Dokumentacja

Projekt gry „Minesweeper”

Kolan Piotr

1. Działanie programu

1.1. Deklaracja zmiennych

```
bool IsVisable = false;

static uint numberOfBombs = 10;
static uint InGameNumberOfBombs = numberOfBombs;
static uint TooMuchBombs = 0;

Random rnd = new Random();

DispatcherTimer timer = new DispatcherTimer();
DateTime startOfGame = DateTime.MinValue;

//to będzie można zmienić poprzez nowe okienko
int dimensionX = 10;
int dimensionY = 10;

//domyślenie ustawienia planszy są 10 na 10
short[,] Positions = new short[10, 10];
Button[,] Buttons = new Button[10, 10];
List<Button> checkedFields = new List<Button>();
```

IsVisable – stosowana do wyświetlania i usuwania planszy z ekranu

numberOfBombs – liczba w grze

InGameNumberOfBombs – liczba bomb wyświetlana podczas trwania gry

TooMuchBombs – parametr sprawdzający czy nie zostało „oflagowane” pole bez bomby, używane do sprawdzenia warunków zwycięstwa

Rnd – zmienna do losowania liczb, użyta przy generowaniu bomb

Timer, startOfGame – zmienne wykorzystywane do licznika czasu gry.

DimensionX/Y – Wymiary planszy do gry

Positions[,] – Tablica wartości pól

Buttons[,] – Tablica pól

checkedFields – sprawdzone pole, zmienna wykorzystywana do sprawdzenia warunków zwycięstwa

1.2. Losowanie bomb

```
private void GenerateBombs() {
    for(int x = 0; x < dimensionX; x++) {
        for(int y = 0; y < dimensionY; y++) {
            Positions[x, y] = 0;
        }
    }
    for(int bombs = 0; bombs < numberOfBombs; bombs++) {
        int x = rnd.Next(0, dimensionX);
        int y = rnd.Next(0, dimensionY);
        if(Positions[x, y] == 0) {
            Positions[x, y] = 10;
        }
        else { bombs--; }
    }
}
```

Funkcja zeruje wartości pól gdyby już wcześniej została rozpoczęta gra, i losuje nowe pola bez powtórzeń.

1.3. Przydzielanie wartości pozostałym polom

```
private void GeneratePositionValue() {
    byte bombCounter;
    for(int x = 0;x < dimensionX;x++) {
        for(int y = 0;y < dimensionY;y++) {
            //jeśli pole posiada bombe to skip żeby nie zmienic jej wartości
            if(Positions[x, y] == 10) { continue; }
            bombCounter = 0;

            for(int currentPositionX = -1;currentPositionX < 2;currentPositionX++) {
                int checkX = x + currentPositionX;

                for(int currentPositionY = -1; currentPositionY < 2;currentPositionY++) {
                    int checkY = y + currentPositionY;

                    // jeśli będzie mniejsze od zera lub wieksze od wymiarów to wychodze poza właściwe wymiary planszy, zmienna nie będzie zliczać samej siebie pole [0,0]
                    if(checkX < 0 || checkY < 0 || checkX >= dimensionX || checkY >= dimensionY) { continue; }
                    if(checkX == x && checkY == y) { continue; }
                    if(Positions[checkX, checkY] == 10) { bombCounter++; }
                }
            }
            // jeśli nie ma bomb w koło to pole przyjmuje wartość pustego pola
            if(bombCounter == 0) { Positions[x, y] = 11; }
            else { Positions[x, y] = bombCounter; }
        }
    }
}
```

Funkcja służy do przypisywania wartości pozostałym polom.

1.4. Tworzenie przycisków

```
private void GenerateButtons() {
    for(int x = 0; x < dimensionX; x++) {
        for(int y = 0; y < dimensionY; y++) {
            Button button = new Button();
            button.Name = $"Button_{x}_{y}";
            button.Width = (FieldCanvas.Width / dimensionX);
            button.Height = (FieldCanvas.Height / dimensionY);
            button.Click += Reveal_Click;
            button.MouseRightButtonDown += Flagged_Click;
            button.Tag = $"{x},{y}";
            //button.Content = $"{Positions[x, y]}"; // pole do testów
            button.Background = Brushes.Red;
            button.BorderBrush = Brushes.Red;
            Buttons[x, y] = button;
        }
    }
}
```

Funkcja odpowiednie właściwości przypisuje właściwości przyciskom.

