



Wherigo for dummies

Índice



Objects

- [Cartridge](#)
- [Player](#)
- [environment](#)
- [Current object](#)
- [Zones](#)
- [Characters](#)
- [Items](#)
- [Task](#)
- [Variable](#)
- [Input](#)
- [Timers](#)
- [Functions](#)
- [Media](#)
- [Current objects](#)

Actions

- [If/then](#)
- [Random](#)
- [Increment/Decrement](#)
- [Set](#)
- [Move](#)
- [Dialog](#)
- [Message](#)
- [Show Screen](#)
- [Show object detail](#)
- [Input](#)
- [Start timer](#)
- [Stop timer](#)
- [Function Call](#)
- [Play sound](#)
- [Stop sound](#)
- [Save](#)
- [Save and exit](#)
- [Lua user code](#)
- [Coment](#)

Expressions

- [Compare](#)
- [and](#)
- [or](#)
- [numeric operation](#)
- [Concatenate](#)
- [Contain](#)
- [Value](#)
- [Answer](#)
- [Command target](#)
- [Message button](#)
- [Distance to zone](#)
- [Random number](#)
- [In emulator](#)
- [Date/time](#)
- [Lua user expresion](#)

- [Ficheros del cartucho](#)
- [Propiedades comunes](#)
- [Propiedades de Items y characters](#)
- [Propiedades en tiempo de ejecución](#)
- [Notas y trucos](#)
- [Glosario](#)
- [Acentos](#)



Wherigo for dummies

Objetos



Objetos son todas las cosas que pueden aparecer en nuestro cartucho/programa: zonas, personajes, cosas, el jugador, media, tareas etc:

- Los objetos tienen propiedades. Las propiedades se pueden definir cuando creamos el cartucho o podemos cambiarlas durante el juego(tiempo de ejecución) utilizando la acción set.
- Cuando un objeto interactúa con otro o cambia su estado, genera un evento a ese evento la asociamos unas acciones (handler) que es lo que va a suceder en respuesta a ese evento.
- Podemos crear todos los objetos que necesitemos
- Tipos:
 - Zonas:son espacios físicos que podemos definir la posición del jugador determinada por el GPS define dónde está, generando lo eventos necesarios
 - Personajes (characters):Interactúan con el jugador, portan objetos, se mueven etc
 - Cosas(Items): pueden transportarse, usarse, estar en una zona...
 - Tareas (task): cosas que hay que realizar para completar el cartucho
 - El jugador
 - El cartucho, timers.....



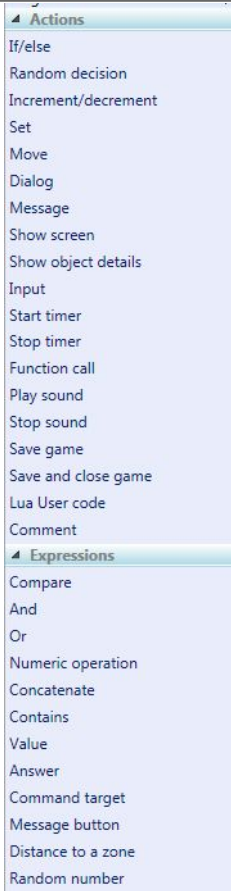
Wherigo for dummies

Actions Expressions



Actions

- Se pueden utilizar por sí mismas.
- Ejecutan acciones en el flujo del programa o sobre los objetos
- En algunas necesitamos indicarle sobre qué objeto tienen que actuar, en otras hay que añadir expresiones para que funcionen.



Expressions

- No funcionan por sí solas, siempre necesitan ir acompañando a una acción.
- Siempre devuelven un valor el cual es procesado por una acción
- El valor devuelto puede ser de cualquier tipo, (verdadero/falso, un número o una cadena)



Wherigo for dummies

Propiedades comunes de los objetos



General	
Name	Cartridge
Description	
Identifier	Automatic

Display	
Display	<input checked="" type="checkbox"/>
Image	None
Icon	None

Excepto Identifier, todas las podemos consultar y cambiar durante la ejecución del programa.

Existen unas propiedades comunes a todos los objetos:

- Name: es el nombre del objeto, podemos ponerlo cuando lo creamos o lo podemos modificar durante el programa
- Descripción es el texto que aparece cuando lo mostramos con el comando [Show object detail](#)
- Identifier: es una propiedad que usa internamente para llamar a los objetos. Es mejor dejarla en automático.
- Display, si esta marcada el objeto será visible por el jugador, en caso contrario no. En el objeto cartridge esta propiedad tiene que ser visible
- Image, es la imagen que se mostrará a pantalla completa. Tenemos que crearla previamente como un [Media](#)
- Es la imagen que se mostrará en las pantallas principales. Tiene que tener un tamaño de 32x32



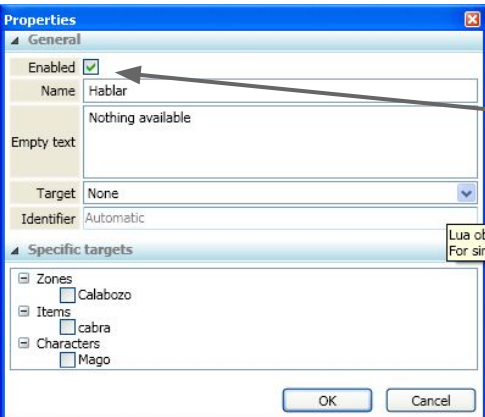
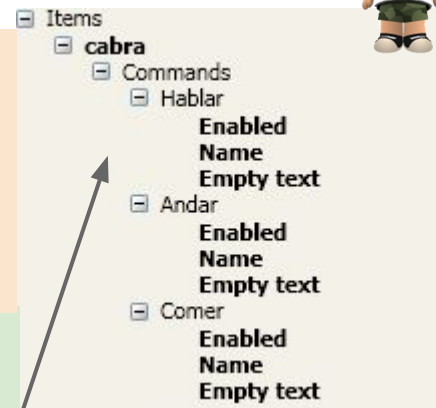
Wherigo for dummies

Propiedades comunes de items y characters I



- Location es donde va a aparecer el objeto inicialmente. El lugar inicial puede ser cualquier cosa que tenga inventory count (un items, una zone, un characters o el jugador (player)).
- Podemos fijar una posición, **no se para qué sirve**

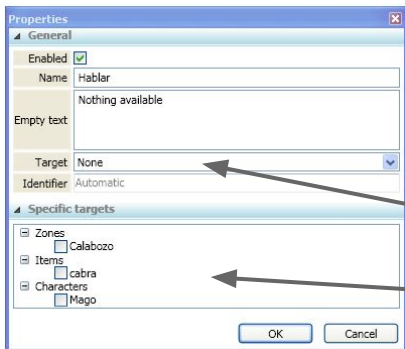
- Commands (comandos) es una de las formas más potentes de interactuar con el objeto, son comunes también con las zonas.
- Por cada comando que añadamos aparece un botón en ese objeto con el cual podemos ejecutar acciones.
- Los comandos pueden estar habilitados (Enabled) o permanecer invisibles.
- Empty text es el texto que aparece cuando ninguno de los objetivos del comando está accesible.
- Las propiedades Enabled, Name, y Empty text pueden ser modificadas durante el diseño desde el programa.
- En función del objetivo (target) Los comandos pueden ser: standalone, para múltiples objetos y script multi objetos





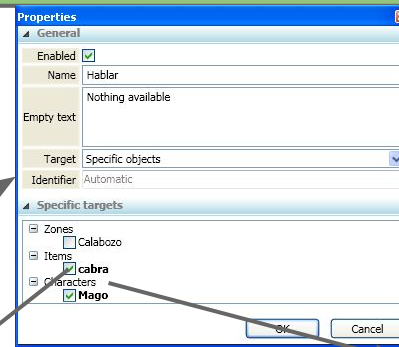
Wherigo for dummies

Propiedades comunes de items y characters II



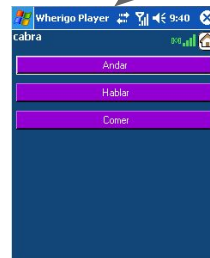
Standalone

- Es el más simple, actúa únicamente sobre el objeto en el cual lo hemos creado.
- Al crearlo seleccionamos Target a None.
- No tenemos que seleccionar ningún objetivo

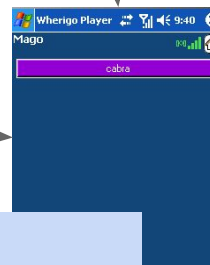


Multi objeto

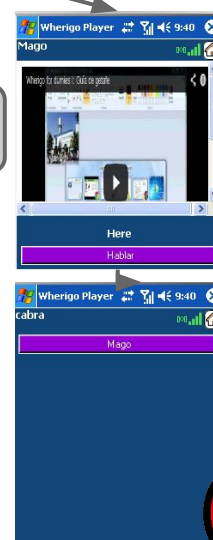
- Seleccionamos los objetos objetivo.
- Target se pone a Specific object.
- El nuevo botón se crea en todos los objetos que sean objetivos(target) del comando. Hay un único evento en el objeto en el cual se creó.
- Cuando seleccionamos el comando, se nos abre una ventana para seleccionar el objetivo.
- En la acciones asociadas al comando podemos discriminar el objeto objetivo y hacer acciones distintas dependiendo del objetivo



Aparece un botón por cada comando.



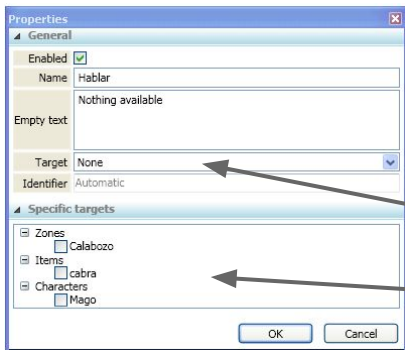
Aquí seleccionamos el objetivo





Wherigo for dummies

Propiedades comunes de items y characters III



Script multiobjetos

- Actúa sobre todos los objetos. Podría ser por ejemplo inspeccionar.
- Al crearlo seleccionamos Target a All.
- Este comando aparecerá en todos los Items, Characters.
- Para saber exactamente cual ha sido seleccionado deberemos utilizar [Command target](#)

Existe una propiedad en los comandos que no esta abierta en el Urwigo pero que podemos programar dentro de Lua User Funtions esMakeReciprocal y hace que un comando sólo esté eN el objeto origen y no en el objeto destino. Su uso es:
`objPatada.Commands.cmdAtacar.MakeReciprocal =false`



Wherigo for dummies

Propiedades en tiempo de ejecución



Aquí aparecen las propiedades en tiempo de ejecución

- Son las propiedades de los objetos que podemos utilizar durante la ejecución del programa para comprobar o cambiar su estado. Por ejemplo la posición de una zona la fijamos únicamente desde el compilador, es en tiempo de compilación. La propiedad name la podemos consultar cuando se ejecuta el programa, es en tiempo de ejecución.
- Aquellas a la que se puede cambiar su valor, diremos que son escribibles p.e. Image o Name
- Las que no podemos cambiar su valor diremos que son de solo lectura p.e. Inventory count o elapsed



Wherigo for dummies Cartridge



Objects and properties

- Cartridge**
- Complete
- Activity type
- Display
- Image
- Icon
- Name
- Description
- Start description
- Version
- Company
- Author

Cartridge

Activity type: Guide

Start: Play anywhere

Start description:

Version:

Company:

Author:

Logging: ☐

Other

Obfuscate strings: ☒

Encrypt answers: ☒

Obfuscate identifiers: ☒

Inline Lua 'require': ☐

Lua 'require' LUA_PATH:

Events

On start: [handled](#)

On end: [handled](#)

On resume: [handled](#)

On save: [unhandled](#)

Global zone events

On zone enter: [unhandled](#)

On zone exit: [unhandled](#)

On zone distant: [unhandled](#)

On zone proximity: [unhandled](#)

On zone activity change: [unhandled](#)

Emulator protection

Enabled: ☐

Displayed text:

- Activity type: sirve únicamente para clasificar el cartucho en la página de Wherigo.
- Start: punto de inicio del cartucho. Si no ponemos nada se puede jugar en cualquier sitio.
- Star description: descripción del punto de inicio
- Versión: sirva para controlar las versiones del cartucho
- Logging: graba un fichero de los pasos seguidos para completar el cartucho. Se usa con propósitos de depuración. El fichero generado se puede subir luego a la página de Wherigo
- Others: las funciones en esta sección , encripta las cadenas de texto para que si abrimos el cartucho con un editor de texto no puedan verse.
- Events. On start se ejecuta al iniciar el cartucho, On end se ejecuta cuando se termina el cartucho, On save cuando se guarda , On restore cuando reanudamos el juego .
- Global zone events:hacen lo mismo que sus equivalentes de cada zona particular lo único que esto saltan cuando ocurre el evento en cualquiera de las zonas. Estos evento se ejecutan antes que los definidos en cada zona. Pueden servir por ejemplo para ejecutar un sonido cuando entramos en una zona.
- Emulator protection sirve para que el cartucho no se pueda correr en un emulador. El texto de debajo es el que aparecerá cuando se ejecute en un emulador indicando que no se puede.

