**(Średnio)zaawansowane programowanie w C++ (ZPR2012L)**

**Dokumentacja końcowa**

1. **Temat:**

Implementacja prostego symulatora ruchu miejskiego.

1. **Opis modelu:**

Podczas symulacji poruszają się obiekty różnego typu: samochody małe, duże oraz piesi. W różnych miejscach ulic rozmieszczone są "inteligentne" kamery – każda posiada swoje parametry: dokładność, kąt widzenia, kierunek obserwacji. Kamery są próbkowane co 1 sek. Jeżeli coś zauważy – generuje obserwacje i podaje współrzędne zaobserwowanych obiektów zmierzone zgodnie ze swoimi parametrami oraz w celu weryfikacji działania symulatora, rzeczywiste współrzędne wraz z czasem logowania.

1. **Problemy napotkane podczas tworzenia aplikacji:**

Podczas tworzenia aplikacji, głównym problemem były ograniczenia czasowe. Z tego powodu nie udało nam się zrealizować funkcjonalności dodatkowej projektu. Jednym z problemów który ukazał się pod koniec projektu, gdy symulacja była skończona, wybór biblioteki do stworzenia prostego GUI w postaci biblioteki Allegro dołożył sporą ilość pracy poprzez brak prostej obsługi zdarzeń. Jednakże została wybrana tylko do celów wizualizacyjnych co nie było problemem.

1. **Struktura aplikacji:**

Aplikacja została podzielona na trzy główne części: Model, Widok i Kontroler (MVC).

Model odpowiada za wszelkie obliczenia związane z przebiegiem symulacji. Kontroler zajmuje się czytaniem danych z plików konfiguracyjnych oraz zarządzaniem odświeżania modelu oraz widoku poprzez kontrolujący to Timer.

Widok odrysowuje przebieg symulacji w postaci prostych obiektów geometrycznych:

* Samochody - prostokąty różniące się kolorem w zależności od rozmiaru,
* Piesi - trójkąty,
* Kamery - koła + wycinek kół pokazujący ich zasięg.

1. **Zrealizowana funkcjonalność:**
2. Dane pobierane są z plików XML, których szablony dołączone są do projektu w katalogu *template\_xml/* Dane domyślnie szukane są w katalogu *xml\_data/*
   * Mapa ulic w postaci węzłów grafu i krawędzi pomiędzy nimi (graf nieskierowany). *[streets.xml]*
   * Obiekty w poruszające się w symulacji, ich parametry i trasy. *[objects.xml]*
   * Kamery i ich parametry. *[dispatcher.xml]*
3. Samochody poruszają się po gładkich trasach, nie skręcają 'w miejscu'. Przyśpieszają i zwalniają przed zakrętami.
4. Piesi poruszają się po prostych trasach, skręcają 'w miejscu'.
5. Zakręty na trasach samochodów wyznaczane są automatycznie przez wyliczenie odpowiednich krzywych beziera zależnych wielkością 'promienia' od stałej zdefiniowanej w projekcie.
6. Kamery logują obserwacje co 1 sekundę do pliku o nazwie złożonej z numeru (timestamp) z rozszerzeniem *.txt*, w chwili pierwszego zauważonego obiektu.
7. Wizualizacja graficzna w postaci prostych obiektów geometrycznych oraz proste menu pozwalające wystartować, pauzować, restartować, zapętlić lub zakończyć symulację. Zapętlenie polega na tym, że gdy obiekt dojedzie do końca swojej trasy - zaczyna ją przemierzać od początku.
8. Pliki z danymi domyślnie szukane są w katalogu *xml\_data/* o nazwach podanych w (pkt. a). W przypadku braku któregokolwiek z plików, należy wskazać prawidłową, względną do uruchamianej aplikacji, ścieżkę do pliku lub zakończyć działanie programu.
9. Ze względu na ograniczenia czasowe nie zostały zrealizowane funkcje dodatkowe z dokumentacji wstępnej.
10. **Konwencja jednostek:**
    1. Początek układu współrzędnych (0, 0) w lewym górnym rogu
    2. Jednostki: 1.0 = 1 metr
    3. Widoczny obszar symulacji to 200 x 200 [m]
    4. Kąty podawane w stopniach, w aplikacji zamieniane na radiany
11. **Parametry obiektów:**
    1. Kamery:
       1. ID - liczba całkowita
       2. Położenie - współrzędne x, y
       3. Kierunek patrzenia - kąt w stopniach, [ 0=E, 90=S, 180=W, 270=N ]
       4. Kąt widzenia - całkowity kąt widzenia w stopniach
       5. Zasięg widzenia - liczba metrów
       6. Precyzja - podawana w %
    2. Obiekty:

