Manual CompeScripts

CompeGPS Team S.L.

support@compegps.com
http://www.compegps.com

Índice

1 Introducción

Este documento es una guía para todas aquellas personas que estén interesadas en conocer el funcionamiento del lenguaje CompeScript, el lenguaje de scripts para automatizar algunas acciones en el Programa CompeGPS LAND.

Los scripts, pueden ejecutarse usando CompeGPS version 6.2 o superiores, desde el menú “Archivo > Arrancar archivo de script”.

CompeScript es un lenguaje con una sintaxis muy parecida a java o c++.

El documento esta dividido principalmente por cuatro partes. La primera parte describe la sintaxis de CompeScript, la segunda parte contiene una explicación sobre el control del flujo de ejecución de este mismo lenguaje, y la tercera parte lista y hace una descripción sobre las funciones que este contiene. La cuarta parte contiene algunos ejemplos.

Nota: Las palabras o caracteres contenidos entre dobles comillas (“”) forman parte de las cadenas de caracteres reservadas del lenguaje CompeScript.

2 Sintaxis de CompeScript

Al igual que los lenguajes naturales, los lenguajes computacionales tienen una gramática, es decir, un conjunto de reglas que describen los elementos que componen el lenguaje y la forma correcta de utilizarlos. Una parte de la gramática, la sintaxis, se ocupa de las reglas para combinar adecuadamente los elementos del lenguaje de forma que tengan un sentido. En el caso de los lenguajes computacionales, enseña a construir sentencias que describan operaciones correctas de un computador.

Si nos referimos concretamente a la sintaxis, sus reglas pueden ser expresadas en forma de una lista, que adecuadamente interpretada, describe estas reglas sintácticas. A continuación se expone la sintaxis permitida en CompeScript para las sentencias.

2.1 Tipos de datos

A diferencia de muchos lenguajes de programación, CompeScript utiliza tan solo un tipo de datos general válido para realizar todas las operaciones necesarias de programación. El tipo de dato utilizado en CompeScript es del tipo “Object”.

2.2 Tipos de objetos

Seguidamente se puede ver una lista de los objetos predefinidos existentes en CompeScript, junto con los atributos que estos tienen

Objeto	Atributos	Descripción
CompeGPS		
CompeScript		
Config	Todos los que conforman el archivo config.ini	Permite definir o modificar cualquier atributo del archivo config.ini

A parte de estos objetos ya predefinidos se pueden crear mas tipos de objetos en CompeScript.

2.3 Inicialización de los objetos

Todos los objetos son inicializados con la asignación de un objeto del mismo tipo que tiene la variable a la que le queremos asignar. Y se inicializa de la siguiente forma:

```
Nombre_objeto = objeto;
```

2.4 Operadores

CompeScript utiliza la mayoría de los operadores más comunes en otros lenguajes de programación. Estos tipos de operadores se pueden clasificar en cuatro grupos: aritméticos (+, -, *, /, %), de asignación (=, +=, -=, *=, /=), relacionales (==, <, >, <=, >=, !=) y lógicos (&&, ||).

2.4.1 Operadores aritméticos

Los operadores aritméticos son todos ellos operadores binarios, CompeScript utiliza los cinco operadores siguietes:

- Suma: +
- Resta: -
- Multiplicación: *
- División: /
- Resto: %

Todos los operadores se pueden aplicar en variables y expresiones. El resultado es el que se obtiene de aplicar la operación correspondiente entre los dos operadores.

2.4.2 Operadores de asignación

Los operadores de asignación atribuyen a una variable (depositan en la zona de memoria correspondiente a dicha variable) el resultado de una expresión o el valor de otra variable.

El operador de asignación más utilizado es el operador de igualdad “=”.

Los otros operadores de asignación existentes en CompeScript (=, +=, -=, *=, /=) simplifican algunas operaciones recurrentes sobre una misma variable. Su forma general es:

variable op = expresion;

donde “op” representa cualquiera de los operadores. La expresión anterior es equivalente a:

variable = variable op expresion;

2.4.3 Operadores relacionales

Los operadores relacionales permiten estudiar si se cumplen o no condiciones, que darán paso a posibles alternativas en el código del programa. Así que estos operadores producen un resultado u otro según se cumplan o no algunas condiciones.