Ejemplos

Comandos relacionados: [Propiedades comunes](#), [Compare](#), [and](#), [or](#)



Wherigo for dummies

Cartridge II



Objects and properties

- [-] Cartridge
 - Complete
 - Activity type
 - Display
 - Image
 - Icon
 - Name
 - Description
 - Start description
 - Version
 - Company
 - Author

Los cartuchos son únicos para cada jugador y se generan en el momento que el jugador se los bajan. En el cartucho va el nombre del jugador que se lo ha descargado (tenemos que estar registrados para poderlo descargar) y cuando se instala en nuestro dispositivo se registran los datos de la máquina. Esto implica que:

- No nos pueden pasar el fichero de otro jugador porque cuando lo acabemos solo lo podremos registrar como completado a nombre de quien se lo descargó. Por tanto, siempre debemos descargar nuestro propio fichero desde la nuestra cuenta en Wherigo.com.
- Los ficheros GWS en los cuales se salva la ejecución del juego, no son intercambiables y no funcionarán en nuestro dispositivo.
- El número que se genera cuando acabamos el cartucho y que no permite registrarlo como acabado, también es único. No podemos registrar como acabado un cartucho que un número que nos den.
- Para indicar que el cartucho se ha completado, pondremos Compete a True

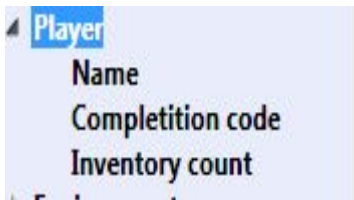
[Ejemplos](#)

Comandos relacionados: [Propiedades comunes](#), [Compare](#), [and](#), [or](#)



Wherigo for dummies

Player/Environment



- Name y Completion code se rellenan en la página de Wherigo cuando nos bajamos el cartucho y no se pueden cambiar. Son el nombre del jugador y el código necesario para desbloquear el juego. name se puede utilizar para llamar al jugador por su nombre durante el juego.
- Inventory count es el número de objetos en el inventario del jugador.
- Cuando queremos que algo pase al inventario del jugador haremos move to Player
- El jugador se usa como el resto de los objetos.

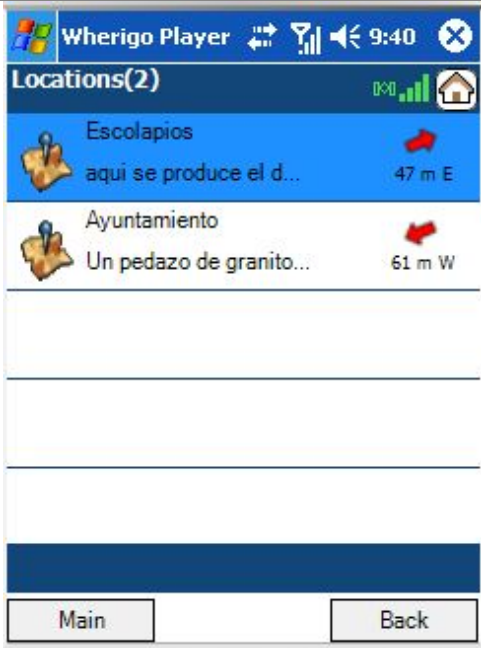
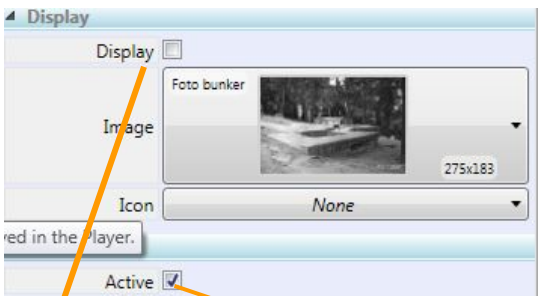


- Estas propiedades se usan internamente para saber sobre qué máquina está corriendo el cartucho y así poder evitar que el cartucho guardado en un equipo se pase a otro y funcione.
- Device ID y device identifican el dispositivo en el cual se está ejecutando el cartucho.

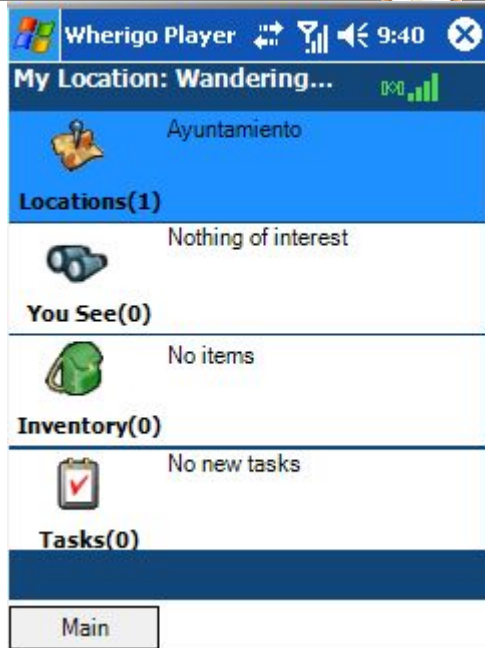


Wherigo for dummies

Zones I



Dos zonas con Display y
Active activados



Solo una zona con Display y
Active activados

Estados	Activo	Inactivo
Visible	Se ve Hay eventos	No se ve No hay eventos
Invisible	No se ve Hay eventos	No se ve No hay eventos

Cuando cambiamos una zona de activa a inactiva o viceversa se activa el evento **On activity change**

[Ejemplos](#) Comandos relacionados: [Propiedades comunes](#), [Propiedades en tiempo de ejecución](#)



Wherigo for dummies

Zones II

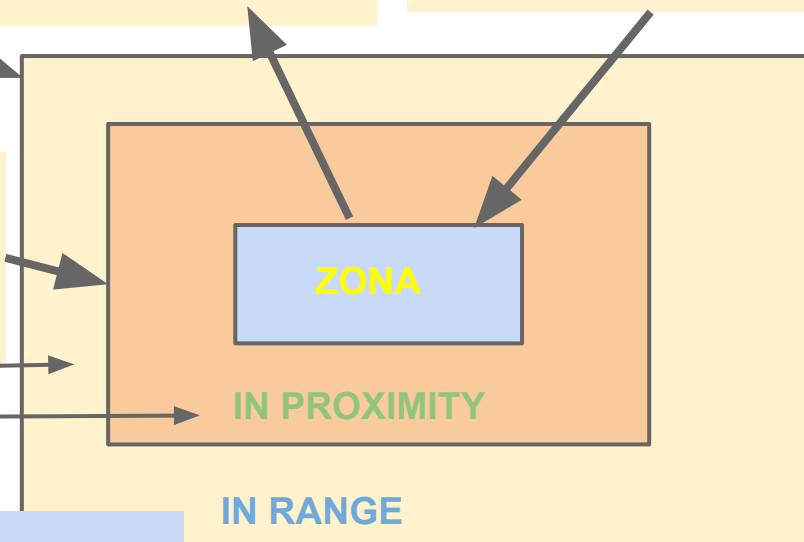
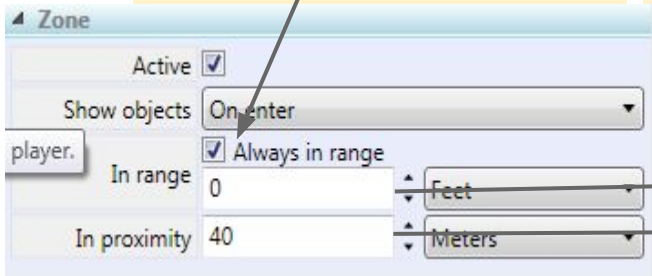


Al entrar en la zona in range, evento **On distant**.
Si ponemos valor en el campo In range, las zonas sólo serán visibles cuando el jugador está dentro de la zona in range.
Si queremos poderlas hacer visibles en cualquier momento independientemente de la distancia, debemos activar **Always in range**

Al salir de la zona, evento **On exit**. Se genera cada vez que salgamos y la zona esté activa

Al entrar en la zona, evento **On enter**: Se genera este evento cada vez que entremos en la zona y esté activa

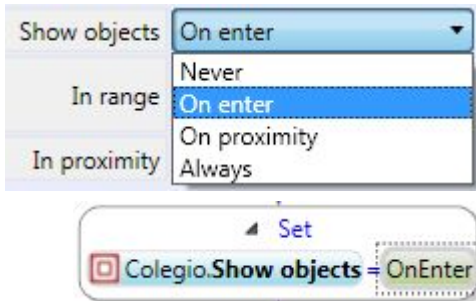
Al entrar en la zona in proximity, evento **On proximity**





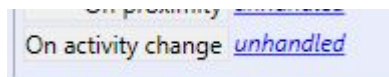
Wherigo for dummies

Zones III

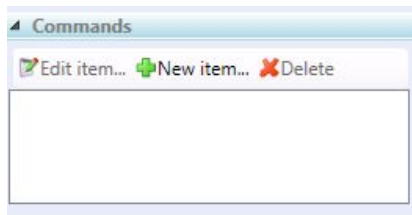


Show object: Cuando se muestran los objetos contenidos en una zona. Si la propiedad display del objeto está desactivada el objeto nunca es visible.

- Always se muestran siempre que la zona esté activa y el jugador esté en rango. Aunque la zona no esté visible los objetos contenidos en ella sí lo están.
- On proximity, los objetos aparecen mientras el jugador esté dentro de la zona On proximity y se ven aunque la zona no sea visible.
- On enter: se muestran cuando el jugador entra en la zona.
- Never: nunca se ven. Esta opción tiene sentido porque Show object la podemos cambiar desde el programa de manera que en unos momentos puede ser visible y otros no.