*Wspólne parametry:*

* + 1. Typ - samochód / pieszy
    2. ID - rejestracja samochodu / nazwa pieszego
    3. Prędkość maksymalna - prędkość maksymalna dla samochodu / prędkość pieszego - obie podawane w [ km / h ]

*Dodatkowe parametry samochodów:*

* + 1. Przyśpieszenie - podawane w [ m/s ]
    2. Masa - podawana w [ kg ]
    3. Rozmiar - mały / duży

*Trasy obiektów:*

* + 1. Lista kolejnych punktów trasy - współrzędne x, y
  1. Mapy ulic:

*Lista wierzchołków:*

* + 1. ID - numer wierzchołka
    2. Położenie - współrzędne x, y

*Lista krawędzi:*

* + 1. Łączenie - numery wierzchołków pomiędzy którymi występuje krawędź

1. **Wykorzystanie dodatkowych bibliotek:**
   1. Allegro 5 - wizualizacja symulacji
   2. Ważniejsze biblioteki z boost:
      1. Boost.Thread - użyta to wielowątkowości aplikacji
      2. Boost.Chrono - użyta do liczenia czasu
      3. Boost.Bind - w celu ograniczenia ilości kodu
      4. itp. mniej istotne.
   3. Kontenery i algorytmy z biblioteki standardowej STL
2. **Użyte wzorce projektowe:**

Architektura projektowanego symulatora okazała się na tyle prosta, że wystarczyło użycie prostych rozwiązań. Użyte zostały tylko przestawione poniżej wzorce w mniejszym lub większym udziale:

* + - Obserwator - ????!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    - Singleton - logger - logowanie do pliku danych pochodzących z symulacji
    - Wizytator - tu ty coś mówiłeś!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    - Komenda - boost.bind - anonimowe funktory

1. **Kompilacja i uruchamianie programu:**
   1. Windows, Visual Studio 2010: (instrukcja w pkt. 15)
      1. Wymaga linkowania bibliotek boost do projektu
      2. Wymaga dołączenia biblioteki allegro
      3. Wskazanie katalogów z plikami nagłówkowymi
   2. Unix, g++: (instrukcja instalacji bibliotek w pkt. 14)
      1. Polecenie make - kompilacja programu
      2. make clean - usunięcie plików pośrednich
      3. make test - Czy w ogóle będą te testy ?!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
2. **Testy ?!!!!!!!!!!!!!!!!!!!!!!!!!!!:**

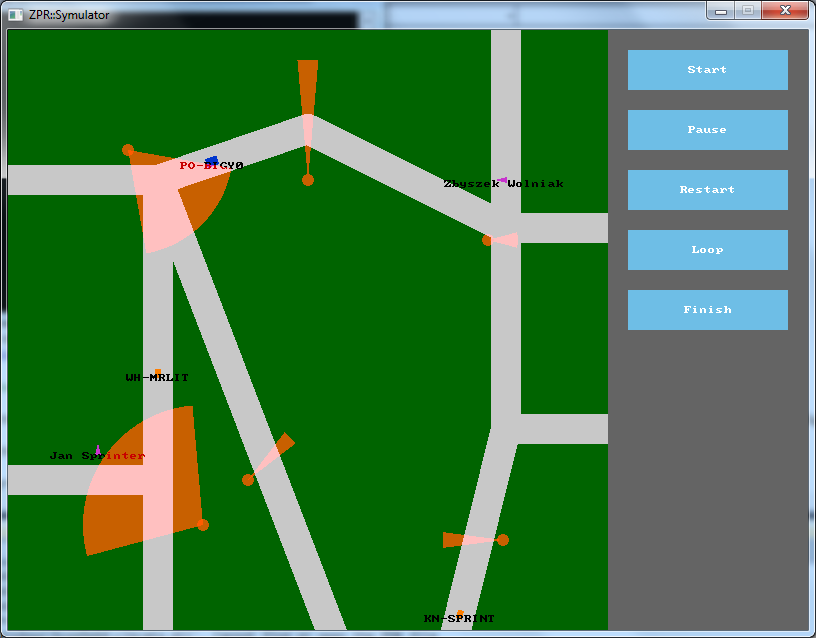
Jakiś tekst o testach!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