En CompeScript, un 0 representa la condición “false”, y cualquier número distinto a 0 equivale a la condición “true”.

Los operadores relacionales son los siguientes:

- Igual que: ==
- Menor que: <
- Mayor que: >
- Menor o igual que: <=
- Mayor o igual que: >=
- Distinto que: !=

Todos los operadores relacionales son operadores binarios, y su forma general es la siguiente:

expresión1 op expresion2

donde “op” es uno de los operadores (==, <, >, <=, >=, !=). El funcionamiento de estos operadores es evaluar “expresion1” y “expresión2”, y se comparan los valores resultantes. Si la condición representada por el operador relacional se cumple, el resultado es 1 (true); si la condición no se cumple, el resultado es 0 (false).

2.4.4 Operadores lógicos

Los operadores lógicos son operadores binarios que permiten combinar los resultados de los operadores relacionales, comprobando que se cumplen simultáneamente varias condiciones, que se cumple una u otra, etc. CompeScript tiene dos operadores lógicos: el operador “Y” (&&) y el operador “O” (||). En inglés son los operadores “and” y “or”. Su forma general es la siguiente:

expresion1	||	expresion2
expresion1	&&	expresion2

2.5 Comentarios

CompeScript permite al programador introducir comentarios en los ficheros fuente que contienen el código de su programa. La misión de los comentarios es servir de explicación o aclaración sobre cómo está echo el programa, para el mismo programador en un futuro o para otras personas. El compilador ignora por completo los comentarios.

Los caracteres “/*” se emplean para iniciar un comentario introducido entre el código del programa, el comentario termina con los caracteres “*/”.

Además se considera que son comentarios todo aquel texto que está desde dos barras consecutivas “//” hasta el fin de la línea.

3 Control del flujo de ejecución

Sistemas de Información Geográfica, tanto en aspectos catastrales y de agricultura como de enseñanza,

3.1 Bifurcaciones

3.1.1 Sentencia IF

Esta sentencia de control permite ejecutar o no una sentencia según se cumpla o no una determinada condición. Esta sentencia tiene la siguiente forma general:

```
if (expresion){
        sentencia
}
```

Se evalúa “expresion”, si el resultado es “true” (!=0), se ejecuta “sentencia”; si el resultado es “false” (=0), se salta “sentencia” y se prosigue en la línea siguiente. Hay que recordar que “sentencia” puede ser una sentencia simple o compuesta.

3.1.2 Sentencia IF… ELSE

Esta sentencia permite realizar una bifurcación, ejecuta una parte u otra del programa según se cumpla o no una cierta condición. La forma general es la siguiente:

```
If (expresion){
        sentencia1
}else{
        sentencia2
}
```

Se evalúa “expresion”, si el resultado es “true” (!=0), se ejecuta “sentencia1” y se prosigue en la línea siguiente a “sentencia2”; si el resultado es “false” (=0), se salta “sentencia1”, se ejecuta “sentencia2” y se prosigue en la línea siguiente. Hay que recordar aquí también que “sentencia1” y “sentencia2” pueden ser sentencias simples o compuestas.

3.1.3 Sentencia IF… ELSE Múltiple

Text

3.2 Bucles

Además de bifurcaciones, en CompeScript existen también varias sentencias que permiten repetir una serie de veces la ejecución de unas líneas de código. Esta repetición se realiza, bien un número determinado de veces, bien hasta que se cumpla una determinada condición de tipo lógico o aritmético. De modo genérico, a estas sentencias se les denomina bucles. Las dos construcciones de CompeScript para realizar bucles son el “while”, el “do-while” y el “for”.