Las zonas tienen un evento On activity change el cual se activa cuando pasamos la zona de activa a inactiva o viceversa. Recordar que también existe este evento global para todas las zonas. Los para la misma hecho eventos globales se ejecutan antes que los locales

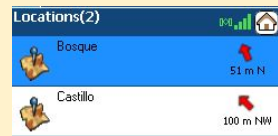
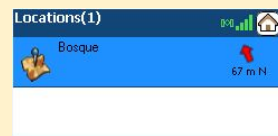
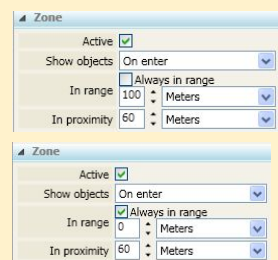


A las zonas les podemos poner comandos de la misma manera que a los Items y a los Character

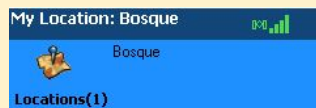
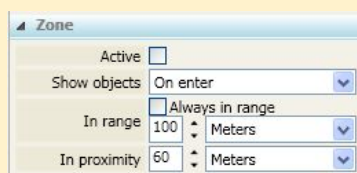


Wherigo for dummies

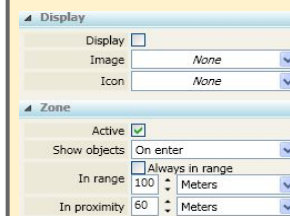
Ejemplo de zonas



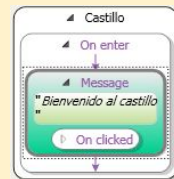
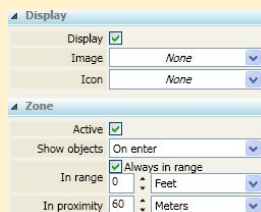
Tenemos dos zonas, bosque (la grande) y castillo (pequeña). Ambas están activas y visible. Bosque tiene in range fijado a 100 mm y castillo siempre in range. El bosque es siempre visible desde cualquier parte, pero el castillo hasta que no estemos a 100 m In range, no nos aparecerá en pantalla. Para que aparezca no hay que hacer nada, simplemente aparece cuando el jugador se acerca



Ahora castillo no está activo, por lo que por muy cerca que estemos no aparece y además no genera eventos. Para hacerlo visible tenemos que añadir código que lo haga activo



Aquí el castillo está activo pero no visible. Por tanto no podemos verlo pero al entrar en él se genera el evento On enter que nos muestra el mensaje



Esta es la situación más normal ambas zonas activa y visibles, con siempre en rango activado. Las zonas son visible para el jugador y se generan todos los eventos asociados. Como tenemos proximity puesto a 60m ,cuando estemos a esa distancia se generará el evento On proximity



Wherigo for dummies

Ejemplo de zonas II



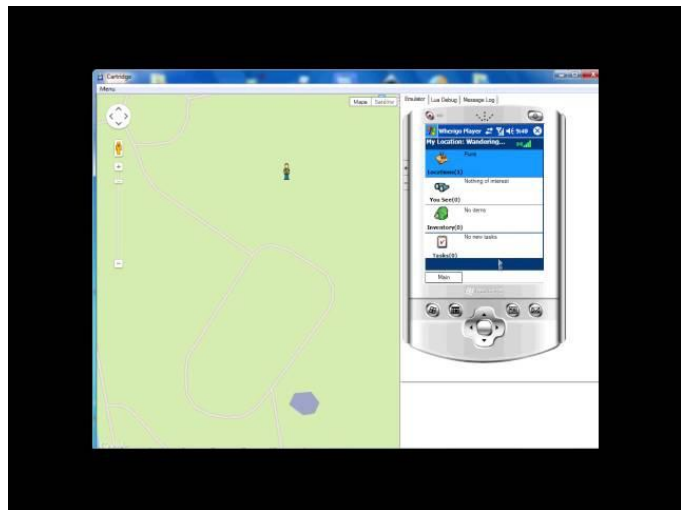
Zone

Active ☒

Show objects On enter

In range ☐ Always in range
100 Meters

In proximity 60 Meters



Cuando nos acercamos a una zona los eventos saltan en el siguiente orden:

- On distant
- On proximity
- On Enter

Al alejarnos:

- On exit
- On proximity
- On distant







Al salir de una zona se genera primero On exit y luego On proximity pero son prácticamente simultáneos por lo que si utilizamos la pantalla en los dos On prox sobrescribe a On Exit.



Wherigo for dummies

Characters



General	
Name	Mamá
Description	Mi mamá me mimó mucho
Identifier	Automatic
Display	
Display	<input checked="" type="checkbox"/>
Image	<div>Mama  140x160</div>
Icon	<div>Icomama  28x32</div>
Character	
Gender	Female
Location	
 Mi casa	
Anywhere	<input type="button" value="Clear"/> <input type="button" value="Edit in map..."/>
Commands	
 Edit item...  New item...  Delete	
Hablar	
Enabled	handled
Targets:	None
Events	
On click	unhandled

Son junto con los items lo objetos fundamentales para elaborar la historia de un cartucho. Tienen las propiedades comunes de todos los objetos, comparten los comandos con los ítems y las zonas.

- Tienen la propiedad gender(género) que puede ser masculino, femenino o neutro. Podemos usarlo para señalar el género del objeto o como bit que señale alguna situación especial.
- Puede llevar objetos y pueden ser movidos de una zona a otra
- Pueden ejecutar y recibir comandos
- Tienen la propiedad Inventory count la cual nos indica cuántos objetos lleva el personaje. Esta propiedad es de sólo lectura . Para mover un objeto a un personaje usamos [Move](#)



Wherigo for dummies

Items



Properties

General


Name: Lobo

Description:

Identifier: Automatic

Display

Display: ☒

Image: Lobo  230x180

Icon: None

Item

Locked: ☐

Opened: ☐

Location

Orilla derecha

Anywhere Clear Edit in map...

Commands

Edit item... New item... Delete

Subir a la barca

Enabled: handled

Targets: None

Events

On click: unhandled

- Nombre, descripción, identificación y display igual que en las zonas y personajes.
- Locked y Opened son bits a disposición del programador, aunque tienen un nombre asignado les podemos dar la utilidad que necesitamos. Podemos utilizarlos para guardar propiedades o situaciones del Item
- Location lugar donde aparecerá el ítem la primera vez. También podemos poner una posición exacta en la propiedad de debajo. Normalmente es Anywhere pero lo podemos asignar una posición.
- La visibilidad de un objeto está definida por su propiedad display y por la propiedad [Show object](#) de la zona que lo contiene.
- Commands, son los comandos que le vamos a asignar al objeto, generan botones en la pantalla del objetos. Estos botones tienen sus propios eventos.
- Event on click: evento que se produce al seleccionar el objeto, no se recomienda usarlo porque en los Garmin no funciona y cuelga el programa.
- Cuando una cosa es contenedor de otra ni aparece en su inventario (parece un error del programa), en contain si lo detecta.

Ver también: [Propiedades comunes](#), [Propiedades de Items y characters](#), [Propiedades en tiempo de ejecución](#)



Wherigo for dummies task



General

Name: Hacer la compra

Description:

Identifier: Automatic

Display

Display: ☒

Image: None

Icon: None

Task

Active: ☐

Complete: ☐

Correctness: None

Events

On active changed: [unhandled](#)

On complete changed: [unhandled](#)

On correct chng.: [unhandled](#)

On click: [unhandled](#)

Task

Active: ☐

Complete: ☐

Correctness: None

Events

On active changed: None

Un cartucho podría funcionar sin ninguna tarea, pero el hecho de que haya tareas simplifica la vida del jugador puesto que le recuerda que cosas tiene pendientes o que cosas le faltan por hacer. Sirven para guiar y recordar al jugador las cosas que tiene que hacer. Por eso es mejor crear tareas, mantienen al jugador en tensión y con una idea clara de lo que tiene que hacer.

- Para que la tarea aparezca en el juego debemos poner Display y Active a True.
- Cuando la tarea está completa ponemos Complete a true. Aparecerán en la lista de tareas como completada.
- El campo Correctness tiene tres opciones posibles none, Correct, Not correct. Sirve par indicar si la tarea se ha completado correctamente o no. Por ejemplo si tiene que hacer una carrera en un tiempo determinado, puede haberla completado pero no en el tiempo correcto
- Tiene asignados unos eventos lo cuales saltan cuando cambia el estado de la tarea. Los eventos están ligados a las propiedades Active, Complete y Correctness y se activan cuando cambian estas propiedades. Los eventos sólo se generan cuando la tarea está activa.
- Podemos crear tareas internas (que el jugador no ve) y que nos pueden servir para controlar el desarrollo de la acción. Para hacer esto, la tarea debe de estar con Display a false y Active a tue

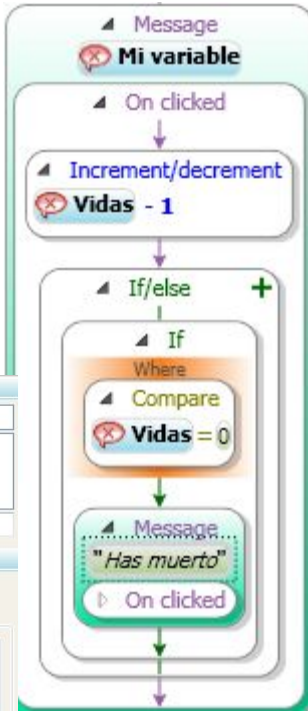
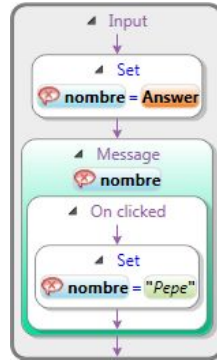


Wherigo for dummies variables



Sirven para almacenar valores numéricos, palabras, frases o estados que se generan en un momento del juego y que queremos usar posteriormente. A las variables debemos asignarles un nombre que la identificará.

- Hay distintos tipos. Pueden ser números, picando number. Cadenas de texto, picando string o un estado verdadero falso, seleccionando la casilla correspondiente.
- Para asignarles un valor debemos utilizar el comando [Set](#) el valor que se ponga al otro lado debe ser del mismo tipo que la variable.
- Con las variables numéricas podemos usar [Increment/Decrement](#) para cambiar su valor.
- Si queremos comprobar si una variable tiene un determinado valor deberemos utilizar [Compare](#) el cual nos devolverá verdadero si se cumple la condición
- Podemos utilizarla en un mensaje para que el jugador conozca su valor.



Variables

nombre

Name	Description
nombre	

Value

☐ True (True)

☐ False (False)

☒ String (text)

0

General

Name: Vidas

Description:

Identifier: Automatic

Value

☐ True (True)

☐ False (False)

☒ String (text)

10

☒ Number

10

General

Name: Mi variable

Description:

Identifier: Automatic

Value

☐ True (True)

☐ False (False)

☒ String (text)

Buenos dias

☐ Number

0

Comandos relacionados: [Set](#), [Compare](#), [Increment/Decrement](#)



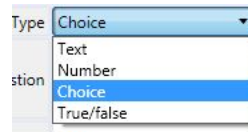
Wherigo for dummies inputs



The screenshot shows the Wherigo software interface with the following components:

- Input Configuration:** A panel on the left showing the configuration for an input object. The 'Type' is set to 'Choice'. The 'Question' is '¿que haces?'. The 'Input choices' are 'uno', 'dos', 'tres', and 'cuatro'.
- Events:** A panel on the left showing the events for the input object. The 'On get input' event is set to 'handled' and the 'On error' event is set to 'handled'.
- Object Properties:** A panel on the right showing the properties of the input object. The 'Name' is 'Pregunta5'. The 'Description' is empty. The 'Identifier' is 'Automatic'.
- Display:** A panel on the right showing the display settings. The 'Display' checkbox is checked. The 'Image' and 'Icon' are set to 'None'.
- Input Choices:** A panel on the right showing the input choices. The choices are 'uno', 'dos', 'tres', and 'cuatro'.
- On start event:** A diagram showing the 'On start' event triggering the 'Input' object 'Pregunta5'.
- Pregunta5 object:** A diagram showing the 'Pregunta5' object containing an 'Input' object, which in turn triggers a 'Set' action: 'Mi variable = Answer'.

