Орієнтований граф

Створено системою Doxygen 1.12.0

Розділ 1

Алфавітний покажчик простору імен

1.1 Package List

Повний список документованих пакетів.

Розділ 2

Ієрархічний покажчик класів

2.1 Ієрархія класів

Список успадкувань впорядковано наближено до алфавіту

Розділ 3

Алфавітний покажчик класів

3.1 Класи

Класи, структури, об'єднання та інтерфейси з коротким описом.

Розділ 4

Покажчик файлів

4.1 Файли

Повний список файлів.

Розділ 5

Опис простору імен

5.1 Простір імен GraphApp

Класи

- class App

  Interaction logic for App.xaml.
- class Arrowhead

  Представляє стрілку на кінці ребра.
- class Edge

  Представляє ребро між двома вершинами.
- class GraphElement

  Абстрактний клас, що представляє графічний елемент на полотні (Canvas).
- class MainWindow

  Головне вікно програми для побудови та маніпуляції графами.
- class Vertex

  Представляє вершину в графі, яка є графічним елементом на Canvas.

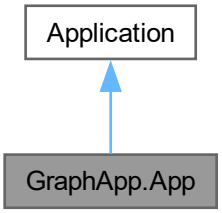
Розділ 6

Класи

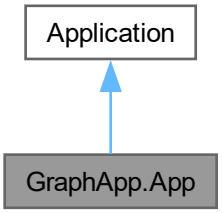
6.1 Клас GraphApp.App

Interaction logic for App.xaml.

Схема успадкувань для GraphApp.App



Діаграма зв'язків класу GraphApp.App:



6.1.1 Детальний опис

Interaction logic for App.xaml.

Див. визначення в файлі App.xaml.cs, рядок 14

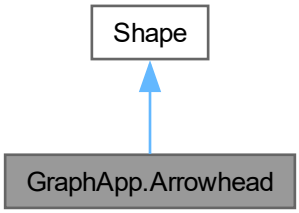
Документація цього класу була створена з файлу:

- GraphApp-main/GraphApp/App.xaml.cs

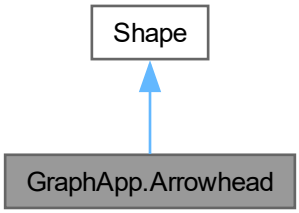
6.2 Клас GraphApp.Arrowhead

Представляє стрілку на кінці ребра.

Схема успадкувань для GraphApp.Arrowhead



Діаграма зв'язків класу GraphApp.Arrowhead:



Захищені елементи

- override void OnRender (DrawingContext drawingContext)

Властивості

- override Geometry DefiningGeometry [get]

  Визначає геометрію для малювання стрілки.

6.2.1 Детальний опис

Представляє стрілку на кінці ребра.

Клас Arrowhead є допоміжним графічним елементом для відображення стрілки на кінці ребра. Стрілка вказує напрямок між вершинами.

Див. визначення в файлі MainWindow.xaml.cs, рядок 645

6.2.2 Опис методів компонент

6.2.2.1 OnRender()

override void GraphApp.Arrowhead.OnRender (
DrawingContext drawingContext) [protected]

Див. визначення в файлі MainWindow.xaml.cs, рядок 667

6.2.3 Повний список властивостей

6.2.3.1 DefiningGeometry

override Geometry GraphApp.Arrowhead.DefiningGeometry [get], [protected]

Визначає геометрію для малювання стрілки.

Повертає

Geometry об'єкт, що представляє форму стрілки.

Див. визначення в файлі MainWindow.xaml.cs, рядок 651

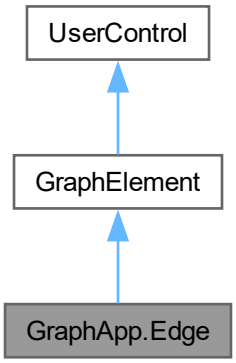
Документація цього класу була створена з файлу:

- GraphApp-main/GraphApp/MainWindow.xaml.cs

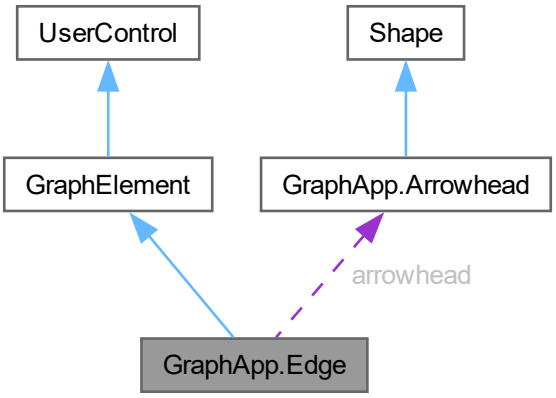
6.3 Клас GraphApp.Edge

Представляє ребро між двома вершинами.

Схема успадкувань для GraphApp.Edge



Діаграма зв'язків класу GraphApp.Edge:



Загальнодоступні елементи

- Edge (Vertex startVertex, Vertex endVertex)

  Конструктор для створення ребра між двома вершинами.
- override void UpdatePosition ()

  Оновлює положення лінії та стрілки між вершинами.

Загальнодоступні елементи успадковано з GraphApp.GraphElement

- void UpdatePosition ()
  Оновлює положення елемента на полотні.

Властивості

- Vertex StartVertex [get]
  Вершина, з якої починається ребро.
- Vertex EndVertex [get]
  Вершина, на якій закінчується ребро.

Приватні елементи

- void Vertex_SizeChanged (object sender, SizeChangedEventArgs e)
  Обробник події зміни розміру вершини, оновлює позиції ребер.

Приватні дані

- Line line
  Лінія, що представляє ребро.
- Arrowhead arrowhead
  Стрілка на кінці ребра.

6.3.1 Детальний опис

Представляє ребро між двома вершинами.

Клас Edge є графічним елементом, що представляє ребро (лінію) між двома вершинами на Canvas. Він дозволяє відображати лінії та стрілки між вершинами і підтримує оновлення положення при зміні положення вершин.

Див. визначення в файлі MainWindow.xaml.cs, рядок 534

6.3.2 Конструктор(и)

6.3.2.1 Edge()

```
GraphApp.Edge.Edge (
            Vertex startVertex,
            Vertex endVertex)
```

Конструктор для створення ребра між двома вершинами.

Аргументи

startVertex	Вершина, з якої починається ребро.
endVertex	Вершина, на якій закінчується ребро.

Див. визначення в файлі MainWindow.xaml.cs, рядок 561

6.3.3 Опис методів компонент

6.3.3.1 UpdatePosition()

override void GraphApp.Edge.UpdatePosition ()

Оновлює положення лінії та стрілки між вершинами.

Рахує координати лінії і стрілки між двома вершинами та оновлює положення відповідних елементів.

Див. визначення в файлі MainWindow.xaml.cs, рядок 590

6.3.3.2 Vertex_SizeChanged()

void GraphApp.Edge.Vertex_SizeChanged (
object sender,
SizeChangedEventArgs e) [private]

Обробник події зміни розміру вершини, оновлює позиції ребер.

Аргументи

sender	Об'єкт, що викликає подію.
e	Аргументи події.