1. **Podsumowanie:**

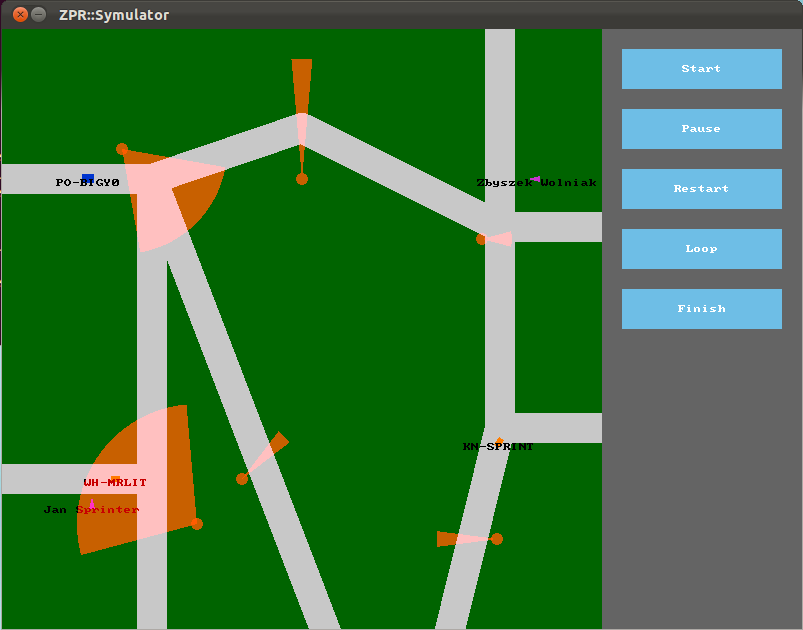
Podczas pracy nad projektem natknęliśmy się na niejeden problem, jednakże dzięki dostępnym rozbudowanym bibliotekom oraz przykładom z wykładów, które były bardzo pomocne, rozwiązania niekiedy stawały się trywialne i praca szybko posuwała się do przodu. Znajomość niektórych wzorców projektowych pomogła na etapie projektowania, aczkolwiek w naszej aplikacji znaleźliśmy zastosowanie tylko dla kilku z wielu. Nie chcieliśmy też na siłę używać wzorców nie pasujących do projektu, gdyż wiele rzeczy rozwiązaliśmy prostymi technikami programowania, nie utrudniając sobie tym samym.

1. **Przykładowy screen aplikacji:**

Windows:



Unix:



1. **Instrukcja instalacji biblioteki allegro i boost pod Ubuntu:**

**ALLEGRO:**

sudo apt-get install subversion

sudo apt-get install cmake

sudo apt-get install make

mkdir allegro-svn

cd allegro-svn

svn co https://alleg.svn.sourceforge.net/svnroot/alleg/allegro/branches/5.1

cd 5.1

mkdir Build

cd Build

cmake .. -DGRADE\_DEBUG=on -DSHARED=off

make

sudo make install

**BOOST:**

Download boost\_1\_49\_0.tar.bz2.

http://sourceforge.net/projects/boost/files/boost/1.49.0/boost\_1\_49\_0.tar.bz2/download

