Dokumentacja techniczna

Puzzle Game

**Opracowanie:**

Bartosz Sowa

Krzysztof Sobiech

Jakub Wietecha

Spis treści

[1 Opis aplikacji 3](#_Toc473055300)

[2 Techniki programowania 3](#_Toc473055301)

[3 Instrukcja 3](#_Toc473055302)

[4 Specyfikacja interfejsów w języku IDL 4](#_Toc473055303)

[5 Diagramy UML 6](#_Toc473055304)

[5.1 Diagram Klas 6](#_Toc473055305)

[5.2 Diagram Komponentów 6](#_Toc473055306)

[5.3 Diagram Aktywności 7](#_Toc473055307)

[5.4 Diagram Przypadków Użycia 7](#_Toc473055308)

[6 Harmonogram 8](#_Toc473055309)

# Opis aplikacji

Aplikacja *Puzzle Game* została stworzona w ramach projektu studenckiego. Służy do ratowania ludzi w przypadku nadmiaru czasu wolnego.

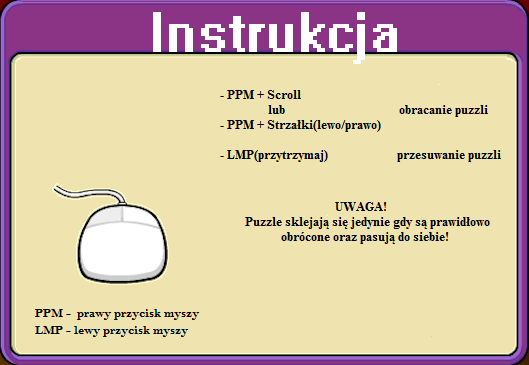
W aplikacji mamy możliwość wybrania dowolnego zdjęcia, a następnie stworzenia z niego puzzli. Głównym celem użytkownika jest ułożenie puzzli w taki sposób aby otrzymać identyczny obrazek, jaki wybrał na początku.

Dostęp do aplikacji możliwy jest z poziomu systemu użytkownika po uprzednim jej ściągnięciu. Aplikacja nie wymaga instalacji. Do poprawnego działania aplikacji wymagane jest oprogramowanie Windows 7 lub wyższe oraz .NET Framework 4.5.

# Techniki programowania

Aplikacja została opracowana przy użyciu Windows Presentation Foundation w środowisku programistycznym „*Visual Studio 2015*”. WPF służy do tworzenia nowoczesnych aplikacji pulpitu-biznesowych w systemie Windows. Kod źródłowy został stworzony w języku *C#*.

# Instrukcja



# Specyfikacja interfejsów w języku IDL

module Puzzle

{

public struct Coordinate

{

attribute int X;

attribute int Y;

};

public interface IPuzzlePiece

{

attribute int ID;

attribute int ClusterId;

attribute Coordinate Location;

attribute double Width;

attribute double Height;

attribute int Rotation;

attribute Image PieceImage;

attribute CroppedBitmap PieceImage;

};

public interface IPuzzleCluster

{

attribute int ID;

attribute sequence <IPuzzlePiece> Pieces;

};

public interface IPuzzleEngine

{

sequence <CroppedBitmap> CutImagesToPieces(in BitmapImage photo, in int puzzleSize);

bool DetermineIfMergePieces(in IPuzzlePiece current, in IPuzzlePiece adjacent);

sequence <int> DetermineAdjacentPieceIDs(in sequence <Coordinate> coordinates, in int numberOfColumns, in int numberOfRows);

void AlignPiecesPositions(in IPuzzleCluster cluster);

void DeleteShadowEffect(in IPuzzleCluster cluster, inout int zindex);

void DropShadowEffect(in IPuzzleCluster cluster);

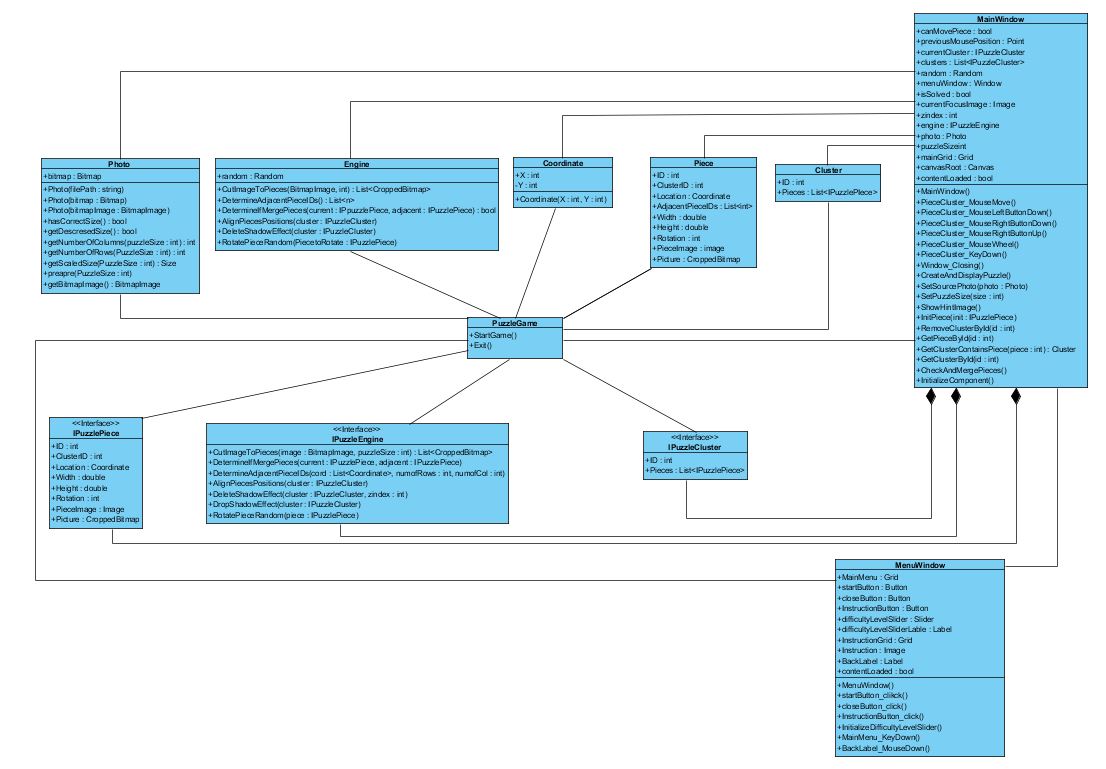
void RotatePieceRandom(in IPuzzlePiece piece);

