Content

1. Content Requirements

1.1 Functional Requirements

1.1.1 There should be the most basic trace drawing: drawing lines, circles, rectangles, triangles, any Curves (except lines, there are two other types: (1) filling (2) drawing edges).
1.1.2 Clear the screen.
1.1.3 Brush size setting, you can easily change the brush size.
1.1.4 There is a rubber. Set the rubber to be large. Hours can be selected (at least three types: large, medium and small).
1.1.5 To quickly select colors (red, green, yellow, blue, black, etc.) and any color selection.
1.1.6 There can be background music.
1.1.7 The user can keep the current picture.
1.1.8 The user can open pictures.
1.1.9 The drawing area can be changed at will.
1.1.10 Cancel the operation (only cancel the shape action drawn before).

1.2 Non-Functional Requirements

1.2.1 There is a status bar, which shows which shape the current operation is and whether it is filled.
1.2.2 There are student classes, students' Chinese Pinyin names (not abbreviations) and SCN numbers on the interface.
1.2.3. Instructions for use (help).
1.2.4 There are nested and self written functions.
1.2.5 The control must be placed on the left (part can be placed on the top).
1.2.6 There must be a menu.

2. User Interface Design

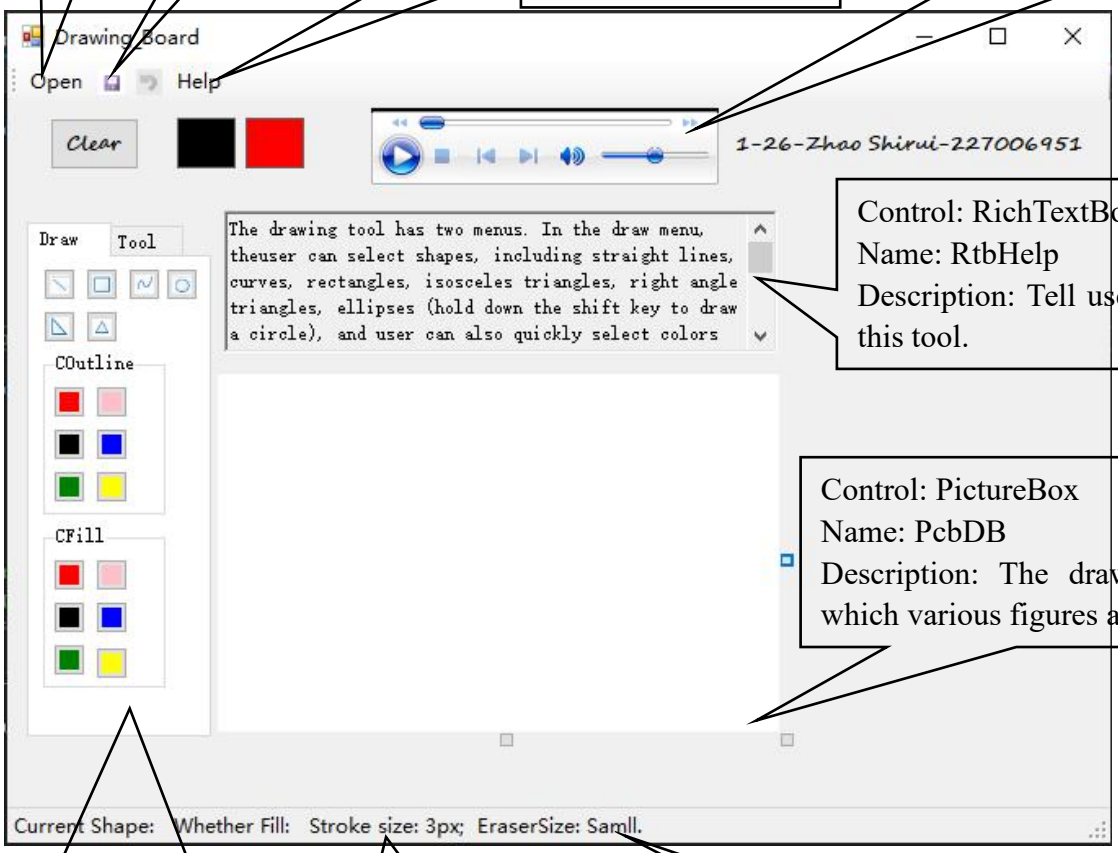
This is a Sketchpad interface where users can draw straight lines, curves, isosceles triangles, right triangles, rectangles, ellipses, circles, etc. The user can select whether to fill and draw the border, and user can also select the fill color and border color. Users can adjust the canvas size or choose to clear the screen directly. Users can open or save a picture. The program also has a music interface with a built-in music that can be played repeatedly. If the user cannot operate this interface, click help to view the instructions.

Control: ToolStripButton
Name: TsbOpen
Description: Open a picture file.

Control: ToolStripButton
Name: TsbSave
Description: Save drawn picture.

Control: ToolStripButton
Name: TsbHelp
Description: Click to pop up RtbHelp

Control: AxWindowsMediaPlayer
Name: WmpPlayer
Description: Play saved music.



Control: RichTextBox
Name: RtbHelp
Description: Tell users how to use this tool.

Control: PictureBox
Name: PcbDB
Description: The drawing area on which various figures are drawn.

Control: TabControl
Name: TbCDB
Description: Menu interface, users can switch between drawing and tools

Control: ToolStripStatusLabel
Name: TslStrokesize
Description: Used to tell the user the current pen thickness.

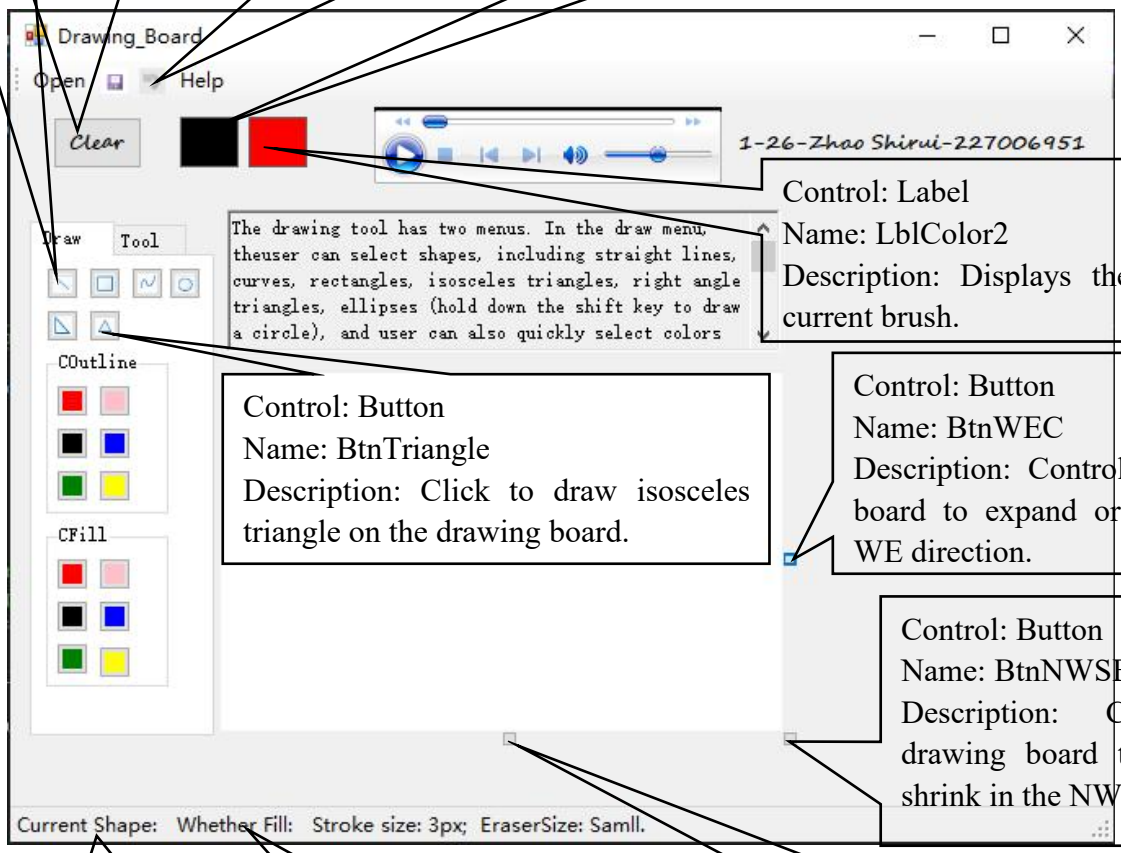
Control: ToolStripStatusLabel
Name: TslEraserSize
Description: Used to tell the user the current eraser size.

Control: Button
Name: BtnLine
Description: Click to draw a straight line on the drawing board.

Control: Button
Name: BtnClear
Description: Clear the canvas

Control: ToolStripButton
Name: TsbUndo
Description: Undo the user's last drawing trace.

Control: Label
Name: LblColor1
Description: Displays the color of the current pen.



Control: Label
Name: LblColor2
Description: Displays the color of the current brush.

Control: Button
Name: BtnTriangle
Description: Click to draw isosceles triangle on the drawing board.

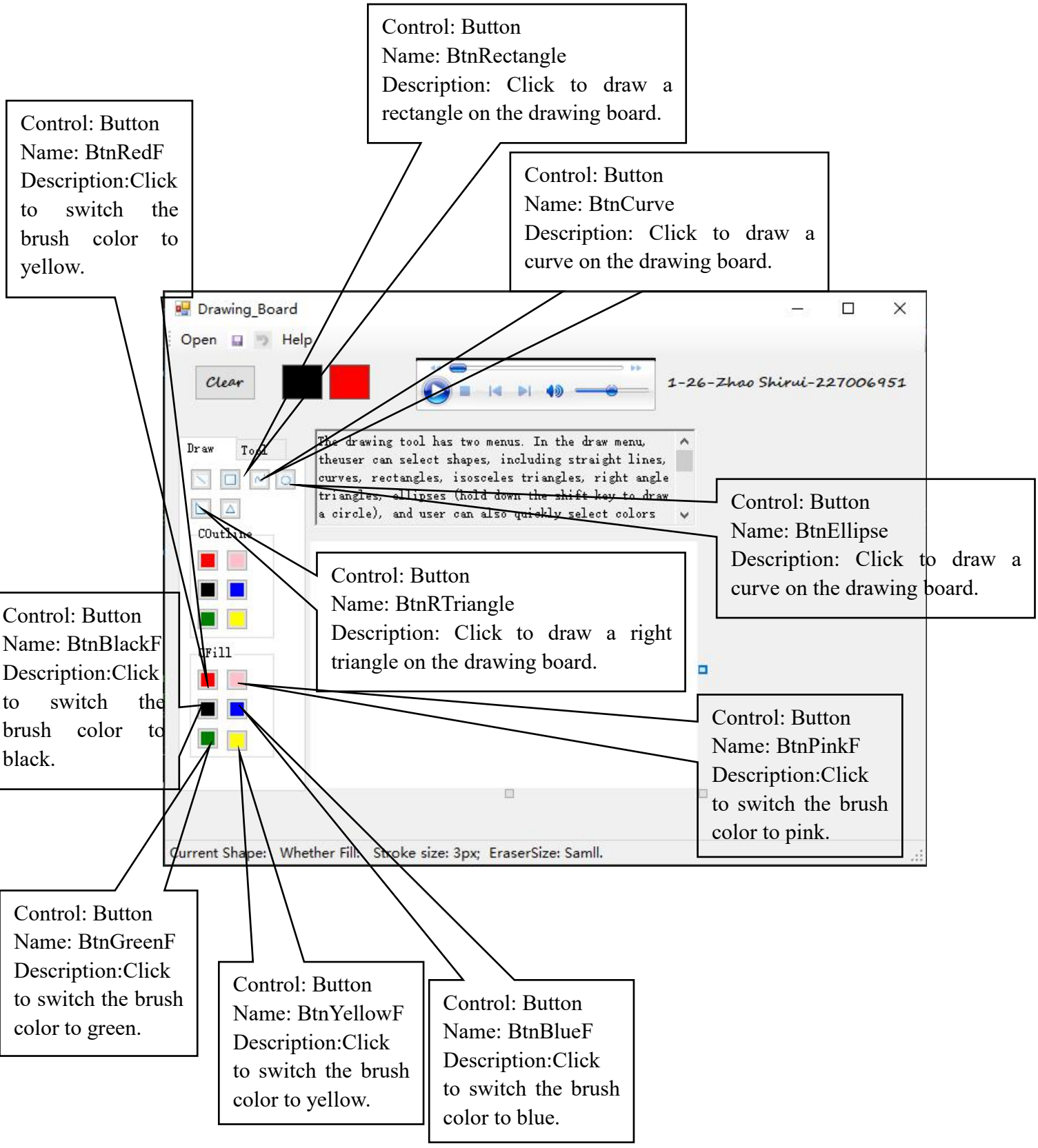
Control: Button
Name: BtnWEC
Description: Controls the drawing board to expand or shrink in the WE direction.

Control: Button
Name: BtnNWSEC
Description: Controls the drawing board to expand or shrink in the NWSE direction.

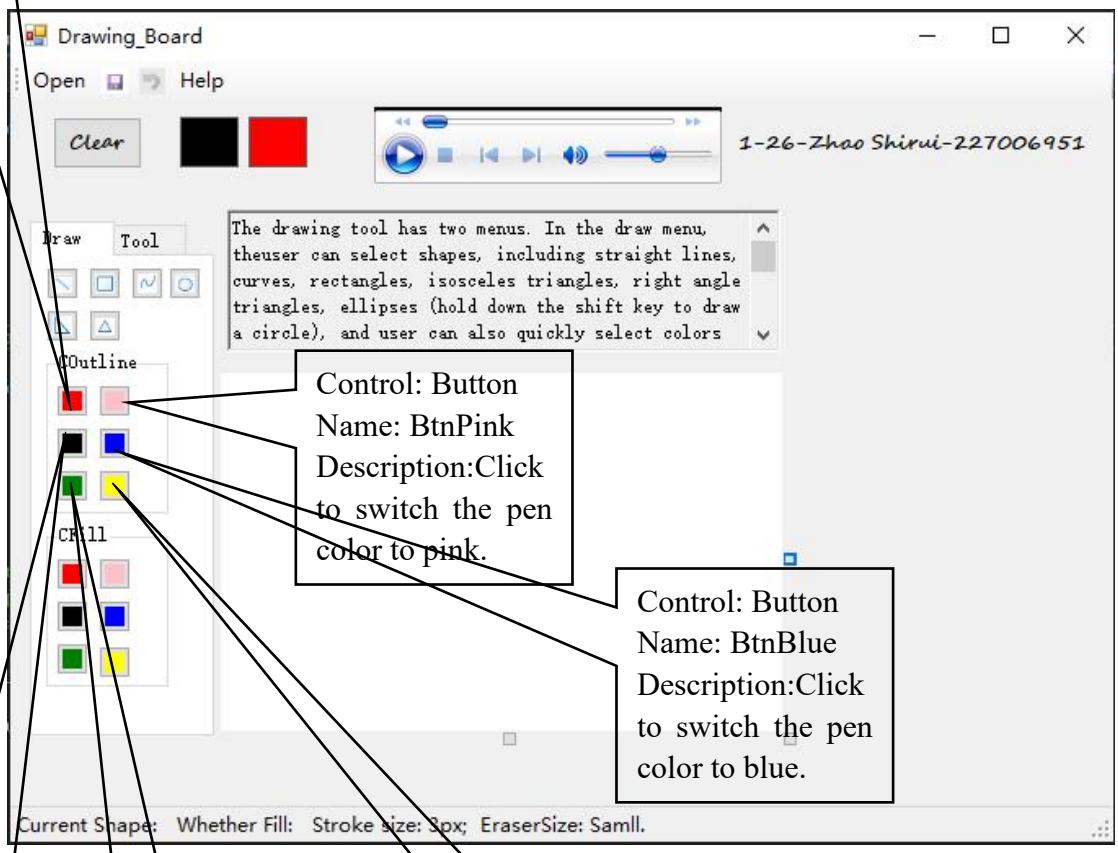
Control: Button
Name: BtnNSC
Description: Controls the drawing board to expand or shrink in the NS direction.

Control: ToolStripStatusLabel
Name: TslShape
Description: Displays the shape of the current drawing.

Control: ToolStripStatusLabel
Name: TslWFill
Description: Prompts the user whether to fill in the current.


Control: Button
Name: BtnRectangle
Description: Click to draw a rectangle on the drawing board.
Control: Button
Name: BtnRedF
Description:Click to switch the brush color to yellow.
Control: Button
Name: BtnCurve
Description: Click to draw a curve on the drawing board.
Drawing_Board
Open
Help
Clear
1-26-Zhao Shirui-227006951
Draw
Tool
The drawing tool has two menus. In the draw menu, theuser can select shapes, including straight lines, curves, rectangles, isosceles triangles, right angle triangles, ellipses (hold down the shift key to draw a circle), and user can also quickly select colors
COutline
Fill
Current Shape: Whether Fill: Stroke size: 3px; EraserSize: Samll.
Control: Button
Name: BtnEllipse
Description: Click to draw a curve on the drawing board.
Control: Button
Name: BtnRTriangle
Description: Click to draw a right triangle on the drawing board.
Control: Button
Name: BtnBlackF
Description:Click to switch the brush color to black.
Control: Button
Name: BtnPinkF
Description:Click to switch the brush color to pink.
Control: Button
Name: BtnGreenF
Description:Click to switch the brush color to green.
Control: Button
Name: BtnYellowF
Description:Click to switch the brush color to yellow.
Control: Button
Name: BtnBlueF
Description:Click to switch the brush color to blue.

Control: Button
Name: BtnRed
Description:Click to switch the pen color to red.



Control: Button
Name: BtnPink
Description:Click to switch the pen color to pink.

Control: Button
Name: BtnBlue
Description:Click to switch the pen color to blue.

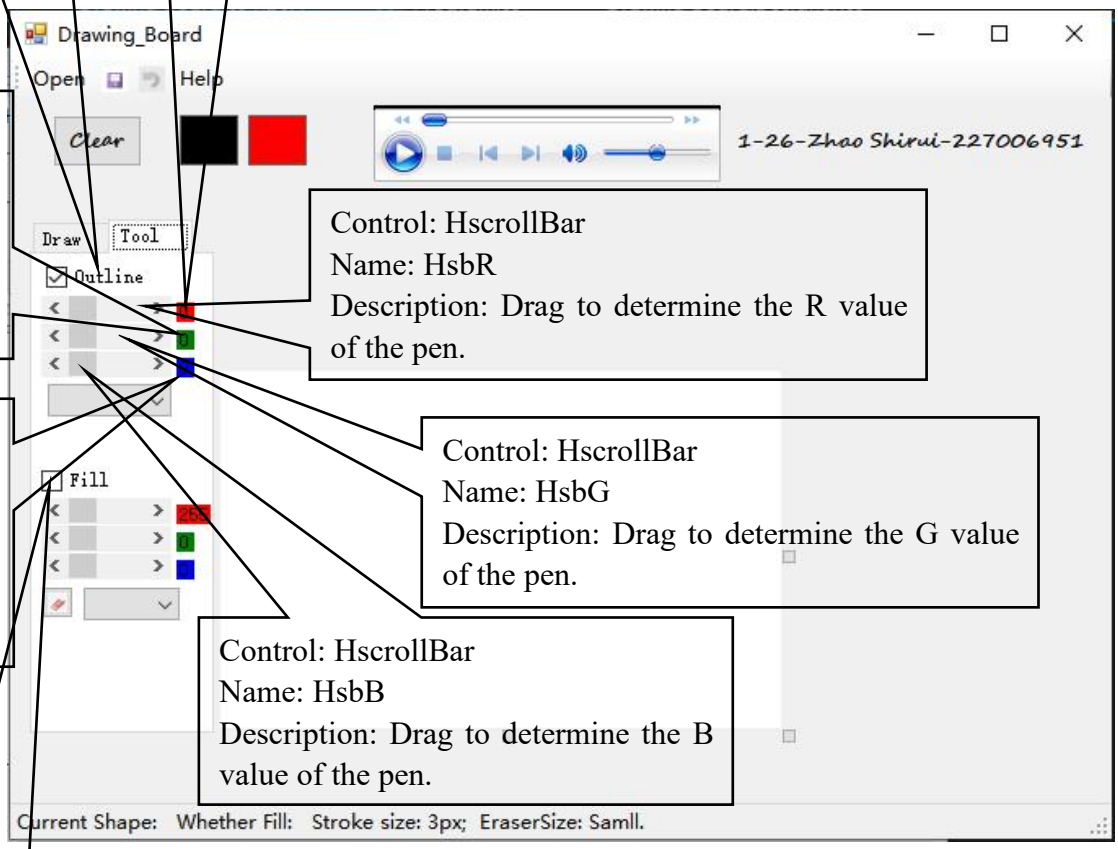
Control: Button
Name: BtnBlack
Description:Click to switch the pen color to black.

