Earwigo, the Web Builder

Tiny Adventure

Tutorial

By Geo Magician

Version 0.8 (14th October 2011)

1 Foreword from the Author

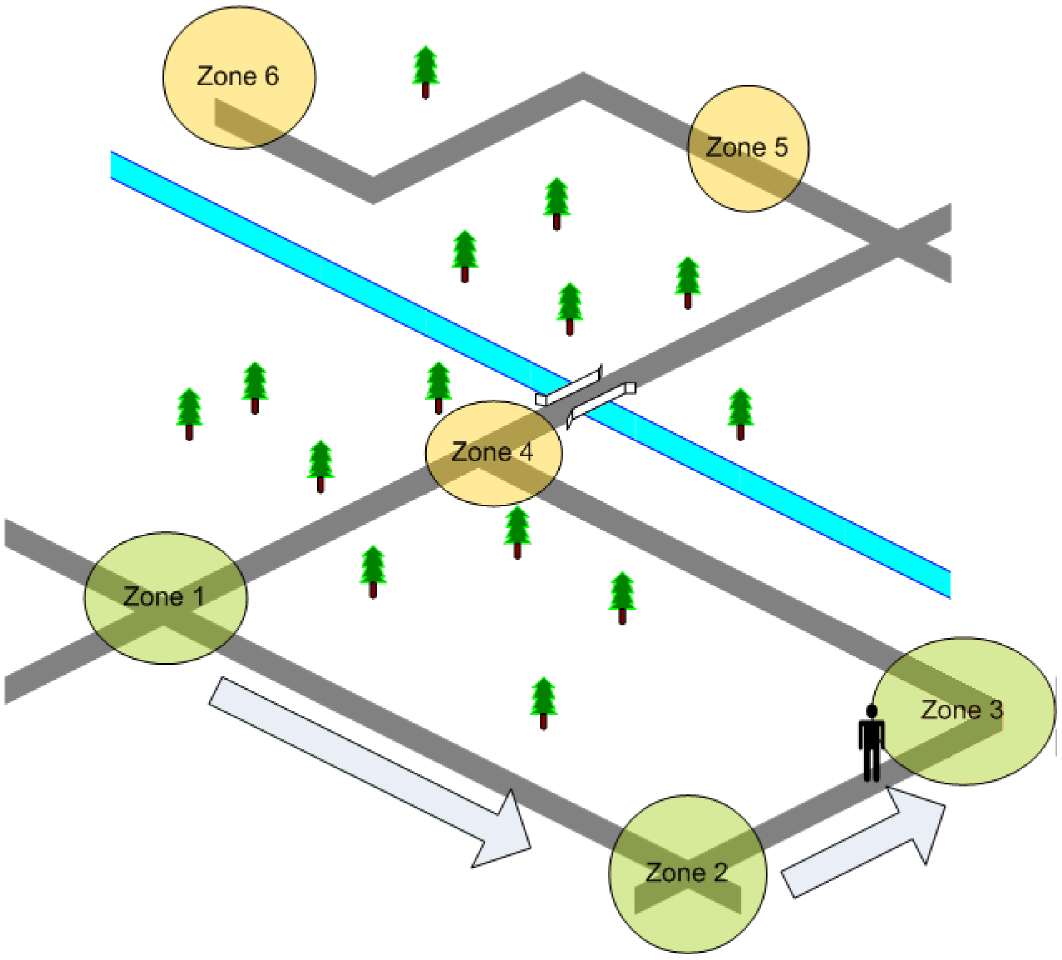
This tutorial is currently still under construction. I know we all hate this kind of disclaimer but I have to let you know that the tutorial will come to a sudden end without completing the cartridge. So far people report that the tutorial helped a lot to get started (and they demand that I complete it 😉). When you run out of tutorial, the following links from the Groundspeak Wherigo Wiki can be helpful:

- The Builder section can be used to explains the Earwigo tabs
- The object classes are more for the programmers that use Earwigo

Having said this … let us have a look at

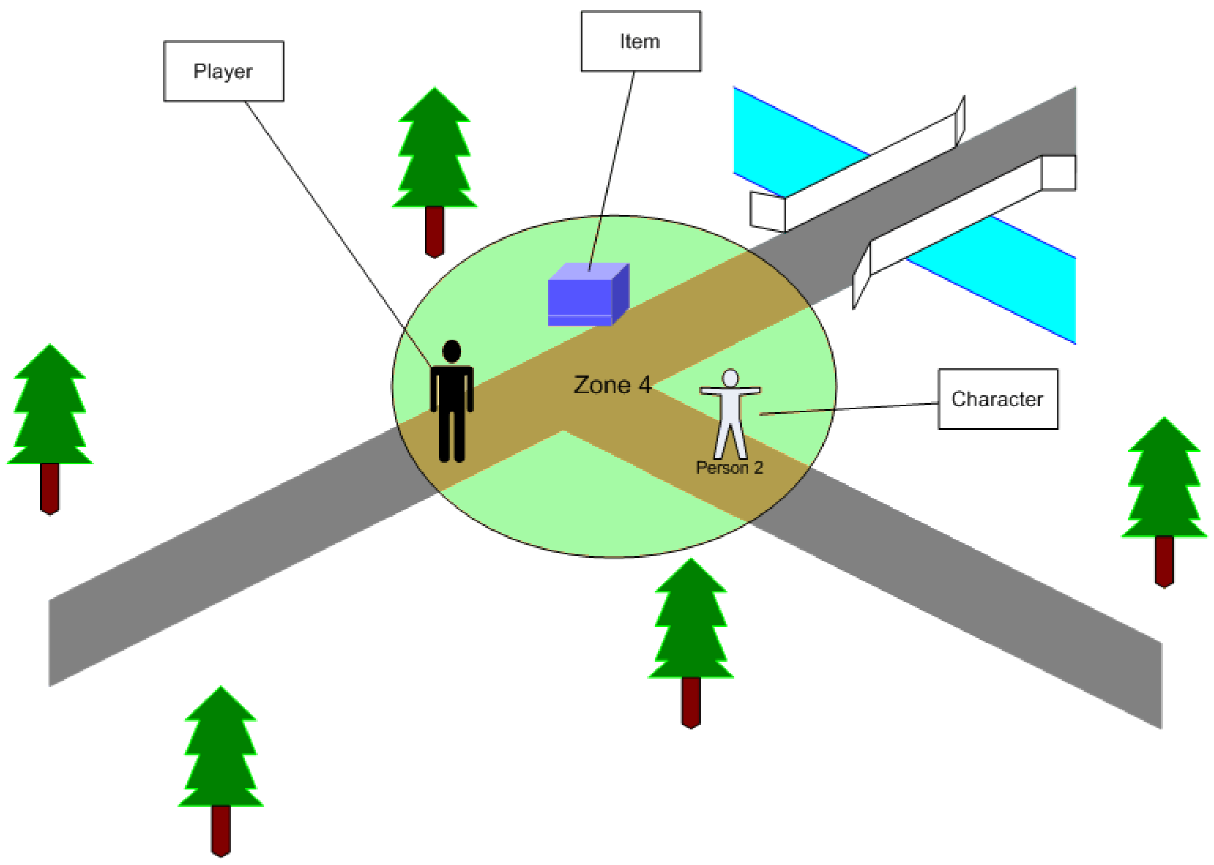
2 The Wherigo Concept

The basic concept of a Wherigo cartridge is the creation of zones on a map.



In the above example 6 zones are created on the map and the human player (black) is walking from zone to zone to solve the cartridge. You can see that some zones are already visible to the human player (they are marked in green), other zones (marked in orange) are still hidden from the human player.

The human player can use his GPS or PDA to navigate to a visible (green) zone. Our human player is currently on the way to zone 3. This zone will contain a puzzle, an interaction or a clue that activates zone 4.



Zone 4 or any other zone can contain objects as depicted above. The human player can interact with the Character "Person 2" or he can pick up or manipulate the Item. This interactions will then enable zone 5 and the player can continue his quest. Once zone 5 is visible, the human player can locate it on his screen and walk towards it. He navigates from zone to zone until he reaches his final target (zone 6). This is where the cartridge will end.

3 The Wherigo Program

3.1 LUA the Wherigo Language

The Wherigo programming language is quite similar to other progamming languages. The Wherigo language is called LUA. The good news is, you will typically not see the LUA program when you use the Earwigo builder. The LUA program is written automatically "behind the scene" when you associate certain objects in Earwigo. Earwigo will write all required program code for you and you will not have to create a single line of LUA code. In other words, you do not need to be a programmer to write a cartridge (although it helps 😉).

Some very fancy ideas are however only possible if you can tweak the code by hand. Earwigo provides an "Author Script" section (on the Cartridge tab) and the option to insert LUA statements at any given place in the builder for this purpose. If you are experienced with LUA or if you have found a section of code in this Wiki that will enhance your cartridge, this is where the hand written LUA code fits in.

If you have the need to see the LUA version of your cartridge (I have not done this since the beta testing phase of Earwigo) you can do this on the “Manage Cartridge Screen”.

3.2 Event Driven Programming

LUA is an event driven language and this has caused me some headache in the past. As a programmer I am used to writing program code in a sequence and the code will be executed in this sequence. This is also the way any human without programming experience would think things should happen. However this is not the case with LUA. Every program code associated with one event will be executed when this event fires. If you have six lines of code associated with one event, all six lines will be executed (almost) simultaneously.

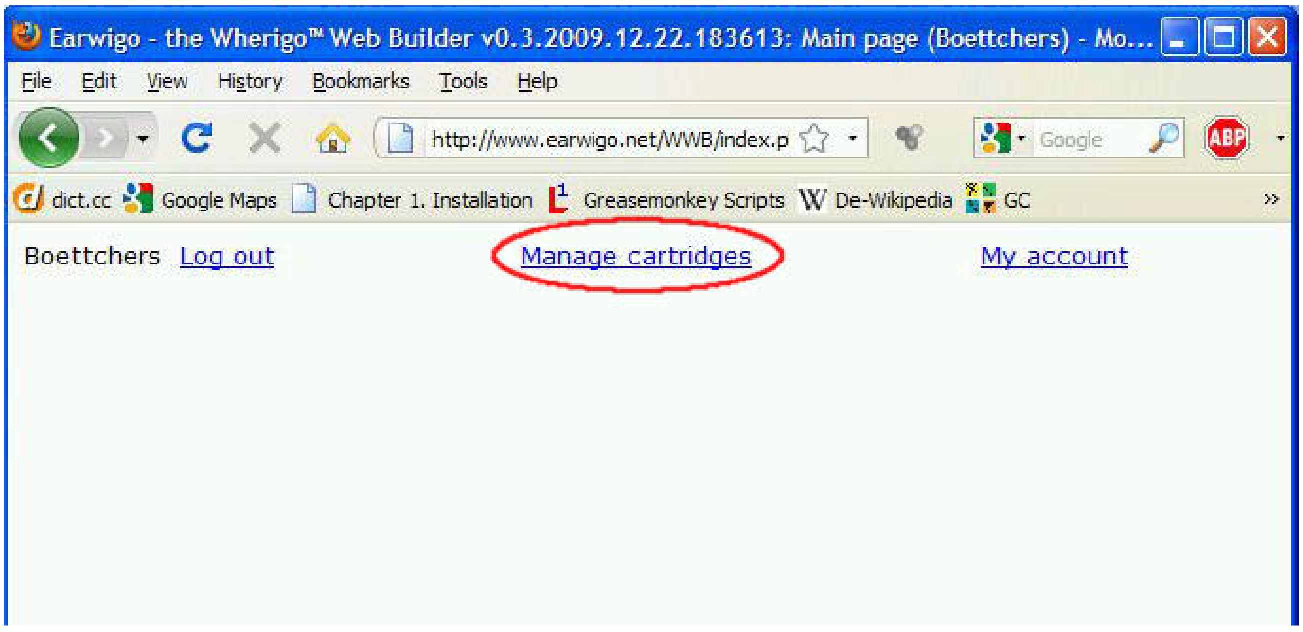
A typical mistake with Earwigo is therefore to have more than one text output associated with one event. The result is that both texts will be presented (almost) at the same time. The first message will be displayed and immediately thereafter the second message will overwrite the first message. The user can only see the last text message.

The best way to think about this is to assume that everything that you can see on one page (one event) in Earwigo will happen at the same time. If you need to display two messages, create two events (e.g. pushing a button at the end of your first message). Pushing this button will then be the additional event that triggers your second message.

This event driven thing is not as bad as it sounds. I am sure however that you will fall into this trap once or twice. But now you know what causes it and you can easily adjust your cartridge to compensate for it.

Having said this I think we have had enough theory and it is time for

4 Your First Cartridge

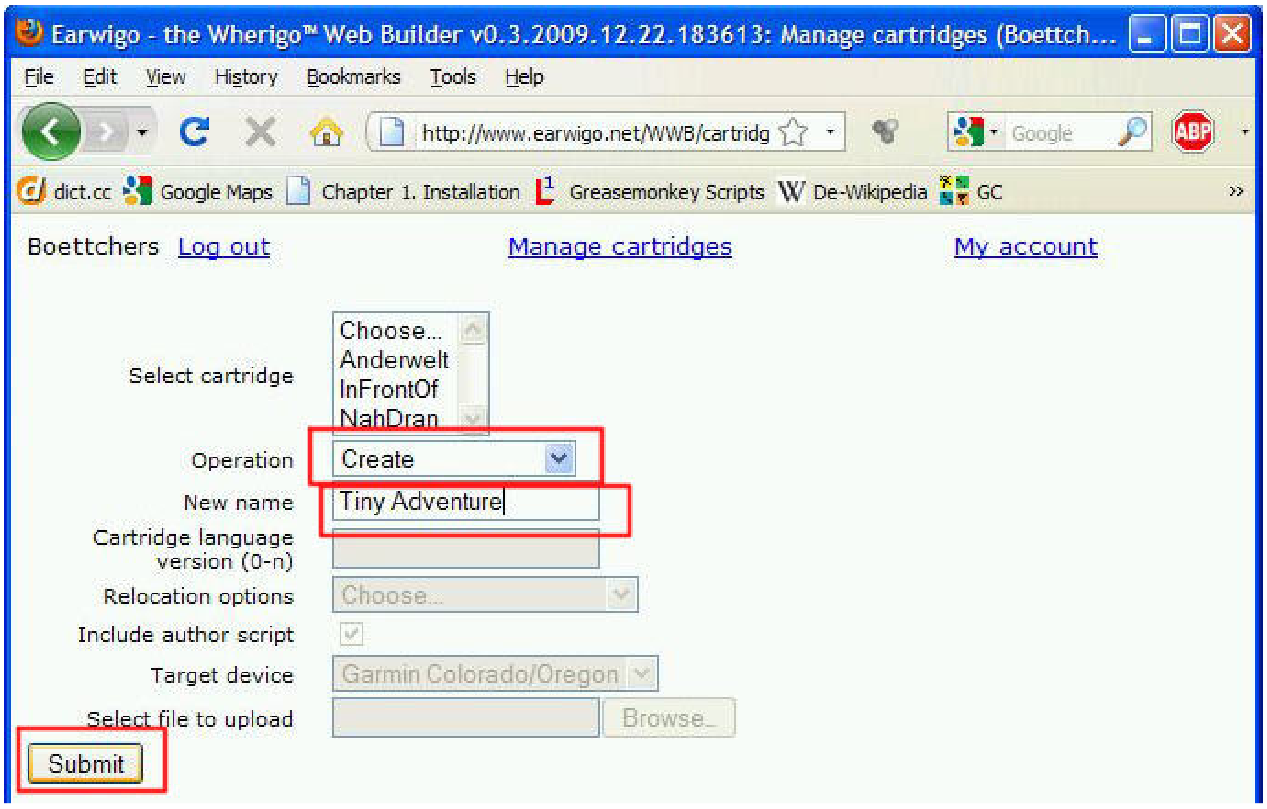

Earwigo - the Wherigo™ Web Builder v0.3.2009.12.22.183613: Main page (Boettchers) - Mo...
File Edit View History Bookmarks Tools Help
http://www.earwigo.net/WWB/index.p
Google
dict.cc Google Maps Chapter 1. Installation Greasemonkey Scripts De-Wikipedia GC

Boettchers Log out Manage cartridges My account

Once you log in for the first time, you'll be presented with an almost blank page with your username and three options (Log out, Manage Cartridges, and My Account). You'll want to click on Manage Cartridges.

4.1 Creating the cartridge

After you click on the Manage Cartridges link, you'll be presented with a powerful and possibly slightly intimidating form. This is the main form for doing things related to cartridges such as creating, deleting, loading, compiling, backing up, etc. For our purposes, we're only interested in a couple of the options.

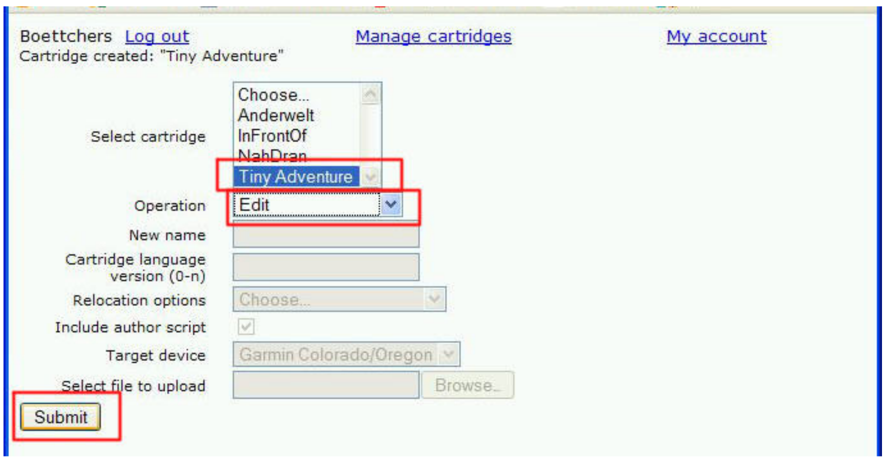

Earwigo - the Wherigo™ Web Builder v0.3.2009.12.22.183613: Manage cartridges (Boettch...
File Edit View History Bookmarks Tools Help
http://www.earwigo.net/WWB/cartridg
Google
dict.cc Google Maps Chapter 1. Installation Greasemonkey Scripts De-Wikipedia GC
Boettchers Log out
Manage cartridges
My account
Browse...
Submit

Select cartridge	Choose... / Anderwelt / InFrontOf / NahDran
Operation	Create
New name	Tiny Adventure
Cartridge language version (0-n)	
Relocation options	Choose...
Include author script	✓
Target device	Garmin Colorado/Oregon
Select file to upload	

- In the "Operation" drop down box, select "Create"
- The "New name" field will now be enabled… fill in the name "Tiny Adventure".
- Once that's all done, hit the "Submit" button.

Earwigo will now create a blank cartridge for you and reload the Manage screen.

4.2 Loading the cartridge



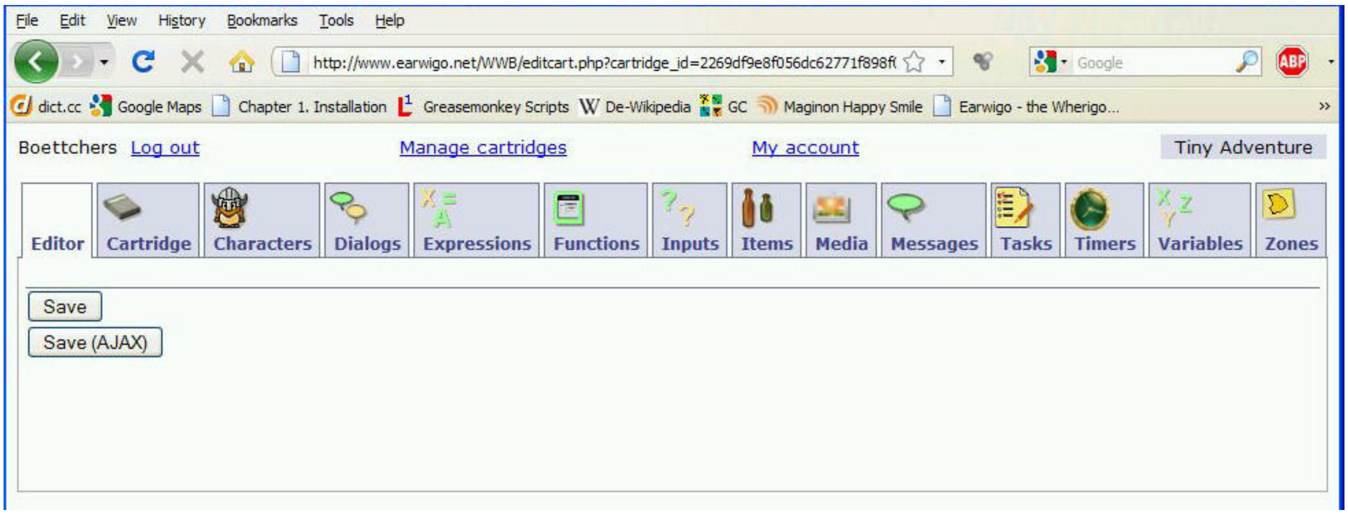
Cartridge created: "Tiny Adventure"

Select cartridge	Choose... / Anderwelt / InFrontOf / NahDran / Tiny Adventure
Operation	Edit
New name	
Cartridge language version (0-n)	
Relocation options	Choose...
Include author script	✓
Target device	Garmin Colorado/Oregon
Select file to upload	

To begin editing the cartridge, select “Tiny Adventure” from the “Select cartridge” box and then choose “Edit” as the “Operation”. Click “Submit” to go to start the cartridge editor.

5 Editing the cartridge

This edit cartridge screen is probably where you'll spend most of your time in Earwigo.