- Sirven para: preguntar cosas al jugador o hacer una selección entre varias opciones.
- Tenemos que elegir el tipo de dato que queremos entrar, puede ser: un texto, un número, una opción verdadero/falso y que el jugador seleccione una opción posible entre varias.



- Se puede utilizar de manera similar y en sustitución de los comandos que tienen los objetos.
- Primero creamos el objeto Input como el resto de los objetos. Luego le asignamos comandos al evento On get input el cual se genera cuando el usuario introduce la entrada y al evento On error que se genera cuando el tipo de entrada del jugador no coincide en tipo con el esperado, v.g. si pedimos un número y introducimos una palabra.
- En el evento On get input debemos utilizar Answer que es la variable donde está la entrada del jugador. Si el valor lo vamos a utilizar más tarde lo guardamos en una variable o podemos usarlo en un [if/then](#) para discriminar la respuesta.
- Para llamar a la entrada y que sea visible para el jugador, utilizamos la acción [Input](#).

[Ejemplos](#) Comandos relacionados: [Input](#), [Answer](#), [Variable](#)

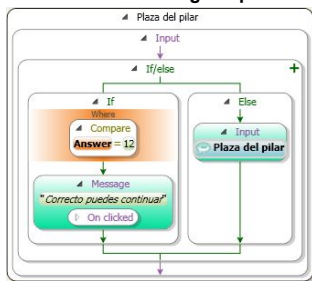


Wherigo for dummies ejemplos inputs



1.- Creamos el objeto

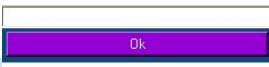
2.- Creamos los eventos On get input



Esto es lo que verá el jugador



Cuantas farolas hay en la plaza?



2.- y On error



3.- Presentamos la pregunta al jugador

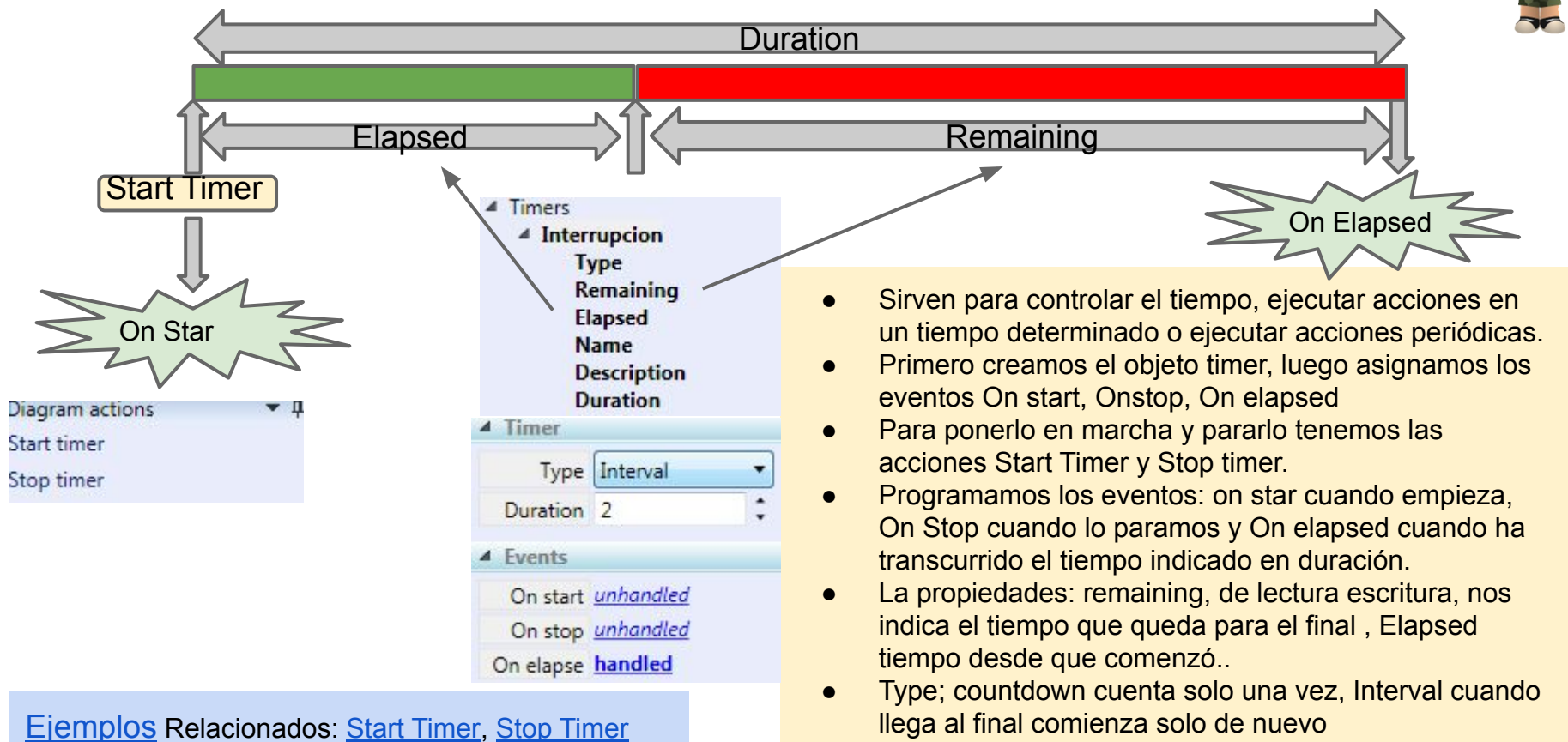


- Sirven para: preguntar cosas al jugador, o hacer una selección entre varias opciones.
- Tenemos que elegir el tipo de dato que queremos entrar, puede ser: un texto, un número, una opción verdadero o falso o que el jugador seleccione una opción posible entre varias.
- Se puede utilizar de manera similar y en sustitución de los comandos.
- Primero, se crea el objeto, eligiendo el tipo de entrada que queremos. Aquí ponemos también la pregunta.
- Segundo, Asignamos las funciones a los eventos *On get input* (cuanto el usuario introduce la entrada) y *On error* (si la entrada no coincide en tipo). Para que todo sea correcto debemos gestionar ambos. En *On error* lo más normal es repetir la pregunta. *On get input*, verificamos si la pregunta es correcta o guardamos el valor elegido.
- En el punto adecuado de nuestro programa presentamos la pregunta al jugador utilizan la acción Input.

Ejemplos Comandos relacionados: [Input](#), [Input](#), [Answer](#), [Variable](#)



Wherigo for dummies timers



Ejemplos Relacionados: Start Timer, Stop Timer



Wherigo for dummies ejemplos timers



Un cronómetro

Creamos el timer

Creamos el evento

Ponemos en marcha el cronómetro

Se genera un cronómetro de 10 Sg una sola vez, cuando transcurren estos, se genera el evento On Elapsed en el cual escribimos el mensaje "Ya han pasado 10 Sg". Solo se generan los eventos On Start al ponerlo en marcha (lo hemos dejado unhandler y On elapse que escribe el mensaje.

Un Intervalo

Creamos el timer

Creamos los eventos On star y On stop

En este caso es un timer de tipo Interval por lo que está generando el evento On elapse cada 10 Sg. El evento On star se genera cuando el programa lo pone en marcha y el evento On stop se genera cuando se para el timer. La propiedad Intervalo.remaining nos indica cuanto falta para el nuevo evento On elapsed y la propiedad Intervalo.elapse nos da el tiempo transcurrido desde el último evento On elapsed.

Dos timer simultáneos

Dos timers simultáneos Intervalo cada segundo y Cronómetro 10 Sg. Cada segundo por la interrupción On elapsed de Intervalo mostramos los valores elapsed y remaining de cronómetro. En pantalla vemos los dos valores de cronómetro que se refrescan cada segundo por medio del evento de intervalo. En evento On stop no se genera nunca

Contador de minutos

Contador de minutos. Tenemos un timer (Intervalo) que genera un evento On elapsed cada 60 Sg y tenemos un a variable (minutos). Cada vez que se genera el evento On elapsed se incrementa el valor de la variable minutos. Cuando paramos Intervalo se genera un evento On stop el cual no muestra un mensaje con el número de minutos transcurridos.



Wherigo for dummies funtions



Properties

General

Name	Name
Description	
Identifier	Automatic

Events

On call [unhandled](#)

Sirve para agrupar un conjunto de acciones y expresiones y así poderlas usar varias veces en el programa sin tener que escribir otra vez el código.

Las acciones que queremos que se hagan dentro de la función, las escribimos en el evento On call de la misma función.

Cuando queramos que se ejecuten estas acciones solo tenemos que llamar a la función utilizando el comando Function call.

Podemos llamar a la función todas las veces que necesitemos y desde cualquier parte del programa.

Function call

Name



Wherigo for dummies media



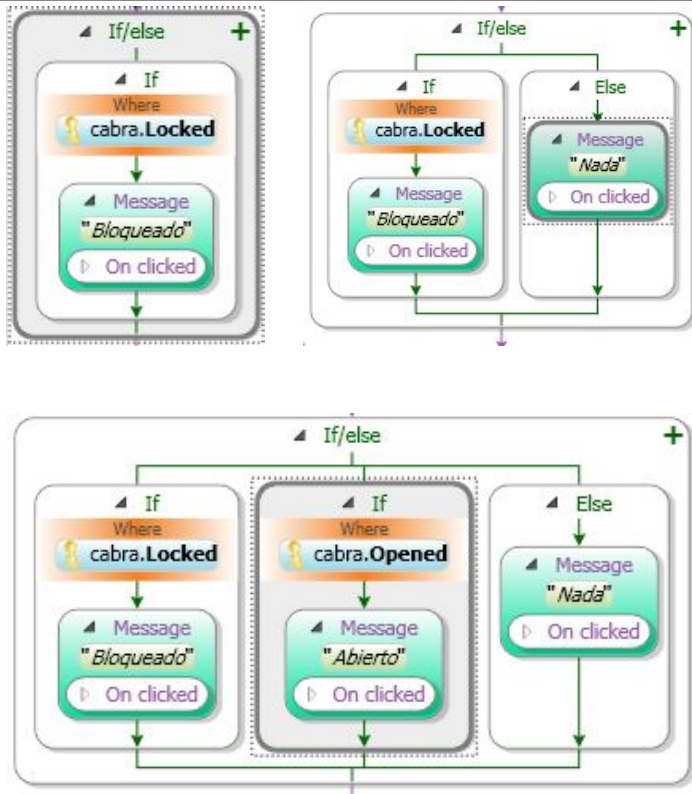
- MEDIA son todos las imágenes y sonido que aparezcan en nuestro juego.
- Las imágenes deben de tener una anchura máxima de 230 pixel la altura entre 170 pixel para Garmin. Si queremos que quede sitio debajo no deberíamos hacer la imágenes mayores de 230x120. En Garmin se pueden reproducir sonidos en formato FDL, Podemos encontrar algunos: [sonidos fdl](#)
- Para pocket pc(QVGA)250 por 130-180 . Si queremos ejecutar en Android podemos poner imágenes de hasta 460x340.
- Podemos asignar dos ficheros de medios, uno para Garmin y otro para pocket pc (Android, Iphone...). Para ello al asignar el fichero, pondremos uno con nombre fichero_garmin.jpg y otro con el nombre fichero_ppc.jpg. Asignaremos ambos al mismo media.
- Los iconos tienen que ser de 32x32
- Poner archivos lo más pequeño posible.
- Si creamos un media y no le asignamos un archivo se nos genera un error: "**medium must have at least one resource**".
- El tamaño en el cual se mostrará la imagen lo marca los puntos por pulgada a los cuales hayamos generado la imagen.

