Projekt Zespołowy

Moduł generowania pojazdów i map
z plików graficznych

Wojciech Kowalik, Konrad Miśkiewicz, Mateusz Pielat

25 listopada 2015

Spis treści

1 Streszczenie

Projekt ma na celu stworzenie aplikacji symulującej ruch pojazdu w labiryncie. Pojazd będzie omijał przeszkody poruszając się do przodu i wykonując skręty w miejscu. Aplikacja będzie podzielona na kilka podstawowych modułów, które razem umożliwią przygotowanie i przeprowadzenie symulacji.

Dokument opisuje technologie oraz rozwiązania zastosowane przy implementacji modułu generowania pojazdów i map z plików graficznych. Moduł ten umożliwia użytkownikowi załadowanie pliku graficznego w jednym z dozwolonych formatów, a następnie jego trasowanie. Użytkownik otrzymuje listę wygenerowanych wielokątów, z których następnie może stworzyć mapę lub pojazd. Aby możliwe było stworzenie mapy, lista wygenerowanych wielokątów nie może być pusta, natomiast stworzenie pojazdu będzie możliwe wtedy i tylko wtedy, gdy na tej liścia będzie znajdował się jeden wielokąt. Moduł umożliwia zapisanie pracy w plikach XML, które będą wykorzystywane przez inne moduły aplikacji.

2 Przygotowanie modułu

Moduł został zaprojektowany z wykorzystaniem technologii *WPF* i *MonoGame*. Ponownie wykorzystane zostały pliki XML zaprojektowane przy okazji tworzenia modułu edycji ręcznej map i pojazdów, które nie uległy żadnym modyfikacjom. Do przygotowania głównej funkcjonalności tego modułu – trasowania, wykorzystana została biblioteka *D3DPotrace*. Trasowanie zostało dokładniej opisane w dalszej części dokumentu.

2.1 Wymagania funkcjonalne modułu

- System powinien umożliwiac wczytanie pliku graficznego w jednym z dozwolonych formatów.
- System powinien umożliwiać dobór parametrów trasowania:
  - *area threshold*
  - *color threshold*

  (parametry te zostały opisane w dalszej części dokumentacji)
- System powinien wyświetlać użytkownikowi wynik trasowania dla aktualnie dobranych parametrów.
- System powinien umożliwiać stworzenie mapy z wygenerowanych kształtów (warunkiem umożliwiającym stworzenie mapy jest liczba wygenerowanych wielokątów większa od 0).
- System powinien umożliwiać stworzenie pojazdu z wygenerowanego kształtu (warunkiem umożliwiającym stworzenie pojazdu jest liczba wygenerowanych wielokątów równa 1 - pojazd nie może składać się z dwóch oddzielnych części).
- System powinien umożliwiać zapisanie wygenerowanych map/pojazdów do plików wektorowych.

2.2 Przykłady reprezentacji mapy i pojazdu w XML

Moduł generowania pojazdów i map z plików graficznych, podobnie jak moduł ich ręcznej edycji wymagał użycia plików XML opisujących odpowiednio mapę i pojazd. W tym etapie wektorowa reprezentacja pojazdu i mapy nie uległa zmianie, jednak dla porządku ponownie zaprezentowane zostały przykładowe pliki XML.

2.2.1 Reprezentacja mapy

```
<vmd xmlns="pl.pw.mini.KowMisPie.SRL" width="512" height="512">
	<polygon>
		<point x = "160" y="119" />
		<point x = "58" y="284" />
		<point x = "357" y="275" />
	</polygon>
	<polygon>
		<point x = "393" y="32" />
		<point x = "390" y="398" />
		<point x = "457" y="400" />
	</polygon>
	<polygon>
		<point x="30" y="316" />
		<point x="27" y="60" />
		<point x="357" y="60" />
		<point x="362" y="23" />
		<point x="466" y="21" />
	</polygon>
</vmd>
```

2.2.2 Reprezentacja pojazdu

```
<vvd xmlns="pl.pw.mini.KowMisPie.SRL">
	<orientation>
		<point x="189" y="144" />
		<angle>1.9091524329963763</angle>
	</orientation>
	<polygon>
		<point x="112" y="93" />
		<point x="359" y="95" />
		<point x="433" y="165" />
		<point x="371" y="222" />
		<point x="111" y="222" />
	</polygon>
</vvd>
```

