

Projekt Semestralny

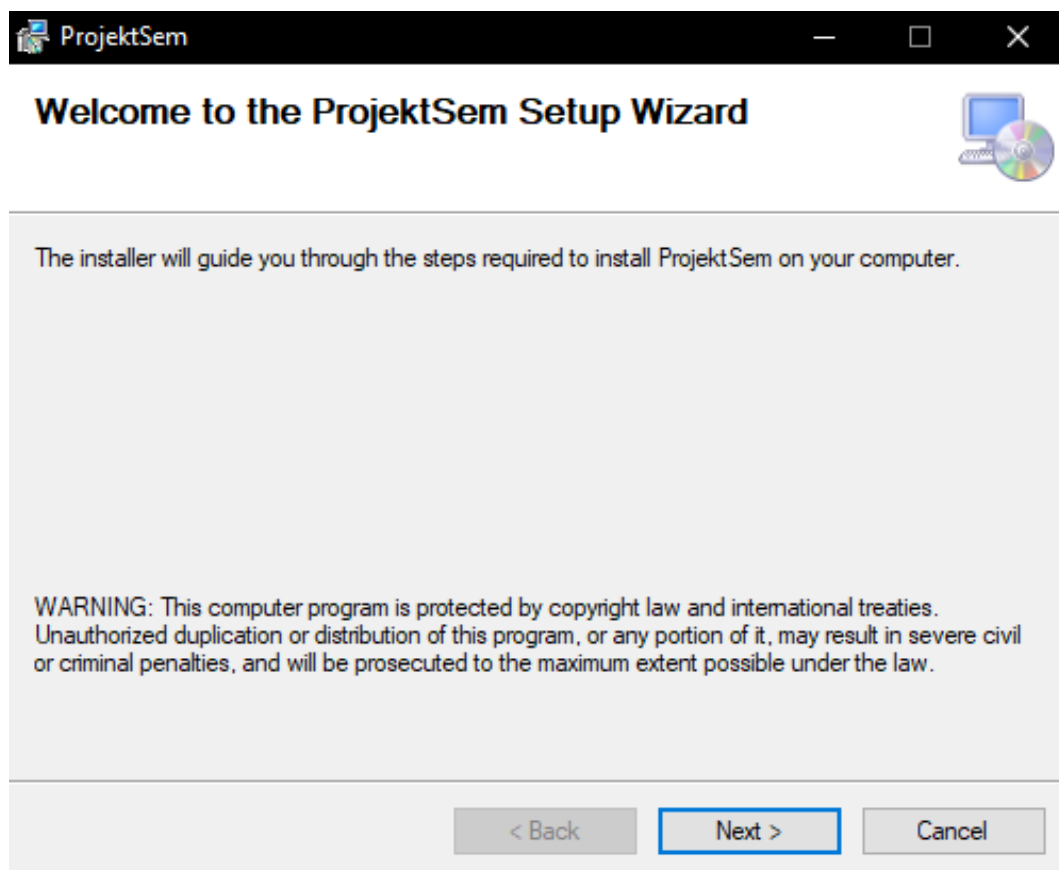
Opis programu

Omawiany w tej instrukcji program pozwala na tworzenie prostych rysunków na różnej wielkości planszach, oraz tworzenie krótkich animacji z wykorzystaniem gotowych plansz.

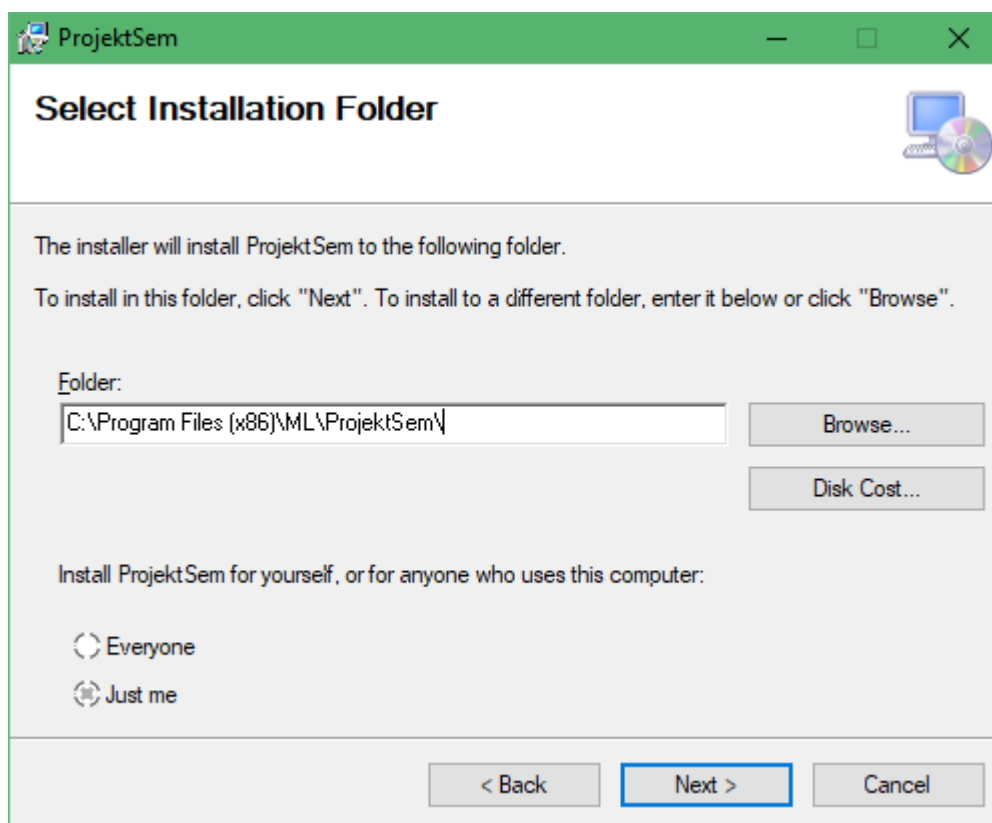
Instalacja

Aby zainstalować program uruchamiamy plik setup.exe pod ścieżką
ProjektSemestralny\Setup\Release

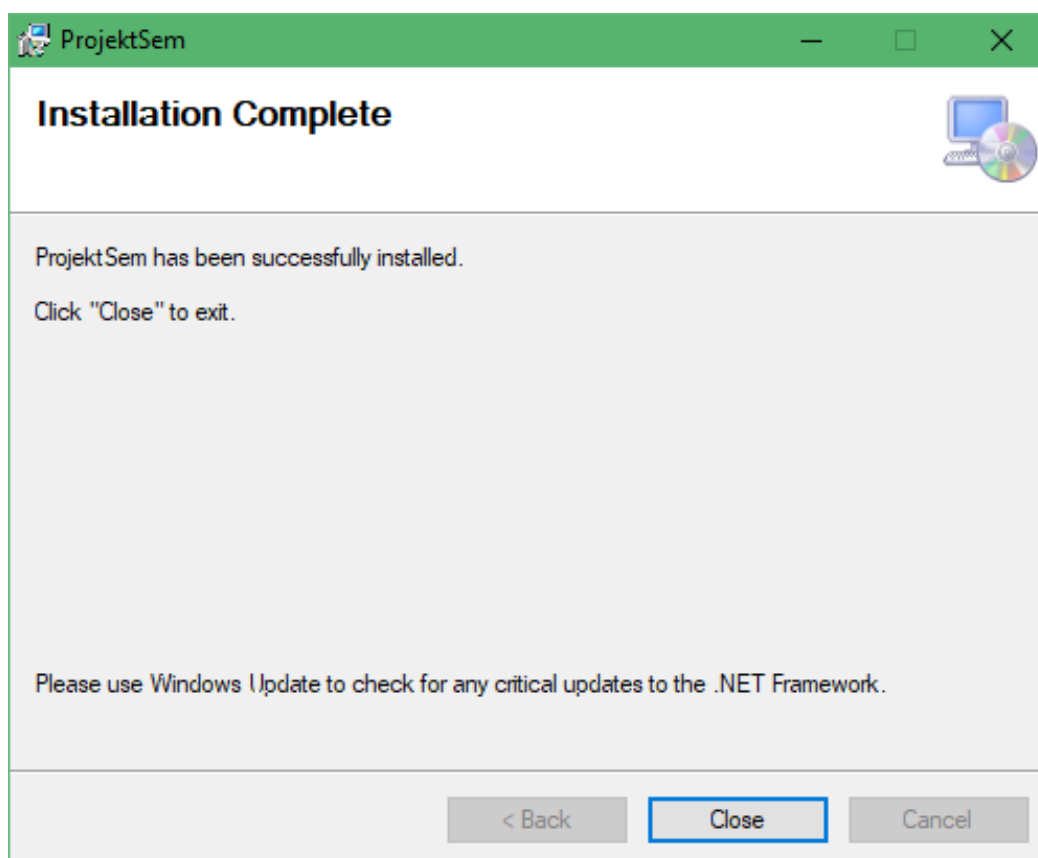
Pojawi się okno instalacji. Klikamy next.



Wybieramy folder docelowy i klikamy next.



Klikamy kolejne next, czekamy aż program się zainstaluje i klikamy close



Gotowe! Program został zainstalowany pod podaną ścieżką.

Opis działania programu

1. Menu główne

W celu stworzenia animacji potrzebujemy kilka gotowych planszy.

W pierwszej kolejności Klikamy przycisk „Nowy pusty projekt”.

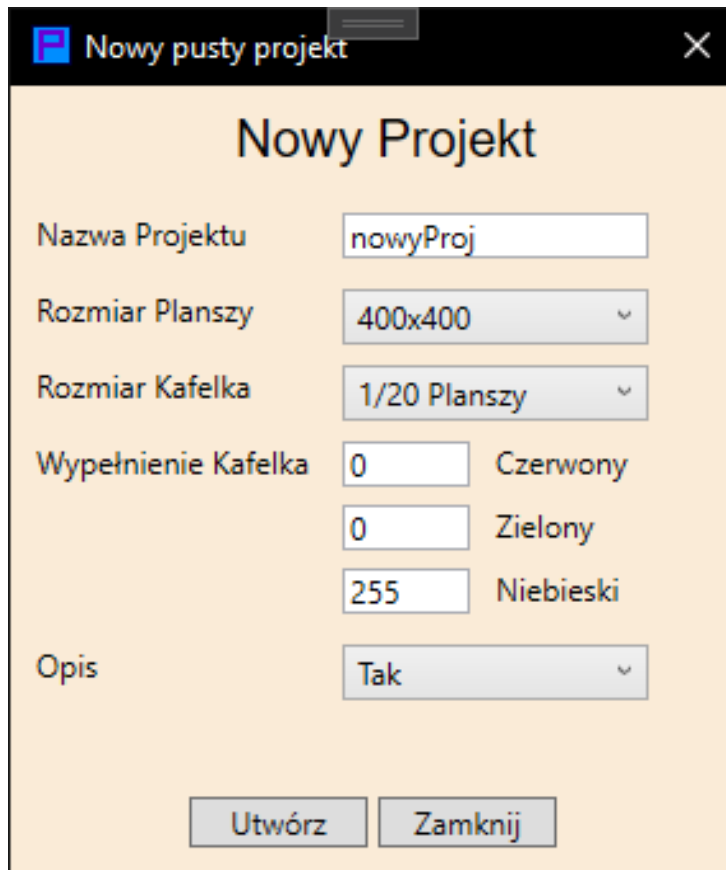


2. Nowy pusty projekt

- wpisujemy nazwę swojej planszy (bez polskich znaków, max. 15 znaków),
- wybieramy wielkość planszy z dostępnych (400x400, 640x640, 800x800),
- rozmiar elementów na planszy (różne wielkości w zależności od wielkości planszy)
- domyślny kolor elementów na planszy w kolorze RGB (wybieramy wartości z przedziału 0-255)
- prywatny opis planszy (zaznaczamy tak/nie)

Po wprowadzaniu wszystkich danych klikamy przycisk „Utwórz”. Po kilku sekundach okno nowego projektu powinno się zamknąć.