[Play sound](#), [Stop sound](#)



Wherigo for dummies

if/then



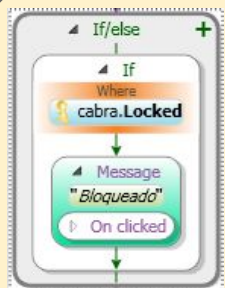
- Es uno de las acciones más usadas en la construcción de cartuchos.
- Evalúa una propiedad o variable (debe de devolver verdadero o falso (true/false)), si es verdadero ejecuta las acciones que hayamos puesto detrás , en caso contrario, ejecuta las acciones que haya en el else, si las hay.
- En caso de haber varios caminos de if dentro de una misma acción, se evalúan de izquierda a derecha y sólo ejecutará el primero que cumpla la condición, no evaluándose las demás.
- Cuando la propiedad que queremos evaluar devuelve un valor distinto de verdadero falso, debemos de combinar el if/else con la función [compare](#) la cual compara dos cosas pudiendo elegir si queremos que sean iguales o distintas y devuelve verdadero (true) cuando se cumple la opción seleccionada.
- Cuando queremos evaluar varias condiciones simultáneas utilizaremos [and](#), [or](#)

[Ejemplos](#) Comandos relacionados: [Compare](#), [and](#), [or](#)

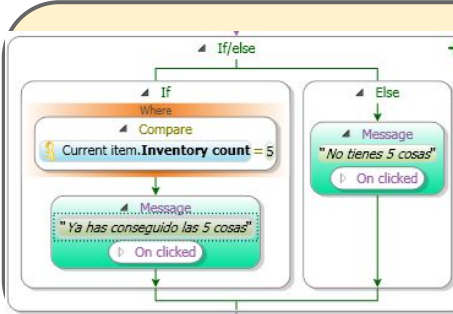


Wherigo for dummies

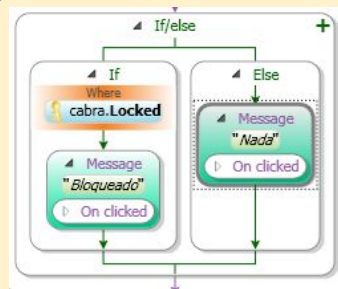
Ejemplos if/then



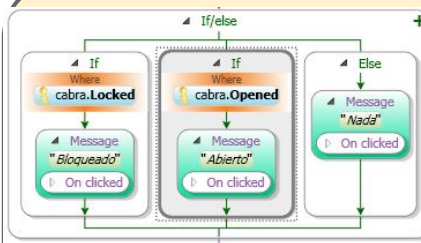
Evalúa la propiedad `cabra.Locked` si es verdadero (true) nos muestra el mensaje "bloqueado" en caso contrario no hace nada.



Comprueba que la cuenta de inventario es igual a 5 si es verdadero escribe el mensaje "ya has conseguido las cinco cosas" y si la cuenta de inventario no es 5 no da el mensaje "No tienes 5 cosas"



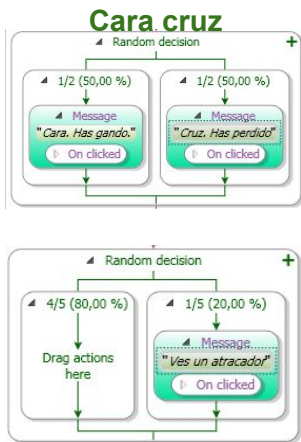
Evalúa la propiedad `cabra.Locked` si es verdadero (true) nos muestra el mensaje "bloqueado" en caso contrario nos muestra el mensaje "nada"



Primero evalúa la propiedad `cabra.Locked` si es verdadero (true) nos muestra el mensaje "bloqueado" en caso contrario, evalúa la propiedad `cabra.Opened` si es verdadero nos muestra el mensaje "abierto", si es falso no muestra el mensaje "nada"



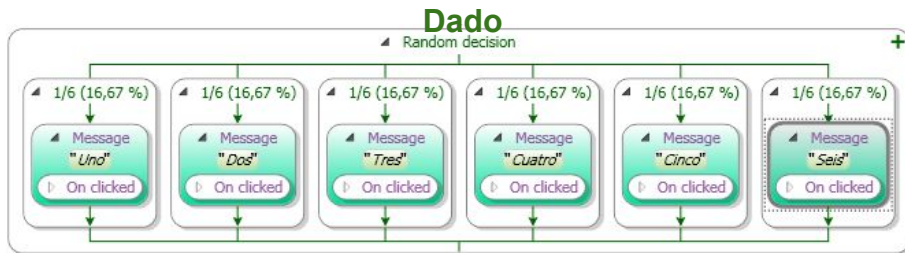
Wherigo for dummies random



Esta acción sirve que el programa haga aleatoriamente acciones distintas. Es similar a tirar un dado o una moneda y que la acción transcurra por sitios distintos dependiendo del valor sacado.

- Tenemos que seleccionar qué probabilidad le asignamos a cada opción.

- Cuanto mayor sea el valor, mayor es la probabilidad.





Wherigo for dummies increment/decrement



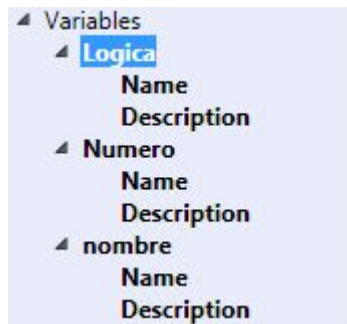
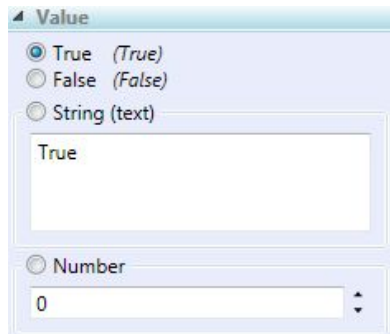
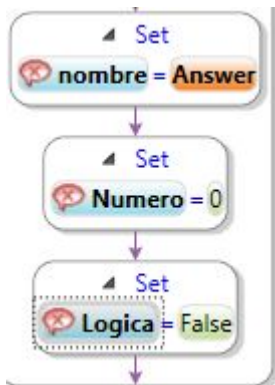
Con esta acción podemos aumentar o disminuir el valor de una variable numérica.

Poner un valor positivo para aumentar y un valor negativo para disminuir.

Ver también: [numeric operation](#), [Variable](#)



Wherigo for dummies set



Sirve para asignar un valor a: una variable o a una propiedad en tiempo de ejecución, que sea de lectura/escritura, un valor nuevo.

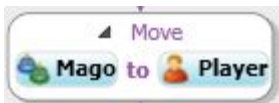
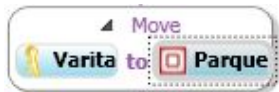
- Tenemos que arrastran una cosa a cada lado del igual. Lo que pongamos a la izquierda tiene que ser escribible, es caso contrario nos dará un error: **Left expression is read only.**
- Lo que pongamos a la derecha tiene que coincidir en tipo con lo que tengamos a la izquierda. En caso contrario nos da un error: **Right expression must be convertible to...**

Ver: [Propiedades en tiempo de ejecución](#), [Variable](#), [Value](#)





Wherigo for dummies move

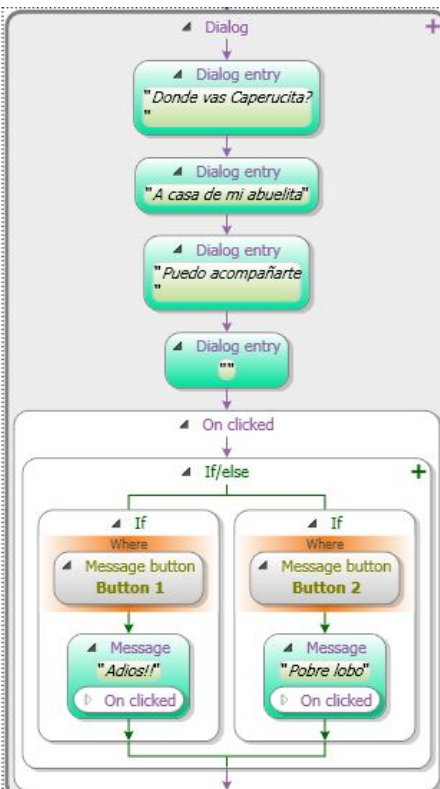


Con esta acción, podemos cambiar el lugar donde se encuentra un [Items](#), personaje ([Caracteres](#)) de un lugar a otro.

- Pueden ser movidos los objetos, los personajes .
- Se pueden **mover a**: una zona, un personaje y al jugador.
- Cuando un objeto es movido a otro este pasa a formar parte de la cuenta de inventario del objeto al cual se mueve.
- Si ponemos la varita en el mago, el mago en el jugador y el jugador en el parque, la cuenta de inventario de cada uno de ellos valdrá uno porque no se cuentan los objetos que están dentro de otros.
- Podemos detectar si un objeto está dentro de otro por medio de [Contain](#). Si preguntamos si parque contiene la varita nos devolverá verdadero. Porque la varita está en la caja que tiene el mago que esta en el parque.
- He detectado que cuando un objeto está dentro de otro hay problemas con la cuenta de inventario del objeto contenedor, parece un Bug del programa por lo que recomiendo no utilizar un objeto dentro de otro.

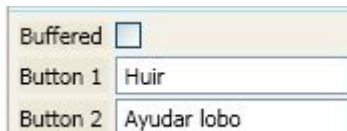


Wherigo for dummies dialog



Es la forma que Wherigo nos ofrece para hacer diálogos entre los personajes o del personaje con el jugador. A un diálogo le podemos añadir tantas entradas como queramos, cada entrada puede tener su propia imagen y su propio texto. Podemos añadir dos botones que son únicos para todo el dialogo y solo se muestran junto con el último Dialog entry. Después de cada entrada nos muestra un botón OK que nos permite pasar a la siguiente. En la última se nos muestran los dos botones donde podemos elegir una acción entre dos posibilidades.

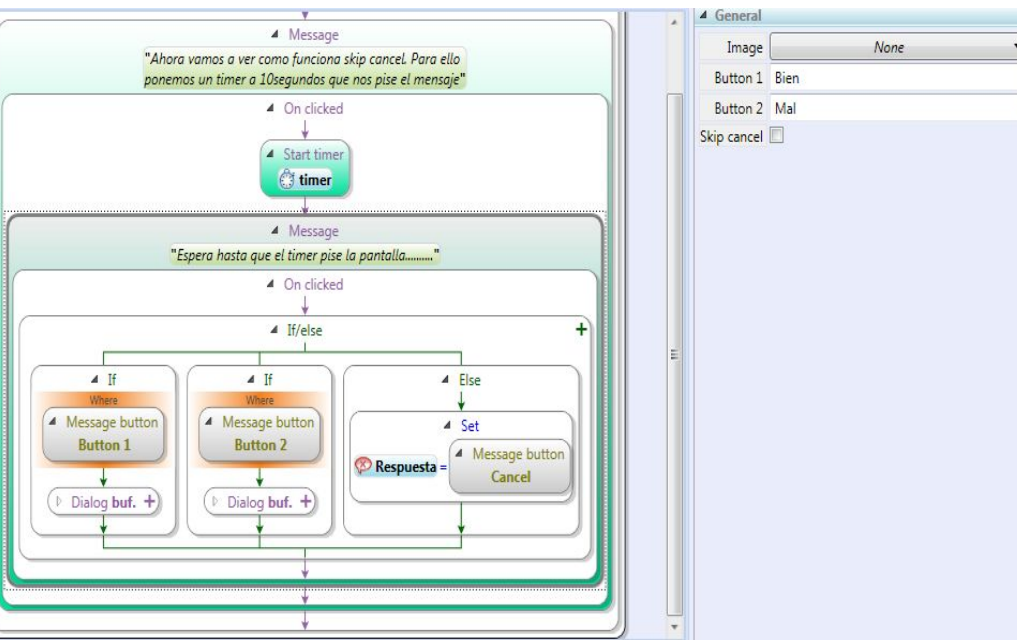
- Buffered el diálogo no se muestra instantáneamente, se pone en la pila y espera al siguiente objeto que utilice la pantalla para presentarlo junto con él. Con esto evitamos que los mensajes se sobreescriban en la pantalla.
- Junto con cada Dialog entry se muestra una imagen la cual puede pertenecer a un personaje, pero no es necesario que exista el personaje. Lo único necesario es haber creado la imagen como Media





