AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Baza danych 2

Dokumentacja projektu

Obsługa danych przestrzennych dwuwymiarowych.

Łukasz Wajda, 12.06.2022 r.

Spis treści

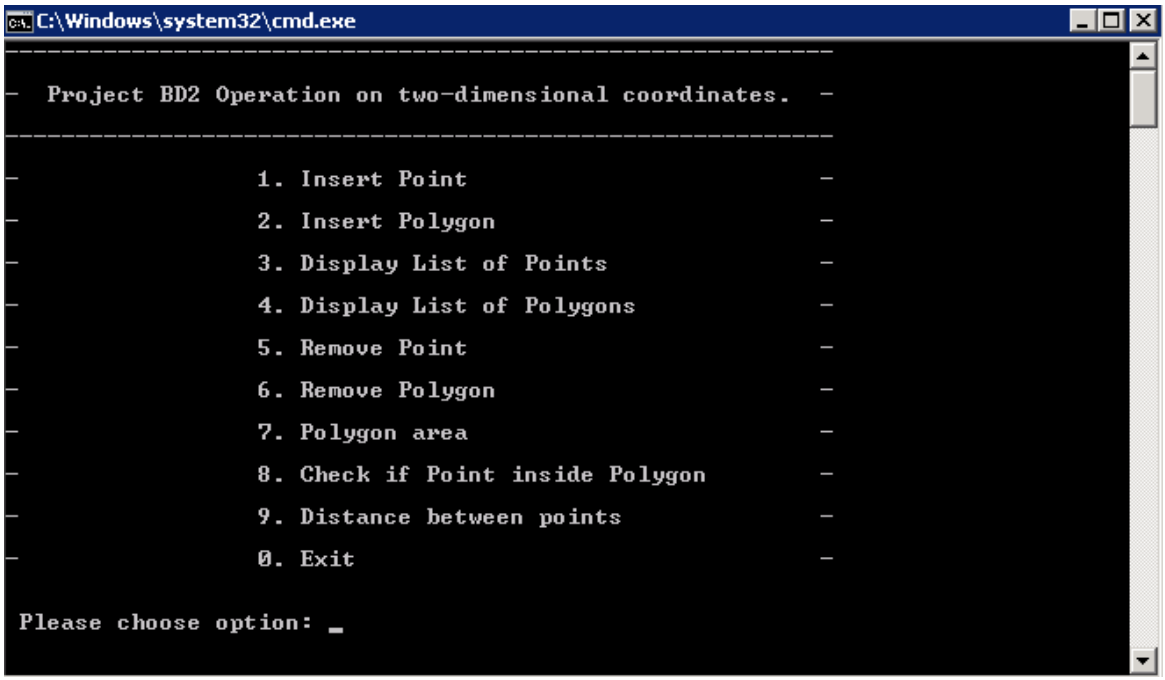
1. Założenia projektu.

Celem projektu było przygotowanie API i jego implementacji umożliwiającej przetwarzanie danych dwuwymiarowych z wykorzystaniem własnych typów danych **CLR**, **UDT** oraz metod. Przygotowane metody umożliwiają: wyznaczenie odległości pomiędzy punktami, wyznaczenie pola określonego obszaru, sprawdzenie czy punkt należy do danego obszaru.

2. Opis funkcjonalności API.

W ramach aplikacji możliwe jest dokonanie operacji na punktach oraz wielokątach. Aplikacja pozwala dodawać dowolną ilość punktów określonych przez współrzędne całkowite X oraz Y oraz wyznaczanie odległości między nimi jak i sprawdzenie czy punkt należy do obszaru wyznaczonego przez wielokąt. Wielokąty określone są przez kombinację punktów. Kolejność dodawanych punktów wielokąta nie ma znaczenia – zaimplementowano algorytm sortujący wierzchołki. Dla wielokątów umożliwiono wyznaczanie pola obszaru.

W celu wygodnej obsługi złożonych typów danych udostępniono interfejs konsolowy.



Rysunek 1 Menu główne.