File Edit View History Bookmarks Tools Help
http://www.earwigo.net/WWB/editcart.php?cartridge_id=2269df9e8f056dc62771f898f(
Google
dict.cc Google Maps Chapter 1. Installation Greasemonkey Scripts De-Wikipedia GC Maginon Happy Smile Earwigo - the Wherigo...
Boettchers Log out
Manage cartridges
My account
Tiny Adventure
Editor
Cartridge
Characters
Dialogs
Expressions
Functions
Inputs
Items
Media
Messages
Tasks
Timers
Variables
Zones
Save
Save (AJAX)

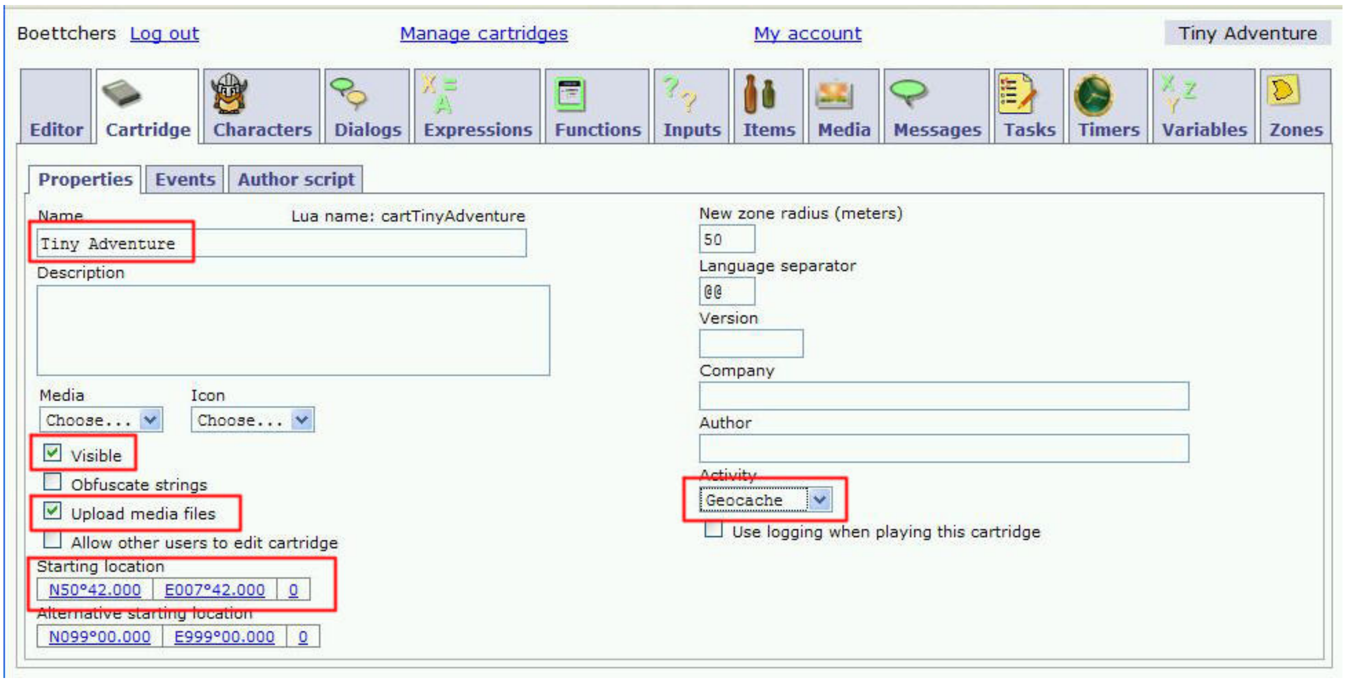
You will always start on the “Editor” tab of the editor. There are currently only two options on this tab, both of them “Save”.

- The standard “Save” option will save your changes and return control to the editor.
- The “Save and Reload” will save your changes and then reload the editor and cartridge from the server. This can be used if you want to ensure a clean start after encountering trouble (which we don't hope 😉).

If you can't see the icons or if you want to disable/change them, click on the “My account” link.

5.1 Initial Cartridge settings

Your first trip should always lead you to the “Cartridge” tab. This is where the overall cartridge settings are configured. Click the “Cartridge” tab to see the initial cartridge settings.


Boettchers Log out | Manage cartridges | My account | Tiny Adventure

Name	Tiny Adventure
Lua name	cartTinyAdventure
Description	
Media	Choose...
Icon	Choose...
Visible	✓
Obfuscate strings	☐
Upload media files	✓
Allow other users to edit cartridge	☐
Starting location	N50°42.000 E007°42.000 0
Alternative starting location	N099°00.000 E999°00.000 0
New zone radius (meters)	50
Language separator	@@
Version	
Company	
Author	
Activity	Geocache
Use logging when playing this cartridge	☐

- The **Name** section shows the current name of the cartridge. Since we are working on the cartridge “Tiny Adventure”, this is the name that you see in this section.

- The cartridge is set to **Visible** by default. I do not think that this has an effect in the current Wherigo players but in general this property makes sure the cartridge is displayed by the Wherigo player.

- **Use logging when playing cartridge** saves a log of the players progress through the cartridge for debugging. We do not need this option during the tutorial.

- **Obfuscate Strings** is used to “scramble” the characters in the Wherigo cartridge. This might be necessary because the normal cartridge file contains all text files in readable format. If the solution to your cartridge is contained in the cartridge text, you might want to obfuscate the cartridge's strings. Since this (totally lua compatible) option poses the risk that the cartridge might not play on some players I try to conceal the cartridge secrets by other means. I do for example really like to ask the human player things like “What 5 digit number do you see at this location?” It is quite hard to find the answer to this kind of question without going to the actual location 😎. But as said earlier, the **Obfuscate** option invokes a perfectly legal lua subroutine and it works well. I have not yet heard any complaints about it.

- **Prevent cartridge from running in emulator** is a really useful function that detects the Groundspeak emulator (a tool that you can use to play a cartridge on google maps). The function displays a warning message and closes the cartridge when it detects the emulator. Use this option if you want to avoid that people solve your cartridge on the sofa 😜. This option is similar to the obfuscate string option. It is perfectly legal in the lua world but some Wherigo players might not like it. I personally prefer the “5 digit trick”.

- Make sure to enable the **Upload media file** check mark. This option allows you to upload graphics directly from your computer to the Earwigo server. This is the most convenient way to work with graphics.

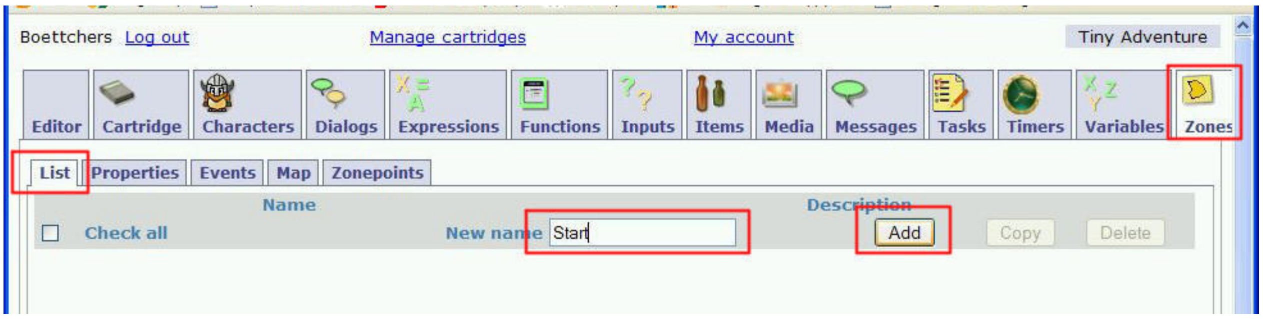
- **Allow other users to edit cartridge** offers the opportunity to develop a cartridge in a team. If this option is enabled you can send the link to the cartridge to your buddy and have him work on the same cartridge. A good team combination would e.g. be the code crunching lua programmer and the well spoken story teller. We will not use this option during the tutorial. *Side Note: Don't use this option to “donate” your cartridge. If you want to “donate” your cartridge to someone in another country and allow him to set up a copy of your cartridge in his country, you should send him a backup cartridge. He will be able to upload the backup to his account.*

- Always set a **Starting location** for your cartridge. This value determines where the cartridge is placed until it is opened (in other words, this is where other people will see your cartridge on a map). This value is also essential for the functionality of the WWB “Zones”. All zones will automatically be created at this location until you specify more precise coordinates. If no “Starting location” is specified, new zones can not be created.

- You might want to set the **Activity type** of your cartridge to an appropriate value. I don't know what effect this value might have but why take chances 😉. We will use Geocache for our tutorial.

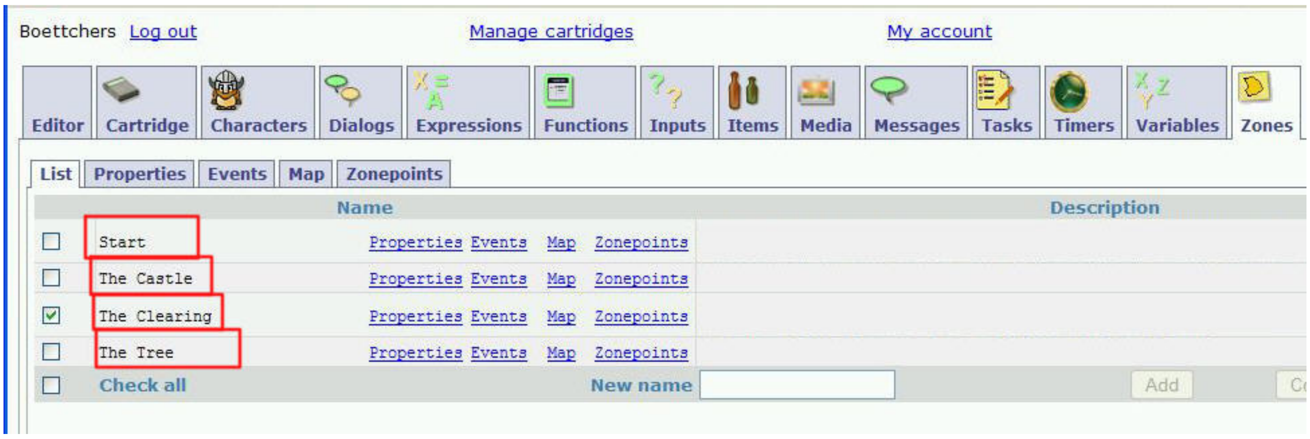
5.2 Creating Zones

Since Wherigo is a location based system I like to start with the creation of geographic zones. Select the “Zones” tab and you should see the “Zones/List” sub-tab. Click on that to see a list of all the zones in your cartridge. If you follow this tutorial the list should be empty at this time (we will change that in a second 😎).



- The first zone that we will create is named “Start”, so enter the name “Start” in the “New name” field.

- Click the “Add” button to add the Start zone to the list (This button is disabled if you have not set a “Starting location” in the “Cartridge” tab).



And here is your first small task for this tutorial:

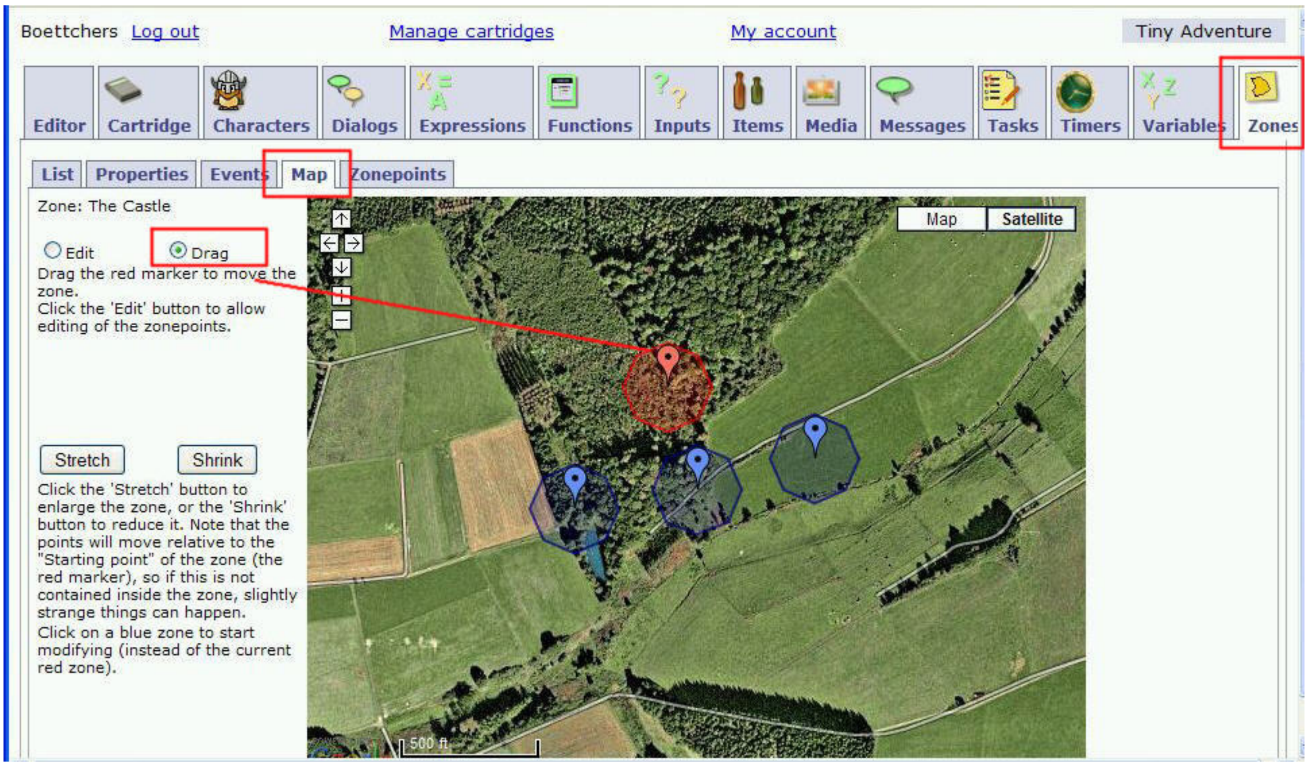
- Continue to add zones until you have entered the four zones

1. Start
2. The Castle
3. The Clearing
4. The Tree

5.3 Using the map with zones

5.3.1 Unclutter the zones

Now click the “Map” sub-tab to start to position the zones. If you have entered the “Starting Location” correctly, you should see a part of the world that is depicted on the following screen-shot (a location somewhere in Germany). *Select “Satellite” to work with the view that I am using in this tutorial.*



Zone: The Castle

Edit [✓ Drag]

Drag the red marker to move the zone.
Click the 'Edit' button to allow editing of the zonepoints.

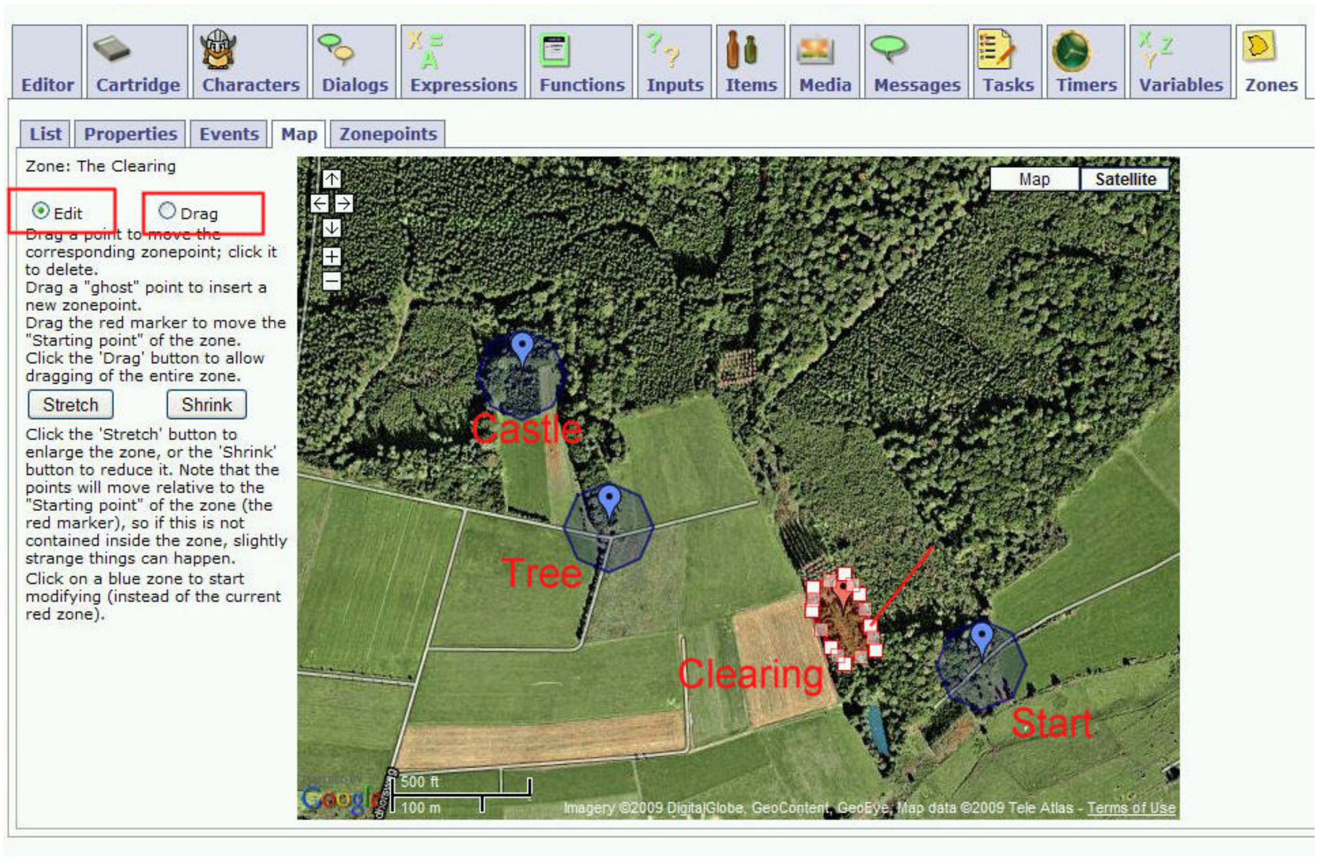
Click the 'Stretch' button to enlarge the zone, or the 'Shrink' button to reduce it. Note that the points will move relative to the "Starting point" of the zone (the red marker), so if this is not contained inside the zone, slightly strange things can happen.
Click on a blue zone to start modifying (instead of the current red zone).

Your map should look slightly different for the moment because all of your zones will be on top of each other. This is due to the fact that all zones are created at the “Starting Location” of the cartridge. Since all zones have an identical shape (predetermined by the “New zone radius”) you will only be able to see the uppermost zone (on top of all others). Let's change this and move the zones away from each other to give you access to all zones.

- Select the “Drag” option to start moving (dragging) the zones.
- Highlight the topmost zone by clicking into the zone area
- Grab the zone marker pin and drag the zone to a nearby location.
- A zone indicator in the top left corner of the “Map” sub-tab window will keep you in-formed which zone you are currently working on (“zone: The Castle” in the above screen-shot).

5.3.2 Position the zones

Your next task in this tutorial is to move all 4 zones to the locations indicated in the next screenshot. Use the zone indicator in the top left corner of the “Map” sub-tab window to make sure that you are working with the desired zone.



Zone: The Clearing

[✓ Edit] Drag

Drag a point to move the corresponding zonepoint; click it to delete.
Drag a "ghost" point to insert a new zonepoint.
Drag the red marker to move the "Starting point" of the zone.
Click the 'Drag' button to allow dragging of the entire zone.

Stretch Shrink

Click the 'Stretch' button to enlarge the zone, or the 'Shrink' button to reduce it. Note that the points will move relative to the "Starting point" of the zone (the red marker), so if this is not contained inside the zone, slightly strange things can happen.
Click on a blue zone to start modifying (instead of the current red zone).

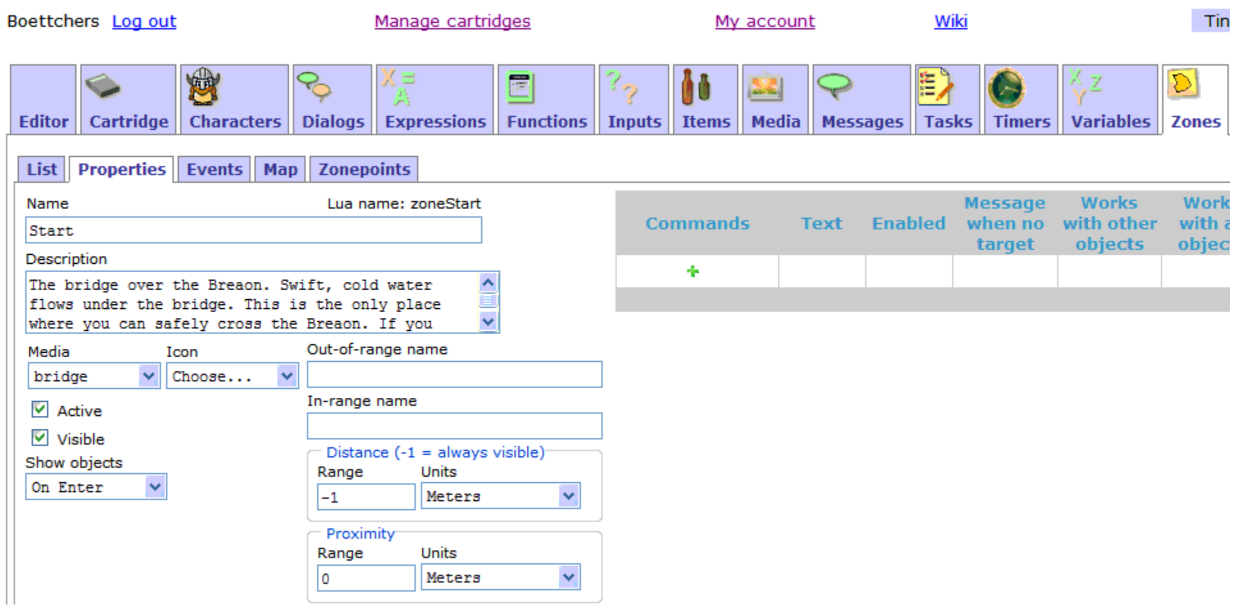
This almost completes the positioning of the zones but I have one last task for you. The zone “The Clearing” is used to mark an existing clearing in the woods. The automatic zone shape is a circle and does not follow the outline of the clearing that you can see in the above screen-shot. Your last map task is to change the shape of the zone “The Clearing”.

- Select the Edit option on the map sub-tab
- Drag the **solid** zone points (indicated by the straight line in the above screen-shot) to change the shape of the zone.
- Create a few additional zone points by dragging **ghost** points (the semi transparent points next to the solid zone points).
- Now delete the superfluous points. Click the newly created zone points to delete them again.
- Your end result should look similar to the above shape of “The Clearing”. You don't have to exactly copy my shape, but try to make sure that your zone marks the clearing on the satellite image.

Remark: It is not safe to asume that all Wherigo players use the border of the zones to determine zone entry and zone proximity. Some Wherigo Players might use the Zone-Center-Point for related calculations. If you adjust the Zone shape to such an extent that the Zone-Center-Point (the marker pin) is no longer within the shape or badly off center, you should adjust the center point as well.