Control: Button
Name: BtnGreen
Description:Click to switch the pen color to green.

Control: Button
Name: BtnYellow
Description:Click to switch the pen color to Yellow.

Control: CheckBox
Name: CbxOutline
Description: Click OK to select whether to draw the border of the drawing.

Control: Label
Name: LblR
Description: Display the value of pen's R.



Control: Label
Name: LblG
Description: Display the value of pen's G.

Control: HscrollBar
Name: HsbR
Description: Drag to determine the R value of the pen.

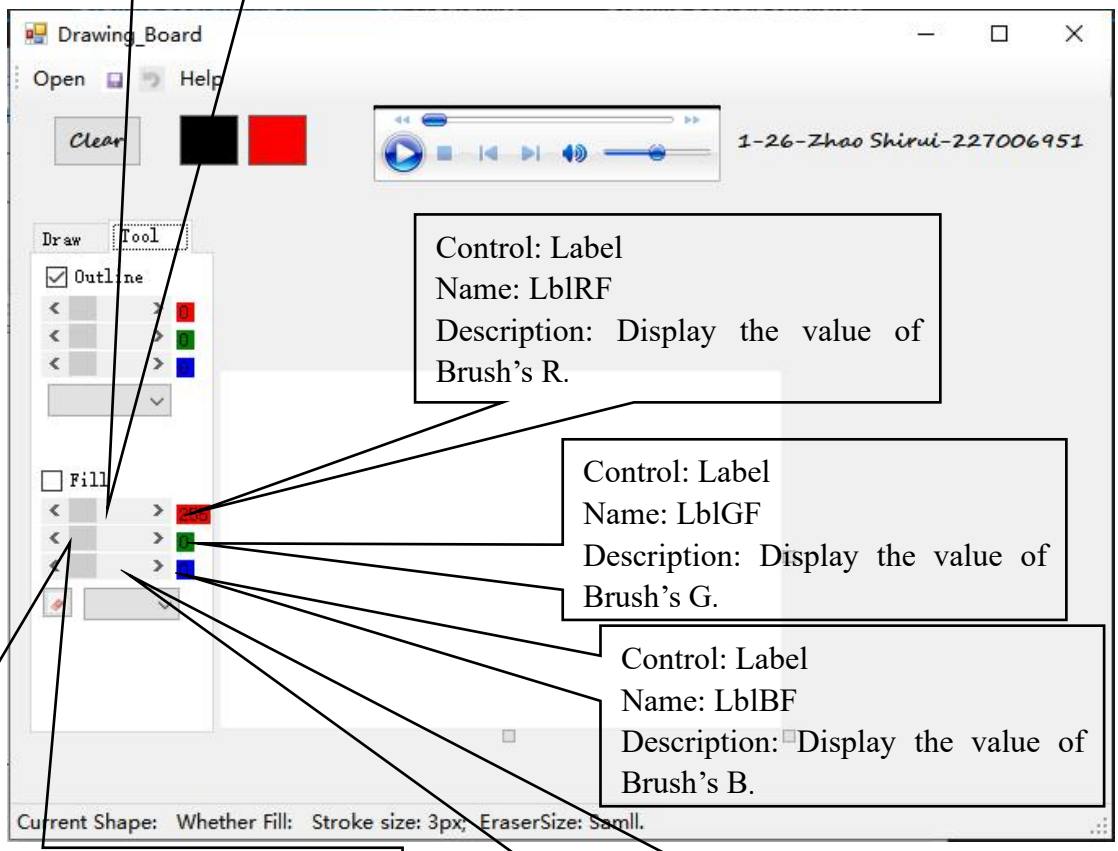
Control: Label
Name: LblB
Description: Display the value of pen's B.

Control: HscrollBar
Name: HsbG
Description: Drag to determine the G value of the pen.

Control: HscrollBar
Name: HsbB
Description: Drag to determine the B value of the pen.

Control: CheckBox
Name: CbxOutline
Description: Click to determine whether to color the drawing.

Control: HscrollBar
Name: HsbRF
Description: Drag to determine the R value of the brush.



Control: Label
Name: LblRF
Description: Display the value of Brush's R.

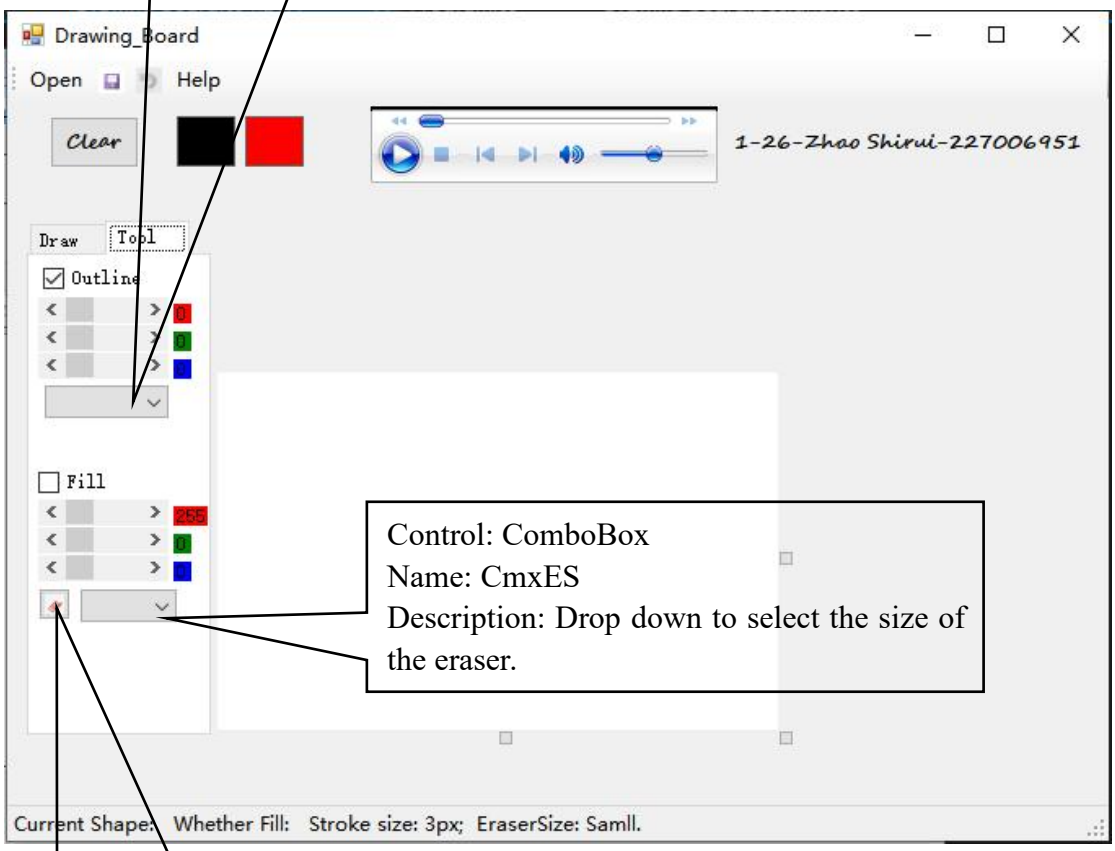
Control: Label
Name: LblGF
Description: Display the value of Brush's G.

Control: Label
Name: LblBF
Description: Display the value of Brush's B.

Control: HscrollBar
Name: HsbGF
Description: Drag to determine the G value of the brush.

Control: HscrollBar
Name: HsbBF
Description: Drag to determine the B value of the brush.

Control: ComboBox
Name: CmxThickness
Description: Drop down to select the size of the pen.



Control: ComboBox
Name: CmxES
Description: Drop down to select the size of the eraser.

Control: Button
Name: BtnEraser
Description: Click to use eraser on the drawing board.

3. Draw Program Model Code Listing

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;



namespace _1_26_ZhaoShiruiOutcome3
{
    /// <summary>
    /// Programer: Zhao Shirui.
    /// Create Date:2022-05-25.
    /// Last Up Date:2022-06-08
    /// Description: This is a Drawing_Board Class.
    ///                 Users can draw straight lines, curves, isosceles triangles,
    ///                 right angle triangles, rectangles
    ///                 and ellipses in this interface (press and hold shift
    ///                 to draw circles at the same time).
    ///                 The drawing tool can erase the content
    ///                 with an eraser or clear the screen with one click.
    ///                 The drawing tool can select the color
    ///                 and fill color of different strokes, and can also adjust the size
of strokes.
    ///                 This class can also perform undo operations, and open and
save pictures.
    /// </summary>

    //Description:
    //Call the function to initialize the form control in the Drawing_form class.
    public partial class FrmDrawing_Board : Form
    {
        public FrmDrawing_Board()
        {
            InitializeComponent();

        }
        //Global variable

        //Boolean variable,In order,
        //it controls whether the canvas size changes,
```

```
            //whether the user can draw, and whether help clauses are visible.
            bool change = false, draw = false, flag1 = true, flag2 = false;
            //Integer variable, which controls the moving direction, vertical coordinate,
            //horizontal coordinate and drawing type by times.
            int flag, top, left, type = 0;
            //The abstract base class of a graph. Apply for an abstract class named G for
drawing.
            Graphics g;
            //Pen class, used to control the size and color of strokes when drawing.
            Pen pen, eraser;
            // Point class, used to determine the start and end coordinates when drawing.
            Point startpt, endpt;
            //The bitmap class is used to display drawing shapes, tracks, and undo
operations.
            Bitmap im, imtemp, now;
            //List class, used to store points of curve track.
            List<Point> plist;
            //The brush class, which controls the color of the filled drawing,
            Brush brush;
            //The stack class stores bitmaps and is used to undo operations.
            Stack<Bitmap> undo;
            //String class, used for the path and name of the music file.
            string home, mp3File;



            //The sketchpad move function.
            //When the sketchpad moves,
            //the buttons controlling the movement of the sketchpad will move together.
            private void DBLocation()
            {
                //Get the horizontal and vertical coordinates of the drawing board.
                top = PcbDB.Location.Y;
                left = PcbDB.Location.X;
                //Control is located at the bottom of the middle of the drawing board.
                BtnNSC.Location = new Point(left + PcbDB.Width / 2, top +
PcbDB.Height);
                //Control is located in the lower right corner of the palette.
                BtnNWSEC.Location = new Point(left + PcbDB.Width, top +
PcbDB.Height);
                //Control is located in the middle of the left side of the palette.
                BtnWEC.Location = new Point(left + PcbDB.Width, top +
PcbDB.Height / 2);
            }
```

```
        //Drawing board size function.
        //When the drawing board size changes, the drawing area also changes.
        private void DBSize()
        {
            //Instantiate a bitmap and graphics for canvas size change.
            Bitmap bit = new Bitmap(PcbDB.Width, PcbDB.Height);
            Graphics gr = Graphics.FromImage(bit);
            gr.FillRectangle(new SolidBrush(Color.White), 0, 0, PcbDB.Width,
PcbDB.Height);
            gr.DrawImage(im, 0, 0);
            gr = PcbDB.CreateGraphics();
            gr.DrawImage(bit, 0, 0);
            im = bit;
        }

    //Sketchpad move function.
        //When the palette moves along the we, NWSE and NS directions, it will be
repositioned.
        //Parameter: the shaping variable flag is used to control which direction to
move.
        private void DBMove(int flag)
        {
            //Instantiate a point. The position of the point is the position of the
mouse.
            Point p = PointToClient(Control.MousePosition);

            switch (flag)
            {
                case 1:
                    PcbDB.Width = p.X - left;
                    DBSize();
                    break;
                case 2:
                    PcbDB.Width = p.X - left;
                    PcbDB.Height = p.Y - top;
                    DBSize();
                    break;
                case 3:
                    PcbDB.Height = p.Y - top;
                    DBSize();
                    break;

            }
```

```
        }
        //Color function. When the color changes, the RGB value will be
represented.
        private void color()
        {
            LblR.Text = pen.Color.R.ToString();
            LblG.Text = pen.Color.G.ToString();
            LblB.Text = pen.Color.B.ToString();
            LblColor1.BackColor = pen.Color;

        }

        //Adjust the horizontal scroll bar function.
        //When the brush color changes, the value of the horizontal scroll bar also
changes.
        private void HScolor()
        {
            HsbR.Value = pen.Color.R;
            HsbG.Value = pen.Color.G;
            HsbB.Value = pen.Color.B;

        }

        //Description:
        //This is Drawing_Board load event.
        // First, determine the position of the control panel moving control.
        //Secondly, initialize the created object.
        //Then assign a value to the music file path
        //to display the initial color of the brush and fill.
        //Finally, set the undo control to unavailable.

        private void Drawing_Board_Load(object sender, EventArgs e)
        {
            DBLocation();
            im = new Bitmap(PcbDB.Width, PcbDB.Height);
            g = Graphics.FromImage(im);
            g.Clear(Color.White);
            g.SmoothingMode = SmoothingMode.AntiAlias;
            pen = new Pen(Color.Black, 3);
            pen.StartCap = LineCap.Round;
            pen.EndCap = LineCap.Round;
            pen.LineJoin = LineJoin.Round;
            plist = new List<Point>();
            brush = new SolidBrush(Color.Red);
```

```
        eraser = new Pen(Color.White, 5);
        undo = new Stack<Bitmap>();
        home = Application.StartupPath;
        mp3File = home + @"/reverse.flac";
        WmpPlayer.URL = mp3File;
        WmpPlayer.Ctlcontrols.stop();
        RtbHelp.Visible = false;
        color();
        LblColor2.BackColor = Color.Red;


        if (undo.Count == 0)
        {
            TsbUndo.Enabled = false;
        }

}
//Description:
//Mouse hover event: when the mouse hovers over this control,
//the style of the mouse will be changed to the style in the WE direction.
private void BtnWEC_MouseHover(object sender, EventArgs e)
{
    this.Cursor = Cursors.SizeWE;
}
//Description:
//Mouse leave event.
//When the mouse leaves the control, the mouse returns to the original style.
private void BtnWEC_MouseLeave(object sender, EventArgs e)
{
    this.Cursor = Cursors.Arrow;
}
//Description:
//Mouse down event: when the left mouse button is pressed,
//the boolean variable change is changed to true
//and the integer variable flag is changed to 1.
private void BtnWEC_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        change = true;
        flag = 1;
    }
}
//Description:
```

```
        //Mouse down event: when the left mouse button is pressed,
        //the boolean variable change is changed to true
        //and the integer variable flag is changed to 2.
        private void BtnNWSEC_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                change = true;
                flag = 2;
            }
        }
        //Description:
        //Mouse hover event: when the mouse hovers over this control,
        //the style of the mouse will be changed to the style in the NWSE direction.
        private void BtnNWSEC_MouseHover(object sender, EventArgs e)
        {
            this.Cursor = Cursors.SizeNWSE;
        }
        //Descripetion:
        //Mouse leave event.
        //When the mouse leaves the control, the mouse returns to the original style.
        private void BtnNWSEC_MouseLeave(object sender, EventArgs e)
        {
            this.Cursor = Cursors.Arrow;
        }
        //Description:
        //The mouse movement event can only be triggered
        //when the left mouse button is pressed and the boolean variable is true.
        //The changed flag is passed to and the control for moving the canvas is
located.
        private void BtnNWSEC_MouseMove(object sender, MouseEventArgs e)
        {
            if (change && e.Button == MouseButtons.Left)
            {
                DBMove(flag);
                DBLocation();
            }
        }
        //Description:
        //The mouse up event.
        //When the mouse is up,
        //the boolean variable that controls the size of the drawing board changes to
false.
        private void BtnNWSEC_MouseUp(object sender, MouseEventArgs e)
```

```
        {
            change = false;
        }
        //Description:
        //Mouse down event: when the left mouse button is pressed,
        //the boolean variable change is changed to true
        //and the integer variable flag is changed to 1.
        private void BtnNSC_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                change = true;
                flag = 3;
            }
        }
        //Description:
        //Mouse hover event: when the mouse hovers over this control,
        //the style of the mouse will be changed to the style in the NS direction.
        private void BtnNSC_MouseHover(object sender, EventArgs e)
        {
            this.Cursor = Cursors.SizeNS;
        }
        //Description:
        //Mouse leave event.
        //When the mouse leaves the control, the mouse returns to the original style.
        private void BtnNSC_MouseLeave(object sender, EventArgs e)
        {
            this.Cursor = Cursors.Arrow;
        }
        //Description:
        //Click the line event, click the button,
        //the value of type will change to 1,
        //and the status bar will prompt that the current drawing mode is to draw a
straight line.
        private void BtnLine_Click(object sender, EventArgs e)
        {
            type = 1;
            TslShape.Text = "Current Shape: Line.";
        }
        //Description:
        //Click the rectangle event, click the button,
        //the value of type will change to 2,
        //and the status bar will prompt that the current drawing mode is to draw a
rectangle.
```

```
        private void BtnRectangle_Click(object sender, EventArgs e)
        {
            type = 2;
            TslShape.Text = "Current Shape: Rectangle.";
        }
        //Descripetion:
        //Click the triangle event, click the button,
        //the value of type will change to 3,
        //and the status bar will prompt that the current drawing mode is to draw a
triangle.
        private void BtnRTriangle_Click(object sender, EventArgs e)
        {
            type = 3;
            TslShape.Text = "Current Shape: Right Triangle.";
        }
       //Description:
        //Click the clear event and click this button. The drawing board will be
cleared.
        //The value of type will change to 0.
        private void BtnClear_Click(object sender, EventArgs e)
        {
            type = 0;
            g.Clear(Color.White);
            PcbDB.BackgroundImage = im;
            PcbDB.Refresh();
        }
        //Description:
        //Sketchpad mouse down event: when the mouse is pressed on the
sketchpad,
        //if the value of type is not 0 and the left mouse button is pressed,
        //Set the boolean variable draw to true,
        //the starting position of the drawing can be recorded,
        //and the bitmap of now can be initialized and stored.
        //If the value of type is 4 or 7,
        //the coordinates of the starting point are also recorded,
        //and the bitmap is initialized and stored.
        private void PcbDB_MouseDown(object sender, MouseEventArgs e)
        {
            if (type != 0 && e.Button == MouseButtons.Left)
            {
                draw = true;
                startpt = e.Location;
                now = new Bitmap(im);
                undo.Push(now);
```

```
            }
            if (type == 4 || type == 7)
            {
                plist.Add(e.Location);
                now = new Bitmap(im);
                undo.Push(now);
            }


        }
        //Description:
        // For the mouse movement event on the drawing board,
        //when the draw value is true,
        //the position of the mouse will be updated in real time each time it is
moved,
        //and a temporary class diagram imtemp will be instantiated at the same
time.
        //If both the brush and fill options are selected, the drawn figure is the filled
figure.
        //If type is not of types 4 and 7, G is instantiated from imtemp.
        //Then, different types of pictures are drawn
        //according to the values of different types (from 1 to 7).
        //Finally, set the image of pcbdb to imtemp and refresh it in real time.
        private void PcbDB_MouseMove(object sender, MouseEventArgs e)
        {
            if (draw)
            {
                endpt = e.Location;
                imtemp = new Bitmap(im);
                if (type != 4 && type != 7)
                {
                    g = Graphics.FromImage(imtemp);

                }
                switch (type)
                {
                    case 1:
                        g.DrawLine(pen, startpt, endpt);
                        break;
                    case 2:
                        if (CbxOutline.Checked)
                        {
                            g.DrawRectangle(pen, startpt.X, startpt.Y, endpt.X
```

```
- startpt.X, endpt.Y - startpt.Y);
                    g.DrawRectangle(pen, endpt.X, endpt.Y, startpt.X -
endpt.X, startpt.Y - endpt.Y);
                    g.DrawRectangle(pen, startpt.X, endpt.Y, endpt.X -
startpt.X, startpt.Y - endpt.Y);
                    g.DrawRectangle(pen, endpt.X, startpt.Y, startpt.X
- endpt.X, endpt.Y - startpt.Y);
                }
                if (CbxFill.Checked)
                {
                    g.FillRectangle(brush, startpt.X, startpt.Y, endpt.X
- startpt.X, endpt.Y - startpt.Y);
                    g.FillRectangle(brush, endpt.X, endpt.Y, startpt.X -
endpt.X, startpt.Y - endpt.Y);
                    g.FillRectangle(brush, startpt.X, endpt.Y, endpt.X -
startpt.X, startpt.Y - endpt.Y);
                    g.FillRectangle(brush, endpt.X, startpt.Y, startpt.X
- endpt.X, endpt.Y - startpt.Y);
                }
                break;
            case 3:
                Point p1 = new Point(startpt.X, startpt.Y);
                Point p2 = new Point(startpt.X, endpt.Y);
                Point p3 = new Point(endpt.X, endpt.Y);
                Point[] parray = { p1, p2, p3 };
                if (CbxOutline.Checked)
                {
                    g.DrawPolygon(pen, parray);
                }
                if (CbxFill.Checked)
                {
                    g.FillPolygon(brush, parray);
                }
                break;
            case 4:
                plist.Add(e.Location);
                g.DrawCurve(pen, plist.ToArray());
                break;
            case 5:
                p1 = new Point(startpt.X, startpt.Y);
                p2 = new Point((endpt.X + startpt.X) / 2, endpt.Y);
                p3 = new Point(endpt.X, startpt.Y);
                Point[] parray1 = { p1, p2, p3 };
                if (CbxOutline.Checked)
```