- Przycisk zamknij – w celu zamknięcia okna bez zapisywania zmian



The screenshot shows a dialog box titled "Nowy pusty projekt" with a close button (X) in the top right corner. The main heading is "Nowy Projekt". Below it, there are several input fields and dropdown menus:

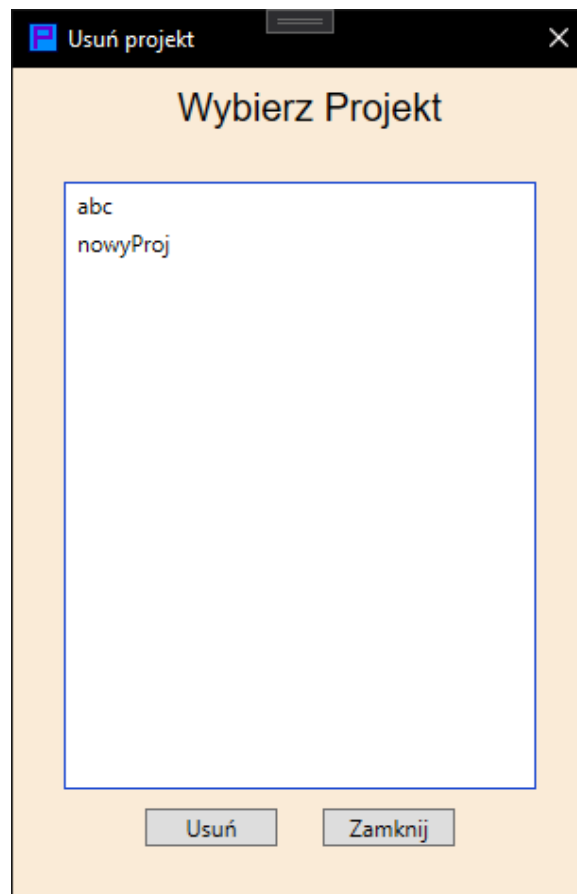
- Nazwa Projektu:** A text input field containing "nowyProj".
- Rozmiar Planszy:** A dropdown menu showing "400x400".
- Rozmiar Kafełka:** A dropdown menu showing "1/20 Planszy".
- Wypełnienie Kafełka:** Three color selection options, each with a small input field and a color name:
 - Czerwony: Input field contains "0".
 - Zielony: Input field contains "0".
 - Niebieski: Input field contains "255".
- Opis:** A dropdown menu showing "Tak".

At the bottom of the dialog, there are two buttons: "Utwórz" and "Zamknij".

3. Usuń projekt

W celu usunięcia niechcianego projektu klikamy przycisk „Usuń projekt” w menu głównym.

Pojawi się nowe okno. Wybieramy niechcianą planszę z listy i klikamy przycisk „Usuń”.



The screenshot shows a dialog box titled "Usuń projekt" with a close button (X) in the top right corner. The main heading is "Wybierz Projekt". Below it, there is a list box containing two items:

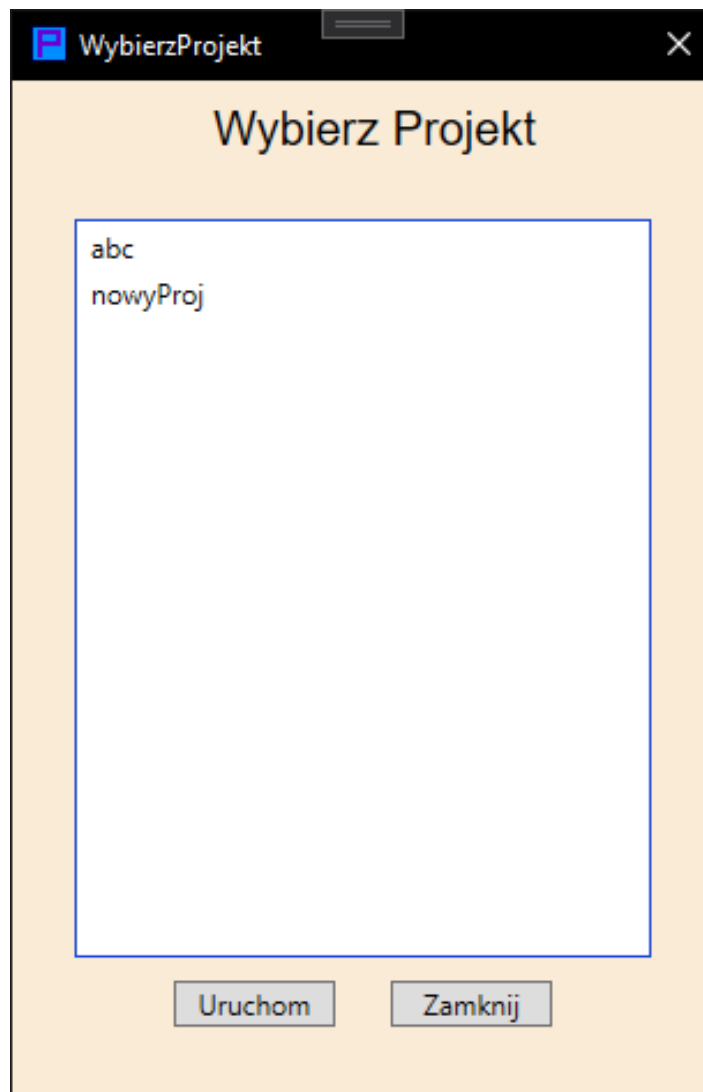
- abc
- nowyProj

At the bottom of the dialog, there are two buttons: "Usuń" and "Zamknij".

4. Wybierz projekt

Aby wprowadzić zmiany na naszej planszy klikamy przycisk „Wybierz projekt” w menu głównym.

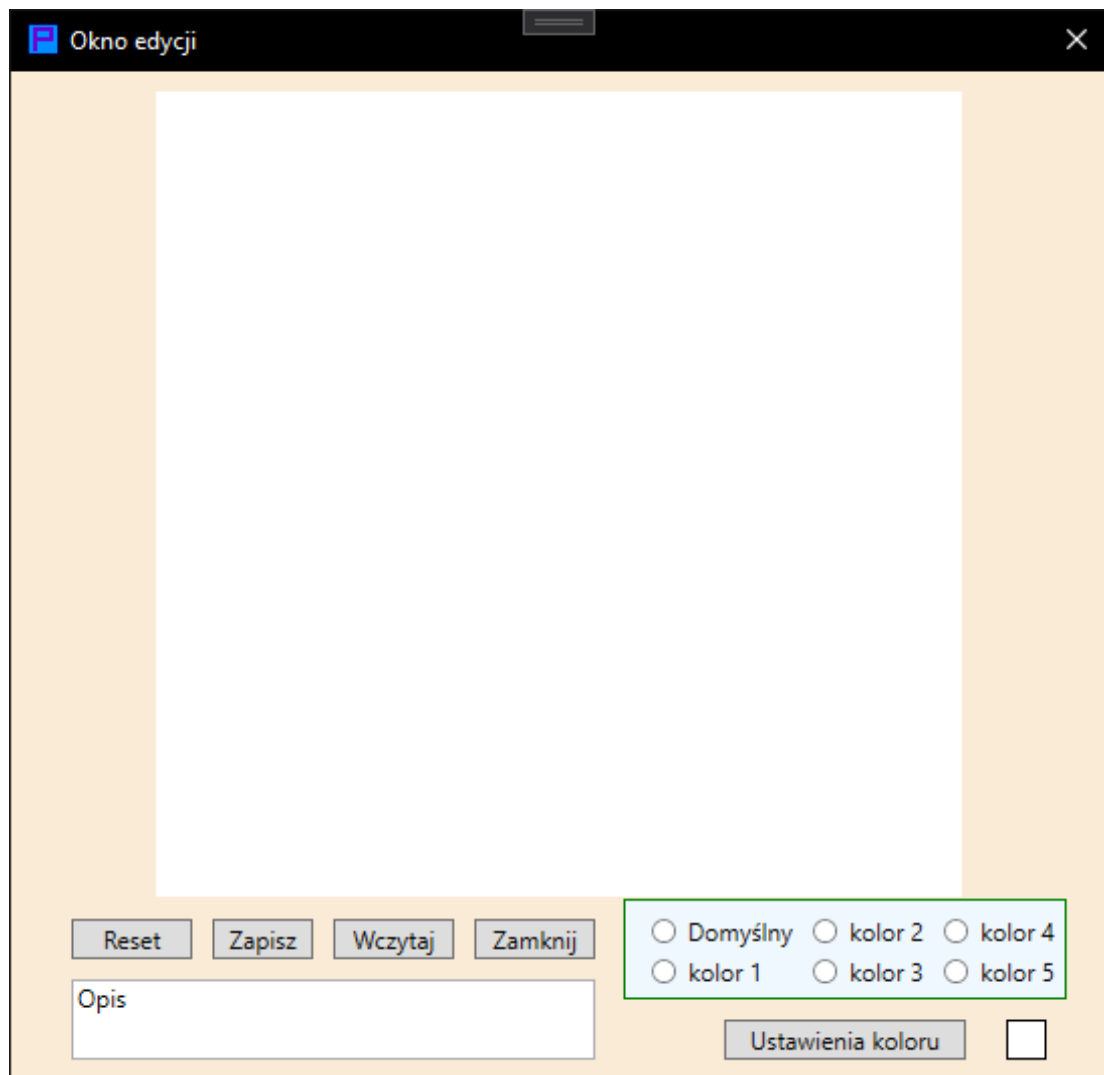
Po otwarciu nowego okna wybieramy nasz projekt z listy i klikamy Uruchom.



5. Okno edycji

Po kliknięciu przycisku „Uruchom” w kroku 4 powinno otworzyć się okno edycji.

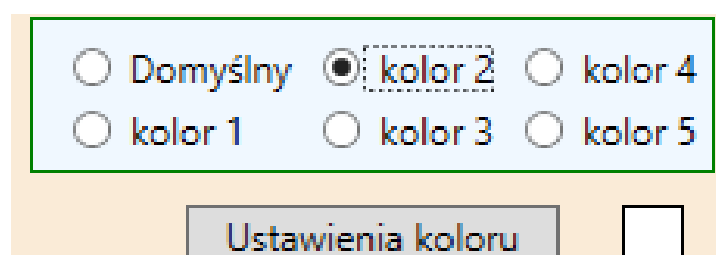
W pierwszej kolejności klikamy przycisk „Wczytaj” (plansza powinna zmienić kolor na domyślny)



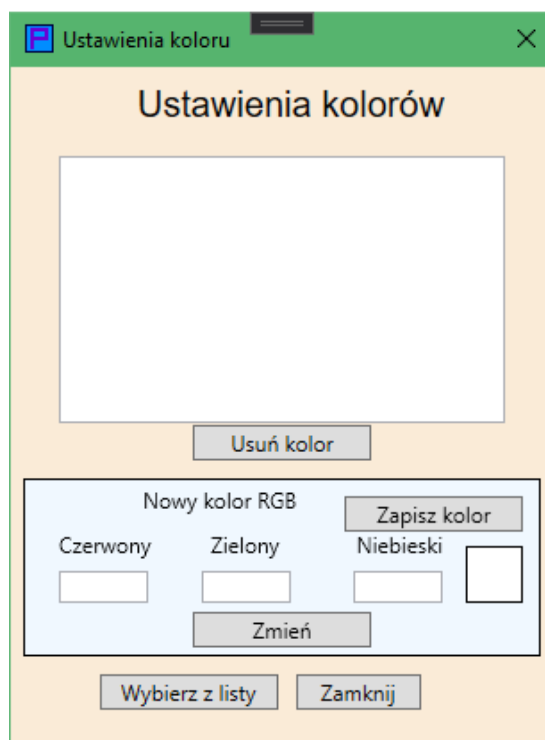
Następnie ustawiamy paletę kolorów po prawej stronie okna.