W aplikacji konsolowej poruszamy się wybierając odpowiednie opcje, a następnie zatwierdzając klawiszem Enter. Wybierając opcję 1 lub 2 zostaniemy poproszeni o podanie odpowiednich danych w celu zapisania punktu/wielokąta do bazy danych. Przykładowe wejście jest podane w nawiasach kwadratowy. Wybierając opcje 3 lub 4 zostanie wyświetlona lista dostępnych punktów/wielokątów w bazie danych. Wybierając opcje 5 lub 6 zostaje wyświetlona aktualna lista elementów z bazy, które możemy usunąć podając ID rekordu. Wybierając opcję 7 zostaje wyświetlona lista dostępnych wielokątów, a po wyborze ID wielokąta zostaje wyświetlona obliczona wartość pola.

Wybierając opcję 8 w pierwszej kolejności zostaje wyświetlona lista dostępnych wielokątów, a po wyborze ID wielokąta, zostaje wyświetlona lista punktów dostępnych do wyboru. Wybierając opcję 9 zostanie wyświetlona lista dostępnych punktów i zostaniemy poproszeni o wybranie dwóch punktów, aby policzyć dystans między nimi. Wybranie opcji 0 zamyka aplikację.

3. Opis typów danych oraz metod udostępnionych w ramach API.

3.1. Klasy User Defined Types.

Każda poniżej wymieniana klasa posiada metody takie jak:

- `konstruktory,`
- `zestaw getterów oraz seterów,`
- `public void Read(System.IO.BinaryReader value)` – metoda deserializująca obiekt, dla klasy Point wygenerowana automatycznie, dla Polygon zaimplementowano własną,
- `public void Write(System.IO.BinaryWriter value)`– metoda serializująca obiekt, dla klasy Point wygenerowana automatycznie, dla Polygon zaimplementowano własną,
- `public override string ToString()`,
- `public static Point/Polygon Parse(SqlString value)`

Zdefiniowano dwie klasy reprezentujące odpowiednio punkt oraz wielokąt:
Point – klasa reprezentująca typ złożony – punkt

Pola klasy:

- `private int _x`– współrzędna X punktu,
- `private int _y`– współrzędna Y punktu,
- `private bool isNull`- wartość definiująca czy obiekt jest zainicjalizowany.

Metody klasy:

- `public SqlDouble distance(Point p)`– metoda obliczająca odległość między dwoma punktami.

Polygon – klasa reprezentująca typ złożony – wielokąt

Pola klasy:

- `private List<Point> polygonPoints`- lista punktów tworzących wielokąt,
- `private bool isNull` - wartość definiująca czy obiekt jest zainicjalizowany.

Metody klasy:

- `private bool isPolygonCorrect()` – metoda walidująca poprawność wielokąta,
- `private Point averagePointInside()` – metoda zwracająca punkt referencyjny znajdujący się we wnętrzu wielokąta,
- `private void sortAngular()` – metoda sortuje punkty tworzące wielokąt,

- `public SqlDouble area()` – metoda obliczająca pole wielokąta,
- `public SqlBoolean pointInsidePolygon(Point P)` – metoda sprawdzająca czy punkt należy do wielokąta.

3.2. Klasy aplikacji konsolowej.

Aplikacja konsolowa składa się z jednej klasy - **Program**.

Pola klasy:

- `static String sqlconnection` – zawiera connection string do bazy danych.

Metody klasy:

- `static void Main(string[] args)` – główna metoda klasy,
- `private static bool AppMenu()` – metoda implementująca interfejs konsolowy,
- `private static void getPoints()` – metoda wyświetlająca punkty dostępne w bazie danych,
- `private static void getPolygons()` – metoda wyświetlająca wielokąty dostępne w bazie,
- `private static void addPoint(string value)` – metoda wprowadzająca rekord do tabeli punktów,
- `private static void addManuallyPolygon(string value)` – metoda wprowadzająca rekord do tabeli wielokątów,
- `private static void addPolygonUsingPointFromTable (string value)` – metoda wprowadzająca rekord do tabeli wielokątów,
- `private static void removePoint(string pointID)` – metoda usuwająca punkt,
- `private static void removePolygon(string polygonID)` – metoda usuwająca wielokąt,
- `private static void returnAreaValueFromPolygon(string id)` – metoda wyświetlająca pole wielokąta,
- `private static void pointInsidePolygon(string idPolygon, string idPoint)` – metoda wyświetlająca informację o przynależności punktu do wielokąta,
- `private static void distanceOfPoints(string firstPoint, string secondPoint)` – metoda wyświetlająca dystans pomiędzy dwoma punktami.

