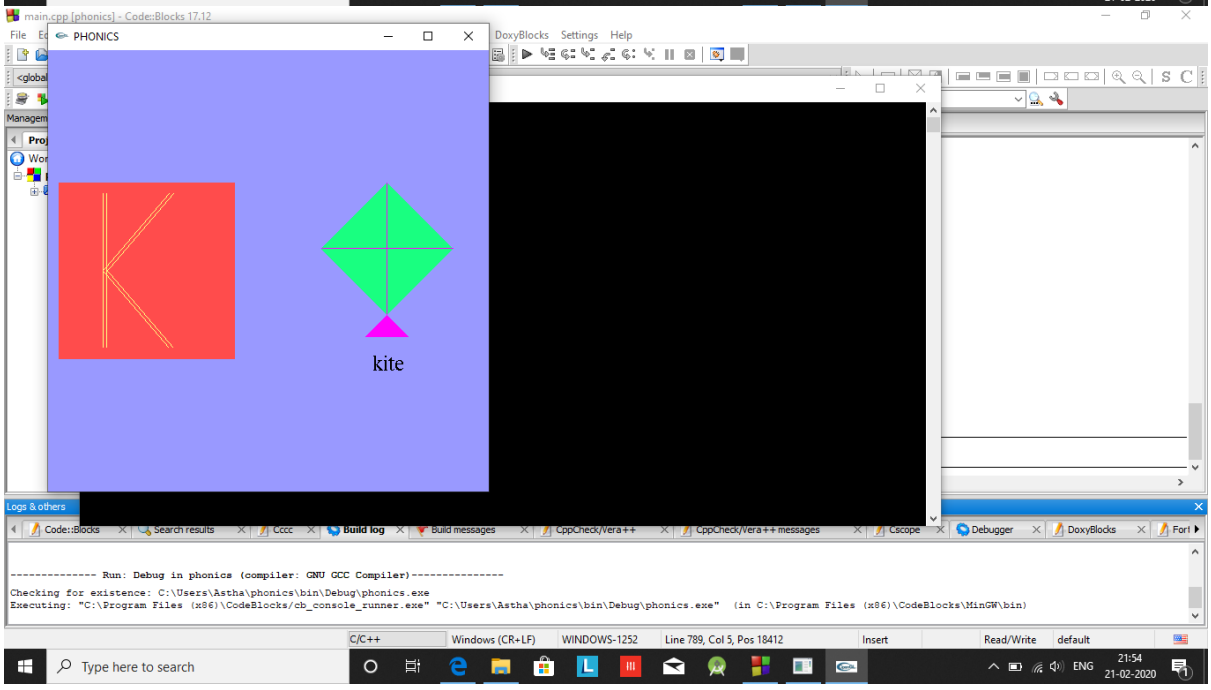

PHONICS
pot

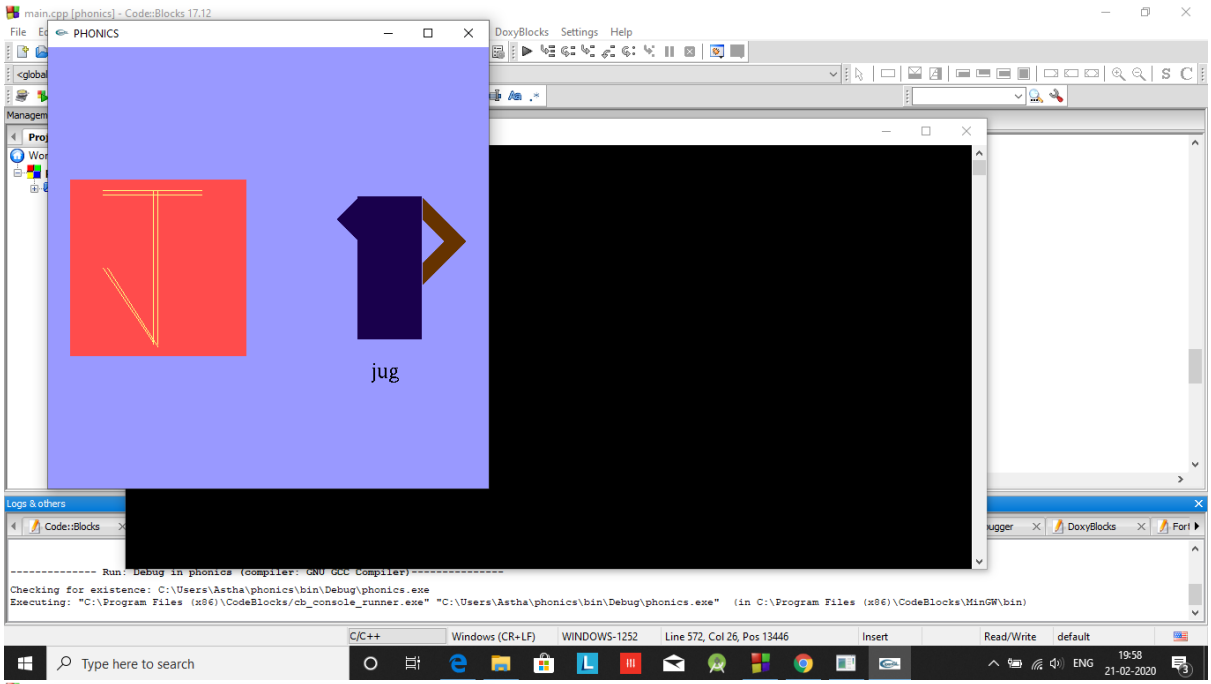

PHONICS
orange

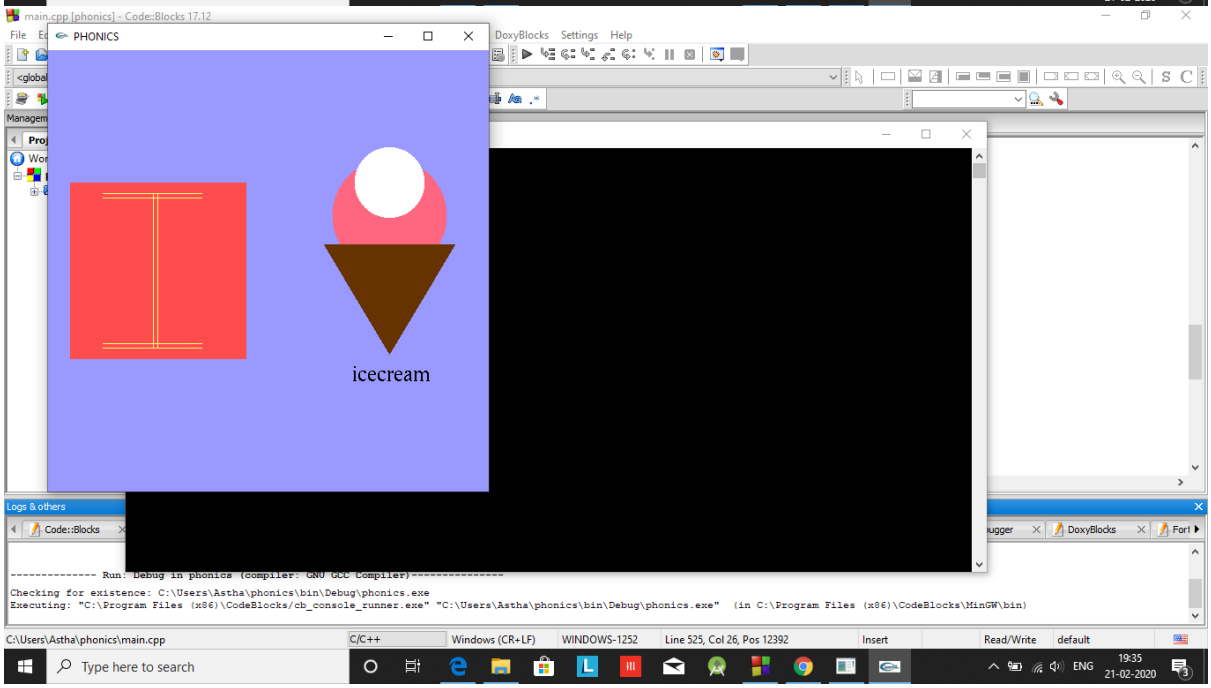

PHONICS
