Projekt Zespołowy

Moduł tworzenia pojazdów i map

Wojciech Kowalik, Konrad Miśkiewicz, Mateusz Pielat

16 listopada 2015

1 Opis projektu

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

Pierwszy z modułów będzie umożliwiać wczytywanie i przetwarzanie plików graficznych przedstawiających proste wielokąty. Plik taki będzie odpowiednio konwertowany do postaci wektorowej, dzięki czemu możliwe będzie wykorzystanie go w symulacji jako pojazd lub przeszkoda do ominięcia. W przypadku pojazdu, użytkownik będzie musiał dodatkowo zdefiniować kierunek i zwrot wielokąta.

Kolejny moduł będzie udostępniać narzędzie do tworzenia pojazdów i map kursorem bezpośrednio w interfejsie aplikacji. Będzie się to odbywać poprzez dodawanie kolejnych punktów wielokąta kursorem. Podobnie jak wyżej, w przypadku pojazdu konieczne będzie określenie jego przodu. Tworzenie mapy polegać będzie natomiast na stworzeniu kilku takich wielokątów na określonym obszarze.

Po wczytaniu przygotowanego pojazdu i mapy użytkownik będzie mógł zdefiniować punkt startowy i końcowy dla trasy na której chciałby zasymulować ruch pojazdu. Wykorzystywany tutaj moduł symulowania ruchu obliczy najbardziej optymalną drogę lub stwierdzi, że dla danych ustawień takowa nie istnieje. Już po rozpoczęciu symulacji użytkownik będzie mógł ją pauzować, zmieniać prędkość animacji oraz swobodnie śledzić jej przebieg przy pomocy suwaka. Sam algorytm będzie oparty na przeszukiwaniu grafów, zaś wizualizacją w 2D zajmie się jeden z dostępnych silników graficznych.

Ponadto, użytkownik będzie miał możliwość zapisania tak wygenerowanej symulacji w celu jej szybkiego odtworzenia w przyszłości.

2 Słownik pojęć

```
<vmd xmlns="pl.pw.mini.KowMisPie.SRL">
  <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" />
    <point x="458" y="33" />
  </polygon>
  <polygon>
    <point x="233" y="327" />
    <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" />
    <point x="467" y="7" />
    <point x="16" y="42" />
    <point x="15" y="334" />
    <point x="258" y="406" />
    <point x="315" y="358" />
  </polygon>
</vmd>
```

```
<vvd xmlns="pl.pw.mini.KowMisPie.SRL">
  <orientation>
    <point x="264" y="158" />
    <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>
```

3 User stories

Poniższe scenariusze opisują możliwe przypadki użycia aplikacji z punktu widzenia użytkownika. Każdy scenariusz dotyczy poprawnego wykorzystania danego modułu.

3.0.1 Scenariusz ręcznego tworzenia pojazdu:

1. Użytkownik otwiera okno tworzenia pojazdu.
2. Użytkownik wyznacza punkty stanowiące wierzchołki wielokąta opisującego pojazd.
3. Użytkownik wyznacza oś i zwrot pojazdu.
4. Użytkownik zatwierdza projekt, a system zapisuje go do postaci wektorowej.

3.0.2 Scenariusz ręcznego tworzenia mapy:

1. Użytkownik otwiera okno tworzenia mapy.
2. Użytkownik wyznacza punkty stanowiące wierzchołki przeszkód mapy.
3. Użytkownik zatwierdza projekt, a system zapisuje go do postaci wektorowej.

3.0.3 Scenariusz tworzenia pojazdu z pliku graficznego:

1. Użytkownik wczytuje plik graficzny.
2. Użytkownik dobiera parametry trasowania.
3. System trasuje plik graficzny do postaci wektorowej.
4. Użytkownik wyznacza oś i zwrot pojazdu.
5. Użytkownik zatwierdza projekt, a system zapisuje go do postaci wektorowej.

3.0.4 Scenariusz tworzenia mapy z pliku graficznego:

1. Użytkownik wczytuje plik graficzny.
2. Użytkownik dobiera parametry trasowania.
3. System trasuje plik graficzny do postaci wektorowej.
4. Użytkownik zatwierdza projekt, a system zapisuje go do postaci wektorowej.

3.0.5 Scenariusz symulacji algorytmu:

1. Użytkownik wczytuje pojazd.
2. Użytkownik wczytuje mapę.
3. Użytkownik wskazuje początek i koniec trasy.
4. Użytkownik rozpoczyna symulację.
5. System przedstawia wizualizację, jeżeli takowa jest możliwa.

4 Wymagania funkcjonalne

4.0.1 Moduł wizualizacji

- System powinien umożliwiać wczytywanie uprzednio zapisanych w odpowiednim formacie map i pojazdów.
- System powinien umożliwiać ustawienie początku i końca trasy pojazdu.
- System powinien umożliwiać ustawienie prędkości animacji ruchu pojazdu.
- System powinien umożliwiać pauzowanie trwającej wizualizacji.
- System powinien umożliwiać przejście suwakiem do dowolnego momentu symulacji.
- System powinien umożliwiać zapisywanie wygenerowanej symulacji.

4.0.2 Moduł rysowania pojazdów i map

- System powinien umożliwiać rysowanie wielokątów reprezentujących pojazdy lub przeszkody omijane przez pojazd.
- System powinien zamykać wielokąt, jeżeli użytkownik nie zamknie go sam.
- System powinien umożliwiać zapisywanie narysowanych pojazdów/map do postaci wektorowej.

