KAI JIE HI 100012607

Pass Task 9 - Shape Drawer

Related Learning Outcomes

ULO1 – Explain the OO Principles

This exercise demonstrated object encapsulation

ULO2 – Use OO Language and Library

Demonstrated class and constructor declaration, the use of conditional statements (e.g. “if”), and assigning values to parameters. Demonstrated the use of readonly, use of default constructor, and the use of List<> by accessing the library

ULO3 – Design, Develop and Test using an IDE

The code was developed using Xamarin Studio to build and run the program, as well as integrated debugging features to step and inspect values.

ULO4 – Communicate using UML Diagrams

I learned how to interpret a UML class diagram and write the related code.

ULO5 – Describe Elements of Good OO Design

The exercise demonstrated correct use of C# coding conventions.

Screenshots

[code running]
TEST

```
using NUnit.Framework;
using System;
using Color = System.Drawing.Color;
using SwinGameSDK;

namespace MyGame
{
    [TestFixture ()]
    public class TestDefaultDrawing
    {
        [Test ()]
        public void TestDefaultinitialisation ()
        {
            Drawing Draw =new Drawing ();
            Assert.IsTrue (Draw.Mybackground == Color.White);
        }
        [Test ()]
        public void testwithcolor ()
        {
            Drawing Draw =new Drawing (Color.Blue);
            Assert.IsTrue (Draw.Mybackground == Color.Blue);
        }

        [ Test()]
        public void TestAddShape()
        {
            Drawing myDrawing= new Drawing();
            int count = myDrawing.ShapeCount;

            Assert.AreEqual (0, count, "Drawing should start with no shap
es");

            myDrawing.AddShape (new Shape());
            myDrawing.AddShape (new Shape());
            count = myDrawing.ShapeCount;

            Assert.AreEqual (2, count, "Adding two shapes should increase
 the count to 2");
        }

        [ Test()]
        public void Shapetest()
        {
            Shape st = new Shape (Color.Blue, 0, 0, 0, 0);
            st.X = 25;
            st.Y = 25;
            st.Width = 100;
            st.Height = 100;


            Assert.IsTrue (st.IsAt(SwinGame.PointAt(50,50)) );
            Assert.IsTrue (st.IsAt(SwinGame.PointAt(25,25)) );
            Assert.IsFalse (st.IsAt(SwinGame.PointAt(10,50)) );
            Assert.IsFalse (st.IsAt(SwinGame.PointAt(50,10)) );
        }
        public void ResizedShapetest()
        {
            Shape rst = new Shape (Color.Blue, 0, 0, 50, 50);
            rst.X = 25;
            rst.Y = 25;
            rst.Width = 100;
            rst.Height = 100;



            Assert.IsTrue (rst.IsAt(SwinGame.PointAt(25,25)) );
            Assert.IsTrue (rst.IsAt(SwinGame.PointAt(25,25)) );
        }

    }
}
```

DRAWING

```
using Color = System.Drawing.Color;
using SwinGameSDK;

namespace MyGame
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;

        public Drawing (Color background)
        {
            _shapes = new List<Shape> ();
            _background = background;
        }

        public Drawing () : this (Color.White)
        {
            //Something come here
        }

        public Color Mybackground
        {
            get
            {
                return _background;
            }
            set
            {
                _background = value;
            }

        }

        public int ShapeCount
        {
            get {return _shapes.Count; }
        }

        public void AddShape(Shape s)
        {
            _shapes.Add(s);
        }

        public void Draw()
        {
            SwinGame.ClearScreen (Mybackground);
            foreach (Shape s in _shapes)
            {
                s.Drawshape ();
            }
        }
        public List <Shape> SelectedShape
        {
            get
            {
                List<Shape> ListofSelectedShape = new List<Shape> ();
                foreach (Shape s in _shapes)
                {
                    if (s.Selected)
                        ListofSelectedShape.Add (s);
                    else
                        ListofSelectedShape.Remove (s);
                }
                return ListofSelectedShape;
            }

        }

        public void SelectShapeAt (Point2D pt)
        {
            foreach (Shape s in _shapes)
            {
                if (s.IsAt(pt))
                {
                    s.Selected = true;
                }
                else
                {
                    s.Selected = false;
                }
            }
        }
        public void RemoveShape (Shape s)
        {
            _shapes.Remove (s);
        }
    }
}
```

