KWANGWOON UNIVERSITY

응 용 소 프 트 웨 어 실 습

Homework #3. MDI 그림판

담당 교수님 : 이윤구 교수님

담당 조교님 : 윤여경 조교님

이훈민 조교님

제 출 날 짜 : 2018. 06. 21.

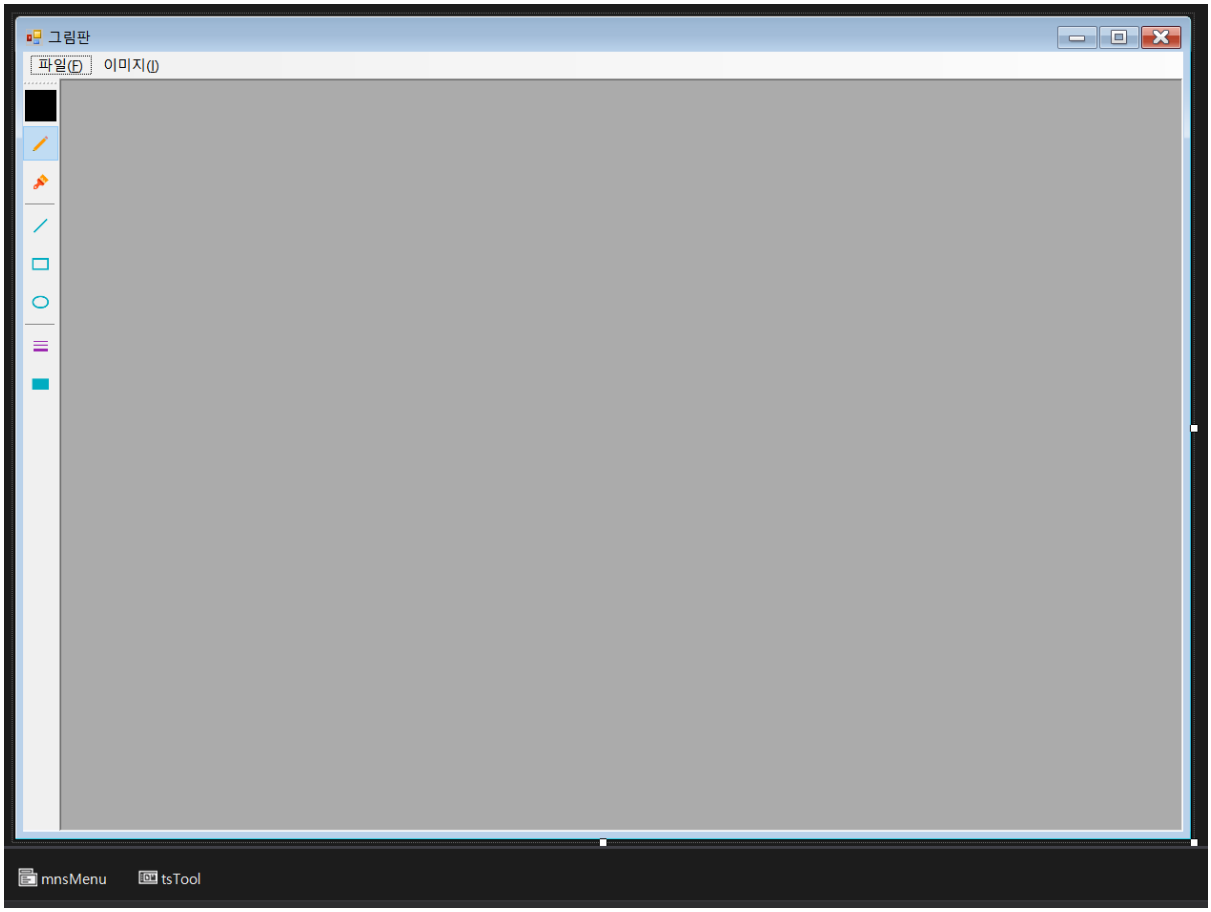
학 과 : 컴퓨터소프트웨어학과

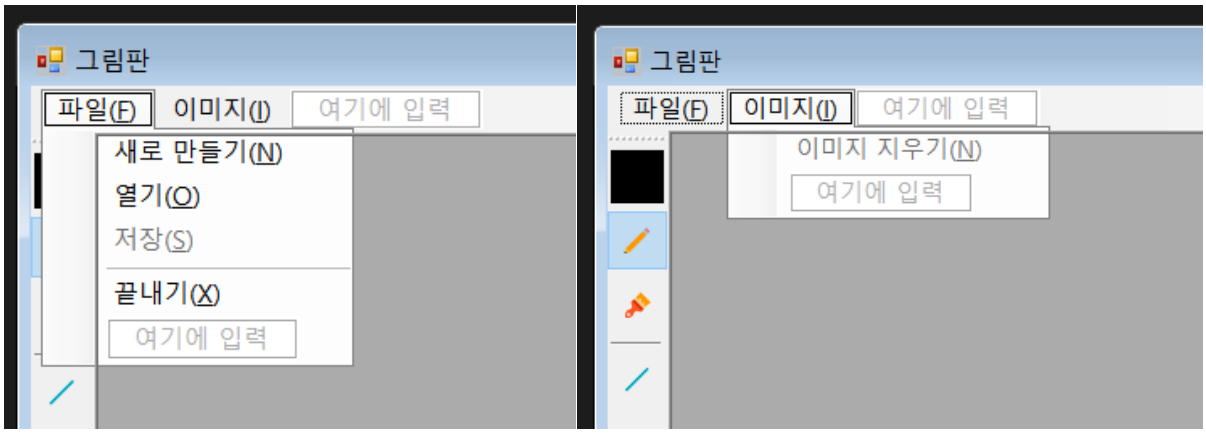
학 번 : 2014707040

이 름 : 유 진 혁

항목 (+보고서 페이지) [x 는 작성하지 못한 경우]	슬라이드 번호	배점 (작성여부)
1.1. 부모 Form 구성 (1점)	3	✓
1.2. 자식 Form 구성 (0.5점)	4	✓
2.1. 새로 만들기 (0.5 점)	4-5	✓
2.2. 열기 (0.5 점)	5-6	✓
2.3. 저장 (0.5 점)	7-8	✓
3.1. 색 (0.5 점)	8	✓
3.2. 연필 (0.5 점)	9-11	✓
3.3. 브러쉬 (0.5 점)	11	✓
4.1. 직선 (0.5 점)	12-13	✓
4.2. 사각형 (0.5 점)	13-15	✓
4.3. 타원 (0.5 점)	15-17	✓
5.1. 선 두께 (1.5 점)	17-18	✓
5.2. 채우기 (1.5 점)	18-19	✓
6.1. 이미지 지우기 (1 점)	19	✓
추가 1. 더블 버퍼링 (1 점)	20	✓
추가 2. 다양한 형식의 이미지 열기 (1 점)	20	✓
추가 3. 다양한 형식의 이미지 저장 (1 점)	21	✓