Див. визначення в файлі MainWindow.xaml.cs, рядок 632

6.3.4 Компонентні дані

6.3.4.1 arrowhead

Arrowhead GraphApp.Edge.arrowhead [private]

Стрілка на кінці ребра.

Див. визначення в файлі MainWindow.xaml.cs, рядок 544

6.3.4.2 line

Line GraphApp.Edge.line [private]

Лінія, що представляє ребро.

Див. визначення в файлі MainWindow.xaml.cs, рядок 539

6.3.5 Повний список властивостей

6.3.5.1 EndVertex

Vertex GraphApp.Edge.EndVertex [get]

Вершина, на якій закінчується ребро.

Див. визначення в файлі MainWindow.xaml.cs, рядок 554

6.3.5.2 StartVertex

Vertex GraphApp.Edge.StartVertex [get]

Вершина, з якої починається ребро.

Див. визначення в файлі MainWindow.xaml.cs, рядок 549

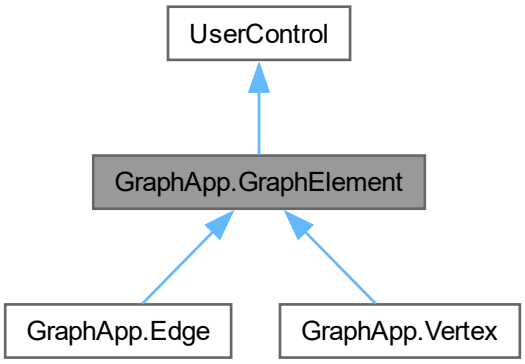
Документація цього класу була створена з файлу:

- GraphApp-main/GraphApp/MainWindow.xaml.cs

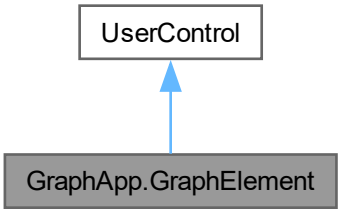
6.4 Клас GraphApp.GraphElement

Абстрактний клас, що представляє графічний елемент на полотні (Canvas).

Схема успадкувань для GraphApp.GraphElement



Діаграма зв'язків класу GraphApp.GraphElement:



Загальнодоступні елементи

- void UpdatePosition ()
  Оновлює положення елемента на полотні.

6.4.1 Детальний опис

Абстрактний клас, що представляє графічний елемент на полотні (Canvas).

Цей клас наслідує від UserControl і служить базовим класом для всіх елементів графа, таких як вершини і ребра. Дочірні класи повинні реалізувати метод UpdatePosition, який оновлює положення елемента на полотні.

Див. визначення в файлі MainWindow.xaml.cs, рядок 413

6.4.2 Опис методів компонент

6.4.2.1 UpdatePosition()

void GraphApp.GraphElement.UpdatePosition () [abstract]

Оновлює положення елемента на полотні.

Цей метод повинен бути реалізований у дочірніх класах, щоб змінювати координати графічного елемента, відповідно до його нової позиції.

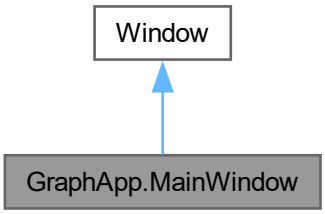
Документація цього класу була створена з файлу:

- GraphApp-main/GraphApp/MainWindow.xaml.cs

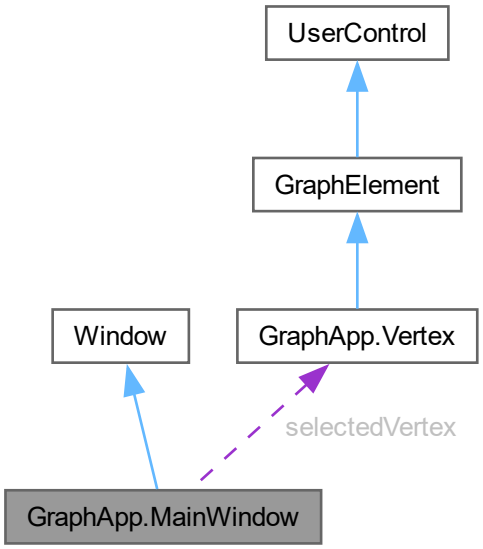
6.5 Клас GraphApp.MainWindow

Головне вікно програми для побудови та маніпуляції графами.

Схема успадкувань для GraphApp.MainWindow



Діаграма зв'язків класу GraphApp.MainWindow:



Загальнодоступні елементи

- MainWindow ()

  Конструктор головного вікна.
- void UpdateEdgePositions (Vertex vertex)

  Оновлення позицій ребер, які з'єднані з заданою вершиною

Приватні елементи

- void AddVertexButton_Click (object sender, RoutedEventArgs e)
- void CreateVertex ()

  Створює нову вершину на графічному полотні.
- void Vertex_MouseDown (object sender, MouseButtonEventArgs e)

  Обробник події натискання мишею на вершину. Або починає перетягування, або створює ребро.
- void Vertex_MouseMove (object sender, MouseEventArgs e)

  Обробник події руху миші над вершиною. Оновлює позицію вершини під час перетягування.
- void Vertex_MouseUp (object sender, MouseButtonEventArgs e)

  Обробник події відпускання кнопки миші після перетягування вершини
- void Edge_MouseDown (object sender, MouseButtonEventArgs e)

  Клас, що містить методи для роботи з ребрами та їх створення
- void CreateEdge (Vertex startVertex, Vertex endVertex)

  Створення ребра між двома вершинами
- void StartEdgeCreation ()

  Початок створення ребра (перша вершина вибрана)
- void StopEdgeCreation ()

  Зупинка створення ребра
- Vertex GetVertexUnderMouse (Point position)

  Отримання положення вершини
- bool IsMouseOverVertex (Point position, Vertex vertex)

  Перевірка, чи знаходиться курсор миші над заданою вершиною
- void RemoveVertexButton_Click (object sender, RoutedEventArgs e)

  Обробка події натиснення кнопки видалення вершини
- void RemoveSelectedVertex ()

  Видалення вибраної вершини
- void RemoveEdge (Edge edge)

  Видалення ребра
- void RemoveEdgesConnectedToVertex (Vertex vertex)

  Видалення всіх ребер, що з'єднані з заданою вершиною
- void ToggleEdgeCreationButton_Checked (object sender, RoutedEventArgs e)

  Обробка події включення кнопки створення ребра (активація кнопки Create Edge)
- void ToggleEdgeCreationButton_Unchecked (object sender, RoutedEventArgs e)

  Обробка події вимкнення кнопки створення ребра (деактивація кнопки Create Edge)
- void AdjacencyMatrix ()

  Метод створення матриці суміжності
- void Window_SizeChanged (object sender, SizeChangedEventArgs e)

  перевірка зміни розміра вікна

Приватні дані

- List< Vertex > vertices
- List< Edge > edges
- bool isDragging
- bool isCreatingEdge
- Vertex selectedVertex
- Line edgeLine

6.5.1 Детальний опис

Головне вікно програми для побудови та маніпуляції графами.