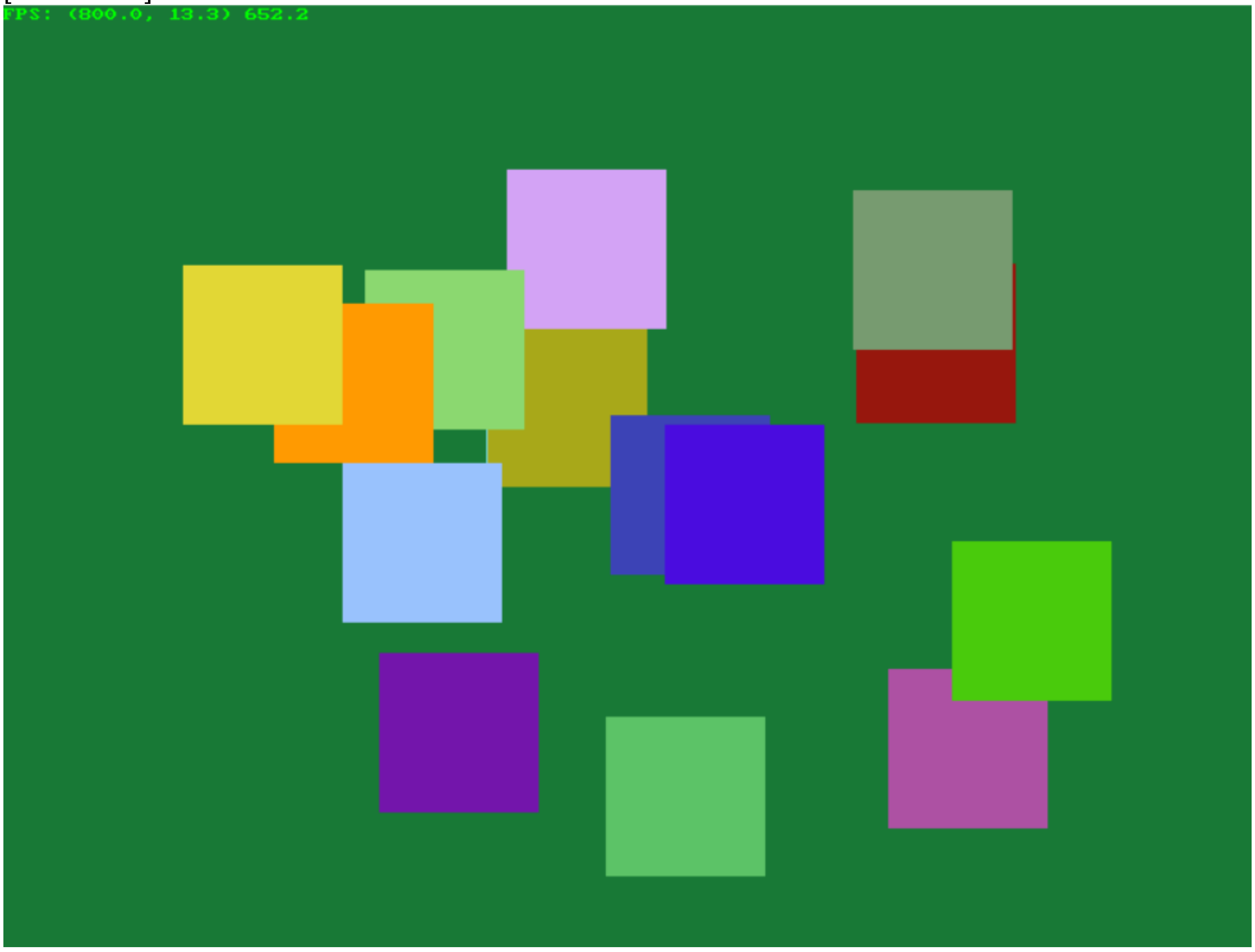
SHAPE CLASS

```
using System;
using Color = System.Drawing.Color;
using SwinGameSDK;
namespace MyGame
{
    public class Shape
    {
        private Color _color;
        private float _x, _y;
        private int _width, _height;
        private bool _selected;
        /// <summary>
        /// Initializes a new instance of the <see cref="MyGame.Shape"/>
class.
        /// </summary>
        /// <param name="clr">Clr.</param>
        /// <param name="x">The x coordinate.</param>
        /// <param name="y">The y coordinate.</param>
        /// <param name="width">Width.</param>
        /// <param name="height">Height.</param>
        public Shape (Color clr,float x, float y, int width, int height)
        {
            _color = clr;
            _x = x;
            _y = y;
            _width = width;
            _height = height;
        }
        /// <summary>
        /// Initializes a new instance of the <see cref="MyGame.Shape"/>
class.
        /// </summary>
        public Shape() : this(Color.AliceBlue, 0,0,100,100)
        {

        }
        //draw shape to the screen
        public void Drawshape()
        {
            if (_selected)
            {
                DrawOutline ();
            }
            SwinGame.FillRectangle (_color,_x,_y, _width, _height);
        }

        public bool IsAt(Point2D pt)
        {
            return (SwinGame.PointInRect (pt, _x, _y, _width, _height));
        }

        public Color Color
        {
            get { return _color; }
            set { _color = value; }
        }

        public float X
        {
            get { return _x;}
            set {_x = value;}
        }
        public float Y
        {
            get { return _y;}
            set {_y = value;}
        }
        public int Width
        {
            get { return _width;}
            set {_width = value;}
        }
        public int Height
        {
            get { return _height;}
            set {_height = value;}
        }

        public bool Selected
        {
            get { return _selected;}
            set{ _selected = value;}
        }
        public void DrawOutline ()
        {
            SwinGame.DrawRectangle (Color.Black, _x - 2, _y + 2, _width +
 4, _height - 4);
        }
    }
}
```