2.3 Dozwolone formaty plików graficznych

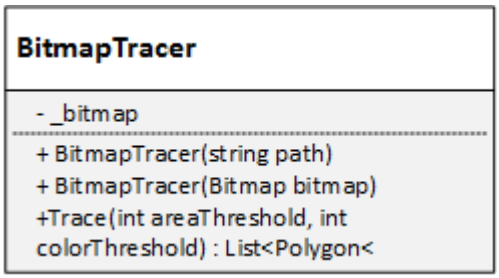
Aplikacja umożliwia trasowanie plików graficznych znajdujących się w jednym z poniższych formatów:

- bmp
- png
- jpg
- jpeg
- gif

Uwaga: Dla formatu gif plik powinien być statycznym obrazem. Dla animacji znajdującej się w tym formacie poprawne działanie modułu trasowania nie jest gwarantowane.

3 Zmiany w architekturze systemu

W porównaniu z poprzednią wersją aplikacji, architektura nie uległa znaczącym zmianom. Najważniejszą klasą, która pojawiła się w systemie jest klasa *BitmapTracer*. Poniżej przedstawiono jej strukturę oraz opis pól i metod.



Schemat 1. Klasa BitmapTracer

Powyższa klasa zawiera jedno pole prywatne - *_bitmap*, które jest polem oznaczonym modyfikatorem *readonly*. Jest ono inicjalizowane w konstruktorze klasy. Pole to przechowuje obiekt typu *Bitmap*, który będzie podlegał trasowaniu.

Poza dwoma konstruktorarmi, jedyną i kluczową metodą klasy *BitmapTracer* jest metoda *Trace(int, int)*. Jako parametry przyjmuje ona *area threshold* i *color threshold*, natomiast jako wynik swojej pracy zwraca nam listę rozpoznanych wielokątów.

4 Trasowanie pliku graficznego

4.1 Biblioteka

Trasowanie odbywa się przy pomocy wrappera zewnętrznej biblioteki *D3DPotrace* będącej dotnetowym portem biblioteki *Potrace*. Jest to jedna z dwóch (obok *AutoTrace*) popularnych bibliotek do trasowania obrazów rastrowych o otwartym kodzie.

Działanie samej biblioteki oparte jest na koncepcji hirarchizacji trasowanych obszarów. Przykładowo: umieszczona na białym tle czarna, wypełniona w środku plama jest obszarem traktowanym jako (zamknięty) wielokąt. Jeżeli na takiej czarnej plamie umieścimy całkowicie się w niej zawierającą plamę białą (tworząc czarny obwarzanek), to zostanie ona uznana za potomka plamy czarnej. Jeszcze mniejsza plama czarna zawarta w białej będzie potomkiem białej i prapotomkiem pierwszej czarnej plamy itd. Ponieważ wielokąty zawarte całkowicie w innych wielokątach nie mają żadnego znaczenia (pojazd omijając przeszkodę i tak bierze pod uwagę tylko jej krawędź, a nie jak jest w środku "podziurawiona"), dzięki takiej hierarchizacji danych możemy w łatwy sposób odrzucać wszystkie nieistotne dla obliczenia ścieżki kształty.