Див. визначення в файлі MainWindow.xaml.cs, рядок 29

6.5.2 Конструктор(и)

6.5.2.1 MainWindow()

GraphApp.MainWindow.MainWindow ()

Конструктор головного вікна.

Ініціалізує компоненти, змінні та налаштування для управління графом. Після виклику цього конструктора вікно готове до взаємодії з користувачем.

- Ініціалізуються змінні:
  - vertices: новий список для зберігання вершин.
  - edges: новий список для зберігання ребер.
  - isDragging: встановлюється в false, щоб вказати, що в даний момент не відбувається перетягування.
  - isCreatingEdge: встановлюється в false, щоб вказати, що не створюється нове ребро.
  - selectedVertex: встановлюється в null, оскільки жодна вершина не вибрана.
  - edgeLine: також встановлюється в null, оскільки немає створюваного ребра.

Після ініціалізації, вміст мітки (label) оновлюється, щоб відобразити інформацію про вибрану вершину.

Див. визначення в файлі MainWindow.xaml.cs, рядок 61

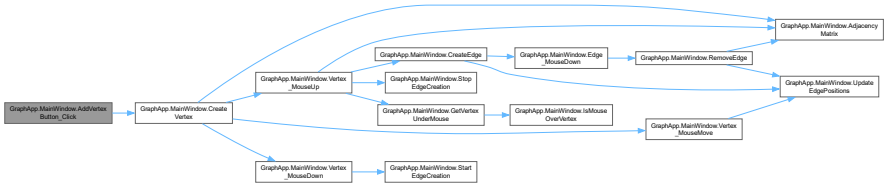
6.5.3 Опис методів компонент

6.5.3.1 AddVertexButton_Click()

void GraphApp.MainWindow.AddVertexButton_Click (
object sender,
RoutedEventArgs e) [private]

Див. визначення в файлі MainWindow.xaml.cs, рядок 75

Граф всіх викликів цієї функції:



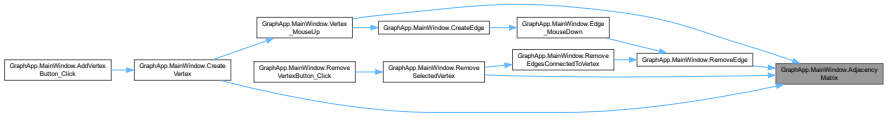
6.5.3.2 AdjacencyMatrix()

void GraphApp.MainWindow.AdjacencyMatrix () [private]

Метод створення матриці суміжності

Див. визначення в файлі MainWindow.xaml.cs, рядок 354

Граф викликів для цієї функції:



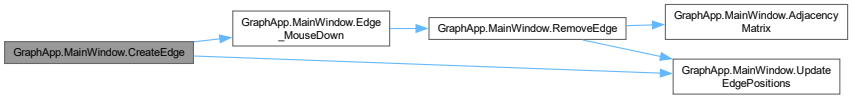
6.5.3.3 CreateEdge()

void GraphApp.MainWindow.CreateEdge (
Vertex startVertex,
Vertex endVertex) [private]

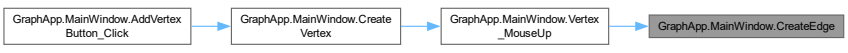
Створення ребра між двома вершинами

Див. визначення в файлі MainWindow.xaml.cs, рядок 193

Граф всіх викликів цієї функції:



Граф викликів для цієї функції:



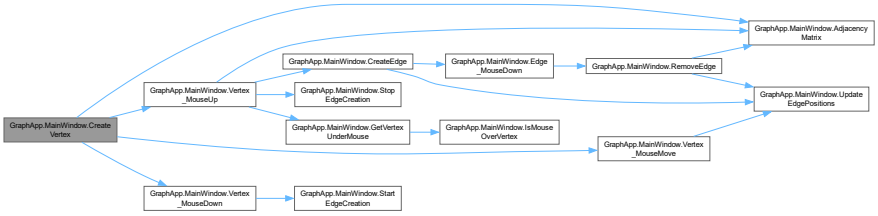
6.5.3.4 CreateVertex()

void GraphApp.MainWindow.CreateVertex () [private]

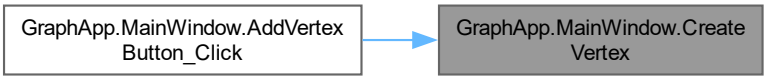
Створює нову вершину на графічному полотні.

Див. визначення в файлі MainWindow.xaml.cs, рядок 83

Граф всіх викликів цієї функції:



Граф викликів для цієї функції:



6.5.3.5 Edge_MouseDown()

void GraphApp.MainWindow.Edge_MouseDown (
object sender,
MouseButtonEventArgs e) [private]

Клас, що містить методи для роботи з ребрами та їх створення

Див. визначення в файлі MainWindow.xaml.cs, рядок 184

Граф всіх викликів цієї функції:

Граф викликів для цієї функції:

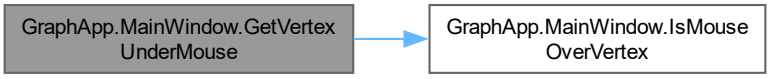
6.5.3.6 GetVertexUnderMouse()

Vertex GraphApp.MainWindow.GetVertexUnderMouse (
Point position) [private]

Отримання положення вершини

Див. визначення в файлі MainWindow.xaml.cs, рядок 250

Граф всіх викликів цієї функції:



Граф викликів для цієї функції:

6.5.3.7 IsMouseOverVertex()

bool GraphApp.MainWindow.IsMouseOverVertex (
Point position,
Vertex vertex) [private]

Перевірка, чи знаходиться курсор миші над заданою вершиною

Див. визначення в файлі MainWindow.xaml.cs, рядок 262

Граф викликів для цієї функції:

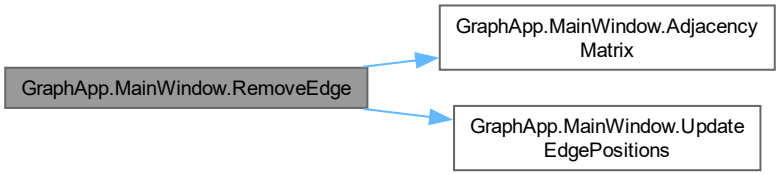
6.5.3.8 RemoveEdge()

void GraphApp.MainWindow.RemoveEdge (
Edge edge) [private]

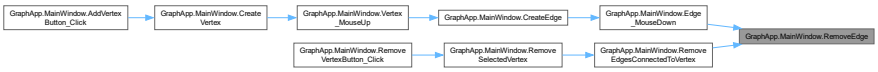
Видалення ребра

Див. визначення в файлі MainWindow.xaml.cs, рядок 307

Граф всіх викликів цієї функції:



Граф викликів для цієї функції:



6.5.3.9 RemoveEdgesConnectedToVertex()

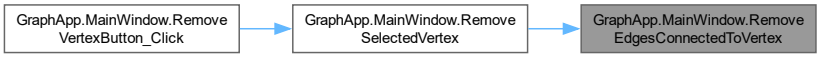
void GraphApp.MainWindow.RemoveEdgesConnectedToVertex (
Vertex vertex) [private]