3.2.1 Sentencia WHILE

Esta sentencia permite ejecutar repetidamente, mientras se cumpla una determinada condición, una sentencia o bloque de sentencias. La forma general es como sigue:

```
while (expresion_de_control){
        sentencia
}
```

Se evalúa “expresion_de_control” y si el resultado es “false” se salta “sentencia” y prosigue la ejecución. Si el resultado es “true” se ejecuta “sentencia” y se vuelve a evaluar “expresion_de_control”. La ejecución de “sentencia” prosigue hasta que “expresion_de_control” se hace “false”, en cuyo caso la ejecución continua en la línea siguiente.

3.2.2 Sentencia FOR

La sentencia “for” es el tipo de bucle mas utilizado en la mayoría de lenguajes de programación orientados a objetos. Su forma general es la siguiente:

```
for (inicializacion; expresión_de_control; actualizacion){
        sentencia;
}
```

Antes de iniciarse el bucle se ejecuta “inicializacion”, que es una o más sentencias que asignan valores iniciales a ciertas variables o contadores. A continuación se evalúa “expresión_de_control” y si es “false” se prosigue en la sentencia siguiente a la construcción “for”; si es “true” se ejecutan “sentencia” y “actualizacion”, y se vuelve a evaluar “expresión_de_control”. El proceso prosigue hasta que “expresión_de_control” sea “false”. La parte de “actualizacion” sirve para actualizar variables o incrementar contadores.

4 Funciones

Una función es una parte de código independiente del programa principal y de otras funciones, que puede ser llamada enviándole unos datos o no, para que realice una determinada tarea y/o proporcione unos resultados.

Seguidamente se expone un listado de las diferentes funciones disponibles que están definidas en CompeScript, posteriormente se explica de que forma hay que utilizarlas.

4.1 Llamada de una función

La llamada a una función se hace incluyendo el nombre, seguido de una lista de argumentos separados por comas y encerrados entre paréntesis. A los argumentos incluidos en la llamada se les llama argumentos actuales.

Cuando se llama a una función, se ejecuta el código correspondiente a la función hasta que se llega a una sentencia “return” o al final del cuerpo de la función, y entonces se devuelve el control al programa que realizó la llamada.

4.2 Funciones de Clase ‘CompeScript’

Función	Retorno	Descripción
CompeScript.RequiredVersion(6.2)		Solicita que se esté ejecutando el número de versión pasado como parámetro, en caso contrario termina la ejecución.
CompeScript.CreateRandom (min,max)	Número	Crea un numero aleatorio
CompeScript.SetProjection (projection string, datum name);	Proyeccion	Crea una nueva proyeccion.
CompeScript.Sleep (milisegs)		Detiene la ejecución del script durante un tiempo.
CompeScript.Exit()		Finaliza la ejecución del script.

4.3 Funciones de Clase 'CompeGPS'

Función	Retorno	Descripción
CompeGPS.Exec (int,params)		Ejectuta una comanda de CompeGPS. Cada comanda tiene un numero asociado.
CompeGPS.Open(string)		Abrir el objeto pasado como parámetro.
CompeGPS.CloseAll()		Cerrar todo.
CompeGPS.CreateWpts()	Waypoints (wpts)	Crea lista de waypoints.
CompeGPS.OpenTrackserver(string)		Abre el trackserver.
CompeGPS.DownloadTrackGPS(0)	Vuelo (vol)	Descarga un track del GPS.
CompeGPS.Selected()	Objeto	Retorna el objeto seleccionado.
CompeGPS.LandAltitude(coord)	Número	Retorna la altitud del punto de coordenada que se pasa como parámetro.
CompeGPS.InputUser(string,"2");	Objeto	Pregunta al usuario con el string pasado como parámetro, y retorna un objeto.
CompeGPS.DirScan ("C:\mapas\Comarcas*.imp")	Objeto (lista elementos)	Retorna la lista de archivos con la extensión pasada como parámetro.
CompeGPS.NewVectorMap("world2.mpv")	Objeto (mapa)	Crea y Retorna un nuevo mapa vectorial.
CompeGPS.DownloadPlugIn ("CompePlugIn_Google.dll",version);		Descarga el plugin pasado por parámetro.
CompeGPS.ZoomTo (map,zoom)		Realiza un zoom del mapa y la región pasados como parámetros.
CompeGPS.Repaint2D()		Repinta la ventana de visualización 2D.
CompeGPS.UnifyMaps ("c:\\map1.imp","c:\\map2.ecw",5,20,0)		Unifica todos los mapas cargadps. Se le dan como parámetros los dos mapas a unir, los m/pix, Parámetro 1: Nombre del archivo de calibración del nuevo mapa Parámetro 2: Nombre de la imagen del nuevo mapa Parámetro 3: Metros/Píxel del nuevo mapa. Si es 0, usará la escala del mejor mapa cargado, y si es -1, usará la escala de la vista actual. Parámetro 4: Compresión de la imagen resultante. (Solo para guardar a ECW). Parámetro 5: true / false. Indica si pintar sobre el mapa los tracks y waypoints cargados o no.
CompeGPS.MessageBox ("Hello my friend!")		Muestra un mensaje por pantalla.
CompeGPS.Warning ("Waypoint		Muestra un aviso al usuario por pantalla.