4. Opis implementacji.

Wymaganiami projektu było udostępnienie możliwości obliczania pola obszaru, odległości między punktami, sprawdzenia przynależności punktu do obszaru wyznaczonego przez wielokąt. W celu spełnienia zadanych wymagań zaimplementowano metody, które będą zajmowały się obliczaniem wyników po stronie bazy danych.

4.1 Odległość pomiędzy dwoma punktami

Skorzystano z odległości euklidesowej między dwoma punktami. Zakładając, że mamy dane punkty: A(x1, y1) oraz B(x2, y2) odległość między A i B możemy obliczyć korzystając z następującego wzoru:

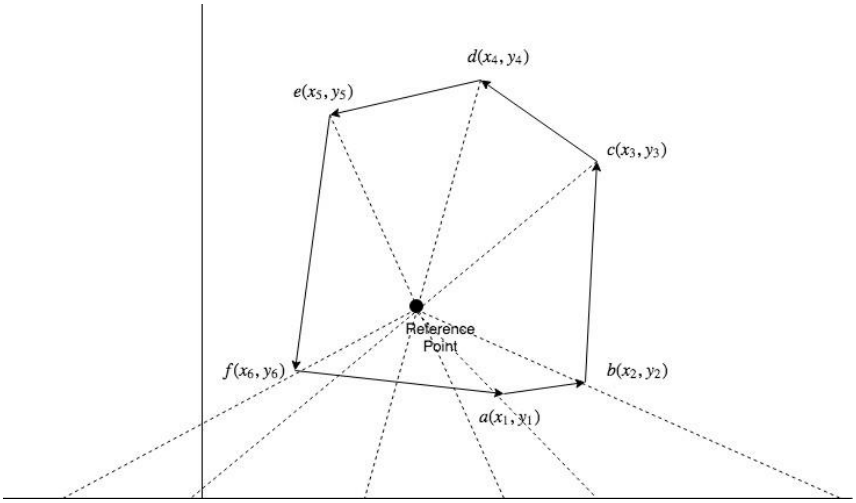
$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

4.2 Pole obszaru zadanego przez wielokąt.

Jeśli wierzchołki wielokąta są posortowane zgodnie z kierunkiem wskazówek zegara lub przeciwnie to pole wielokąta możemy otrzymać stosując Algorytm Gaussa nazywany również Shoelace Formula:

$$A = \frac{1}{2}\left|\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right|$$

Dlatego, że wymogiem algorytmu jest to, aby wierzchołki były posortowane zgodnie lub przeciwnie do ruchu wskazówek zegara zaimplementowano sortowanie wierzchołków.
Do sortowania wierzchołków używany jest punkt referencyjny, który znajduje się we wnętrzu wielokąta. Założeniem programu jest przechowywanie wielokątów wypukłych więc współrzędne punktu referencyjnego będą średnią arytmetyczną współrzędnych wierzchołków.

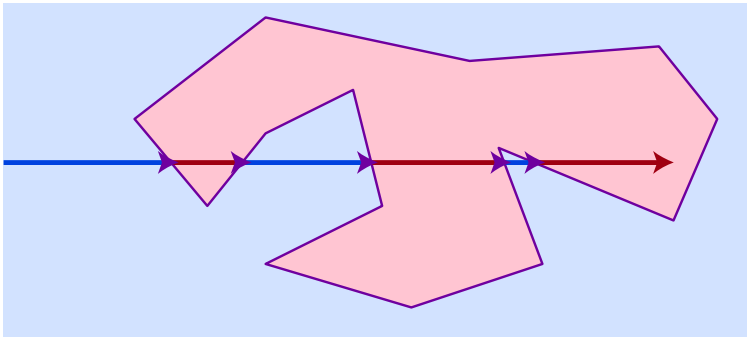


Rysunek 2 Sortowanie wierzchołków wielokąta według punktu referencyjnego.

4.3 Sprawdzanie czy punkt należy do wnętrza wielokąta.

W celu sprawdzenia czy punkt należy do wnętrza wielokąta skorzystano z algorytmu Ray Casting. Algorytm ten sprawdza ilość przecięć pomiędzy promieniem przechodzącym przez punkt, a krawędziami wielokąta. Jeśli punkt znajduje się na zewnątrz wielokąta, promień przetnie jego krawędzie parzystą liczbę razy, a jeśli punkt znajduję się wewnątrz wielokąta –

nieparzystą liczbę razy.