Видалення всіх ребер, що з'єднані з заданою вершиною

Див. визначення в файлі MainWindow.xaml.cs, рядок 320

Граф всіх викликів цієї функції:

Граф викликів для цієї функції:



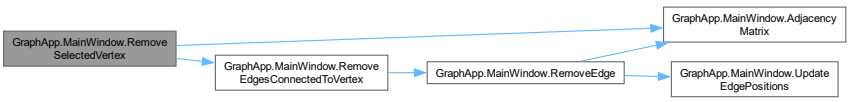
6.5.3.10 RemoveSelectedVertex()

void GraphApp.MainWindow.RemoveSelectedVertex () [private]

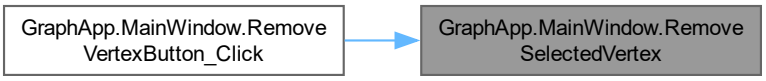
Видалення вибраної вершини

Див. визначення в файлі MainWindow.xaml.cs, рядок 279

Граф всіх викликів цієї функції:



Граф викликів для цієї функції:



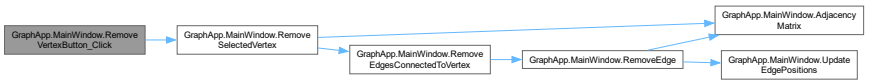
6.5.3.11 RemoveVertexButton_Click()

void GraphApp.MainWindow.RemoveVertexButton_Click (
object sender,
RoutedEventArgs e) [private]

Обробка події натиснення кнопки видалення вершини

Див. визначення в файлі MainWindow.xaml.cs, рядок 273

Граф всіх викликів цієї функції:



6.5.3.12 StartEdgeCreation()

void GraphApp.MainWindow.StartEdgeCreation () [private]

Початок створення ребра (перша вершина вибрана)

Див. визначення в файлі MainWindow.xaml.cs, рядок 208

Граф викликів для цієї функції:

6.5.3.13 StopEdgeCreation()

void GraphApp.MainWindow.StopEdgeCreation () [private]

Зупинка створення ребра

Див. визначення в файлі MainWindow.xaml.cs, рядок 225

Граф викликів для цієї функції:

6.5.3.14 ToggleEdgeCreationButton_Checked()

void GraphApp.MainWindow.ToggleEdgeCreationButton_Checked (
object sender,
RoutedEventArgs e) [private]

Обробка події включення кнопки створення ребра (активація кнопки Create Edge)

Див. визначення в файлі MainWindow.xaml.cs, рядок 338

6.5.3.15 ToggleEdgeCreationButton_Unchecked()

void GraphApp.MainWindow.ToggleEdgeCreationButton_Unchecked (
object sender,
RoutedEventArgs e) [private]

Обробка події вимкнення кнопки створення ребра (деактивація кнопки Create Edge)

Див. визначення в файлі MainWindow.xaml.cs, рядок 344

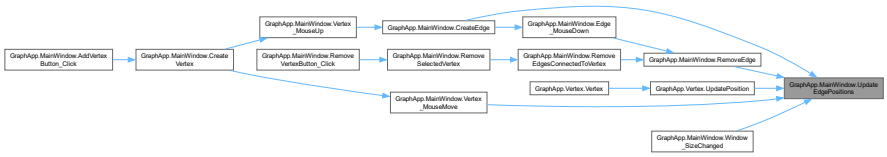
6.5.3.16 UpdateEdgePositions()

void GraphApp.MainWindow.UpdateEdgePositions (
Vertex vertex)

Оновлення позицій ребер, які з'єднані з заданою вершиною

Див. визначення в файлі MainWindow.xaml.cs, рядок 238

Граф викликів для цієї функції:



6.5.3.17 Vertex_MouseDown()

void GraphApp.MainWindow.Vertex_MouseDown (
object sender,
MouseButtonEventArgs e) [private]

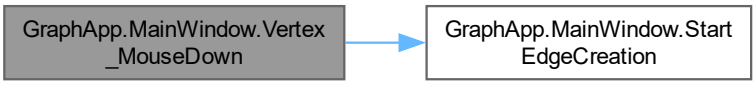
Обробник події натискання мишею на вершину. Або починає перетягування, або створює ребро.

Аргументи

sender	Об'єкт, що ініціював подію.
e	Параметри події.

Див. визначення в файлі MainWindow.xaml.cs, рядок 111

Граф всіх викликів цієї функції:



Граф викликів для цієї функції:

6.5.3.18 Vertex_MouseMove()

void GraphApp.MainWindow.Vertex_MouseMove (
object sender,
MouseEventArgs e) [private]

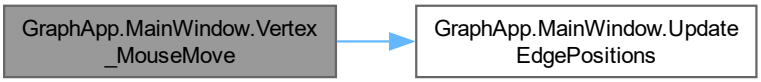
Обробник події руху миші над вершиною. Оновлює позицію вершини під час перетягування.

Аргументи

sender	Об'єкт, що ініціював подію.
e	Параметри події.

Див. визначення в файлі MainWindow.xaml.cs, рядок 131

Граф всіх викликів цієї функції:



Граф викликів для цієї функції:

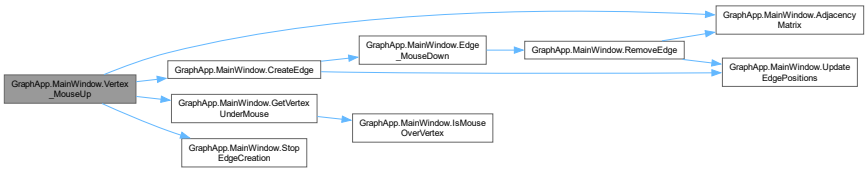
6.5.3.19 Vertex_MouseUp()

void GraphApp.MainWindow.Vertex_MouseUp (
object sender,
MouseButtonEventArgs e) [private]

Обробник події відпускання кнопки миші після перетягування вершини

Див. визначення в файлі MainWindow.xaml.cs, рядок 164

Граф всіх викликів цієї функції:



Граф викликів для цієї функції:

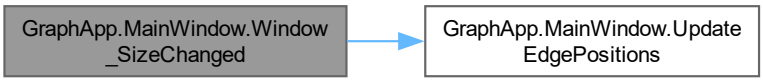
6.5.3.20 Window_SizeChanged()

void GraphApp.MainWindow.Window_SizeChanged (
object sender,
SizeChangedEventArgs e) [private]

перевірка зміни розміра вікна

Див. визначення в файлі MainWindow.xaml.cs, рядок 398

Граф всіх викликів цієї функції:



6.5.4 Компонентні дані

6.5.4.1 edgeLine

Line GraphApp.MainWindow.edgeLine [private]

Лінія, що представляє створюване ребро.

Див. визначення в файлі MainWindow.xaml.cs, рядок 36

6.5.4.2 edges

List<Edge> GraphApp.MainWindow.edges [private]

Список ребер графа.

Див. визначення в файлі MainWindow.xaml.cs, рядок 32

6.5.4.3 isCreatingEdge

bool GraphApp.MainWindow.isCreatingEdge [private]

Вказує, чи створюється нове ребро.