openned!")		
CompeGPS.new (string)	Objeto	Retorna un nuevo objeto, del tipo especificado en 'string'. Los tipos aceptados actualmente son "TWaypoints" "TCoord" "TArray"
CompeGPS.CloseWeb()		Closes the CompeGPS Web viewer window

4.4 Funciones de otros Objetos de CompeGPS

Otros objetos de CompeGPS, como mapas, tracks, waypoints, etc, tambien tienen funciones en CompeScript para automatizar ciertas tareas.

Las siguientes funciones, son comunes a todos los objetos

Función	Retorno	Descripción
Objeto.Exec(765,"")		Ejecuta el número de comando pasado por parámetro.
Objeto.Save()		Guardar el objeto.
Objeto.SaveAs(string)		Guardar el objeto asignando el nombre pasado como parámetro (el parámetro debe incluir el path de donde se debe guardar).
Objeto.UnTouch()		Resets the modified flag of the object.
Objeto.Count()	Número	Retorna el número de ítems que contiene el objeto.
Objeto.Elem(i)	Objeto (item)	Retorna el ítem seleccionado a partir del índice pasado como parámetro.
Objeto1.Add (object2)		Añade un nuevo objeto en el interior del Objeto 1. Por ejemplo, permite insertar un waypoint dentro de un archivo de waypoints.
Objeto.Close()		Cierra el objeto: CompeGPS lo descarga.

4.4.1 Funciones de Mapas

Función	Retorno	Descripción
map.PolygonCount()	Número	Retorna el número de polígonos que contiene el mapa.
map.Polygon(i)	Objeto (polígono)	Retorna el polígono seleccionado a partir del índice pasado como parámetro.
map.AddPolygon (poligon)		Añade el polígono pasado por parámetro al mapa.
map.BringToFront()		Posiciona la pantalla de visualización del mapa delante de todo.
map.Reproject (proyection string,datum)		Cambia la proyeccion de un mapa. Solo sirve con mapas vectoriales. Ejemplo: map.Reproject ("1,Lat/Lon","ED50");

map.Contains(.coords)	**boolean**	Retorna si la coordenada pasada como parámetro está dentro del mapa.
map.Clear();		Deletes all the vectors of the vector map.
map.CreateSubMap		
map.GeoCode_To_Coord		
map.CreateVectorLayer		

4.4.2 Funciones de un Poligono

Función	Retorno	Descripción
pol.LayerId()	Entero	Retorna el numero entero ID de la capa a la que pertenece el polígono.
pol.LayerName()	String	Retorna el nombre de la capa a la que pertenece el polígono.
pol.GetBorderColor()	Color (un numero entero en rgb)	Retorna el color del borde del poligono
pol.SetLayer(layer id)		Cambia el poligono de capa.