```
                {
                    g.DrawPolygon(pen, parray1);
                }
                if (CbxFill.Checked)
                {
                    g.FillPolygon(brush, parray1);
                }
                break;
            case 6:
                if (CbxOutline.Checked)
                {
                    if (Control.ModifierKeys == Keys.Shift &&
e.Button == MouseButtons.Left)
                    {
                        g.DrawEllipse(pen, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.X - startpt.X));
                    }
                    else
                        g.DrawEllipse(pen, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.Y - startpt.Y));
                }
                if (CbxFill.Checked)
                {
                    if (Control.ModifierKeys == Keys.Shift &&
e.Button == MouseButtons.Left)
                    {
                        g.FillEllipse(brush, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.X - startpt.X));
                    }
                    else
                        g.FillEllipse(brush, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.Y - startpt.Y));
                }
                break;
            case 7:
                plist.Add(e.Location);
                g.DrawCurve(eraser, plist.ToArray());
                break;

        }

        PcbDB.Image = imtemp;
        PcbDB.Refresh();
```

```
            }
        }
 //Description:
        //Click the isosceles triangle event, click the button,
        //the value of type will change to 5,
        //and the status bar will prompt that the current drawing mode is to draw an
isosceles triangle.
        private void BtnTriangle_Click(object sender, EventArgs e)
        {
            type = 5;
            TslShape.Text = "Current Shape: Isosceles Triangle.";
        }

        //Description:
        //Click the ellipse event, click the button,
        //the value of type will change to 6,
        //and the status bar will prompt that the current drawing mode is to draw an
ellipse.

        private void BtnEllipse_Click(object sender, EventArgs e)
        {
            type = 6;
            TslShape.Text = "Current Shape: Ellipse.";
        }

        //Description:
        //Adjust the horizontal scroll bar value event.
        //When the horizontal scroll bar value changes,
        //the color will be displayed in RGB,
        //and the current pen color and RGB specific value will be prompted.
        private void HsbR_Scroll(object sender, ScrollEventArgs e)
        {
            LblR.Text = HsbR.Value.ToString();
            pen.Color = Color.FromArgb(HsbR.Value, HsbG.Value, HsbB.Value);
            LblColor1.BackColor = Color.FromArgb(HsbR.Value, HsbG.Value,
HsbB.Value);
        }

        //Description:
        //Adjust the horizontal scroll bar value event.
        //When the horizontal scroll bar value changes,
        //the color will be displayed in RGB,
        //and the current pen color and RGB specific value will be prompted.
        private void HsbG_Scroll(object sender, ScrollEventArgs e)
```

```
        {
            LblG.Text = HsbG.Value.ToString();
            pen.Color = Color.FromArgb(HsbR.Value, HsbG.Value, HsbB.Value);
            LblColor1.BackColor = Color.FromArgb(HsbR.Value, HsbG.Value,
HsbB.Value);
        }

        //Description:
        //Adjust the horizontal scroll bar value event.
        //When the horizontal scroll bar value changes,
        //the color will be displayed in RGB,
        //and the current pen color and RGB specific value will be prompted.
        private void HsbB_Scroll(object sender, ScrollEventArgs e)
        {
            LblB.Text = HsbB.Value.ToString();
            pen.Color = Color.FromArgb(HsbR.Value, HsbG.Value, HsbB.Value);
            LblColor1.BackColor = Color.FromArgb(HsbR.Value, HsbG.Value,
HsbB.Value);
        }

        //Description:
        //Adjust the horizontal scroll bar value event.
        //When the horizontal scroll bar value changes,
        //the color will be displayed in RGB,
        //and the current brush color and RGB specific value will be prompted.
        private void HsbRF_Scroll(object sender, ScrollEventArgs e)
        {
            LblRF.Text = HsbRF.Value.ToString();
            brush = new SolidBrush(Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value));
            LblColor2.BackColor = Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value);
        }

        //Description:
        //Adjust the horizontal scroll bar value event.
        //When the horizontal scroll bar value changes,
        //the color will be displayed in RGB,
        //and the current brush color and RGB specific value will be prompted.
        private void HsbGF_Scroll(object sender, ScrollEventArgs e)
        {
            LblGF.Text = HsbGF.Value.ToString();
            brush = new SolidBrush(Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value));
```

```
            LblColor2.BackColor = Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value);
        }

        //Description:
        //Adjust the horizontal scroll bar value event.
        //When the horizontal scroll bar value changes,
        //the color will be displayed in RGB,
        //and the current brush color and RGB specific value will be prompted.
        private void HsbBF_Scroll(object sender, ScrollEventArgs e)
        {
            LblBF.Text = HsbBF.Value.ToString();
            brush = new SolidBrush(Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value));
            LblColor2.BackColor = Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value);
        }

        //Description:
        //Stroke size adjustment event.
        //This is a combo box selection box.
        //The user can select different stroke sizes,
        //and the status prompt bar will prompt the current stroke size.
        private void CmxThickness_SelectedIndexChanged(object sender,
EventArgs e)
        {

            switch (CmxThickness.Text)
            {
                case "1px":
                    pen.Width = 1;
                    TslStrokesize.Text = "1 px";
                    break;
                case "3px":
                    pen.Width = 3;
                    TslStrokesize.Text = "3 px";
                    break;
                case "5px":
                    pen.Width = 5;
                    TslStrokesize.Text = "5 px";
                    break;
                case "8px":
                    pen.Width = 8;
                    TslStrokesize.Text = "8 px";
```

```
                break;
            case "10px":
                pen.Width = 10;
                TslStrokesize.Text = "10 px";
                break;
            case "15px":
                pen.Width = 15;
                TslStrokesize.Text = "15 px";
                break;
            case "20px":
                pen.Width = 20;
                TslStrokesize.Text = "20 px";
                break;
        }
    }

    //Description:
    //Rubber sizing event.
    //This is a combo box selection box.
    //The user can select different stroke sizes,
    //and the status prompt bar will prompt the current rubber size.
    private void CmxES_SelectedIndexChanged(object sender, EventArgs e)
    {
        switch (CmxES.Text)
        {
            case "Small":
                eraser.Width = 5;
                TslEraserSize.Text = "EraserSize: Small.";
                break;
            case "Medium":
                eraser.Width = 10;
                TslEraserSize.Text = "EraserSize: Medium.";
                break;
            case "Big":
                eraser.Width = 20;
                TslEraserSize.Text = "EraserSize: Big.";
                break;
        }

    }

    //Description:
    //Quickly select the brush color event.
    //Click this button, and the pen color will switch to Red,
```

```
//and the interface will display the current pen color.
private void BtnRed_Click(object sender, EventArgs e)
{
    pen.Color = Color.Red;
    color();
    HScolor();

}

//Description:
//Quickly select the brush color event.
//Click this button, and the pen color will switch to Pink,
//and the interface will display the current pen color.
private void BtnPink_Click(object sender, EventArgs e)
{
    pen.Color = Color.Pink;
    color();
    HScolor();
}

//Description:
//Quickly select the brush color event.
//Click this button, and the pen color will switch to Black/
//and the interface will display the current pen color.
private void BtnBlack_Click(object sender, EventArgs e)
{
    pen.Color = Color.Black;
    color();
    HScolor();
}

//Description:
//Quickly select the brush color event.
//Click this button, and the pen color will switch to Blue,
//and the interface will display the current pen color.
private void BtnBlue_Click(object sender, EventArgs e)
{
    pen.Color = Color.Blue;
    color();
    HScolor();
}

//Description:
//Quickly select the brush color event.
```

```
//Click this button, and the pen color will switch to Green,
//and the interface will display the current pen color.
private void BtnGreen_Click(object sender, EventArgs e)
{
    pen.Color = Color.Green;
    color();
    HScolor();
}
//Description:
//Quickly select the brush color event.
//Click this button, and the pen color will switch to Yellow,
//and the interface will display the current pen color.
private void BtnYellow_Click(object sender, EventArgs e)
{
    pen.Color = Color.Yellow;
    color();
    HScolor();
}
//Description:
//Quickly select the brush color event,
//click this button, the brush color will switch to Red,
//and the interface will display the current brush color.
private void BtnRedF_Click(object sender, EventArgs e)
{
    brush = new SolidBrush(Color.Red);
    LblColor2.BackColor = (Color.Red);
    HsbRF.Value = Color.Red.R;
    HsbGF.Value = Color.Red.G;
    HsbBF.Value = Color.Red.B;
}

//Description:
//Quickly select the brush color event,
//click this button, the brush color will switch to Pink,
//and the interface will display the current brush color.
private void BtnPinkF_Click(object sender, EventArgs e)
{
    brush = new SolidBrush(Color.Pink);
    LblColor2.BackColor = (Color.Pink);
    HsbRF.Value = Color.Pink.R;
    HsbGF.Value = Color.Pink.G;
    HsbBF.Value = Color.Pink.B;
}
```

```
//Description:
//Quickly select the brush color event,
//click this button, the brush color will switch to Black,
//and the interface will display the current brush color.
private void BtnBlackF_Click(object sender, EventArgs e)
{
    brush = new SolidBrush(Color.Black);
    LblColor2.BackColor = (Color.Black);
    HsbRF.Value = Color.Black.R;
    HsbGF.Value = Color.Black.G;
    HsbBF.Value = Color.Black.B;
}

//Description:
//Quickly select the brush color event,
//click this button, the brush color will switch to Blue,
//and the interface will display the current brush color.
private void BtnBlueF_Click(object sender, EventArgs e)
{
    brush = new SolidBrush(Color.Blue);
    LblColor2.BackColor = (Color.Blue);
    HsbRF.Value = Color.Blue.R;
    HsbGF.Value = Color.Blue.G;
    HsbBF.Value = Color.Blue.B;
}

//Description:
//Quickly select the brush color event,
//click this button, the brush color will switch to Green,
//and the interface will display the current brush color.
private void BtnGreenF_Click(object sender, EventArgs e)
{
    brush = new SolidBrush(Color.Green);
    LblColor2.BackColor = (Color.Green);
    HsbRF.Value = Color.Green.R;
    HsbGF.Value = Color.Green.G;
    HsbBF.Value = Color.Green.B;
}

//Description:
//Quickly select the brush color event,
//click this button, the brush color will switch to Yellow,
//and the interface will display the current brush color.
private void BtnYellowF_Click(object sender, EventArgs e)
```

```
        {
            brush = new SolidBrush(Color.Yellow);
            LblColor2.BackColor = (Color.Yellow);
            HsbRF.Value = Color.Yellow.R;
            HsbGF.Value = Color.Yellow.G;
            HsbBF.Value = Color.Yellow.B;
        }

        //Description:
        //Save the current picture event.
        //When you click this button,
        //if the picture on the canvas is blank,
        //it will not be saved. If it is a picture with drawing traces,
        //it will be saved and saved in a specific format.
        private void TsbSave_Click(object sender, EventArgs e)
        {
            PcbDB.Refresh();
            for (int i = 0; i < im.Width; i++)
            {
                for (int j = 0; j < im.Height; j++)
                {
                    if ((im.GetPixel(i, j).A + im.GetPixel(i, j).R + im.GetPixel(i,
j).G
                        + im.GetPixel(i, j).B) != 1020)
                    {
                        SfdSave.Filter = "BMP file|*.bmp|JPG file|*.jpg|PNG
file|*.png|" +
                     "TIFF file|*.tif|GIF file|*.gif";
                        SfdSave.Title = "Save this file";

                        if (SfdSave.ShowDialog() == DialogResult.OK)
                        {
                            string filename = SfdSave.FileName;
                            string filetext = filename.Remove(0,
filename.Length - 3);
                            switch (filetext)
                            {
                                case "bmp":
                                    im.Save(filename,
System.Drawing.Imaging.ImageFormat.Bmp);
                                    break;
                                case "jpg":
                                    im.Save(filename,
System.Drawing.Imaging.ImageFormat.Jpeg);
```

```
                                        break;
                                    case "png":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Png);
                                        break;
                                    case "tif":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Tiff);
                                        break;
                                    case "gif":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Gif);
                                        break;
                                }
                                goto flag;

                            }
                            else
                                goto flag;
                        }

                    }
                }
            flag:;

            }
            //Description:
            //Whether to select events for filling.
            //If you are sure to select filling,
            //the status bar will prompt that it has been filled.
            //If no fill is selected, the status bar indicates that it is not filled.
            private void CbxFill_CheckedChanged(object sender, EventArgs e)
            {
                if (CbxFill.Checked)
                {
                    TslWFill.Text = "Whether Fill: Yes. ";
                }
                if (CbxFill.Checked == false)
                {
                    TslWFill.Text = "Whether Fill: No. ";
                }
            }
            //Description:
            //Event of music player status change.
```

```
//If the current music is played,
//the music player will repeat the second time after 1 second.
private void WmpPlayer_StatusChange(object sender, EventArgs e)
{
    if ((int)WmpPlayer.playState == 1)
    {
        System.Threading.Thread.Sleep(1000);
        WmpPlayer.Ctlcontrols.play();
    }
}

//Description:
//Click the help event.
//When you click this button,
//a message prompt window will be displayed on the drawing board
//to prompt the user how to use the drawing tool.
//Click the drawing board again and the prompt window will disappear.
private void TsbHelp_Click(object sender, EventArgs e)
{

    if (flag1)
    {
        RtbHelp.Visible = true;
        flag1 = false;
        flag2 = true;
    }
    else if (flag2)
    {
        RtbHelp.Visible = false;
        flag1 = true;
        flag2 = false;
    }

}
//Description:
//Picture saving event: when the user clicks this button,
//if a picture already exists on the palette,
//the user will be prompted to save the existing picture first.
//If yes is selected, the current picture will be saved first,
//and then a new picture will be opened.
//If you click No, a new picture will be opened directly.
private void TsbOpen_Click(object sender, EventArgs e)
{
    for (int i = 0; i < im.Width; i++)
```

```
                {
                    for (int j = 0; j < im.Height; j++)
                    {
                        if ((im.GetPixel(i, j).A + im.GetPixel(i, j).R + im.GetPixel(i,
j).G
                            + im.GetPixel(i, j).B) != 1020)
                        {
                            if (MessageBox.Show("Are you sure to save the
picture?", "Save file",
                                MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
                            {
                                TsbSave_Click(sender, e);
                                goto flag;
                            }
                            else
                            {
                                goto flag;
                            }
                        }
                    }
                }
            flag:
                OfdOpen.Filter = "BMP file|*.bmp|JPG file|*.jpg|PNG file|*.png|" +
                    "TIFF file|*.tif|GIF file|*.gif";
                OfdOpen.Title = "Open a picture";
                if (OfdOpen.ShowDialog() == DialogResult.OK)
                {
                    string filename = OfdOpen.FileName;
                    Image image = Image.FromFile(filename);
                    g.Clear(System.Drawing.Color.White);
                    g.DrawImage(image, 0, 0);
                    PcbDB.Image = im;
                    PcbDB.Refresh();
                }
            }
            //Description:
            //Trace cancellation event.
            //Click this button to cancel the previous drawing process.
            //When there is no picture on the current screen,
            //clicking this button will no longer be available.
            private void TsbUndo_Click(object sender, EventArgs e)
            {
                if (undo.Count == 0)
```

```
            {
                TsbUndo.Enabled = false;
            }
            if (undo.Count != 0)
            {
                TsbUndo.Enabled = true;
                im = undo.Pop();
                g = Graphics.FromImage(im);
                PcbDB.Image = im;
                PcbDB.Refresh();
            }


        }

        //Descripetion:
        //Click the eraser event, click the button,
        //the value of type will change to 7,
        //and the status bar will prompt that the current drawing mode is to use
eraser.
        private void BtnEraser_Click(object sender, EventArgs e)
        {
            type = 7;
            TslShape.Text = "Current Shape: Eraser.";
        }

        //Descripetion:
        //Click the curve event, click the button,
        //the value of type will change to 4,
        //and the status bar will prompt that the current drawing mode is to draw the
curve.
        private void btnCurve_Click(object sender, EventArgs e)
        {
            type = 4;
            TslShape.Text = "Current Shape: Curve.";
        }
       //Description:
        //Pcbdb mouse up event:
        //when the mouse is up,
        //the position of the mouse is automatically recorded,
        //and according to the drawing trace,
        //an identical figure is drawn in IM and stored in the pcbdb image.
        //When the left mouse button is raised, the boolean variable draw will
become false.
```

```
        private void PcbDB_MouseUp(object sender, MouseEventArgs e)
        {
            if (draw)
            {
                endpt = e.Location;
                g = Graphics.FromImage(im);

                switch (type)
                {
                    case 1:
                        g.DrawLine(pen, startpt, endpt);
                        break;
                    case 2:
                        if (CbxOutline.Checked)
                        {
                            g.DrawRectangle(pen, startpt.X, startpt.Y, endpt.X
- startpt.X, endpt.Y - startpt.Y);
                            g.DrawRectangle(pen, endpt.X, endpt.Y, startpt.X -
endpt.X, startpt.Y - endpt.Y);
                            g.DrawRectangle(pen, startpt.X, endpt.Y, endpt.X -
startpt.X, startpt.Y - endpt.Y);
                            g.DrawRectangle(pen, endpt.X, startpt.Y, startpt.X
- endpt.X, endpt.Y - startpt.Y);
                        }
                        if (CbxFill.Checked)
                        {
                            g.FillRectangle(brush, startpt.X, startpt.Y, endpt.X
- startpt.X, endpt.Y - startpt.Y);
                            g.FillRectangle(brush, endpt.X, endpt.Y, startpt.X -
endpt.X, startpt.Y - endpt.Y);
                            g.FillRectangle(brush, startpt.X, endpt.Y, endpt.X -
startpt.X, startpt.Y - endpt.Y);
                            g.FillRectangle(brush, endpt.X, startpt.Y, startpt.X
- endpt.X, endpt.Y - startpt.Y);
                        }
                        break;
                    case 3:
                        Point p1 = new Point(startpt.X, startpt.Y);
                        Point p2 = new Point(startpt.X, endpt.Y);
                        Point p3 = new Point(endpt.X, endpt.Y);
                        Point[] parray = { p1, p2, p3 };
                        if (CbxOutline.Checked)
                        {
                            g.DrawPolygon(pen, parray);
```

```
                }
                if (CbxFill.Checked)
                {
                    g.FillPolygon(brush, parray);
                }
                break;
            case 4:
                plist.Clear();
                break;
            case 5:
                p1 = new Point(startpt.X, startpt.Y);
                p2 = new Point((endpt.X + startpt.X) / 2, endpt.Y);
                p3 = new Point(endpt.X, startpt.Y);
                Point[] parray1 = { p1, p2, p3 };
                if (CbxOutline.Checked)
                {
                    g.DrawPolygon(pen, parray1);
                }
                if (CbxFill.Checked)
                {
                    g.FillPolygon(brush, parray1);
                }
                break;
            case 6:
                if (CbxOutline.Checked)
                {
                    if (Control.ModifierKeys == Keys.Shift &&
e.Button == MouseButtons.Left)
                    {
                        g.DrawEllipse(pen, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.X - startpt.X));
                    }
                    else
                        g.DrawEllipse(pen, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.Y - startpt.Y));
                }
                if (CbxFill.Checked)
                {
                    if (Control.ModifierKeys == Keys.Shift &&
e.Button == MouseButtons.Left)
                    {
                        g.FillEllipse(brush, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.X - startpt.X));
                    }
```

```
                                    else
                                        g.FillEllipse(brush, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.Y - startpt.Y));
                                }
                                break;
                            case 7:
                                plist.Clear();
                                break;
                        }

                        PcbDB.Image = im;
                        PcbDB.Refresh();
                        draw = false;
                        TsbUndo.Enabled = true;
                    }
                }

                //Descripetion:
                //The mouse movement event can only be triggered
                //when the left mouse button is pressed and the boolean variable is true.
                //The changed flag is passed to and the control for moving the canvas is
located.
                private void BtnNSC_MouseMove(object sender, MouseEventArgs e)
                {
                    if (change && e.Button == MouseButtons.Left)
                    {
                        DBMove(flag);
                        DBLocation();
                    }
                }