Див. визначення в файлі MainWindow.xaml.cs, рядок 34

6.5.4.4 isDragging

bool GraphApp.MainWindow.isDragging [private]

Вказує, чи відбувається перетягування вершини.

Див. визначення в файлі MainWindow.xaml.cs, рядок 33

6.5.4.5 selectedVertex

Vertex GraphApp.MainWindow.selectedVertex [private]

Вибрана вершина.

Див. визначення в файлі MainWindow.xaml.cs, рядок 35

6.5.4.6 vertices

List<Vertex> GraphApp.MainWindow.vertices [private]

Список вершин графа.

Див. визначення в файлі MainWindow.xaml.cs, рядок 31

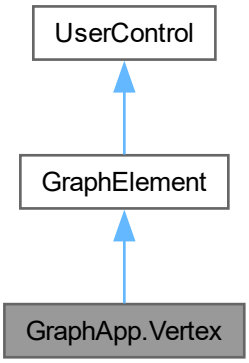
Документація цього класу була створена з файлу:

- GraphApp-main/GraphApp/MainWindow.xaml.cs

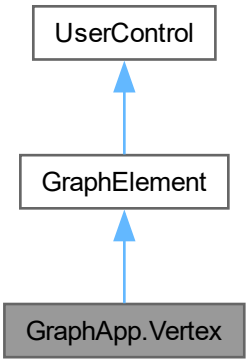
6.6 Клас GraphApp.Vertex

Представляє вершину в графі, яка є графічним елементом на Canvas.

Схема успадкувань для GraphApp.Vertex



Діаграма зв'язків класу GraphApp.Vertex:



Загальнодоступні елементи

- Vertex (int vertexNumber)

  Конструктор для створення вершини з заданим номером.
- override void UpdatePosition ()

  Оновлює положення вершини на Canvas.

Загальнодоступні елементи успадковано з GraphApp.GraphElement

- void UpdatePosition ()

  Оновлює положення елемента на полотні.

Властивості

- int VertexNumber [get, set]

  Номер вершини.
- double CenterX [get]

  Повертає X-координату центру вершини.
- double CenterY [get]

  Повертає Y-координату центру вершини.

Приватні елементи

- Style CreateRoundButtonStyle ()

  Створює стиль для вершини, щоб надати їй круглої форми.
- ControlTemplate CreateControlTemplate ()

  Створює шаблон управління для вершини з круглою формою.

6.6.1 Детальний опис

Представляє вершину в графі, яка є графічним елементом на Canvas.

Клас Vertex успадковується від GraphElement та представляє окрему вершину у графі, яка може бути переміщена мишею на Canvas. Також цей клас дозволяє відображати номер вершини і має властивості для доступу до її центральних координат.

Див. визначення в файлі MainWindow.xaml.cs, рядок 432

6.6.2 Конструктор(и)

6.6.2.1 Vertex()

GraphApp.Vertex.Vertex (
int vertexNumber)

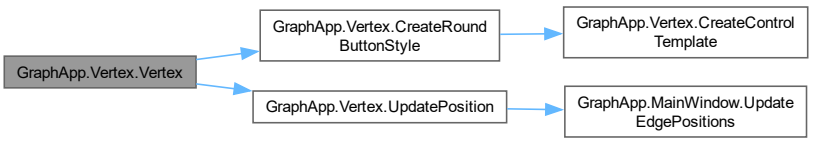
Конструктор для створення вершини з заданим номером.

Аргументи

vertexNumber	Номер вершини.

Див. визначення в файлі MainWindow.xaml.cs, рядок 455

Граф всіх викликів цієї функції:



6.6.3 Опис методів компонент

6.6.3.1 CreateControlTemplate()

ControlTemplate GraphApp.Vertex.CreateControlTemplate () [private]

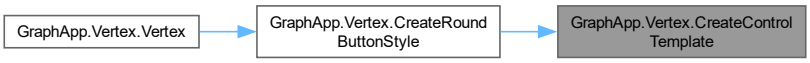
Створює шаблон управління для вершини з круглою формою.

Повертає

Шаблон елемента управління.

Див. визначення в файлі MainWindow.xaml.cs, рядок 486

Граф викликів для цієї функції:



6.6.3.2 CreateRoundButtonStyle()

Style GraphApp.Vertex.CreateRoundButtonStyle () [private]

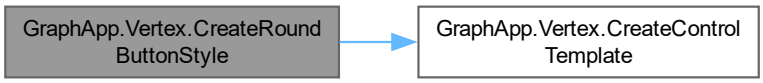
Створює стиль для вершини, щоб надати їй круглої форми.

Повертає

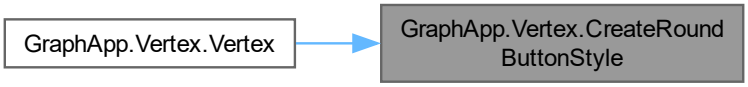
Стиль елемента.

Див. визначення в файлі MainWindow.xaml.cs, рядок 470

Граф всіх викликів цієї функції:



Граф викликів для цієї функції:



6.6.3.3 UpdatePosition()

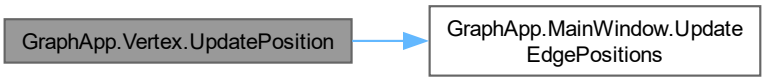
override void GraphApp.Vertex.UpdatePosition ()

Оновлює положення вершини на Canvas.

Використовується для оновлення координат вершини на Canvas та оновлення позицій пов'язаних з нею ребер.

Див. визначення в файлі MainWindow.xaml.cs, рядок 512

Граф всіх викликів цієї функції:



Граф викликів для цієї функції:

6.6.4 Повний список властивостей

6.6.4.1 CenterX

double GraphApp.Vertex.CenterX [get]

Повертає X-координату центру вершини.

Повертає

Центр вершини по X.

Див. визначення в файлі MainWindow.xaml.cs, рядок 443

6.6.4.2 CenterY

double GraphApp.Vertex.CenterY [get]

Повертає Y-координату центру вершини.

Повертає

Центр вершини по Y.

Див. визначення в файлі MainWindow.xaml.cs, рядок 449

6.6.4.3 VertexNumber

int GraphApp.Vertex.VertexNumber [get], [set]

Номер вершини.

Див. визначення в файлі MainWindow.xaml.cs, рядок 437

Документація цього класу була створена з файлу:

- GraphApp-main/GraphApp/MainWindow.xaml.cs

Розділ 7

Файли

7.1 Файл GraphApp-main/GraphApp/App.xaml.cs

Класи

- class GraphApp.App

  Interaction logic for App.xaml.

Простори імен

- namespace GraphApp

7.2 App.xaml.cs

Див. документацію.

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Configuration;
00004 using System.Data;
00005 using System.Linq;
00006 using System.Threading.Tasks;
00007 using System.Windows;
00008
00009 namespace GraphApp
00010 {
00014     public partial class App : Application
00015     {
00016     }
00017 }
```

7.3 Файл GraphApp-main/GraphApp/AssemblyInfo.cs

7.4 AssemblyInfo.cs

Див. документацію.