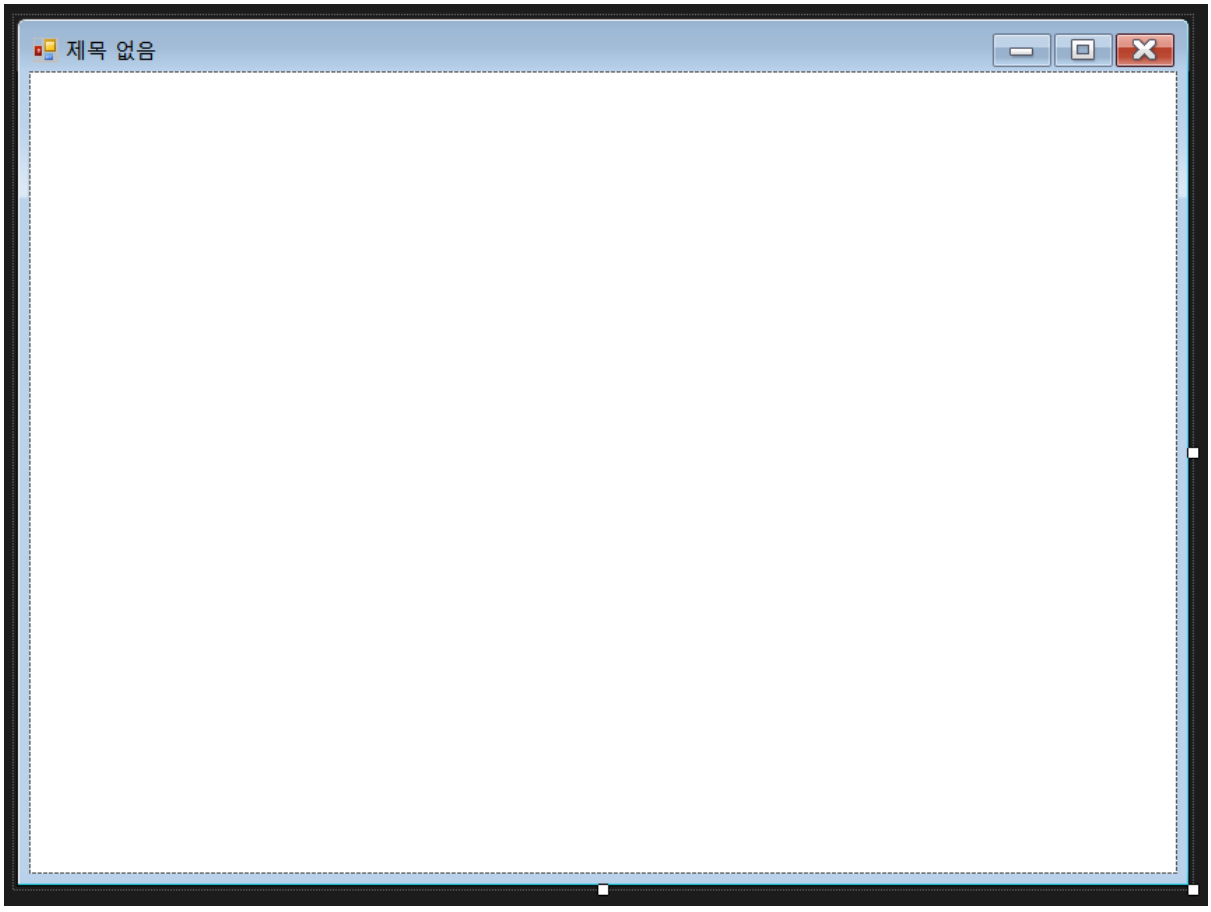
1.1. 부모 Form 구성 (1 점)





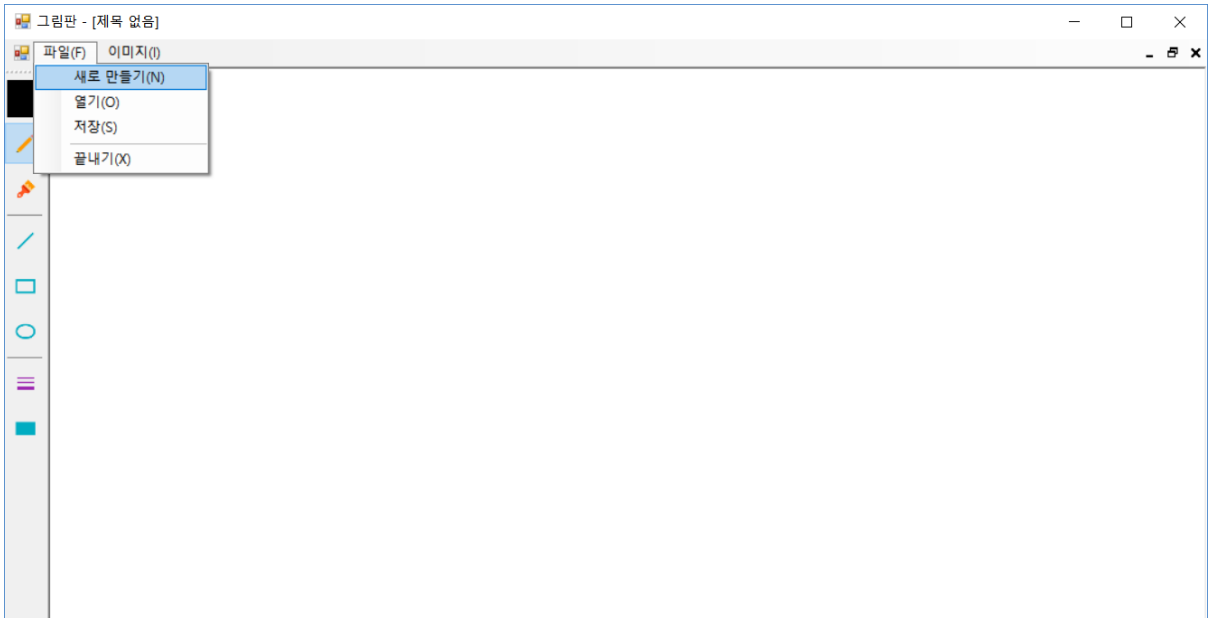
저장 버튼과 이미지 지우기 버튼은 비활성이 기본값이고, 좌측 ToolStrip 버튼은 연필이 기본으로 체크되게 했다.

1.2. 자식 Form 구성 (0.5 점)



자식 Form 의 Text 속성은 "제목 없음"이 기본값이다.

2.1. 새로 만들기 (0.5 점)

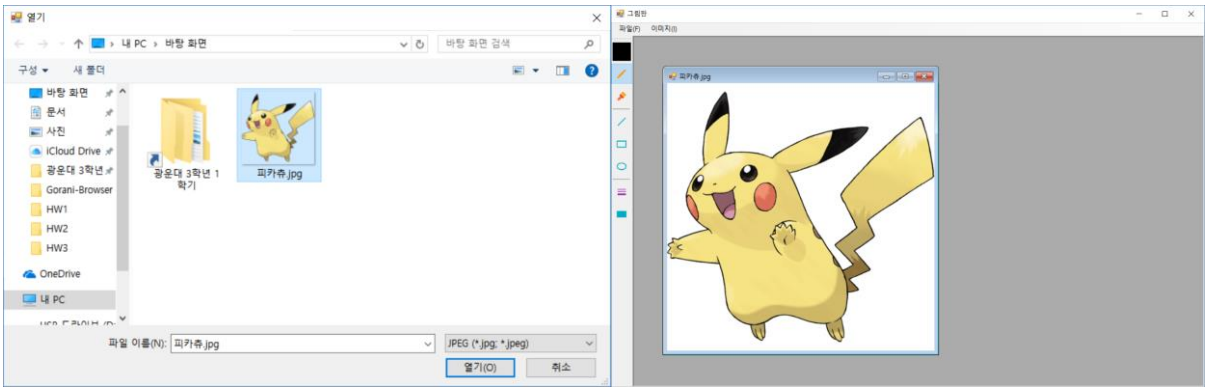


```
// 새로 만들기
private void tsmiNew_Click(object sender, EventArgs e)
{
    frmChild child = new frmChild();    // 자식 Form 생성
    if (MdiChildren.Length == 0)    // 자식 Form 처음 생성될 때
    {
        child.WindowState = FormWindowState.Maximized;  // 창 최대화
        tsmiSave.Enabled = true;    // 저장 버튼 활성화
        tsmiClear.Enabled = true;   // 이미지 지우기 버튼 활성화
    }
    else
    {
        // 열려 있는 자식 Form들 창 크기 기본으로
        foreach (Form form in MdiChildren)
            form.WindowState = FormWindowState.Normal;
    }
    child.MdiParent = this; // 부모 지정
    child.Show();
}
```

<부모 Form 코드>

새로 만들기 ToolStripMenuItem 을 클릭하면 자식 Form 객체를 생성한다. MDI 자식 Form 의 개수를 확인하여 자식 Form 이 없으면 창 상태를 최대화 상태로 설정하고 저장 버튼과 이미지 지우기 버튼을 활성화한다. 열어 놓은 MDI 자식 Form 이 있다면, 열려 있는 MDI 자식 Form 들의 창 상태를 기본으로 바꾼다. 자식 Form 의 MDI 부모 Form 을 현재 Form 으로 지정하고 자식 Form 을 띄운다.

