|  |
| --- |
| Gastaldo TS2020 Production |
| TS2020KMLtoCSV Converter |
| Progetto per applicazione in C# |

|  |
| --- |
| Daniele Gastaldo  29 settembre 2020 |

SOMMARIO

[1 Introduzione 2](#_Toc51753501)

[1.1 Nome progetto 2](#_Toc51753502)

[1.2 Obiettivi progetto 2](#_Toc51753503)

[1.3 Framework utilizzato 2](#_Toc51753504)

[1.4 Linguaggio utilizzato 2](#_Toc51753505)

[1.5 Tipo di app 2](#_Toc51753506)

[2 Struttura dei files kml 3](#_Toc51753507)

[2.1 Xml Namespaces 3](#_Toc51753508)

[2.2 Radice del file kml 3](#_Toc51753509)

[2.3 Marker puntuali 3](#_Toc51753510)

[2.4 Marker lineari 3](#_Toc51753511)

[2.5 Marker poligonali 4](#_Toc51753512)

# Introduzione

## Nome progetto

TS2020KMLtoCSVConverter

## Obiettivi progetto

Scrivere un programma informatico capace di trasformare uno o più files di markers geografici in formato kml (estensione del formato di files proprietario di Google e utilizzato per importare ed esportare i dati geografici da Google Maps e Google Earth Pro) in uno o più files in formato csv utilizzabili successivamente con il blueprint editor di Railworks Train Simulator 2020 (RWTS2020 o TS2020).

## Framework utilizzato

Versione 0: .Net Core Framework (perché in questo modo l’app può essere distribuita con poco sforzo su ambienti di diverse piattaforme).

## Linguaggio utilizzato

Versione 0: C# (C-sharp).

## Tipo di app

Versione 0.1: App console di .Net Core Framework (scritta in C#).

# Struttura dei files kml

In questo paragrafo sono riportati quindi gli appunti riguardanti la struttura dei files kml (così come vengono generati da Google Earth Pro). Tuttavia, ho riportato qui solo quelle parti di tale struttura che sono d’interesse per il progetto KMLToCSVConverter, tralasciando di analizzarla completamente. La motivazione di tale scelta è che un tipico file kml contiene molti elementi che, pur essendo necessari a Google Earth Pro (che li genera e deve essere in grado di interpretarli correttamente) non sono in alcun modo utili agli scopi che tale progetto si pone.

## Xml Namespaces

Lo schema di questa struttura sarà utilizzato per estrarre dal file in esame in valore del namespace “xlmns”, il namespace cui appartengono gli elementi di nostro successivo interesse.

1. RADICE: un singolo elemento di nome “kml”;
   1. ATTRIBUTO: un singolo namespace xml di nome “xmlns”.

## Radice del file kml

Questo schema di struttura contiene gli elementi dell’albero di file kml da percorrere, a partire dalla radice del documento kml, per arrivare al livello dei singoli placemark, il livello di nostro interesse.

1. RADICE: un singolo elemento di nome “Document”;
   1. ELEMENTO: un singolo elemento di nome “Folder”;
      1. ELEMENTO: uno o più elementi di nome “Placemark”.

## Marker puntuali

Lo schema seguente è quello di un marker puntuale (le puntine da disegno sulla mappa di Google Earth Pro).

1. RADICE: un singolo elemento di nome “Placemark”;
   1. ELEMENTO: un singolo elemento di nome “name”;
   2. ELEMENTO: un singolo elemento di nome “Point”;
      1. ELEMENTO: un singolo elemento di nome “coordinates”.

## Marker lineari

Di seguito riportiamo la struttura di dati per un marker lineare (le linee rette colorate della mappa di Google Earth Pro).

1. RADICE: un singolo elemento di nome “Placemark”;
   1. ELEMENTO: un singolo elemento di nome “name”;
   2. ELEMENTO: un singolo elemento di nome “LineString”;
      1. ELEMENTO: un singolo elemento di nome “coordinates”.

## Marker poligonali

Infine, riportiamo lo schema della struttura di un marker poligonale (le forme colorate della mappa di Google Earth Pro).

1. RADICE: un singolo elemento di nome “Placemark”;
   1. ELEMENTO: un singolo elemento di nome “name”;
   2. ELEMENTO: un singolo elemento di nome “Polygon”;
      1. ELEMENTO: un singolo elemento di nome “OuterBoundaryIs”;
         1. ELEMENTO: un singolo elemento di nome “LinearRing”;
            1. ELEMENTO: un singolo elemento di nome “coordinates”.

# **Analisi del codice (versione 1.0)**

## Parsing degli input forniti come argomenti dalla linea di comando

### Prerequisiti

L’unico prerequisitio di questo compito è la definizione di un entry point (cioè di un metodo main che definisce il punto d’inizio dell’esecuzione del programma ed è contenuto nella classe Program contenua nel namespace avente lo stesso nome dell’applicazione) per l’applicazione del tipo

static void Main (String[] args)

in modo che l’applicazione sia a conoscenza del fatto di dover acquisire gli input forniti dall’utente come argomenti dalla linea di comando.

L’elemento di indice 0 della lista di stringhe è il nome dell’applicazione mentre nel caso degli argomenti veri e propri, i parametri passati in input per gli argomenti che li richiedono andranno scritti dopo il rispettivo comando e separati da esso da un simbolo di uguale “=”. In questo modo, gli argomenti che richiedono un input potranno facilmente essere compresi dal programma e utilizzati successivamente.