```
00001 using System.Windows;
00002
00003 [assembly: ThemeInfo(
00004     ResourceDictionaryLocation.None, //where theme specific resource dictionaries are located
00005                                      //(used if a resource is not found in the page,
00006                                      // or application resource dictionaries)
00007     ResourceDictionaryLocation.SourceAssembly //where the generic resource dictionary is located
00008                                               //(used if a resource is not found in the page,
00009                                               // app, or any theme specific resource dictionaries)
00010 )]
```

7.5 Файл GraphApp-main/GraphApp/MainWindow.xaml.cs

Файл містить реалізацію головного вікна програми та логіку взаємодії з графом (вершинами та ребрами).

Класи

- class GraphApp.MainWindow

  Головне вікно програми для побудови та маніпуляції графами.
- class GraphApp.GraphElement

  Абстрактний клас, що представляє графічний елемент на полотні (Canvas).
- class GraphApp.Vertex

  Представляє вершину в графі, яка є графічним елементом на Canvas.
- class GraphApp.Edge

  Представляє ребро між двома вершинами.
- class GraphApp.Arrowhead

  Представляє стрілку на кінці ребра.

Простори імен

- namespace GraphApp

7.5.1 Детальний опис

Файл містить реалізацію головного вікна програми та логіку взаємодії з графом (вершинами та ребрами).

Див. визначення в файлі MainWindow.xaml.cs

7.6 MainWindow.xaml.cs

Див. документацію.

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Collections.ObjectModel;
00004 using System.Linq;
00005 using System.Text;
00006 using System.Threading.Tasks;
00007 using System.Windows;
00008 using System.Windows.Controls;
00009 using System.Windows.Controls.Primitives;
00010 using System.Windows.Data;
00011 using System.Windows.Documents;
00012 using System.Windows.Input;
00013 using System.Windows.Media;
00014 using System.Windows.Media.Imaging;
00015 using System.Windows.Navigation;
00016 using System.Windows.Shapes;
00017
00023 namespace GraphApp
00024 {
00029     public partial class MainWindow : Window
00030     {
00031     private List<Vertex> vertices;
00032     private List<Edge> edges;
00033     private bool isDragging;
00034     private bool isCreatingEdge;
00035     private Vertex selectedVertex;
00036     private Line edgeLine;
```

```
00061         public MainWindow()
00062         {
00063             InitializeComponent();
00064             // Ініціалізуємо змінні
00065             vertices = new List<Vertex>();
00066             edges = new List<Edge>();
00067             isDragging = false;
00068             isCreatingEdge = false;
00069             selectedVertex = null;
00070             edgeLine = null;
00071             label.Content = $"Selected Vertex: None\nPosition (X:{0} Y:{0})";
00072         }
00073 
00074 
00075         private void AddVertexButton_Click(object sender, RoutedEventArgs e)
00076         {
00077             CreateVertex();
00078         }
00079 
00083         private void CreateVertex()
00084         {
00085             // Змінна, що визначає номер вершини
00086             int vertexNumber = vertices.Count + 1;
00087             // Створення нового об'єкта вершини
00088             var obj = new Vertex(vertexNumber);
00089             // Поліморфна змінна типу GraphElement, яка посилається на об'єкт вершини
00090             GraphElement vertex = obj;
00091             // Прикріплення обробників подій до вершини
00092             vertex.MouseDown += Vertex_MouseDown;
00093             vertex.MouseMove += Vertex_MouseMove;
00094             vertex.MouseUp += Vertex_MouseUp;
00095             // Встановлення позиції вершини на Canvas
00096             Canvas.SetLeft(vertex, 100);
00097             Canvas.SetTop(vertex, 100);
00098             // Додавання вершини до колекції на Canvas
00099             graphCanvas.Children.Add(vertex);
00100             // Додавання вершини до колекції vertices
00101             vertices.Add((Vertex)vertex);
00102             // Оновлення матриці суміжності
00103             AdjacencyMatrix();
00104         }
00105 
00111         private void Vertex_MouseDown(object sender, MouseButtonEventArgs e)
00112         {
00113             if (isCreatingEdge)
00114             {
00115                 selectedVertex = sender as Vertex;
00116                 StartEdgeCreation();
00117             }
00118             else
00119             {
00120                 selectedVertex = sender as Vertex;
00121                 isDragging = true;
00122                 selectedVertex.CaptureMouse();
00123             }
00124         }
00125 
00131         private void Vertex_MouseMove(object sender, MouseEventArgs e)
00132         {
00133             if (isDragging)
00134             {
00135                 // Отримання нових координат вершини
00136                 double newX = e.GetPosition(graphCanvas).X - (selectedVertex.ActualWidth / 2);
00137                 double newY = e.GetPosition(graphCanvas).Y - (selectedVertex.ActualHeight / 2);
00138 
00139                 // Перевірка, чи вершина не виходить за межі Canvas
00140                 double canvasWidth = graphCanvas.ActualWidth;
00141                 double canvasHeight = graphCanvas.ActualHeight;
00142 
00143                 if (newX < 0)
00144                     newX = 0;
00145                 else if (newX + selectedVertex.ActualWidth > canvasWidth)
00146                     newX = canvasWidth - selectedVertex.ActualWidth;
00147 
00148                 if (newY < 0)
00149                     newY = 0;
00150                 else if (newY + selectedVertex.ActualHeight > canvasHeight)
00151                     newY = canvasHeight - selectedVertex.ActualHeight;
00152 
00153                 label.Content = $"Selected Vertex: {selectedVertex.Content}\nPosition (X:{newX} Y:{newY})";
00154                 // Зміна позиції вершини
00155                 Canvas.SetLeft(selectedVertex, newX);
00156                 Canvas.SetTop(selectedVertex, newY);
00157                 UpdateEdgePositions(selectedVertex);
00158             }
00159         }
00160 
```

```
00161
00162
00164       private void Vertex_MouseUp(object sender, MouseButtonEventArgs e)
00165       {
00166           if (isCreatingEdge)
00167           {
00168               var targetVertex = GetVertexUnderMouse(e.GetPosition(graphCanvas));
00169               if (targetVertex != null && targetVertex != selectedVertex)
00170               {
00171                   CreateEdge(selectedVertex, targetVertex);
00172               }
00173               StopEdgeCreation();
00174           }
00175           else if (isDragging)
00176           {
00177               isDragging = false;
00178               selectedVertex.ReleaseMouseCapture();
00179           }
00180           AdjacencyMatrix();
00181       }
00182
00184       private void Edge_MouseDown(object sender, MouseButtonEventArgs e)
00185       {
00186           if (isCreatingEdge) return;
00187
00188           var clickedEdge = sender as Edge;
00189           RemoveEdge(clickedEdge);
00190       }
00191
00193       private void CreateEdge(Vertex startVertex, Vertex endVertex)
00194       {
00195           // Створюємо нове ребро з початковою та кінцевою вершинами
00196           var edge = new Edge(startVertex, endVertex);
00197           // Додано обробник події кліку мишею на ребро