- Domyślny – domyślny kolor jaki ma plansza (nie można jej zmienić)
- kolor 1-5 – dostępny kolor (domyślnie biały)

Klikamy jeden z kolorów oznaczonych numerami od 1 do 5 i klikamy ustawienia koloru.

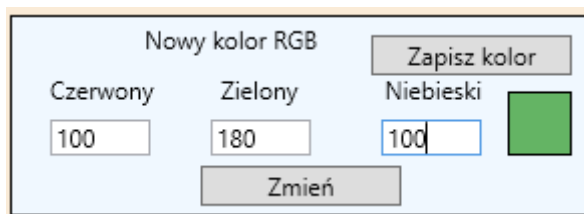


6. Ustawienia koloru

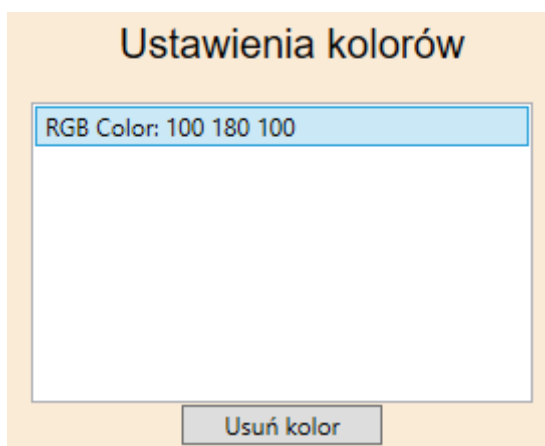


Aby stworzyć nowy kolor wprowadź wartość koloru RGB: czerwony, zielony, niebieski (wartości z przedziału 0-255) i kliknij przycisk „Zmień”.

Powinien zmienić się kolor kwadratu na ten wybrany przez nas.



Następnie klikamy przycisk „Zapisz kolor” aby zapisać go w liście (lista zapisanych kolorów jest dostępna przy zmianie każdego koloru w projekcie)

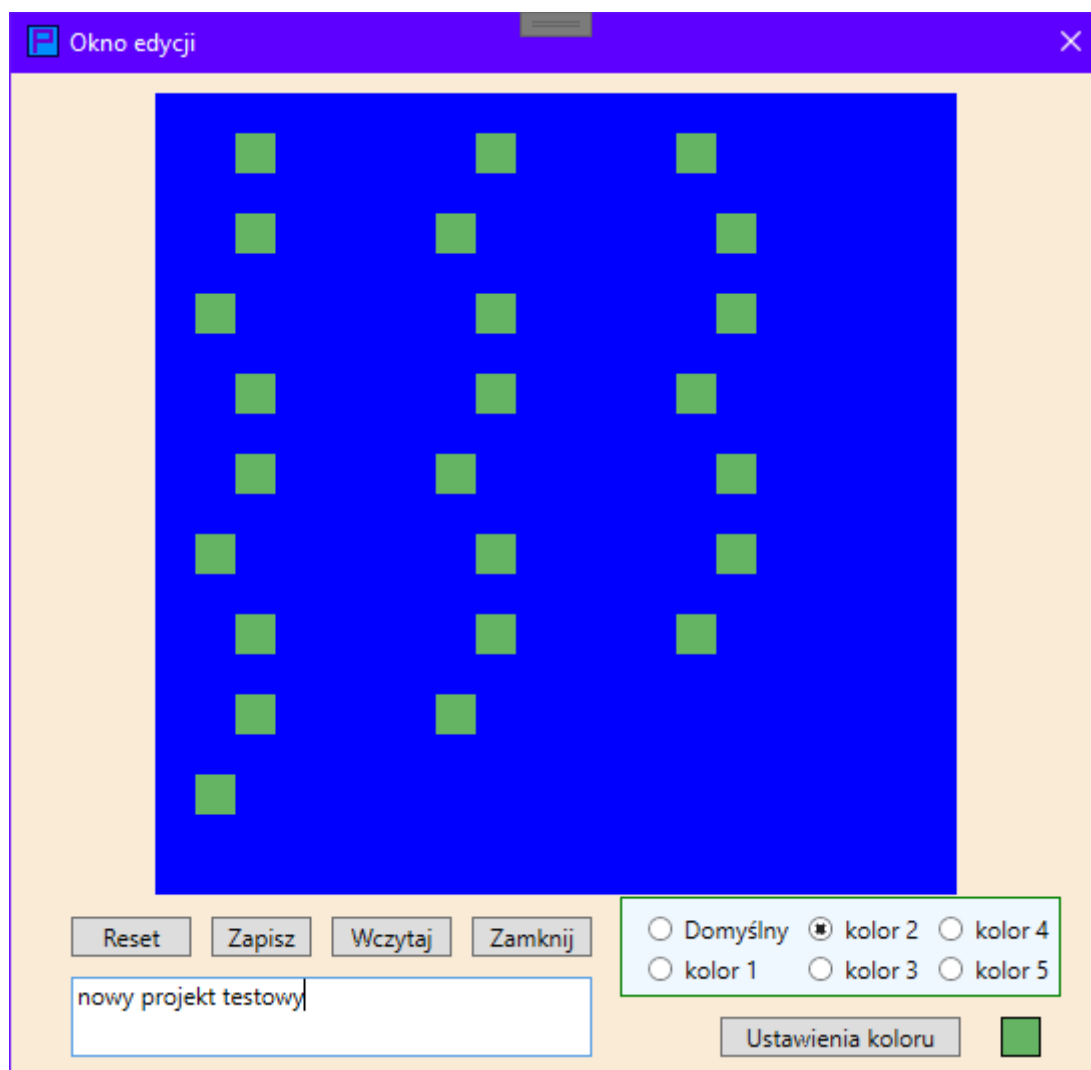


-Przycisk „Usuń kolor” usuwa wybrany kolor z listy.

Wybieramy nasz kolor z listy i klikamy przycisk „Wybierz z listy”.

Teraz możemy wprowadzić zmiany w naszym rysunku.

Klikając na plansze LPM automatycznie zmieniamy kolor elementu na ten sam co w prawym dolnym rogu okna.



-Opis zapisuje się automatycznie przy wprowadzaniu zmian,

- Reset – ustawia zawartość planszy na domyślną
- Zapisz – zapisuje aktualną planszę
- Wczytaj – wczytuje ostatnio zapisaną planszę.
- Zamknij – zamyka okno edycji.

Po wprowadzeniu wszystkich zmian na planszy klikamy „Zapisz” a następnie zamknij.

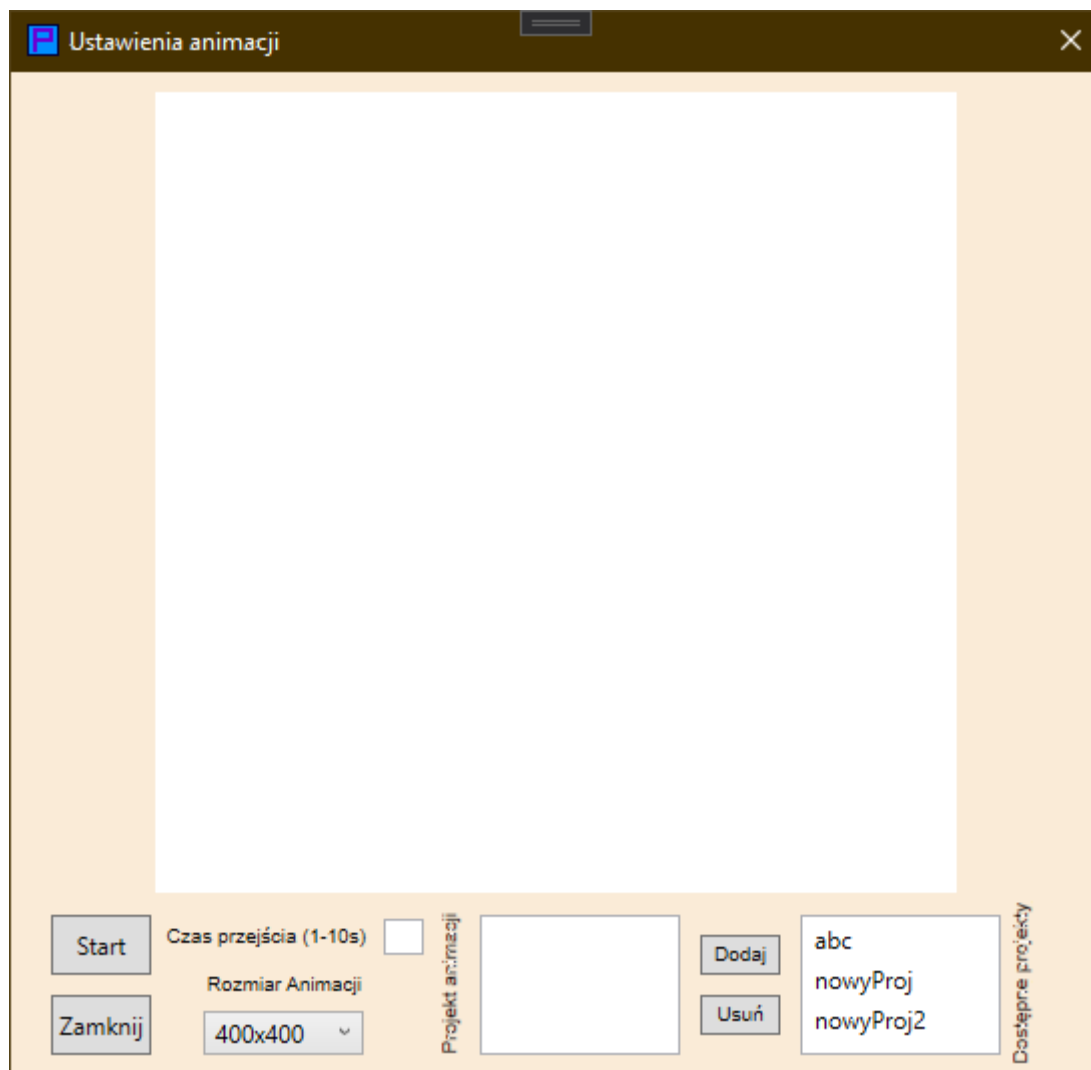
7. Ustawienia animacji

Po kliknięciu przycisku w menu powinno pojawić się nowe okno

Na początku wybieramy rozmiar planszy pod napisem „Rozmiar animacji” z dostępnych 400x400, 640x640, 800x800