nest

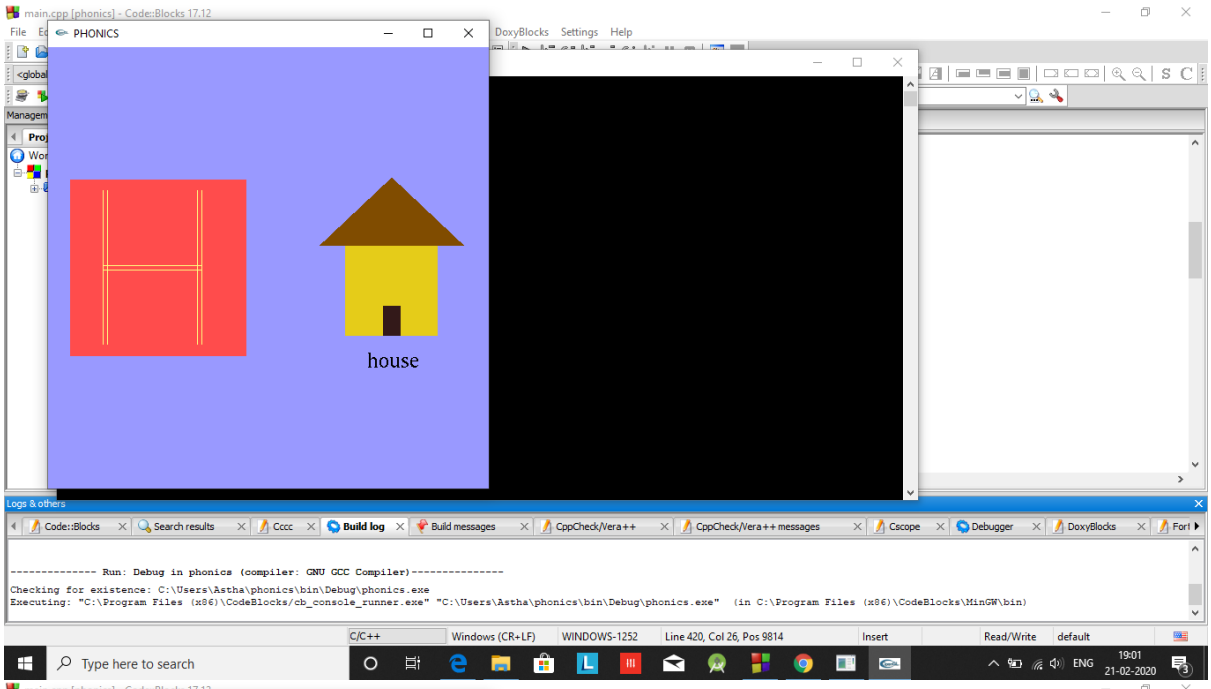

PHONICS
moon

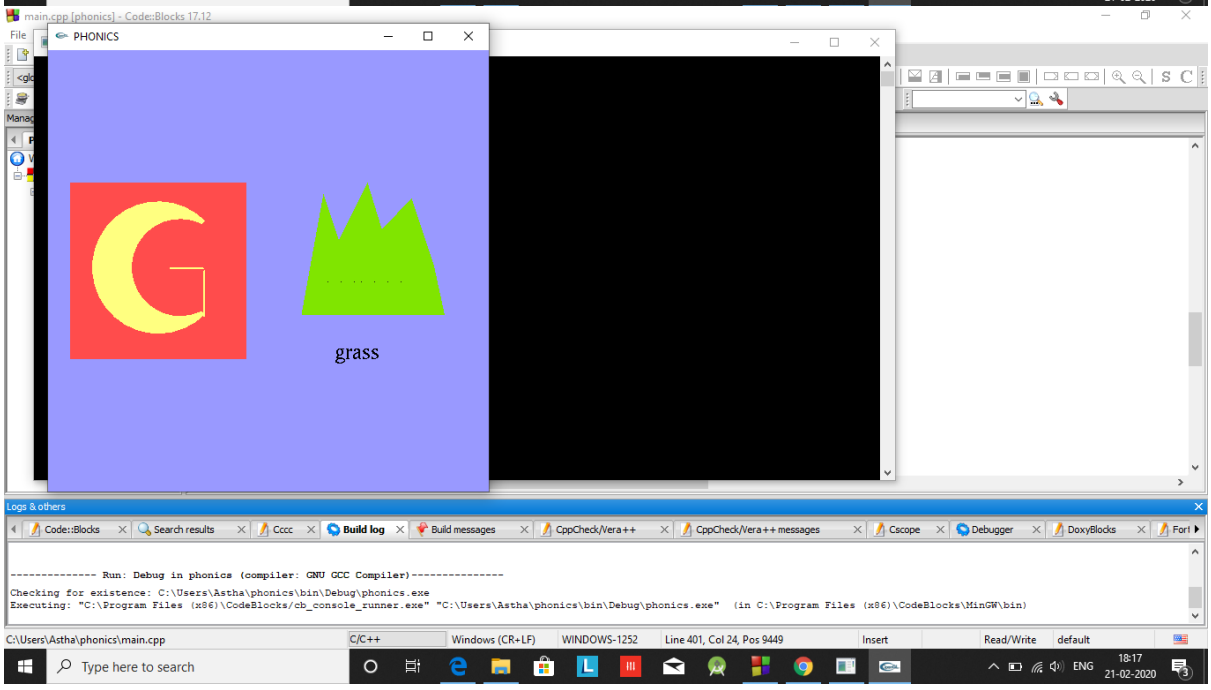

PHONICS
leaf

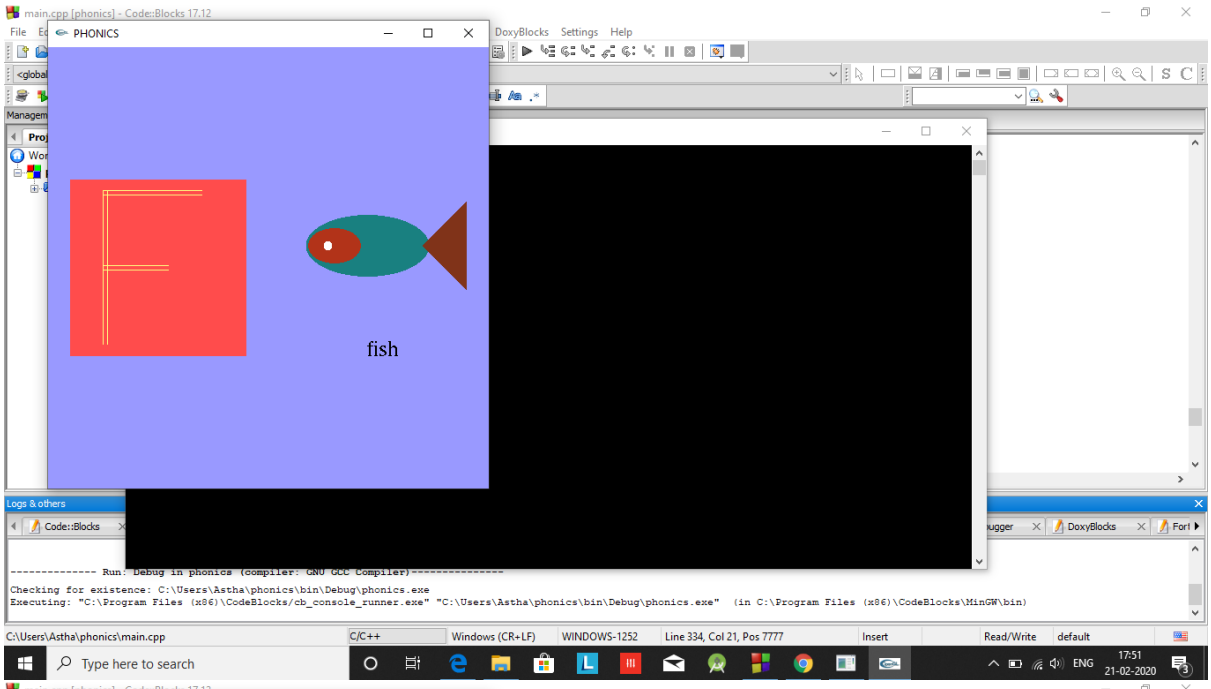