4.0.3 Moduł generowania pojazdów i map z plików graficznych

- System powinien umożliwiać wczytanie pliku graficznego w jednym z dozwolonych formatów.
- System powinien umożliwiać dobór parametrów trasowania.
- System powinien wyświetlać użytkownikowi wynik trasowania dla aktualnie dobranych parametrów.
- System powinien umożliwiać zapisywanie wygenerowanych pojazdów/map do postaci wektorowej.

4.0.4 Moduł algorytmu

- System powinien znajdować ścieżkę między punktem startowym a punktem końcowym dla danego pojazdu z uwzględnieniem przeszkód.
- System powinien eksportować wynik obliczeń do listy rozkazów.

5 Wymagania niefunkcjonalne

- Aplikacja powinna poprawnie działać na systemach operacyjnych z rodziny *Windows*.
- System powinien poinformować użytkownika, gdy przeprowadzenie symulacji jest niemożliwe.
- Dozwolone formaty dla plików graficznych: **bmp**, **jpeg**, **png**, **gif**.
- Rysowanie pojazdów i map nie powinno odbywać się poprzez rysowanie ciągłe, a raczej poprzez dodawanie kolejnych punktów do wielokątów.
- Podczas rysowania wielokątów ich krawędzie nie mogą się przecinać.
- System powinien dać się łatwo tłumaczyć na różne języki:
  - Powinien mieć wbudowany język polski i angielski.
  - Dodanie nowego języka powinno ograniczać się do zmian w pliku konfiguracyjnym.

6 Metodologia pracy

Aplikacja będzie tworzona zgodnie z modelem przyrostowym (ang. *incremental development*). Zakłada on realizację jedynie pewnej części systemu w każdym kolejnym 'cyklu pracy. Poszczególne porcje funkcjonalności powinny być spójne, aby możliwe było przetestowanie i dostarczenie ich w wersji finalnej klientowi.

6.1 Uzasadnienie wyboru

Preferencja modelu przyrostowego nad modelem kaskadowym wynika ze struktury naszej aplikacji. Można ją podzielić na wiele odrębnych modułów działających (do pewnego stopnia) oddzielnie i niezależnie od siebie. Ponadto, ta metodologia zakłada brak konieczności definiowania z góry szczegółowych wymagań poszczególnych części systemu, dzięki czemu proces tworzenia jest bardziej elastyczny niż w przypadku modelu kaskadowego.

7 Harmonogram pracy

Architektura systemu		
WK	repozytorium i struktura katalogów	17.11.15
KM	przygotowanie diagramu klas	
ALL	zdefiniowanie reprezentacji pojazdu i mapy	
ALL	zdefiniowanie reprezentacji rozkazów ruchu pojazdu	
Interfejs graficzny		
MP	przygotowanie mock-up poszczególnych okien interfejsu	
KM	implementacja interakcji pomiędzy oknami	
Tworzenie pojazdów i map		
KM	serializacja i deserializacja do XML	
WK	tworzenie pojazdów i przeszkód przy pomocy kursora	
MP	selekcja trasowanych grafik rastrowych pod względem rozszerzeń i treści	01.12.15
MP	trasowanie z użyciem wybranej biblioteki; wyświetlanie efektu użytkownikowi dla odpowiednio dobranych parametrów trasowania	
Mock-up algorytmu		15.12.15
KM	losowe generowanie tras	
KM	eksport trasy do listy rozkazów	
Wizualizacja		
WK	tworzenie przeszkód i pojazdu na mapie	
WK	ustawianie początku i końca trasy	
MP	animacja ruchu na podstawie rozkazów	
KM	selekcja trasowanych grafik rastrowych pod względem rozszerzeń i treści	
MP	implementacja suwaka osi czasu wizualizacji	
Algorytm		12.01.16
ALL	opracowanie koncepcji algorytmu wyszukiwania drogi	
ALL	implementacja algorytmu wyszukiwania drogi	
ALL	eksport i mapowanie wyników obliczeń do listy rozkazów	

ALL - zadania przydzielone dla wszystkich
WK - zadania przydzielone dla Wojciecha Kowalika
KM - zadania przydzielone dla Konrada Miśkiewicza
MP - zadania przydzielone dla Mateusza Pielata

8 Kamienie milowe

Każdy kamień milowy stanowi oddzielny, funkcjonalny moduł systemu. Wszystkie z nich kończą się testami i dokumentacją.

1. **Moduł tworzenia map i pojazdów przez użytkownika**

   Zostanie stworzony interfejs użytkownika oraz możliwe będzie ręczne tworzenie map i pojazdów z wykorzystaniem wbudowanego edytora i zapisywanie ich do formatu wektorowego

2. **Moduł wczytywania map i pojazdów z plików graficznych**

   Możliwe będzie wczytywanie pojazdów i map z plików graficznych oraz eksportowanie ich do formatu wektorowego

3. **Moduł wizualizacji ruchu**

   Użytkownik może wybierać pojazd, mapę oraz zaznaczać początek i koniec trasy. Mock-up algorytmu wygeneruje losową trasę dla pojazdu, która zostanie przedstawiona użytkownikowi. Użytkownik będzie miał możliwość ustawienia prędkości symulacji, pauzy oraz przejścia w czasie suwakiem.

4. **Moduł algorytmu wyszukiwania drogi**

   Zaimplementowany zostanie algorytm wyznaczania drogi, który zastąpi wcześniejszy mock-up. Wszystkie moduły aplikacji zostaną ze sobą zintegrowane.