# - cTiled -

**Tiled map loader for Cerberus-X**

These functions are used to load **TMX** files created with the Tiled map editor (www.mapeditor.org). With few lines of code you can show complex maps or if you prefer use all the variables to create complex things.

## Examples:

- **Basic** - It contains a very basic example of how a *TMX* file is loaded and how the data is used to draw a map with all its *layers* and *tilesets*.
- **Animation** - It is a simple example of how you can create animations with tiles, as well as how the tileset is used externally (*TSX* files). It can be done in many ways.
- **Properties** - This example shows how to extract the properties of the objects. Properties can be present in "layers", "tileset" and many others. They contain information added by the user.
- **Shapes** - The *objects* can have many shapes to be used on the map. The most common is the rectangle but you can use others such as ellipses, points or even create polygons. In this example I visually show what shapes it has and how they could be used.
- **Tilesets** - This example shows how several different *tileset* can be used on the same map to add more images, effects or animations.

Too easy!

## [ FUNCTIONS ]

### **CTILED**( filename:**String** )
Create the environment to be able to load and use tmx files.

- *filename:* File name ".tmx" to load. For now it only accepts Tile layer format in CSV, XML and Base64 uncompressed. If it is in compressed Base64 it cannot be used (for now).

### Example:

```
Import ctiled

Function Main()

  New Game()

End

Class Game Extends App

  Field tld:CTILED

  Method OnCreate()

    tld = New CTILED("basic.tmx")

  End

  Method OnRender()

    Cls(0, 0, 0)

    ' Extracts data from all tile sets.
    For Local ts:=Eachin tld.tileset

      ' Extracts data from all layers.
      For Local lyr:=Eachin tld.layer

        For Local d:Int=0 Until lyr.data.Count

        ' Tile position.
        Local x:Int = d Mod lyr.width
        Local y:Int = d / lyr.width
        Local id:Int = lyr.data.ToArray[y * lyr.width + x] - ts.firstgid

        ' Draw the tile.
        If id>=0 Then DrawImage(ts.image, x * tld.tileWidth, y * tld.tileHeight, id)

        Next
      Next
    Next

  End

End
```

### Limitations:
- Only can use the Tile layer format in *CSV*, *XML* and *Base64* not compressed.
- There are many variables that cannot be used yet.

## Map

**Field version:String**
The TMX format version. Was “1.0” so far, and will be incremented to match minor Tiled releases.

**Field tiledVersion:String**
The Tiled version used to save the file.

**Field orientation:String**
Map orientation. Tiled supports “*orthogonal*”, “*isometric*”, “*staggered*” and “*hexagonal*”

**Field renderOrder:String**
The order in which tiles on tile layers are rendered.

**Field width:Int**
The map width in tiles.

**Field height:Int**
The map height in tiles.

**Field tileWidth:Int**
The width of a tile.

**Field tileHeight:Int**
The height of a tile.

**Field infinite:Int**
Whether this map is infinite. An infinite map has no fixed size and can grow in all directions.

**Field nextLayerID:Int**
Stores the next available ID for new layers.

**Field nextObjectID:Int**
Stores the next available ID for new objects.

**Field backgroundColor:Color**
The background color of the map.

**Field tileset:=New List<cTILESET>**
It contains an array with all the tilesets that the map contains.

**Field layer:=New List<cLAYER>**
It contains an array with all the layers that the map contains.

**Field objectgroup:=New List<cOBJECTGROUP>**
It contains an array with all the objects groups that the map contains.

**Field properties:=New List<cPROPERTY>**
Contains an array with all the properties. To use correctly you must know the name and type of the variable to use.

# Class cTILESET

**Field firstgid:Int**
The first global tile ID of this tileset (this global ID maps to the first tile in this tileset).

**Field name:String**
The name of this tileset.

**Field tileWidth:Int**
The (maximum) width/height of the tiles in this tileset.

**Field tileHeight:Int**
The (maximum) width/height of the tiles in this tileset.

**Field tileCount:Int**
The number of tiles in this tileset

**Field columns:Int**
The number of tile columns in the tileset. For image collection tilesets it is editable and is used when displaying the tileset.