5.4 Zone Properties

You can change the Zone properties when you click on the *Property* link of a particular zone. You can leave almost all Zone properties as they are right now, but I need to give you some idea how to use the properties. Suffer through the theory 😉and try to re-member that you can find detailed information in the tutorial if you need it. If the theory is to boring or you want to get started quickly, you can actually skip this section and continue with the Zone Property Essentials.



Name: Start
Lua name: zoneStart

Description:

```
The bridge over the Breaon. Swift, cold water
flows under the bridge. This is the only place
where you can safely cross the Breaon. If you
```

Media: bridge
Icon: Choose...
Out-of-range name:
In-range name:
Active ✓
Visible ✓
Show objects: On Enter

Distance (-1 = always visible)
Range: -1 Units: Meters

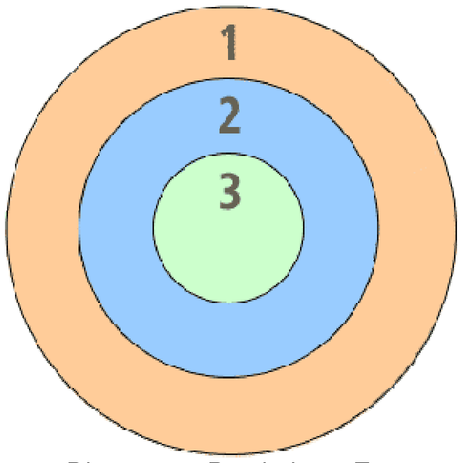
Proximity
Range: 0 Units: Meters

Commands	Text	Enabled	Message when no target	Works with other objects	Work with a objec
+					

Leave all values as they are for now. You will receive a tutorial task at the end of this section. Now is the time for some theory. The following values can be changed on the Zone's Property tab:

5.4.1 Distance and Proximity

There are two different measures of the player's “closeness” to a zone: **Distance** and **Proximity**. Each is a distance such as 100 feet or 200 meters. Visualize these as rings of the specified widths surrounding the zone.



1:Distance 2:Proximity 3:Zone

As the player approaches the zone, S/He will

- first cross the boundary formed by the **Distance** ring,
- then the one formed by the **Proximity** ring,
- then the boundary of the **Zone** itself.

As the player enters each band, an event associated with that band fires. By associating Statements with each event, your cartridge can react to the the fact that the human player approaches the zone. In addition, there is an event which fires when the user exits the zone. There are no exit events for the **Proximity** and **Distance** bands. However, the entry events for these bands fire again on the way out.

When the player leaves the zone, S/He will pass the rings in the reverse order. S/He will

- leave the **Zone** and
- enter the **Proximity** ring again,
- leave the **Proximity** ring and enter the **Distance** ring again.
- The player will finaly leave the **Distance** ring (no event is associated with this)

If you set the **Distance** range to -1 in the builder, the **Distance** entry event fires immediately upon cartridge startup (for initially active zones) or when first activate the zone (for inactive zones).

Important: The **Distance** setting also has an effect on Zone Visibility. See the sections Active and Visible, below.

5.4.2 Active

The Active property determines whether or not the Wherigo player engine processes events related to the zone. If the zone is inactive, events are not processed and the zone is not visible (even if the Visible property is true). In other words: An inactive Zone does **NOTHING**.

You might think it is a smart idea to activate all Zones from the start to make sure that you have some action when you need it. Smart idea, but NO. For each active Zone the Wherigo player must check, whether the user is in, out or in proximity to the zone. This uses a lot of the limited GPS-CPU power. If you have to many active zones, the Wherigo player will waste to much time to perform the checks and your cartridge will crash more likely. As a good compromise ❗**you should not have more than 5 Zones active**❗ at any given time. Activate new Zones and remember to de-activate unused (old) Zones.

5.4.3 Visible

Do not confuse Visibility and active Zones. Only active Zones can be visible or invisible. Inactive Zones do not exist at all. The Visible property determines whether or not an active Zone is visible to the human player on his Wherigo device. This translates to:

- The Zone appears in the Wherigo Player´s locations list
- The human player can “see” it
- The human player can navigate to it

If a zone is active and invisible it will create all Zone events but it will not be detectable for the human player (you can use this if you want to surprise the human player).

Important: A Visible and Active zone is only visible when the player is within the **Distance** range of the zone. If the **Distance** range is -1, the player is always within **Distance** and the **Zone** is always visible (provided it is active).

5.4.4 Zone Property Essentials

- A zone needs to be active in order to be evaluated by the Wherigo players. An inactive Zone is completely skipped by the Wherigo players.
- A Zone is normaly visible. Only visible Zones are listed and can be navigated to.
- Leave the value for Distance at -1 until you know exactly what you are doing 😉.
- Do not have more than 5 active Zones at any time. Deactivate unused Zones.

5.4.5 Tutorial Task for Zone Properties

O.K. enough theory for the moment. Let us use our new knowledge and improve our TinyAdventure. Every cartridge needs minimum one Zone that is active and visible. Therefore go to the properties of the “Start” Zone and select

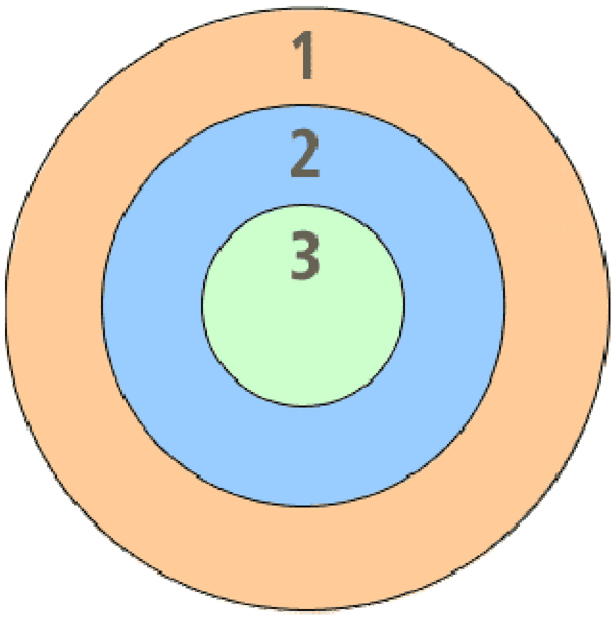
- Active and
- Visible for the “Start” Zone.

5.5 Zone Events

An extremely important feature of Wherigo cartridges are the events. The whole game-play is based on a chain of events. One event is always preparing the next event in the row. It is the task of the cartridge programer to make sure that there is always another event available that will move the game forward. If you run out of events, the cartridge is stuck. A good example for events are zone events. They are executed (triggered, they fire) when the human player physically enters a certain location (zone, ring, band) on the earth.

We have already discussed the different areas of a zone. Let us now have a look at the events that are associated with these zone areas.

5.5.1 OnDistance OnProximity OnEnter OnExit Events



`1:Distance   2:Proximity   3:Zone`

As the player approaches the zone, S/He will

- first cross the boundary formed by the **Distance** ring,

- then the one formed by the **Proximity** ring,
- then the boundary of the **Zone** itself.

As the player enters each band, an event associated with that band fires. By associating Statements with each event, your cartridge can react to the the fact that the human player approaches the zone. In addition, there is an event which fires when the user exits the zone. There are no exit events for the **Proximity** and **Distance** bands. However, the entry events for these bands fire again on the way out.

When the player leaves the zone, S/He will pass the rings in the reverse order. S/He will

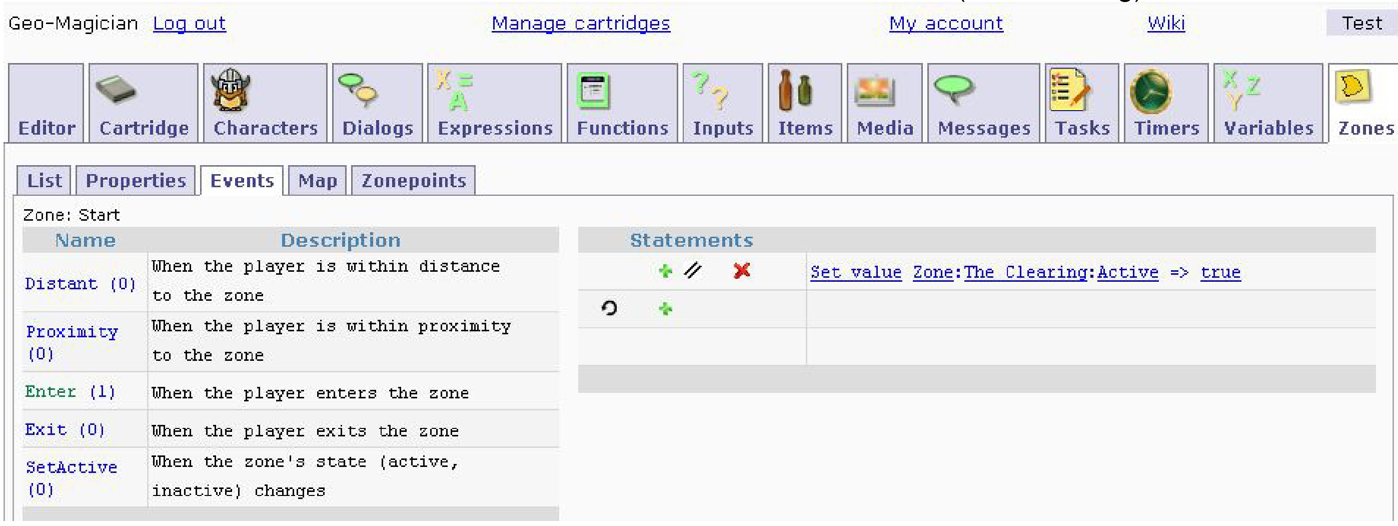
- leave the **Zone** and
- enter the **Proximity** ring again,
- enter the **Distance** ring again.
- The player will finally leave the **Distance** ring (no event is associated with this)

If you set the **Distance** range to -1 in the builder, the **Distance** entry event fires immediately upon cartridge startup (for initially active zones) or with the first activation of the zone (for inactive zones).

5.5.2 Using Zone Events

Using an event in Earwigo is done on an Events sub-tab associated with each zone. The easiest example of event based programing is the usage of the OnEnter event. This event fires every time that the human player walks into the zone as it is outlined on the map. If you use this OnEnter event to always activate the next zone you can create a very basic cartridge that guides the human player from zone to zone.

We will use the OnEnter Event of the Start zone to activate the next zone (The Clearing).



Zone: Start

Name	Description
Distant (0)	When the player is within distance to the zone
Proximity (0)	When the player is within proximity to the zone
Enter (1)	When the player enters the zone
Exit (0)	When the player exits the zone
SetActive (0)	When the zone's state (active, inactive) changes

Statements
Set value Zone:The Clearing:Active => true

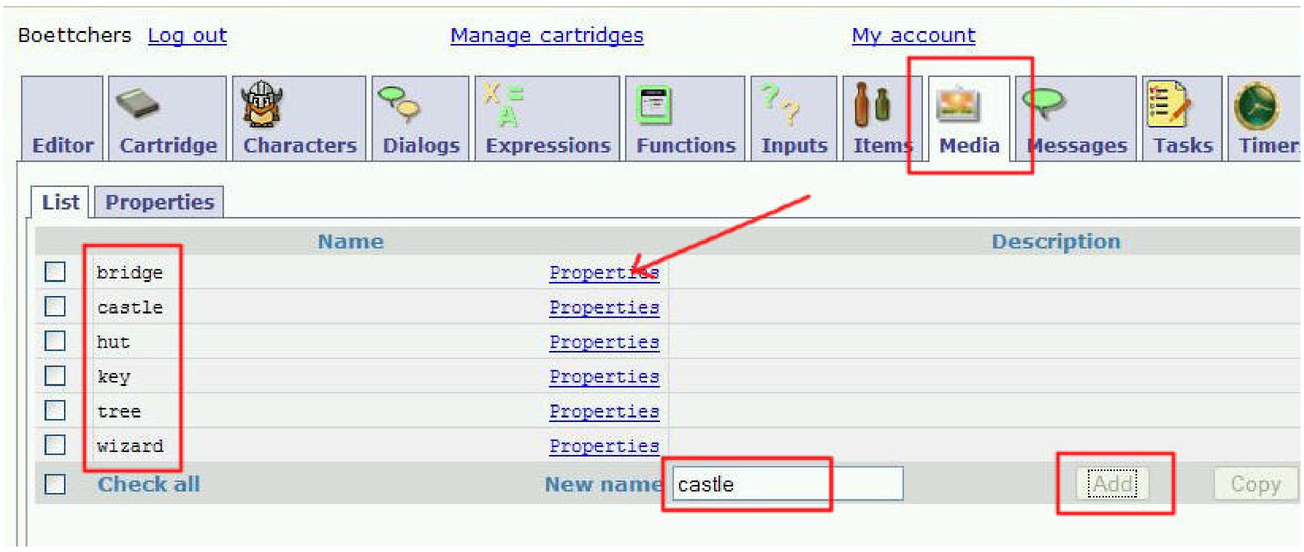
If the human player enters the Start Zone he will trigger the associated OnEnter Event for this zone. This Event will activate the Clearing Zone. The activation of The Clearing will add a new item to the players location list. The human player can select the new location and navigate to it.

5.6 Working with Media

So far, so good. You can now create zones that interact with the human player's position. But the current representation of the zone is just a word in the player's location menu. We are about to change this. Zones (and other objects) can have a picture (called Media object) associated with them.

5.6.1 Creating Media objects

Media objects are created similar to the creation of zone elements. We will create the media elements that we have decided to use in "Tiny Adventure". If you need more media elements you can always add more to the list during your cartridge creation.



Name		Description
bridge	Properties	
castle	Properties	
hut	Properties	
key	Properties	
tree	Properties	
wizard	Properties	

New name: castle

- Select the "Media" tab
- Go to the "List" sub-tab (no entries so far)
- Create the following media objects
  - bridge
  - castle
  - clearing
  - hut
  - key
  - tree
  - wizard

To upload the media files (pictures in our case) that will be associated with the "Media" object click on the "Properties" link of the "Media" object (indicated by an arrow in the above screen-shot). You can also first select the desired "Media" object (check-box in front of the "Media" object) and then select the "Properties" sub-tab.

5.6.2 Uploading media files

5.6.2.1 Newsflash

IE 8 does currently not support this feature. We (that is our programmer ☺) are working on a fix. For now IE8 users have to:

- disable the option "Upload media files" on the cartridge tab
- enter the file names manually for each media object on the media/properties sub-tab
- save the cartridge
- put all media files (the pictures) in a zip archive
- go to the "Manage Cartridge" screen
- select your cartridge
- select the "Operation" upload media
- use the button "Browse" to select your zip file
- select the "Operation" Compile cartridge

- and hit the “Submit” button
-

or change to Firefox (SCNR 😃)

5.6.2.2 Copyright side note

Allow me a quick side note with regard to the images. For this tutorial I have prepared the images for you. They have been taken from Wikimedia Commons and are mostly in the public domain. Some require you to mention the name of the author if you want to use the pictures. Appropriate information for each picture is provided on the download page. Make sure you stick to the license if you want to use the pictures for your own projects.

Click this link to go to the picture page and download the images for "Tiny Adventure".

5.6.2.3 Image size side note

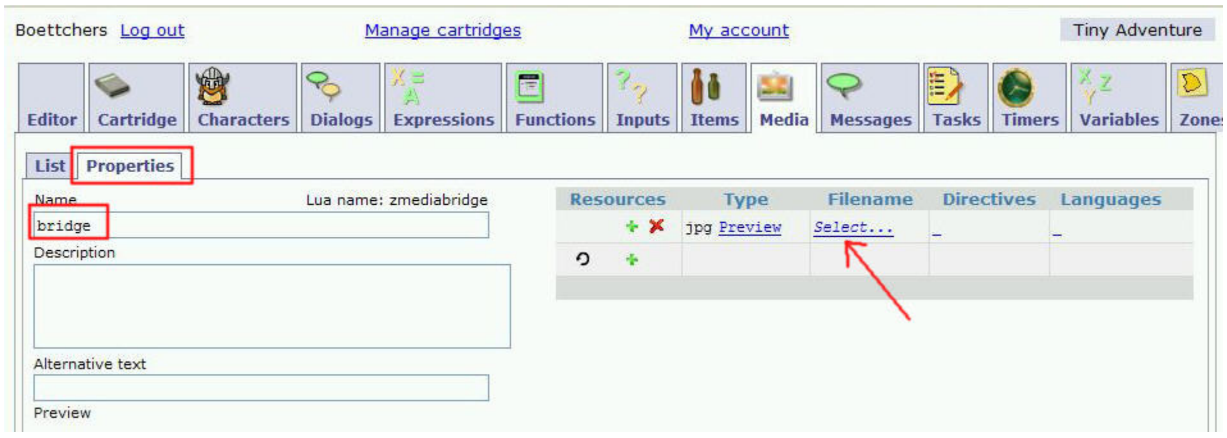
I have tried many variations of graphic sizes, only to find out that the Wherigo players re-size all pictures to fit the screen. The following sizes will typically fit the players screen and will not be resized (giving you the best possible resolution).

- Introductory image (allows app. 1 line of add. text) → 230 pixels in width and 280 pixels in height.
- Follow on image (allows several lines of text) → 230 pixels in width and 200 pixels in height.

If you need pictures in landscape orientation I suggest you use the portrait orientation instead and have the user of your cartridge rotate the player (his physical device) by 90° to look at your graphic; this will give you the highest resolution. In other words I suggest that you produce your landscape image rotated by 90°. This is similar to the images that are created when you rotate your real digital camera by 90° before you take the picture. Here is an example.

5.6.2.4 Uploading the files

For our tutorial you should have clicked on the “Properties” link behind the **bridge** Media object to go to the “Media/Properties” sub-tab, as shown in the screenshot above this section.



Name: bridge
Lua name: zmediabridge
Description
Alternative text
Preview

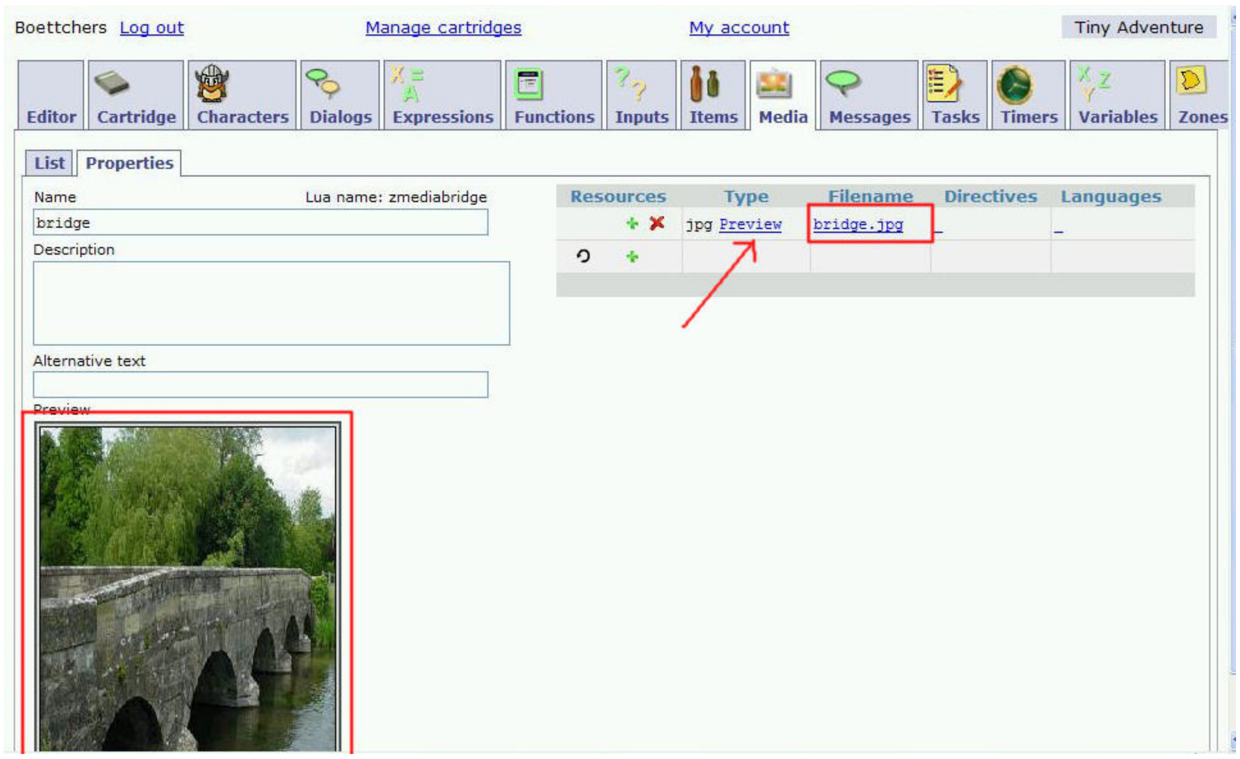
Resources	Type	Filename	Directives	Languages
	jpg Preview	Select...	_	_

On the “Properties” sub-tab for the bridge you can see a page that allows you to add media resources to your media object (don't ask me what happens if you assign more than one picture file to one media object).

- ✚Click on the green plus sign to add a media resource
- Click on “Select” to browse your files for the desired media file.
- After you have selected the media file from your computer the picture will be uploaded to the server and a preview of the image will be presented in the lower left corner of the page.

If your page does not show a “Browse” button after you have clicked on the the “Select” option, you have not enabled “Upload media files” on the “Cartridge” tab.

The following screen-shot shows the bridge “Media/Properties” sub-tab with the preview image. If you are not able to see the preview image, click on the preview link (it will obviously only work after you have uploaded your image).