PHONICS
kite

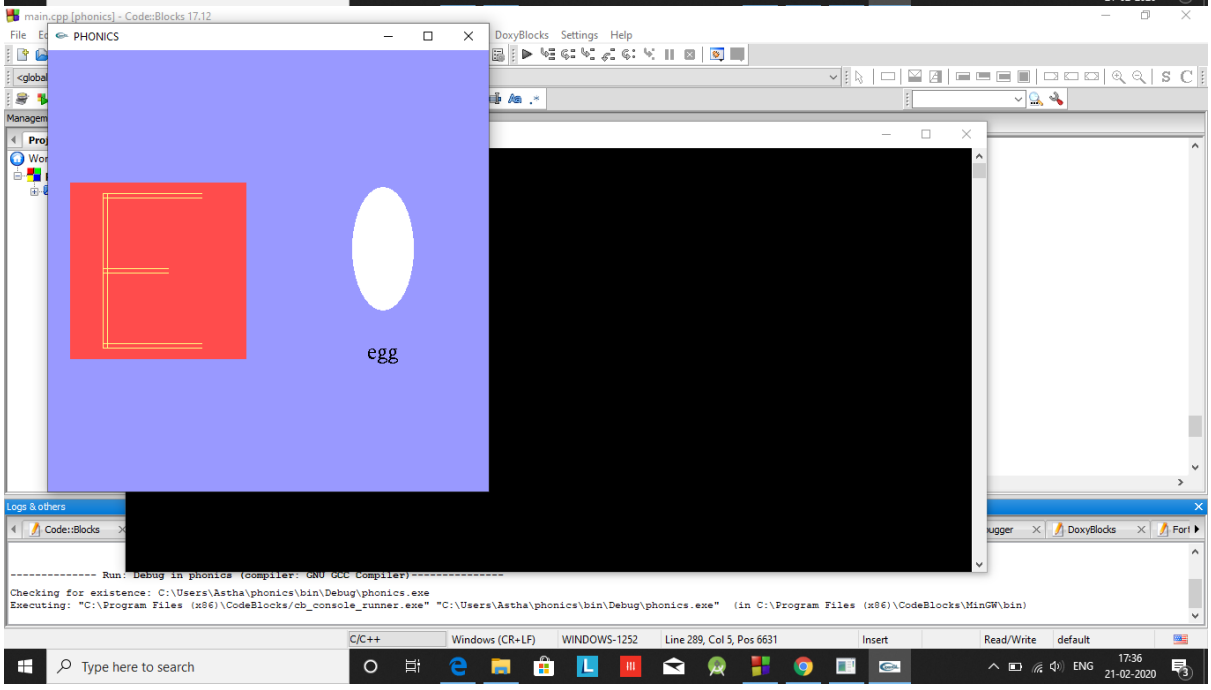

PHONICS
jug

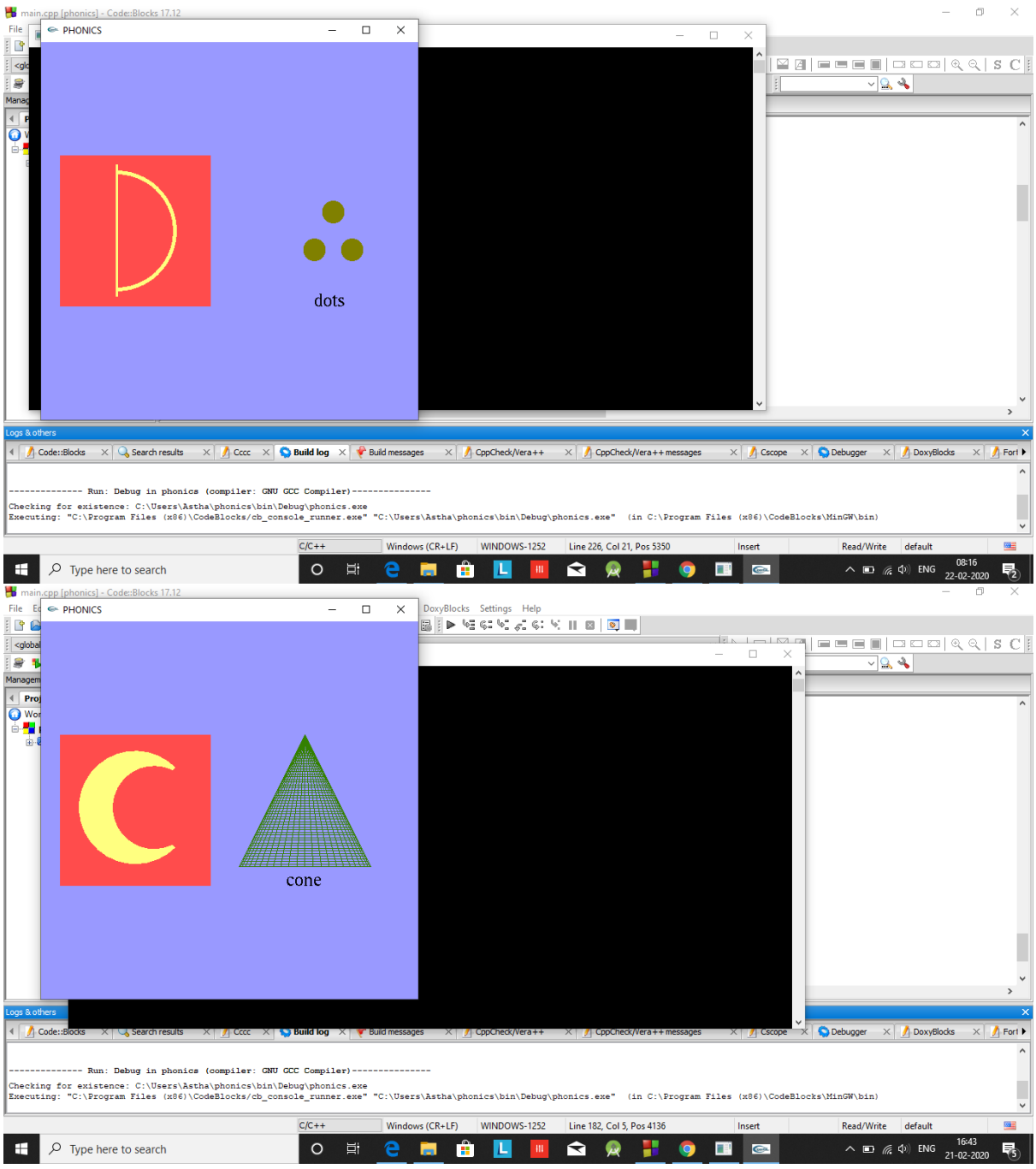

PHONICS
icecream


PHONICS
house
Logs & others
Build log