00198           edge.MouseDown += Edge_MouseDown;
00199           // Додаємо ребро до колекції ребер
00200           edges.Add(edge);
00201           // Додаємо ребро до графічного полотну
00202           graphCanvas.Children.Add(edge);
00203           // Оновлюємо позиції ребер для початкової вершини
00204           UpdateEdgePositions(startVertex);
00205       }
00206
00208       private void StartEdgeCreation()
00209       {
00210           // Створюємо новий об'єкт лінії
00211           edgeLine = new Line
00212           {
00213               Stroke = Brushes.Black,
00214               StrokeThickness = 2,
00215               X1 = selectedVertex.CenterX,
00216               Y1 = selectedVertex.CenterY,
00217               X2 = selectedVertex.CenterX,
00218               Y2 = selectedVertex.CenterY
00219           };
00220           // Додаємо лінію до графічного канвасу
00221           graphCanvas.Children.Add(edgeLine);
00222       }
00223
00225       private void StopEdgeCreation()
00226       {
00227           //якщо вже була створена лінія
00228           if (edgeLine != null)
00229           {
00230               // Видаляємо лінію з графічного канвасу
00231               graphCanvas.Children.Remove(edgeLine);
00232               // Звільняємо пам'ять, присвоюючи значення null
00233               edgeLine = null;
00234           }
00235       }
00236
00238       public void UpdateEdgePositions(Vertex vertex)
00239       {
00240           foreach (var edge in edges)
00241           {
00242               if (edge.StartVertex == vertex || edge.EndVertex == vertex)
00243               {
00244                   edge.UpdatePosition();
00245               }
00246           }
00247       }
00248
00250       private Vertex GetVertexUnderMouse(Point position)
00251       {
00252           foreach (var vertex in vertices)
00253           {
00254               if (IsMouseOverVertex(position, vertex))
```

```
00255             {
00256                 return vertex;
00257             }
00258         }
00259         return null;
00260     }
00262     private bool IsMouseOverVertex(Point position, Vertex vertex)
00263     {
00264         double left = Canvas.GetLeft(vertex);
00265         double top = Canvas.GetTop(vertex);
00266         double right = left + vertex.ActualWidth;
00267         double bottom = top + vertex.ActualHeight;
00268 
00269         return (position.X >= left && position.X <= right && position.Y >= top && position.Y <= bottom);
00270     }
00271 
00273     private void RemoveVertexButton_Click(object sender, RoutedEventArgs e)
00274     {
00275         RemoveSelectedVertex();
00276     }
00277 
00279     private void RemoveSelectedVertex()
00280     {
00281         if (selectedVertex != null)
00282         {
00283             // Видалення ребер, що з'єднуються з вибраною вершиною
00284             RemoveEdgesConnectedToVertex(selectedVertex);
00285             // Видалення вибраної вершини з графічного контейнера
00286             graphCanvas.Children.Remove(selectedVertex);
00287             // Знаходження індексу видаленої вершини в списку вершин
00288             int removedIndex = vertices.IndexOf(selectedVertex);
00289             // Видалення вершини зі списку вершин
00290             vertices.RemoveAt(removedIndex);
00291             // Позначення вибраної вершини як null
00292             selectedVertex = null;
00293             // Зменшення індексів вершин, які мають індекси більші за видалену вершину
00294             for (int i = removedIndex; i < vertices.Count; i++)
00295             {
00296                 vertices[i].VertexNumber -= 1;
00297                 vertices[i].Content = vertices[i].VertexNumber;
00298             }
00299         }
00300         // Оновлення вмісту мітки з вибраною вершиною
00301         label.Content = $"Selected Vertex: None\nPosition (X:{0} Y:{0})";
00302         // Оновлення матриці суміжності
00303         AdjacencyMatrix();
00304     }
00305 
00307     private void RemoveEdge(Edge edge)
00308     {
00309         // Видаляємо ребро з візуального контейнера
00310         graphCanvas.Children.Remove(edge);
00311         // Видаляємо ребро зі списку ребер
00312         edges.Remove(edge);
00313         // Оновлюємо позиції ребер, які виходять з початкової вершини
00314         UpdateEdgePositions(edge.StartVertex);
00315         // Оновлюємо матрицю суміжності
00316         AdjacencyMatrix();
00317     }
00318 
00320     private void RemoveEdgesConnectedToVertex(Vertex vertex)
00321     {
00322         // Цикл для проходження по усім ребрам у зворотному порядку
00323         for (int i = edges.Count - 1; i >= 0; i--)
00324         {
00325             // Отримуємо поточне ребро
00326             var edge = edges[i];
00327 
00328             // Перевіряємо, чи ребро з'єднане з початковим або кінцевим вершинами
00329             if (edge.StartVertex == vertex || edge.EndVertex == vertex)
00330             {
00331                 // Видаляємо ребро
00332                 RemoveEdge(edge);
00333             }
00334         }
00335     }
00336 
00338     private void ToggleEdgeCreationButton_Checked(object sender, RoutedEventArgs e)
00339     {
00340         isCreatingEdge = true;
00341     }
00342 
00344     private void ToggleEdgeCreationButton_Unchecked(object sender, RoutedEventArgs e)
00345     {
00346         if (isCreatingEdge && edgeLine != null)
00347         {
00348             graphCanvas.Children.Remove(edgeLine);
```

```
00349                 edgeLine = null;
00350             }
00351             isCreatingEdge = false;
00352         }
00354         private void AdjacencyMatrix()
00355         {
00356             // Створення матриці суміжності
00357             int[,] adjacencyMatrix = new int[vertices.Count, vertices.Count];
00358             // Заповнення матриці суміжності з використанням індексів вершин у списку
00359             foreach (var edge in edges)
00360             {
00361                 int startVertexIndex = vertices.IndexOf(edge.StartVertex);
00362                 int endVertexIndex = vertices.IndexOf(edge.EndVertex);
00363                 adjacencyMatrix[startVertexIndex, endVertexIndex] = 1;
00364             }
00365             // Очистити колекції стовпців та рядків у DataGrid
00366             matrixDataGrid.Columns.Clear();
00367             matrixDataGrid.Items.Clear();
00368             // Додати стовпець для номерів вершин
00369             matrixDataGrid.Columns.Add(new DataGridTextColumn
00370             {
00371                 Header = " ",
00372                 Binding = new Binding($"[{0}]")
00373             });
00374             // Додати стовпці для елементів матриці та заповнити дані
00375             for (int i = 0; i < vertices.Count; i++)
00376             {
00377                 var column = new DataGridTextColumn
00378                 {
00379                     Header = $"{vertices[i].VertexNumber}",
00380                     Binding = new Binding($"[{i + 1}]")
00381                 };
00382 
00383                 matrixDataGrid.Columns.Add(column);
00384             }
00385             // Заповнити рядки матриці
00386             for (int i = 0; i < vertices.Count; i++)
00387             {
00388                 var item = new ObservableCollection<object>();
00389                 item.Add(vertices[i].VertexNumber);
00390                 for (int j = 0; j < vertices.Count; j++)