W katalogu gdzie chcemy wstawić bibliotekę, wykonać: (zazwyaczaj /usr/local/boost\_1\_49\_0)

sudo tar --bzip2 -xf /sciezka/wybrana/wyzej/do/boost\_1\_49\_0.tar.bz2

cd /usr/local/boost\_1\_49\_0

sudo ./bootstrap.sh --prefix=/usr/local/

sudo ./b2 install

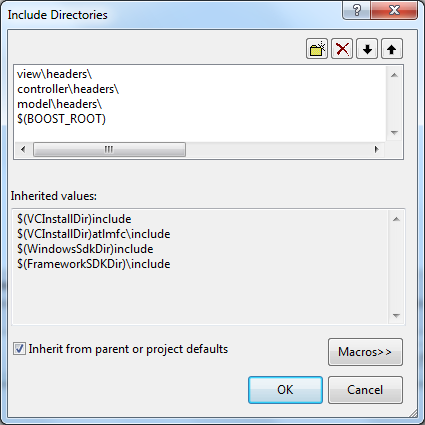
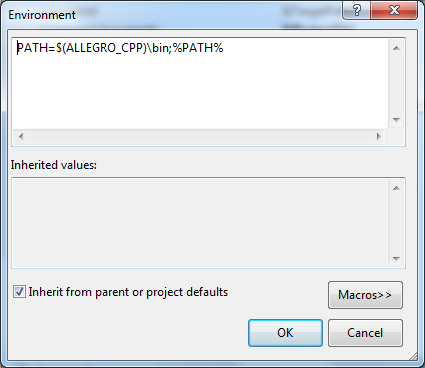
1. **Instrukcja instalacji w środowisku Windows (Visual Studio 2010):**

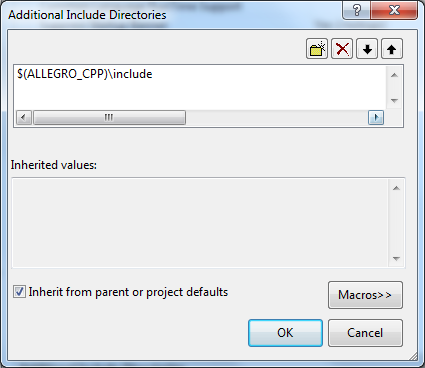
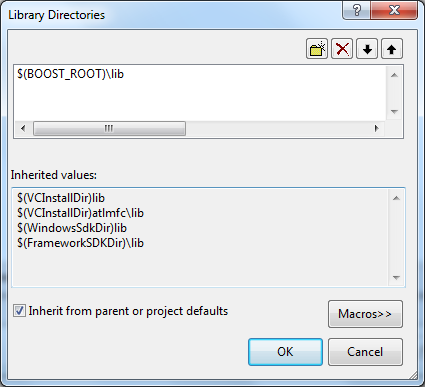
Biblioteka: http://static.allegro.cc/file/library/allegro/5.0.6/allegro-5.0.6-msvc-10.0.zip

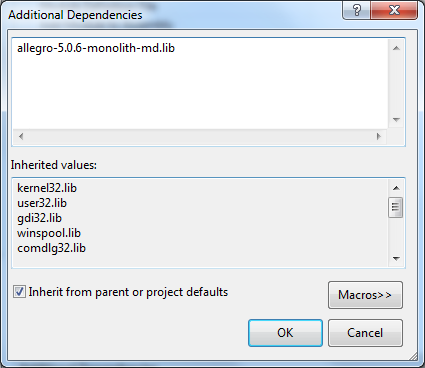
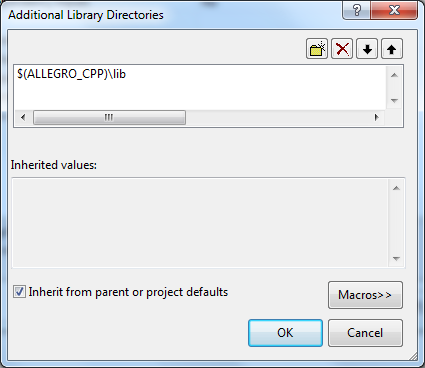
Instrukcja instalacji:

http://wiki.allegro.cc/index.php?title=Windows,\_Visual\_Studio\_2010\_and\_Allegro\_5

+ lokalizacja plików nagłówkowych:







1. **!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! czy cos jescze ?!!?!?!**