1.5. Tworzenie planszy

```
private void AddButtons() {
    if(!IsVisable) {
        for(int x = 0; x < dimensionX; x++) {
            for(int y = 0; y < dimensionY; y++) {
                Canvas.SetLeft(Buttons[x, y], x * Buttons[x, y].Width);
                Canvas.SetTop(Buttons[x, y], y * Buttons[x, y].Height);
                FieldCanvas.Children.Add(Buttons[x, y]);
            }
        }
        IsVisable = true;
    }
}
```

```
<Canvas Width="540" Height="500" x:Name="FieldCanvas" Margin="20,50,20,20">
</Canvas>
```

Plansza jest tworzona poprzez przypisanie przycisków do Canvas wewnątrz pliku .Xaml. Lokalizacja przycisku jest wyliczana po wymiarach przycisków.

1.6. Ściąganie planszy z okna gry

```
private void RemoveButtons() {
    if(IsVisable) {
        for(int x = 0; x < dimensionX; x++) {
            for(int y = 0; y < dimensionY; y++) {
                FieldCanvas.Children.Remove(Buttons[x, y]);
            }
        }
        IsVisable = false;
    }
}
```

Usuwa wcześniej przypisane przyciski z canvy

1.7. Funkcja obliczająca czas gry

```
private void Timer_Tick(object sender, EventArgs e) {
    TimeSpan dod = DateTime.Now - startOfGame;
    if(dod > TimeSpan.Zero) {
        Timer.Content = dod.ToString(@"mm\:ss");
    }
}
```

Służy do ustawiania stanu timera.

1.8. Warunki końca gry

```
private void EndGameEventAppear(bool casecode) {
    if(casecode == false) {
        timer.Stop();
        for(int x = 0; x < dimensionX; x++) {
            for(int y = 0; y < dimensionY; y++) {
                Buttons[x, y].Content = Positions[x, y].ToString();
                Buttons[x, y].IsEnabled = false;
            }
        }
        MessageBox.Show("przegrałeś!");
    }
    else {
        timer.Stop();
        for(int x = 0; x < dimensionX; x++) {
            for(int y = 0; y < dimensionY; y++) {
                Buttons[x, y].IsEnabled = false;
            }
        }
        MessageBox.Show("wygrałeś!");
    }
}
```

Funkcja przechowuje warunki zwycięstwa gry i zakańcza ją gdy te się spełnią

1.9. Funkcje do odsłaniania pól

```
8 references
private void Reveal_Click(object sender, RoutedEventArgs e) {
    Button button = (Button) sender;
    int x = Convert.ToInt32(button.Tag.ToString().Split(',').GetValue(0));
    int y = Convert.ToInt32(button.Tag.ToString().Split(',').GetValue(1));
    //jeśli bomba
    if(Positions[x, y] == 10) {
        button.Background = Brushes.DarkRed;
        EndGameEventAppear(false);
    }
    //jeśli pole puste
    else if(Positions[x, y] == 11 && button.Background != Brushes.Orange) {
        button.Background = Brushes.Green;
        button.BorderBrush = button.Background;
        button.Click -= Reveal_Click;
        button.MouseRightButtonDown -= Flagged_Click;
        checkedFields.Add(button);
        ReavealAdjacentEmptyFields(button);
        if(checkedFields.Count == (dimensionX * dimensionY) - InGameNumberOfBombs) {
            EndGameEventAppear(true);
        }
    }
    //jeśli pole z liczbą
    else {
        button.Background = Brushes.Green;
        button.BorderBrush = button.Background;
        button.Content = Positions[x, y];
        button.Click -= Reveal_Click;
        button.Click += Number_Click;
        button.MouseRightButtonDown -= Flagged_Click;
        checkedFields.Add(button);
        if(checkedFields.Count == (dimensionX * dimensionY) - InGameNumberOfBombs) {
            EndGameEventAppear(true);
        }
    }
}
```

```
private void ReavealAdjacentEmptyFields(Button button) {
    int x = Convert.ToInt32(button.Tag.ToString().Split(',').GetValue(0));
    int y = Convert.ToInt32(button.Tag.ToString().Split(',').GetValue(1));

    for(int currentPositionX = -1; currentPositionX < 2; currentPositionX++) {
        int checkX = x + currentPositionX;

        for(int currentPositionY = -1; currentPositionY < 2; currentPositionY++) {
            int checkY = y + currentPositionY;

            // jeśli będzie mniejsze od zera lub wieksze od wymiarów to wychodze poza właściwe wymiary planszy, zmienna nie będzie zliczać samej sieb
            if(checkX < 0 || checkY < 0 || checkX >= dimensionX || checkY >= dimensionY) { continue; }
            if(checkX == x && checkY == y) { continue; }

            //wczytuje przycisk z macierzy
            Button adjButton = Buttons[checkX,checkY];

            int xAdj = Convert.ToInt32(adjButton.Tag.ToString().Split(',').GetValue(0));
            int yAdj = Convert.ToInt32(adjButton.Tag.ToString().Split(',').GetValue(1));

            //sprawdzam wczytany przycisk
            if(Positions[xAdj,yAdj] == 11 && adjButton.Background != Brushes.Orange) {
                if(adjButton.Background != Brushes.Green) {
                    adjButton.Background = Brushes.Green;
                    adjButton.BorderBrush = adjButton.Background;
                    adjButton.Click -= Reveal_Click;
                    adjButton.MouseRightButtonDown -= Flagged_Click;
                    checkedFields.Add(adjButton);
                    ReavealAdjacentEmptyFields(adjButton);
                    if(checkedFields.Count == (dimensionX * dimensionY) - InGameNumberOfBombs) {
                        EndGameEventAppear(true);
                    }
                }
```

```
            }
            //jeśli pole z liczbą
            if(Positions[xAdj,yAdj]  != 10 && adjButton.Background != Brushes.Orange) {
                if(adjButton.Background != Brushes.Green) {
                    adjButton.Background = Brushes.Green;
                    adjButton.BorderBrush = adjButton.Background;
                    adjButton.Content = Positions[xAdj, yAdj].ToString();
                    adjButton.Click -= Reveal_Click;
                    adjButton.Click += Number_Click;
                    adjButton.MouseRightButtonDown -= Flagged_Click;
                    checkedFields.Add(adjButton);
                    if(checkedFields.Count == (dimensionX * dimensionY) - InGameNumberOfBombs) {
                        EndGameEventAppear(true);
                    }
                }
            }
        }
    }
}
```