                //Descripetion:
                //The mouse up event.
                //When the mouse is up,
                //the boolean variable that controls the size of the drawing board changes to
false.
                private void BtnNSC_MouseUp(object sender, MouseEventArgs e)
                {
                    change = false;
                }

                //Descripetion:
                //The mouse up event.
                //When the mouse is up,
```

```
        //the boolean variable that controls the size of the drawing board changes to
false.
        private void BtnWEC_MouseUp(object sender, MouseEventArgs e)
        {
            change = false;

        }

        //Descripetion:
        //The mouse movement event can only be triggered
        //when the left mouse button is pressed and the boolean variable is true.
        //The changed flag is passed to and the control for moving the canvas is
located.
        private void BtnWEC_MouseMove(object sender, MouseEventArgs e)
        {
            if (change && e.Button == MouseButtons.Left)
            {
                DBMove(flag);
                DBLocation();
            }
        }
    }


}
```

4.Highlight Code Listing

Class Name: FrmDrawing_Board

4.1 Public Variables In Class

```
 //Global variable

        //Boolean variable,In order,
        //it controls whether the canvas size changes,
        //whether the user can draw, and whether help clauses are visible.
        bool change = false, draw = false, flag1 = true, flag2 = false;
        //Integer variable, which controls the moving direction, vertical coordinate,
        //horizontal coordinate and drawing type by times.
        int flag, top, left, type = 0;
        //The abstract base class of a graph. Apply for an abstract class named G for
drawing.
        Graphics g;
        //Pen class, used to control the size and color of strokes when drawing.
        Pen pen, eraser;
        // Point class, used to determine the start and end coordinates when drawing.
        Point startpt, endpt;
        //The bitmap class is used to display drawing shapes, tracks, and undo
operations.
        Bitmap im, imtemp, now;
        //List class, used to store points of curve track.
        List<Point> plist;
        //The brush class, which controls the color of the filled drawing,
        Brush brush;
        //The stack class stores bitmaps and is used to undo operations.
        Stack<Bitmap> undo;
        //String class, used for the path and name of the music file.
        string home, mp3File;
```

4.2 Class's startup load function load event

```
//Description:
        //This is Drawing_Board load event.
        // First, determine the position of the control panel moving control.
        //Secondly, initialize the created object.
```

```
        //Then assign a value to the music file path
        //to display the initial color of the brush and fill.
        //Finally, set the undo control to unavailable.

        private void Drawing_Board_Load(object sender, EventArgs e)
        {
            DBLocation();
            im = new Bitmap(PcbDB.Width, PcbDB.Height);
            g = Graphics.FromImage(im);
            g.Clear(Color.White);
            g.SmoothingMode = SmoothingMode.AntiAlias;
            pen = new Pen(Color.Black, 3);
            pen.StartCap = LineCap.Round;
            pen.EndCap = LineCap.Round;
            pen.LineJoin = LineJoin.Round;
            plist = new List<Point>();
            brush = new SolidBrush(Color.Red);
            eraser = new Pen(Color.White, 5);
            undo = new Stack<Bitmap>();
            home = Application.StartupPath;
            mp3File = home + @"/reverse.flac";
            WmpPlayer.URL = mp3File;
            WmpPlayer.Ctlcontrols.stop();
            RtbHelp.Visible = false;
            color();
            LblColor2.BackColor = Color.Red;


            if (undo.Count == 0)
            {
                TsbUndo.Enabled = false;
            }

        }
```

4.3 Methods written by individuals.

4.3.1 Method DBLocation()

```
    //The sketchpad move function.
        //When the sketchpad moves,
        //the buttons controlling the movement of the sketchpad will move together.
        private void DBLocation()
```

```
        {
            //Get the horizontal and vertical coordinates of the drawing board.
            top = PcbDB.Location.Y;
            left = PcbDB.Location.X;
            //Control is located at the bottom of the middle of the drawing board.
            BtnNSC.Location = new Point(left + PcbDB.Width / 2, top +
PcbDB.Height);
            //Control is located in the lower right corner of the palette.
            BtnNWSEC.Location = new Point(left + PcbDB.Width, top +
PcbDB.Height);
            //Control is located in the middle of the left side of the palette.
            BtnWEC.Location = new Point(left + PcbDB.Width, top +
PcbDB.Height / 2);
        }
```

4.3.2 Method DBSize()

```
        //Drawing board size function.
        //When the drawing board size changes, the drawing area also changes.
        private void DBSize()
        {
            //Instantiate a bitmap and graphics for canvas size change.
            Bitmap bit = new Bitmap(PcbDB.Width, PcbDB.Height);
            Graphics gr = Graphics.FromImage(bit);
            gr.FillRectangle(new SolidBrush(Color.White), 0, 0, PcbDB.Width,
PcbDB.Height);
            gr.DrawImage(im, 0, 0);
            gr = PcbDB.CreateGraphics();
            gr.DrawImage(bit, 0, 0);
            im = bit;
        }
```

4.3.3 Method DBMove(int flag)

```
        //Sketchpad move function.
        //When the palette moves along the we, NWSE and NS directions, it will be
repositioned.
        //Parameter: the shaping variable flag is used to control which direction to
move.
        private void DBMove(int flag)
        {
            //Instantiate a point. The position of the point is the position of the
```

```
mouse.
                Point p = PointToClient(Control.MousePosition);

                switch (flag)
                {
                    case 1:
                        PcbDB.Width = p.X - left;
                        DBSize(); Reference method 4.3.2
                        break;
                    case 2:
                        PcbDB.Width = p.X - left;
                        PcbDB.Height = p.Y - top;
                        DBSize();Reference method 4.3.2
                        break;
                    case 3:
                        PcbDB.Height = p.Y - top;
                        DBSize();Reference method 4.3.2
                        break;

                }

            }
```

4.3.4 Method color()

```
            //Color function. When the color changes, the RGB value will be
represented.
            private void color()
            {
                LblR.Text = pen.Color.R.ToString();
                LblG.Text = pen.Color.G.ToString();
                LblB.Text = pen.Color.B.ToString();
                LblColor1.BackColor = pen.Color;

            }
```

4.3.5 Method HScolor()

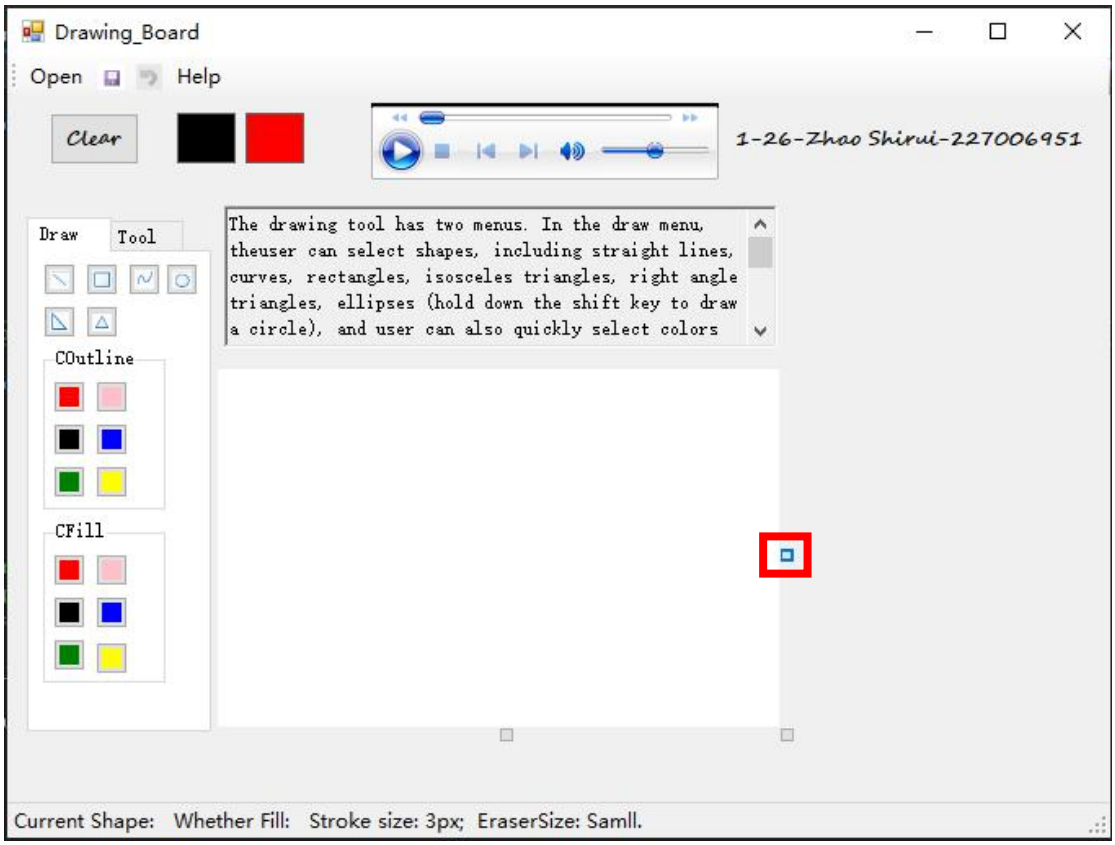
```
            //Adjust the horizontal scroll bar function.
            //When the brush color changes, the value of the horizontal scroll bar also
changes.
            private void HScolor()
```

```
{
    HsbR.Value = pen.Color.R;
    HsbG.Value = pen.Color.G;
    HsbB.Value = pen.Color.B;

}
```

4.4 System Generated Event Code.

4.4.1 BtnWEC_MouseHover(object sender, EventArgs e)

BtnWEC_MouseHover(object sender, EventArgs e) is a mouse hover event. The main function of the mouse over event is that when the mouse hovers over this button, the mouse will become an arrow pointing towards WE.

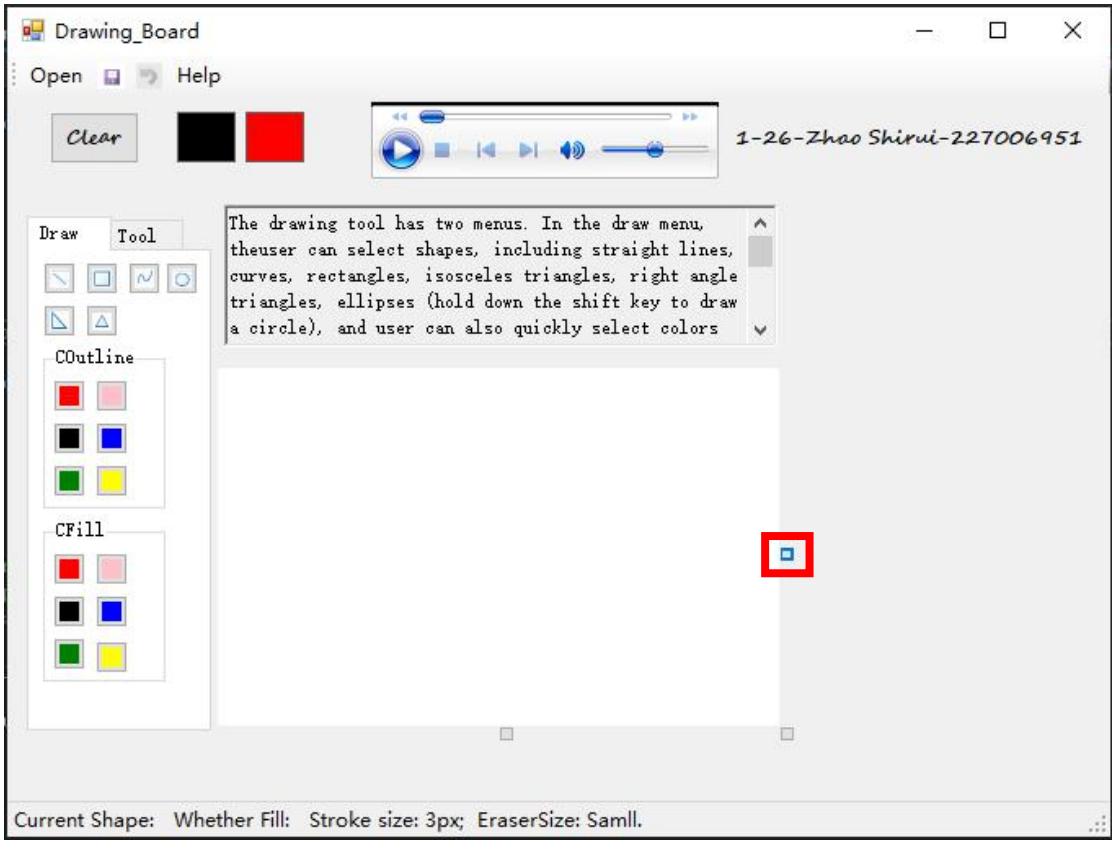



```
//Description:
        //Mouse hover event: when the mouse hovers over this control,
        //the style of the mouse will be changed to the style in the WE direction.
        private void BtnWEC_MouseHover(object sender, EventArgs e)
        {
```

```
        this.Cursor = Cursors.SizeWE;
    }
```

4.4.2 BtnWEC_MouseLeave(object sender, EventArgs e)

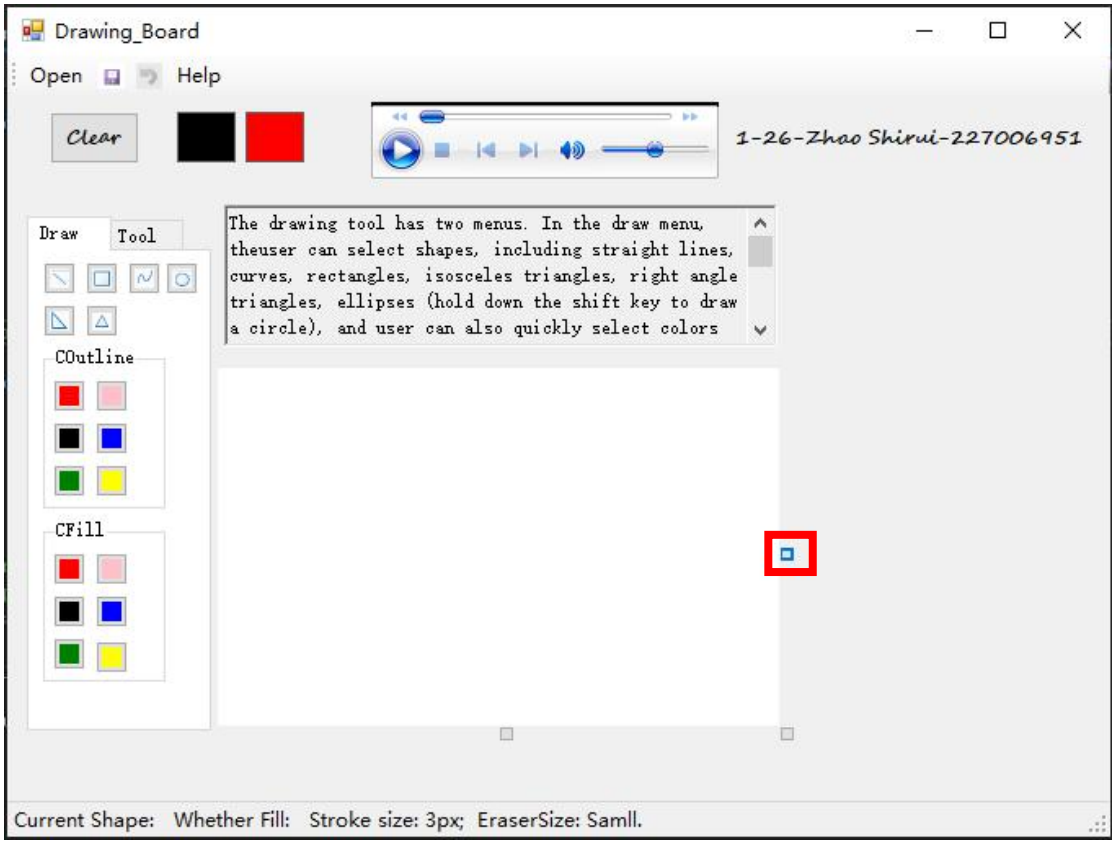
BtnWEC_MouseLeave(object sender, EventArgs e) is a mouse leave event. The main function of the mouse leave event is that when the mouse leaves this button, the mouse will change to its original shape.




```
//Description:
        //Mouse leave event.
        //When the mouse leaves the control, the mouse returns to the original style.
        private void BtnWEC_MouseLeave(object sender, EventArgs e)
        {
            this.Cursor = Cursors.Arrow;
        }
```

4.4.3 BtnWEC_MouseDown(object sender, MouseEventArgs e)

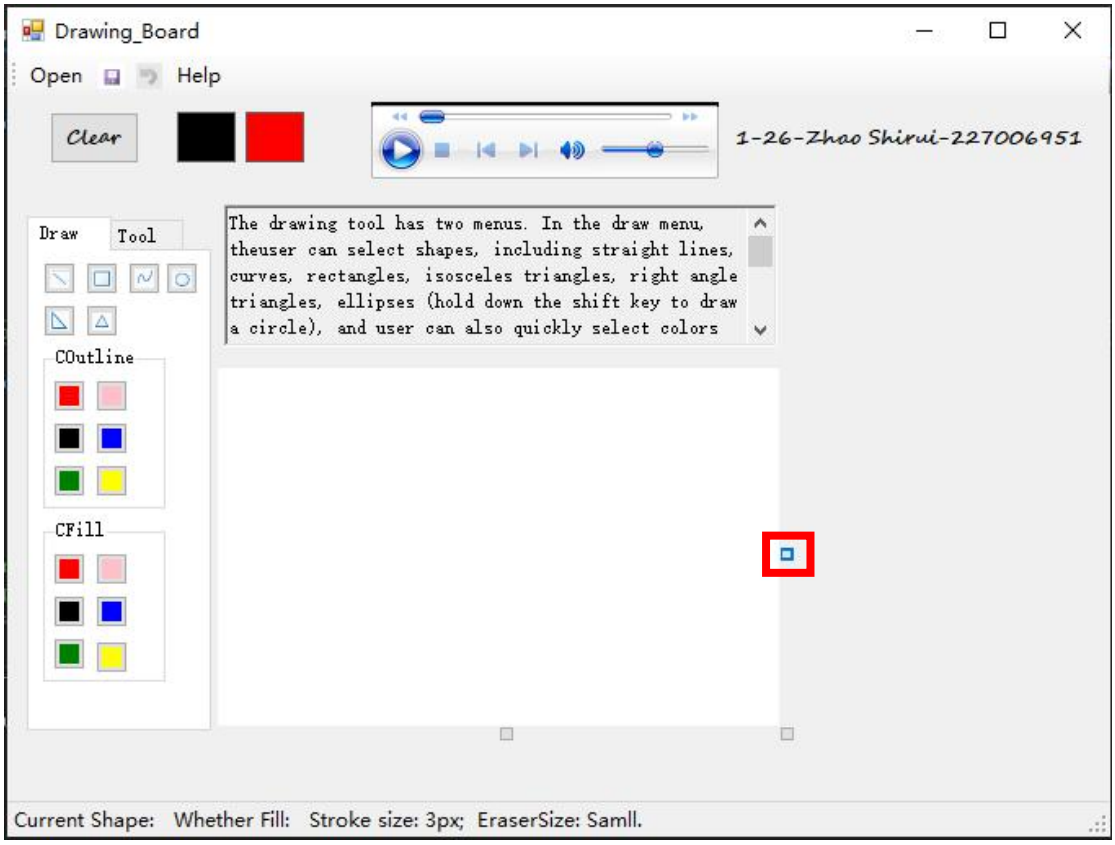
BtnWEC_MouseDown(object sender, MouseEventArgs e) is a mouse down event. When the left mouse button is pressed, the boolean variable change becomes true and the control flag becomes 1.