2.2. 열기 (0.5 점)




```
// 그리기 함수
private void pnlBoard_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    g.SmoothingMode = SmoothingMode.AntiAlias; // 앤티 앨리어싱 적용

    img = new Bitmap(img);  // 이미지 새로고침
    g.DrawImage(img, new Point(0, 0));  // 이미지 그리기
```

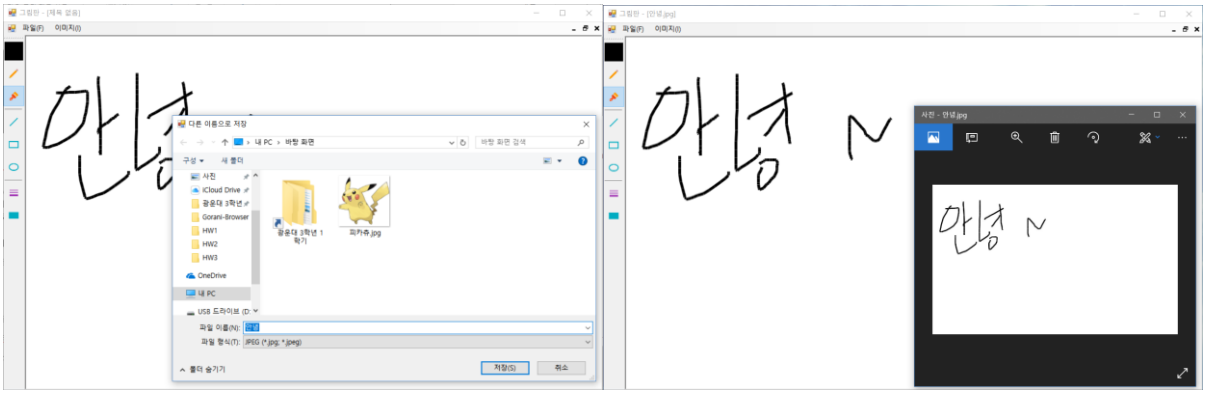
<자식 Form 코드>

```
// 열기
private void tsmiOpen_Click(object sender, EventArgs e)
{
    OpenFileDialog ofdOpen = new OpenFileDialog(); // Dialog 생성
    // 확장자 필터 설정
    ofdOpen.Filter = "JPEG (*.jpg; *.jpeg)|*.jpg; *jpeg|"
        + "비트맵 파일 (*.bmp)|*.bmp |"
        + "GIF (*.gif)|*.gif|"
        + "PNG (*.PNG)|*.png|"
        + "모든 그림 파일 (*.jpg; *.jpeg; *.bmp; *.gif; *.png)|*.jpg; *.jpeg; *.bmp; *.gif; *.png";

    if (ofdOpen.ShowDialog() == DialogResult.OK)
    {
        frmChild child = new frmChild();    // 자식 Form 생성
        child.img = Image.FromFile(ofdOpen.FileName);   // 이미지 열기
        if (MdiChildren.Length == 0)    // 자식 Form 처음 생성될 때
        {
            tsmiSave.Enabled = true;    // 저장 버튼 활성화
            tsmiClear.Enabled = true;   // 이미지 지우기 버튼 활성화
        }
        else
        {
            // 열려 있는 자식 Form 창 크기 기본으로
            foreach (Form form in MdiChildren)
                form.WindowState = FormWindowState.Normal;
        }
        child.MdiParent = this; // 부모 지정
        child.Size = new Size(child.img.Width + 16, child.img.Height + 39); // Form 크기 설정
        child.Text = Path.GetFileName(ofdOpen.FileName);    // Form Text 설정
        child.Show();
    }
}
```

<부모 Form 코드>

열기 ToolStripMenuItem 을 클릭하면 OpenFileDialog 를 생성하고 확장자 필터를 설정한다. Dialog 창을 띄우고 확인 버튼 입력이 들어오면 자식 Form 을 생성한 뒤, 자식 Form 의 Image 형 멤버 변수 img 에 선택한 파일을 읽어온다. MDI 자식 Form 의 개수를 확인하여 처음 생성하는 거라면 저장 버튼과 이미지 버튼을 활성화하고, 아니라면 열려 있는 MDI 자식 Form 의 창 상태를 기본으로 변경한다. MDI 자식 Form 의 부모를 현재 Form 으로 지정하고 Form 크기와 Text 속성을 설정한 뒤, Form 창을 띄운다. 자식 Form 은 img 변수에 있는 그림을 가져와 Graphics 객체로 Panel 에 그린다.

2.3. 저장 (0.5 점)

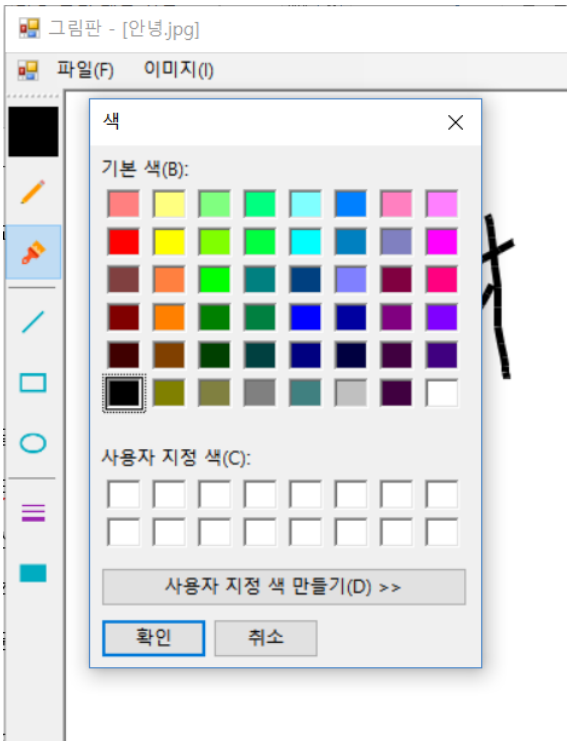


```
// 저장
private void tsmiSave_Click(object sender, EventArgs e)
{
    SaveFileDialog sfdSave = new SaveFileDialog();  // Dialog 생성
    // 확장자 필터 설정
    sfdSave.Filter = "JPEG (*.jpg; *.jpeg)|*.jpg; *jpeg|"
        + "비트맵 파일 (*.bmp)|*.bmp |"
        + "GIF (*.gif)|*.gif|"
        + "PNG (*.PNG)|*.png";

    if(sfdSave.ShowDialog() == DialogResult.OK)
    {
        Panel panel = (ActiveMdiChild as frmChild).board;   // 활성화된 자식 Form의 패널
        Image img = (ActiveMdiChild as frmChild).img;   // 활성화된 자식 Form의 img 변수
        Bitmap bmp = new Bitmap(panel.Width, panel.Height); // 이미지 담을 bitmap 생성
        panel.DrawToBitmap(bmp, new Rectangle(0, 0, panel.Width, panel.Height));    // 패널 이미지 bitmap에 저장

        if (img != null) img.Dispose(); // 이미지 리소스 해제

        try
        {
            // 이미 있는 파일 덮어쓰기
            if (File.Exists(sfdSave.FileName))
                File.Delete(sfdSave.FileName);

            switch (sfdSave.FilterIndex)
            {
                case 1:
                    bmp.Save(sfdSave.FileName, ImageFormat.Jpeg);   // JPG로 저장
                    break;

                case 2:
                    bmp.Save(sfdSave.FileName, ImageFormat.Bmp);    // BMP로 저장
                    break;

                case 3:
                    bmp.Save(sfdSave.FileName, ImageFormat.Gif);    // GIF로 저장
                    break;
                case 4:
                    bmp.Save(sfdSave.FileName, ImageFormat.Png);    // PNG로 저장
                    break;
                default:
                    break;
            }
        }
        finally { bmp.Dispose(); } // bitmap 리소스 해제
        ActiveMdiChild.Text = Path.GetFileName(sfdSave.FileName);   // 자식 Form Text 변경
        img = Image.FromFile(sfdSave.FileName); // 저장한 이미지로 다시 열기
    }
}
```