Funkcja Reveal_Click służy do odsłania wartości pól, gdy odsłonięte pole posiada bombę zostaje wywołana funkcja kończąca grę z właściwym parametrem.

Gdy zostanie otwarte puste pole zostaje wywołana funkcja ReavealAdjacentEmptyFields otwierające pozostałe sąsiadujące puste pola oraz okalające je pola z liczbami.

1.10. Flagowanie Bomb

```
private void Flagged_Click(object sender, RoutedEventArgs e) {
    Button button = (Button)sender;
    if(button.Background != Brushes.Orange) {
        //XAML CODE -> <Image Width="40" Height="40" Name="imgDynamic"/>
        //var path = Path.Combine(Environment.CurrentDirectory, "Bilder", "D:\\pcz\\programowanie\\niskopoziomowy\\bomb_grafic\\obraz.jpg");
        //var uri = new Uri(path);
        //imgDynamic.Source = new BitmapImage(uri);
        int x = Convert.ToInt32(button.Tag.ToString().Split(',').GetValue(0));
        int y = Convert.ToInt32(button.Tag.ToString().Split(',').GetValue(1));
        if(Positions[x, y] == 10) {
            checkedFields.Add(button);
            InGameNumberOfBombs--;
            BombCounter.Content = "Bombs left:" + (InGameNumberOfBombs - TooMuchBombs).ToString();
            if(InGameNumberOfBombs == 0 && TooMuchBombs == 0) {
                EndGameEventAppear(true);
            }
        }
        else {
            TooMuchBombs++;
            BombCounter.Content = "Bombs left:" + (InGameNumberOfBombs - TooMuchBombs).ToString();
        }
        button.Background = Brushes.Orange;
        button.Click -= Reveal_Click;
    }
    else {
        int x = Convert.ToInt32(button.Tag.ToString().Split(',').GetValue(0));
        int y = Convert.ToInt32(button.Tag.ToString().Split(',').GetValue(1));
        if(Positions[x, y] == 10) {
            checkedFields.Remove(button);
            InGameNumberOfBombs++;
            BombCounter.Content = "Bombs left:" + (InGameNumberOfBombs - TooMuchBombs).ToString();
        }
        else {
            TooMuchBombs--;
            BombCounter.Content = "Bombs left:" + (InGameNumberOfBombs - TooMuchBombs).ToString();
            if(InGameNumberOfBombs == 0 && TooMuchBombs == 0) {
                EndGameEventAppear(true);
            }
        }
        button.Background = Brushes.Red;
        button.Click += Reveal_Click;
    }
}
```

Funkcja do flagowanie bomb, oznacza pole flagą.