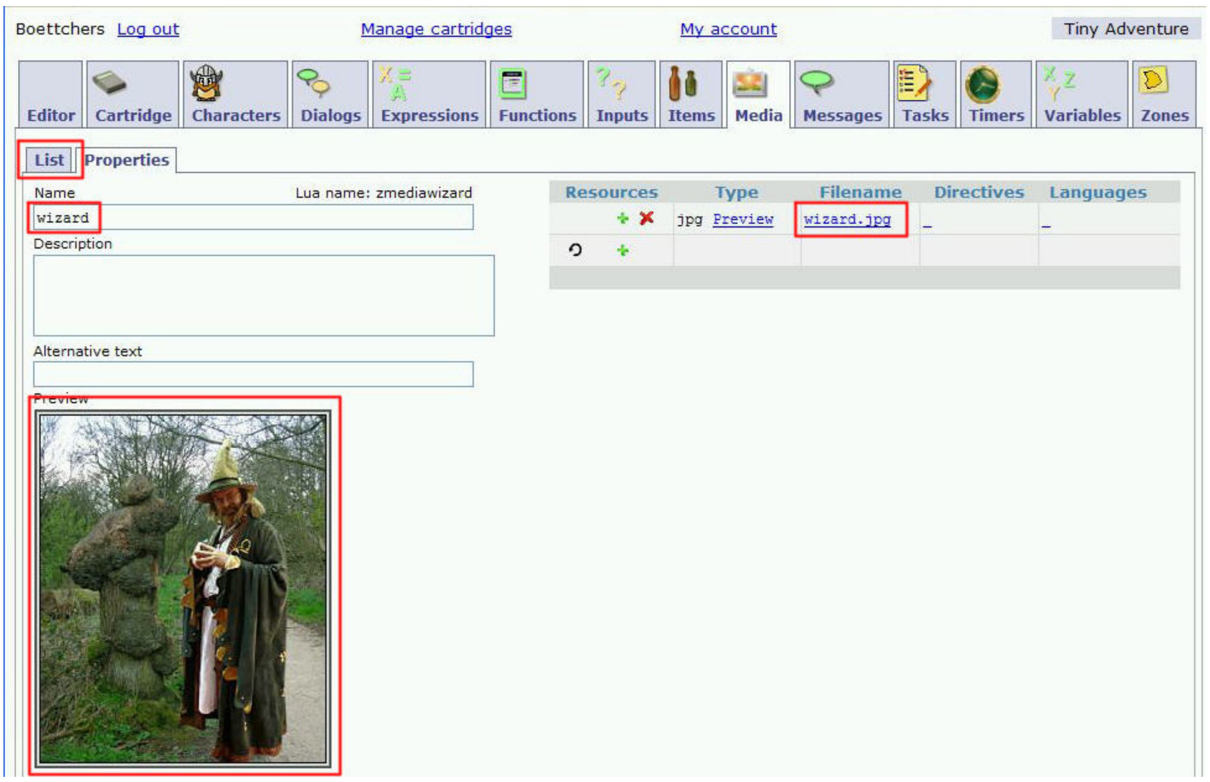
Name
Lua name: zmediabridge
bridge
Description
Alternative text
Preview

Resources	Type	Filename	Directives	Languages
✚ ✖	jpg Preview	bridge.jpg	_	_
↺ ✚				

I assume that you will find it quite easy to upload the images and therefore your next assignment in this tutorial is to upload the media files for all your media objects. Please upload the files for:

- bridge
- castle
- hut
- key
- tree
- wizard

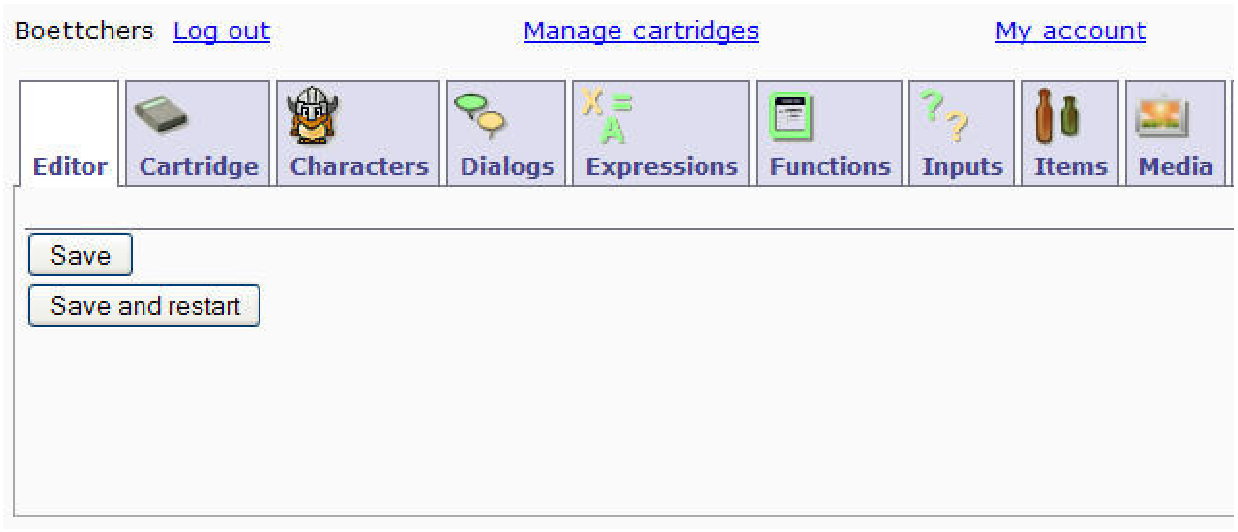
You should have no trouble to identify the appropriate files by their file names. Don't worry if you have more files than you need right now. The superfluous files will be needed later on (when we decide to beef up the cartridge a little more 😎). Your last media assignment (wizard) should look similar to the screen-shot below.



5.7 Saving the cartridge

After all the work that you have done so far, I think it is about time to make sure that your progress is saved and that a sudden computer hiccup does not cause the loss of all your data. At this stage, your changes are not yet permanently saved to the server. Go to the “Editor” tab and press the “Save” button to permanently store your data on the server.

The data will be stored under the cartridge name “Tiny Adventure” and you can select it on the Cartridge Management screen.



Working on an existing Earwigo cartridge does not permanently change the cartridge until you go to the “Editor” tab and save your changes. This comes in handy if you make a mistake and you want to go back to the starting condition of the cartridge (just go back to the manage cartridge screen and open the cartridge again). It is also quite useful if you just want to try something. It does, however, mean that you have to ensure to **save the cartridge before you close your browser** or before you navigate away from the Earwigo Web Editor. To help you to remember this, the editor will detect any attempt to navigate to another page or close the window if you have unsaved changes, and give you a chance to go back and save.

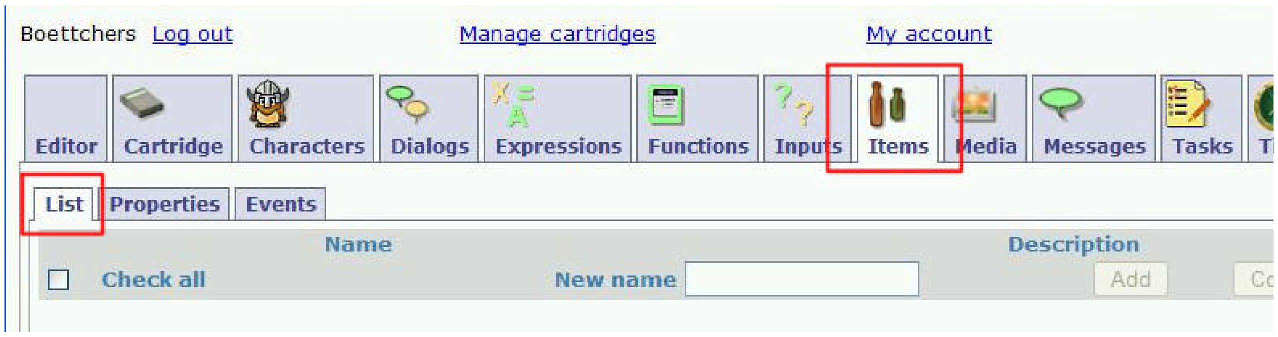
There are currently two “Save” options on this tab. The standard “Save” will just saves your changes and is the quicker option. “Save and restart” will save your changes, then reload the editor and cartridge from the server. Either “Save” should be OK; the “Save and restart” gives you the slight reassurance of working with a completely fresh copy of the editor.

5.8 Creating items

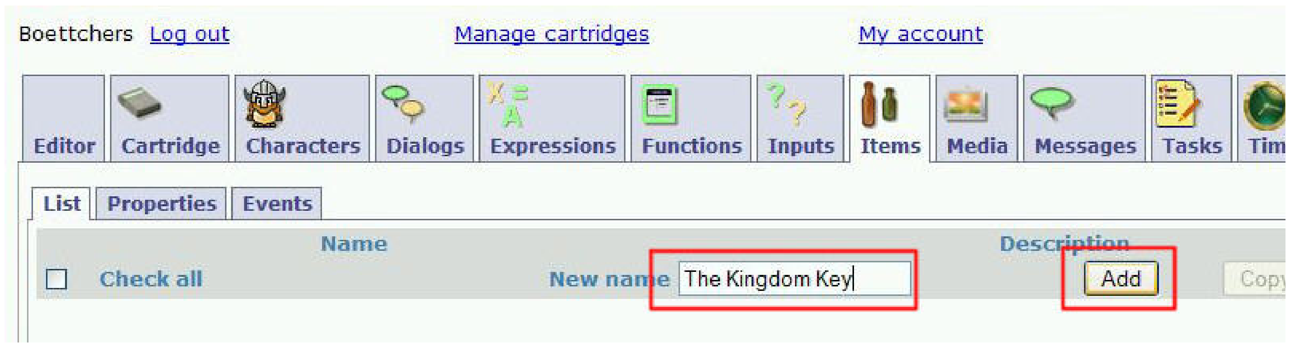
Our cartridge already has all the locations (Zones) that we will use to guide the player around in our adventure. There is however nothing in our world yet. It is about time to change this. Let's go ahead and add items to the cartridge. Items can be used in the cartridge rather like an item that you find in the real world. You can find Items in the Zones, pick them up and put them into your bag (Inventory) and you can use them alone or together with other items. However, none of these useful actions is done automatically, so you will need the ideas from this section to make Items work for you.

5.8.1 Adding items to the items list

Let us start with the creation of the “Kingdom Key”. Klick on the “Items” tab and go to the “List” sub-tab as it is shown below this text.

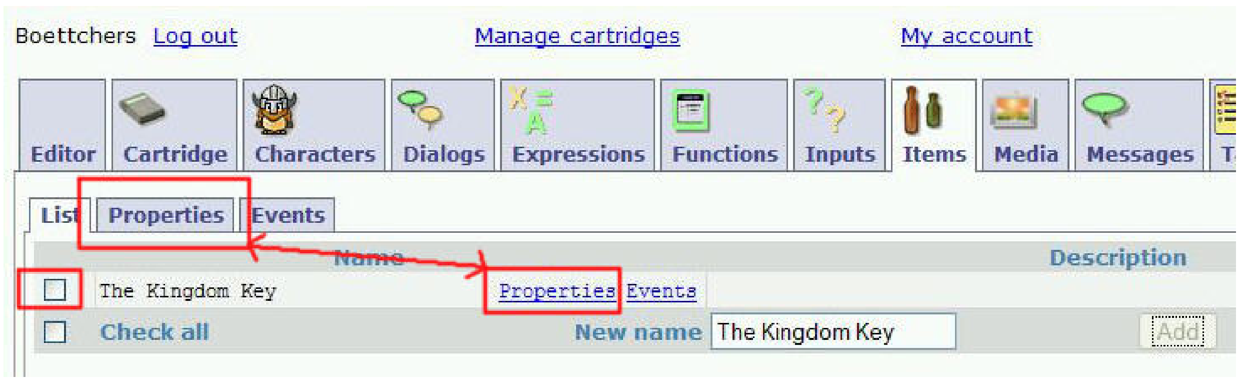


Now Add the “Kingdom Key” to the empty “Items/List”.



5.8.2 Assigning item properties

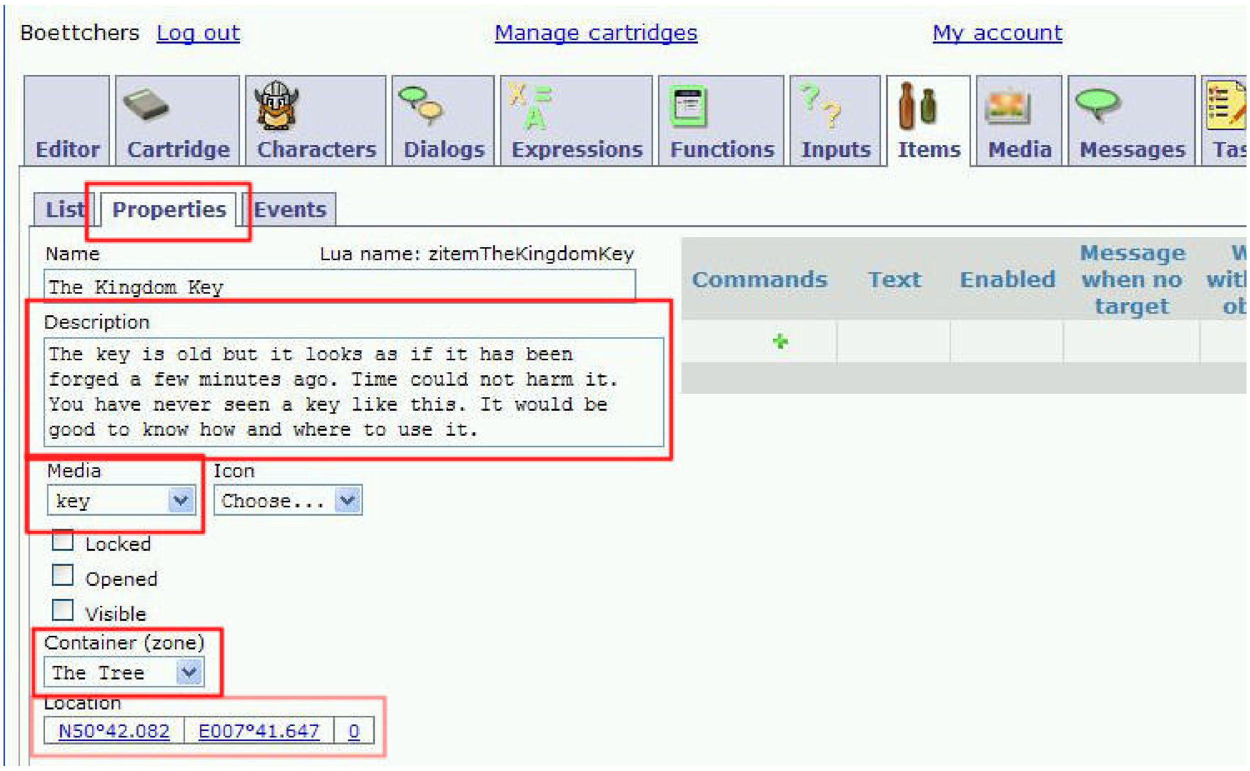
Once the “Kingdom Key” is added to the list, go to its properties page. You can do this in the same way that you did with the media objects. The quickest option is to click on the “Properties” link behind the “Kingdom Key”. You could also check the Kingdom Key's check box and click on the Items/Properties” sub-tab. The following screen-shot illustrates the process again.



Name		Description
The Kingdom Key	Properties Events	

New name: The Kingdom Key

The “Properties” sub-tab is typically where your story is put into play. This is where you communicate with your players and where you set up the locations of all items and characters in the game. I will discuss each setting of the “Kingdom Key's” properties below the following screen-shot.



Name: The Kingdom Key
Lua name: zitemTheKingdomKey

Description: The key is old but it looks as if it has been forged a few minutes ago. Time could not harm it. You have never seen a key like this. It would be good to know how and where to use it.

Media: key
Icon: Choose...

Locked
Opened
Visible

Container (zone): The Tree

Location: N50°42.082 E007°41.647 0

5.8.2.1 Item/Description

This text is displayed under the picture that is shown to the player when s/he looks at the item. Use the description to help your player understand what s/he has found. Use your imagination to breath life into your objects!

5.8.2.2 Item/Media

This dropdown menu lets you select the picture that is shown to the player to illustrate your item. Several items can “share” the same picture if you decide to do so. The dropdown list contains all the media objects that we have defined in the previous chapter of this tutorial. Please select only picture media, sound is not (yet?) handled by this procedure. You have to follow this guideline if you want to use sound in your cartridges.

5.8.2.3 Item/Locked

This is a property that you can use if your item is a box or a similar container. You can check the status of the Locked property and allow the player to open the item only if Item/Locked is set to NO. We are currently dealing with a key and do therefore not need this setting, but maybe holding this key will allow us to unlock something later!

5.8.2.4 Item/Opened

This is a property that you can use if your item is capable of being opened; for example, a door. You can check the status of the Opened property and allow the player to pass the door only if the status of Item/Opened is YES. We are currently dealing with a key and do therefore not need this setting.

5.8.2.5 Item/Visible

This property is useful if you want to hide objects and make them appear in your cartridge. If the Visible property is NO, your item will not be shown to the player (even if the item is in the same Zone as the player). You need to set Items/Visible to YES in order to let your players see the object. I have decided to hide the key in the “Tiny Adventure” and therefore the status of this property remains NO (unchecked).

5.8.2.6 Item/Container(zone)

This dropdown box allows you to position the item inside one of the zones that you have already created in the first section of this tutorial. In the “Tiny Adventure” the Kingdom Key is hidden in the roots of The Tree. Once you select The Tree as the container for the Kingdom Key the Item/Location will automatically be updated to represent the coordinates of The Tree. Both settings (Container and Location) are independent properties of your item but to the best of my knowledge different Wherigo Players use either the Container or the Location (coordinate) to determine item position and interaction. To avoid trouble with different players I like to leave the Item/Location as it is automatically updated.

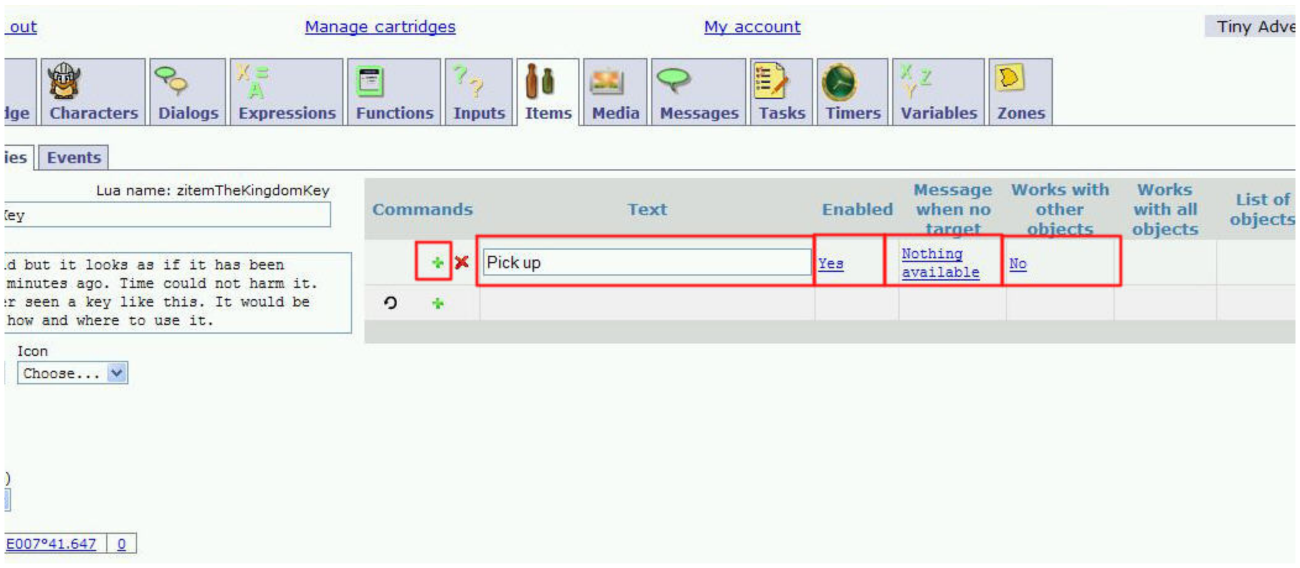
Look at the screenshot above and complete the settings for “The Kingdom Key”

1. Description
2. Media
3. Container (zone)

5.8.3 Creating Item/Commands

Items can be used together with standard Wherigo functions but there is an incompatibility of the Item's standard “On Click” event. If you build this event into your cartridge it can not be played on a Garmin device. I avoid the Item's “On Click” event mostly because I want my cartridge to be compatible with the large number of Garmin devices out there.

The second reason to avoid the "On Click" event is that there is a much nicer approach to using an item. This nicer way is called Item Commands. Item Commands are much more specific than the "On Click" event that expects your human player to click on the item on his screen out of pure curiosity. If you use an Item Command, the item will be displayed together with one or more command buttons under your item. We will learn how to use Item Commands in this section.



Lua name: zitemTheKingdomKey

Key

d but it looks as if it has been
minutes ago. Time could not harm it.
r seen a key like this. It would be
how and where to use it.

Commands	Text	Enabled	Message when no target	Works with other objects	Works with all objects	List of objects
	Pick up	Yes	Nothing available	No		

5.8.3.1 Text

Is the text that your human player will see on the button(s) under the image of your item. Consider the actions that you intend to do with your item. Each action will become a text input. If you are for example creating commands for a key, good ideas could be:

- pick up
- lock door
- unlock door
- hide

5.8.3.2 Enabled

Allows you to switch the respective command button(s) on or of. This feature is nice if you don't want to give away the future usage of your item. Events in the progress of your cartridge will give you opportunities to control this value (e.g. switch on the button "unlock door" once the player is standing in front of the door).

5.8.3.3 Message when no target

Refers to the "Works with other objects" selection box that will be described in the next paragraph. This is the text that your human player will see if none of the ***"works with" objects*** (see next paragraph) is within reach of your player. A good example for this is a key that works together with a door. If the door is not visible to the player (is not in the zone of the player) the unlock command will not fire but it will display the message that you have entered in this field. For the key example this could be: "There is no lock here".

5.8.3.4 Works with other objects

If you want to create an item that needs to be used together with other objects (characters, zones or items) you have to enable this option. This option has to be YES in order to get access to the “Works with all objects selector” and the “List of objects” (see next chapter). This option makes sure the event associated with this command does not fire when only the command is selected. The event will now only fire if the command is selected and one of the required objects is nearby and selected.

Event firing is changed from: **“command required”** to **“command plus object required”**.

5.8.3.5 Works with all objects

Select YES if any object will do the trick for you. Select NO if you want to provide a detailed list of objects (see next chapter) that can be used with your item.

5.8.3.6 List of objects

All possible objects (characters, zones or items) of your cartridge are offered in a drop down list. Select the objects by holding CTRL and clicking on the desired list entries. Your selection will be highlighted. If you want to accept the list, click to the right of the selection box. All selected objects will now be visible under the “List of objects” headline.

If the human player activates the associated command all objects on this list that are visible to him will be displayed. The human player can select the desired object. But remember the fact that there will be only one event associated with this list. Every visible object from this list will trigger the same event. You can easily see this because there is only one command event available on the events tab.