Wherigo for dummies

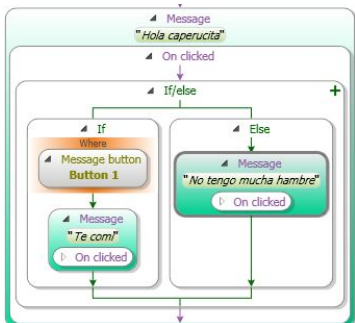
Buffered y skip cancel



- Estos dos comandos tienen sentido porque el programa se ejecuta de manera recurrente. Esto quiere decir que mientras estábamos viendo un mensaje o esperando una entrada, puede saltar un evento de que se ha entrado en una zona o ha acabado un timer. Si estos eventos utilizan la pantalla, pisarán la presentación que hay en ese momento y se perderá la selección o el mensaje.
- Como hemos comentado, buffered sirve para retardar la presentación del mensaje y hacerlo junto con el siguiente. Lo pone en la cola de mensajes y cuando llega a un mensaje que hay que presentar se sacan primero todos los que haya en la cola.
- Skip cancel (sin seleccionar), cuando la pantalla es pisada, ejecuta el evento on click y pone el botón cancel a verdadero. Gracias a esto sabremos que se ha pisado la pantalla y obrar en consecuencia.



Wherigo for dummies message



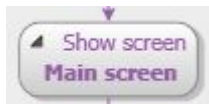
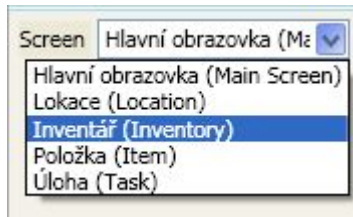
Es la manera básica de interacción con el usuario. Este comando nos permite mostrar un texto sin imagen, una imagen sin texto o ambas cosas juntas. Además le podemos asignar dos botones, a los cuales les podemos personalizar los textos y que permiten que el jugador haga una selección entre dos opciones.

- El mensaje a mostrar tiene que ser una cadena de texto o una variable o una propiedad de un objeto.
- Si queremos hacer un mensaje que combine a la vez varios elementos, usaremos la expressions [Concatenate](#)
- Además del texto podemos asignar una imagen en la propiedad Image.
- Los textos que aparecen en los botones son configurables.
- Para saber que botón ha pulsado el jugador deberemos utilizar [Message button](#) .
- [Skip cancel](#): cuando otra ventana sobreescribe el mensaje mientras se espera a que el jugador pulse un botón, podemos elegir lo que ocurre. Si Skip Cancel esta picado, no se genera el evento Onclick y la respuesta se pierde. Si Skip Cancel no esta picado, se genera el evento On click y [Message button](#) contendrá Cancel por lo que podemos hacer una acción para corregir la sobreescritura.



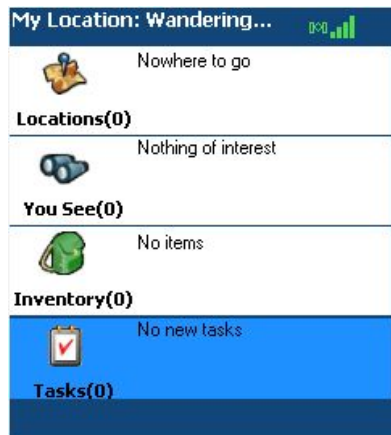
Wherigo for dummies

Show screen



Nos muestra la pantalla del jugador que selecciones, podemos elegir entre la pantalla principal, el inventario, los Items o las tareas.

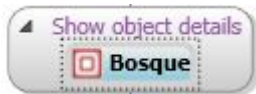
Si queremos que el cartucho se pueda jugar en un Garmin la única opción que debemos seleccionar es Main screen. El resto bloquean el GPS inmediatamente.





Wherigo for dummies

Show object detail



Sirve para mostrar los detalles de un objeto: imagen y comentario. Es una manera de indicar al jugador donde tiene que fijar la atención. Si al entrar en una zona en la que hay un objeto queremos que se fije en él, ponemos un Show object details del objeto para que sea evidente que está ahí.

Si aparece un personaje y queremos que el jugador lo sepa podemos mostrar un diálogo con él o mostrar los detalles con esta acción.

Podemos utilizarlo con Items, Characters y Zones



Wherigo for dummies input (acción)



Input

Type: Choice

Question: ¿que haces?

Input choices: uno, dos, tres, cuatro

Events: On get input [handled](#), On error [handled](#)

Inputs

- Acciones**
 - Display
 - Image
 - Icon
 - Name
 - Description
 - Question

Sirve para presentar una pregunta que hayamos creado previamente en pantalla.

Hay dos input, el [Input](#) objeto y el input acción, se utilizan conjuntamente.

- Primero creamos la pregunta creando un nuevo objeto Input .
- Luego la mostramos al jugador utilizando la acción Input para mostrar la pregunta en pantalla.

Es decir, input acción llama a input comando.



Diagram actions

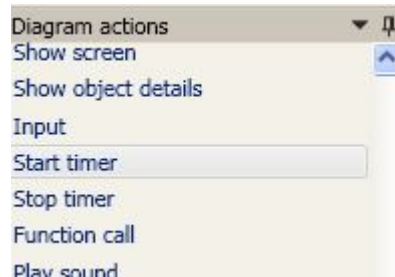
Show object details

Input



Wherigo for dummies

Start Timer / Stop Timer



Start Timer

Pone en marcha un timer. Tenemos que arrastrar a la ventana el timer que queremos poner en marcha. Junto con esta acción se genera el evento Onstart del timer correspondiente.



Stop Timer

Detiene un timer, tenemos que arrastrar a la ventana el timer que queremos parar. Junto con esta acción se genera el evento On stop del timer correspondiente. Al parar el timer se resetea el valor de la cuenta y cuando volvemos a empezar, comienza de nuevo en el valor marcado en duración.



Wherigo for dummies

Play sound/ Stop sound



Play sound pone en marcha un sonido de sonido. El sonido tiene que estar en un fichero que hemos tenido que agregar como media.

El sonido para solo cuando llega el final del fichero. Si hemos puesto un sonido muy largo y queremos que se detenga antes tenemos que usar la acción Stop sound la cual para todos los sonidos que están sonando ese momento..



Wherigo for dummies

Save/Save and exit



Save game

Save and close game

Guarda la situación actual del juego de forma que si se cuelga, no funciona o queremos continuar con el desarrollo del cartucho otro día podemos volver al punto justo donde estábamos.

Para restaurar el juego de nuevo al punto en el que estábamos debemos salir del juego y volver a entrar, el cartucho detecta que tenemos el juego guardado y nos pregunta si queremos restaurar a ese punto. Save y close game es lo mismo que save pero además nos cierra el juego.

Podemos salvar el juego las veces que queramos pero cada vez que salvamos sobreescibe a la anterior.

El fichero en el que se guarda el juego tiene extensión GWC.

Los cartuchos guardados se generan de manera única para cada dispositivo y jugador que descarga el cartucho. Por tanto compartir el fichero de un juego guardado no funciona.

Podemos hacer que el juego se guarda automáticamente asignando al evento global de zonas On exit el comando save, así cada vez que salgamos de una zona se salva el juego. También podemos hacerlo por tiempo creando un timer que vaya salvando periódicamente el juego.

En los dispositivos Garmin el comando Save and exit, no funciona, continua el juego sin salvar.



Wherigo for dummies compare



Expressions

Compare

And

Or

Operator equals (=)

Ignore case equals (=)

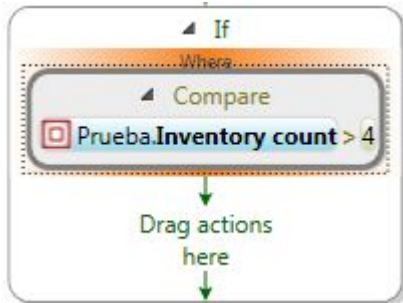
not equal (≠)

less than (<)

greater than (>)

less or equal (≤)

greater or equal (≥)



Operator equals (=)

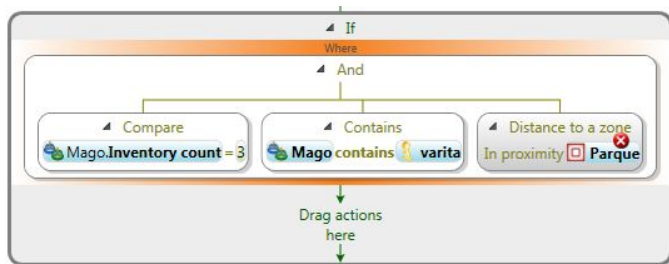
Ignore case ☒

Sirve para hacer comparaciones. Podemos comparar si son iguales, distintos, >, <.. dependiendo de la selección que hagamos con la casilla Operator. Se pueden comparar variable lógicas (true/false), cadenas o números, dependiendo de qué expresión pongamos. Siempre tenemos que evaluar cosas iguales. En caso de comparar cadenas (string) podemos seleccionar la casilla Ignore case en cuyo caso da igual que la cadena este en mayusculas o minusculas. MiTesoro será igual a mitesoro.

Comandos relacionados: [If/else](#), [and](#), [or](#)

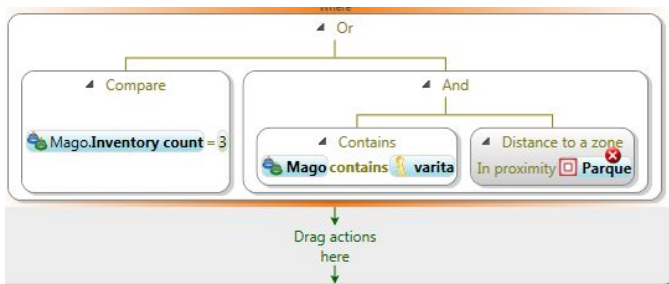


Wherigo for dummies and/or



AND

Realiza la función and lógica entre dos o más variables expresiones se devuelven verdadero/falso. La función and implica que para que sea verdadero todas ellas deben devolver verdadero, con que una sola devuelve falso el resultado es falso.



OR

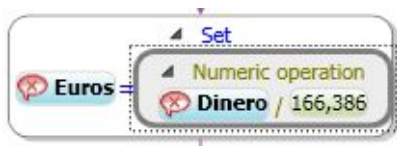
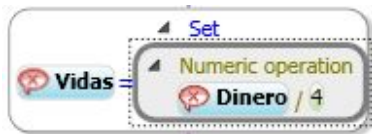
Realiza una función lógica or entre dos o más variables o expresiones que devuelven verdadero/falso. La función or devuelve verdadero si una de las variables es verdadera.

And y or se pueden combinar entre sí en expresiones mas complejas.



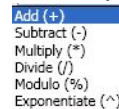
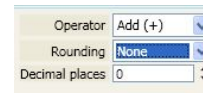
Wherigo for dummies

numeric expresion



Con esta expression podemos hacer una operación matemática sobre una variable.

