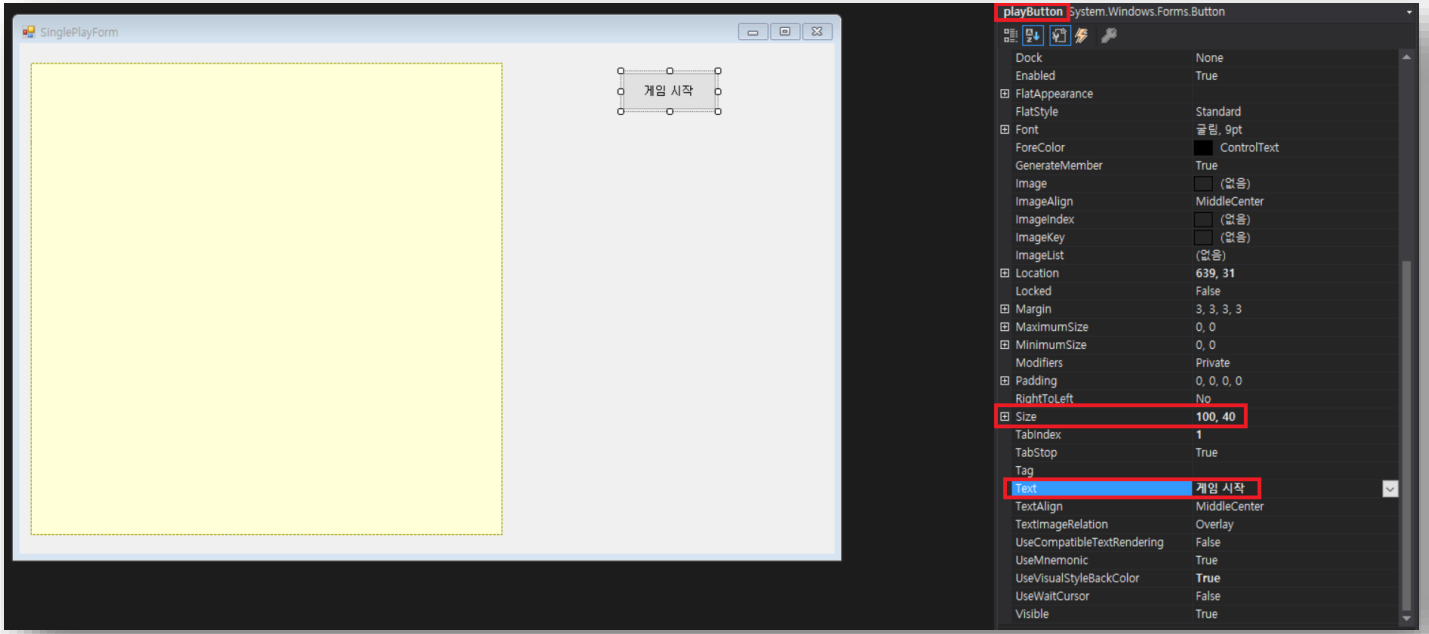
컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 67강 - 오목 혼자하기 판정 기능 구현하기