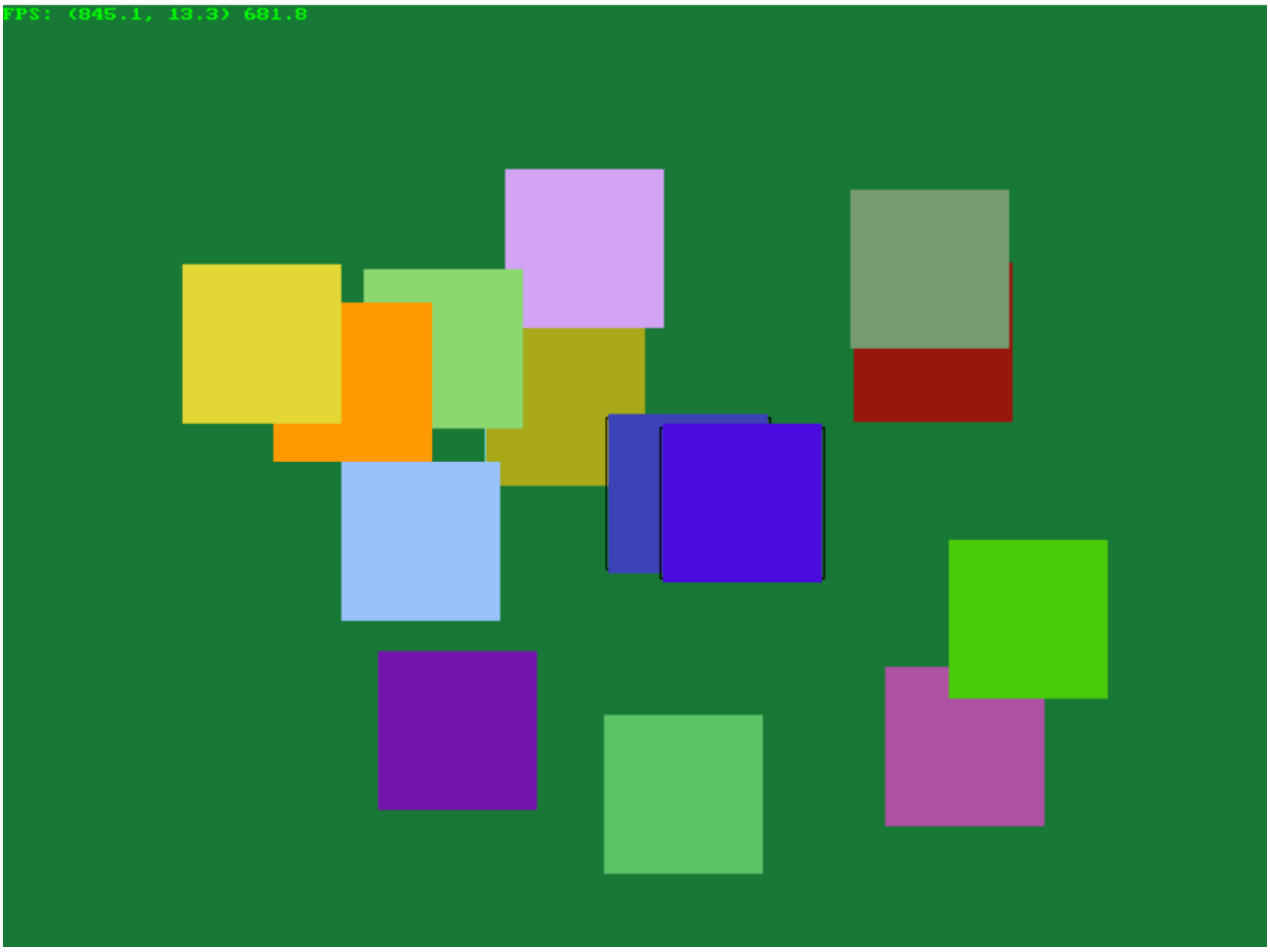
MAIN CLASS

```
using System;
using System.Reflection;
using SwinGameSDK;
using Color = System.Drawing.Color;
using System.Collections.Generic;

namespace MyGame
{
    public class GameMain
    {
        public static void Main()
        {
            SwinGame.OpenAudio ();
            SwinGame.OpenGraphicsWindow ("GameMain", 800, 600);
            SwinGame.ShowSwinGameSplashScreen ();

            Drawing myDrawing = new Drawing();
            while(false == SwinGame.WindowCloseRequested())
            {
                SwinGame.ProcessEvents();

                SwinGame.ClearScreen(Color.White);
                SwinGame.DrawFramerate(0,0);

                myDrawing.Draw ();

                Point2D mouseLocation = SwinGame.MousePosition();

                if (SwinGameSDK.Input.MouseClicked(MouseButton.LeftButton
))
                {
                    Shape myShape = new Shape();
                    myShape.Color = SwinGame.RandomRGBColor (255);
                    myShape.X = SwinGame.MouseX();
                    myShape.Y = SwinGame.MouseY();
                    myDrawing.AddShape (myShape);
                }

                if (SwinGame.MouseClicked (MouseButton.RightButton))
                {
                    myDrawing.SelectShapeAt (SwinGame.MousePosition());
                }
                if (SwinGame.KeyTyped(KeyCode.vk_DELETE))
                {
                    List<Shape> selected = myDrawing.SelectedShape;
                    foreach (Shape s in selected)
                    {
                        myDrawing.RemoveShape(s);
                    }
                }

                //if (myShape.IsAt (mouseLocation))
                //{
                    if (SwinGame.KeyTyped (KeyCode.vk_SPACE))
                    {
                        myDrawing.Mybackground = SwinGame.RandomRGBColor(
255);

                    }
                //}
                SwinGame.DrawFramerate(0,0);
                SwinGame.RefreshScreen ();

            }
            //End the audio
            SwinGame.CloseAudio();

            //Close any resources we were using
            SwinGame.ReleaseAllResources();
        }
    }
}
```