<부모 Form 코드>

저장 ToolStripMenuItem 을 클릭하면, SaveFileDialog 를 생성하고 저장 확장자 필터를 설정한다. Dialog 창을 띄우고 확인 버튼 입력이 들어오면 활성화된 MDI 자식 Form 의 Panel 과 img 변수를 가져오고, 이미지를 담을 Bitmap 객체를 생성한다. 자식 Form 의 Panel 을 Bitmap 객체 담고 img 변수에 담긴 이미지 리소스를 일시적으로 해제한다. 저장하려는 이름으로 해당 파일이 이미 존재하나 확인하고 존재한다면 기존 파일을 삭제한다. 확장자 필터 인덱스 순서에 따라 Bitmap 객체에 담긴 이미지를 각 확장자에 맞게 지정 경로에 저장한다. Bitmap 객체의 리소스를 해제하고 자식 Form 의 Text 속성을 변경한 뒤 img 변수를 저장한 이미지로 다시 설정한다.

3.1. 색 (0.5 점)

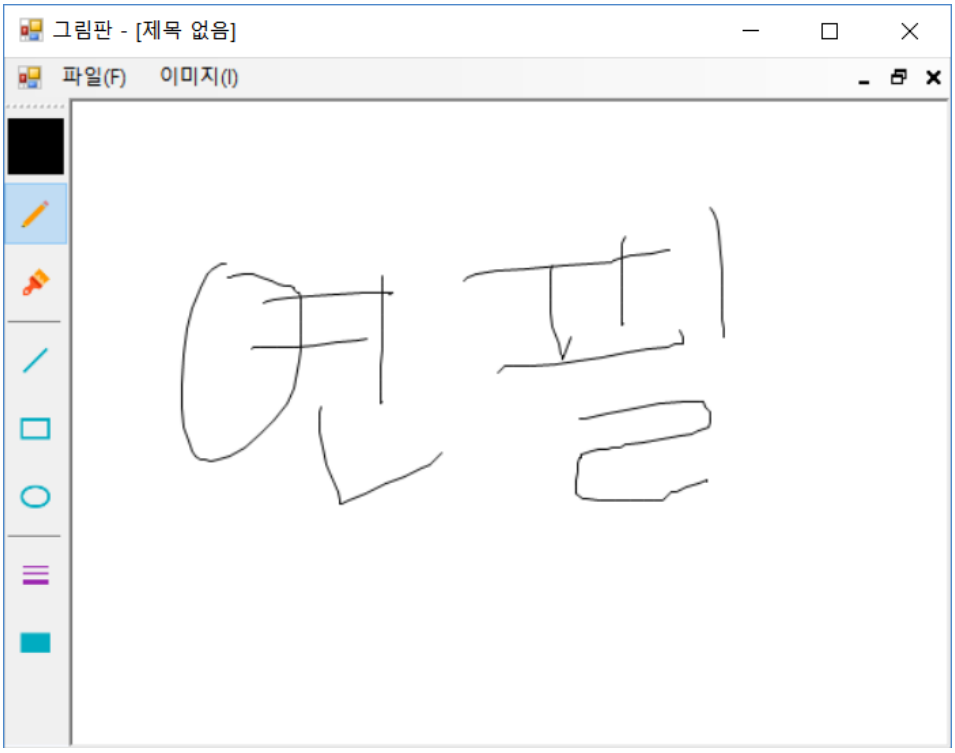



```
// 색 변경
private void tsbtnColor_Click(object sender, EventArgs e)
{
    ColorDialog cldColor = new ColorDialog();   // Dialog 생성
    if (cldColor.ShowDialog() == DialogResult.OK)
        tsbtnColor.BackColor = cldColor.Color;  // 선택한 색으로 배경 변경
}
```

<부모 Form 코드>

ToolStrip 의 맨 위쪽 버튼을 클릭하면 ColorDialog 를 생성하고 띄운 뒤, 선택한 색으로 버튼의 배경색으로 바꾼다.

3.2. 연필 (0.5 점)




```
/// <summary> 연필/브러쉬 (곡선) 표현 클래스
class MyPen
{
    // 변수 선언부
    private List<MyLines> lines; // 직선 집합

    // 기본 생성자
    public MyPen()
    {
        lines = new List<MyLines>();
    }

    // 직선을 컬렉션에 추가하는 함수
    public void setLine(Point start, Point finish, int thick, Color color)
    {
        MyLines line = new MyLines(); // 직선 생성
        line.setPoint(start, finish, thick, color); // 직선 설정
        lines.Add(line); // 컬렉션에 추가
    }

    // 멤버 변수 getter 함수
    public List<MyLines> getLine() => lines;
}
```

곡선을 표현하는 클래스 MyPen 을 정의했다. MyPen 은 직선을 표현하는 클래스 MyLines 의 컬렉션을 담는다. setLine 함수는 인자로 받은 두 점을 잇는 직선을 만들어서 컬렉션에 추가하는 함수이다.

```
// 마우스 누를 때 눌린 위치 저장
private void pnlBoard_MouseDown(object sender, MouseEventArgs e)
{
    p.X = e.X;
    p.Y = e.Y;
    isClicked = true;
}
```

```
// 마우스 눌린 상태로 이동 시 추가된 도형 설정
private void pnlBoard_MouseMove(object sender, MouseEventArgs e)
{
    if (!isClicked) return;
    if (e.Button == MouseButtons.Left)
    {
        switch (parent.mode)
        {
            // 연필 모드
            case DrawMode.Pencil:
                myPen[myPen.Count - 1].setLine(p, e.Location, 1, parent.color);
                // 좌표 갱신
                p.X = e.X;
                p.Y = e.Y;
                break;
```

```
// 마우스 놓으면 그려질 새 도형 추가
private void pnlBoard_MouseUp(object sender, MouseEventArgs e)
{
    isClicked = false;
    if (e.Button == MouseButtons.Left)
    {
        switch (parent.mode)
        {
            case (DrawMode.Pencil | DrawMode.Brush):
                myPen.Add(new MyPen());
                break;
```