```
//Description:
        //Mouse down event: when the left mouse button is pressed,
        //the boolean variable change is changed to true
        //and the integer variable flag is changed to 1.
        private void BtnWEC_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                change = true;
                flag = 1;
            }
        }
```

4.4.4 BtnWEC_MouseMove(object sender, MouseEventArgs e)

BtnWEC_ MouseMove (object sender, mouseeventargs E) is a mouse movement event. When change is true and the left mouse button is clicked, the drawing board can be enlarged or reduced in the direction controlled by this button according to the incoming flag.

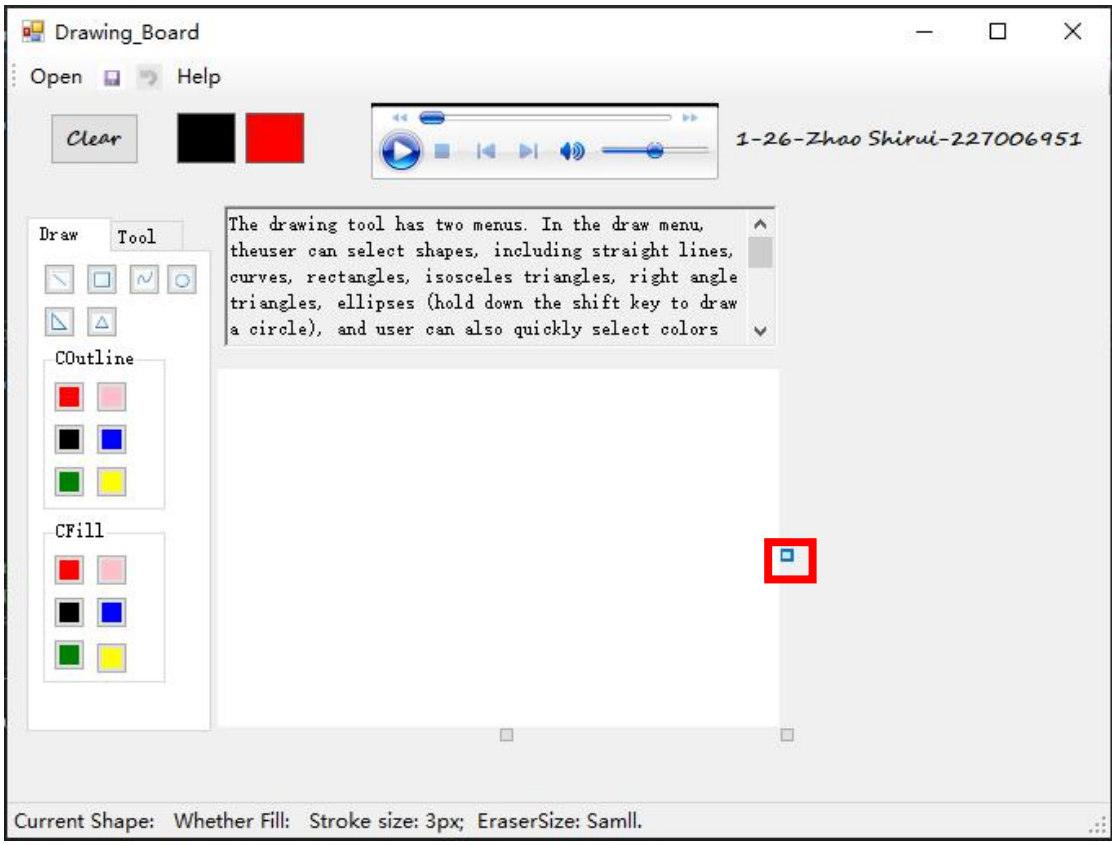



```
 //Descripetion:
        //The mouse movement event can only be triggered
        //when the left mouse button is pressed and the boolean variable is true.
        //The changed flag is passed to and the control for moving the canvas is
located.
        private void BtnWEC_MouseMove(object sender, MouseEventArgs e)
        {
            if (change && e.Button == MouseButtons.Left)
            {
                DBMove(flag); References method 4.3.3.
                DBLocation(); References method 4.3.1
            }
        }
```

The methods 4.3.1 and 4.3.3 are cited.

4.4.5 BtnWEC_MouseUp(object sender, MouseEventArgs e)

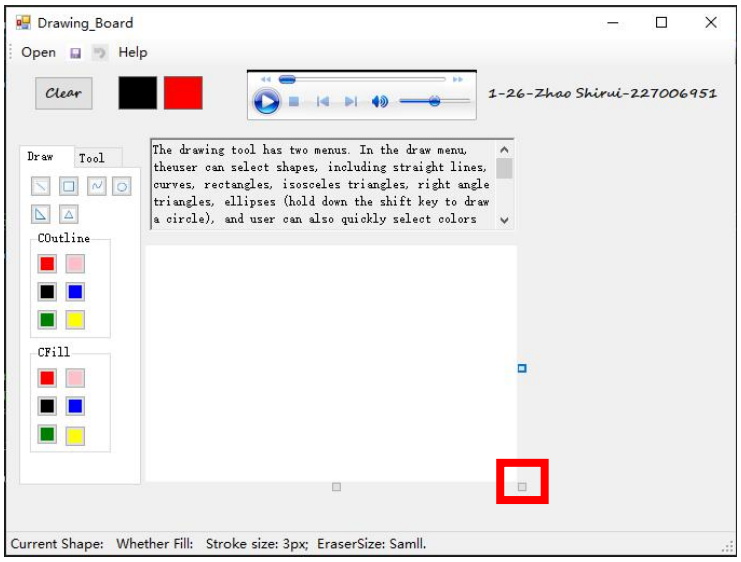
BtnWEC_MouseUp(object sender, MouseEventArgs e) is a mouse up event. When the mouse is up, the boolean variable change that controls the movement of the drawing board becomes false.




```
//Descripetion:
        //The mouse up event.
        //When the mouse is up,
        //the boolean variable that controls the size of the drawing board changes to
false.
        private void BtnWEC_MouseUp(object sender, MouseEventArgs e)
        {
            change = false;

        }
```

4.4.6 BtnNWSEC_MouseHover(object sender, EventArgs e)

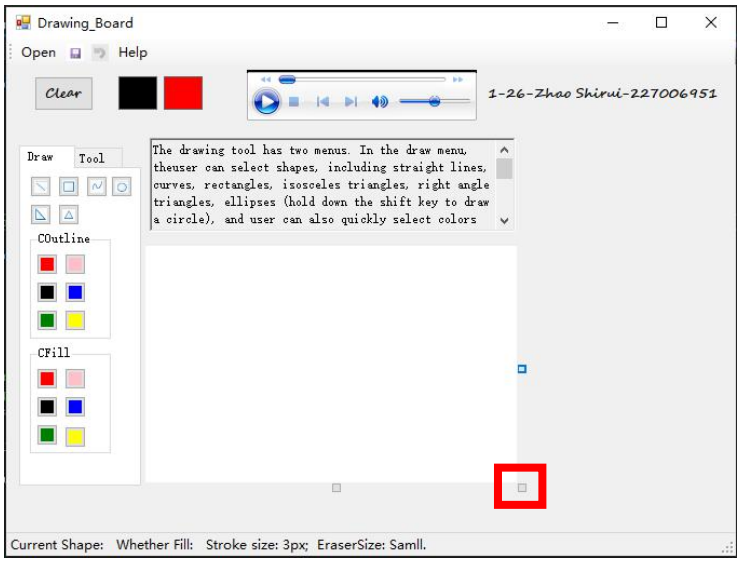
BtnNWSEC_MouseHover(object sender, EventArgs e) is a mouse hover event. The main function of the mouse over event is that when the mouse hovers over this button, the mouse will become an arrow pointing towards NWSE.




```
//Description:
        //Mouse hover event: when the mouse hovers over this control,
        //the style of the mouse will be changed to the style in the NWSE direction.
        private void BtnNWSEC_MouseHover(object sender, EventArgs e)
        {
            this.Cursor = Cursors.SizeNWSE;
        }
```

4.4.7 BtnNWSEC_MouseLeave(object sender, EventArgs e)

BtnNWSEC_MouseLeave(object sender, EventArgs e) is a mouse leave event. The main function of the mouse leave event is that when the mouse leaves this button, the mouse will change to its original shape.

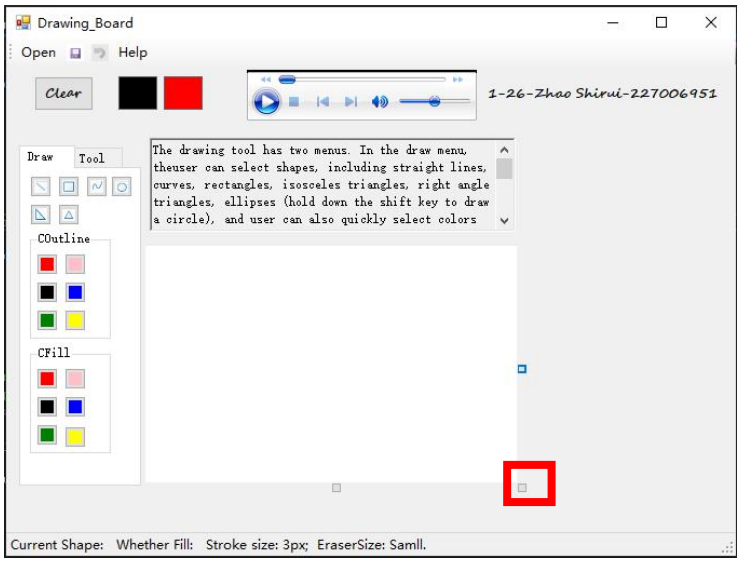



```
//Descripetion:
        //Mouse leave event.
```

```
//When the mouse leaves the control, the mouse returns to the original style.
private void BtnNWSEC_MouseLeave(object sender, EventArgs e)
{
    this.Cursor = Cursors.Arrow;
}
```

4.4.8 BtnNWSEC_MouseDown(object sender, MouseEventArgs e)

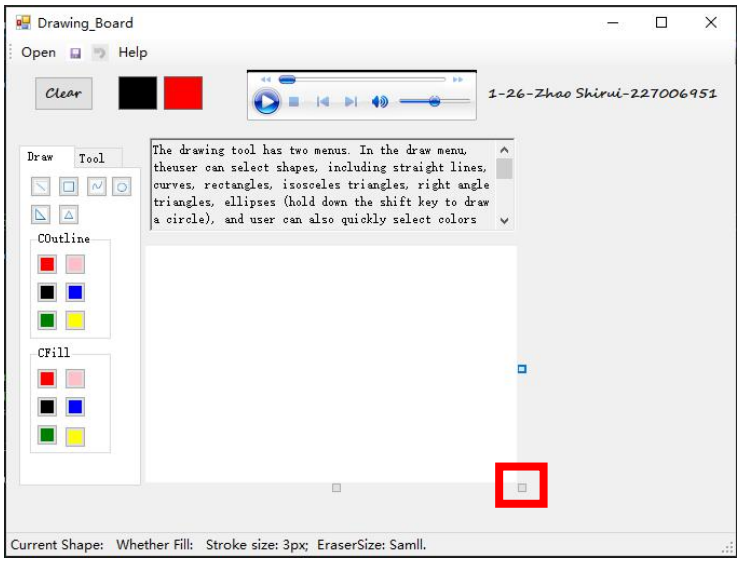
BtnWNSEC_MouseDown(object sender, MouseEventArgs e) is a mouse down event. When the left mouse button is pressed, the boolean variable change becomes true and the control flag becomes 2.




```
//Description:
    //Mouse down event: when the left mouse button is pressed,
    //the boolean variable change is changed to true
    //and the integer variable flag is changed to 2.
    private void BtnNWSEC_MouseDown(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            change = true;
            flag = 2;
        }
    }
```

4.4.9 BtnNWSEC_MouseMove(object sender, MouseEventArgs e)

BtnNWSEC_ MouseMove (object sender, mouseeventargs E) is a mouse movement event. When change is true and the left mouse button is clicked, the drawing board can be enlarged or reduced in the direction controlled by this button according to the incoming flag.

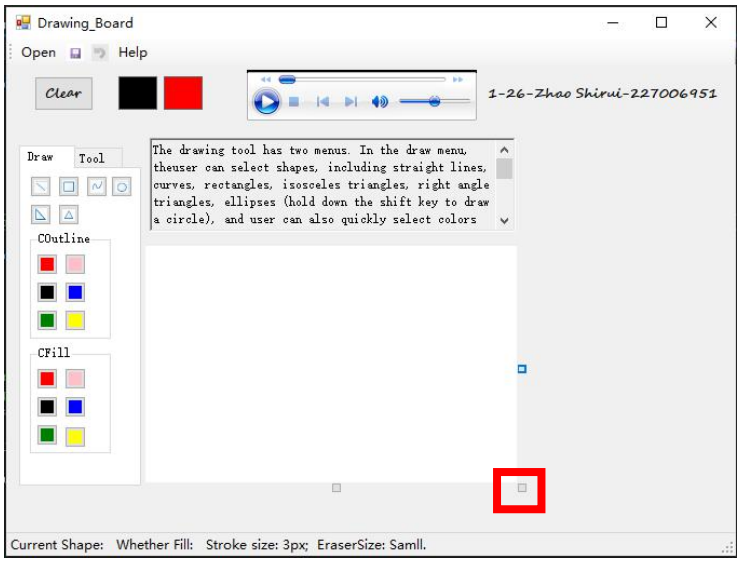



```
//Description:
        //The mouse movement event can only be triggered
        //when the left mouse button is pressed and the boolean variable is true.
        //The changed flag is passed to and the control for moving the canvas is
located.
        private void BtnNWSEC_MouseMove(object sender, MouseEventArgs e)
        {
            if (change && e.Button == MouseButtons.Left)
            {
                DBMove(flag); References method 4.3.3.
                DBLocation(); References method 4.3.1
            }
        }
```

The methods 4.3.1 and 4.3.3 are cited.

4.4.10 BtnNWSEC_MouseUp(object sender, MouseEventArgs e)

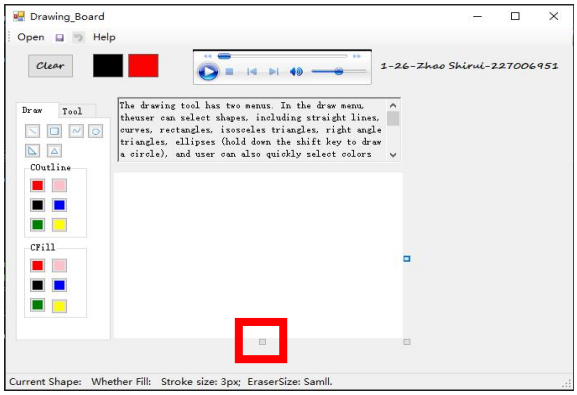
BtnNWSEC_MouseUp(object sender, MouseEventArgs e) is a mouse up event. When the mouse is up, the boolean variable change that controls the movement of the drawing board becomes false.




```
//Description:
        //The mouse up event.
        //When the mouse is up,
        //the boolean variable that controls the size of the drawing board changes to
false.
        private void BtnNWSEC_MouseUp(object sender, MouseEventArgs e)
        {
            change = false;
        }
```

4.4.11 BtnNSC_MouseHover(object sender, EventArgs e)

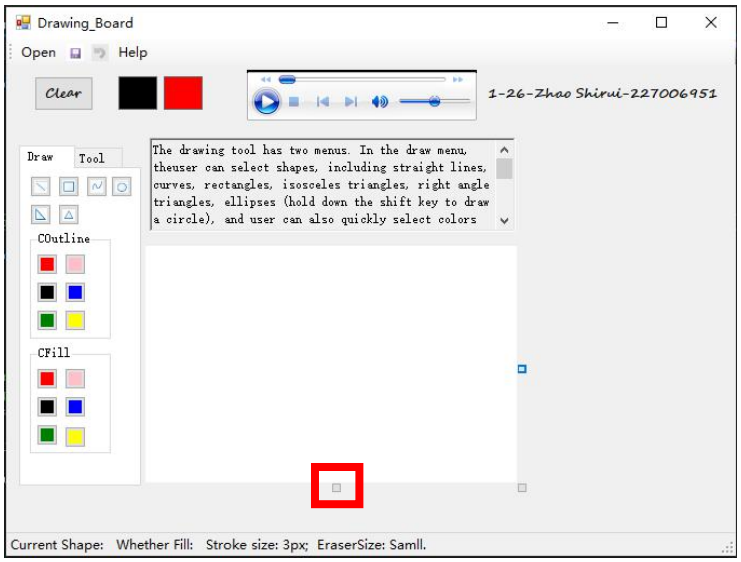
BtnNSC_MouseHover(object sender, EventArgs e) is a mouse hover event. The main function of the mouse over event is that when the mouse hovers over this button, the mouse will become an arrow pointing towards NS.



```
//Description:
        //Mouse hover event: when the mouse hovers over this control,
        //the style of the mouse will be changed to the style in the NS direction.
        private void BtnNSC_MouseHover(object sender, EventArgs e)
        {
            this.Cursor = Cursors.SizeNS;
        }
```

4.4.12 BtnNSC_MouseLeave(object sender, EventArgs e)

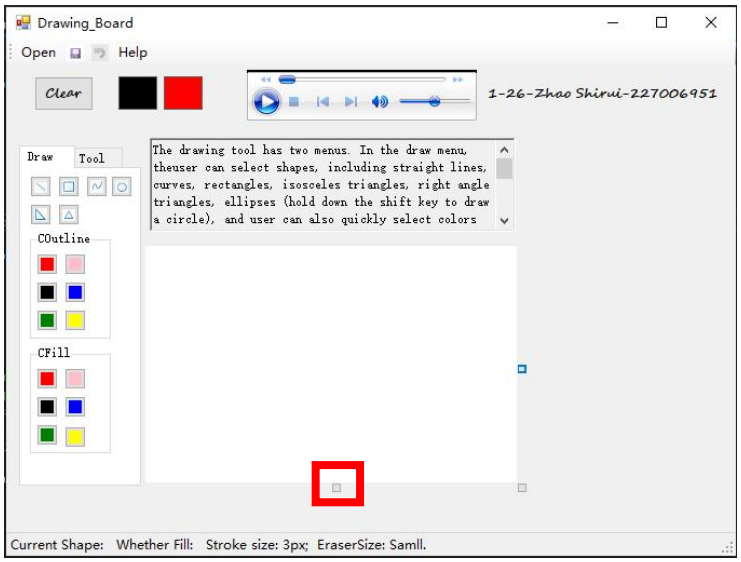
BtnNSC_MouseLeave(object sender, EventArgs e) is a mouse leave event. The main function of the mouse leave event is that when the mouse leaves this button, the mouse will change to its original shape.




```
 //Descripetion:
        //Mouse leave event.
        //When the mouse leaves the control, the mouse returns to the original style.
        private void BtnNSC_MouseLeave(object sender, EventArgs e)
        {
            this.Cursor = Cursors.Arrow;
        }
```

4.4.13 BtnNSC_MouseDown(object sender, MouseEventArgs e)

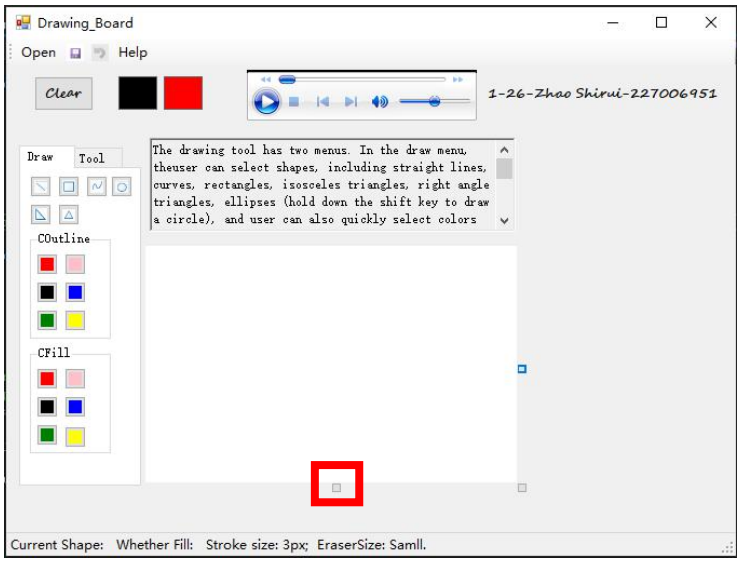
BtnNSC_MouseDown(object sender, MouseEventArgs e) is a mouse down event. When the left mouse button is pressed, the boolean variable change becomes true and the control flag becomes 3.