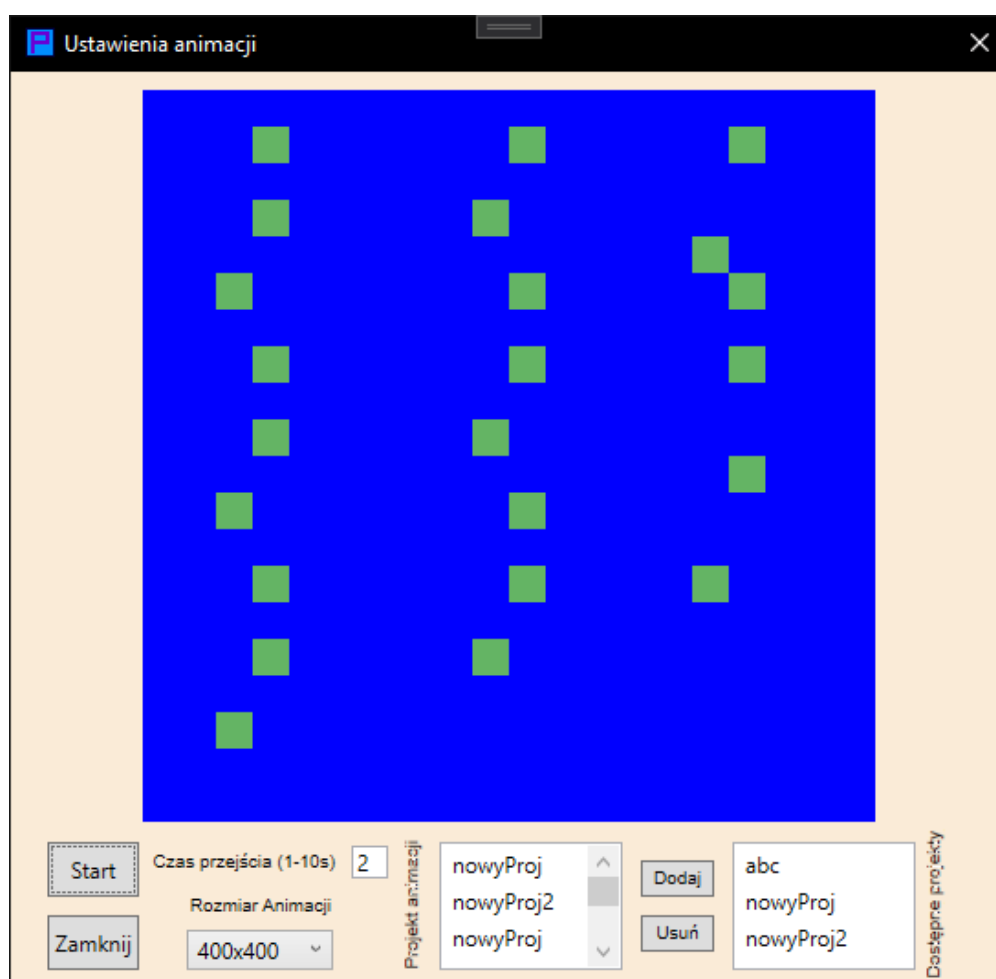
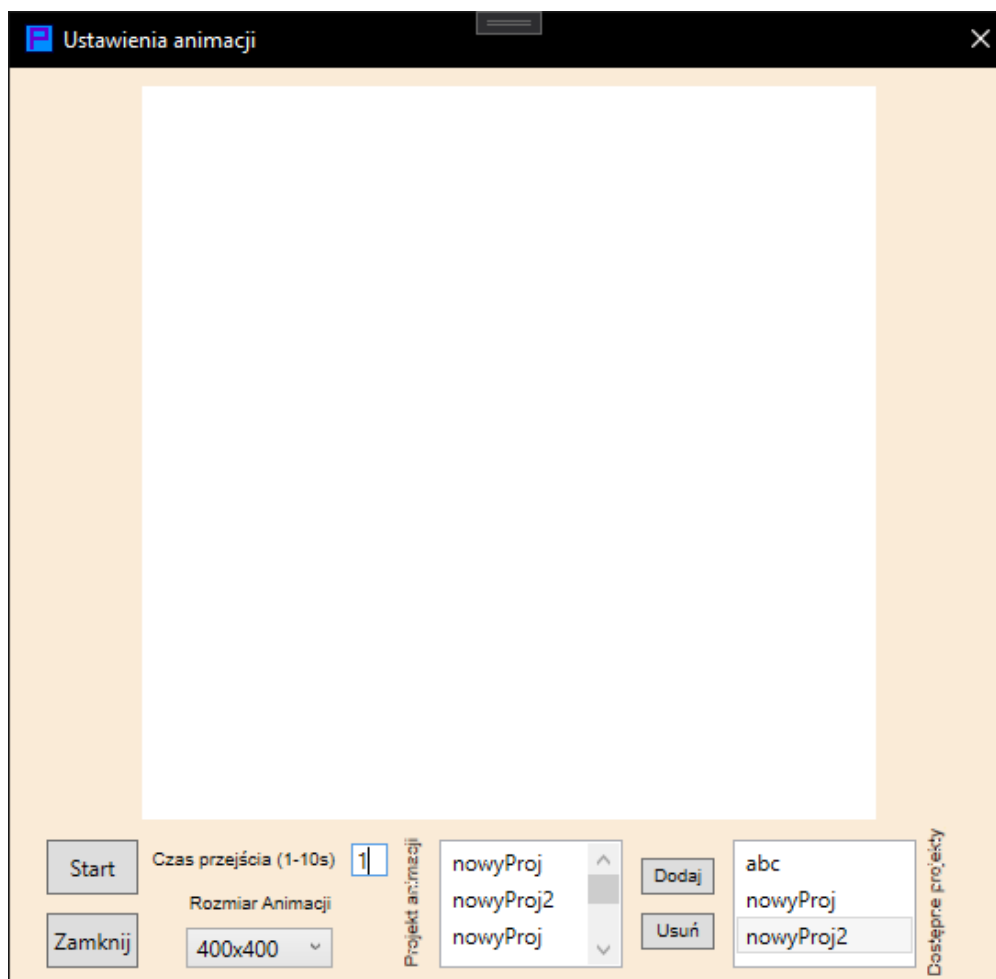
Po prawej stronie mamy listę dostępnych projektów.

Teraz możemy przyciskami „Dodaj” i „Usuń” edytować kolejkę w naszej animacji.



Dodajemy interesujące nas projekty do drugiej listy,

Następnie ustawiamy czas przejścia między wyświetlanymi planszami (1-10 sekund) i klikamy przycisk „Start”.



Koniec!

Opis kodu programu

1. MainWindow.xaml.cs

Inicjalizacja zawartości okna menu głównego i wycentrowanie widoku.

```
public MainWindow()  
{  
    WindowStartupLocation = WindowStartupLocation.CenterScreen;  
    InitializeComponent();  
}
```

Przyciski otwarcia okien nowego projektu, usunięcia projektu, wybrania projektu, animacji.

```
private void Button_Click(object sender, RoutedEventArgs e)  
{  
    NowyProjekt win = new NowyProjekt();  
    win.ShowDialog();  
}  
  
private void Button_Click_1(object sender, RoutedEventArgs e)  
{  
    UsunProjekt win = new UsunProjekt();  
    win.ShowDialog();  
}  
  
private void Button_Click_2(object sender, RoutedEventArgs e)  
{  
    WybierzProjekt win = new WybierzProjekt();  
    win.ShowDialog();  
}  
  
private void Button_Click_3(object sender, RoutedEventArgs e)  
{  
    UstawieniaAnimacji win = new UstawieniaAnimacji();  
    win.ShowDialog();  
}
```

2. NowyProjekt.xaml.cs

Inicjalizacja zawartości okna menu głównego i wycentrowanie widoku.

```
public NowyProjekt()  
{  
    WindowStartupLocation = WindowStartupLocation.CenterScreen;  
    InitializeComponent();  
}
```

Przycisk zamknij

```
private void Button_Click(object sender, RoutedEventArgs e)  
{  
    Close();  
}
```

Przycisk nowy projekt

```
private void Button_Click_1(object sender, RoutedEventArgs e)  
{  
    newProject();  
}
```

```
private void newProject()
```

```
byte green = 0; byte red = 0; byte blue = 0;
```

Komunikat po wpisaniu błędnych wartości koloru

```
try
{
if (greenFill != null)
green = byte.Parse(greenFill.Text);
if (redFill != null)
red = byte.Parse(redFill.Text);
if (blueFill != null)
blue = byte.Parse(blueFill.Text);
}
catch (Exception)
{
MessageBox.Show("Wpisano błędne wartości kolorów!");
}
```

Sprawdzenie poprawności wpisanych danych

```
if (
projectName.Text != ""
& sizeBoard.SelectedItem != null
& squareSize.SelectedItem != null
& Description.SelectedItem != null
)

if (projectName.Text.Length > 15)
    MessageBox.Show("Nazwa projektu nie może mieć więcej niż 15 znaków!");
```

Przypisanie wartości początkowych w bazie danych

```
int sizeBoardNumber = 0;
int sizeSquareNumber = 0;
bool desc = false;
if (sizeBoard.SelectedItem == sizeBoard640) sizeBoardNumber = 640;
else if (sizeBoard.SelectedItem == sizeBoard800) sizeBoardNumber = 800;
else if (sizeBoard.SelectedItem == sizeBoard400) sizeBoardNumber = 400;
```

Ustawienie Control boxa z wielkościami elementów planszy

```
if (squareSize.Text == "1/64 Planszy") sizeSquareNumber = sizeBoardNumber / 64;
else if (squareSize.Text == "1/40 Planszy") sizeSquareNumber = sizeBoardNumber / 40;
else if (squareSize.Text == "1/32 Planszy") sizeSquareNumber = sizeBoardNumber / 32;
else if (squareSize.Text == "1/20 Planszy") sizeSquareNumber = sizeBoardNumber / 20;
else if (squareSize.Text == "1/16 Planszy") sizeSquareNumber = sizeBoardNumber / 16;
```

Opis tak/nie

```
if (Description.SelectedItem == yesDescription) desc = true;
```

Wprowadzanie wartości do bazy danych do tabeli NewProject

```
ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
NewProject newItem = new NewProject()
{
    projectName = projectName.Text,
    boardSize = sizeBoardNumber,
    description = desc,
    squareSize = sizeSquareNumber,
    descNew = "Opis"
};
```

Wprowadzanie informacji o domyślnym kolorze do bazy danych do tabeli SquareFill

```
SquareFill newSquare = new SquareFill()
{
    defaultRed = red,
    defaultGreen = green,
    defaultBlue = blue
};
```

Wprowadzanie domyślnych kolorów palety(białych) do bazy danych do tabeli DefaultColor

```
for (int i = 1; i <= 6; i++)
{
    DefaultColor defaultColor = new DefaultColor()
    {
        rgb_red = 255,
        rgb_blue = 255,
        rgb_green = 255,
        positionNumber = (byte)i
    };
    db.DefaultColors.Add(defaultColor);
}

db.NewProjects.Add(newItem);
db.SquareFills.Add(newSquare);
```

Wprowadzanie ilości elementów i domyślnego koloru do bazy danych do tabeli BoardColor

```
for (int i = 1; i <= (sizeBoardNumber / sizeSquareNumber) * (sizeBoardNumber / sizeSquareNumber); i++)
{
    BoardColor boardCol = new BoardColor()
    {
        rgb_blue = blue,
        rgb_green = green,
        rgb_red = red,
        square_number = i
    };
    db.BoardColors.Add(boardCol);
}
db.SaveChanges();
```

Sprawdzenie czy aktualna tablica z globalnymi wartościami jest pusta i wprowadzenie wartości w razie pustej tablicy – tabela GlobalValues

```
var globValue = from l in db.GlobalValues
select l;
if (globValue.Any() == false)
{
    GlobalValue global = new GlobalValue()
    {
        actualProject = 0,
    };
    db.GlobalValues.Add(global);
    db.SaveChanges();
}
```

Sprawdzenie czy aktualna tablica z globalnymi kolorami jest pusta i wprowadzenie wartości w razie pustej tablicy – tabela GlobalColor

```
var globCol = from c in db.GlobalColors
select c;
if (globCol.Any() == false)
{
    GlobalColor color = new GlobalColor()
    {
        choosenColorRed = 255,
        choosenColorBlue = 255,
        choosenColorGreen = 255,
    };
    db.GlobalColors.Add(color);
    db.SaveChanges();
}
```

Wprowadzanie domyślnego koloru (białego) do bazy danych do tabeli GlobalColor

```
foreach (var item in globCol)
{
    item.choosenColorRed = 255;
    item.choosenColorBlue = 255;
    item.choosenColorGreen = 255;
}
```