Run: Debug in phonics (compiler: GNU GCC Compiler)
Checking for existence: C:\Users\Astha\phonics\bin\Debug\phonics.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "C:\Users\Astha\phonics\bin\Debug\phonics.exe" (in C:\Program Files (x86)\CodeBlocks\MinGW\bin)
Type here to search

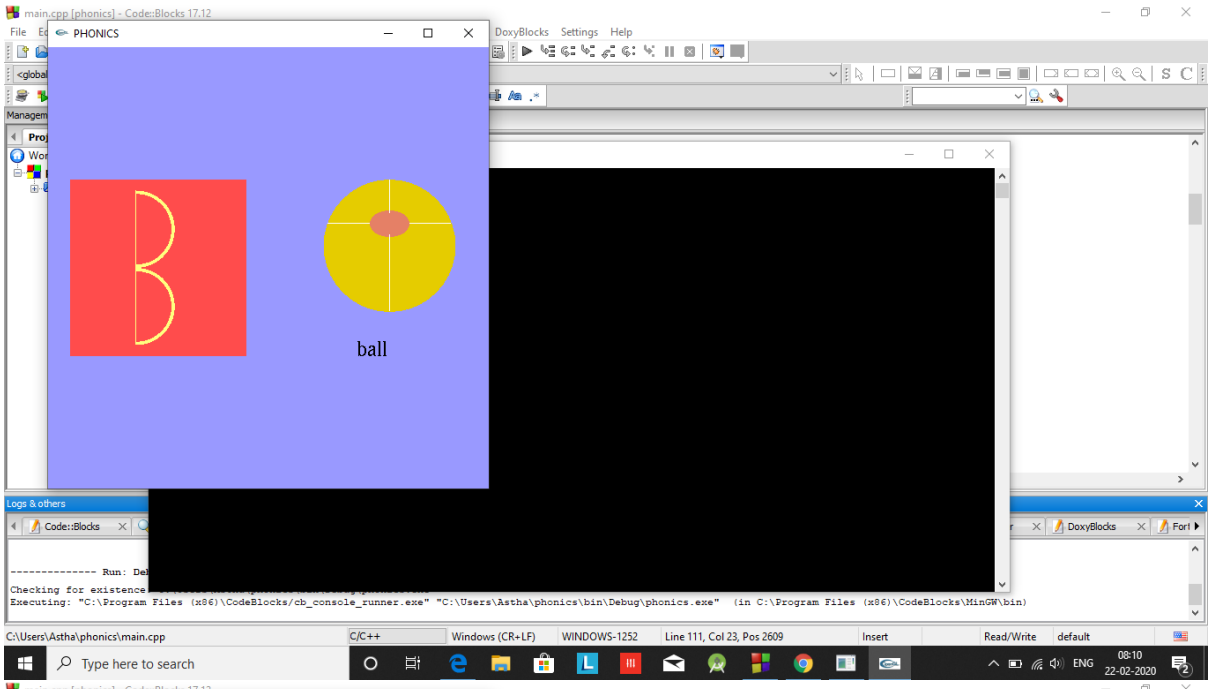

PHONICS
grass
Logs & others
Build log
Run: Debug in phonics (compiler: GNU GCC Compiler)
Checking for existence: C:\Users\Astha\phonics\bin\Debug\phonics.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "C:\Users\Astha\phonics\bin\Debug\phonics.exe" (in C:\Program Files (x86)\CodeBlocks\MinGW\bin)
Type here to search