- Tenemos tres opciones: tipo de operación, redondeo del resultado y número de decimales.
- Las posible operaciones son: suma(+), resta(-), multiplicación(*), división(/), calcular el resto de una división (%), potencias (^)



- El redondeo puede ser: ninguno, siempre hacia arriba (1.1=2), siempre hacia abajo (1.9=1) o redondeo (1.5=1 o 1.6=2)



- Número de decimales a presentar dos 3.14 ; cuatro 3.1416



Wherigo for dummies contain

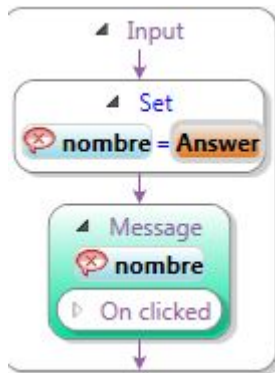
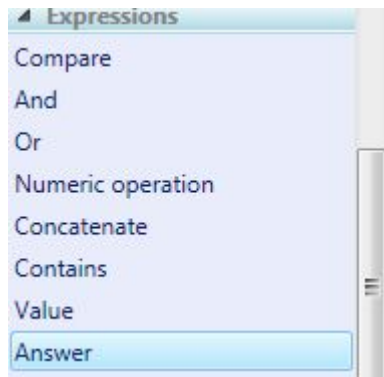


Sirve para saber si un objeto o personaje está en un determinado sitio. A contain le tenemos que pasar dos valores. En el ejemplo, si la piedra está en el parque, nos devuelve verdadero (true) y si no esta nos devuelve falso (false). También devuelve true cuando un objeto contiene a otro que a su vez contiene otro.



Wherigo for dummies

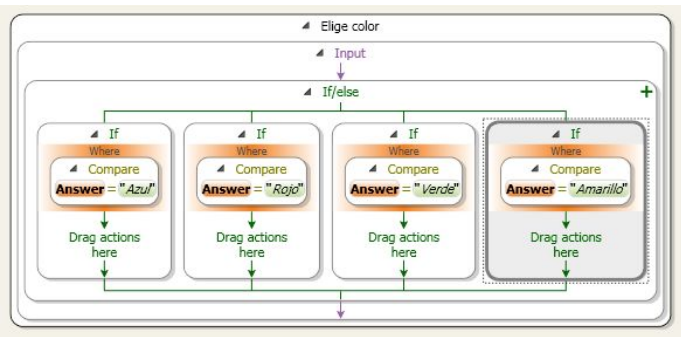
Answer



En esta expresión está la respuesta que el jugador ha dado a una pregunta ([Input](#)).

Su uso es como el de cualquier otra variable, pero su contenido desaparece cuando salimos del evento que lo gestiona por lo que si queremos usarlo posteriormente tenemos que guardar su valor en otra variable.

Si queremos ejecutar una acción dependiendo de la respuesta, entonces utilizamos un If/else comparando con la expresión Answer.

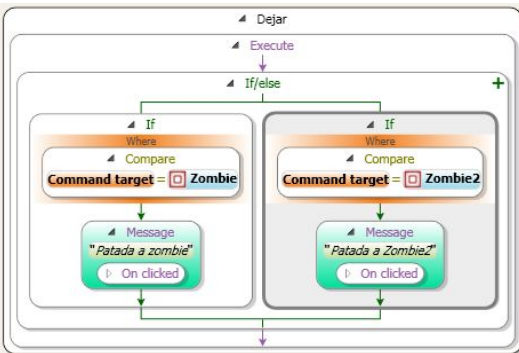


[Ejemplos](#) Comandos relacionados: [Input](#), [Input](#), [Answer](#), [Variable](#)



Wherigo for dummies

command target

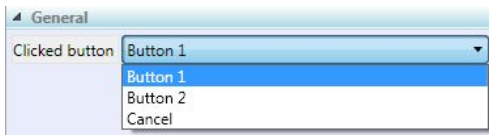


Se usa en conjunción con los comandos que podemos añadir a los objetos, los personajes y los lugares. Sirve para gestionar en los comandos que afectan a varios objetos. Sólo se pueden utilizar dentro de la rutina que atiende al evento ligado al botón. Si el comando no tiene acceso a alguno de los objetivos (target) el comando no se ejecuta.

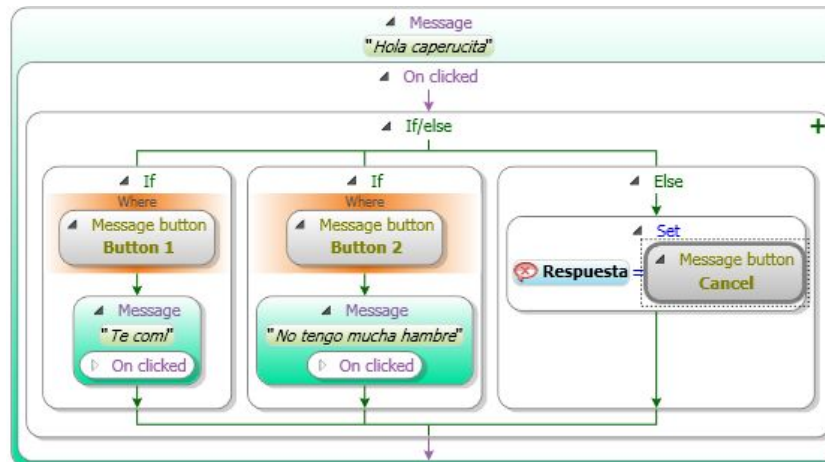
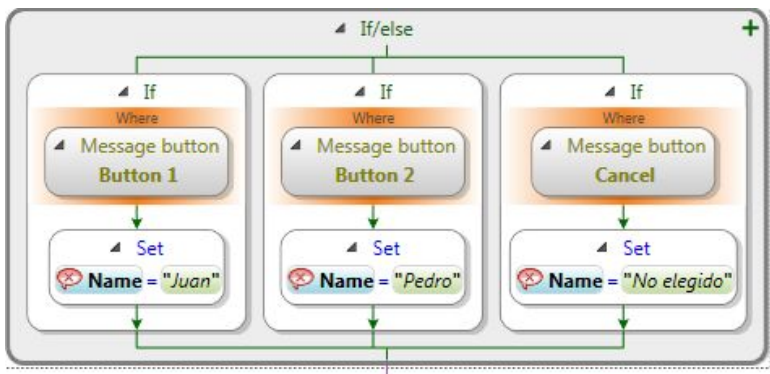
El único uso que he encontrado es el de utilizarlo en una comparación para ver cual fue el objeto objetivo. Este comando trabaja con la variable target que contiene una copia del objeto destino. Por tanto utilizando Lua code, podemos escribir algo como `target.Active=false` y desactivamos el objeto.



Wherigo for dummies message button



Esta expresión se utiliza conjuntamente con [Dialog](#) y [Message](#) . Sirve para saber que botón ha sido pulsado en el evento On click del mensaje correspondiente.. La expresión devuelve true en caso que el botón seleccionado haya sido pulsado. Podemos seleccionar tres posibles botones a comprobar el 1, el 2 y cancel . Cancel devuelve true en caso de que la ventana haya sido sobrescrita pero sólo en los mensajes y teniendo desactivado [Skip cancel](#) .



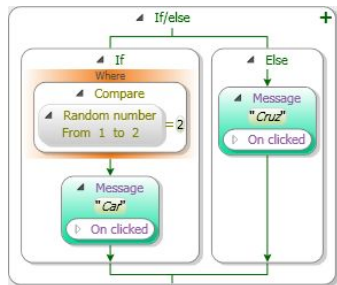


Wherigo for dummies

random number



Dado



Genera un número aleatorio entre dos valores que podemos seleccionar.

From	<input type="text" value="1"/>
To	<input type="text" value="5"/>

- Los números pueden ser positivos o negativos.
- Tienen todos la misma probabilidad de salir.
- Combinandolo con if/else tiene un uso muy similar a [Random](#)

Ver también: [Random](#)



Wherigo for dummies

Notas y trucos

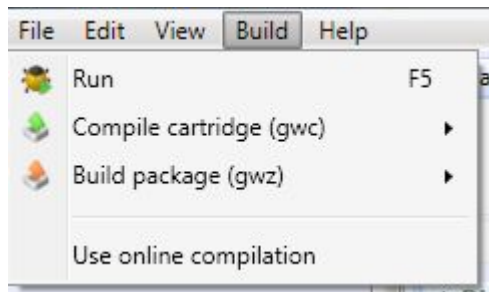


- Todos los eventos programados se ejecutan de manera concurrente, lo cual significa que cuando el programa queda parado esperando la acción del jugador, por ejemplo cuando está esperando una entrada o que pulse un botón, el programa se sigue ejecutando y por tanto las acciones pueden variar dependiendo del momento en que se pulse el botón. Esto implica, además, que como la pantalla es única si dos mensajes llegan a ella el último sobrescribe al primero y por tanto el mensaje que hubiera no es leído por el jugador. Para evitar esto en las entradas hay varios mecanismos implementados. En los [Message](#) está la opción [Skip cancel](#) que no evita que las ventanas se sobrescriben pero informa al programa levantando el evento On click con [Message button](#) igual a cancel. Con esto el programa sabe que ha sido pisado y que no se han ejecutado las acciones posteriores. En el caso de los [Dialog](#) se ha implementado la propiedad buffered que lo que hace es retrasar la aparición de los diálogos hasta el próximo uso de pantalla poniéndolos delante de este, pero si una ventana de diálogo es pisada por otra nunca llega a generarse el evento On click por lo que no podemos hacer nada.. En las [Input](#) no se ha previsto ningún mecanismo por lo que al programar el cartucho deberíamos desactivar todos los posibles eventos que pudieran sobrescribir la pantalla para que no haya problemas.
- Aunque a las zonas se le pueden poner comandos o hacerlas objetivo de comandos, luego en el programa de Android esto no se soporta. Los comandos solo con Characters e Items
- Aunque ZObject tiene la propiedad para saber la distancia del jugador a cualquier objeto(todos heredan de ZObject) la realidad es que en el programa de Android solo se calculan las distancias a las del jugador a las zonas. Si queremos saber la distancia a un objeto tenemos que utilizar `Wherigo.VectorToPoint(obj1,obj2)` el cual nos devuelve un vector con la distancia y el rumbo al objeto.



Wherigo for dummies

Ficheros



Existen tres tipos de ficheros:

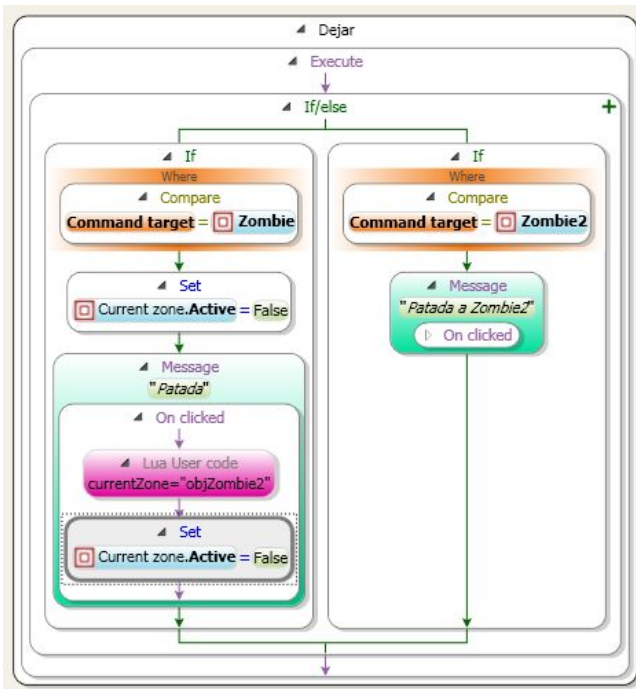
- GWC es el cartucho ya compilado y el que ponemos en nuestro dispositivo para jugar. Se obtiene compilando el GWZ
- GWZ es un fichero ZIP que podemos abrir utilizando un descompresor. En el encontramos un fichero con extensión .lua que es el que contiene el código del cartucho y todos los ficheros auxiliares (imágenes y sonidos) que utilizamos en el juego. Este es el fichero que tenemos que subir a la página de Wherigo.
- GWL es un fichero de log que se genera para usarlo en la depuración de los cartuchos. Para que el fichero se genere, tenemos que tener activada la casilla login del objeto cartucho
- GWS es el fichero que se genera cuando guardamos nuestro cartucho y el que utilizamos para volver al mismo sitio cuando volvemos a ejecutar un cartucho..