<자식 Form의 Mouse 이벤트 코드>

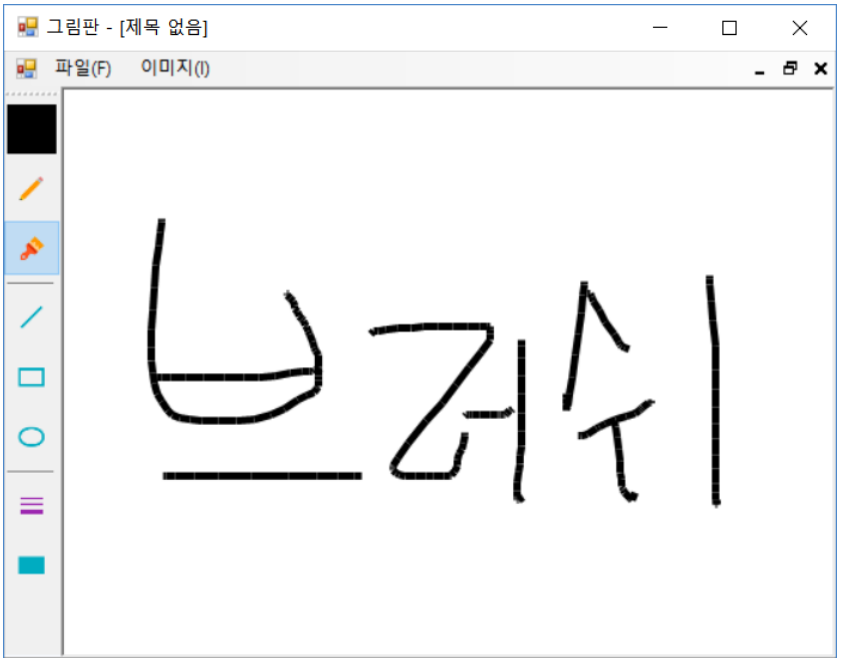
자식 Form 의 Panel 위에서 마우스 클릭을 했을 때 수행되는 코드이다. 마우스 단추를 누르면 마우스 포인터 위치를 저장한다. 마우스 이동 시 이동한 위치와 저장된 위치의 좌표로 두께 1 인 직선을 만들어 추가하고, 이동한 곳의 위치를 새로 저장한다. 마우스 단추를 놓으면 그려질 곡선을 담는 컬렉션에 새 곡선을 추가한다.

```
// 저장된 곡선 그리기
foreach(MyPen pencil in myPen)
{
    foreach (MyLines line in pencil.getLine())
    {
        pen = new Pen(line.getColor(), line.getThick());
        g.DrawLine(pen, line.getPoint1(), line.getPoint2());
    }
}
```

<자식 Form의 Paint 이벤트 코드>

자식 Form 의 Paint 이벤트에서 저장된 곡선을 그린다. 곡선들이 저장된 컬렉션 myPen 의 항목을 foreach 문으로 탐색한다. 곡선은 직선의 집합이므로 각 항목 안의 직선 집합 컬렉션을 다시 foreach 문으로 탐색하여 직선들을 그린다.

3.3. 브러쉬 (0.5 점)

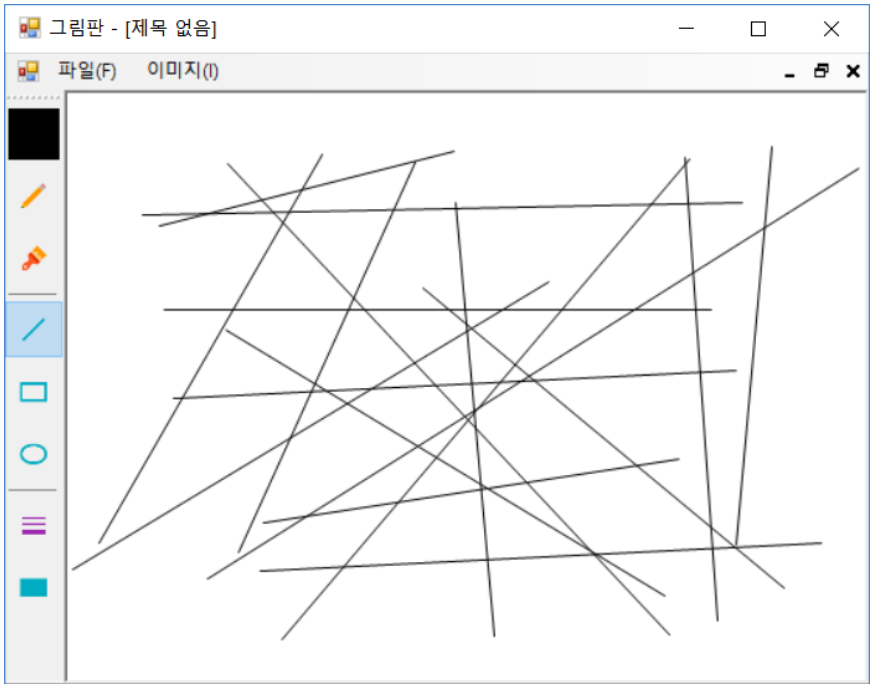



```
// 브러쉬 모드
case DrawMode.Brush:
    myPen[myPen.Count - 1].setLine(p, e.Location, 5, parent.color);
    // 좌표 갱신
    p.X = e.X;
    p.Y = e.Y;
    break;
```

<자식 Form의 Mouse 이벤트 코드>

MouseMove 이벤트에서 두께가 5 인 직선으로 myPen 컬렉션에 추가하는 것 외엔 연필 구현 방법과 동일하다.

4.1. 직선 (0.5 점)




```
/// <summary> 직선 표현 클래스
class MyLines
{
    // 변수 선언부
    private Point[] point = new Point[2];   // 직선 시작점과 끝점
    private int thick;  // 직선 두께
    private Color color;    // 직선 색상

    // 기본 생성자
    public MyLines()...

    // 직선 설정 함수
    public void setPoint(Point start, Point finish, int thick, Color color)
    {
        point[0] = start;
        point[1] = finish;
        this.thick = thick;
        this.color = color;
    }

    // 멤버 변수 getter 함수
    public Point getPoint1() => point[0];
    public Point getPoint2() => point[1];
    public int getThick() =>  thick;
    public Color getColor() => color;
}
```

직선을 표현하는 클래스 MyLines 를 정의했다. MyLines 는 직선의 요소인 시작점과 끝점, 선의 두께, 색상을 담는다. setPoint 함수로 직선의 멤버 변수를 설정한다.

```
// 직선 모드
case DrawMode.Line:
    myLines[myLines.Count - 1].setPoint(p, e.Location, parent.thick, parent.color);
    break;
```

```
case DrawMode.Line:
    myLines.Add(new MyLines());
    break;
```

<자식 Form의 Mouse 이벤트 코드>

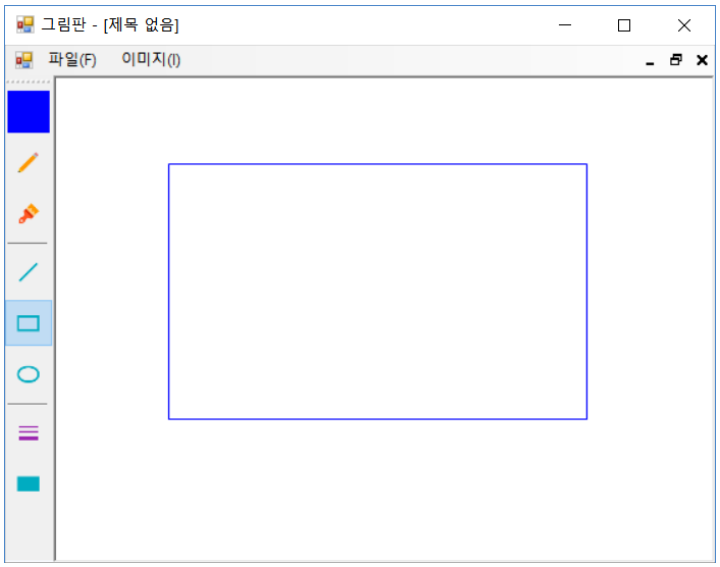
마우스 단추를 누르면 해당 위치를 저장하고, 마우스를 이동하면 저장된 위치와 현재 위치를 잇는 직선을 만든다. parent.thick 은 부모 Form 에서 선택된 선의 두께 값이고, parent.color 는 부모 Form 의 ToolStripButton 중 색 버튼의 배경색을 반환한다. 마우스 단추를 놓으면 그려질 새 직선을 컬렉션에 추가한다.

```
// 저장된 직선 그리기
foreach(MyLines line in myLines)
{
    pen = new Pen(line.getColor(), line.getThick());
    g.DrawLine(pen, line.getPoint1(), line.getPoint2());
}
```

<자식 Form의 Paint 이벤트 코드>

자식 Form 의 Paint 이벤트에서 컬렉션에 저장된 직선을 그린다.