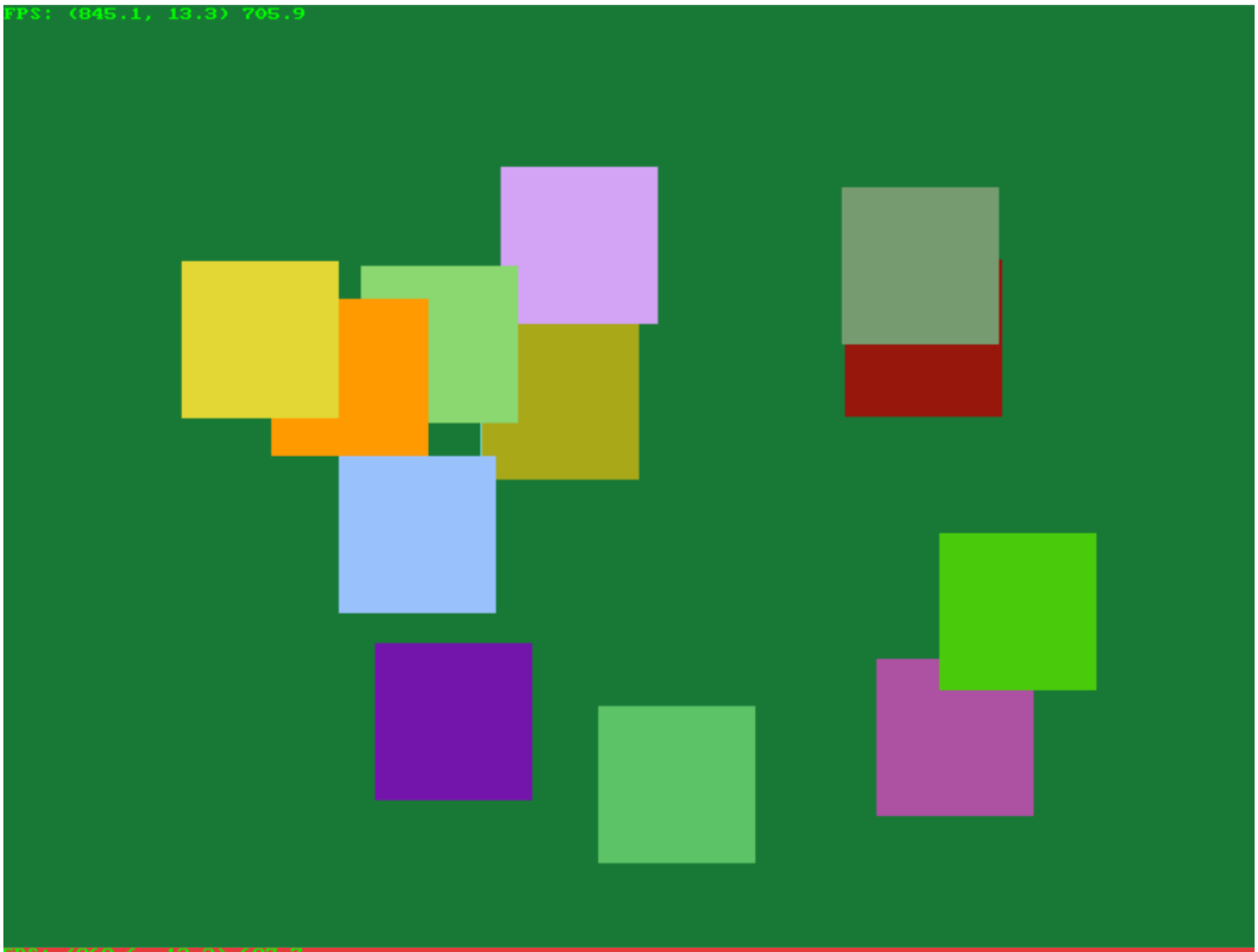
[use of IDE]