};

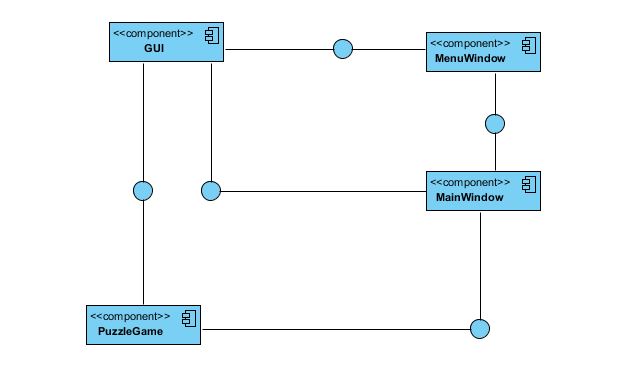
}

# Diagramy UML

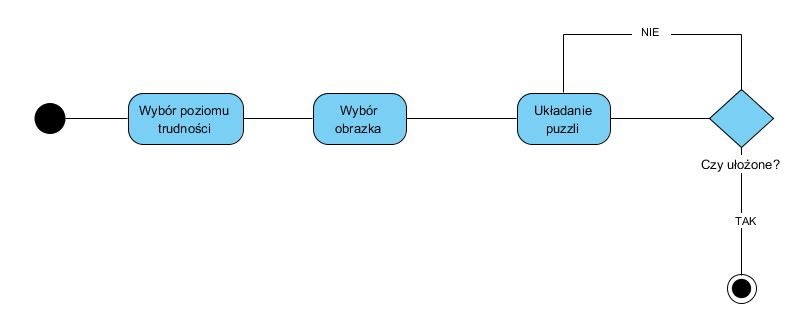
## Diagram Klas



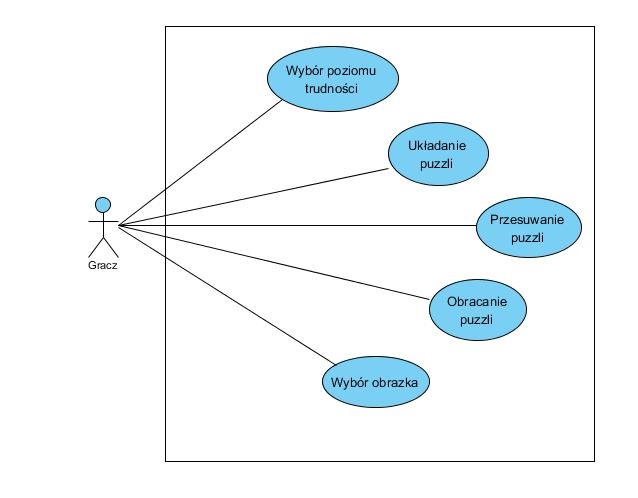
## Diagram Komponentów



## Diagram Aktywności



## Diagram Przypadków Użycia



# Harmonogram

**20.11.2016** - Model reprezentujący kawałek oraz grupę puzzli - ***Bartosz Sowa***

**20.11.2016** - Metoda dzieląca obrazek na kawałki/puzzle - ***Jakub Wietecha***

**27.11.2016** - Diagramy UML (Use Case/Activity/Component) - ***Bartosz Sowa/ Krzysztof Sobiech/ Jakub Wietecha***  
**27.11.2016** - Eventy umożliwiające przesuwanie puzzli - ***Krzysztof Sobiech***

**04.12.2016** - Wczytywanie oraz walidacja wczytanego zdjęcia - ***Jakub Wietecha***

**04.12.2016** - Możliwość łączenia kawałków puzzli - ***Bartosz Sowa***

**11.12.2016** - Główna metoda tworząca puzzle z wczytanego obrazka (wyświetlenie i umożliwienie interakcji z użytkownikiem) - ***Bartosz Sowa***

**18.12.2016** - Menu startowe aplikacji - ***Krzysztof Sobiech***

**20.12.2016** - Możliwość obracania pojedynczego kawałka puzzla - ***Bartosz Sowa***

**20.12.2016** - Wyrównywanie puzzli po ich połączeniu - ***Jakub Wietecha***

**15.01.2017** - Końcowe poprawki (refaktoryzacja kodu) - ***Bartosz Sowa/ Krzysztof Sobiech/   
Jakub Wietecha***