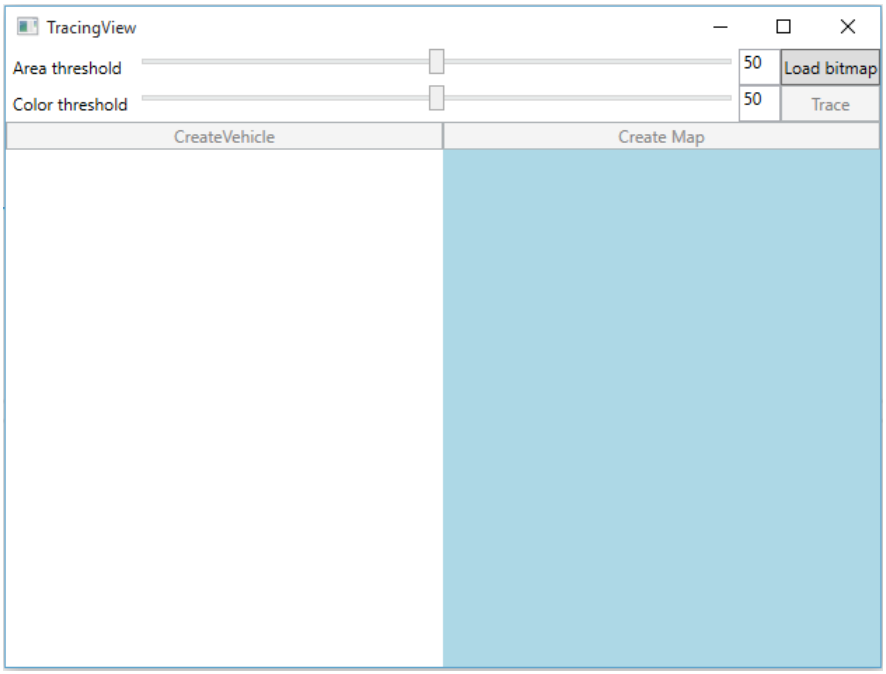
4.2 Wrapper

Poza odrzucaniem niepotrzebnych figur wrapper ułatwia korzystanie z biblioteki upubliczniając jedynie te parametry, których kastomizacja ma sens. Przykładowo, wrapper ma zakodowany na stałe współczynnik określający w jakim stopniu krawędzie trasowane są do linii beziera zamiast linii prostych. De facto dba o to, żeby krawędzie wyjściowe były wyłącznie odcinkami prostymi.

Wrapper upublicznia dwa parametry:
- Color threshold - procentowa wartość od 0 do 100. Określa stopień w jakim muszą różnić się dwa różnokolorowe fragmenty obrazka, aby zostały zinterpretowane jako dwa różne wielokąty. Przykładowo: mała wartość sprawi, że dwa nachodzące na siebie kwadraty - jeden czerwony, drugi czerwony z niewielką domieszką żółtego - zostaną przetrasowane do dwóch odrębnych obiektów. Duża wartość sprawi, że algorytm zwróci dla tego samego wejścia jeden kształt o polu będącym sumą pól kwadratów - chyba, że kwadraty będą miały zupełnie inne kolory (np. ciemny czerwony i jasny niebieski).
- Area threshold - minimalnym pole (de facto ilość pikseli) jakie musi mieć figura, aby nie była zignorowana przez algorytm trasujący. Wartość równa 0 sprawia, że nawet najmniejsze elementy na bitmapie zostaną potraktowane jak wielokąty. Ten parametr może być szczególnie przydatny w celu pozbycia się artefaktów generowanych przez aparat, gdy obrazek wejściowy jest zdjęciem.

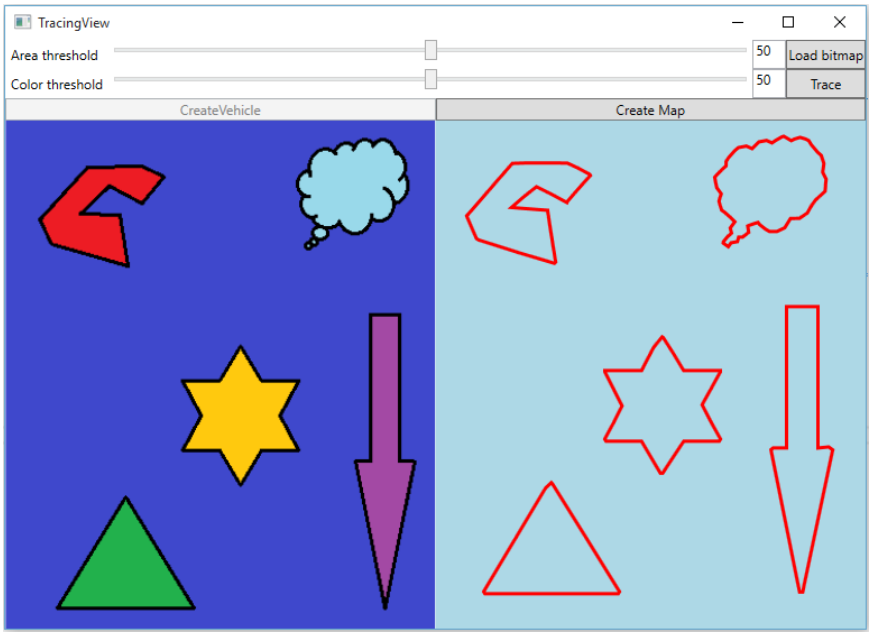
5 Działanie modułu

Użytkownik po wybraniu z menu głównego opcji „Wczytaj plik graficzny” zostaje przeniesiony do odpowiedniego widoku, w którym będzie mógł zdefiniować parametry trasowania a następnie je zrealizować. Bezpośrednio po wejściu do widoku modułu jedynym aktywnym przyciskiem jest przycisk "Wczytaj obraz", po wciśnięciu którego nastąpi otwarcie okna dialogowego i użytkownik będzie miał możliwość pliku graficznego który ma zostać przetworzony. Ponadto podczas pracy z modułem trasowania, cały czas aktywne pozostają suwaki umożliwiające zmianę parametrów *area threshold* i *color threshold*.