If we go back to the key example, one key could open several locks (lock1, Lock2, Lock3, old Lock, …) that are located in different zones of your cartridge. The human player would be offered only one lock per zone because all other locks are not visible to the human player. However you will have to make sure that the single event associated with this command checks the players location (zone) and opens the correct lock.

5.8.3.7 Tutorial tasks

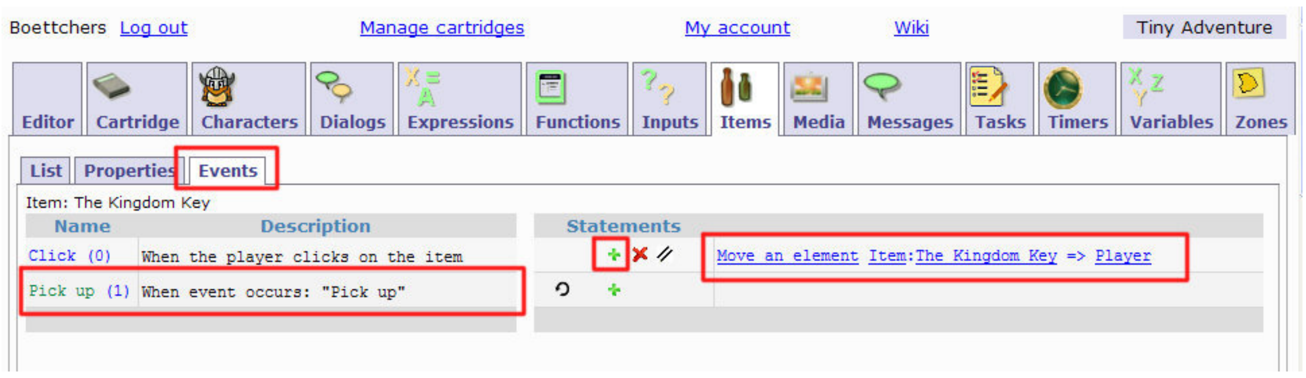
- In order to create an Item Command click on the green plus sign in the Commands section.
- For the moment the “Pick up” command is all I need you to create. We will add other options later in this tutorial.
- I plan to have this command visible as soon as the key is close to the player, therefore select Enabled:YES.
- Leave the “Message when no target” as it is. We will use the default “Nothing available”.
- For our first command in the tutorial we will not yet use “Works with other objects”, leave the default NO.

5.8.4 Item Events

You might have been wondering why we were creating the Item/Command in the previous chapter. This chapter will (hopefully) answer your question.

Commands are one of the best ways to interact with the player. The human player will see a list of commands on his Wherigo player below the object that he is currently looking at. For our example he will see a button with the command “Pick up” below “The Kingdom Key”. I hope it is obvious why we have created the command “Pick up” (to give your human player the chance to pick up the Kingdom Key 😉). However before this button has an effect, you need to associate one ore more “Statements” with the command. The “Statements” are necessary to tell the software what to do when the “Command” button is pressed (remember that the command was just a word that you typed in).

The place to connect the “Command” to “Statements” is the “Events” sub-tab. Go to the “Events” sub-tab of “The Kingdom Key”. On this sub-tab you can see that your own “Pick up” command has become an event for “The Kingdom Key”. If this event “fires” as we like to say, a specific set of statements will be executed.



Item: The Kingdom Key

Name	Description
Click (0)	When the player clicks on the item
Pick up (1)	When event occurs: "Pick up"

Statements
Move an element Item:The Kingdom Key => Player

Select your “Pick up” command from the list on the left hand side. The input matrix for program statements will come to live and offer you a green plus sign to add more statements to the command event. In our case a single line will be sufficient to tell the software what to do when the event “Pick up” fires.

Move an element Item:The Kingdom Key □ Player:(Inventory)

will do the trick. This statement tells the software to move “The Kingdom Key” inside the player's inventory. The key will be stored in the players inventory until the player decides to use the key. Which, of course, will be another “Command” and “Event” pair, similar to the one that we have just created.

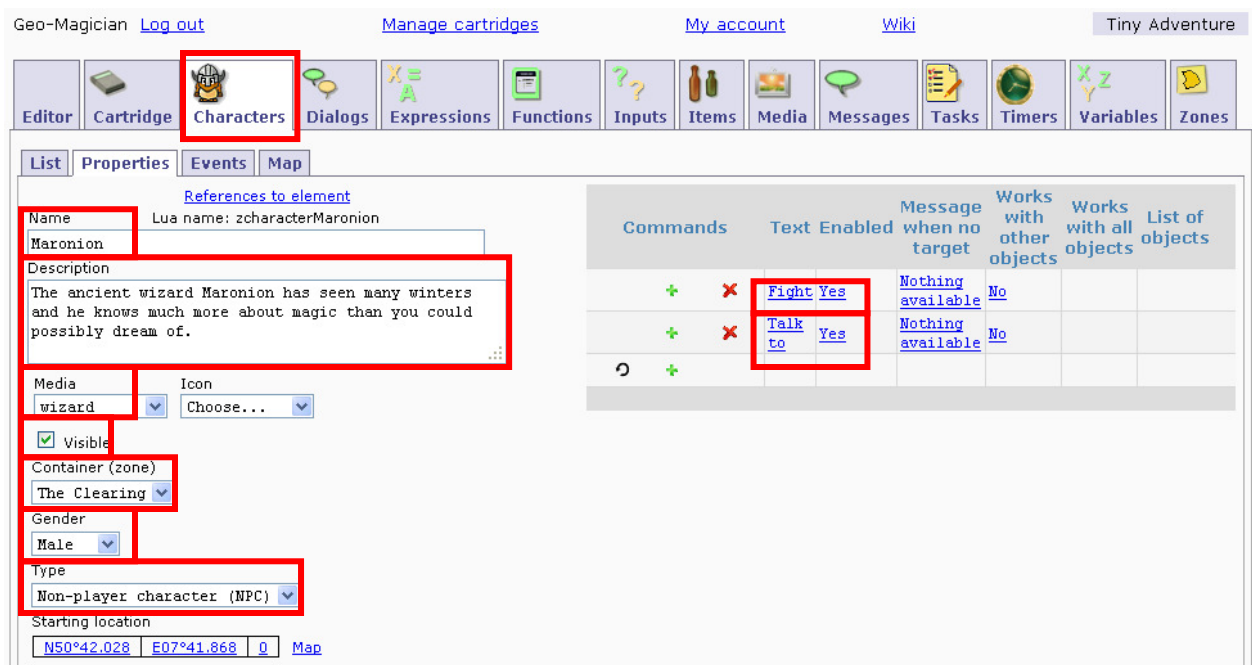
5.9 Creating Characters

Characters are your best chance to bring your cartridge to life. Populate your zones with living creatures that interact with your human player. Characters are easy to create and we will see how it works in this tutorial section.

Go to the “Characters” tab and select the “List” sub-tab. If you are following the tutorial the list should be empty. Create your first character Maronion with the Add feature. If you have followed the tutorial from the beginning, you should have no trouble to create

Maronion. If you need a reminder go to Adding Items and read how to add an item to the list.

Once Maronion is in your list, click on the “Properties” link to go to Maronions “Properties” sub-tab.



Name — Lua name: zcharacterMaronion

Maronion

Description

The ancient wizard Maronion has seen many winters and he knows much more about magic than you could possibly dream of.

Media: wizard — Icon: Choose...

Visible ✓

Container (zone): The Clearing

Gender: Male

Type: Non-player character (NPC)

Starting location: N50°42.028 E07°41.868 0 Map

Commands		Text	Enabled	Message when no target	Works with other objects	Works with all objects	List of objects
✚	✖	Fight	Yes	Nothing available	No		
✚	✖	Talk to	Yes	Nothing available	No		
↺	✚						

You will recognize that Character “Properties” and Item “Properties” are very similar. The following is almost a copy of the text that I have used for Items. If you feel comfortable with the Properties tab, enter the values depicted above and take this short cut.

Since you are still reading, here is the explanation for the values that can be found on the Character's Properties tab.

5.9.1 Character Properties

5.9.1.1 Character/Description

This text is displayed under the picture that is shown to the player when s/he looks at the character. Use the description to help your player understand who is the person, ghost, wizard,… that he has just met. “You see a wizard” will do but “The old wizard is fragile and you can tell that he has served his king faithfully for most of his life. He might look weak but if you rely on him, he will not let you down. Whatever the cost!” will introduce your character AND create a feeling for your creature. Feelings are essential for Wherigo cartridges because the imagination of your human player is your biggest asset 😉.

5.9.1.2 Character/Media

This dropdown menu lets you select the picture that is shown to the human player to illustrate your character. Several characters can “share” the same picture if you decide to do so. The dropdown list contains all the media objects that we have defined in the previous chapter of this tutorial. Please select only picture media, sound is not (yet?) handled by this procedure. You have to follow this guideline if you want to use sound in your cartridges.

5.9.1.3 Character/Visible

This property is useful if you want to hide characters and make them appear in your cartridge. If the Visible property is NO, your character will not be shown to the player (even if the character is in the same Zone as the player). You need to set Character/Visible to YES in order to let your players see your creatures.

It would be a nightmare if all visible characters would be displayed to your human player. Characters are therefore automatically hidden from view through the "Show_objects:_On_enter" capability of a zone. This (default) selection of the zones will hide characters from the human player until he enters the zone that contains the character. We will be using the "Show_objects:_On_enter" feature and can therefore select "Visible: YES" (Checkmark). The combination of the character's "Visible" and the zone's "Show_objects:_On_enter" makes sure that the character will be visible in the moment that our human player enters the zone that contains the character.

5.9.1.4 Character/Container(zone)

This dropdown box allows you to position the Character inside one of the zones that you have already created in the first section of this tutorial. Maronion lives in the hut, place him in the zone "The Hut". Once you select the zone, the Character/Location will automatically be updated to represent the proper coordinates.

Wherigo is zone oriented and you will have no need to change the Character/Location by hand in order to program a cartridge. The location is automatically updated and I suggest that you restrict yourself to zones for your first cartridge (Wherigo allows you to misalign the character's zone and his location and that is where things get really fuzzy → avoid it for your first cartridge, or read some forums to get a better understanding of the concept 😉

5.9.1.5 Character/Gender

The gender of the character. I have no good idea how to use this to our advantage right now but it gives you a better description of your characters. This might come in handy if you intend to check the gender of a character before you start certain actions.

5.9.1.6 Character/Type

I guess this property will be used in future cartridges to enable linked Wherigo games. For the time being "non-player-character" is my choice.

5.9.1.7 Tutorial tasks (character creation)

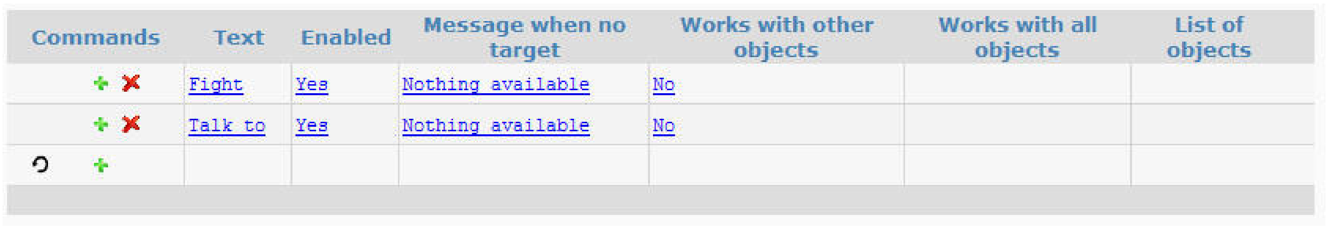
Look at the screen-shot above and complete the settings for "Maronion"

1. Description
2. Media
3. Visible
4. Container (zone)
5. Gender
6. Type

5.9.2 Character Commands

You are almost done with the creation of Maronion but I intend to tell you how our human player can interact with our wizard Maronion. We will use Commands to offer “button” commands to our human player. The idea of the buttons is to offer a choice of actions and see if our human player uses a sensible approach to interact with the powerful wizard. We will program this interaction in the following sections of the tutorial. Enter the Commands:

1. Fight (Wound the player when he chooses the Command “Fight” and send him back to the bridge to get cured from his wounds).
2. Talk to (Let the player continue the cartridge with the Command “Talk to” by revealing the location of the tree).



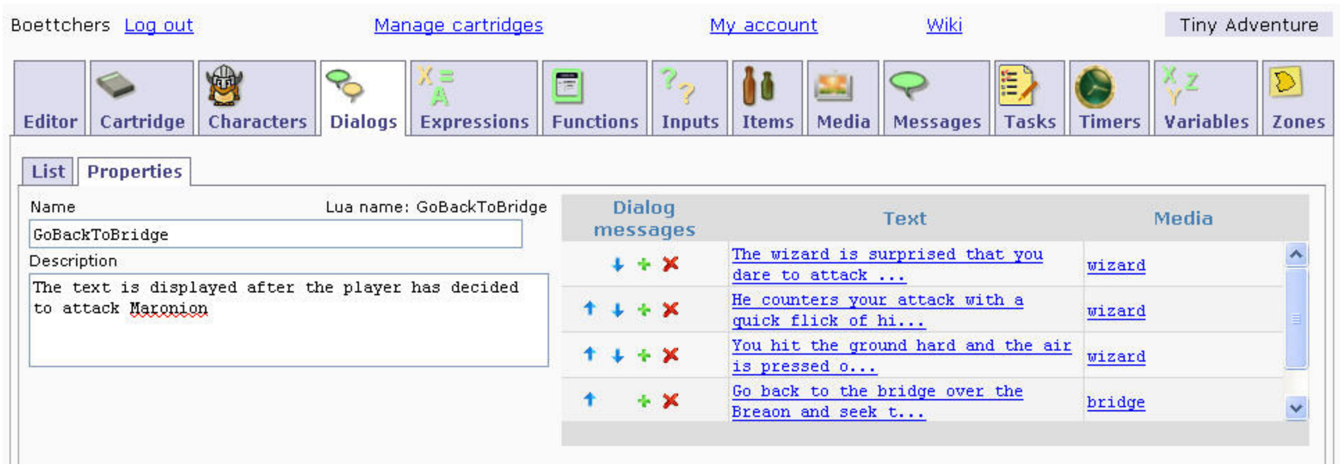
Commands	Text	Enabled	Message when no target	Works with other objects	Works with all objects	List of objects
✚ ✖	Fight	Yes	Nothing available	No		
✚ ✖	Talk to	Yes	Nothing available	No		
✚						

Check this section if you need a reminder how to enter Commands.

5.10 Creating Dialogs

If our human player is bold enough to attack Maronion we need to tell him, that he has selected the wrong approach. Talking to the human player is done via Messages or Dialogs. We will use the Dialog for the “Fight” option.

The Dialog offers as many text fields as you might need. Use Dialogs when you have a larger portion of text that you prefer to present in digestible chunks. The Dialog also offers you the opportunity to change the media for each individual text portion. The disadvantage of Dialogs is that they do not offer any buttons (and no events).



Name: GoBackToBridge
Lua name: GoBackToBridge
Description: The text is displayed after the player has decided to attack Maronion

Dialog messages	Text	Media
↓ ✚ ✖	The wizard is surprised that you dare to attack ...	wizard
↑ ↓ ✚ ✖	He counters your attack with a quick flick of hi...	wizard
↑ ↓ ✚ ✖	You hit the ground hard and the air is pressed o...	wizard
↑ ✚ ✖	Go back to the bridge over the Breaon and seek t...	bridge

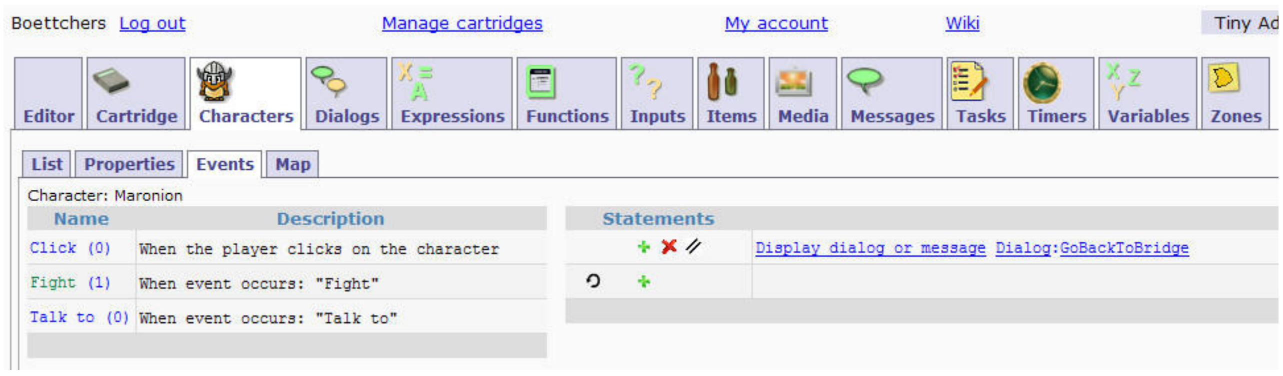
For our adventure I have chosen to use the Dialog after the fight because the text that I want to display is a little longer and it feels better if a longer text is displayed in smaller chunks (text >850 characters might actually crash your cartridge, read this FAQ for details).

Your next task for the tutorial is to create a Dialog **GoBackToBridge** that consists of 4 text portions:

- The wizard is surprised that you dare to attack him but he is obviously not concerned at all.
- He counters your attack with a quick flick of his wrist and sends you flying through the air.
- You hit the ground hard and the air is pressed out of your lungs. Your body is badly bruised and hurts terrible. You are in no condition to continue the adventure any longer.
- Your only hope is to go back to the bridge over the Breaon and seek the assistance of the river spirit. She will restore you and you will be fit to try the adventure again.

This is the Dialog that we intend to present when the human player selects the “Fight” option. I hope you have already guessed how we will do this… We will once again use the events that are automatically created through the creation of the Commands (Fight and Talk). Go to Maronion's Event's sub tab and connect the Dialog to the Fight button Event.

- Click on “When event occurs: “Fight” and enter the Statement “Display dialog or message Dialog:GoBackToBridge” as depicted below.



Character: Maronion

Name	Description
Click (0)	When the player clicks on the character
Fight (1)	When event occurs: "Fight"
Talk to (0)	When event occurs: "Talk to"

Statements
Display dialog or message Dialog:GoBackToBridge

5.11 Decisions (Introduction)

In the previous chapter we have given our human player a choice how to interact with our wizard Maronion.

The human player had the folowing choices

- (A) Fight: Wound the player when he chooses the Command “Fight” and send him back to the bridge to get cured from his wounds.
- (B) Talk: Let the player continue the cartridge with the Command “Talk” by revealing the location of the tree.

We have already dealt with option (A) and you might think we could do (B) in a similar manner:

Display dialog or message Message:RevealTreeLocation
Set value Zone:The Tree:Visible => true
Set value Zone:The Tree:Active => true

Don't do this yet, please read on!

The above version would almost work but there is one major flaw in the above “program”. The following sequence of events can happen if we don't do some “programing” to avoid it:

- Our human player could decide to fight (by choosing the appropriate Command/button “Fight”,
- He would be sent back to the bridge as desired
- But instead of walking the way back to the bridge our player could decide to approach Maronion again and select the Command/button “Talk” this time.
- The Event “Talk” would be executed and our player could just walk to the tree without any trouble. This is obviously not what we want to happen (he should have walked to the bridge first to get re-vitalized).

If a human being would be in charge of the cartridge instead of Wherigo the human could have avoided this easily but he would have used two things that we have not yet programed into the “Tiny Adventure”.

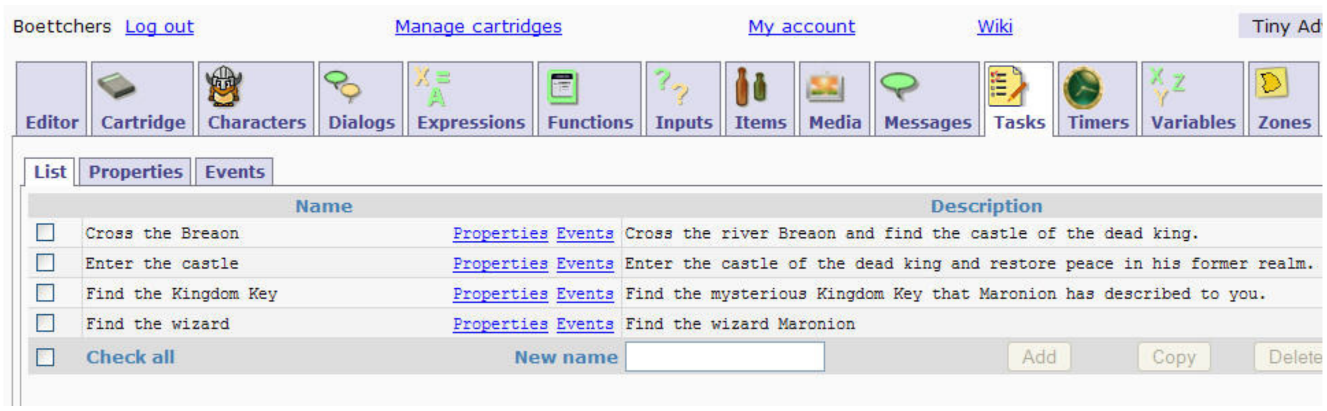
- The human would **remember** that our player has pressed the fight option first
- The human would make a **decision** based on his memory (let the player walk to the tree or tell him that he is supposed to walk back to the bridge)

We have to do the same for our cartridge.

- We need the cartridge to **remember** that the player has already pressed the “Fight” button. We will use a Task to do this.
- The **decision** is called an ”IF Then Else” statement and we will use it (IF Then Else) once we have dealt with remembering.