Rysunek 3 Wizualizacja działania algorytmu Ray Casting (crossing number).

5. Prezentacja przeprowadzonych testów.

W celu sprawdzenia poprawnego działania aplikacji zostały przygotowane testy jednostkowe. Zostało wykonanych 15 testów jednostkowych testujących utworzone typy UDT wraz z ich metodami.

Passed	TestFalseForPolygonPoi	TwoDimensionalUnitTests
Passed	TestDistanceOfPoints	TwoDimensionalUnitTests
Passed	TestPolygonTryToInsert	TwoDimensionalUnitTests
Passed	TestGetYFromPoint	TwoDimensionalUnitTests
Passed	TestTrueForPolygonPoir	TwoDimensionalUnitTests
Passed	TestPolygonTryToInsert	TwoDimensionalUnitTests
Passed	TestPolygonInsertTrue	TwoDimensionalUnitTests
Passed	TestPointToString	TwoDimensionalUnitTests
Passed	TestPointTryToInsertSt	TwoDimensionalUnitTests
Passed	TestPointTryToInsertDc	TwoDimensionalUnitTests
Passed	TestInsertWithEqualVer	TwoDimensionalUnitTests
Passed	TestPolygonToString	TwoDimensionalUnitTests
Passed	TestInsertWithLessThar	TwoDimensionalUnitTests
Passed	TestAreaOfPolygon	TwoDimensionalUnitTests
Passed	TestGetXFromPoint	TwoDimensionalUnitTests

Rysunek 4 Wykonane testy jednostkowe.

W celu uruchomienia testów należy otworzyć projekt TwoDimensionalTypeData i wybrać opcję Test -> Run -> All Tests in Solution.

W przypadku aplikacji konsolowej, która wyświetla interfejs użytkownika skupiono się na testach manualnych. Aplikacja ta wywołuje odpowiednie metody typów złożonych UDT, które są obsługiwane przez bazę danych, a testy tych metod zostały wykonane wcześniej.

6. Typy oferowane przez SQL SERVER

Warto również wspomnieć o tym, że SQL Server udostępnia predefiniowane obiekty przechowywujące dane przestrzenne, niektóre z nich:

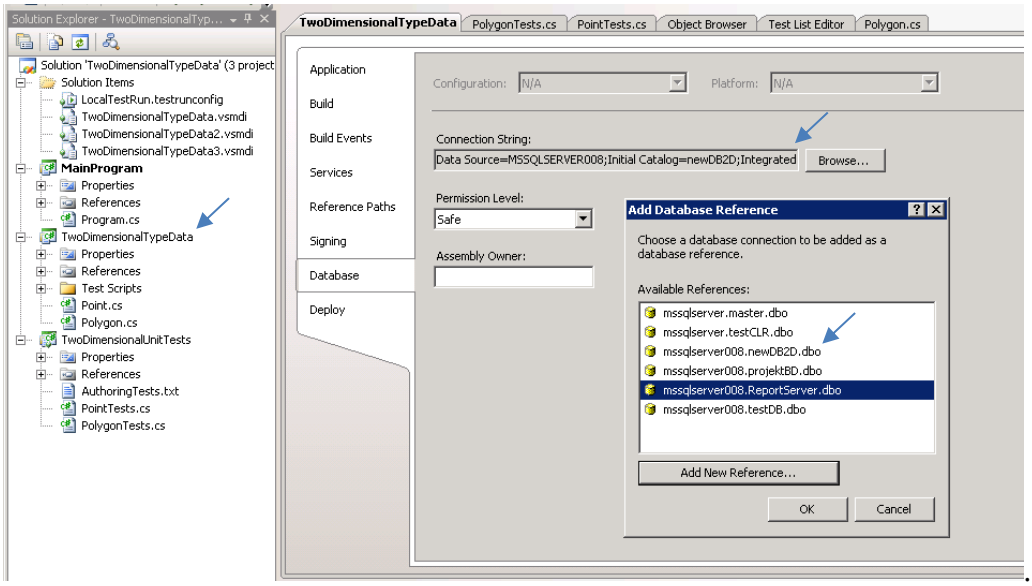
- **Point** – reprezentuje punkt na płaszczyźnie, posiada dwa pola, które reprezentują długość oraz szerokość geograficzną.
- **Polygon** – reprezentuje dwuwymiarową powierzchnię zdefiniowaną jako sekwencję punktów, które są wierzchołkami. Punkty te formują zewnętrzną granicę oraz zero lub więcej wewnętrznych pierścieni.

