Program analizujący pliki GPX

Opis programu:

Program analizuje plik GPX wskazany przez użytkownika. Analiza polega na:

- obliczeniu długości, czasu trasy zawartej w pliku,
- wyznaczeniu średniej prędkości oraz minimalnej i maksymalnej wysokości,
- narysowaniu szkicu trasy i wykresu prędkości.

Dodatkowo użytkownik ma możliwość podzielenia trasy na odcinki ze względu na zmianę prędkości, czy też zmianę kierunku ścieżki.

Sposób użycia:

Uruchomienie programu odbywa się bez dodatkowych parametrów. Użytkownik za pomocą menu może wybrać plik GPX, który ma zostać poddany analizie. Po wyborze, w przypadku błędu otwarcia lub analizy pliku zostanie wyświetlony stosowny komunikat. Jeśli otwarcie pliku i jego analiza przebiegną pomyślnie, w głównym oknie programu zostanie wyświetlony szpic mapy i wykresu prędkości oraz wyżej wymienione statystyki. W sekcji “kryteria podziału”, użytkownik ma możliwość wyboru rodzaju podziału oraz podania jego parametrów. Parametr “minimalna zmiana” pozwala określić minimalny promień skrętu w przypadku podziału ze względu na zakręty oraz minimalną różnicę prędkości w przypadku podziału ze względu na prędkość, natomiast parametr “minimalny czas” określa minimalny czas odcinka. Podział następuje po kliknięciu przycisku “OK”. Klikając na statystki danego odcinka, zostanie podświetlona odpowiednia część szkicu mapy i wykresu.

Szczegóły techniczne:

Plik `ui.glade` musi się znajdować w katalogu z aplikacją.

Wykorzystane biblioteki niestandardowe:

- GTK 3.10.8 (w tym Cairo 1.13.1, GLib 2.40.2) (wymagane GTK 3.0)
- expat 2.10

Opcje kompilacji i linkowania:

- Kompilacja: `-xc -std=c99 -Wall -Wextra `pkg-config gtk+-3.0 --cflags``
- Linkowanie: `` `pkg-config gtk+-3.0 --libs` -lexpat ``

Wymagania:

- Windows XP lub nowszy, Linux

Platforma testowa:

- ubuntu 14.04 LTS (64-bit)
- Intel® Core™ i3 CPU M 330
- Mobility Radeon 5650 HD
- 4 GB RAM DDR3

Opis implementacji:

Program został podzielony na 5 modułów:

- main
- gtk_gui
- structs
- xml
- analyzer

Moduł **main**:

Znajduje się w nim funkcja main, która inicjuję cały program.

Moduł **gtk_gui**:

Odpowiedzialny jest za prezentację wyników, obsługę zdarzeń, wywoływanie funkcji z innych modułów.
Główna funkcja `gtk_gui` inicjuje bibliotekę GTK oraz buduje interfejs użytkownika za pomocą instrukcji `gtk_builder_new_from_file("ui.glade")`. Następnie łączy widgety z odpowiednimi funkcjami. Rysowanie szkicu mapy oraz wykresu prędkości odbywa się za pomocą biblioteki Cairo.

Zmiana sposobu wyświetlania danych w TreeView (na podstawie http://scentric.net/tutorial/sec-treeview-col-celldatafunc.html):

```
static void time_cell_data_func(GtkTreeViewColumn *tree_column, GtkCellRenderer *cell,
                                GtkTreeModel *tree_model, GtkTreeIter *iter, gpointer data)
{
    double time;
    gtk_tree_model_get(tree_model, iter, TIME_COLUMN, &time, -1);
    char *str_time = time_to_string(time);
    g_object_set(cell, "text", str_time, NULL);
    g_free(str_time);
}
```

Moduł **structs**:

Zawiera trzy podstawowe struktury: `point`, `segment`, `seglist` oraz funkcje pomocnicze do ich obsługi.

- `point` - zawiera informacje o punkcie otrzymanym z parsera,
- `segment` - łączy punkty w odcinki oraz przechowuje informacje na temat odcinka (prędkość, długość)
- `seglist` - łączy segmenty w całość oraz zawiera podstawowe statystyki

Moduł **xml**:

Parsuje otrzymany plik przy pomocy biblioteki expat, jako wynik zwraca listę segmentów (typu `seglist`)

Moduł **analyzer**:

Odpowiedzialny jest za analizowanie punktów - przygotowuje je do wyświetlenia na mapie, segmentów - oblicza statystyki dla segmentów oraz dzielenie trasy na odcinki ze względu na prędkość lub zmianę kierunku.

Obliczanie odległości na podstawie współrzędnych geograficznych (wzorowane na http://stackoverflow.com/a/365853):

```
void analyze_segments(SegListPtr list_ptr)
{
    SegmentPtr tmp_seg = get_first_segment(list_ptr);
    while(tmp_seg != NULL) {
        double lat2 = get_latitude(get_end_point(tmp_seg));
        double lat1 = get_latitude(get_start_point(tmp_seg));
        double dlon = DEG_TO_RAD(get_longitude(get_end_point(tmp_seg)) -
get_longitude(get_start_point(tmp_seg)));
        double dlat = DEG_TO_RAD(lat2 - lat1);

        lat2 = DEG_TO_RAD(lat2);
        lat1 = DEG_TO_RAD(lat1);

        double a = square(sin(dlat/2))+square(sin(dlon/2))*cos(lat1)*cos(lat2);
        a = 2*atan2(sqrt(a), sqrt(1-a));
        double distance = R * a;
....
```

Wyznaczanie kierunku skrętu ścieżki (wzorowane na http://stackoverflow.com/a/27645131):

```
        v1.x = p2.x - p1.x; v1.y = p2.y - p1.y;
        v2.x = p3.x - p2.x; v2.y = p3.y - p2.y;
        cross_product = v1.x * v2.y - v1.y * v2.x;
```

Obliczanie promienia skrętu (http://matematyka.pisz.pl/strona/1630.html):

```
        RAD_TO_DEG(acos((v1.x*v2.x+v1.y*v2.y)/(vector_length(v1)*vector_length(v2))));
```