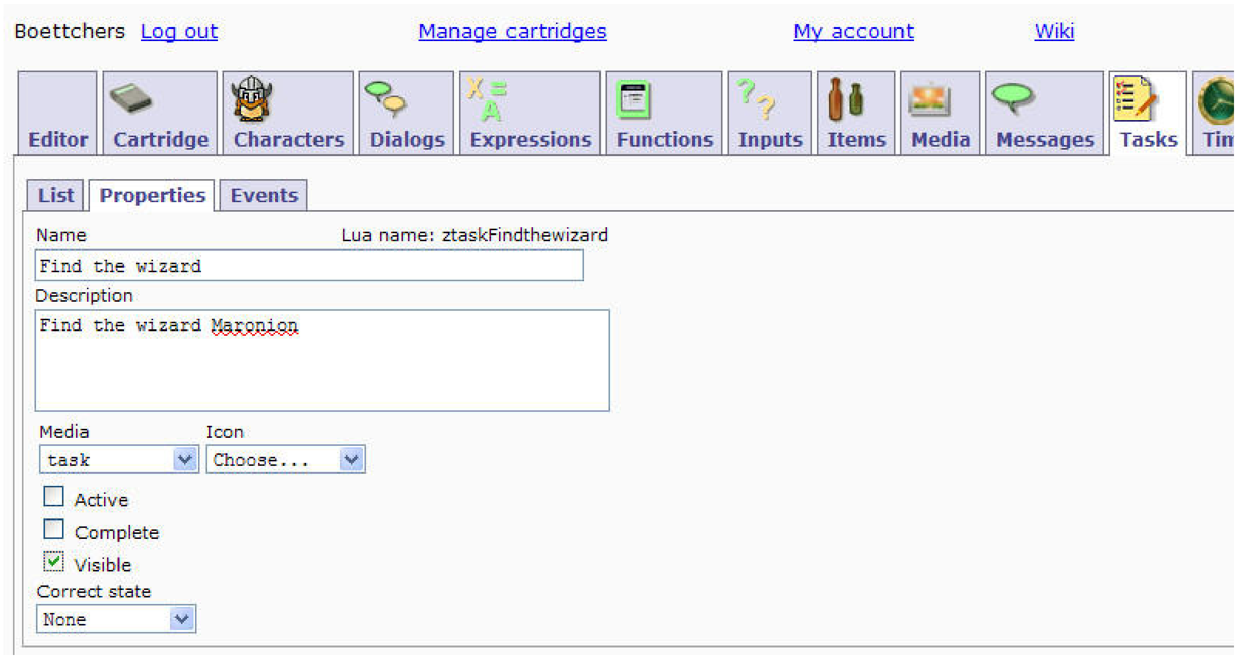
5.12 Working with Tasks

We will create a task “Find the wizard” to **remember** the users reaction when he meets Maronion. Create this and all other Tasks depicted in the screen-shot below. Remember to switch to each task's property tab to enter the task descriptions.



Name		Description
Cross the Breaon	Properties Events	Cross the river Breaon and find the castle of the dead king.
Enter the castle	Properties Events	Enter the castle of the dead king and restore peace in his former realm.
Find the Kingdom Key	Properties Events	Find the mysterious Kingdom Key that Maronion has described to you.
Find the wizard	Properties Events	Find the wizard Maronion

The “Find the wizard” Task has the following Properties:



Name Lua name: ztaskFindthewizard
Find the wizard
Description
Find the wizard Maronion
Media: task
Icon: Choose...
☐ Active
☐ Complete
☑ Visible
Correct state: None

5.12.1.1 Active

A task can be active (true) or inactive (false). You should make a task active when you want the player to work on it. Tasks show up in the task list only when they are both active and visible. My guess is that events are not processed for inactive tasks. But that's just a guess. We will leave the task inactive for now because we are currently only preparing the cartridge. We will change this value to active during the course of the cartridge.

5.12.1.2 Complete

A task can be complete (true) or incomplete (false). You can use this property to track whether or not the player has completed the task. Otherwise, the status does not affect the visibility of the task. The initial value of "Complete" is (false). We will set this value to (true) once the player has successfully talked to Maronion.

5.12.1.3 Visible

An active task can be visible (true) or not visible (false). This controls visibility in the task list for active tasks. Inactive tasks are never visible. We will leave the task invisible for the time being because the meeting with Maronion will happen in the future (the course) of our cartridge.

5.12.1.4 Correct State

A task can have one of three "Correct State" values which reflect its correctness: "None" (unknown), "NotCorrect" and "Correct". You can use this property to track the outcome of a completed task in more detail. We will use this value to remember how the player interacted with Maronion. We will set the value to

- None: as long as the player has not yet reacted to the sudden appearance of Maronion
- NotCorrect: if the human player decides to fight against the mighty magician
- Correct: if the human player decides to communicate with the mage

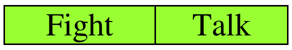
If you are a mighty programmer you will know that we could have used a variable to remember the user choice (fight or talk). I prefer to use the Task because it is displayed in the player software and because tasks can be used to guide the user in his adventure.

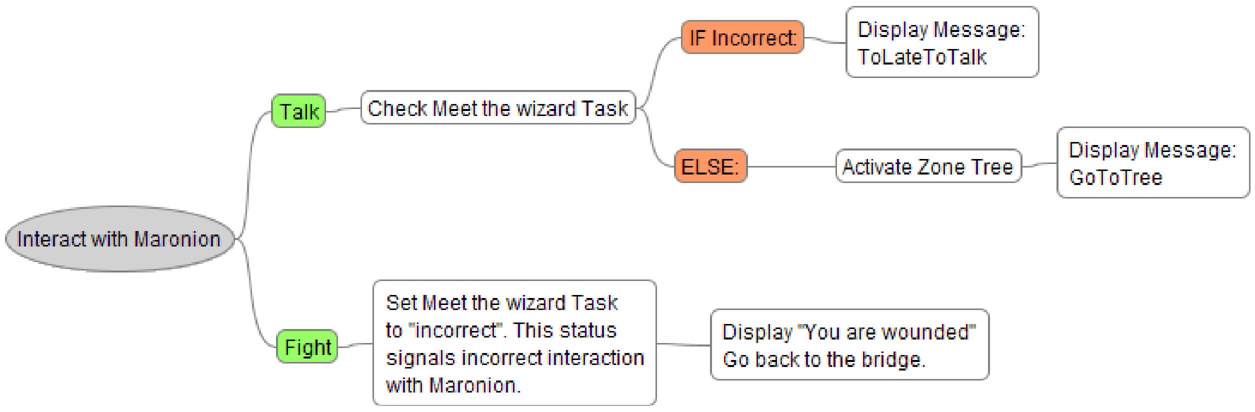
5.13 Decisions (Continued)

You have just learned that we can use a task to **remember** things that have happened in the cartridge (we are using the Task's Correct property to remember the player's action). Memory or remembering something was the first item that we had to accomplish in order to program our “Player meets Maronion” scenario. The other item was to be able to make a **decision**. Decisions in programs are called IF THEN ELSE statements.

IF THEN ELSE statements need a little brain work before you can program them into the WWB. You should typically decide what you intend to do before you start the IF THEN ELSE statement. The best way to sort things out is a decision tree, it will help you to create the logic concept of your idea. Don't panic (yet), this is just one way to look at things.

We have 2 possible selections (buttons):




Interact with Maronion
Talk
Check Meet the wizard Task
IF Incorrect:
Display Message: ToLateToTalk
ELSE:
Activate Zone Tree
Display Message: GoToTree
Fight
Set Meet the wizard Task to "incorrect". This status signals incorrect interaction with Maronion.
Display "You are wounded" Go back to the bridge.

Fight will:

- display a Dialog "GoBackToBridge" that informs the player that Maronion fought back
- set the Task "Meet the wizard" to "incorrect" to remember the human player's incorrect re-action

Talk will:

- check the "Meet the wizard" Task to make sure that the human player has not already tried to fight Maronion (in this case the "Meet the wizard" Task would be "incorrect", see above)
- If incorrect
  - it will display a message ToLateToTalk to tell the human player that his attack is re-membered and that he needs to go back to the bridge.
- Else
  - it will display a Dialog GoToTree
  - it will activate the Zone "Tree"

These actions will once again be programmed on Maronion's Events tab. However, be-fore we can program the new events, we need to prepare the items that we will be using during the event programming. If you look at the decision tree above you will recognize that we need to:

5.13.1 ToDo List

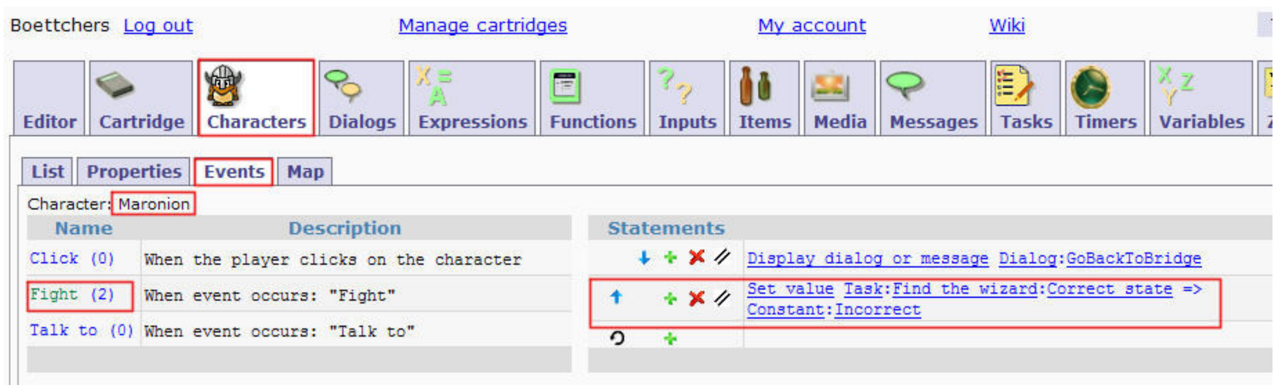
1. create a Dialog “GoBackToBridge” (complete)
2. set the Task “Meet the wizard” to incorrect
3. check the Task “Meet the wizard” for the property Correctness=incorrect
4. display a Message “ToLateToTalk”
5. display a Dialog “GoToTree”
6. activate the Zone “Tree”

The ToDo list will help us keep track of what we need to do. You will find a reference to the list at the beginning of each of the following sections. The next 5 sections of the tu-torial will deal with completing the items from the list above. **We will come back to the IF THEN ELSE statement** once all items from the list are complete.

5.14 Changing Properties - Set Value

ToDo List 2)

We have decided that we will have to change the status of the “Meet the wizard”'s “Correctness” property to incorrect when the button “Fight” has been pressed. The key to this is the event that was automatically created when we designed the buttons “Talk to” and “Fight”. Both events can now be found on Maronion's Events tab.



Character: Maronion

Name	Description
Click (0)	When the player clicks on the character
Fight (2)	When event occurs: "Fight"
Talk to (0)	When event occurs: "Talk to"

Statements
Display dialog or message Dialog:GoBackToBridge
Set value Task:Find the wizard:Correct state => Constant:Incorrect

Add the Statement “Set value Task:Find the wizard:Correct state Constant:Incorrect” to Maronion's “Fight” button Event.

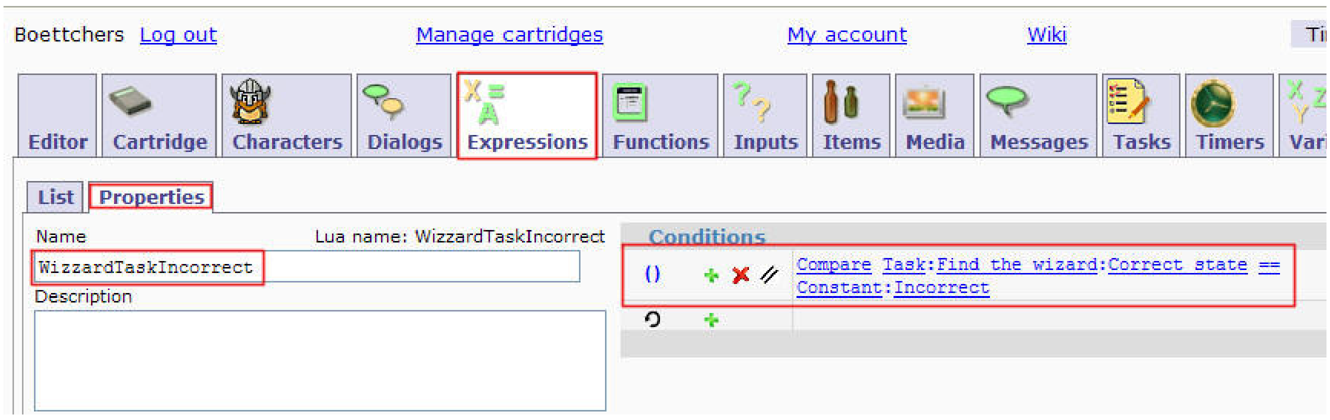
5.15 Creating Expressions

ToDo List 3)

If you want to compare things or if you want to analyze values, you have to use an Expression in Earwigo. The Expression represents the comparison and can be used whenever you need it.

The typical use of an expression is inside an IF THEN statement (IF Expression THEN …). With other words: any IF THEN statement that you intend to use in the WWB needs an Expression (and that is the reason why I ask you to create an expression in this tutorial 😉).

We are preparing an IF THEN statement that will check which button our human player has pressed when we gave him the choice to attack the wizard or talk to him. I hope that YOU remember that we programmed the Wherigo player software to remember this fact through the **Task** “Meet the wizard” (quick reminder). If we want to find out that the human player has attacked the wizard, we need to check the “Meet the wizard” **Task** and look at the **Correctness** property. If Correctness is “incorrect”, we know that our human player was stupid enough to attack a magician .



Create the Expression "WizzardTaskIncorrect" as depicted above.

For our tutorial we do only need the comparison "is equal to" == but there are a few other comparisons that I should mention:

• A == B	A is equal to B
• A ~= B	A is not equal to B
• A > B	A is larger than B
• A < B	A is smaller than B
• A >= B	A is larger or equal to B
• A <= B	A is smaller or equal to B

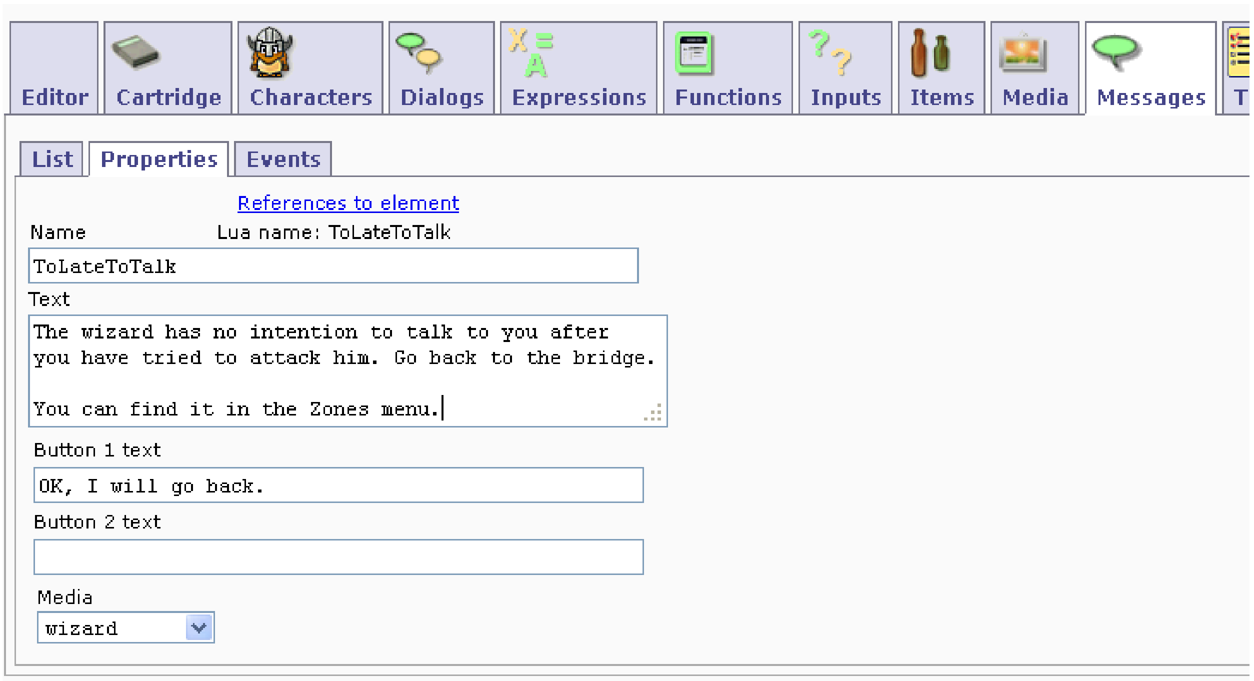
5.16 Messages and Message Buttons

ToDo List 4)

The next item on our ToDo list is a message that is presented to the human player to inform him that he can not talk to Maronion after he has decided to attack the wizard. The Message is named "ToLateToTalk".

A Message offers only one text field that can contain a maximum of 850 characters (text strings larger than 850 characters might crash your cartridge, read this FAQ for details).

The benefit of a Message is that it also offers 2 buttons (and the respective event). You can use the buttons to communicate with the player. Use Messages for short Instructions or when you need to get a reaction from the player (button press). If you are using Buttons you should typically also go to the Events tab of you message. You will find only one event ("When a button is pressed") that controls the functionality of the buttons. Use this event to program your button actions. I will not go into details now but if you intend to use more than one button, you will most likely need an Expression (Button was clicked Button 1) and an IF THEN ELSE statement. **AB HIER WIKI UPDATE NOTWENDIG**



Name Lua name: ToLateToTalk

ToLateToTalk

Text

The wizard has no intention to talk to you after you have tried to attack him. Go back to the bridge.

You can find it in the Zones menu.

Button 1 text

OK, I will go back.

Button 2 text

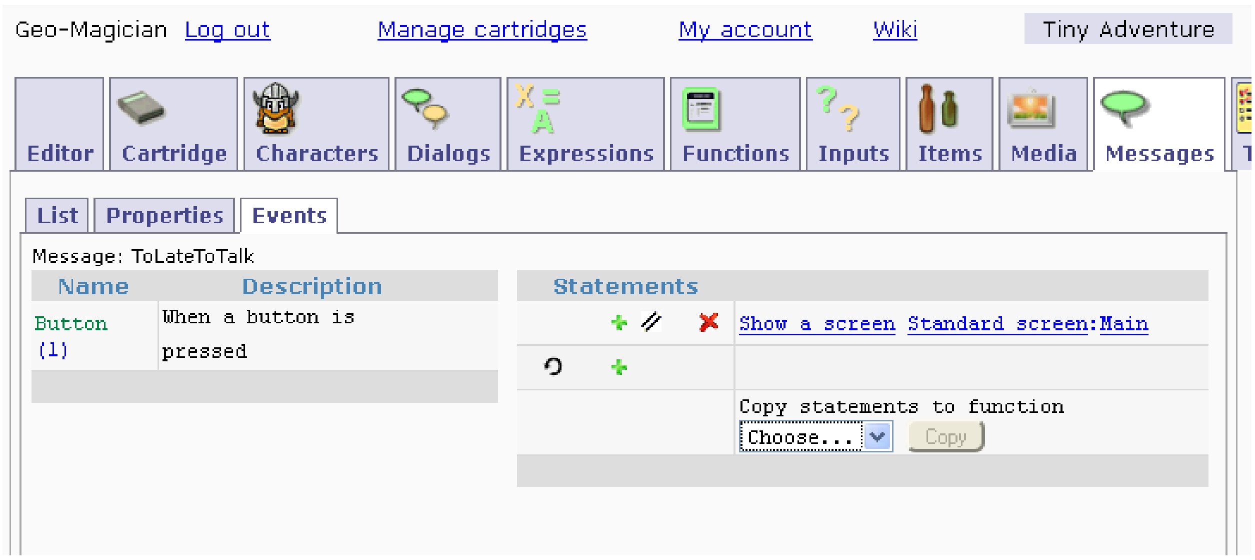
Media

wizard

We will use the above message to inform the human player that Maronion is not interested to talk after he has been attacked. We will use only one Message button right now. You can use more buttons but it is to early for this in the tutorial. So your next tutorial task is to create the message “ToLateToTalk” and enter the Properties depicted above.

5.16.1.1 Message Button Event

The buttons are handled on the Events tab of the message. Since we are only using one button we can use the button event without the need to check for the pressed button. Since there is only one butto the Event “When a button is pressed” is always invoked by our “OK, I will go back” button. We will react on this button by presenting the direction to the bridge (which is our Zone “Start”) to our human player.



Message: ToLateToTalk

Name	Description
Button (1)	When a button is pressed

Statements

Show a screen Standard screen:Main

Copy statements to function

Choose... Copy

You might be wondering why we have not used the more convenient Wherigo command: “Show_a_screen Zone:Start”. Good thinking but due to a Garmin bug this is not a good idea. Since we want to keep our cartridge usable for all players we have used a slightly different approach to ensure the cartridge will work on all devices.

Only one Show_a_screen command is save to be used. It is the command "Show_a_screen Standard_screen:Main". This command will show the home or main menu page. The home page will not directly guide the human player back to the starting position but we can point him towards it with a little extra work. The idea is to present a message with instructions followed by the "Show_a_screen Standard:Main" command. If you have a look at the "ToLateToTalk" message you can see that we have already provided the required extra information for the human player. All we need to do now is to present the main menu and hope that the player can find the described page (they typically do ;-)

5.17 Simulating Interaction (Dialog)

ToDo List 5)

5.17.1 Concept

The last but one item on our ToDo list is a Dialog that informs our human player about the location of the tree. We have already talked about Dialogs in this tutorial. I will therefore use this Dialog to show you a a little trick that I use to create a little more reality in my cartridges. The concept is quite simple but I like to use the effect.

We will create a dialog that "predicts" the interaction of our human player. The dialog is presented by switching between two media (images). One image is the magician Maronion, the other image is used to represent our human hero. This dialog looks nicer on the Wherigo player but I hope you get the idea:

5.17.2 Demonstration

You carefully approach the wizard and kindly ask him

	Excuse me sir, would you mind to spare a minute and answer a few questions?
	Not at all young one. I am delighted too meet you and I do for sure have the time to help you out. What is it?
	Puuuhhh… I am glad that you are such a nice wizard. I was afraid you would get angry and start to …

	Why should I get angry, you are kind enough and I see no reason to harm you.
	That is a relief. May I ask you straightaway a strange question?
	Go ahead, let me see what you want to know.