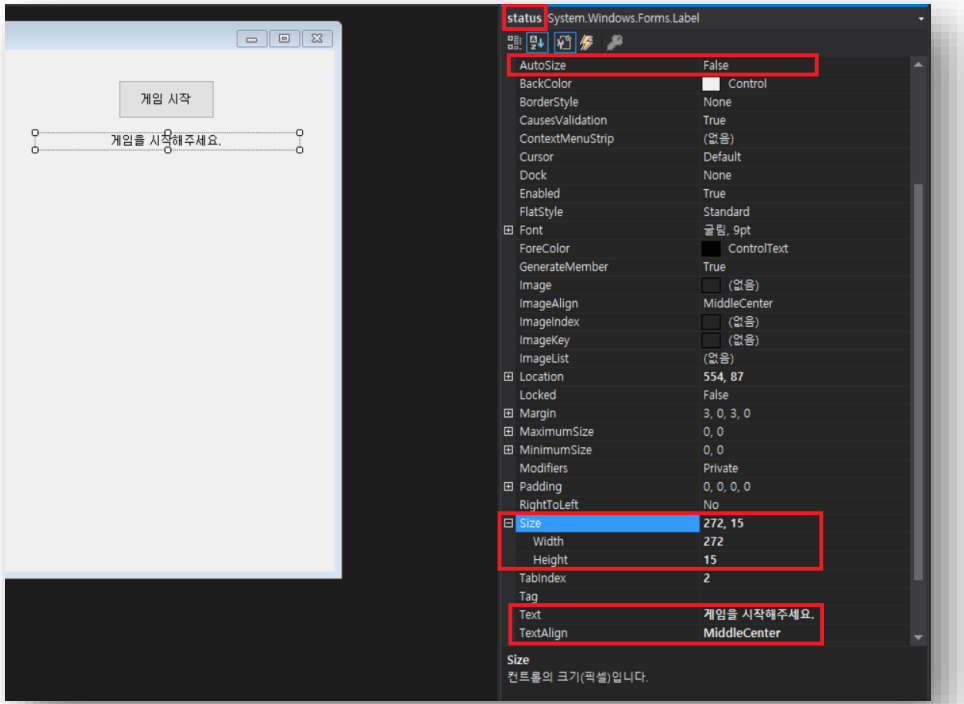
오목 혼자하기 판정 기능 구현하기

게임 시작 버튼 디자인하기



오목 혼자하기 판정 기능 구현하기

상태 라벨 디자인하기



오목 혼자하기 판정 기능 구현하기

싱글 플레이를 위한 변수 선언

```
private const int rectSize = 33; // 오목판의 셀 크기
private const int edgeCount = 15; // 오목판의 선 개수

private enum Horse { none = 0, BLACK, WHITE};
private Horse[,] board = new Horse[edgeCount, edgeCount];
private Horse nowPlayer = Horse.BLACK;

private bool playing = false;
```

오목 혼자하기 판정 기능 구현하기

승리 판정 함수

```
private bool judge() // 승리 판정 함수
{
  for (int i = 0; i < edgeCount - 4; i++) // 가로
    for (int j = 0; j < edgeCount; j++)
      if (board[i, j] == nowPlayer && board[i + 1, j] == nowPlayer && board[i + 2, j] == nowPlayer &&
          board[i + 3, j] == nowPlayer && board[i + 4, j] == nowPlayer)
            return true;
  for (int i = 0; i < edgeCount; i++) // 세로
    for (int j = 4; j < edgeCount; j++)
      if (board[i, j] == nowPlayer && board[i, j - 1] == nowPlayer && board[i, j - 2] == nowPlayer &&
          board[i, j - 3] == nowPlayer && board[i, j - 4] == nowPlayer)
            return true;
  for (int i = 0; i < edgeCount - 4; i++) // Y = X 직선
    for (int j = 0; j < edgeCount - 4; j++)
      if (board[i, j] == nowPlayer && board[i + 1, j + 1] == nowPlayer && board[i + 2, j + 2] == nowPlayer &&
          board[i + 3, j + 3] == nowPlayer && board[i + 4, j + 4] == nowPlayer)
            return true;
  for (int i = 4; i < edgeCount; i++) // Y = -X 직선
    for (int j = 0; j < edgeCount - 4; j++)
      if (board[i, j] == nowPlayer && board[i - 1, j + 1] == nowPlayer && board[i - 2, j + 2] == nowPlayer &&
          board[i - 3, j + 3] == nowPlayer && board[i - 4, j + 4] == nowPlayer)
            return true;
  return false;
}
```

오목 혼자하기 판정 기능 구현하기

새로고침 함수

```
private void refresh()
{
  this.boardPicture.Refresh();
  for (int i = 0; i < edgeCount; i++)
    for (int j = 0; j < edgeCount; j++)
      board[i, j] = Horse.none;
}
```

오목 혼자하기 판정 기능 구현하기

게임시작 버튼 이벤트 함수

```
private void playButton_Click(object sender, EventArgs e)
{
  if(!playing)
  {
    refresh();
    playing = true;
    playButton.Text = "재시작";
    status.Text = nowPlayer.ToString() + " 플레이어의 차례입니다.";
  }
  else
  {
    refresh();
    status.Text = "게임이 재시작되었습니다.";
  }
}
```

오목 혼자하기 판정 기능 구현하기

오목판 마우스 클릭 이벤트 함수 ①

```
private void boardPicture_MouseDown(object sender, MouseEventArgs e)
{
  if(!playing)
  {
    MessageBox.Show("게임을 실행해주세요.");
    return;
  }
  Graphics g = this.boardPicture.CreateGraphics();
  int x = e.X / rectSize;
  int y = e.Y / rectSize;
  if (x < 0 || y < 0 || x >= edgeCount || y >= edgeCount)
  {
    MessageBox.Show("테두리를 벗어날 수 없습니다.");
    return;
  }
  if(board[x, y] != Horse.none) return;
  board[x, y] = nowPlayer;
```

오목 혼자하기 판정 기능 구현하기

오목판 마우스 클릭 이벤트 함수 ②

```
  if(nowPlayer == Horse.BLACK)
  {
    SolidBrush brush = new SolidBrush(Color.Black);
    g.FillEllipse(brush, x * rectSize, y * rectSize, rectSize, rectSize);
  }
  else
  {
    SolidBrush brush = new SolidBrush(Color.White);
    g.FillEllipse(brush, x * rectSize, y * rectSize, rectSize, rectSize);
  }
  if(judge())
  {
    status.Text = nowPlayer.ToString() + "플레이어가 승리했습니다.";
    playing = false;
    playButton.Text = "게임시작";
  }
  else
  {
    nowPlayer = ((nowPlayer == Horse.BLACK) ? Horse.WHITE : Horse.BLACK);
    status.Text = nowPlayer.ToString() + " 플레이어의 차례입니다.";
  }
}
```

오목 혼자하기 판정 기능 구현하기

솔루션 빌드 및 실행