### Argomenti utilizzati

Di seguito scriveremo l’elenco degli argomenti utilizzabili con questa versione dell’applicazione

* -D= (--pathforcsvfiles=): percorso di output dei files csv analizzati dal programma; se non diversamente specificato dall’utente, il valore di default deve corrispondere ad una cartella di nome “CSV” contenuta all’interno della cartella dove risiede l’eseguibile;
* -d= (--csvfilenameroot=): stringa in cui l’utente può passare all’applicazione una radice comune per i nomi dei files csv generati dal programma;
* -H, -h, -U, -u (--help, --usage): commando da utilizzare per far elencare all’applicazione le possibili opzioni da utilizzare per passare gli argomenti all’applicazione;
* -L= (--logfilepath=): stringa in cui può essere definito un percorso dove salvare il file di log del programma; normalmente, il file di log deve essere salvato nella stessa cartella dove risiede l’eseguibile, per cui il suo valore di default deve essere tale percorso;
* -l= (--logfilename=): stringa in cui l’utente può specificare un nome per il file di log del programma; il valore di default è “log.txt”;
* -O= (--pathtokmlfiles=): percorso di origine dei files kml da analizzare; se non diversamente specificato, il valore di default deve essere una cartella di nome “KML” contenuta nella cartella dove risiede l’eseguibile;
* -o= (--originfilenamefilter=): comando da utilizzare come filtro dei files kml di origine; il valore di default è “\*.kml”, in modo da selezionare qualsiasi file kml presente nel percorso di origine e nelle sue sottocartelle;
* -V= (--verbosity=): valore utilizzato per regolare la quantità di informazioni da salvare nel file di log; il valore di default è “low” (solo informazioni riguardanti il threading dell’applicazione), mentre gli altri valori sono “debug” (inserimento informazioni riguardanti i singoli placemarks analizzati), “benchmark” (dati sui tempi di analisi dei singoli placemarks) e “all” (tutte le informazioni possibili a spese della dimensione del file di log);
* -v (--version): comando privo di input utilizzato per far stampare all’applicazione nella console le informazioni sulla versione dell’applicazione utilizzata e le informazioni di contatto dell’autore.

Nel caso in cui un utente fornisca un’opzione non inclusa nel precedente elenco, la risposta del programma deve essere fornire all’utente tutte le informazioni necessarie riguardo la versione e l’uso del programma (analogamente a quanto farebbe come se si fossero dati i comandi -v e -u insieme e in successione).

### Implementazione

Affinché questo primo compito possa essere correttamente implementato, si è deciso di procedere all’implementazione del parsing degli argomenti direttamente all’interno dell’entry point dell’applicazione, dato che è il primo compito che deve essere portato a termine dall’applicazione prima di iniziare l’analisi. Tuttavia, per rendere più agevole la fornitura di dettagli e informazioni sia riguardo all’applicazione (nome, versione e informazioni di contatto dell’autore) sia riguardo alle opzioni utilizzabili con l’applicazione, abbiamo organizzato queste informazioni in 2 funzioni separate, che verranno richiamate dal metodo Main nel punto richiesto dalla funzione di parsing. Inoltre, è stata inserita anche una funzione che definisca le variabili che dovranno contenere valori testuali di default e che verrà chiamata prima del parsing degli argomenti forniti da riga di comando.

#### Funzione PrintVersion

Obiettivo della funzione è quello di stampare a schermo sulla Console la stringa contenente il nome, la versione dell’applicazione utilizzata e le informazioni di contatto dell’autore in caso di richieste di informazioni, segnalazione di problemi e sottomissione di richieste di nuove caratteristiche.

##### Direttive richieste dalla funzione

* using System;

##### Codice implementato

static void PrintVersion()

/\*

Obiettivo della funzione è quello di stampare a schermo sulla Console la stringa contenente il nome,

la versione dell’applicazione utilizzata e le informazioni di contatto dell’autore in caso di richieste

di informazioni, segnalazione di problemi e sottomissione di richieste di nuove caratteristiche.

\*/

{

Console.WriteLine("Train Simulator 2020 KML to CSV Converter"

+ Environment.NewLine

+ @"A C# Console Application by Daniele Gastaldo"

+ Environment.NewLine

+ "for Train Simulator 2020 Downloadable Content (DLC) developers."

+ Environment.NewLine

+ "Version 1.0"

+ Environment.NewLine

+ "For further informations, feature requests and bug reports"

+ Environment.NewLine

+ "please contact the author at the e-mail adress"

+ Environment.NewLine

+ @"daniele.gastaldo.1991@outlook.it"

);

}

#### Funzione PrintUsage

L’obiettivo della funzione PrintUsage è quello di stampare a schermo, oltre alle informazioni sull’applicazione (nome, versione e le informazioni di contatto dell’autore) anche le istruzioni d’uso delle varie opzioni che possono essere fornite al programma tramite la linea di comando (tra cui anche la necessità o meno dell’argomento per ciascun comando).

Questo compito verrà eseguito richiamando prima la funzione PrintVersion precedentemente definita e aggiungendo in coda a quanto stampato da tale funzione le informazioni d’uso, organizzate in un modo simile a quanto succede al resto delle informazioni nella funzione PrintVersion.

##### Direttive richieste dalla funzione

* using System;

##### Codice implementato

#### Parsing nel Main method

##### Direttive richieste dalla funzione

##### Codice implementato