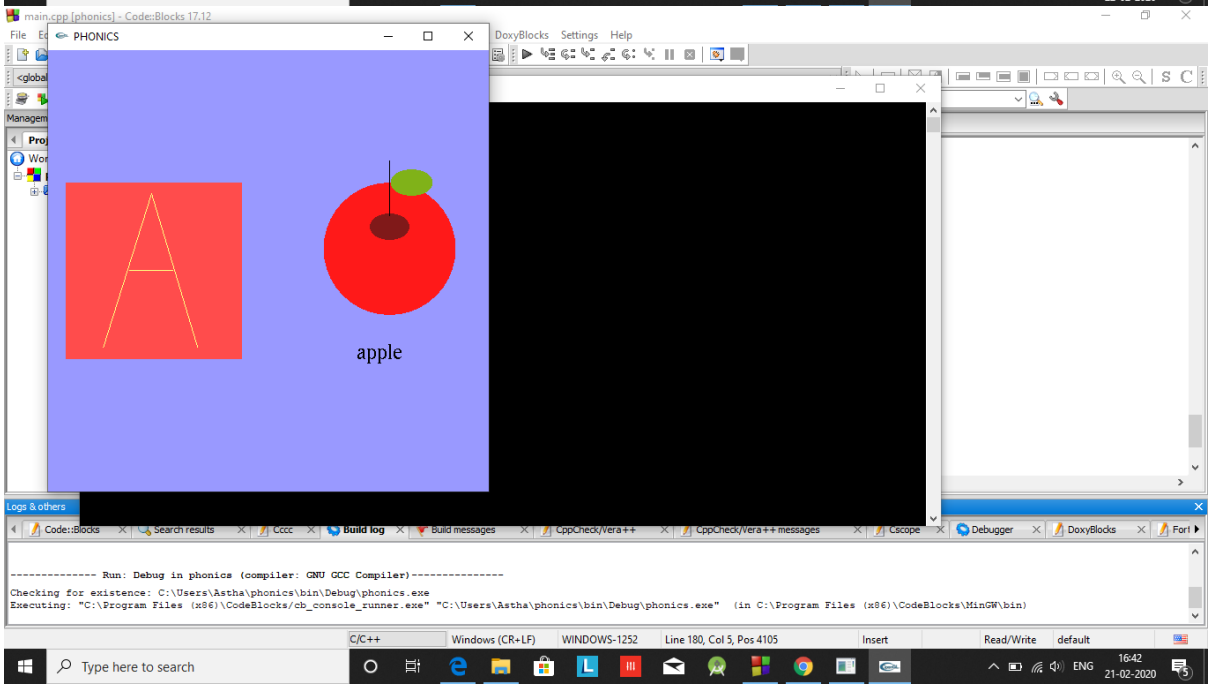

PHONICS
fish


PHONICS
egg


PHONICS
dots
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
DoxyBlocks
Fort
---------- Run: Debug in phonics (compiler: GNU GCC Compiler)----------
Checking for existence: C:\Users\Astha\phonics\bin\Debug\phonics.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "C:\Users\Astha\phonics\bin\Debug\phonics.exe" (in C:\Program Files (x86)\CodeBlocks\MinGW\bin)
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 226, Col 21, Pos 5350
Insert
Read/Write
default
Type here to search
ENG
08:16
22-02-2020

PHONICS
cone
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
DoxyBlocks
Fort
---------- Run: Debug in phonics (compiler: GNU GCC Compiler)----------
Checking for existence: C:\Users\Astha\phonics\bin\Debug\phonics.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "C:\Users\Astha\phonics\bin\Debug\phonics.exe" (in C:\Program Files (x86)\CodeBlocks\MinGW\bin)
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 182, Col 5, Pos 4136
Insert
Read/Write
default
Type here to search
ENG
16:43
21-02-2020


PHONICS
ball


PHONICS
apple

CONCLUSION

Finally we conclude that this program clearly illustrate the Phonics Play using openGL and has been completed successfully and is ready to be demonstrated.