```
//Description:
        //Mouse down event: when the left mouse button is pressed,
        //the boolean variable change is changed to true
        //and the integer variable flag is changed to 3.
        private void BtnNSC_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                change = true;
                flag = 3;
            }
        }
```

4.4.14 BtnNSC_MouseMove(object sender, MouseEventArgs e)

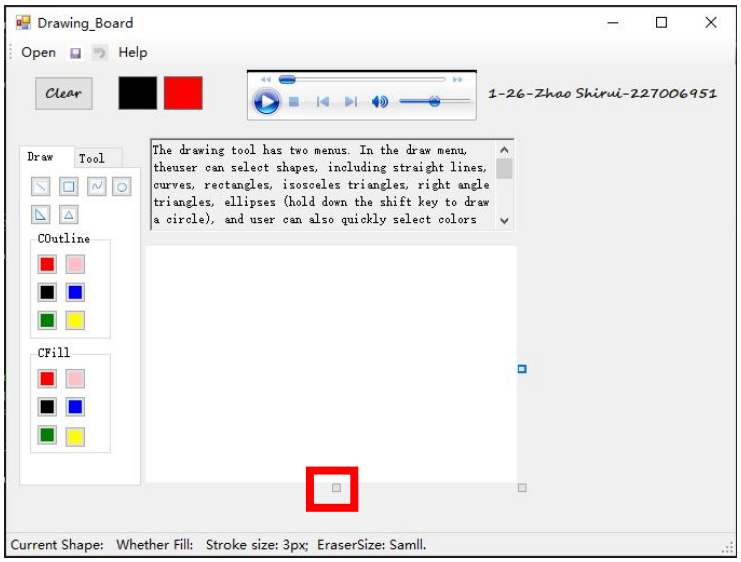
BtnNSC_ MouseMove (object sender, mouseeventargs E) is a mouse movement event. When change is true and the left mouse button is clicked, the drawing board can be enlarged or reduced in the direction controlled by this button according to the incoming flag.




```
//Description:
        //The mouse movement event can only be triggered
        //when the left mouse button is pressed and the boolean variable is true.
        //The changed flag is passed to and the control for moving the canvas is
located.
        private void BtnNSC_MouseMove(object sender, MouseEventArgs e)
        {
            if (change && e.Button == MouseButtons.Left)
            {
                DBMove(flag); References method 4.3.3.
                DBLocation(); References method 4.3.1
            }
        }
```

4.4.15 BtnNSC_MouseUp(object sender, MouseEventArgs e)

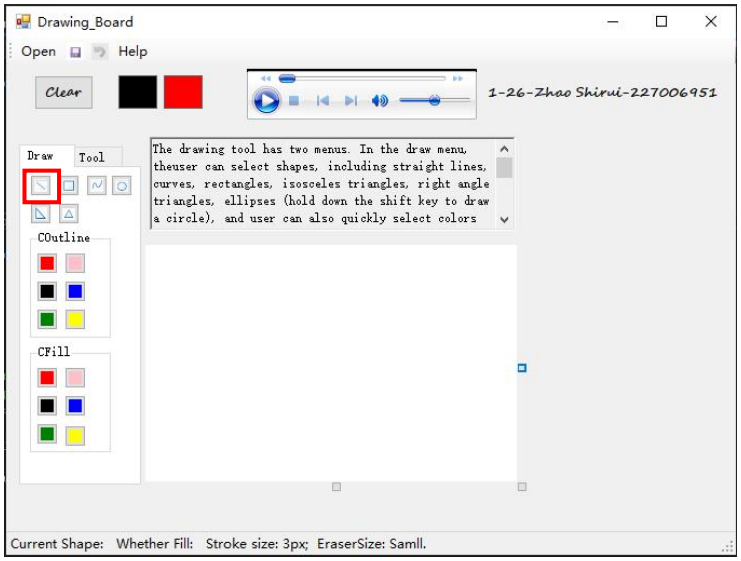
BtnNSC_MouseUp(object sender, MouseEventArgs e) is a mouse up event. When the mouse is up, the boolean variable change that controls the movement of the drawing board becomes false.




```
//Description:
        //The mouse up event.
        //When the mouse is up,
        //the boolean variable that controls the size of the drawing board changes to false.
        private void BtnNSC_MouseUp(object sender, MouseEventArgs e)
        {
            change = false;
        }
```

4.4.16 BtnLine_Click(object sender, EventArgs e)

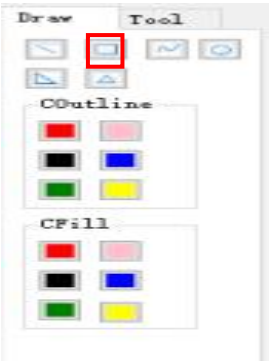
After clicking this button, the user can draw a straight line on the drawing board.




```
//Description:
            //Click the line event, click the button,
            //the value of type will change to 1,
            //and the status bar will prompt that the current drawing mode is to draw a
straight line.
            private void BtnLine_Click(object sender, EventArgs e)
            {
                  type = 1;
                  TslShape.Text = "Current Shape: Line.";
          }
```

4.4.17 BtnRectangle_Click(object sender, EventArgs e)

After clicking this button, the user can draw a rectangle on the drawing board.

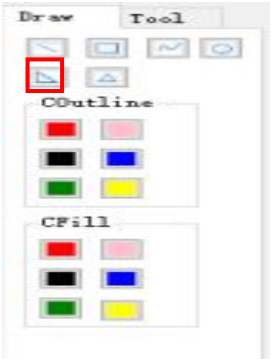



```
 //Description:
```

```
        //Click the rectangle event, click the button,
        //the value of type will change to 2,
        //and the status bar will prompt that the current drawing mode is to draw a
rectangle.
        private void BtnRectangle_Click(object sender, EventArgs e)
        {
            type = 2;
            TslShape.Text = "Current Shape: Rectangle.";
        }
```

4.4.18 BtnRTriangle_Click(object sender, EventArgs e)

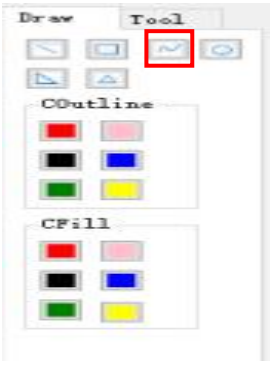
After clicking this button, the user can draw a right triangle line on the drawing board.




```
//Descripetion:
        //Click the triangle event, click the button,
        //the value of type will change to 3,
        //and the status bar will prompt that the current drawing mode is to draw a
triangle.
        private void BtnRTriangle_Click(object sender, EventArgs e)
        {
            type = 3;
            TslShape.Text = "Current Shape: Right Triangle.";
        }
```

4.4.19 BtnCurve_Click(object sender, EventArgs e)

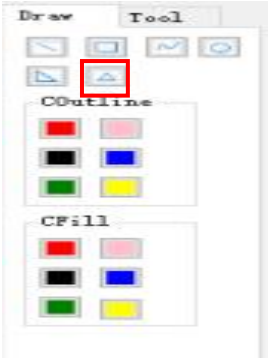
After clicking this button, the user can draw a curve on the drawing board.




```
//Descripetion:
        //Click the curve event, click the button,
        //the value of type will change to 4,
        //and the status bar will prompt that the current drawing mode is to draw the
curve.
        private void btnCurve_Click(object sender, EventArgs e)
        {
            type = 4;
            TslShape.Text = "Current Shape: Curve.";
        }
```

4.4.20 BtnTriangle_Click(object sender, EventArgs e)

After clicking this button, the user can draw an isosceles triangle on the drawing board.

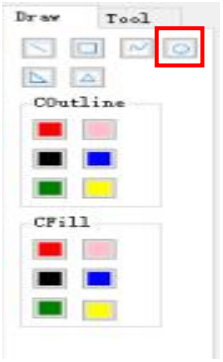



```
//Description:
        //Click the isosceles triangle event, click the button,
        //the value of type will change to 5,
        //and the status bar will prompt that the current drawing mode is to draw an
isosceles triangle.
        private void BtnTriangle_Click(object sender, EventArgs e)
        {
```

```
            type = 5;
            TslShape.Text = "Current Shape: Isosceles Triangle.";
        }
```

4.4.21 BtnEllipse_Click(object sender, EventArgs e)

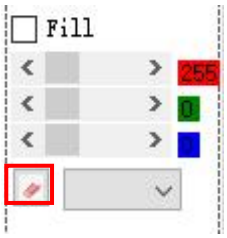
After clicking this button, the user can draw an ellipse on the drawing board.




```
//Description:
        //Click the ellipse event, click the button,
        //the value of type will change to 6,
        //and the status bar will prompt that the current drawing mode is to draw an
ellipse.

        private void BtnEllipse_Click(object sender, EventArgs e)
        {
            type = 6;
            TslShape.Text = "Current Shape: Ellipse.";
        }
```

4.4.22 BtnEraser_Click(object sender, EventArgs e)

After clicking this button, the user can use the eraser on the drawing board.




```
  //Descripetion:
        //Click the eraser event, click the button,
        //the value of type will change to 7,
        //and the status bar will prompt that the current drawing mode is to use
eraser.
```

```
private void BtnEraser_Click(object sender, EventArgs e)
{
    type = 7;
    TslShape.Text = "Current Shape: Eraser.";
}
```

4.4.23 BtnClear_Click(object sender, EventArgs e)

After clicking this button, the canvas will be cleared.


```
//Description:
    //Click the clear event and click this button. The drawing board will be cleared.
    //The value of type will change to 0.
    private void BtnClear_Click(object sender, EventArgs e)
    {
        type = 0;
        g.Clear(Color.White);
        PcbDB.BackgroundImage = im;
        PcbDB.Refresh();
    }
```

4.4.24 PcbDB_MouseDown(object sender, MouseEventArgs e)

PcbDB_ MouseDown (object sender, mouseeventargs E) is a mouse down event. When the left mouse button is pressed, the program automatically records the coordinates of the mouse on the drawing board, changes the boolean variable draw to true, and stores the current drawing.

```
//Description:
            //Sketchpad mouse down event: when the mouse is pressed on the
sketchpad,
            //if the value of type is not 0 and the left mouse button is pressed,
            //Set the boolean variable draw to true,
            //the starting position of the drawing can be recorded,
            //and the bitmap of now can be initialized and stored.
            //If the value of type is 4 or 7,
            //the coordinates of the starting point are also recorded,
            //and the bitmap is initialized and stored.
            private void PcbDB_MouseDown(object sender, MouseEventArgs e)
            {
                if (type != 0 && e.Button == MouseButtons.Left)
                {
                    draw = true;
                    startpt = e.Location;
                    now = new Bitmap(im);
                    undo.Push(now);

                }
                if (type == 4 || type == 7)
                {
                    plist.Add(e.Location);
                    now = new Bitmap(im);
                    undo.Push(now);
                }
```

4.4.25 PcbDB_MouseMove(object sender, MouseEventArgs e)

PcbDB_ MouseMove (object sender, mouseeventargs E) is a mouse movement event. When the mouse starts to move, the program will draw the corresponding image according to the value of type. If fill is checked, some graphics will have filled colors.

```
        //Description:
            // For the mouse movement event on the drawing board,
            //when the draw value is true,
            //the position of the mouse will be updated in real time each time it is
moved,
            //and a temporary class diagram imtemp will be instantiated at the same
time.
            //If both the brush and fill options are selected, the drawn figure is the filled
figure.
            //If type is not of types 4 and 7, G is instantiated from imtemp.
            //Then, different types of pictures are drawn
            //according to the values of different types (from 1 to 7).
            //Finally, set the image of pcbdb to imtemp and refresh it in real time.
            private void PcbDB_MouseMove(object sender, MouseEventArgs e)
            {
                if (draw)
                {
                    endpt = e.Location;
                    imtemp = new Bitmap(im);
                    if (type != 4 && type != 7)
                    {
                        g = Graphics.FromImage(imtemp);

                    }
                    switch (type)
                    {
                        case 1:
                            g.DrawLine(pen, startpt, endpt);
                            break;
                        case 2:
                            if (CbxOutline.Checked)
                            {
                                g.DrawRectangle(pen, startpt.X, startpt.Y, endpt.X
- startpt.X, endpt.Y - startpt.Y);
                                g.DrawRectangle(pen, endpt.X, endpt.Y, startpt.X -
endpt.X, startpt.Y - endpt.Y);
                                g.DrawRectangle(pen, startpt.X, endpt.Y, endpt.X -
startpt.X, startpt.Y - endpt.Y);
                                g.DrawRectangle(pen, endpt.X, startpt.Y, startpt.X
- endpt.X, endpt.Y - startpt.Y);
                            }
                            if (CbxFill.Checked)
                            {
                                g.FillRectangle(brush, startpt.X, startpt.Y, endpt.X
```

```
- startpt.X, endpt.Y - startpt.Y);
                    g.FillRectangle(brush, endpt.X, endpt.Y, startpt.X -
endpt.X, startpt.Y - endpt.Y);
                    g.FillRectangle(brush, startpt.X, endpt.Y, endpt.X -
startpt.X, startpt.Y - endpt.Y);
                    g.FillRectangle(brush, endpt.X, startpt.Y, startpt.X
- endpt.X, endpt.Y - startpt.Y);
                }
                break;
            case 3:
                Point p1 = new Point(startpt.X, startpt.Y);
                Point p2 = new Point(startpt.X, endpt.Y);
                Point p3 = new Point(endpt.X, endpt.Y);
                Point[] parray = { p1, p2, p3 };
                if (CbxOutline.Checked)
                {
                    g.DrawPolygon(pen, parray);
                }
                if (CbxFill.Checked)
                {
                    g.FillPolygon(brush, parray);
                }
                break;
            case 4:
                plist.Add(e.Location);
                g.DrawCurve(pen, plist.ToArray());
                break;
            case 5:
                p1 = new Point(startpt.X, startpt.Y);
                p2 = new Point((endpt.X + startpt.X) / 2, endpt.Y);
                p3 = new Point(endpt.X, startpt.Y);
                Point[] parray1 = { p1, p2, p3 };
                if (CbxOutline.Checked)
                {
                    g.DrawPolygon(pen, parray1);
                }
                if (CbxFill.Checked)
                {
                    g.FillPolygon(brush, parray1);
                }
                break;
            case 6:
                if (CbxOutline.Checked)
                {
```

```
                                    if (Control.ModifierKeys == Keys.Shift &&
e.Button == MouseButtons.Left)
                                    {
                                        g.DrawEllipse(pen, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.X - startpt.X));
                                    }
                                    else
                                        g.DrawEllipse(pen, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.Y - startpt.Y));
                                }
                                if (CbxFill.Checked)
                                {
                                    if (Control.ModifierKeys == Keys.Shift &&
e.Button == MouseButtons.Left)
                                    {
                                        g.FillEllipse(brush, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.X - startpt.X));
                                    }
                                    else
                                        g.FillEllipse(brush, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.Y - startpt.Y));
                                }
                                break;
                            case 7:
                                plist.Add(e.Location);
                                g.DrawCurve(eraser, plist.ToArray());
                                break;

                        }

                        PcbDB.Image = imtemp;
                        PcbDB.Refresh();

                    }
                }
```

4.4.26 PcbDB_MouseUp(object sender, MouseEventArgs e)

PcbDB_ Mouseup (object sender, mouseeventargs E) is a mouse up event. When the left mouse button is clicked, the program will record the graph drawn during the just moving process, save it, and then display it on the canvas.

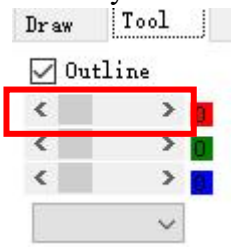

```
//Description:
        //Pcbdb mouse up event:
        //when the mouse is up,
        //the position of the mouse is automatically recorded,
        //and according to the drawing trace,
        //an identical figure is drawn in IM and stored in the pcbdb image.
        //When the left mouse button is raised, the boolean variable draw will
become false.
        private void PcbDB_MouseUp(object sender, MouseEventArgs e)
        {
            if (draw)
            {
                endpt = e.Location;
                g = Graphics.FromImage(im);

                switch (type)
                {
                    case 1:
                        g.DrawLine(pen, startpt, endpt);
                        break;
                    case 2:
                        if (CbxOutline.Checked)
                        {
                            g.DrawRectangle(pen, startpt.X, startpt.Y, endpt.X
- startpt.X, endpt.Y - startpt.Y);
                            g.DrawRectangle(pen, endpt.X, endpt.Y, startpt.X -
endpt.X, startpt.Y - endpt.Y);
                            g.DrawRectangle(pen, startpt.X, endpt.Y, endpt.X -
```

```
startpt.X, startpt.Y - endpt.Y);
                    g.DrawRectangle(pen, endpt.X, startpt.Y, startpt.X
- endpt.X, endpt.Y - startpt.Y);
                }
                if (CbxFill.Checked)
                {
                    g.FillRectangle(brush, startpt.X, startpt.Y, endpt.X
- startpt.X, endpt.Y - startpt.Y);
                    g.FillRectangle(brush, endpt.X, endpt.Y, startpt.X -
endpt.X, startpt.Y - endpt.Y);
                    g.FillRectangle(brush, startpt.X, endpt.Y, endpt.X -
startpt.X, startpt.Y - endpt.Y);
                    g.FillRectangle(brush, endpt.X, startpt.Y, startpt.X
- endpt.X, endpt.Y - startpt.Y);
                }
                break;
            case 3:
                Point p1 = new Point(startpt.X, startpt.Y);
                Point p2 = new Point(startpt.X, endpt.Y);
                Point p3 = new Point(endpt.X, endpt.Y);
                Point[] parray = { p1, p2, p3 };
                if (CbxOutline.Checked)
                {
                    g.DrawPolygon(pen, parray);
                }
                if (CbxFill.Checked)
                {
                    g.FillPolygon(brush, parray);
                }
                break;
            case 4:
                plist.Clear();
                break;
            case 5:
                p1 = new Point(startpt.X, startpt.Y);
                p2 = new Point((endpt.X + startpt.X) / 2, endpt.Y);
                p3 = new Point(endpt.X, startpt.Y);
                Point[] parray1 = { p1, p2, p3 };
                if (CbxOutline.Checked)
                {
                    g.DrawPolygon(pen, parray1);
                }
                if (CbxFill.Checked)
                {
```

```
                    g.FillPolygon(brush, parray1);
                }
                break;
            case 6:
                if (CbxOutline.Checked)
                {
                    if (Control.ModifierKeys == Keys.Shift &&
e.Button == MouseButtons.Left)
                    {
                        g.DrawEllipse(pen, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.X - startpt.X));
                    }
                    else
                        g.DrawEllipse(pen, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.Y - startpt.Y));
                }
                if (CbxFill.Checked)
                {
                    if (Control.ModifierKeys == Keys.Shift &&
e.Button == MouseButtons.Left)
                    {
                        g.FillEllipse(brush, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.X - startpt.X));
                    }
                    else
                        g.FillEllipse(brush, startpt.X, startpt.Y,
(endpt.X - startpt.X), (endpt.Y - startpt.Y));
                }
                break;
            case 7:
                plist.Clear();
                break;
        }

        PcbDB.Image = im;
        PcbDB.Refresh();
        draw = false;
        TsbUndo.Enabled = true;
    }
}
```

4.4.27 HsbR_Scroll(object sender, ScrollEventArgs e)

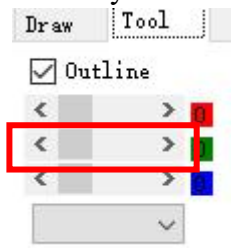
HsbR_ Scroll (object sender, scrolleventargs E) is a horizontal scroll bar value change event. When the horizontal scroll bar is dragged, the R value of the pen color changes similarly.




```
//Description:
        //Adjust the horizontal scroll bar value event.
        //When the horizontal scroll bar value changes,
        //the color will be displayed in RGB,
        //and the current pen color and RGB specific value will be prompted.
        private void HsbR_Scroll(object sender, ScrollEventArgs e)
        {
            LblR.Text = HsbR.Value.ToString();
            pen.Color = Color.FromArgb(HsbR.Value, HsbG.Value, HsbB.Value);
            LblColor1.BackColor = Color.FromArgb(HsbR.Value, HsbG.Value,
HsbB.Value);
        }
```

4.4.28 HsbG_Scroll(object sender, ScrollEventArgs e)

HsbG_ Scroll (object sender, scrolleventargs E) is a horizontal scroll bar value change event. When the horizontal scroll bar is dragged, the G value of the pen color changes similarly.

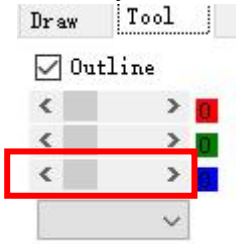



```
//Description:
        //Adjust the horizontal scroll bar value event.
        //When the horizontal scroll bar value changes,
        //the color will be displayed in RGB,
        //and the current pen color and RGB specific value will be prompted.
        private void HsbG_Scroll(object sender, ScrollEventArgs e)
        {
            LblG.Text = HsbG.Value.ToString();
```

```
            pen.Color = Color.FromArgb(HsbR.Value, HsbG.Value, HsbB.Value);
            LblColor1.BackColor = Color.FromArgb(HsbR.Value, HsbG.Value,
HsbB.Value);
        }
```

4.4.29 HsbB_Scroll(object sender, ScrollEventArgs e)

HsbB_ Scroll (object sender, scrolleventargs E) is a horizontal scroll bar value change event. When the horizontal scroll bar is dragged, the B value of the pen color changes similarly.