00391                 {
00392                     item.Add(adjacencyMatrix[i, j]);
00393                 }
00394                 matrixDataGrid.Items.Add(item);
00395             }
00396         }
00398         private void Window_SizeChanged(object sender, SizeChangedEventArgs e)
00399         {
00400             foreach (var item in vertices)
00401             {
00402                 UpdateEdgePositions(item);
00403             }
00404 
00405         }
00406     }
00413 public abstract class GraphElement : UserControl
00414 {
00419     public abstract void UpdatePosition();
00420 }
00421 
00422 
00432     public class Vertex : GraphElement
00433     {
00437     public int VertexNumber { get; set; }
00438 
00443     public double CenterX => Canvas.GetLeft(this) + (ActualWidth / 2);
00444 
00449     public double CenterY => Canvas.GetTop(this) + (ActualHeight / 2);
00450 
00455     public Vertex(int vertexNumber)
00456     {
00457         VertexNumber = vertexNumber;
00458         Content = vertexNumber.ToString();
00459         Width = 50;
00460         Height = 50;
00461         Background = Brushes.LightBlue;
00462         Style = CreateRoundButtonStyle();
00463         UpdatePosition();
00464     }
00465 
00470     private Style CreateRoundButtonStyle()
00471     {
00472         var style = new Style(typeof(Vertex));
00473 
00474         style.Setters.Add(new Setter(WidthProperty, 50.0));
00475         style.Setters.Add(new Setter(HeightProperty, 50.0));
```

```
00476         style.Setters.Add(new Setter(BackgroundProperty, Brushes.LightBlue));
00477         style.Setters.Add(new Setter(TemplateProperty, CreateControlTemplate()));
00478
00479         return style;
00480     }
00481
00486     private ControlTemplate CreateControlTemplate()
00487     {
00488         var template = new ControlTemplate(typeof(Vertex));
00489         var border = new FrameworkElementFactory(typeof(Border));
00490         border.SetValue(Border.BorderThicknessProperty, new Thickness(1));
00491         border.SetValue(Border.BorderBrushProperty, Brushes.Black);
00492         border.SetValue(Border.BackgroundProperty, new TemplateBindingExtension(BackgroundProperty));
00493         border.SetValue(Border.CornerRadiusProperty, new CornerRadius(25));
00494         var contentPresenter = new FrameworkElementFactory(typeof(ContentPresenter));
00495         contentPresenter.SetValue(ContentPresenter.HorizontalAlignmentProperty, HorizontalAlignment.Center);
00496         contentPresenter.SetValue(ContentPresenter.VerticalAlignmentProperty, VerticalAlignment.Center);
00497         contentPresenter.SetValue(ContentPresenter.ContentProperty, new TemplateBindingExtension(ContentProperty));
00498
00499         border.AppendChild(contentPresenter);
00500
00501         template.VisualTree = border;
00502
00503         return template;
00504     }
00505
00512     public override void UpdatePosition()
00513     {
00514         double newX = Canvas.GetLeft(this);
00515         double newY = Canvas.GetTop(this);
00516
00517         Canvas.SetLeft(this, newX);
00518         Canvas.SetTop(this, newY);
00519
00520         // Оновлення позиції ребер, пов'язаних з цією вершиною
00521         MainWindow mainWindow = Application.Current.MainWindow as MainWindow;
00522         mainWindow.UpdateEdgePositions(this);
00523     }
00524     }
00525
00534     public class Edge : GraphElement
00535 {
00539     private Line line;
00540
00544     private Arrowhead arrowhead;
00545
00549     public Vertex StartVertex { get; }
00550
00554     public Vertex EndVertex { get; }
00555
00561     public Edge(Vertex startVertex, Vertex endVertex)
00562     {
00563         StartVertex = startVertex;
00564         EndVertex = endVertex;
00565
00566         line = new Line
00567         {
00568             Stroke = Brushes.Black,
00569             StrokeThickness = 2
00570         };
00571         arrowhead = new Arrowhead
00572         {
00573             Fill = Brushes.Black
00574         };
00575         UpdatePosition();
00576         Panel.SetZIndex(this, -1);
00577         Content = new Canvas { Children = { line, arrowhead } };
00578
00579         // Оновлення положення стрілки при зміні розташування вершин
00580         StartVertex.SizeChanged += Vertex_SizeChanged;
00581         EndVertex.SizeChanged += Vertex_SizeChanged;
00582     }
00583
00590     public override void UpdatePosition()
00591     {
00592         double startX = StartVertex.CenterX;
00593         double startY = StartVertex.CenterY;
00594         double endX = EndVertex.CenterX;
00595         double endY = EndVertex.CenterY;
00596
00597         double angle = Math.Atan2(endY - startY, endX - startX);
00598
00599         double offset = StartVertex.ActualHeight / 2;
00600
00601         double lineStartX = startX + Math.Cos(angle) * offset;
00602         double lineStartY = startY + Math.Sin(angle) * offset;
00603         double lineEndX = endX - Math.Cos(angle) * offset;
```

```
00604         double lineEndY = endY - Math.Sin(angle) * offset;
00605
00606         line.X1 = lineStartX;
00607         line.Y1 = lineStartY;
00608         line.X2 = lineEndX;
00609         line.Y2 = lineEndY;
00610
00611         double arrowheadSize = 10;
00612         double halfArrowheadSize = arrowheadSize / 2;
00613         double arrowheadOffset = StartVertex.ActualHeight / 2 + halfArrowheadSize;
00614
00615         double arrowheadX = lineStartX + (lineEndX - lineStartX) / 1.02;
00616         double arrowheadY = lineStartY + (lineEndY - lineStartY) / 1.02;
00617
00618         arrowhead.Width = arrowheadSize;
00619         arrowhead.Height = arrowheadSize;
00620         Canvas.SetLeft(arrowhead, arrowheadX - halfArrowheadSize);
00621         Canvas.SetTop(arrowhead, arrowheadY - halfArrowheadSize);
00622         arrowhead.RenderTransform = new RotateTransform(angle * (180 / Math.PI), halfArrowheadSize,
      halfArrowheadSize);
00623
00624         Canvas.SetZIndex(this, Math.Min(Canvas.GetZIndex(StartVertex), Canvas.GetZIndex(EndVertex)) - 1);
00625     }
00626
00632     private void Vertex_SizeChanged(object sender, SizeChangedEventArgs e)
00633     {
00634         UpdatePosition();
00635     }
00636 }
00637
00645     public class Arrowhead : Shape
00646     {
00651         protected override Geometry DefiningGeometry
00652         {
00653             get
00654             {
00655                 double arrowheadSize = 10;
00656                 double halfArrowheadSize = arrowheadSize / 2;
00657                 PathGeometry geometry = new PathGeometry();
00658                 PathFigure figure = new PathFigure();
00659                 figure.StartPoint = new Point(0, 0);
00660                 figure.Segments.Add(new LineSegment(new Point(arrowheadSize, halfArrowheadSize), isStroked: true));
00661                 figure.Segments.Add(new LineSegment(new Point(0, arrowheadSize), isStroked: true));
00662                 figure.IsClosed = true;
00663                 geometry.Figures.Add(figure);
00664                 return geometry;
00665             }
00666         }
00667         protected override void OnRender(DrawingContext drawingContext)
00668         {
00669             var customPen = new Pen(Brushes.Black, 2.0);
00670             drawingContext.DrawGeometry(Fill, customPen, DefiningGeometry);
00671         }
00672     }
00673
00674 }
```

Предметний покажчик