4.2. 사각형 (0.5 점)



```
/// <summary> 사각형 표현 클래스
class MyRect
{
    // 변수 선언부
    private Rectangle rect; // 사각형
    private int thick; // 사각형 선의 두께
    private Color color; // 사각형 색상
    private bool isFilled; // 사각형 채우기 여부

    // 기본 생성자
    public MyRect()...

    // 사각형 설정 함수
    public void setRect(Point start, Point finish, int thick, Color color, bool fill)
    {
        // 사각형 좌표 설정
        rect.X = Math.Min(start.X, finish.X);
        rect.Y = Math.Min(start.Y, finish.Y);
        rect.Width = Math.Abs(finish.X - start.X);
        rect.Height = Math.Abs(finish.Y - start.Y);

        this.thick = thick;
        this.color = color;
        this.isFilled = fill;
    }

    // 멤버 변수 getter 함수
    public Rectangle getRect() => rect;
    public int getThick() => thick;
    public Color getColor() => color;
    public bool getFill() => isFilled;
}
```

사각형을 표현하는 클래스 MyRect 를 정의했다. MyRect 는 사각형 객체와 테두리 선의 두께, 색상, 단색 채우기 여부를 담는다. setRect 함수 인자로 받은 두 점으로 사각형의 좌표를 설정한다.

```
// 사각형 모드
case DrawMode.Rectangle:
    myRect[myRect.Count - 1].setRect(p, e.Location, parent.thick, parent.color, parent.fillMode);
    break;
```

```
case DrawMode.Rectangle:
    myRect.Add(new MyRect());
    break;
```

<자식 Form의 Mouse 이벤트 코드>

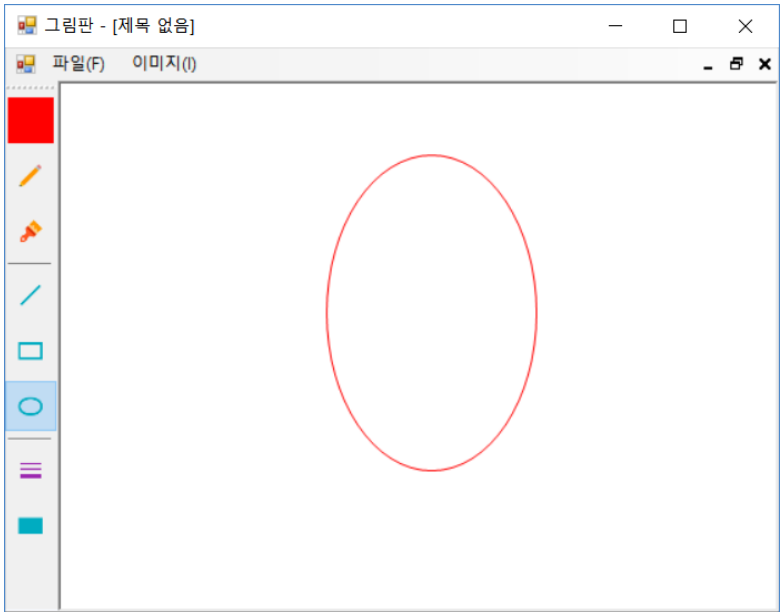
마우스 단추를 누르면 해당 위치를 저장하고, 마우스를 이동하면 저장된 위치와 현재 위치로 사각형을 만든다. parent.fillMode 는 부모 Form 의 단색 채우기 ToolStripMenuItem 체크 여부를 반환한다. 마우스 단추를 놓으면 그려질 새 사각형을 컬렉션에 추가한다.

```
// 저장된 사각형 그리기
foreach(MyRect rect in myRect)
{
    if (rect.getFill())
    {
        brush = new SolidBrush(rect.getColor());
        g.FillRectangle(brush, rect.getRect());
    }
    else
    {
        pen = new Pen(rect.getColor(), rect.getThick());
        g.DrawRectangle(pen, rect.getRect());
    }
}
```

<자식 Form의 Paint 이벤트 코드>

자식 Form 의 Paint 이벤트에서 단색 채우기 여부를 확인하여 채워져 있으면 SolidBrush 로, 채워져 있지 않으면 Pen 으로 컬렉션에 저장된 사각형을 그린다.

4.3. 타원 (0.5 점)



```
/// <summary> 타원 표현 클래스
class MyOval
{
    // 변수 선언부
    private Rectangle rect0; // 타원에 외접하는 사각형
    private int thick;  // 타원 선의 두께
    private Color color;    // 타원 색상
    private bool isFilled;  // 타원 채우기 여부

    // 기본 생성자
    public MyOval()...

    // 타원 설정 함수
    public void setRect0(Point start, Point finish, int thick, Color color, bool fill)
    {
        rect0.X = Math.Min(start.X, finish.X);
        rect0.Y = Math.Min(start.Y, finish.Y);
        rect0.Width = Math.Abs(start.X - finish.X);
        rect0.Height = Math.Abs(start.Y - finish.Y);
        this.thick = thick;
        this.color = color;
        this.isFilled = fill;
    }

    // 멤버 변수 getter 함수
    public Rectangle getRect0() => rect0;
    public int getThick() => thick;
    public Color getColor() => color;
    public bool getFill() => isFilled;
}
```

타원을 표현하는 클래스 MyOval 를 정의했다. MyOval 은 타원에 외접하는 사각형 객체와 테두리 선의 두께, 색상, 단색 채우기 여부를 담는다. setRect0 함수 인자로 받은 두 점으로 외접 사각형의 좌표를 설정한다.

```
        // 타원 모드
        case DrawMode.Oval:
            myOval[myOval.Count - 1].setRect0(p, e.Location, parent.thick, parent.color, parent.fillMode);
            break;
}


// 패널 다시 그리기
pnlBoard.Invalidate(true);
pnlBoard.Update();
```

```
case DrawMode.Oval:
    myOval.Add(new MyOval());
    break;
```

<자식 Form의 Mouse 이벤트 코드>

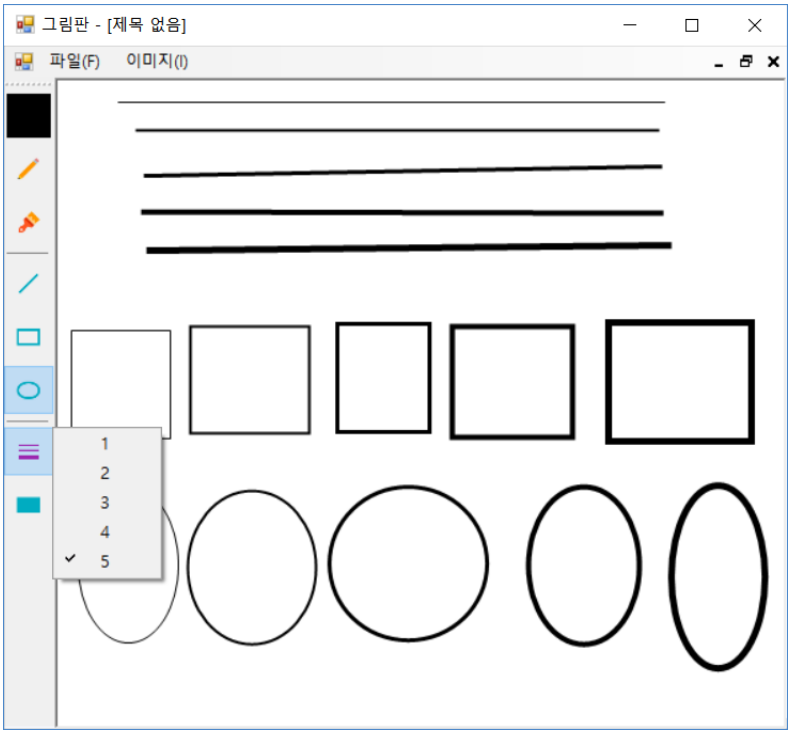
마우스 단추를 누르면 해당 위치를 저장하고, 마우스를 이동하면 저장된 위치와 현재 위치로 타원을 만든다. 자식 Form 은 Panel 에 마우스 왼쪽 단추를 누르고 이동할 때 마다 Panel 을 새로 그린다. 마우스 단추를 놓으면 그려질 새 타원을 컬렉션에 추가한다.

```
// 저장된 타원 그리기
foreach(MyOval oval in myOval)
{
    if (oval.getFill())
    {
        brush = new SolidBrush(oval.getColor());
        g.FillEllipse(brush, oval.getRect0());
    }
    else
    {
        pen = new Pen(oval.getColor(), oval.getThick());
        g.DrawEllipse(pen, oval.getRect0());
    }
}
```

<자식 Form의 Paint 이벤트 코드>

자식 Form 의 Paint 이벤트에서 단색 채우기 여부를 확인하여 채워져 있으면 SolidBrush 로, 채워져 있지 않으면 Pen 으로 컬렉션에 저장된 타원을 그린다.