```
//Description:
        //Adjust the horizontal scroll bar value event.
        //When the horizontal scroll bar value changes,
        //the color will be displayed in RGB,
        //and the current pen color and RGB specific value will be prompted.
        private void HsbB_Scroll(object sender, ScrollEventArgs e)
        {
            LblB.Text = HsbB.Value.ToString();
            pen.Color = Color.FromArgb(HsbR.Value, HsbG.Value, HsbB.Value);
            LblColor1.BackColor = Color.FromArgb(HsbR.Value, HsbG.Value,
HsbB.Value);
        }
```

4.4.30 HsbRF_Scroll(object sender, ScrollEventArgs e)

HsbRF_Scroll(object sender, ScrollEventArgs e) is a horizontal scroll bar value change event. When the horizontal scroll bar is dragged, the R value of the brush color changes similarly.


```
//Description:
        //Adjust the horizontal scroll bar value event.
        //When the horizontal scroll bar value changes,
```

```
        //the color will be displayed in RGB,
        //and the current brush color and RGB specific value will be prompted.
        private void HsbRF_Scroll(object sender, ScrollEventArgs e)
        {
            LblRF.Text = HsbRF.Value.ToString();
            brush = new SolidBrush(Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value));
            LblColor2.BackColor = Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value);
        }
```

4.4.31 HsbGF_Scroll(object sender, ScrollEventArgs e)

HsbGF_Scroll(object sender, ScrollEventArgs e) is a horizontal scroll bar value change event. When the horizontal scroll bar is dragged, the G value of the brush color changes similarly.


```
//Description:
        //Adjust the horizontal scroll bar value event.
        //When the horizontal scroll bar value changes,
        //the color will be displayed in RGB,
        //and the current brush color and RGB specific value will be prompted.
        private void HsbGF_Scroll(object sender, ScrollEventArgs e)
        {
            LblGF.Text = HsbGF.Value.ToString();
            brush = new SolidBrush(Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value));
            LblColor2.BackColor = Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value);
        }
```

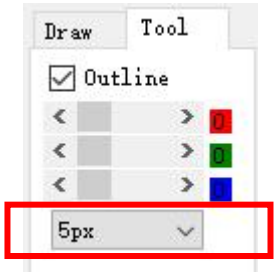
4.4.32 HsbBF_Scroll(object sender, ScrollEventArgs e)

HsbBF_Scroll(object sender, ScrollEventArgs e) is a horizontal scroll bar value change event. When the horizontal scroll bar is dragged, the B value of the brush color changes similarly.


```
    //Description:
            //Adjust the horizontal scroll bar value event.
            //When the horizontal scroll bar value changes,
            //the color will be displayed in RGB,
            //and the current brush color and RGB specific value will be prompted.
            private void HsbBF_Scroll(object sender, ScrollEventArgs e)
            {
                LblBF.Text = HsbBF.Value.ToString();
                brush = new SolidBrush(Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value));
                LblColor2.BackColor = Color.FromArgb(HsbRF.Value, HsbGF.Value,
HsbBF.Value);
            }
```

4.4.33 CmxThickness_SelectedIndexChanged

CmxThickness_ Selectedindexchanged is a combo box option change event. When the selection is changed, the stroke size will change with the option change.




```
//Description:
            //Stroke size adjustment event.
            //This is a combo box selection box.
            //The user can select different stroke sizes,
            //and the status prompt bar will prompt the current stroke size.
            private void CmxThickness_SelectedIndexChanged(object sender,
EventArgs e)
            {

                switch (CmxThickness.Text)
                {
                    case "1px":
                        pen.Width = 1;
```

```
                TslStrokesize.Text = "1 px";
                break;
            case "3px":
                pen.Width = 3;
                TslStrokesize.Text = "3 px";
                break;
            case "5px":
                pen.Width = 5;
                TslStrokesize.Text = "5 px";
                break;
            case "8px":
                pen.Width = 8;
                TslStrokesize.Text = "8 px";
                break;
            case "10px":
                pen.Width = 10;
                TslStrokesize.Text = "10 px";
                break;
            case "15px":
                pen.Width = 15;
                TslStrokesize.Text = "15 px";
                break;
            case "20px":
                pen.Width = 20;
                TslStrokesize.Text = "20 px";
                break;
        }
    }
```

4.4.34 CmxES_SelectedIndexChanged(object sender, EventArgs e)

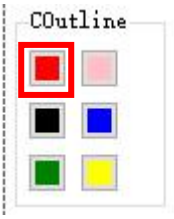
CmxES_ Selectedindexchanged (object sender, EventArgs E) is a combo box option change event. When the selection is changed, the size of the rubber will change with the change of the option.


```
//Description:
        //Rubber sizing event.
        //This is a combo box selection box.
```

```
//The user can select different stroke sizes,
//and the status prompt bar will prompt the current rubber size.
private void CmxES_SelectedIndexChanged(object sender, EventArgs e)
{
    switch (CmxES.Text)
    {
        case "Small":
            eraser.Width = 5;
            TslEraserSize.Text = "EraserSize: Small.";
            break;
        case "Medium":
            eraser.Width = 10;
            TslEraserSize.Text = "EraserSize: Medium.";
            break;
        case "Big":
            eraser.Width = 20;
            TslEraserSize.Text = "EraserSize: Big.";
            break;
    }

}
```

4.4.35 BtnRed_Click(object sender, EventArgs e)

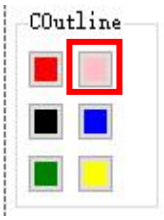
BtnRed_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the pen will turn red, which is reflected in the horizontal scroll bar and RGB color display.




```
//Description:
    //Quickly select the brush color event.
    //Click this button, and the pen color will switch to Red,
    //and the interface will display the current pen color.
    private void BtnRed_Click(object sender, EventArgs e)
    {
        pen.Color = Color.Red;
        color(); References method 4.3.4
        HScolor(); References method 4.3.5

    }
```

4.4.36 BtnPink_Click(object sender, EventArgs e)

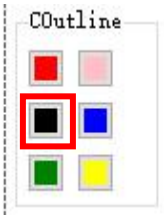
BtnPink_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the pen will turn pink, which is reflected in the horizontal scroll bar and RGB color display.




```
//Description:
        //Quickly select the brush color event.
        //Click this button, and the pen color will switch to Pink,
        //and the interface will display the current pen color.
        private void BtnPink_Click(object sender, EventArgs e)
        {
            pen.Color = Color.Pink;
            color(); References method 4.3.4
            HScolor(); References method 4.3.5

        }
```

4.4.37 BtnBlack_Click(object sender, EventArgs e)

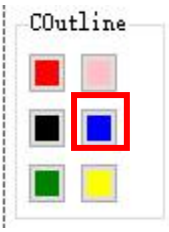
BtnBlack_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the pen will turn black, which is reflected in the horizontal scroll bar and RGB color display.




```
//Description:
        //Quickly select the brush color event.
        //Click this button, and the pen color will switch to Black/
        //and the interface will display the current pen color.
        private void BtnBlack_Click(object sender, EventArgs e)
        {
            pen.Color = Color.Black;
            color(); References method 4.3.4
            HScolor(); References method 4.3.5}
```

4.4.38 BtnBlue_Click(object sender, EventArgs e)

BtnBlue_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the pen will turn blue, which is reflected in the horizontal scroll bar and RGB color display.




```
//Description:
        //Quickly select the brush color event.
        //Click this button, and the pen color will switch to Blue,
        //and the interface will display the current pen color.
        private void BtnBlue_Click(object sender, EventArgs e)
        {
            pen.Color = Color.Blue;
            color(); References method 4.3.4
            HScolor(); References method 4.3.5
        }
```

4.4.39 BtnGreen_Click(object sender, EventArgs e)

BtnGreen_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the pen will turn green, which is reflected in the horizontal scroll bar and RGB color display.


```
//Description:
        //Quickly select the brush color event.
        //Click this button, and the pen color will switch to Green,
        //and the interface will display the current pen color.
        private void BtnGreen_Click(object sender, EventArgs e)
        {
            pen.Color = Color.Green;
            color(); References method 4.3.4
            HScolor(); References method 4.3.5
        }
```

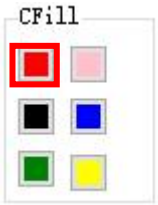
4.4.40 BtnYellow_Click(object sender, EventArgs e)

BtnYellow_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the pen will turn yellow, which is reflected in the horizontal scroll bar and RGB color display.


```
//Description:
        //Quickly select the brush color event.
        //Click this button, and the pen color will switch to Yellow,
        //and the interface will display the current pen color.
        private void BtnYellow_Click(object sender, EventArgs e)
        {
            pen.Color = Color.Yellow;
            color(); References method 4.3.4
            HScolor(); References method 4.3.5
        }
```

4.4.41 BtnRedF_Click(object sender, EventArgs e)

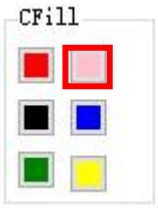
BtnRedF_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the brush will turn red, which is reflected in the horizontal scroll bar and RGB color display.




```
 //Description:
        //Quickly select the brush color event,
        //click this button, the brush color will switch to Red,
        //and the interface will display the current brush color.
        private void BtnRedF_Click(object sender, EventArgs e)
        {
            brush = new SolidBrush(Color.Red);
            LblColor2.BackColor = (Color.Red);
            HsbRF.Value = Color.Red.R;
            HsbGF.Value = Color.Red.G;
            HsbBF.Value = Color.Red.B;}
```

4.4.42 BtnPinkF_Click(object sender, EventArgs e)

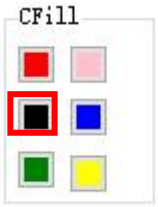
BtnPinkF_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the brush will turn pink, which is reflected in the horizontal scroll bar and RGB color display.




```
//Description:
        //Quickly select the brush color event,
        //click this button, the brush color will switch to Pink,
        //and the interface will display the current brush color.
        private void BtnPinkF_Click(object sender, EventArgs e)
        {
            brush = new SolidBrush(Color.Pink);
            LblColor2.BackColor = (Color.Pink);
            HsbRF.Value = Color.Pink.R;
            HsbGF.Value = Color.Pink.G;
            HsbBF.Value = Color.Pink.B;
        }
```

4.4.43 BtnBlackF_Click(object sender, EventArgs e)

BtnBlackF_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the brush will turn black, which is reflected in the horizontal scroll bar and RGB color display.

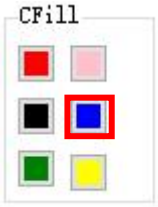



```
//Description:
        //Quickly select the brush color event,
        //click this button, the brush color will switch to Black,
        //and the interface will display the current brush color.
        private void BtnBlackF_Click(object sender, EventArgs e)
        {
            brush = new SolidBrush(Color.Black);
            LblColor2.BackColor = (Color.Black);
```

```
            HsbRF.Value = Color.Black.R;
            HsbGF.Value = Color.Black.G;
            HsbBF.Value = Color.Black.B;
        }
```

4.4.44 BtnBlueF_Click(object sender, EventArgs e)

BtnBlueF_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the brush will turn blue, which is reflected in the horizontal scroll bar and RGB color display.




```
//Description:
        //Quickly select the brush color event,
        //click this button, the brush color will switch to Blue,
        //and the interface will display the current brush color.
        private void BtnBlueF_Click(object sender, EventArgs e)
        {
            brush = new SolidBrush(Color.Blue);
            LblColor2.BackColor = (Color.Blue);
            HsbRF.Value = Color.Blue.R;
            HsbGF.Value = Color.Blue.G;
            HsbBF.Value = Color.Blue.B;
        }
```

4.4.45 BtnGreenF_Click(object sender, EventArgs e)

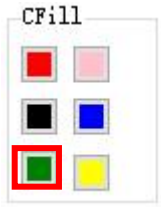
BtnGreenF_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the brush will turn green, which is reflected in the horizontal scroll bar and RGB color display.


```
//Description:
        //Quickly select the brush color event,
        //click this button, the brush color will switch to Green,
        //and the interface will display the current brush color.
```

```
private void BtnGreenF_Click(object sender, EventArgs e)
{
    brush = new SolidBrush(Color.Green);
    LblColor2.BackColor = (Color.Green);
    HsbRF.Value = Color.Green.R;
    HsbGF.Value = Color.Green.G;
    HsbBF.Value = Color.Green.B;
}
```

4.4.46 BtnYellowF_Click(object sender, EventArgs e)

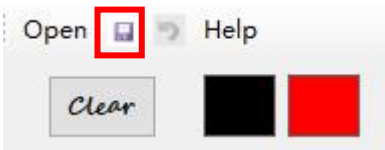
BtnYellowF_ Click (object sender, EventArgs E) is a mouse click event. When the mouse clicks this button, the color of the brush will turn yellow, which is reflected in the horizontal scroll bar and RGB color display.




```
//Description:
    //Quickly select the brush color event,
    //click this button, the brush color will switch to Yellow,
    //and the interface will display the current brush color.
    private void BtnYellowF_Click(object sender, EventArgs e)
    {
        brush = new SolidBrush(Color.Yellow);
        LblColor2.BackColor = (Color.Yellow);
        HsbRF.Value = Color.Yellow.R;
        HsbGF.Value = Color.Yellow.G;
        HsbBF.Value = Color.Yellow.B;
    }
```

4.4.47 TsbSave_Click(object sender, EventArgs e)

TsbSave_ Click (object sender, EventArgs E) is a mouse click event. When the user clicks the tool strip button, the program will save the current drawing content. If the canvas is blank, it is not saved.



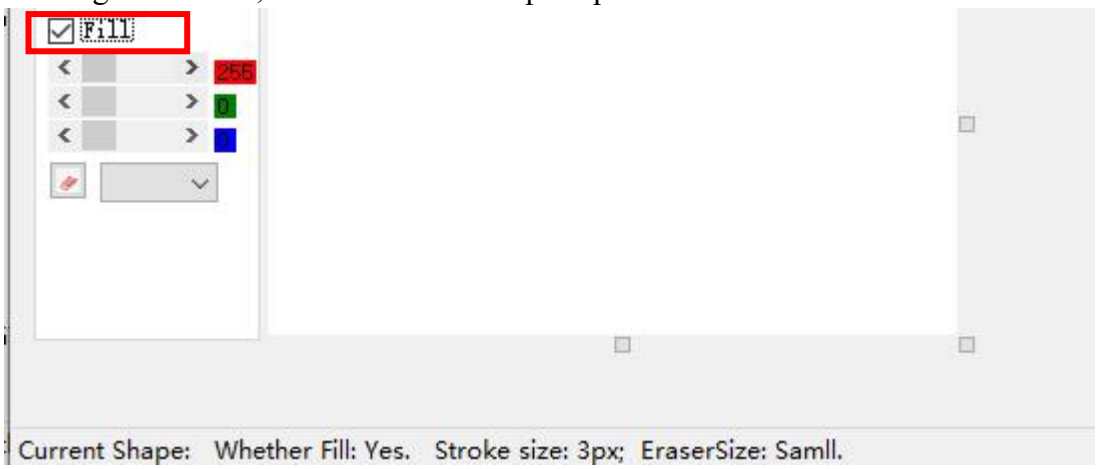

```
//Description:
    //Save the current picture event.
```

```
            //When you click this button,
            //if the picture on the canvas is blank,
            //it will not be saved. If it is a picture with drawing traces,
            //it will be saved and saved in a specific format.
            private void TsbSave_Click(object sender, EventArgs e)
            {
                PcbDB.Refresh();
                for (int i = 0; i < im.Width; i++)
                {
                    for (int j = 0; j < im.Height; j++)
                    {
                        if ((im.GetPixel(i, j).A + im.GetPixel(i, j).R + im.GetPixel(i,
j).G
                            + im.GetPixel(i, j).B) != 1020)
                        {
                            SfdSave.Filter = "BMP file|*.bmp|JPG file|*.jpg|PNG
file|*.png|" +
                         "TIFF file|*.tif|GIF file|*.gif";
                            SfdSave.Title = "Save this file";

                            if (SfdSave.ShowDialog() == DialogResult.OK)
                            {
                                string filename = SfdSave.FileName;
                                string filetext = filename.Remove(0,
filename.Length - 3);
                                switch (filetext)
                                {
                                    case "bmp":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Bmp);
                                        break;
                                    case "jpg":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Jpeg);
                                        break;
                                    case "png":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Png);
                                        break;
                                    case "tif":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Tiff);
                                        break;
                                    case "gif":
```

```
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Gif);
                                        break;
                                    }
                                    goto flag;

                                }
                                else
                                    goto flag;
                            }

                        }
                    }
                flag:;

                }
```

4.4.48 CbxFill_CheckedChanged(object sender, EventArgs e)

CbxFill_ CheckedChanged (object sender, EventArgs E) is an option change event. When it is selected, the user will be prompted that you have selected the filling drawing. Otherwise, the user will not be prompted.




```
 //Description:
            //Whether to select events for filling.
            //If you are sure to select filling,
            //the status bar will prompt that it has been filled.
            //If no fill is selected, the status bar indicates that it is not filled.
            private void CbxFill_CheckedChanged(object sender, EventArgs e)
            {
                if (CbxFill.Checked)
                {
                    TslWFill.Text = "Whether Fill: Yes. ";
                }
```

```
            if (CbxFill.Checked == false)
            {
                TslWFill.Text = "Whether Fill: No. ";
            }
        }
```

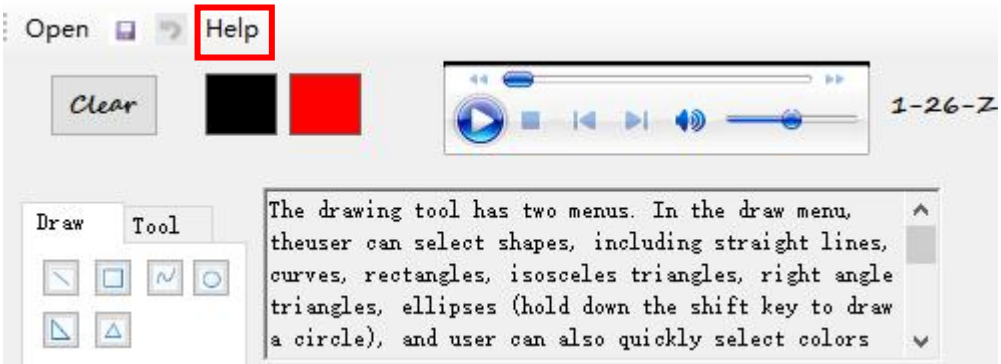
4.4.49 WmpPlayer_StatusChange(object sender, EventArgs e)

WmpPlayer_ Statuschange (object sender, EventArgs E) is a player status change event. When the player ends playing and the end time exceeds one second, the player plays the music again.

```
    //Description:
        //Event of music player status change.
        //If the current music is played,
        //the music player will repeat the second time after 1 second.
        private void WmpPlayer_StatusChange(object sender, EventArgs e)
        {
            if ((int)WmpPlayer.playState == 1)
            {
                System.Threading.Thread.Sleep(1000);
                WmpPlayer.Ctlcontrols.play();
            }
        }
```

4.4.50 TsbHelp_Click(object sender, EventArgs e)

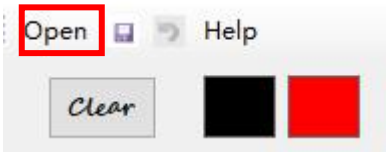
TsbHelp_ Click (object sender, EventArgs E) is a mouse click event. When the user clicks the tool strip button, a help bar will pop up at the top of the interface to help the user use the drawing tool.When the user clicks again, the help bar will disappear.