Rysunek 1. Moduł trasowania

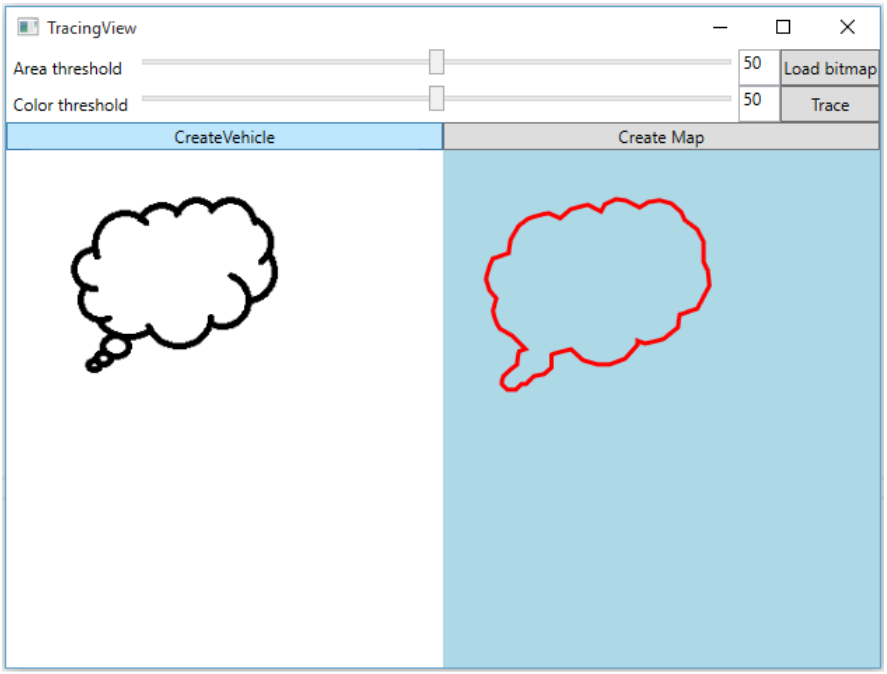
Po wyborze i akceptacji obrazka wejściowego odblokowany zostanie przycisk "Trasuj", po wciśnięciu którego użytkownik będzie mógł zobaczyć wygenerowane kształty. Przykładowe działanie aplikacji zaprezentowano poniżej.



Rysunek 2. Przykład trasowania 1

Jak łatwo zauważyć na powyższym obrazku, aktywowany został przycisk „Utwórz mapę", natomiast przycisk „Utwórz pojazd" pozostał nadal nieaktywny. Stało się tak dlatego, że wygenerowanych zostało kilka wielokątów. Sytuacja ta jest prawidłowa dla obiektu mapy, gdyż może się ona składać z dowolnej liczby wielokątów będących przeszkodami. Jeśli chodzi zaś o pojazdy, to oczekujemy, że pojazd składa się tylko z jednego wielokąta, więc w sytuacji, gdy wygenerowanych kształtów jest więcej stworzenie pojazdu nie będzie możliwe.

Czasami jednak trasowanie obrazków może dawać zaskakujące efekty i w sytuacji, gdy intuicja podpowiada nam, że powinno powstać kilka wielokątów, powstanie tylko jeden. Wszystko to zależy oczywiście również od odpowiedniego doboru parametrów trasowania. Poniższy zaprezentowano sytuację, w której z kilku zamkniętych kształtów na obrazku wejściowym powstaje jeden wielokąt.



Rysunek 3. Przykład trasowania 2

6 Testy

Dla celów testowych aplikacji napisanych zostało kilka testów sprawdzających poprawność zaimplementowanych metod. Oprócz testowania małych i pomocniczych metod, jak np. metody implementujące proste algorytmy geometryczne, została również przetestowana funkcja trasująca obrazy. Wyniki testów są pozytywne – funkcja trasowania dla różnych obrazów testowych i przy założonych wartościach progowania wygenerowała spodziewane listy wielokątów. Jedynym problemem okazało się generowanie zbyt dużej liczby zbędnych punktów (które leżą na krawędzi wielokąta) dla niektórych przypadków, jednak problem ten zostanie rozwiązany w dalszej części prac nad aplikacją.