5.1. 선 두께 (1.5 점)

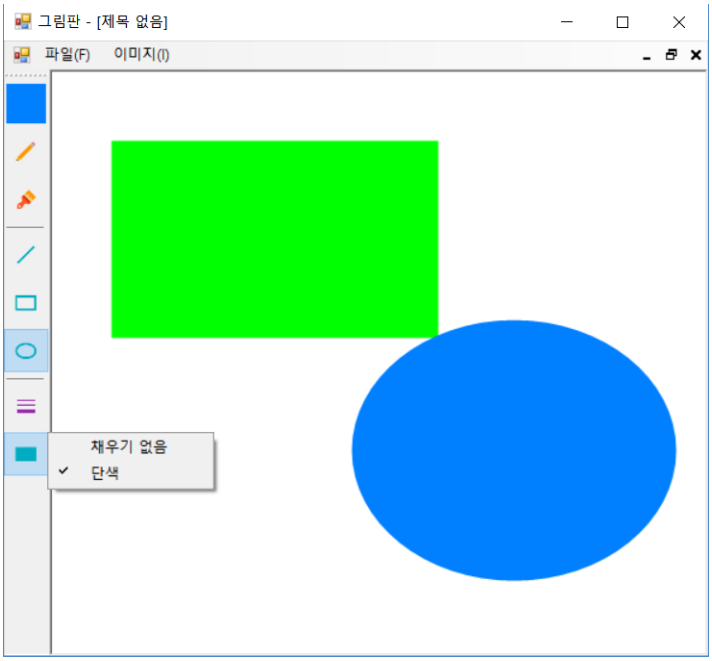


```
// 선 두께 지정 프로퍼티
public int thick
{
    get
    {
        if (tsmi1.Checked)
            return 1;
        else if (tsmi2.Checked)
            return 2;
        else if (tsmi3.Checked)
            return 3;
        else if (tsmi4.Checked)
            return 4;
        else if (tsmi5.Checked)
            return 5;
        else
            return 0;
    }
}
```

<부모 Form 코드>

선 두께 버튼을 ToolStripDropDownButton 으로 만들어 누르면 두께를 선택할 수 있는 ToolStrip 이 뜬다. 부모 Form 에 ToolStripMenuItem 의 선택 여부에 따라 두께 값을 반환하는 thick 프로퍼티를 만들었다. 자식 Form 에서 이 프로퍼티를 참조하여 도형을 그린다.

5.2. 채우기 (1.5 점)

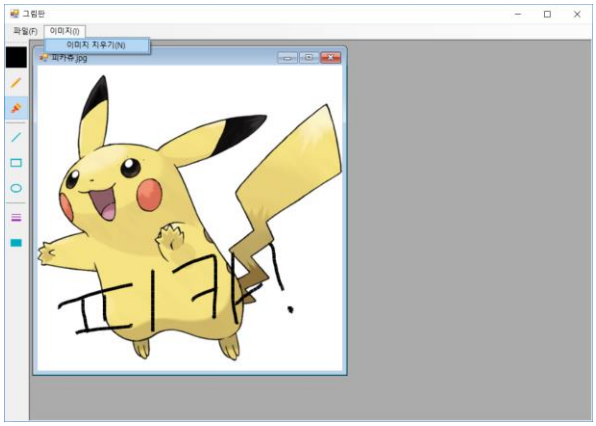


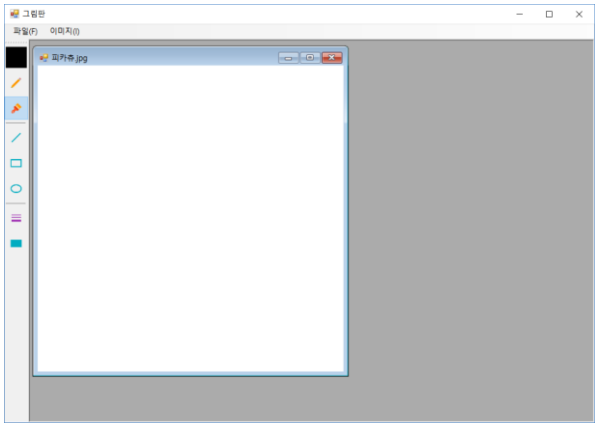

```
// 채우기 모드 반환 프로퍼티
public bool fillMode => tsmiFill.Checked;
```

<부모 Form 코드>

채우기 버튼을 ToolStripDropDownButton 으로 만들어 누르면 채우기 없음/단색을 선택할 수 있는 ToolStrip 이 뜬다. 부모 Form 에 단색 ToolStripMenuItem 의 선택 여부를 반환하는 fillMode 프로퍼티를 만들었다. 자식 Form 에서 이 프로퍼티를 참조하여 도형을 그린다.

6.1. 이미지 지우기 (1 점)





```
// 이미지 지우기 함수
public void Clear()
{
    myPen.Clear();
    myLines.Clear();
    myRect.Clear();
    myOval.Clear();
    myPen.Add(new MyPen());
    myLines.Add(new MyLines());
    myRect.Add(new MyRect());
    myOval.Add(new MyOval());
    img = new Bitmap(pnlBoard.Width, pnlBoard.Height);
    pnlBoard.Refresh();
}
```

<자식 Form 코드>

```
// 이미지 지우기
private void tsmiClear_Click(object sender, EventArgs e)
{
    frmChild child = ActiveMdiChild as frmChild;    // 활성화된 자식 Form
    child.Clear();
}
```

<부모 Form 코드>

Clear 함수는 자식 Form 에 선언된 모든 도형 컬렉션을 비우고 새로 초기화 해준 다음, Panel 크기만큼의 빈 이미지를 img 변수에 할당해서 Panel 을 다시 그리는 함수이다. 부모 Form 에서 이미지 지우기 ToolStripMenuItem 을 클릭하면 활성화된 자식 Form 을 가져와서 Clear 함수를 호출한다.

추가 1. 더블 버퍼링 (1 점)

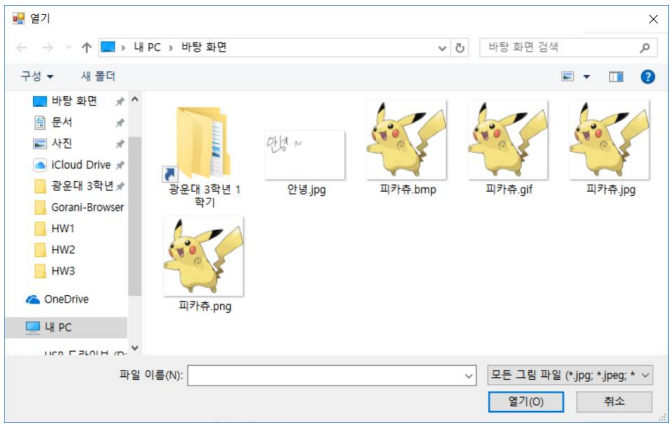
```
// 더블 버퍼링 패널
public class DoubleBufferPanel : System.Windows.Forms.Panel
{
    public DoubleBufferPanel()
    {
        this.SetStyle(System.Windows.Forms.ControlStyles.DoubleBuffer |
                      System.Windows.Forms.ControlStyles.UserPaint |
                      System.Windows.Forms.ControlStyles.AllPaintingInWmPaint, true);
        this.UpdateStyles();
    }
}
```