Zapisanie elementów i zamknięcie okna

```
db.SaveChanges();
Close();
```

Zmiana zawartości Control Boxa wielkości elementów przy wyborze wielkości planszy

```
private void sizeBoard_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (sizeBoard.SelectedItem == sizeBoard640)
    {
        squareSize.Items.Clear();
        ComboBoxItem item = new ComboBoxItem();
        ComboBoxItem item2 = new ComboBoxItem();
        ComboBoxItem item3 = new ComboBoxItem();
        ComboBoxItem item4 = new ComboBoxItem();
        ComboBoxItem item5 = new ComboBoxItem();
        item.Name = "squareSize64";
        item.Content = "1/64 Planszy";
        squareSize.Items.Add(item);
        item2.Name = "squareSize32";
        item2.Content = "1/32 Planszy";
        squareSize.Items.Add(item2);
        item3.Name = "squareSize16";
        item3.Content = "1/16 Planszy";
        squareSize.Items.Add(item3);
        item4.Name = "squareSize40";
        item4.Content = "1/40 Planszy";
        squareSize.Items.Add(item4);
        item5.Name = "squareSize20";
        item5.Content = "1/20 Planszy";
        squareSize.Items.Add(item5);
    }
    else if (sizeBoard.SelectedItem == sizeBoard800)
    {
        squareSize.Items.Clear();
        ComboBoxItem item2 = new ComboBoxItem();
        ComboBoxItem item3 = new ComboBoxItem();
        ComboBoxItem item4 = new ComboBoxItem();
        ComboBoxItem item5 = new ComboBoxItem();

        item2.Name = "squareSize40";
        item2.Content = "1/40 Planszy";
        squareSize.Items.Add(item2);
        item3.Name = "squareSize20";
        item3.Content = "1/20 Planszy";
        squareSize.Items.Add(item3);
        item4.Name = "squareSize32";
        item4.Content = "1/32 Planszy";
        squareSize.Items.Add(item4);
        item5.Name = "squareSize16";
        item5.Content = "1/16 Planszy";
        squareSize.Items.Add(item5);
    }
    else if (sizeBoard.SelectedItem == sizeBoard400)
    {
        squareSize.Items.Clear();
        ComboBoxItem item = new ComboBoxItem();
        ComboBoxItem item2 = new ComboBoxItem();
        ComboBoxItem item3 = new ComboBoxItem();
        item.Name = "squareSize40";
        item.Content = "1/40 Planszy";
        squareSize.Items.Add(item);
        item2.Name = "squareSize20";
        item2.Content = "1/20 Planszy";
        squareSize.Items.Add(item2);
        item3.Name = "squareSize16";
        item3.Content = "1/16 Planszy";
        squareSize.Items.Add(item3);
    }
}
```

3. UsuńProjekt.xaml.cs

Inicjalizacja zawartości okna menu głównego i wycentrowanie widoku.

Wprowadzenie elementów z tablicy NewProject do pustej listy

```
public UsunProjekt()
{
    WindowStartupLocation = WindowStartupLocation.CenterScreen;
    InitializeComponent();
    projectList.Items.Clear();

    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var proj = from p in db.NewProjects
        select p;
    foreach (var item in proj)
    {
        projectList.Items.Add(item.projectName);
    }
}
```

Przycisk zamknij

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    Close();
}
```

Przycisk usuwający projekty z bazy. Sprawdza wszystkie powiązane tabele z NewProject i usuwa elementy o tym samym projectId

```
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    if (projectList.SelectedItem != null)
    {
        ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();

        var proj = from p in db.NewProjects
            select p;
        int idNumer = 0;

        foreach (var item in proj)
        {
            if (idNumer == projectList.SelectedIndex)
            {
                projectList.Items.Remove(item);

                var def = from d in db.DefaultColors
                    where d.id_project == item.id_project
                    select d;
                foreach (var d_item in def)
                {
                    db.DefaultColors.Remove(d_item);
                }
                var square = from s in db.SquareFills
                    where s.id_project == item.id_project
                    select s;
                foreach (var s_item in square)
                {
                    db.SquareFills.Remove(s_item);
                }

                var board = from b in db.BoardColors
                    where b.id_project == item.id_project
                    select b;
                foreach (var b_item in board)
                {
                    db.BoardColors.Remove(b_item);
                }
            }
            idNumer++;
        }
    }
}
```



```
db.NewProjects.Remove(item);

var anim= from a in db.AnimationBoard400
where a.id_project == item.id_project
select a;
foreach (var a_item in anim)
db.AnimationBoard400.Remove(a_item);

var anim2 = from a in db.AnimationBoard640
where a.id_project == item.id_project
select a;
foreach (var a_item in anim2)
db.AnimationBoard640.Remove(a_item);

var anim3 = from a in db.AnimationBoard800
where a.id_project == item.id_project
select a;
foreach (var a_item in anim3)
db.AnimationBoard800.Remove(a_item);

db.NewProjects.Remove(item);

}
idNumer++;
}
db.SaveChanges();

Close();
}
else MessageBox.Show("Nie wybrano projektu!");

}
}
```

4. WybierzProjekt.xaml.cs

Inicjalizacja zawartości okna menu głównego i wycentrowanie widoku.

Wprowadzenie elementów z tablicy NewProject do pustej listy

```
public WybierzProjekt()
{
    WindowStartupLocation = WindowStartupLocation.CenterScreen;
    InitializeComponent();
    projectList.Items.Clear();

    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var proj = from p in db.NewProjects
        select p;
    foreach(var item in proj)
    {
        projectList.Items.Add(item.projectName);
    }
}
```

Przycisk zamknij

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    Close();
}
```

Przycisk wyboru projektu

Sprawdza projectId wybranego projektu z listy w tablicy NewProject i zapisuje go w tablicy GlobalValue jako actualProject. Następnie otwiera okno edycji.

```
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    if (projectList.SelectedItem != null)
    {
        ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
        var globValue = from l in db.GlobalValues
            select l;
        var proj = from p in db.NewProjects
            select p;

        int idNumer = 0;
        foreach (var item in proj)
        {
            foreach(var glob in globValue)
            {
                if (idNumer == projectList.SelectedIndex)
                {
                    glob.actualProject = item.id_project;
                }
                idNumer++;
            }
            db.SaveChanges();
            OknoEdycji win = new OknoEdycji();
            Close();
            win.Show();
        }
        else MessageBox.Show("Nie wybrano projektu!");
    }
}
```

5. OknoEdycji.xaml.cs

Tablice zawierające informacje o wartości koloru RGB każdego elementu w aktualnie wyświetlanej planszy.

```
private byte[] red_color = new byte[10000];
private byte[] green_color = new byte[10000];
private byte[] blue_color = new byte[10000];
```

Tekst opisu do projektu zapisywany w bazie danych

```
private string desc_prop;
```

Inicjalizacja zawartości okna menu głównego i wycentrowanie widoku.

Zmiana wielkości okna i wyświetlanej planszy w zależności od wielkości planszy wyświetlanego projektu

```
public OknoEdycji()
{
    WindowStartupLocation = WindowStartupLocation.CenterScreen;
    InitializeComponent();

    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var globValue = from l in db.GlobalValues
                    select l;
    foreach (var item in globValue)
    {
        var proj = from p in db.NewProjects
                    where p.id_project == item.actualProject
                    select p;
        foreach (var item2 in proj)
        {
            OknoPar.Height = 140 + item2.boardSize;
            OknoPar.Width = 160 + item2.boardSize;
            MainLayer.Height = item2.boardSize;
            MainLayer.Width = item2.boardSize;
        }
    }
    DataContext = this;
    refreshColor();
    LoadBase();
}
```

Ustawienie koloru kwadratu wyświetlanego w prawym dolnym rogu okna

```
public void refreshColor()
{
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var globValue = from l in db.GlobalColors
                    select l;
    foreach (var value in globValue)
    {
        kwadracik = new SolidColorBrush(Color.FromRgb(
            value.chosenColorRed, value.chosenColorGreen, value.chosenColorBlue));
    }
}
```

Wprowadzenie wartości kolorów RGB z tablicy BoardColors do tablic: red_color, green_color, blue_color

Blokada pola opisu w przypadku gdy projekt jest stworzony bez niego

```
private void LoadBase()
{
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();

    int k = 0;
    var globVal = from g in db.GlobalValues
                  select g;
    var proj = from p in db.NewProjects
              select p;

    foreach (var gv in globVal)
        foreach (var item in proj)
            if (item.id_project == gv.actualProject)
            {
                var board = from b in db.BoardColors
                           where b.id_project == gv.actualProject
                           select b;
                foreach (var square in board)
                {
                    blue_color[k] = square.rgb_blue;
                    red_color[k] = square.rgb_red;
                    green_color[k] = square.rgb_green;
                    k++;
                }
                if (item.description) desc_prop = item.descNew;
                else
                {
                    desc_prop = "";
                    Main_desc.IsReadOnly = true;
                }
            }
    db.SaveChanges();
    Main_desc.Text = desc_prop;
}
```

Metody odpowiadające za zmianę koloru kwadratu w oknie edycji po zamknięciu okna ustawień koloru

```
private Brush _kwadracik;

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propertyName)
{
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
}

public Brush kwadracik
{
    get { return _kwadracik; }
    set
    {
        _kwadracik = value;
        OnPropertyChanged(nameof(kwadracik));
    }
}
```

Przycisk zamknij z komunikatem

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    MessageBoxResult result = MessageBox.Show(
        "Niezapisane zmiany zostaną utracone. Chcesz zamknąć projekt?"
        , "Uwaga!", MessageBoxButton.YesNo);
    if (result == MessageBoxResult.Yes) Close();
}
```

Przycisk Ustawienia Koloru. Sprawdza wciśnięty radioButton i otwiera okno ustawień koloru

```
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    if (c10.IsChecked == false)
    {
        if (c11.IsChecked == true |
            c12.IsChecked == true |
            c13.IsChecked == true |
            c14.IsChecked == true |
            c15.IsChecked == true
        )
        {
            UstawieniaKoloru win = new UstawieniaKoloru(this);
            win.Owner = this;
            win.ShowDialog();
        }
        else MessageBox.Show("Nie wybrano koloru!");
    }
    else MessageBox.Show("Nie można zmienić domyślnego koloru!");
}
```

Usuwa wszystkie elementy z planszy i wprowadza nowe

```
private void AddSquare()
{
    MainLayer.Children.Clear();
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var globValue = from l in db.GlobalValues
        select l;

    var proj = from p in db.NewProjects
        select p;
    byte red = 0; byte green = 0; byte blue = 0;
    int k = 0;

    foreach (var glob in globValue)
        foreach (var item in proj)
```

Sprawdzenie obecnego projektu

```
        if (item.id_project == glob.actualProject)
```

Tworzenie nowych kwadratów na planszy na podstawie wielkości planszy i elementów.
Wprowadzenie domyślnego koloru elementów.

```
        for (int i = 0; i < item.boardSize / item.squareSize; i++)
            for (int j = 0; j < item.boardSize / item.squareSize; j++)
            {
                var fl = from f in db.SquareFills
                    where f.id_project == item.id_project
                    select f;
                if (i == 0 & j == 0)
                    foreach (var color in fl)
                    {
                        red = color.defaultRed;
                        green = color.defaultGreen;
                        blue = color.defaultBlue;
                    }

                Rectangle r = new Rectangle
                {
                    Height = item.squareSize,
                    Width = item.squareSize,
                    Fill = new SolidColorBrush(Color.FromRgb(red, green, blue)),
                    Name = "s" + k.ToString()

                };

                r.VerticalAlignment = VerticalAlignment.Top;
                r.HorizontalAlignment = HorizontalAlignment.Left;
                r.Margin = new Thickness(i * item.squareSize, j * item.squareSize, 0, 0);
                r.MouseLeftButtonDown += r_MouseLeftButtonDown;

                MainLayer.Children.Add(r);
                k++;
            }
```

Przypisanie domyślnych wartości w tablicy BoardColors

```
var proj2 = from p in db.NewProjects
    select p;
var globCol = from l in db.SquareFills
    select l;

foreach (var gv in globValue)
    foreach (var item in proj2)
```

```

        if (item.id_project == gv.actualProject)
        {
var board = from b in db.BoardColors
where b.id_project == gv.actualProject
select b;

foreach (var gc in globCol)
    foreach (var square in board)
    {
        square.rgb_blue = gc.defaultBlue;
        square.rgb_red = gc.defaultRed;
        square.rgb_green = gc.defaultGreen;
    }
}

```

Zapisanie wprowadzonych zmian w bazie i wywołanie metody LoadBase zmieniającej zawartość tablic z kolorami.