Porównując obiekt z biblioteki SQL Server do tego utworzonego na potrzeby projektu, stworzony obiekt nie posiada możliwości definiowania wewnętrznych pierścieni.

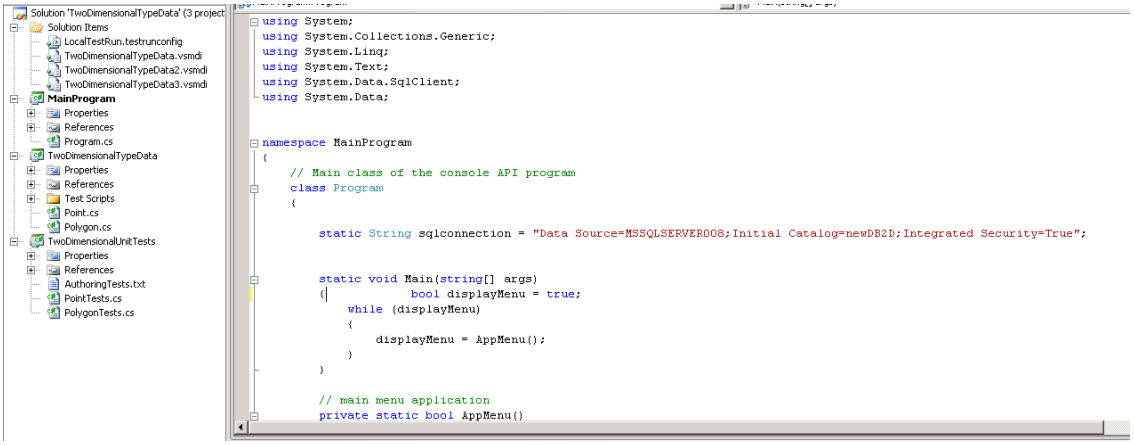
7. Uruchomienie.

Projekt został przygotowany w języku C# przy użyciu Microsoft Visual Studio 2008 (.NET framework 3.5).

- Utworzenie bazy danych. Należy uruchomić skrypt zamieszczony w pliku SQLScripts/ CreateDataBase2D.sql.
- Otworzenie projektu Visual Studio. Należ otworzyć plik TwoDimensionalTypeData.sln z folderu TwoDimensionalTypeData.
- Wybranie bazy do deployu. Klikamy prawym przyciskiem myszy na TwoDimensionalTypeData w Solution Explorer, wybieramy Properties i wybieramy odpowiednią bazę danych



- Uruchamiamy opcję Build, a następnie Deploy UDT.
- Tworzymy tabele, pomocnicze procedury oraz opcjonalnie insert przykładowych wartości. Uruchamiamy plik SQLScripts/PreparationDataBase2d.sql.
- Przechodzimy do Visual Studio i otwieramy plik Program.cs znajdujący się w podprojekcie MainProgram. Należy zmienić wartość obiektu sqlconnection na odpowiedni connection string do bazy danych.



- W celu uruchomienia aplikacji konsolowej przechodzimy do Debug -> Start Without Debugging.
- W celu uruchomienia testów należy zmienić wartości obiektów sqlconnection na odpowiednie wartość connection string w plikach **PointTests.cs** oraz **PolygonTests.cs**. Następnie uruchamiamy test wybierając opcję **Test -> Run -> All Tests in Solution**.

8. Podsumowanie i wnioski.

Przy pomocy UDT w łatwy i intuicyjny sposób można tworzyć rozbudowane typy danych. Ciekawą funkcjonalnością typów definiowanych przez użytkownika jest możliwość zdefiniowania metod i możliwość komunikacji z innymi obiektami. Daje to duże możliwości bazodanowe.

9. Literatura.

- https://newton.fis.agh.edu.pl/~antek/index.php?sub=db2_doc
- https://en.wikipedia.org/wiki/Shoelace_formula
- https://en.wikipedia.org/wiki/Point_in_polygon
- http://www.geomalgorithms.com/code.html
- https://www.geeksforgeeks.org/area-of-a-polygon-with-given-n-ordered-vertices/
- https://algorithmtutor.com/Computational-Geometry/Area-of-a-polygon-given-a-set-of-points/
- https://stackoverflow.com/