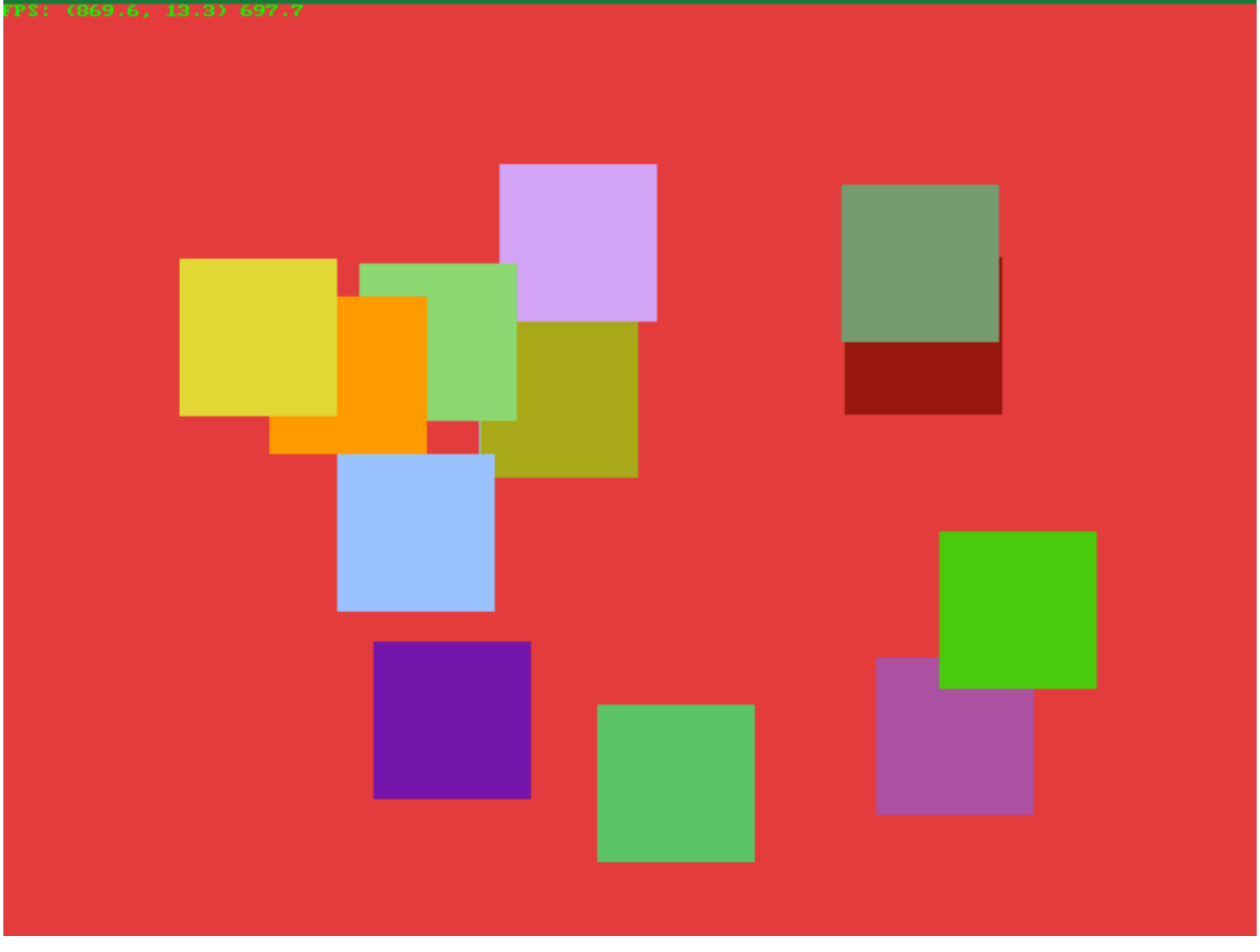
```
FPS: (800.0, 13.3) 652.2
```



```
FPS: (845.1, 13.3) 681.8
```



```
FPS: (845.1, 13.3) 705.9
```



```
FPS: (869.6, 13.3) 697.7
```