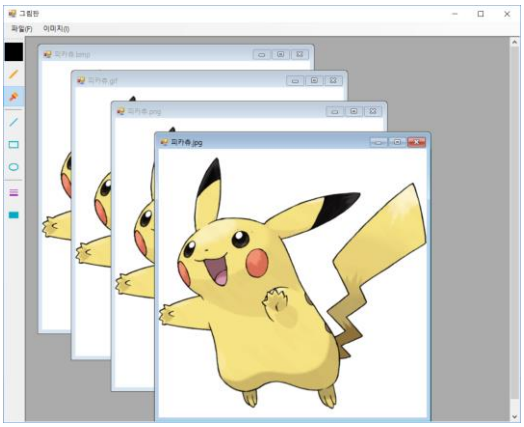
```
private void InitializeComponent()
{
    this.pnlBoard = new DoubleBufferPanel();
```

<자식 Form Designer 코드>

Panel 을 상속받은 DoubleBufferPanel 클래스를 만들어 생성자에 더블 버퍼링 설정 코드를 추가하고 기존 자식 Form Panel 을 DoubleBufferPanel 로 대체했다.

추가 2. 다양한 형식의 이미지 열기 (1 점)



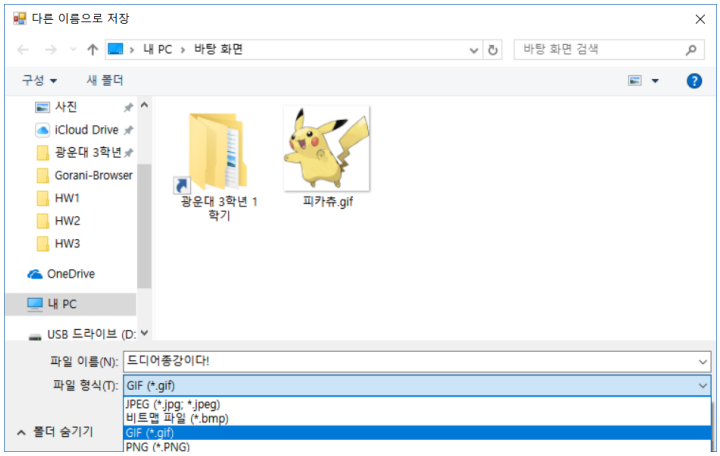


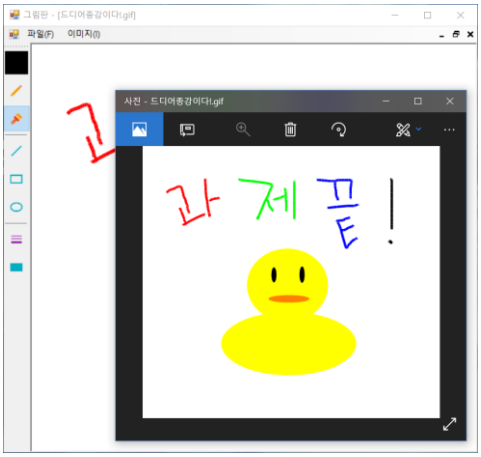
```
// 확장자 필터 설정
ofdOpen.Filter = "JPEG (*.jpg; *.jpeg)|*.jpg; *jpeg|"
    + "비트맵 파일 (*.bmp)|*.bmp |"
    + "GIF (*.gif)|*.gif|"
    + "PNG (*.PNG)|*.png|"
    + "모든 그림 파일 (*.jpg; *.jpeg; *.bmp; *.gif; *.png)|*.jpg; *.jpeg; *.bmp; *.gif; *.png";
```

<부모 Form 열기 이벤트 코드>

OpenFileDialog 의 확장자 필터에 JPG 뿐만 아니라 BMP, GIF, PNG 등을 추가해 다양한 형식의 이미지를 열 수 있게 하였다.

추가 3. 다양한 형식의 이미지 저장 (1 점)






```
// 확장자 필터 설정
sfdSave.Filter = "JPEG (*.jpg; *.jpeg)|*.jpg; *jpeg|"
    + "비트맵 파일 (*.bmp)|*.bmp |"
    + "GIF (*.gif)|*.gif|"
    + "PNG (*.PNG)|*.png";
```

```
switch (sfdSave.FilterIndex)
{
    case 1:
        bmp.Save(sfdSave.FileName, ImageFormat.Jpeg);   // JPG로 저장
        break;

    case 2:
        bmp.Save(sfdSave.FileName, ImageFormat.Bmp);    // BMP로 저장
        break;

    case 3:
        bmp.Save(sfdSave.FileName, ImageFormat.Gif);    // GIF로 저장
        break;
    case 4:
        bmp.Save(sfdSave.FileName, ImageFormat.Png);    // PNG로 저장
        break;
    default:
        break;
}
```

<자식 Form 저장 이벤트>

SaveFileDialog 의 확장자 필터에 JPG 뿐만 아니라 BMP, GIF, PNG 등을 추가해 다양한 형식의 이미지를 저장할 수 있게 하였다. 확장자 필터 인덱스 순서에 따라 Bitmap 객체에 담긴 이미지를 지정한 확장자로 저장한다.

고찰

팀프로젝트 당시, 이미 캡처한 이미지 위에 그림 메모를 추가하는 기능을 구현한 적이 있어 전체적인 구현에 큰 어려움은 없었다. 다만, 그 당시에는 이미지 위에 투명한 PictureBox 을 덮어 그 PictureBox 에 도형이 그려지도록 하였었지만, 이번 과제에서는 다른 방법으로 구현하였다. 주어진 Form 구성 요소 외에 다른 Control 을 사용하지 않고, 곡선을 여러 직선을 저장할 수 있는 컬렉션으로 표현하기 위해서이다. 이를 위해 모든 도형을 클래스로 만들어 컬렉션에 저장하고, MouseMove 이벤트 때 설정한 도형을 Paint 이벤트 때 그리는 방식을 이용하였다. 해당 방법은 수업 때 실습해 본 방법으로 강의 자료를 참고하여 구현할 수 있었다. 그러나 이 방법에 몇 가지 문제점이 있었다. 먼저, 도형을 그릴 때 화면이 깜박거리거나 찢겨져서 보이는 듯한 현상이 발생하였다. 이 문제는 과제 설명에 나온 대로 더블 버퍼링을 사용하여 해결할 수 있었다. 또 다른 문제는 Paint 이벤트에서 각 도형을 종류 별로 차례대로 그리기 때문에 먼저 그리는 종류의 도형은 나중에 그리는 종류의 도형 위에 겹쳐져 그려지지 않는 것이다. 이 문제를 해결하기 위해선 도형을 컬렉션에 저장하여 그리지 않고 바로 그리는 방법을 사용하거나, 각 도형 객체에 순서를 매겨 순서대로 그려지도록 하는 방법을 사용해야 할 것 같다. 추가로, 해당 방법은 그린 도형이 많은 경우, 메모리 공간 사용의 문제와 속도 저하 문제가 발생할 것으로 예상된다.

선 두께와 채우기 버튼을 눌렀을 때 뜨는 메뉴는 과제 설명에 나온 형태대로 구현하려 했으나, 동적으로 생성되는 패널을 만들어 그 위에 버튼을 올리는 방법 외에는 적당한 구현 방법이 떠오르지 않았다. 이 방법은 구현 방법이 번거로운 것 같아 설명에 나온 형태와 조금 다르지만 DropDownButton 을 이용하여 구현을 하였다. DropDownButton 은 쉽게 메뉴를 추가하거나 뺄 수 있고 메뉴 항목의 체크도 가능하며, 메뉴를 선택하면 자동으로 메뉴 창이 사라진다.

본인은 과제에서 요구하는 기능을 전부 구현하였고, 테스트 결과 모든 기능들은 문제없이 작동하는 것으로 보인다.