4.4.3 Funciones de Relieves

Función	Retorno	Descripción
dem.ExportCotours (mapa,100)		Exporta a un dem las curvas de nivel del mapa pasado como parámetro.
CreateSubDem		
FixDemHoles		

4.4.4 Funciones de un Waypoint

Función	Retorno	Descripción
Objeto.Exec(765,"")		Ejecuta el número de comando pasado por parámetro.
vol.SendTrackGPS (0)		Envía track al GPS.
Objeto.Save()		Guardar el objeto.
Objeto.SaveAs(string)		Guardar el objeto asignando el nombre pasado como parámetro (el parámetro debe incluir el path de donde se debe guardar).
Objeto.Count()	Número	Retorna el número de ítems que contiene el objeto.
Objeto.Elem(i)	Objeto (item)	Retorna el ítem seleccionado a partir del índice pasado como parámetro.
map.PolygonCount()	Número	Retorna el número de polígonos que contiene el mapa.
map.Polygon(i)	Objeto (polígono)	Retorna el polígono seleccionado a partir del índice pasado como parámetro.
pol.Layer	String	Retorna el nombre del Layer del polígono.
map.AddPolygon (poligon)		Añade el polígono pasado por parámetro al mapa.
dem.ExportCotours (mapa,100)		Exporta a un dem las curvas de nivel del mapa pasado como parámetro.
map.BringToFront()		Posiciona la pantalla de visualización del mapa delante de todo.
map.Reproject ("1,Lat/Lon","ED50")		
map.Contains(wpt.coords)	Objeto (dentro)	Retorna el contenido de un mapa

4.4.5 Funciones de un archivo de Waypoints

Función	Retorno	Descripción
Wpts.CreateWpt()	Un waypoint	Crea un nuevo waypoint.

4.4.6 Funciones de un Track

Función	Retorno	Descripción
Track.CrearteNewTrackPoint	Punto de track	Crea un nuevo punto de track, y lo retorna.
Track.SendTrackGPS (0)		Envía track al GPS.

Track.Recalculate();		Recalcula todos los parámetros de un track. Importante llamar a esta función si modificamos el track, por ejemplo, añadiendo puntos.

4.4.7 Funciones de Strings

Función	Retorno	Descripción
Str.FileNameEx	String	
Str.FileNameNoEx	String	
Str.strlen	Integer	
Str.Replace		
Str.SubString	String	
operator <<	String	Example: str << “hello”;

5 Ejemplos

5.1 Abrir un mapa y crear un waypoint

```
// Open the map.
map = CompeGPS.Open("ecwp://www.earthetc.com/Images/geodetic/world/Landsat742.ecw");
wpts = CompeGPS.CreateWpts();
wpt1 = wpts.CreateWpt();
wpt1.shortname = "Arenys de Munt";
wpt1.Coord.lat = 41.6090171;
wpt1.Coord.lon = 2.5404108; CompeGPS.ZoomTo(wpt1,20);
```

5.2 Convertir todos los mapas de una carpeta a rmap

```
CompeGPS.CloseAll();
list = CompeGPS.DirScan ("d:\maps\Catalunya\comarques\*.imp");
n = list.Count();
for (i=0;i<n;i=i+1){
        mapname = list.Elem(i);
        map = CompeGPS.Open (mapname);
        map.SaveAs ("*.rmap");
        map.Close();
};
```

5.3 Reproyectar un mapa vectorial

```
CompeGPS.CloseAll();
map = CompeGPS.Open  ("n:\mapas\vector\Andorra.mpv");
map.Reproject ("1,Lat/Lon","ED50");
map.SaveAs("c:\AndorraLL.mpv");
```

5.4 Unificar mapas

```
CompeGPS.CloseAll();
map1 = CompeGPS.Open("n:\mapas\Alpina\corredor.imp");
map2 = CompeGPS.Open("n:\mapas\Alpina\Montseny.imp");
CompeGPS.Open ("C:\waypoints\New Waypoints2.wpt");
// params:         nom imp    ,nom bmp,    escala,compresion,incluir wpts,tracks,etc.
CompeGPS.UnifyMaps ("c:\newmap.imp","c:\newmap.ecw",0,20,0);
```