```

db.SaveChanges();
LoadBase();

}

```

Przycisk resetu

```

private void Button_Click_2(object sender, RoutedEventArgs e)
{
    AddSquare();
}

```

Kod odpowiedzialny za zmianę koloru po kliknięciu myszką na planszę

```

void r_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
    int k = 0;
    Rectangle rec = e.Source as Rectangle;

    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var globValue = from l in db.GlobalColors
                    select l;
    foreach (var gc in globValue)
    {
        Rectangle tb = e.Source as Rectangle;
        tb.Fill = new SolidColorBrush(Color.FromRgb(
            gc.choosenColorRed,
            gc.choosenColorGreen,
            gc.choosenColorBlue));

        k = int.Parse(rec.Name.Trim('s'));
        red_color[k] = gc.choosenColorRed;
        blue_color[k] = gc.choosenColorBlue;
        green_color[k] = gc.choosenColorGreen;
    }

    refreshColor();
}

```

Domyślny radioButton

Zmienia aktualny kolor na domyślny w tablicy GlobalColors

```
private void c10_Checked(object sender, RoutedEventArgs e)
{
    //domyślny rb
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var globValue = from l in db.GlobalValues
                    select l.actualProject;

    var fl = from f in db.SquareFills
             select f;
    var globColor = from c in db.GlobalColors
                    select c;
    foreach (var item in globValue)
        foreach (var colors in globColor)
            foreach (var color in fl)
            {
                if (item == color.id_project)
                {
                    colors.chosenColorRed = color.defaultRed;
                    colors.chosenColorGreen = color.defaultGreen;
                    colors.chosenColorBlue = color.defaultBlue;
                }
            }
    db.SaveChanges();
    refreshColor();
}
```

RadioButonny 1-5

Zmiana koloru w tablicy GlobalColors

```
private void c11_Checked(object sender, RoutedEventArgs e) => changeColor(1);
private void c12_Checked(object sender, RoutedEventArgs e) => changeColor(2);
private void c13_Checked(object sender, RoutedEventArgs e) => changeColor(3);
private void c14_Checked(object sender, RoutedEventArgs e) => changeColor(4);
private void c15_Checked(object sender, RoutedEventArgs e) => changeColor(5);

private void changeColor(int i)
{
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var globValue = from l in db.GlobalValues
                    select l.actualProject;
    var globColor = from c in db.GlobalColors
                    select c;
    var def = from d in db.DefaultColors
              where d.positionNumber == i
              select d;

    foreach (var item in globValue)
        foreach (var colors in globColor)
            foreach (var col in def)
            {
                if (item == col.id_project)
                {
                    colors.chosenColorBlue = col.rgb_blue;
                    colors.chosenColorRed = col.rgb_red;
                    colors.chosenColorGreen = col.rgb_green;
                }
            }
    db.SaveChanges();
    refreshColor();
}
```


Przycisk Wczytaj

Reset elementów na planszy i wprowadzenie nowych na podstawie wartości z tablicy BoardColors

```
private void Button_Click_3(object sender, RoutedEventArgs e)
{
    MainLayer.Children.Clear();
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var globValue = from l in db.GlobalValues
        select l;

    var proj = from p in db.NewProjects
        select p;
    byte red = 0; byte green = 0; byte blue = 0;
    int k = 0;
    int i = 0;
    int j = 0;
    foreach (var glob in globValue)
        foreach (var item in proj)
            if (item.id_project == glob.actualProject)
            {
                var fl = from f in db.BoardColors
                    where f.id_project == item.id_project
                    select f;

                foreach (var color in fl)
                {
                    red = color.rgb_red;
                    green = color.rgb_green;
                    blue = color.rgb_blue;

                    red_color[k] = red;
                    green_color[k] = green;
                    blue_color[k] = blue;

                    Rectangle r = new Rectangle
                    {
                        Height = item.squareSize,
                        Width = item.squareSize,
                        Fill = new SolidColorBrush(Color.FromRgb(red, green, blue)),
                        Name = "s" + k.ToString()
                    };

                    r.VerticalAlignment = VerticalAlignment.Top;
                    r.HorizontalAlignment = HorizontalAlignment.Left;
                    r.Margin = new Thickness(i * item.squareSize, j * item.squareSize, 0, 0);
                    r.MouseLeftButtonDown += r_MouseLeftButtonDown;

                    MainLayer.Children.Add(r);

                    k++;
                    if (j % ((item.boardSize / item.squareSize) - 1) == 0 & j != 0)
                    {
                        i++;
                        j = 0;
                    }
                    else j++;
                }
            }
}
```

Zapis wprowadzonych zmian w tablicach blue_color, red_color, green_color do tablicy BoardColors

```
private void Button_Click_4(object sender, RoutedEventArgs e)
{
    desc_prop = Main_desc.Text;

    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();

    int k = 0;

    var globVal = from g in db.GlobalValues
                  select g;
    var proj = from p in db.NewProjects
              select p;

    foreach (var gv in globVal)
        foreach (var item in proj)
            if (item.id_project == gv.actualProject)
            {
                var board = from b in db.BoardColors
                           where b.id_project == gv.actualProject
                           select b;

                foreach (var square in board)
                {
                    square.rgb_blue = blue_color[k];
                    square.rgb_red = red_color[k];
                    square.rgb_green = green_color[k];
                    k++;
                }
                if (item.description)
                {
                    if (desc_prop != null)
                        item.descNew = desc_prop;
                }
            }
    db.SaveChanges();
}
```

6. UstawieniaKoloru.xaml.cs

Inicjalizacja zawartości okna menu głównego i wycentrowanie widoku.

Aktualizacja wyświetlanego koloru w oknie edycji po zamknięciu okna Ustawień.

```
private OknoEdycji _ok;

public UstawieniaKoloru(OknoEdycji OK)
{
    WindowStartupLocation = WindowStartupLocation.CenterScreen;
    InitializeComponent();
    RefreshList();
    _ok = OK;
}
```

Odświeżenie listy z kolorami po otwarciu okna

```
private void RefreshList()
{
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    colorList.Items.Clear();
    var proj = from p in db.NewColors
               select p;
    foreach (var item in proj)
    {
        colorList.Items.Add(
            "RGB Color: " +
            item.rgb_red
            + " " +
            item.rgb_green
            + " " +
            item.rgb_blue
        );
    }
}
```

Przycik zamknij

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    Close();
}
```

Przycik Zmień

```
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    try
    {
        byte red = byte.Parse(redColor.Text);
        byte green = byte.Parse(greenColor.Text);
        byte blue = byte.Parse(blueColor.Text);
        actualColor.Fill = new SolidColorBrush(Color.FromRgb(
            red, green, blue));
    }
    catch (Exception)
    {
        MessageBox.Show("Błędna wartość!");
        return;
    }
}
```

Zapisanie nowego koloru w tablicy NewColor i dodanie go do listy

```
private void Button_Click_2(object sender, RoutedEventArgs e)
{
    try
    {
        byte red = byte.Parse(redColor.Text);
        byte green = byte.Parse(greenColor.Text);
        byte blue = byte.Parse(blueColor.Text);

        ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
        NewColor newColor = new NewColor()
        {
            rgb_blue = blue,
            rgb_green = green,
            rgb_red = red
        };
        db.NewColors.Add(newColor);
        db.SaveChanges();

        RefreshList();
    }
    catch (Exception)
    {
        MessageBox.Show("Błędna wartość!");
        return;
    }
}
```

Usunięcie wybranego koloru z listy i z tablicy NewColors

```
private void Button_Click_3(object sender, RoutedEventArgs e)
{
    if (colorList.SelectedItem != null)
    {
        ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
        var proj = from p in db.NewColors
                    select p;
        int idNumer = 0;
        foreach (var item in proj)
        {
            if (idNumer == colorList.SelectedIndex)
            {
                colorList.Items.Remove(item);
                db.NewColors.Remove(item);
            }
            idNumer++;
        }
        db.SaveChanges();
        RefreshList();
    }
    else MessageBox.Show("Nie wybrano elementu z listy!");
}
```

Przycisk wybierz z listy

```
private void Button_Click_4(object sender, RoutedEventArgs e)
{
    if (colorList.SelectedItem != null)
    {
        ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
        var globValue = from l in db.GlobalColors
                        select l;
        var proj = from p in db.NewColors
                  select p;
        int idNumer = 0;
        foreach (var item in proj)
        {
            foreach (var glob in globValue)
            {
                if (idNumer == colorList.SelectedIndex)
                {
```

Aktualizacja tablicy GlobalColors

```
                glob.chosenColorRed = item.rgb_red;
                glob.chosenColorGreen = item.rgb_green;
                glob.chosenColorBlue = item.rgb_blue;

                Brush bar = new SolidColorBrush(Color.FromRgb(
                    glob.chosenColorRed,
                    glob.chosenColorGreen,
                    glob.chosenColorBlue));
            }
        }
    }
}
```

Kolor kwadratu w oknie edycji po zamknięciu okna ustawień

```
        _ok.kwadracik = bar;
    }
    idNumer++;
}
int position = 0;

db.SaveChanges();
```

Sprawdzenie wciśniętego radioButtona w oknie edycji

```
    if (_ok.cl1.IsChecked == true) position = 1;
    if (_ok.cl2.IsChecked == true) position = 2;
    if (_ok.cl3.IsChecked == true) position = 3;
    if (_ok.cl4.IsChecked == true) position = 4;
    if (_ok.cl5.IsChecked == true) position = 5;

    changeColor(position);

    Close();
}
else MessageBox.Show("Nie wybrano koloru!");
}
```

Zmiana koloru w tablicy DefaultColors

```
private void changeColor(int i)
{
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var globValue = from l in db.GlobalValues
                    select l.actualProject;
    var globColor = from c in db.GlobalColors
                    select c;
    var def = from d in db.DefaultColors
              where d.positionNumber == i
              select d;
```

```

foreach (var item in globValue)
    foreach (var colors in globColor)
        foreach (var col in def)
        {
            if (item == col.id_project)
            {
                col.rgb_blue = colors.chosenColorBlue;
                col.rgb_red = colors.chosenColorRed;
                col.rgb_green = colors.chosenColorGreen;
            }
        }
    db.SaveChanges();
}

```

7. UstawieniaAnimacji.xaml.cs

Tablice dwuwymiarowe z wartościami kolorów wczytywanych elementów planszy

```

private byte[,] red_color = new byte[50, 10000];
private byte[,] green_color = new byte[50, 10000];
private byte[,] blue_color = new byte[50, 10000];

```

Wielkość planszy

```
private int _boardSize;
```

Listy projektów dodanych do listy z animacją

```

private List<int> _actualID400 = new List<int>();
private List<int> _actualID640 = new List<int>();
private List<int> _actualID800 = new List<int>();

```

numer elementów w listach

```

private int count_400;
private int count_640;
private int count_800;

```

Timer potrzebny do odmierzania czasu między animacjami

```
DispatcherTimer Timer = new DispatcherTimer();
```

Inicjalizacja zawartości okna menu głównego i wycentrowanie widoku.

```

public UstawieniaAnimacji()
{
    WindowStartupLocation = WindowStartupLocation.CenterScreen;
    InitializeComponent();
    Timer.Tick += new EventHandler(TimeClick);
}

```

Łaadowanie listy z projektami z tablicy NewProjects

```

private void LoadProjects()
{
    //pierwsza lista
    DostepneProj.Items.Clear();

    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var proj = from p in db.NewProjects
               where p.boardSize == _boardSize
               select p;
    foreach (var item in proj)
    {
        DostepneProj.Items.Add(item.projectName);
    }
}