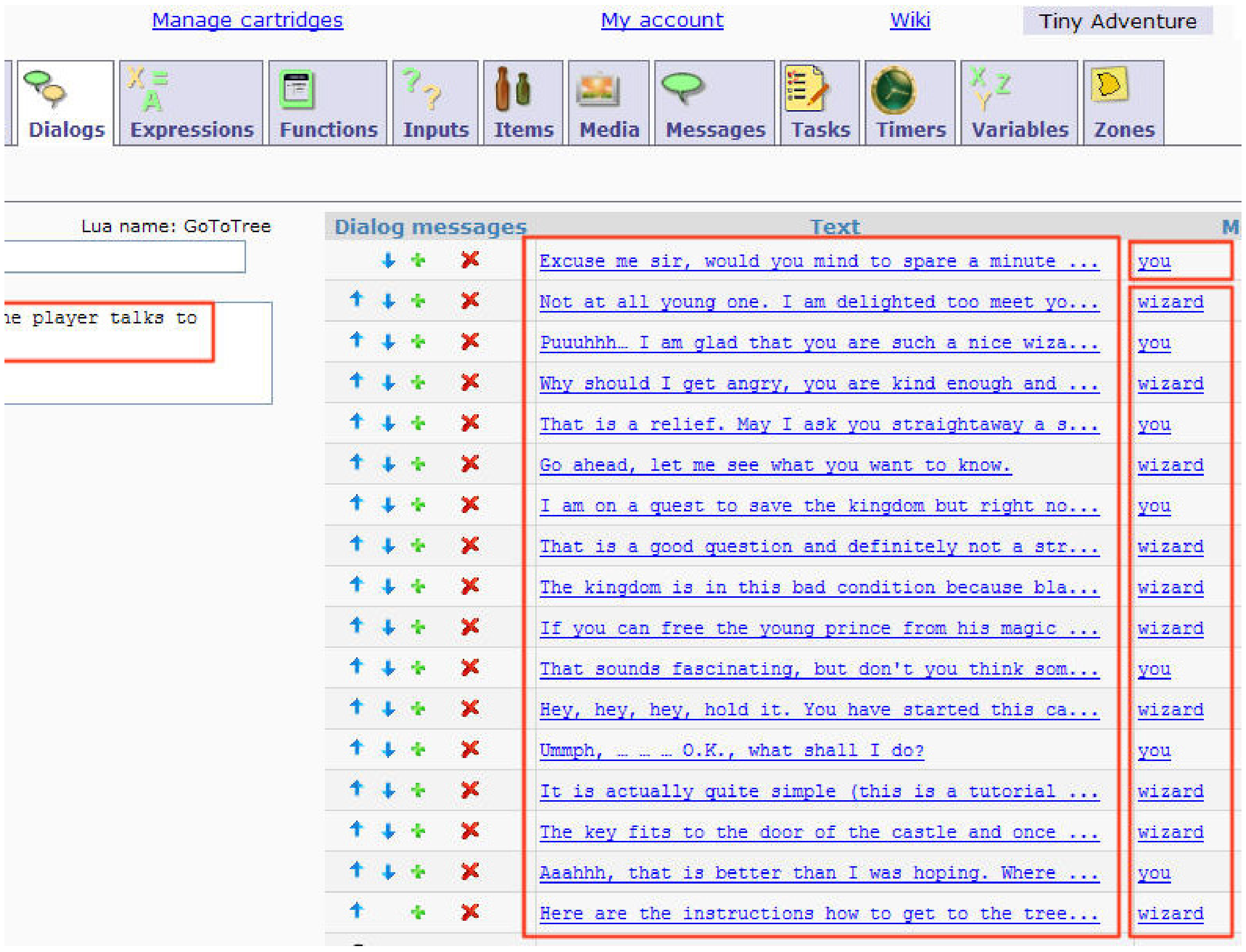
	I am on a quest to save the kingdom but right now I have no clue at all how I could possibly do that. Do you have any idea where I should start?
	That is a good question and definitely not a strange one. The kingdom desperately needs some help. I am glad that you have asked me.
	The kingdom is in this bad condition because black magic has killed the old king and no one is currently sitting on the kingdom throne.

	If you can free the young prince from his magic dungeon you will do a great service to our nation.
	That sounds fascinating, but don't you think someone else should do this “hero thing”? Magic dungeon sounds dangerous, I think I should better get home quick. It was a pleasure to meet you.
	Hey, hey, hey, hold it. You have started this cartridge, now you have to finish it. Do you think I want to stand next to this old tree for the next decade?

	Ummph, … … … O.K., what shall I do?
	It is actually quite simple (this is a tutorial after all 😉). Go to the magic tree and retrieve the kingdom key.
	The key fits to the door of the castle and once you unlock the door the prince will be free and rule the kingdom. That is all.

	Aaahhh, that is better than I was hoping. Where can I find this tree?
	Here are the instructions how to get to the tree. Keep them and you should have no trouble to reach the tree.

5.17.3 Interaction Dialog in Earwigo



Dialog messages	Text	M
	Excuse me sir, would you mind to spare a minute ...	you
	Not at all young one. I am delighted too meet yo...	wizard
	Puuuhhh… I am glad that you are such a nice wiza...	you
	Why should I get angry, you are kind enough and ...	wizard
	That is a relief. May I ask you straightaway a s...	you
	Go ahead, let me see what you want to know.	wizard
	I am on a quest to save the kingdom but right no...	you
	That is a good question and definitely not a str...	wizard
	The kingdom is in this bad condition because bla...	wizard
	If you can free the young prince from his magic ...	wizard
	That sounds fascinating, but don't you think som...	you
	Hey, hey, hey, hold it. You have started this ca...	wizard
	Ummph, … … … O.K., what shall I do?	you
	It is actually quite simple (this is a tutorial ...	wizard
	The key fits to the door of the castle and once ...	wizard
	Aaahhh, that is better than I was hoping. Where ...	you
	Here are the instructions how to get to the tree...	wizard

Create the Dialog “GoToTree” as shown above. Don't forget to create the Media object “You” before you work on the dialog. I think you are smart enough to copy and paste the media text from the example above, but it can't hurt to mention this option ☺

5.18 If Else EnfIF Statements

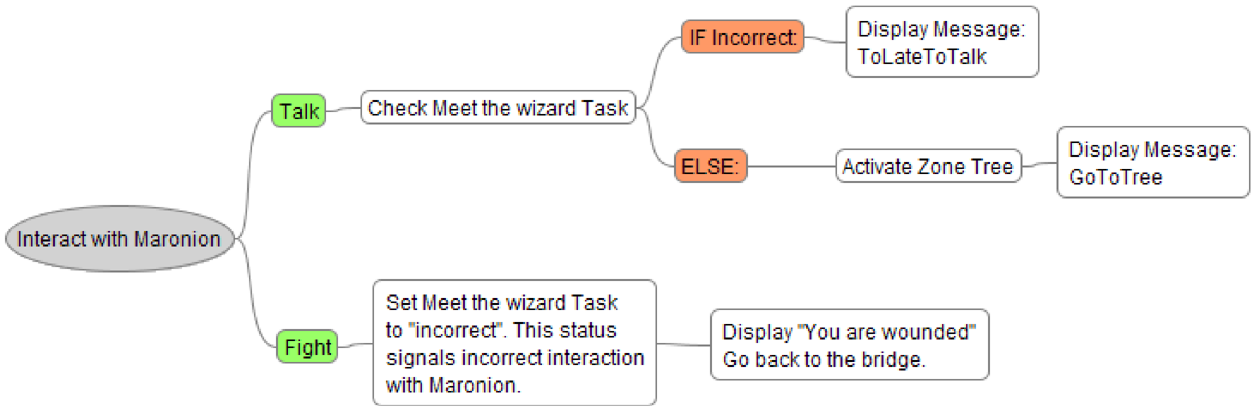
ToDo List 6)

Item 6 on the ToDo list is to enable the Zone “Tree”. I know that you can enable Zones by now (Set_value Zone:Tree:Active true) but this time we have to remember a condition. We have started the last 5 sections of this tutorial because we need to built an IF ELSE EndIF Statement. You might want to read this section again to remember what we intend to achieve.

5.18.1 Flashback (a message from the past)

The situation that our human player currently faces is:

- He has walked to a clearing in the woods
- He has found the wizard Maronion
- He has 2 options to interact with Maronion (we have created two buttons)
    - Fight
    - Talk


IF Incorrect:
Display Message: ToLateToTalk
Talk
Check Meet the wizard Task
ELSE:
Activate Zone Tree
Display Message: GoToTree
Interact with Maronion
Set Meet the wizard Task to "incorrect". This status signals incorrect interaction with Maronion.
Fight
Display "You are wounded" Go back to the bridge.

Fight will:

- display a Dialog "GoBackToBridge" that informs the player that Maronion fought back
- set the Task "Meet the wizard" to "incorrect" to remember the human player's incorrect reaction

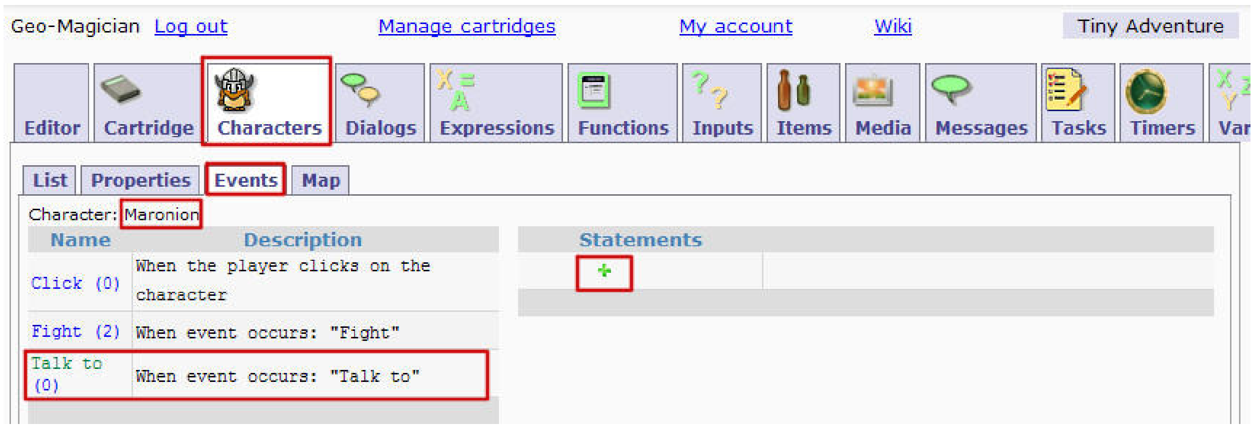
Talk will:

- check the "Meet the wizard" Task to make sure that the human player has not already tried to fight Maronion (in this case the "Meet the wizard" Task would be "incorrect", see above)
- If incorrect
    - it will display a message ToLateToTalk to tell the human player that his attack is remembered and that he needs to go back to the bridge.
- Else
    - it will display a Dialog GoToTree
    - it will activate the Zone "Tree"

We have already created the interaction necessary for “**Fight**”. It is about time to finally complete the interaction for “**Talk**”.

5.18.2 Creating the If Else EndIF Statement

After all the preparation it is now time to create the “Statements” for the Talk button. Go to Maronion's Character Event Tab and select the TalkTo Event. The first thing you have to do is to select the green plus sign …

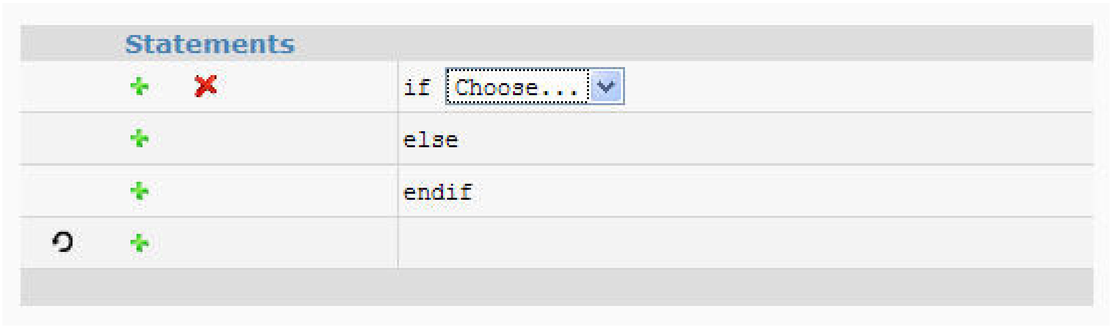


Character: Maronion

Name	Description
Click (0)	When the player clicks on the character
Fight (2)	When event occurs: "Fight"
Talk to (0)	When event occurs: "Talk to"

Statements

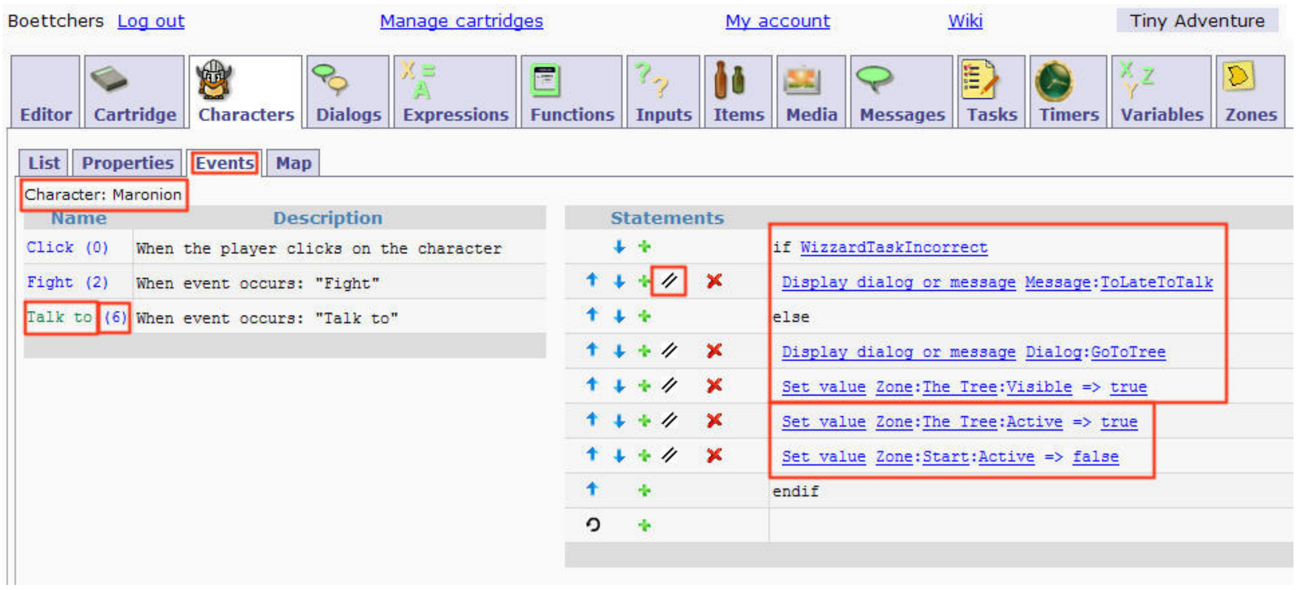
…and to choose “If-Else”. This will automatically set up the commands for the “If-Else-EndIf”



```
if Choose...
else
endif
```

If this is your first IF ELSE EndIF programming it will help to know that:

- all statements **between** the **If** condition and the **ELSE** statement will only be executed if the IF statement is **True**.
- all statements **between** the **ELSE** statement and the **EndIF** will only be executed if the IF statement is **False**
- all statements **under** the **EndIF** statement will be executed in the **normal** way. The EndIf statement closes the IF condition.



Name	Description
Click (0)	When the player clicks on the character
Fight (2)	When event occurs: "Fight"
Talk to (6)	When event occurs: "Talk to"

```
Statements
if WizzardTaskIncorrect
 Display dialog or message Message:ToLateToTalk
else
 Display dialog or message Dialog:GoToTree
 Set value Zone:The Tree:Visible => true
 Set value Zone:The Tree:Active => true
 Set value Zone:Start:Active => false
endif
```

For the tutorial I need you to create the following statements:

1) First select the expression. **"WizzardTaskIncorrect"** for the **if statement.** I hope you remember that we programmed this expression to check the "Find the wizard" task. If the "Find the wizard" task is Incorrect, we know that the player has already incorrectly tried to fight Maronion. He will not be allowed to talk to Maronion. We will therefore

   a. **"Display the dialog or message Message:ToLateToTalk"** to inform the player that he should have made up his mind earlier and that the only hope for him is to return to the bridge (for healing).

2) **"else"** opens the alternate course of actions for us (look at the decision tree if you need a picture for this decision path). Whenever the if statement in number 1) is not true this **else** branch takes over (skipping the commands of section a above). All statements following **else** (a to d below) will only be executed when the if statement in number 1) is false. This tells us that the player has not tried to fight Maronion. We do therefore:

   a. **"Display dialog or message Dialog:GoToTree"** to tell our player that he can now go to the tree.

   b. **"Set value Zone:The Tree:Visible □ true"** switches on Visibility for the zone "The Tree".

   c. **"Set value Zone:The Tree:Active □ true"** activates the zone The Tree (see 5.4.2 for details).

   d. **"Set value Zone:Start:Active □ false"** deactivates the Zone with the name "Start". We do not need this zone from here on and the player can only handle a limited number of zones (max. 5 Zones seems to be a good rule of thumb). It is good practice to deactivate unused, old Zones whenever you activate a new Zone.

3) **"endif"** closes our If statement. From now on statements will be executed as usual (but there are currently no statements below the **endIf** 😉).

   If there were statements below this line they would **all** be executed one after the other. Just like it was before you learned the IF THEN ELSE statement.

5.19 User Input (Asking Questions)

If our Hero manages his encounter with the wizard nicely he will be able to walk to the next zone ("The Tree"). It would be no real adventure if he would be able to find the key on the ground without any other task to accomplish. This gives me the chance to introduce you to the art of asking questions (or how to program a question into a Wherigo cartridge).

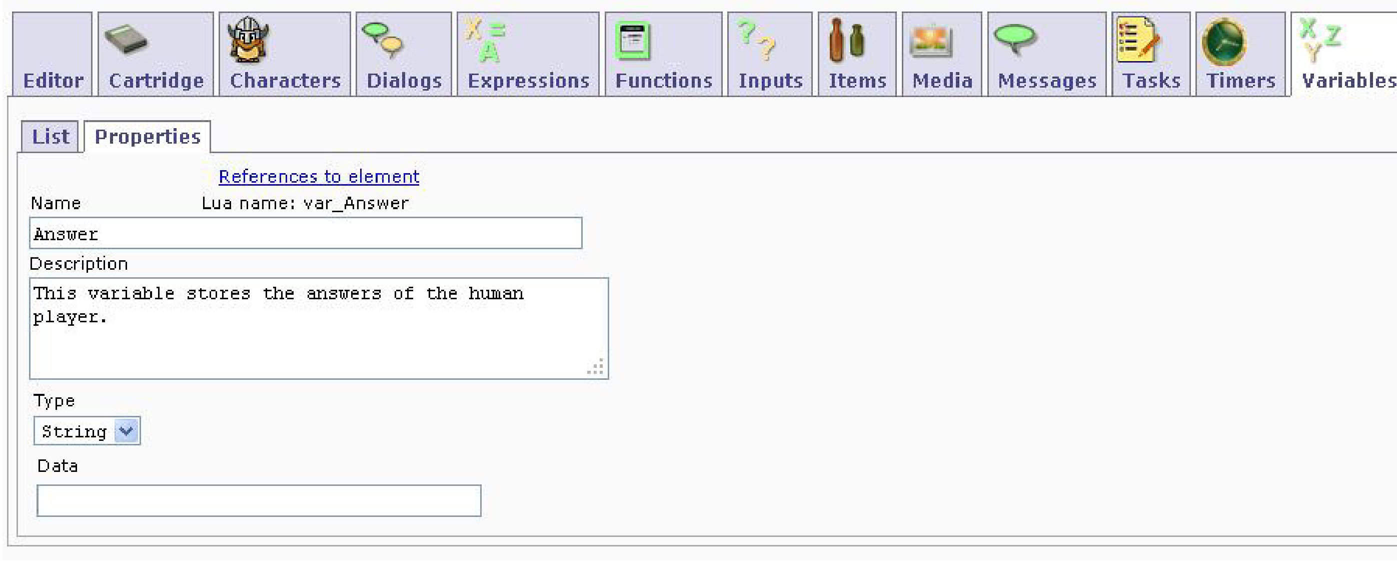
Let us assume a dwarf guards the key and asks our hero a question before he offers the key. You should already know how to create a character from this portion of the tutorial. So I will not elaborate on that topic again. Make sure to set the dwarf's container (on the Properties sub-tab) to show "The Tree" as the dwarfs location. This ensures that the human player can see a dwarf when he enters the tree zone.

Create a Character Command for the dwarf which is labeled "Talk to the dwarf". This Character Command will later be used to initiate the user input sequence that you are about to learn.

Now how do we handle questions and user input?

5.19.1 Creating a variable

The first thing that we need to do is to create a **Variable** that will store the user's input. Go to the **Variables** tab and create the variable "Answer".



Name Lua name: var_Answer

Answer

Description

This variable stores the answers of the human player.

Type

String

Data

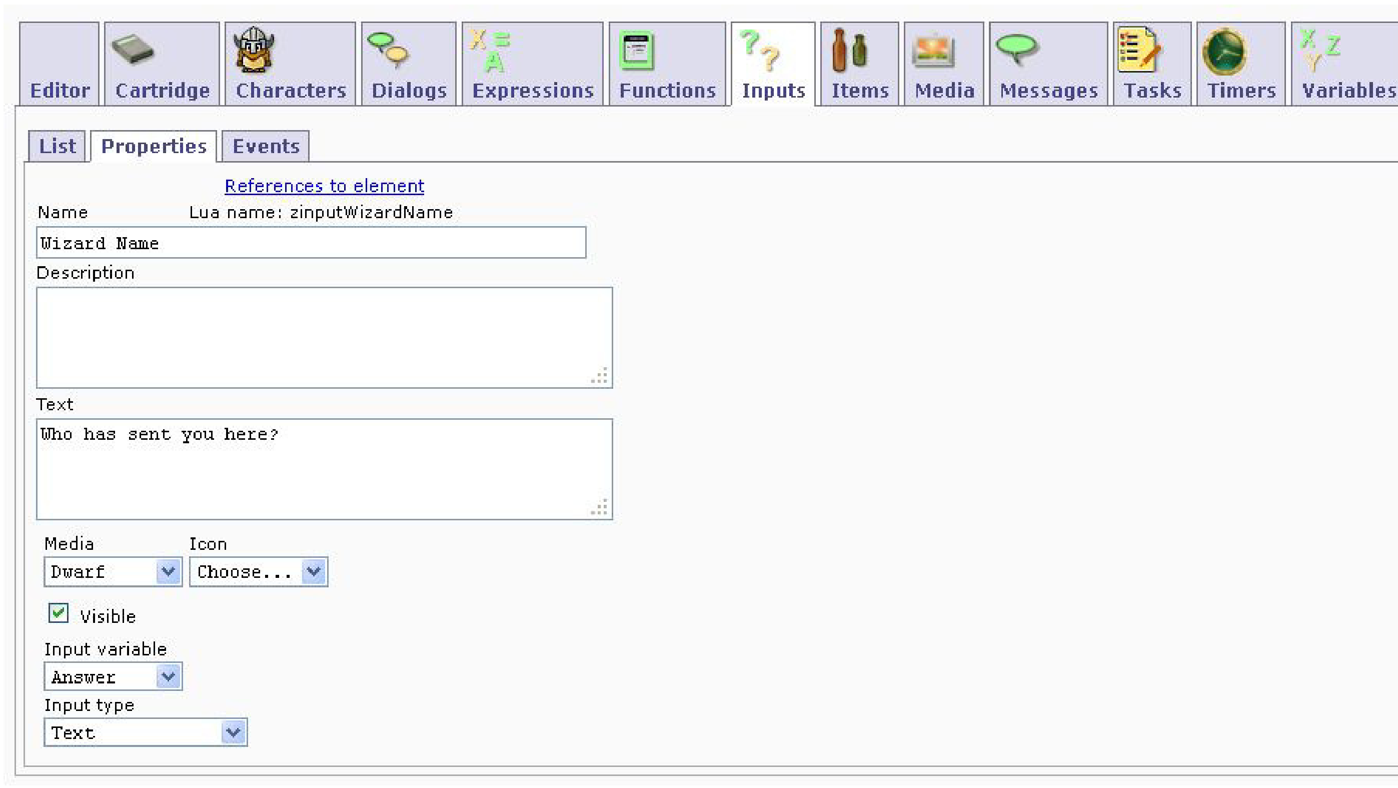
Upon creation you will have to decide which variable type you will be using. This are your options:.