1.11. Dodatkowa funkcja do okrywania pól

```
private void Number_Click(object sender, RoutedEventArgs e) {
    Button button = (Button)sender;
    int flaggedBombCounter = 0;
    int x = Convert.ToInt32(button.Tag.ToString().Split(',').GetValue(0));
    int y = Convert.ToInt32(button.Tag.ToString().Split(',').GetValue(1));

    for(int currentPositionX = -1; currentPositionX < 2; currentPositionX++) {
        int checkX = x + currentPositionX;
        for(int currentPositionY = -1; currentPositionY < 2; currentPositionY++) {
            int checkY = y + currentPositionY;

            // jeśli będzie mniejsze od zera lub wieksze od wymiarów to wychodze poza właściwe wymiary planszy, zmienna nie będzie zliczać samej siebie pole [0,0]
            if(checkX < 0 || checkY < 0 || checkX >= dimensionX || checkY >= dimensionY) { continue; }
            if(checkX == x && checkY == y) { continue; }

            Button adjButton = Buttons[checkX, checkY];

            if(adjButton.Background == Brushes.Orange) { flaggedBombCounter++; }
        }
    }
    if(flaggedBombCounter == Positions[x, y]) {
        for(int currentPositionX = -1; currentPositionX < 2; currentPositionX++) {
            int checkX = x + currentPositionX;
            for(int currentPositionY = -1; currentPositionY < 2; currentPositionY++) {
                int checkY = y + currentPositionY;

                // jeśli będzie mniejsze od zera lub wieksze od wymiarów to wychodze poza właściwe wymiary planszy, zmienna nie będzie zliczać samej siebie pole [0,0]
                if(checkX < 0 || checkY < 0 || checkX >= dimensionX || checkY >= dimensionY) { continue; }
                if(checkX == x && checkY == y) { continue; }

                Button adjButton = Buttons[checkX, checkY];
                if(adjButton.Background != Brushes.Orange && adjButton.Background != Brushes.Green) {
                    Reveal_Click(adjButton, e);
                }
            }
        }
        button.Click -= Number_Click;
    }
}
```

Gdy pole z liczbą zostanie odkryte można nacisnąć na nie lewym przyciskiem mysz i jeśli została oznaczona w okól przycisku liczba bomb określona przez przycisk były odkrywane wszystkie pola w około przycisku. Jeśli pola zostaną źle oflagowane zostaje odkryte pole z bombą kończące gre.

1.12. Rozpoczęcie gry

```
private void startGame_Button_Click(object sender, RoutedEventArgs e) {
    //tworzenie planszy
    GenerateBombs();
    GeneratePositionValue();
    GenerateButtons();
    RemoveButtons();
    AddButtons();

    //ustawienie licznika bomb

    InGameNumberOfBombs = numberOfBombs;
    TooMuchBombs = 0;
    BombCounter.Content = "Bombs left:" + InGameNumberOfBombs.ToString();
    checkedFields = new List<Button>();

    //włączenie licznika czasu
    startOfGame = DateTime.Now;
    timer.Interval = TimeSpan.FromMilliseconds(10);
    timer.Tick += Timer_Tick;
    timer.Start();

}
```

Funkcja startGame_Button_Click rozpoczyna nową grę lub resetuje już trwającą.

1.13. Przycisk do zmiany ustawień

```
private void SettingWindowCall_Click(object sender, RoutedEventArgs e) {
    timer.Stop();
    var newSettings = new Settings_windowxaml();
    var result = newSettings.ShowDialog();
    if (result == true) {
        RemoveButtons();
        // ustawianie nowych wartości
        dimensionX = newSettings.Width;
        dimensionY = newSettings.Height;
        numberOfBombs = newSettings.Bombs;

        Positions = new short[dimensionX, dimensionY];
        Buttons = new Button[dimensionX, dimensionY];

    }
}
```

Przycisk do zmiany ustawień wywołuje nowe okno w którym można podać nowe wartości do zmiany wymiarów planszy oraz ilości bomb.

```
public partial class Settings_windowxaml : Window{
    2 references
    public uint Bombs { get; set; }
    2 references
    public int Width { get; set; }
    2 references
    public int Height { get; set; }
    1 reference
    public Settings_windowxaml() {
        InitializeComponent();
        TextboxBombs.Text = "10";
        TextboxWidth.Text = "10";
        TextboxHeight.Text = "10";
    }
    1 reference
    private void SaveSettingButton_Click(object sender, RoutedEventArgs e) {
        Bombs = Convert.ToUInt32(TextboxBombs.Text);
        Width = Convert.ToInt32(TextboxWidth.Text);
        Height = Convert.ToInt32(TextboxHeight.Text);

        DialogResult = true;
    }
}
```

2. Uruchomienie gry

Aby uruchomić gre należy rozpakować plik Saper.zip

I wewnątrz folderu „Saper” uruchomić plik Saper.exe

Zasady gry są takie same jak w zwykłym saperze, aby wygrać trzeba albo odkryć wszystkie pola poza tymi posiadającymi bombę albo oflagować wszystkie pola z bombą

Dla ułatwienia testowania w funkcji tworzącej przyciski GenerateButtons jest zakomentowana linijka, która wyświetla wartości pół.

Możliwe wartości pól:

1 - 8 – pola z liczbą bomb okalającą je

10 – pole z bombą

11 – puste pole