```

Lista z projektami do Animacji

```
private void LoadAnimList()
{
```

Czyszczenie danych po zmianie wymiaru planszy

```
    AnimationList.Items.Clear();
    _actualID400.Clear();
    _actualID640.Clear();
    _actualID800.Clear();
    count_400 = 0;
    count_640 = 0;
    count_800 = 0;
```

Wczytywanie listy z tablicy w zależności od rozmiaru wybranej planszy

AnimationBoard400/ AnimationBoard640/ AnimationBoard800

```
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    if (_boardSize == 400)
    {
        var anim = from a in db.AnimationBoard400 select a;
        foreach (var a_item in anim)
        {
            var proj = from p in db.NewProjects
                        where p.id_project == a_item.id_project
                        select p;
            foreach (var item in proj)
            {
                AnimationList.Items.Add(item.projectName);
                _actualID400.Add(item.id_project);
            }
        }
    }
    else if (_boardSize == 640)
    {
        var anim = from a in db.AnimationBoard640 select a;
        foreach (var a_item in anim)
        {
            var proj = from p in db.NewProjects
                        where p.id_project == a_item.id_project
                        select p;
            foreach (var item in proj)
            {
                AnimationList.Items.Add(item.projectName);
                _actualID640.Add(item.id_project);
            }
        }
    }
    else if (_boardSize == 800)
    {
        var anim = from a in db.AnimationBoard800 select a;
        foreach (var a_item in anim)
        {
            var proj = from p in db.NewProjects
                        where p.id_project == a_item.id_project
                        select p;
            foreach (var item in proj)
            {
                AnimationList.Items.Add(item.projectName);
                _actualID800.Add(item.id_project);
            }
        }
    }
}
```

Przycisk zamknij

```
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    Close();
}
```

Aktualizacja wyświetlanych danych po zmianie wymiaru planszy

Zmiana wielkości okna

```
private void sizeBoard_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    MainAnimLayer.Children.Clear();
    if (sizeBoard.SelectedItem == sizeBoard400)
    {
        _boardSize = 400;
    }
    if (sizeBoard.SelectedItem == sizeBoard640)
    {
        _boardSize = 640;
    }
    if (sizeBoard.SelectedItem == sizeBoard800)
    {
        _boardSize = 800;
    }

    UstAnim.Height = 140 + _boardSize;
    UstAnim.Width = 160 + _boardSize;
    MainAnimLayer.Height = _boardSize;
    MainAnimLayer.Width = _boardSize;

    LoadProjects();
    LoadAnimList();
}
```


Przycisk Dodaj

Dodaje projekty do tablicy w zależności od rozmiaru wybranej planszy
AnimationBoard400/ AnimationBoard640/ AnimationBoard800

```
private void Button_Click_2(object sender, RoutedEventArgs e)
{
    AddProject();
}

private void AddProject()
{
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();

    var proj = from p in db.NewProjects
                where p.boardSize == _boardSize
                select p;
    int idNumer = 0;

    foreach (var item in proj)
    {
        if (idNumer == DostepneProj.SelectedIndex)
        {
            if (_boardSize == 400)
            {
                AnimationBoard400 newItem1 = new AnimationBoard400()
                {
                    id_project = item.id_project,
                    boardSize = _boardSize,
                };
                db.AnimationBoard400.Add(newItem1);
            }
            else if (_boardSize == 640)
            {
                AnimationBoard640 newItem2 = new AnimationBoard640()
                {
                    id_project = item.id_project,
                    boardSize = _boardSize,
                };
                db.AnimationBoard640.Add(newItem2);
            }
            else if (_boardSize == 800)
            {
                AnimationBoard800 newItem3 = new AnimationBoard800()
                {
                    id_project = item.id_project,
                    boardSize = _boardSize,
                };
                db.AnimationBoard800.Add(newItem3);
            }
            break;
        }
        idNumer++;
    }
    db.SaveChanges();
    AnimationList.Items.Add(DostepneProj.SelectedItem);
}
```

Przycisk Usuń

Usuwa projekty w tablicy w zależności od rozmiaru wybranej planszy
AnimationBoard400/ AnimationBoard640/ AnimationBoard800
Odświeża zawartość listy

```
private void Button_Click_3(object sender, RoutedEventArgs e)
{
    RemoveProject();
    LoadAnimList();
}

private void RemoveProject()
{
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();
    var proj = from p in db.NewProjects
               where p.boardSize == _boardSize
               select p;
    if (_boardSize == 400)
    {
        var anim = from p in db.AnimationBoard400
                   select p;
        int i = 0;
        foreach (var item in anim)
        {
            if (AnimationList.SelectedIndex == i)
            {
                AnimationList.Items.Remove(item);
                db.AnimationBoard400.Remove(item);
            }
            i++;
        }
    }
    else if (_boardSize == 640)
    {
        var anim = from p in db.AnimationBoard640
                   select p;
        int i = 0;
        foreach (var item in anim)
        {
            if (AnimationList.SelectedIndex == i)
            {
                AnimationList.Items.Remove(item);
                db.AnimationBoard640.Remove(item);
            }
            i++;
        }
    }
    else if (_boardSize == 800)
    {
        var anim = from p in db.AnimationBoard800
                   select p;
        int i = 0;
        foreach (var item in anim)
        {
            if (AnimationList.SelectedIndex == i)
            {
                AnimationList.Items.Remove(item);
                db.AnimationBoard800.Remove(item);
            }
            i++;
        }
    }
    db.SaveChanges();
}
```

Przycisk Startu

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    StartAnimation();
}

private void StartAnimation()
{
```

Sprawdzenie poprawnej wartości czasu przejścia animacji

```
int _interval = 1;
try
{
    _interval = int.Parse(interval.Text);
    if (_interval < 0 | _interval > 10) throw new Exception();
}
catch (Exception)
{
    MessageBox.Show("Czas przejścia musi być liczbą całkowitą z przedziału 1-10s");
}

count_400 = 0;
count_640 = 0;
count_800 = 0;
```

Zapis danych do tablic

blue_color, red_color, green_color

```
if (_boardSize == 400)
{
    for (int i = 0; i < _actualID400.Count; i++)
    {
        SaveColors(_actualID400[count_400], count_400);
        count_400++;
    }
    count_400 = 0;
}
else if (_boardSize == 640)
{
    for (int i = 0; i < _actualID640.Count; i++)
    {
        SaveColors(_actualID640[count_640], count_640);
        count_640++;
    }
    count_640 = 0;
}
else if (_boardSize == 800)
{
    for (int i = 0; i < _actualID800.Count; i++)
    {
        SaveColors(_actualID800[count_800], count_800);
        count_800++;
    }
    count_800 = 0;
}
Timer.Interval = new TimeSpan(0, 0, _interval);
Timer.Start();
}
```

Zapis danych do tablic

blue_color, red_color, green_color

```
private void SaveColors(int ID, int count_x)
{
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();

    int k = 0;
    var proj = from p in db.NewProjects
               select p;

    foreach (var item in proj)
        if (item.id_project == ID)
        {
            var board = from b in db.BoardColors
                        where b.id_project == ID
                        select b;

            foreach (var square in board)
            {
                blue_color[count_x, k] = square.rgb_blue;
                red_color[count_x, k] = square.rgb_red;
                green_color[count_x, k] = square.rgb_green;
                k++;
            }
        }
}
```

Tworzenie nowej planszy po upływie ustawionego czasu

```
private void TimeClick(object sender, EventArgs e)
{
    if(_boardSize == 400)
    {
        if (count_400 < _actualID400.Count)
        {
            Load(_actualID400[count_400], count_400);
            count_400++;
        }
        else Timer.Stop();
    }
    else if (_boardSize == 640)
    {
        if (count_640 < _actualID640.Count)
        {
            Load(_actualID640[count_640], count_640);
            count_640++;
        }
        else Timer.Stop();
    }
    else if (_boardSize == 800)
    {
        if (count_800 < _actualID800.Count)
        {
            Load(_actualID800[count_800], count_800);
            count_800++;
        }
        else Timer.Stop();
    }
}
```

Tworzenie planszy z wartościami zapisanymi w tablicach

blue_color, red_color, green_color

```
private void Load(int ID, int index)
{
    MainAnimLayer.Children.Clear();
    ProjektSemestralnyDBEntities db = new ProjektSemestralnyDBEntities();

    var proj = from p in db.NewProjects
               select p;
    byte red = 0; byte green = 0; byte blue = 0;
    int k = 0;
    int i = 0;
    int j = 0;
    foreach (var item in proj)
        if (item.id_project == ID)
        {
            var fl = from f in db.BoardColors
                     where f.id_project == item.id_project
                     select f;

            foreach (var color in fl)
            {
                red = red_color[index, k];
                green = green_color[index,k];
                blue = blue_color[index,k];

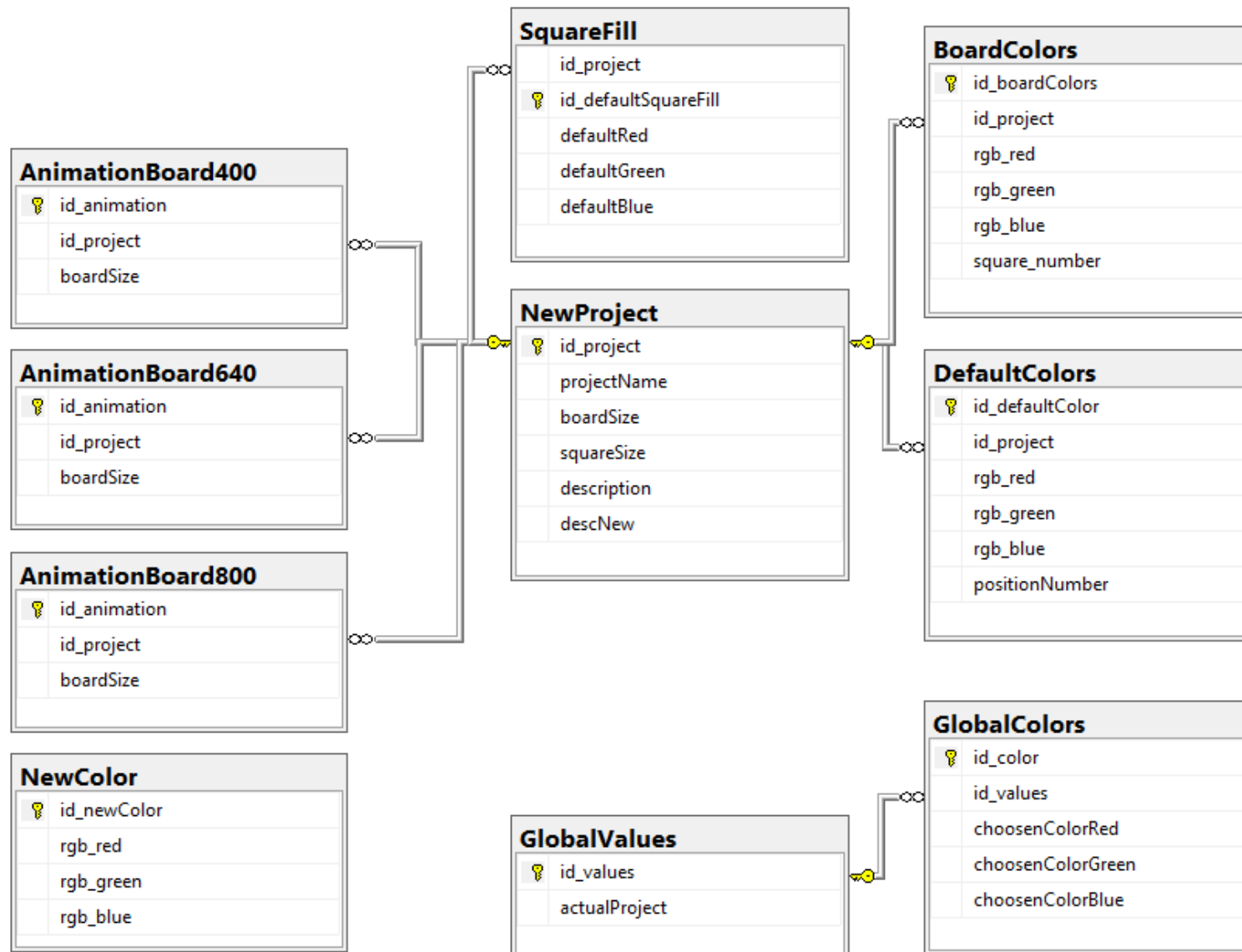
                Rectangle r = new Rectangle
                {
                    Height = item.squareSize,
                    Width = item.squareSize,
                    Fill = new SolidColorBrush(Color.FromRgb(red, green, blue)),
                };