```
//Description:
//Click the help event.
//When you click this button,
//a message prompt window will be displayed on the drawing board
//to prompt the user how to use the drawing tool.
//Click the drawing board again and the prompt window will disappear.
private void TsbHelp_Click(object sender, EventArgs e)
{

    if (flag1)
    {
        RtbHelp.Visible = true;
        flag1 = false;
        flag2 = true;
    }
    else if (flag2)
    {
        RtbHelp.Visible = false;
        flag1 = true;
        flag2 = false;
    }

}
```

4.4.51 TsbOpen_Click(object sender, EventArgs e)

TsbOpen_ Click (object sender, EventArgs E) is a mouse click event. When the user clicks the tool strip button, the program will open a picture on the drawing board according to the user's selection. If there is already a picture on the current palette with traces of drawing, the user will be asked whether to save the current picture.

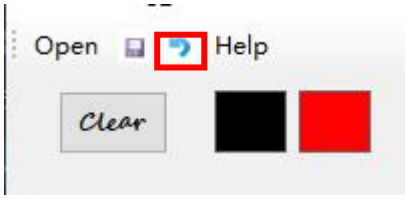



```
//Description:
    //Picture saving event: when the user clicks this button,
    //if a picture already exists on the palette,
    //the user will be prompted to save the existing picture first.
    //If yes is selected, the current picture will be saved first,
    //and then a new picture will be opened.
    //If you click No, a new picture will be opened directly.
    private void TsbOpen_Click(object sender, EventArgs e)
    {
```

```
                for (int i = 0; i < im.Width; i++)
                {
                    for (int j = 0; j < im.Height; j++)
                    {
                        if ((im.GetPixel(i, j).A + im.GetPixel(i, j).R + im.GetPixel(i,
j).G
                            + im.GetPixel(i, j).B) != 1020)
                        {
                            if (MessageBox.Show("Are you sure to save the
picture?", "Save file",
                                MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
                            {
                                TsbSave_Click(sender, e);
                                goto flag;
                            }
                            else
                            {
                                goto flag;
                            }
                        }
                    }
                }
            flag:
                OfdOpen.Filter = "BMP file|*.bmp|JPG file|*.jpg|PNG file|*.png|" +
                    "TIFF file|*.tif|GIF file|*.gif";
                OfdOpen.Title = "Open a picture";
                if (OfdOpen.ShowDialog() == DialogResult.OK)
                {
                    string filename = OfdOpen.FileName;
                    Image image = Image.FromFile(filename);
                    g.Clear(System.Drawing.Color.White);
                    g.DrawImage(image, 0, 0);
                    PcbDB.Image = im;
                    PcbDB.Refresh();
                }
            }
```

4.4.52 TsbUndo_Click(object sender, EventArgs e)

TsbUndo_ Click (object sender, EventArgs E) is a mouse click event. When the user clicks the tool strip button, the program will cancel the current track according to the existing drawing traces. If the current canvas is blank, undo cannot be used.




```
//Description:
        //Trace cancellation event.
        //Click this button to cancel the previous drawing process.
        //When there is no picture on the current screen,
        //clicking this button will no longer be available.
        private void TsbUndo_Click(object sender, EventArgs e)
        {
            if (undo.Count == 0)
            {
                TsbUndo.Enabled = false;
            }
            if (undo.Count != 0)
            {
                TsbUndo.Enabled = true;
                im = undo.Pop();
                g = Graphics.FromImage(im);
                PcbDB.Image = im;
                PcbDB.Refresh();
            }


        }
```

5. Use Ability Tests

5.1 Test 1

	Descriptions	Easy				Hard
1.	How easy to change the pen color.		√			
2.	How easy to change the brush color.		√			
3.	How easy to draw a picture.	√				
4.	How easy to change the Sketchpad size.	√				
5.	How easy to get the help of this tool.	√				
	Please list the APP can be improved in any ways.					
1.The color selection of pen is very convenient, but in the fast color selection, the color of pen cannot be reflected on the horizontal scroll bar in time. 2.The color selection of brush is very convenient, but in the fast color selection, the color of brush cannot be reflected on the horizontal scroll bar in time. 3.When drawing, the drawing track should be the track of the corresponding shape rather than the track of the mouse movement.						

Name of tester : Tu Liang
Date of test :2022-06-15
Test passed/failed : Failed

5.2 Test 2

	Descriptions	Easy				Hard
1.	How easy to use the eraser.	√				
2.	How easy to play the music.		√			
3.	How easy to toggle the stoke size.	√				
4.	How easy to toggle the eraser size.	√				
5.	How easy to fill the graphics.	√				
	Please list the APP can be improved in any ways.					
1. When the current music is played, the music player cannot automatically cycle.						

Name of tester : An Dawn
Date of test :2022-06-15
Test passed/failed : Failed

5.3 Test 3

	Descriptions	Easy				Hard
1.	How easy to save the picture		√			
2.	How easy to open a picture		√			
3.	How easy to execute the undo action	√				
4.	How easy to clear the screen	√				
5.	How timely is the status bar prompt message.	√				
	Please list the APP can be improved in any ways.					
1. If the canvas itself has no drawing trace or the drawing trace is cleared, the picture should not be saved. 2.If the canvas itself has traces of drawing, the program should ask the user whether to save the current picture before opening another picture.						

Name of tester : Niu Huashi
Date of test :2022-06-15
Test passed/failed : Failed

6. Recommendations And Reasons

6.1 Details For Code Fix 1

5.1	Descriptions	Easy				Hard
1.	How easy to change the pen color.		√			

	Recommendations	Reasons
1.	Create a new function that automatically synchronizes to the horizontal scroll bar after quickly adjusting the color of the pen.	The user indicates that the horizontal scroll bar cannot be updated in time during color switching of the pen.

Take the red color of pen as an example.

```
//Description:
//Quickly select the brush color event.
//Click this button, and the pen color will switch to Red,
//and the interface will display the current pen color.
1 个引用
private void BtnRed_Click(object sender, EventArgs e)
{
    pen.Color = Color.Red;
    color();

}
```

Reason for code that needs to be changed:
There is no code to adjust the value of the horizontal scroll bar with the quick selection of pen color.

How to fix:
Add a function to help change the value of the horizontal scroll bar.

Fixed code:

```
0 个引用
private void HScolor()
{
    HsbR.Value = pen.Color.R;
    HsbG.Value = pen.Color.G;
    HsbB.Value = pen.Color.B;

}
```

```
//Description:
//Quickly select the brush color event.
//Click this button, and the pen color will switch to Red,
//and the interface will display the current pen color.
1 个引用
private void BtnRed_Click(object sender, EventArgs e)
{
    pen.Color = Color.Red;
    color();
    HScolor();

}
```

Added code:

```
private void HScolor()
        {
            HsbR.Value = pen.Color.R;
            HsbG.Value = pen.Color.G;
            HsbB.Value = pen.Color.B;

        }
```

```
private void BtnRed_Click(object sender, EventArgs e)
        {
            pen.Color = Color.Red;
            color();
            HScolor();

        }
```

Test again:

5.1	Descriptions	Easy				Hard
1.	How easy to change the pen color.	√				

6.2 Details For Code Fix 2

5.1	Descriptions	Easy				Hard
2.	How easy to change the brush color.		√			

	Recommendations	Reasons
2.	Take red for example, because solidbrush cannot obtain its color, it can only obtain the color value externally and feed it back to the horizontal scroll bar.	The user indicates that the horizontal scroll bar cannot be updated in time during color switching of the brush.

```
}
//Description:
//Quickly select the brush color event,
//click this button, the brush color will switch to Red,
//and the interface will display the current brush color.
1 个引用
private void BtnRedF_Click(object sender, EventArgs e)
{
    brush = new SolidBrush(Color.Red);
    LblColor2.BackColor = (Color.Red);

}
```

Reason for code that needs to be changed:
The value of the brush color is not fed back to the horizontal scroll bar.

How to fix:
Directly make the value of the horizontal scroll bar equal to the RGB value of the color.

Fixed code:

```
//Description:
//Quickly select the brush color event,
//click this button, the brush color will switch to Red,
//and the interface will display the current brush color.
1 个引用
private void BtnRedF_Click(object sender, EventArgs e)
{
    brush = new SolidBrush(Color.Red);
    LblColor2.BackColor = (Color.Red);
    HsbRF.Value = Color.Red.R;
    HsbGF.Value = Color.Red.G;
    HsbBF.Value = Color.Red.B;
}
```

Added code:

```
private void BtnRedF_Click(object sender, EventArgs e)
        {
            brush = new SolidBrush(Color.Red);
            LblColor2.BackColor = (Color.Red);
            HsbRF.Value = Color.Red.R;
            HsbGF.Value = Color.Red.G;
            HsbBF.Value = Color.Red.B;
        }
```

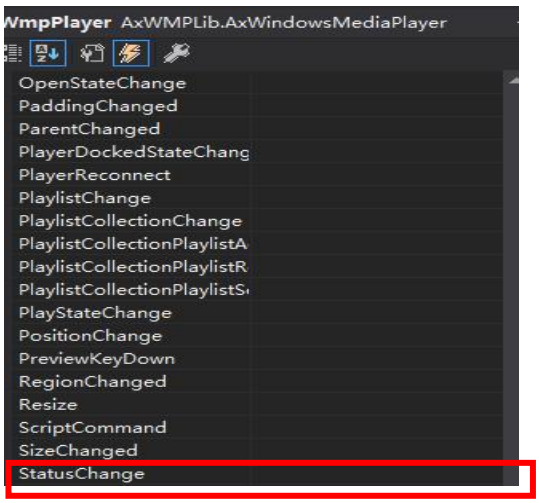
Test again:

5.1	Descriptions	Easy				Hard
2.	How easy to change the brush color.	√				

6.3 Details For Code Fix 3

5.2	Descriptions	Easy				Hard
2.	How easy to play the music.		√			

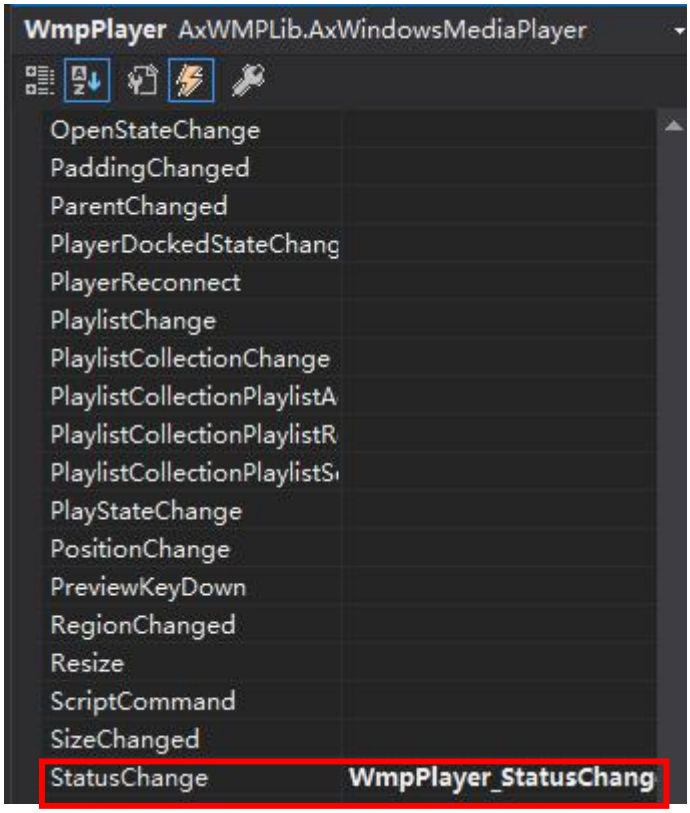
	Recommendations	Reasons
3.	Determine the playing status of the music player. When the music is played, repeat it at an interval of time.	The user indicates that the music player cannot play automatically.



Reason for code that needs to be changed:
The music player state change event was not invoked.

How to fix:
Call the music player state change event to judge the playing state of the music player. If the music player enters the stop state after the music is played, it will be played again after an interval of one second.

Fixed code:




```
//Description:
//Event of music player status change.
//If the current music is played,
//the music player will repeat the second time after 1 second.
1 个引用
private void WmpPlayer_StatusChange(object sender, EventArgs e)
{
    if ((int)WmpPlayer.playState == 1)
    {
        System.Threading.Thread.Sleep(1000);
        WmpPlayer.Ctlcontrols.play();
    }
}
```

Added code:

```
private void WmpPlayer_StatusChange(object sender, EventArgs e)
        {
            if ((int)WmpPlayer.playState == 1)
```

```
        {
            System.Threading.Thread.Sleep(1000);
            WmpPlayer.Ctlcontrols.play();
        }
    }
```

Test again:

5.2	Descriptions	Easy				Hard
2.	How easy to play the music.	√				

6.4 Details For Code Fix 4

5.3	Descriptions	Easy				Hard
1.	How easy to save the picture		√			

	Recommendations	Reasons
4.	Determine each pixel on the drawing board. If it is determined that there is color on the drawing board, the picture can be saved.	The user indicates that pictures without any trace of drawing should not be saved.

```
//Description:
//Save the current picture event.
//When you click this button,
//if the picture on the canvas is blank,
//it will not be saved. If it is a picture with drawing traces,
//it will be saved and saved in a specific format.
2 个引用
private void TsbSave_Click(object sender, EventArgs e)
{

        SfdSave.Filter = "BMP file|*.bmp|JPG file|*.jpg|PNG file|*.png|" +
    "TIFF file|*.tif|GIF file|*.gif";
        SfdSave.Title = "Save this file";

        if (SfdSave.ShowDialog() == DialogResult.OK)
        {
            string filename = SfdSave.FileName;
            string filetext = filename.Remove(0, filename.Length - 3);
            switch (filetext)
            {
                case "bmp":
                    im.Save(filename, System.Drawing.Imaging.ImageFormat.Bmp);
                    break;
                case "jpg":
                    im.Save(filename, System.Drawing.Imaging.ImageFormat.Jpeg);
                    break;
                case "png":
                    im.Save(filename, System.Drawing.Imaging.ImageFormat.Png);
                    break;
                case "tif":
                    im.Save(filename, System.Drawing.Imaging.ImageFormat.Tiff);
                    break;
                case "gif":
                    im.Save(filename, System.Drawing.Imaging.ImageFormat.Gif);
                    break;
            }


        }
```

Reason for code that needs to be changed:
The canvas is not detected, so it is impossible to judge whether the canvas contains drawing traces.

How to fix:
Detect each pixel of the canvas. Once a trace of drawing is detected at any point, the Save option pops up.

Fixed code:

```
2 个引用
private void TsbSave_Click(object sender, EventArgs e)
{
    PcbDB.Refresh();
    for (int i = 0; i < im.Width; i++)
    {
        for (int j = 0; j < im.Height; j++)
        {
            if ((im.GetPixel(i, j).A + im.GetPixel(i, j).R + im.GetPixel(i, j).G
                + im.GetPixel(i, j).B) != 1020)
            {
                SfdSave.Filter = "BMP file|*.bmp|JPG file|*.jpg|PNG file|*.png|" +
            "TIFF file|*.tif|GIF file|*.gif";
                SfdSave.Title = "Save this file";

                if (SfdSave.ShowDialog() == DialogResult.OK)
                {
                    string filename = SfdSave.FileName;
                    string filetext = filename.Remove(0, filename.Length - 3);
                    switch (filetext)
                    {
                        case "bmp":
                            im.Save(filename, System.Drawing.Imaging.ImageFormat.Bmp);
                            break;
                        case "jpg":
                            im.Save(filename, System.Drawing.Imaging.ImageFormat.Jpeg);
                            break;
                        case "png":
                            im.Save(filename, System.Drawing.Imaging.ImageFormat.Png);
                            break;
                        case "tif":
                            im.Save(filename, System.Drawing.Imaging.ImageFormat.Tiff);
                            break;
                        case "tif":
                            im.Save(filename, System.Drawing.Imaging.ImageFormat.Tiff);
                            break;
                        case "gif":
                            im.Save(filename, System.Drawing.Imaging.ImageFormat.Gif);
                            break;
                    }
                    goto flag;

                }
                else
                    goto flag;
            }

        }
    }
flag:;

}
```

Added code:

```
private void TsbSave_Click(object sender, EventArgs e)
        {
```

```
                PcbDB.Refresh();
                for (int i = 0; i < im.Width; i++)
                {
                    for (int j = 0; j < im.Height; j++)
                    {
                        if ((im.GetPixel(i, j).A + im.GetPixel(i, j).R + im.GetPixel(i,
j).G
                            + im.GetPixel(i, j).B) != 1020)
                        {
                            SfdSave.Filter = "BMP file|*.bmp|JPG file|*.jpg|PNG
file|*.png|" +
                         "TIFF file|*.tif|GIF file|*.gif";
                            SfdSave.Title = "Save this file";

                            if (SfdSave.ShowDialog() == DialogResult.OK)
                            {
                                string filename = SfdSave.FileName;
                                string filetext = filename.Remove(0,
filename.Length - 3);
                                switch (filetext)
                                {
                                    case "bmp":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Bmp);
                                        break;
                                    case "jpg":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Jpeg);
                                        break;
                                    case "png":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Png);
                                        break;
                                    case "tif":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Tiff);
                                        break;
                                    case "gif":
                                        im.Save(filename,
System.Drawing.Imaging.ImageFormat.Gif);
                                        break;
                                }
                                goto flag;
```

```
                    }
                    else
                        goto flag;
                }

            }
        }
    flag:;

    }
```

Test again:

5.3	Descriptions	Easy				Hard
1.	How easy to save the picture	√				

6.5 Details For Code Fix 5

5.3	Descriptions	Easy				Hard
2.	How easy to open a picture		√			

	Recommendations	Reasons
5.	Detect each pixel. If the user clicks the open picture button, when any pixel is detected to have traces of drawing, the user will be prompted whether to save the current picture. Click Yes to save the picture. Click No， nothing will happen. Then start the operation of opening the picture.	The user indicates that if the user clicks to open the picture, the canvas will be cleared regardless of whether there is any trace of drawing on the current canvas. This is very inconvenient for users.

```
//Description:
//Picture saving event: when the user clicks this button,
//if a picture already exists on the palette,
//the user will be prompted to save the existing picture first.
//If yes is selected, the current picture will be saved first,
//and then a new picture will be opened.
//If you click No, a new picture will be opened directly.
1 个引用
private void TsbOpen_Click(object sender, EventArgs e)
{

    OfdOpen.Filter = "BMP file|*.bmp|JPG file|*.jpg|PNG file|*.png|" +
        "TIFF file|*.tif|GIF file|*.gif";
    OfdOpen.Title = "Open a picture";
    if (OfdOpen.ShowDialog() == DialogResult.OK)
    {
        string filename = OfdOpen.FileName;
        Image image = Image.FromFile(filename);
        g.Clear(System.Drawing.Color.White);
        g.DrawImage(image, 0, 0);
        PcbDB.Image = im;
        PcbDB.Refresh();
    }
}
```

Reason for code that needs to be changed:
The canvas is not detected, so it is impossible to judge whether the canvas contains drawing traces.

How to fix:
Detects each pixel of the canvas. Once the drawing trace is detected at any point, the user will be prompted whether to save the current picture when clicking open again.

Fixed code:

```
 private void TsbOpen_Click(object sender, EventArgs e)
            {
                for (int i = 0; i < im.Width; i++)
                {
                    for (int j = 0; j < im.Height; j++)
                    {
                        if ((im.GetPixel(i, j).A + im.GetPixel(i, j).R + im.GetPixel(i,
j).G
                            + im.GetPixel(i, j).B) != 1020)
                        {
                            if (MessageBox.Show("Are you sure to save the
picture?", "Save file",
                                MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
                            {
                                TsbSave_Click(sender, e);
                                goto flag;
                            }
                            else
                            {
                                goto flag;
                            }
                        }
                    }
                }
            flag:
                OfdOpen.Filter = "BMP file|*.bmp|JPG file|*.jpg|PNG file|*.png|" +
                    "TIFF file|*.tif|GIF file|*.gif";
                OfdOpen.Title = "Open a picture";
                if (OfdOpen.ShowDialog() == DialogResult.OK)
                {
                    string filename = OfdOpen.FileName;
                    Image image = Image.FromFile(filename);
                    g.Clear(System.Drawing.Color.White);
                    g.DrawImage(image, 0, 0);
```

```
                PcbDB.Image = im;
                PcbDB.Refresh();
            }
        }
```

Test again:

5.3	Descriptions	Easy				Hard
2.	How easy to open a picture	√				

7. History Of Changes

History of program:

Version	Description	Filename	Date
1	The RGB value of the quick color selection of pen in this program cannot be fed back to the horizontal scroll bar in time.	Outcome3.exe	01-06-2022
2	This program will stop automatically after the music player plays the current music, and cannot cycle.	Outcome3.exe	01-06-2022
3	The RGB value of the fast color selection of this program brush cannot be fed back to the horizontal scroll bar in time.	Outcome3.exe	01-06-2022

History of document:

Version	Description	Filename	Date
1	The document contains only the requirements analysis.	Outcome3(1).doc	02-06-2022
2	The document only contains requirements analysis, interface design and source code.	Outcome3(2).doc	03-06-2022
3	The document only contains the requirements analysis, interface design and source code, as well as the description of key codes.	Outcome3(3).doc	03-06-2022

8. References

1. Willis(2012), *Introduction to embedding and playing wav audio files in c#*[online]. Available from: https://blog.csdn.net/stormwy/article/details/7901944 (Accessed: 25 May 2022).

2. Guigen, S(2021), *C#graphics image management and common usage* [online]. Available from: https://blog.csdn.net/better_off/article/details/116375308 (Accessed: 25 May 2022).

3.Azure,Q(2021), *C#bitmap save* [online]. Available from: https://blog.csdn.net/better_off/article/details/116375308 (Accessed: 25 May 2022).

4. Biye,Z (2021), *Design of simple drawing tool based on c#* [online]. Available from: https://blog.csdn.net/newlw/article/details/124295574 Accessed: 25 May 2022).