- String: a text or word
- Number: a number
- Flag: a True/False type of variable. It can only store one of two possible values. It is either True or False.

Since we expect the human player to answer with a word we have to select the **Type** "String" (which is the default setting).

5.19.2 Creating an Input

The next thing that needs to be done is to create a question that somehow delivers the given answer to the variable that we have just created. In Wherigo terminology (and for computer programs) this is called an **Input**. An **Input** combines a question with a variable. The answer that the human player enters for an **Input** is automatically transfered to the storage variable that is selected for this **Input**.



Name Lua name: zinputWizardName
Wizard Name
Description
Text
Who has sent you here?
Media: Dwarf
Icon: Choose...
Visible ✓
Input variable: Answer
Input type: Text

Go to the **Input** tab and create the **Input** “Wizard Name”. Then enter the question “Who has sent you here?” in the text box. Select a media item (a dwarf picture would be nice) and select the storage variable for this **Input**. We intend to store the answer in the variable that we have previously created and named “Answer”. So this is what we select as our **Input variable** from the drop down box. The **Input type** is “Text” because we expect a word (and not a number) as the answer to the question.

5.19.3 Get Input from player

Once the **Input** is created it can be presented to the human player. As always in Wherigo we need an **Event** to do this for us. The easiest way to achieve this would be to connect the input to the **OnEnter Event** of a zone. If you do this remember that the **Input** will overwrite any **Dialog** or **Message** that you have tied to the **OnEnter Event** as well.

5.19.3.1 Text plus input

I have been asked how you can display a text for the **Zone** and ask for the **Input** thereafter. There are several possibilities.

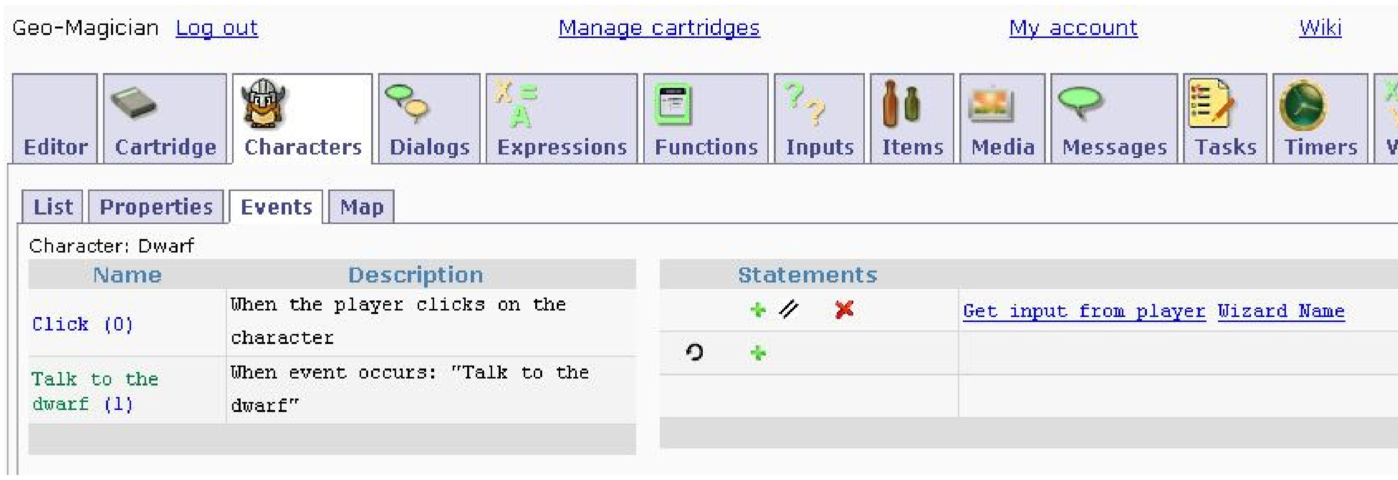
The easiest way to do this is to display a **Message** when the human player enters the **Zone**. A **Message** has a button that creates a new **Event** (the “Button” event on the **Event** sub-tab of the **Message**). Use this Button-Event to display the **Input** (Get input from player).

I have chosen a slightly more complex approach. We will be using the freshly created “Talk to the dwarf” **Event** to do this in our little adventure. The benefit of this approach is that the cartridge has more life in it and that you could also use a **Dialog** instead of a **Message** when the human player enters the **Zone** (**Dialogs** have no Events). We will give the human player a hint that someone wants to talk to him during our zone entry prelude to make sure that he does not miss the dwarf ;-).

Create the following message and display it OnEnter for the Zone “The Tree” (you can do this without any further help, if you need a reminder check this chapter for help).

You have entered the sacred grove of the magic tree. A dwarf is trying to hide from you but you have good eyes and you can see him near the roots of the tree. You approach him respectfully but he is afraid and wants to ask you a question. You should better talk to him.

If the human player has a closer look at the dwarf he will find out that he can talk to the dwarf (command button). If he decides to do so (press the button “Talk to the dwarf”) the **Input** will be displayed. In the game the dwarf will ask the human player the question “Who has sent you here?” (our **Input**).



Character: Dwarf

Name	Description
Click (0)	When the player clicks on the character
Talk to the dwarf (1)	When event occurs: "Talk to the dwarf"

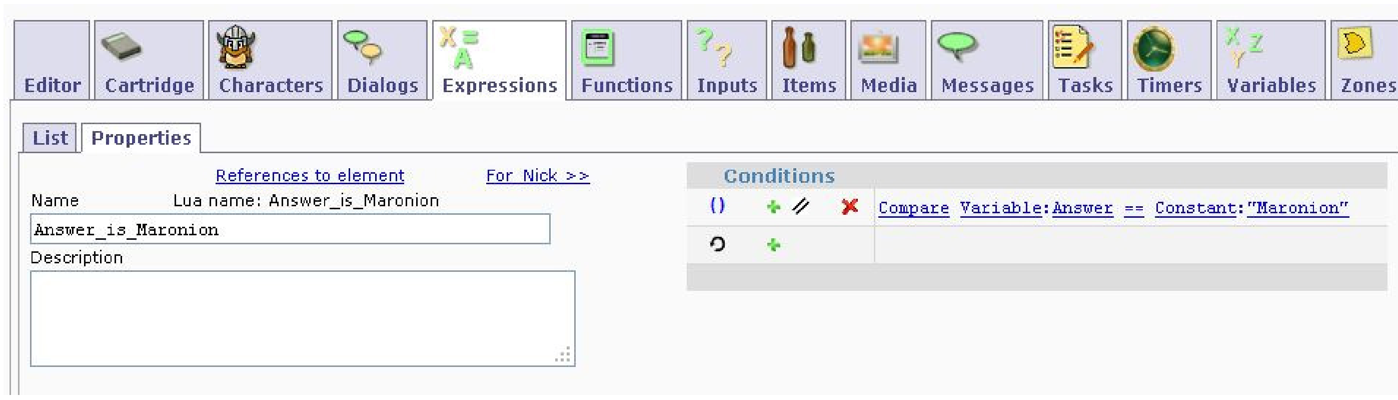
Statements
Get input from player Wizard Name

5.19.4 Evaluating Inputs

In order to evaluate the **Input** we need to remember the decision making chapter of this tutorial. If you did understand how to make a decision you might be able to see that all that is left is to decide whether the human player has entered the correct or incorrect answer (similar to deciding whether our hero has chosen to attack or to talk to Maronion). If you don't quite remember, don't panic 🙂. I will guide you through the procedure step by step.

5.19.4.1 Evaluation: Creating an Expression

The first stop for all decisions is to create the **Expression** that describes what you expect as the solution to your puzzle. In our little adventure the dwarf has asked the human player the question: “who has sent you here”. As our solution to this question we are expecting the answer “Maronion”. If you think back to the previous chapter you might remember that we have used an **Input** to transfer the human's answer to the **Variable** “Answer”. So at this stage we can expect that the Variable “Answer” contains the value “Maronion”. This is exactly what needs to go in our **Expression**.



References to element For Nick >>

Name Lua name: Answer_is_Maronion

Answer_is_Maronion

Description

Conditions

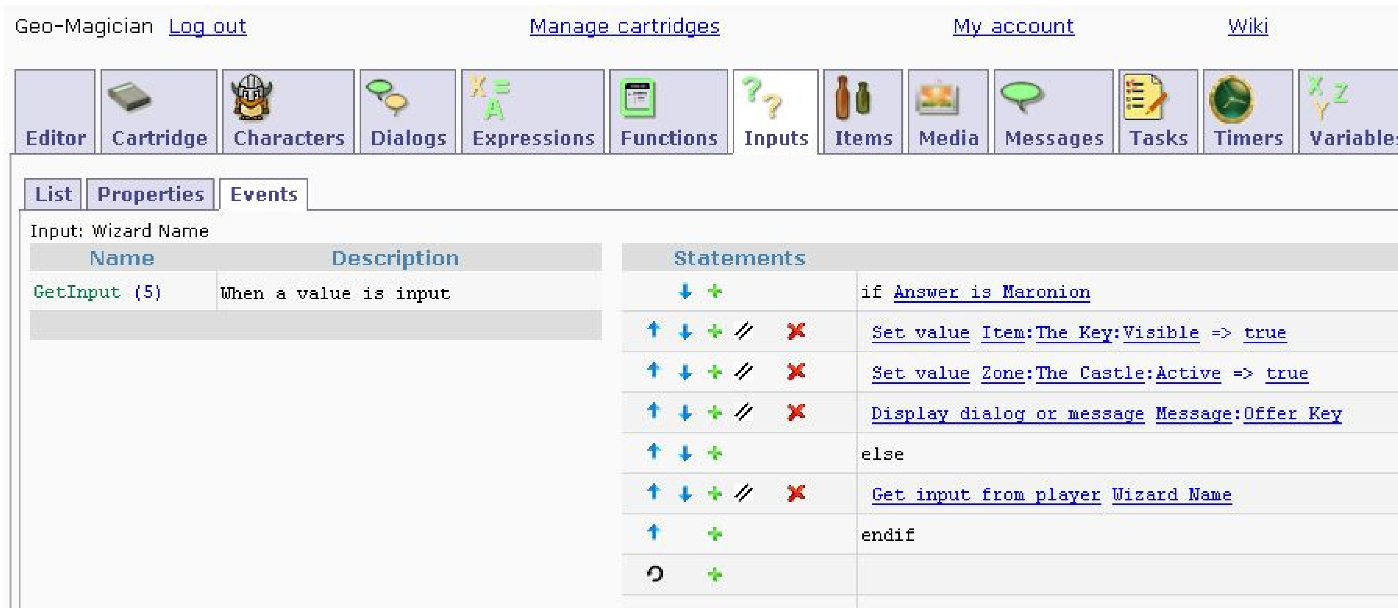
Compare Variable:Answer == Constant:"Maronion"

This expression gives us a chance to evaluate the user's input against the constant value "Ma-ronion".

5.19.4.2 Evaluation: Testing against the Expression

Once you have detailed your solution in the Expression you can test the Expression and find out whether it is true or false. If the Expression (Compare Variable:Answer == Constant:"Maronion") is true we know that the human player has entered the the correct value. We can continue with the cartridge (make the key visible, show another zone, disable the current zone, …). If the Ex-pression (Compare Variable:Answer == Constant:"Maronion") is untrue we know that the human player has entered the wrong answer. We need to give him another chance.

The easiest way to test this is to use the event that is created by the Input itself. Every time that the human player enters a value he triggers the GetInput event. This event tells us that he hu-man has tried to answer our question. Time to test it with an **IF Then Else** structure (sounds familiar 😉?).



Input: Wizard Name

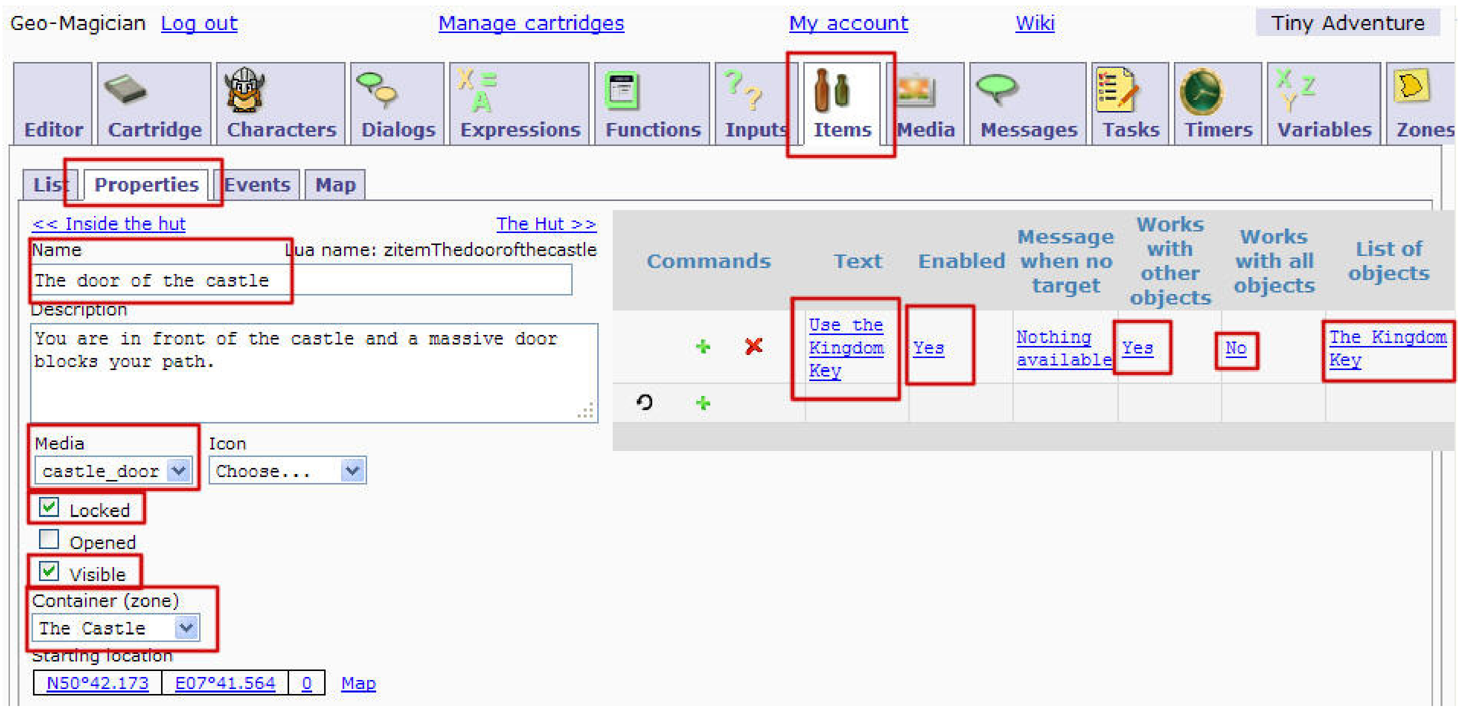
Name	Description
GetInput (5)	When a value is input

Statements

```
if Answer is Maronion
 Set value Item:The Key:Visible => true
 Set value Zone:The Castle:Active => true
 Display dialog or message Message:Offer Key
else
 Get input from player Wizard Name
endif
```

If the user has answered correctly three things will happen

- The key will become visible (your human player should be able to pick up the key and put it into his inventory without any further programming from your side, we have done this already when we created the item "The kingdom key")
- The castle zone will become active (and visible)
- The dwarf will tell our hero to pick up the key. This Message Task



Name: The door of the castle
Lua name: zitemThedoorofthecastle
Description: You are in front of the castle and a massive door blocks your path.
Media: castle_door
Icon: Choose...
[✓] Locked
[ ] Opened
[✓] Visible
Container (zone): The Castle
Starting location: N50°42.173 E07°41.564 0 Map

Commands	Text	Enabled	Message when no target	Works with other objects	Works with all objects	List of objects
+ ✗	Use the Kingdom Key	Yes	Nothing available	Yes	No	The Kingdom Key
↺ +						

If the user has answered incorrectly we will offer the Input again.

- You can program your cartridge this way and it will work. I do however suggest to present a message to let the human player know that he has given the wrong answer before you repeat the input procedure. Do this as a task without any help from the tutorial. All you need to do is to:
  - Create a message that tells the player that he was wrong.
  - Display this message in the Else branch of our If-Else-EndIf decision.
  - Now use the Message event to display the input again.

5.20 Final Exam (You need to learn how to fly all by yourself ;-)

You have almost made it. And now it is time for you to go out there and fly yourself. I know it is mean to leave you and Maronion in this unsolved situation but You are now more than capable to bring this story to a successful end. But don't despair; I will give you a few hints on how to continue:

5.20.1 Zone Command Workaround

The next step is a workaround to overcome the Garmin restriction for ZoneCommands. Just create an item and place it in the zone. See below how it works.

- Create an item named "The door of the castle"
- Select the Zone "The Castle" as the Starting location for "The door of the castle". Make sure the door is visible
- For this door: create the command "Use the Kingdom Key" and select the option "Works with other objects" YES
- Make sure this command does only work with the kingdom key (see below)

5.20.1.1 Object Command "works with other objects"

If you plan to use the “Works with other objects” feature in your own cartridge you need to be aware of two possible “problems”

The “other objects command” **will be available on all selected objects**. For our example this means that the key and the door will have a command labeled “Use the Kingdom Key”. This will work for our example because “Use the Kingdom Key” will make sense when offered together with the door or with the key. Imagine you would have a command like “Put in keyhole”. This will work nicely on the side of the key (put the key in a keyhole). However on the door side things will look bad (put the door in a keyhole :?). You have to remember this every time you use this type of command.

The “other objects command” **will be visible on all selected objects**. This statement does admittedly sound like the previous case but what I am trying to point out is that sometimes you do not want the human player to know in advance that he can use an item in a certain way. An example from one of my own cartridges might make this clearer. In my case the human player can find a photo of a person. He needs the photo to continue his quest because the background tells him where to go next. In a later stage of the game the same photo is used to identify a suspect. It would be bad if I hand the photo to the player and let him know right away that it has an option “Compare photo with suspect” before he has even met the suspect. We can however avoid this if we disable the critical command until it is needed.

One visibility problem does however remain. In my case I have combined two objects that have different visible histories. The photo will be hidden in the players inventory and he might have forgotten about it by the time that he meets the suspect. The suspect itself is a completely different story. He will be in plain sight (aka visible in the same zone as the player). When the human player looks at the suspect he will receive the information that he can “Compare photo with suspect”. This makes it all to easy to figure out what to do (at least if you are as mean as me 😉).

5.20.2 The completion code

5.20.2.1 Displaying the completion code

If the human player wants to log his completed cartridge at www.wherigo.com he needs to enter a “completion code”. This code will not be displayed automatically, you need to do it.

- Select the last event of the cartridge (you can use other events if you know your way around)
- Use this event to mark the cartridge as complete (SetValue/Cartridge/Completion/True)
- Display the following message:

Congratulation, you have completed this cartridge. Your completion code
is: ~~Player.CompletionCode~~ . Write this code down, you need it to log
the cartridge at www.wherigo.com.

Make sure you copy the ~ symbols when you write your Message. They fetch the enclosed variable from the system and include it in your message (I hope you are smart enough now to know that you have to create the Message before you can display it in the event 😉).

5.20.2.2 Creating a completion code item

It is a good idea to prepare a “Completion code” item in addition to the above message. This will give the human player a chance to look at the completion code again when he is at home. The message that we have prepared in the previous section actually requires the player to copy the information in the field. If the human player fails to do so the information will be lost.

The cartridge has to be saved before the human player closes the cartridge or all information will be lost regardless of all your efforts;-).

5.20.2.2.1 Preparation of the "Completion code" item

- Create an item “Completion code”
- Make sure the item is invisible
- The item has no starting location
- Create a Command “Show completion code”
- Use this command to display the previously created completion code message

You might be curious why I am using a command and a message to display this information. The item description looks like a much easier way to do this without all the trouble of creating commands and messages. This is actually a good question and if you have really asked yourself something along this line I think you are ready to put your own cartridges out there 😉. Coming back to the question: the item description is created together with the item itself when the cartridge is started for the first time. At this time the cartridge is still incomplete and there is no completion code available. If you display a message however this will look up the current value of any variable (for example Player.CompletionCode) and display it to the human player.

5.20.2.2.2 Presenting the "Completion code" item at the end of the game

- Move the “Completion code” item to the player (inventory)
- Make the item visible
- Save the cartridge
- Inform the player about his new item

5.20.3 Uploading your cartridge to wherigo.com

I have provided some tips how to upload your cartride in the earwigo FAQ. Check the FAQ for the topic Uploading your Cartridge to Wherigo.com

6 Further information

I guess you have already found the WWB Reference (work in progress) which lists information about the individual tabs of the WWB. The information might be identical to this tutorial (several pages have been copied) or it might contain new information from other authors. The main advantage is that information is listed by WWB tabs which makes it easier to find advice for a specific WWB problem.

The following links from the Groundspeak Wherigo Wiki can be helpful if you can not find the information in our Wiki:

- The Builder section can be used to explain the Earwigo tabs

- The Object classesare more for the programmers that use Earwigo

Have fun creating Wherigo cartridges with Earwigo!

Geo Magician

Geo Magician