                r.VerticalAlignment = VerticalAlignment.Top;
                r.HorizontalAlignment = HorizontalAlignment.Left;
                r.Margin = new Thickness(i * item.squareSize, j * item.squareSize, 0, 0);
                MainAnimLayer.Children.Add(r);

                k++;
                if (j % ((item.boardSize / item.squareSize) - 1) == 0 & j != 0)
                {
                    i++;
                    j = 0;
                }
                else j++;
            }
        }
}
```

Opis bazy danych

Diagram



Opis tabel

1.Tabela NewProject

| Kolumna | Opis | Typ | Pusta wartość | Standardowa wartość | Powiązania |
|-------------|-------------------------------|---------------|---------------|----------------------------------|-------------|
| Id_project | Identyfikator numeru projektu | Int | NOT NULL | Automatycznie zwiększany licznik | Primary key |
| projectName | Nazwa projektu | Varchar(20) | NOT NULL | - | - |
| boardSize | Wielkość planszy | Int | NOT NULL | - | - |
| squareSize | Wielkość elementów planszy | Int | NOT NULL | - | - |
| description | Opis tak/nie | Bool | NOT NULL | - | - |
| descNew | Zawartość opisu projektu | Nvarchar(200) | NOT NULL | - | - |

2.Tabela SquareFill

| Kolumna | Opis | Typ | Pusta wartość | Standardowa wartość | Powiązania |
|----------------------|---|---------|---------------|---------------------------------------|-----------------------------------|
| Id_project | Identyfikator numeru projektu | Int | NOT NULL | Wartość powiązana z tabelą NewProject | Key_ SquareFill_ id_project |
| Id_defaultSquareFill | Identyfikator numeru wypełnienia planszy | Int | NOT NULL | Automatycznie zwiększany licznik | Primary key |
| defaultRed | Domyślna wartość koloru elementów planszy | Tinyint | NOT NULL | 0 | - |
| defaultGreen | Domyślna wartość koloru elementów planszy | Tinyint | NOT NULL | 0 | - |
| defaultBlue | Domyślna wartość koloru elementów planszy | Tinyint | NOT NULL | 0 | - |

3.Tabela DefaultColors

| Kolumna | Opis | Typ | Pusta wartość | Standardowa wartość | Powiązania |
|------------------|-------------------------------------|---------|---------------|---------------------------------------|--------------------------------------|
| Id_project | Identyfikator numeru projektu | Int | NOT NULL | Wartość powiązana z tabelą NewProject | Key_ DefaultColors_ id_project |
| Id_defaultColors | Identyfikator numeru kolorów palety | Int | NOT NULL | Automatycznie zwiększany licznik | Primary key |
| rgb_Red | Wartość kolorów w paletce | Tinyint | NOT NULL | 255 | - |
| rgb_Green | Wartość kolorów w paletce | Tinyint | NOT NULL | 255 | - |
| rgb_Blue | Wartość kolorów w paletce | Tinyint | NOT NULL | 255 | - |
| positionNumber | Numer koloru na paletce 1-6 | Tinyint | NOT NULL | - | - |

4.Tabela GlobalValues

| Kolumna | Opis | Typ | Pusta wartość | Standardowa wartość | Powiązania |
|---------------|--------------------------------|-----|---------------|---------------------|-------------|
| Id_values | Identyfikator numeru projektu | Int | NOT NULL | 0 | Primary key |
| actualProject | Aktualnie edytowany id_project | Int | NOT NULL | - | - |

5.Tabela GlobalColors

| Kolumna | Opis | Typ | Pusta wartość | Standardowa wartość | Powiązania |
|----------------|--|---------|---------------|---|------------------------------------|
| Id_color | Identyfikator numeru zmienianych kolorów | Int | NOT NULL | Automatycznie zwiększany licznik | Primary key |
| Id_values | Identyfikator numeru projektu | Int | NOT NULL | Wartość powiązana z tabelą GlobalValues | Key_ GlobalColors_ id_values |
| chosenColorRed | Aktualna wartość używanego koloru | Tinyint | NOT NULL | 255 | - |
| chosenColorRed | Aktualna wartość używanego koloru | Tinyint | NOT NULL | 255 | - |
| chosenColorRed | Aktualna wartość używanego koloru | Tinyint | NOT NULL | 255 | - |

6.Tabela NewColor

| Kolumna | Opis | Typ | Pusta wartość | Standardowa wartość | Powiązania |
|-------------|---------------------------------------|---------|---------------|----------------------------------|-------------|
| Id_newColor | Identyfikator numeru listy z kolorami | Int | NOT NULL | Automatycznie zwiększany licznik | Primary key |
| rgb_red | wartość koloru w liście | Tinyint | NOT NULL | 255 | - |
| rgb_green | wartość koloru w liście | Tinyint | NOT NULL | 255 | - |
| rgb_blue | wartość koloru w liście | Tinyint | NOT NULL | 255 | - |

7.Tabela BoardColors

| Kolumna | Opis | Typ | Pusta wartość | Standardowa wartość | Powiązania |
|----------------|---|---------|---------------|---------------------------------------|----------------------------|
| Id_boardColors | Identyfikator numeru koloru elementów planszy | Int | NOT NULL | Automatycznie zwiększany licznik | Primary key |
| Id_project | Identyfikator numeru projektu | Int | NOT NULL | Wartość powiązana z tabelą NewProject | Key_BoardColors_id_project |
| rgb_red | wartość koloru w liście | Tinyint | NOT NULL | 255 | - |
| rgb_green | wartość koloru w liście | Tinyint | NOT NULL | 255 | - |
| rgb_blue | wartość koloru w liście | Tinyint | NOT NULL | 255 | - |
| square_number | Numer elementu na planszy | Int | NOT NULL | - | - |

8.Tabela AnimationBoard400

| Kolumna | Opis | Typ | Pusta wartość | Standardowa wartość | Powiązania |
|--------------|-------------------------------|-----|---------------|---------------------------------------|----------------------------------|
| Id_project | Identyfikator numeru projektu | Int | NOT NULL | Wartość powiązana z tabelą NewProject | Key_AnimationBoard400_id_project |
| Id_animation | Identyfikator numeru animacji | Int | NOT NULL | Automatycznie zwiększany licznik | Primary key |
| boardSize | Rozmiar planszy | Int | NOT NULL | - | - |

9.Tabela AnimationBoard640

| Kolumna | Opis | Typ | Pusta wartość | Standardowa wartość | Powiązania |
|--------------|-------------------------------|-----|---------------|---------------------------------------|----------------------------------|
| Id_project | Identyfikator numeru projektu | Int | NOT NULL | Wartość powiązana z tabelą NewProject | Key_AnimationBoard640_id_project |
| Id_animation | Identyfikator numeru animacji | Int | NOT NULL | Automatycznie zwiększany licznik | Primary key |
| boardSize | Rozmiar planszy | Int | NOT NULL | - | - |

10.Tabela AnimationBoard800

| Kolumna | Opis | Typ | Pusta wartość | Standardowa wartość | Powiązania |
|--------------|-------------------------------|-----|---------------|---------------------------------------|----------------------------------|
| Id_project | Identyfikator numeru projektu | Int | NOT NULL | Wartość powiązana z tabelą NewProject | Key_AnimationBoard800_id_project |
| Id_animation | Identyfikator numeru animacji | Int | NOT NULL | Automatycznie zwiększany licznik | Primary key |
| boardSize | Rozmiar planszy | Int | NOT NULL | - | - |

Kod bazy danych

```
--use master
GO
--ALTER DATABASE ProjektSemestralnyDB SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO
--DROP DATABASE ProjektSemestralnyDB
GO

use master
GO
CREATE DATABASE ProjektSemestralnyDB
GO
USE ProjektSemestralnyDB
GO

CREATE TABLE NewProject
(
id_project int identity(1,1) primary key,
projectName varchar(20) NOT NULL,
boardSize int NOT NULL,
squareSize int NOT NULL,
description bit NOT NULL,
descNew nvarchar(200) NOT NULL
);
GO

CREATE TABLE SquareFill
(
id_project int NOT NULL,
id_defaultSquareFill int identity(1,1) primary key,
defaultRed tinyint NOT NULL DEFAULT 0,
defaultGreen tinyint NOT NULL DEFAULT 0,
defaultBlue tinyint NOT NULL DEFAULT 0,
CONSTRAINT Key_SquareFill_id_project FOREIGN KEY(id_project) REFERENCES NewProject
);
GO
CREATE TABLE DefaultColors
(
id_defaultColor int identity(1,1) primary key,
id_project int NOT NULL,
rgb_red tinyint NOT NULL DEFAULT 255,
rgb_green tinyint NOT NULL DEFAULT 255,
rgb_blue tinyint NOT NULL DEFAULT 255,
positionNumber tinyint NOT NULL
CONSTRAINT Key_DefaultColors_id_project FOREIGN KEY(id_project) REFERENCES NewProject
);
GO

CREATE TABLE GlobalValues
(
id_values int primary key,
actualProject int NOT NULL
);
GO
CREATE TABLE GlobalColors
(
id_color int identity(1,1) primary key,
id_values int,
choosenColorRed tinyint NOT NULL DEFAULT 255,
choosenColorGreen tinyint NOT NULL DEFAULT 255,
choosenColorBlue tinyint NOT NULL DEFAULT 255,
CONSTRAINT Key_GlobalColors_id_values FOREIGN KEY(id_values) REFERENCES GlobalValues
);
GO

CREATE TABLE NewColor
(
```

```
id_newColor int identity(1,1) primary key,  
rgb_red tinyint NOT NULL DEFAULT 255,  
rgb_green tinyint NOT NULL DEFAULT 255,  
rgb_blue tinyint NOT NULL DEFAULT 255  
);  
GO
```

```
CREATE TABLE BoardColors  
(  
id_boardColors int identity(1,1) primary key,  
id_project int NOT NULL,  
rgb_red tinyint NOT NULL DEFAULT 255,  
rgb_green tinyint NOT NULL DEFAULT 255,  
rgb_blue tinyint NOT NULL DEFAULT 255,  
square_number int NOT NULL  
CONSTRAINT Key_BoardColors_id_project FOREIGN KEY(id_project) REFERENCES NewProject  
);  
  
GO
```

```
CREATE TABLE AnimationBoard400  
(  
id_animation int identity(0,1) primary key,  
id_project int NOT NULL,  
boardSize int NOT NULL,  
CONSTRAINT Key_AnimationBoard400_id_project FOREIGN KEY(id_project) REFERENCES NewProject  
);
```

```
CREATE TABLE AnimationBoard640  
(  
id_animation int identity(0,1) primary key,  
id_project int NOT NULL,  
boardSize int NOT NULL,  
CONSTRAINT Key_AnimationBoard640_id_project FOREIGN KEY(id_project) REFERENCES NewProject  
);
```

```
CREATE TABLE AnimationBoard800  
(  
id_animation int identity(0,1) primary key,  
id_project int NOT NULL,  
boardSize int NOT NULL,  
CONSTRAINT Key_AnimationBoard800_id_project FOREIGN KEY(id_project) REFERENCES NewProject  
);
```