**Field source:String**
The reference to the tileset image file.

**Field image:Image**
This is the already loaded texture that will be used to display the tiles.

# Class cTILE

**Field id:Int**
The local tile ID within its tileset.

**Field animation:=New List<cFRAME>**
It contains an array with all the frames of the animation that the tile has.

**Field objectgroup:=New List<cOBJECTGROUP>**
It contains an array with all the groups of objects that the tile contains.

# Class cFRAME

### Field tileid:Int
The local ID of a tile within the parent tileset.

### Field duration:Int
How long (in milliseconds) this frame should be displayed before advancing to the next frame.

# Class cLAYER

**Field id:Int**
Unique ID of the layer. Each layer that added to a map gets a unique id.

**Field name:String**
The name of the layer.

**Field width:Int**
The width of the layer in tiles.

**Field height:Int**
The height of the layer in tiles.

**Field tintColor:Color**
A color that is multiplied with any tile objects drawn by this layer.

**Field data:=New List<Int>**
It contains a matrix with all the tile IDs and their position on the map.

**Field properties:=New List<cPROPERTY>**
Contains an array with all the properties. To use correctly you must know the name and type of the variable to use.

# Class cOBJECTGROUP

## Field id:Int
Unique ID of the layer. Each layer that added to a map gets a unique id.

## Field name:String
The name of the object group.

## Field tintcolor:Color
A color that is multiplied with any tile objects drawn by this objectgroup.

## Field object:=New List<cOBJECT>
Contains an array with all the objects in the group.

## Field properties:=New List<cPROPERTY>
Contains an array with all the properties. To use correctly you must know the name and type of the variable to use.

# Class cOBJECT

### Field id:Int
Unique ID of the object. Each object that is placed on a map gets a unique id.

### Field x:Int
The x coordinate of the object in pixels.

### Field y:Int
The y coordinate of the object in pixels.

### Field rotation:Int
The rotation of the object in degrees clockwise around (x, y).

### Field width:Int
The width of the object in pixels.

### Field height:Int
The height of the object in pixels.

### Field shape:Int
The object can have several shapes and each one has different variables:

**eRECTANGLE:** Used to mark an object as a rectangle.

**eELLIPSE:** Used to mark an object as an ellipse. Used to mark an object as an ellipse. The existing *x*, *y*, *width* and *height* attributes are used to determine the size of the ellipse.

**ePOINT:** Used to mark an object as a point. The existing *x* and *y* attributes are used to determine the position of the point.

**ePOLYGON**
- Field points:=New List<cVECTOR> : A list of *x*,*y* coordinates in pixels.

Each polygon object is made up of a space-delimited list of *x*,*y* coordinates. The origin for these coordinates is the location of the parent object. By default, the first point is created as 0,0 denoting that the point will originate exactly where the object is placed.

### Field properties:=New List<cPROPERTY>
Contains an array with all the properties. To use correctly you must know the name and type of the variable to use.

# Class cPROPERTY

**Field name:String**
The name of the property.

**Field type:String**
The type of the property. Can be string, int, float, bool, color, file or object.

**Field boolValue:Bool**
Boolean properties have a value of either “true” or “false”.

**Field intValue:Int**
Integer value of the property.

**Field floatValue:Float**
Float value of the property.

**Field fileValue:String**
File properties are stored as paths relative from the location of the map file.

**Field stringValue:String**
String value of the property.

**Field objectValue:Int**
Object properties can reference any object on the same map and are stored as an integer (the ID of the referenced object, or 0 when no object is referenced).

**Field colorValue:Color**
Color properties are stored in ARGB.

## LICENSED



**cTiled - Zlib license.**



**If you think this tool is useful, you can help me in various ways: help me improve it, advertising where you think is convenient or making a donation.**

**-To contact me or DONATION$-**

or

**themrcerebro@gmail.com**

**THANKS!!!**