Desde Urwigo tenemos 3 opciones compilar el cartucho gwc, esto nos sirve para poder probar nuestro cartucho en un GPS o teléfono antes de subirlo, la compilación la hace Urwigo en nuestro ordenador. Generar el archivo GWZ que es el que tenemos que subir a la página de Wherigo para compartirlo. Usar compilación online genera el GWC pero en lugar de hacerlo en nuestro ordenador lo envía a la página para compilarlo, esta es más real pues es la que posteriormente se distribuirá.



Wherigo for dummies

Current object



- El único current object que está programado es Zone. En los eventos relacionados con las zonas, al producirse el evento lo primero que hace es asignar la zona que ha producido el evento a la variable `currentZone`. En el resto creo que todavía no funciona.
- Falta estudiar como se hace `_G[currentZone].Active=true;` --en el emulador
-

Este ejemplo primero desactiva una zona y luego la otras.

Ver también:



Wherigo for dummies

Acentos

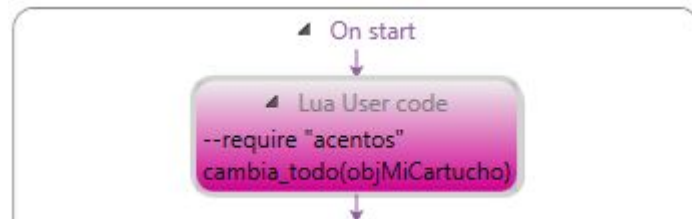


Una de las cosas que peor quedan en los cartuchos es no poder utilizar la ñ, los acentos y la diéresis. El problema se debe a que el compilador de Wherigos utiliza un código de caracteres de 7 bits y por tanto no puede manejar los acentos. Sin embargo, los programas que corren los Wherigos en los dispositivos utilizan un juego de caracteres de 8 bits y por tanto si pueden utilizar acentos. Para solucionar este problemas, lo que necesitamos hacer es que el cartucho se suba a la página de los wherigos sin los caracteres especiales y cuando el programa esté corriendo en el terminal se sustituyan los acentos. Para hacer esta función he creado el fichero `acentos.lua` añadiendo este fichero a nuestros cartuchos podremos utilizar los acentos la diéresis y la ñ.

- Poner el fichero `acentos.lua` en el mismo directorio que el cartucho
- En el campo Identifier de nuestro cartucho poner: "objMiCartucho"
- En el evento On Star del cartucho pondremos una Actions Lua User Code con el siguiente código:
`require "acentos"`
`cambia_todo(objMiCartucho)`

Con estos dos pasos ya tenemos todo listo para poder empezar.

Description	
Identifier	objMiCartucho





Wherigo for dummies

Acentos II



USO:

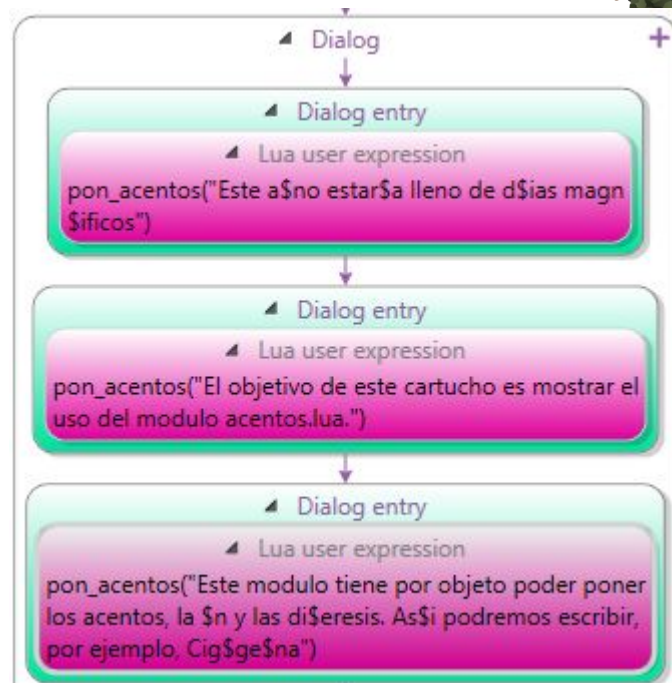
- Acentos: pondremos el carácter \$ seguido de la vocal a acentuar. Por ejemplo: d\$ia, m\$as, \$Arbol.
- La ñ: \$n para minúscula \$N para mayúscula. Ni\$ño, \$Nu
- Para la diéresis: \$g. Cig\$gue\$na

Mensajes y diálogos:

En lugar de poner el texto directamente, pondremos la acción Lua User Code y dentro, llamaremos a la función `pon_acentos("mensaje")`. Siendo mensaje el texto que queremos escribir.

Podemos poner acentos en las Input, en los nombres de los personajes, tareas etc. En el resto de los elementos se cambian los caracteres automáticamente, lo único que hay que hacer es poner el \$ delante de la vocal acentuada.

Nota: Los dispositivos Garmin no soportan los acentos de los comandos de los Items y Characters por lo que no se hace la sustitución.





Wherigo for dummies

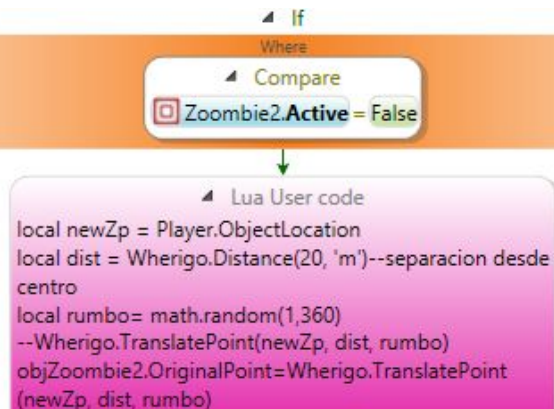
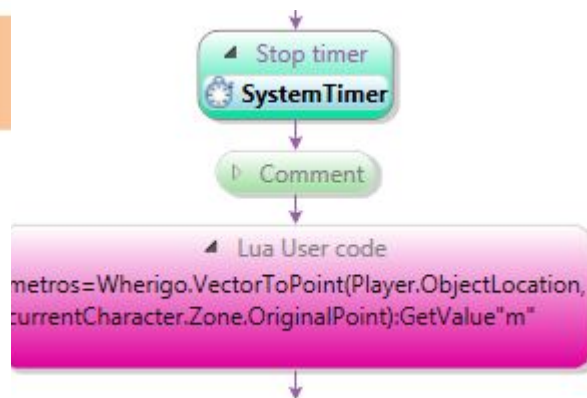
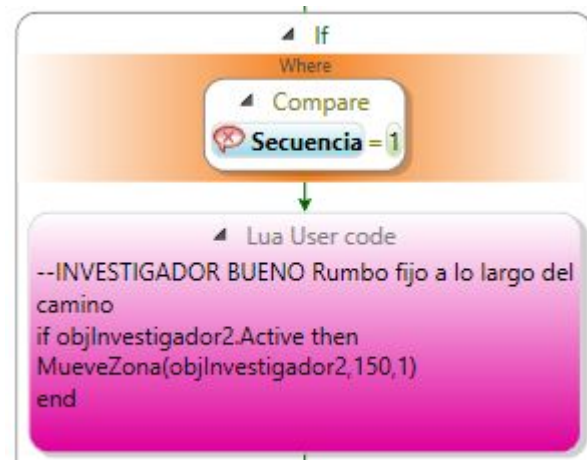
LUA User Code



Esta action nos permite introducir nuestro propio código Lua en un cartucho.

Gracias a este mecanismo podremos programar e integrar aquellas funciones que no se pueden hacer utilizando Urwigo.

Para generar una función debemos conocer el lenguaje de programación LUA





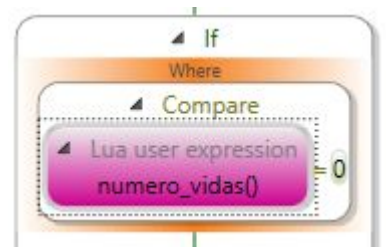
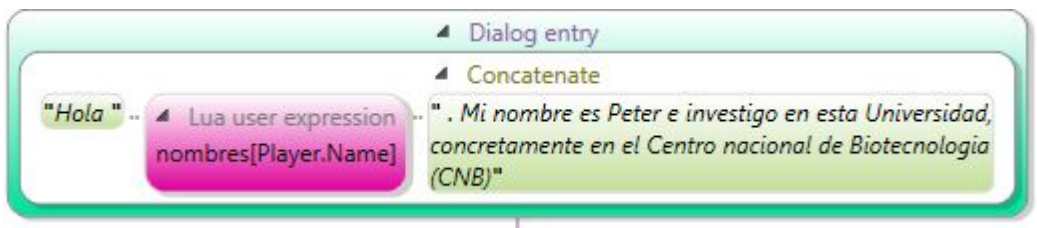
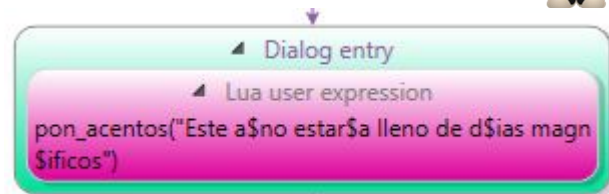
Wherigo for dummies

LUA User Expresion



Esta expresion nos permite introducir nuestro propio código Lua en un cartucho.

Esta pensado para integrar código en Urwigo y como es una expresion debe de devolver un resultado el cual será comparado con algo o un mensaje.





Wherigo for dummies

Lo que no funciona en un Oregon



How to make sure your cartridge works for all players

This section does not deal with the [VWB](#) directly but the fact that the [VWB](#) offers you all options that are specified for a Wherigo cartridge might get you in trouble. Some of the specified “functions” don't work on all devices. If you want to build a cartridge that runs on PDAs, Garmin devices and on cell phones you have to avoid these “[Critical Functions](#)”. This section tries to collect all known issues.

Commands created for a zone

- **Does not work on the Garmin Colorado**
- **Does (most likely) not work on the Garmin Oregon**

These commands are also known as OnZoneCommands. They create the possibility to have interactive zone screens by adding buttons to the zone screens. The “buttons” are also added to the event list of the zone. You could create a zone with different options when you use this function. This function is [critical](#) because it does not work on all Wherigo players.

A workaround could be to display a message after the zone has been entered and use the message buttons (if you want to offer 2 possible actions). If you want to be more sophisticated have a character appear once the zone is entered. The OnCharacterCommands are not critical.

Show Screen

- **Does not work on the Garmin Oregon**

The Garmin Oregon has a known bug that causes the device to freeze when the “Show Screen” command is used.

- Show Screen / Items
- Show Screen / Characters
- Show Screen / Zones
- Show Screen / Tools

Ver también: [Objetos Wherigo, LUA](#)





Wherigo for dummies

distance to zone

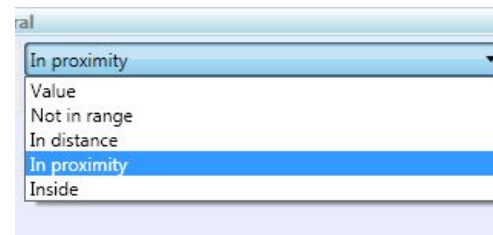
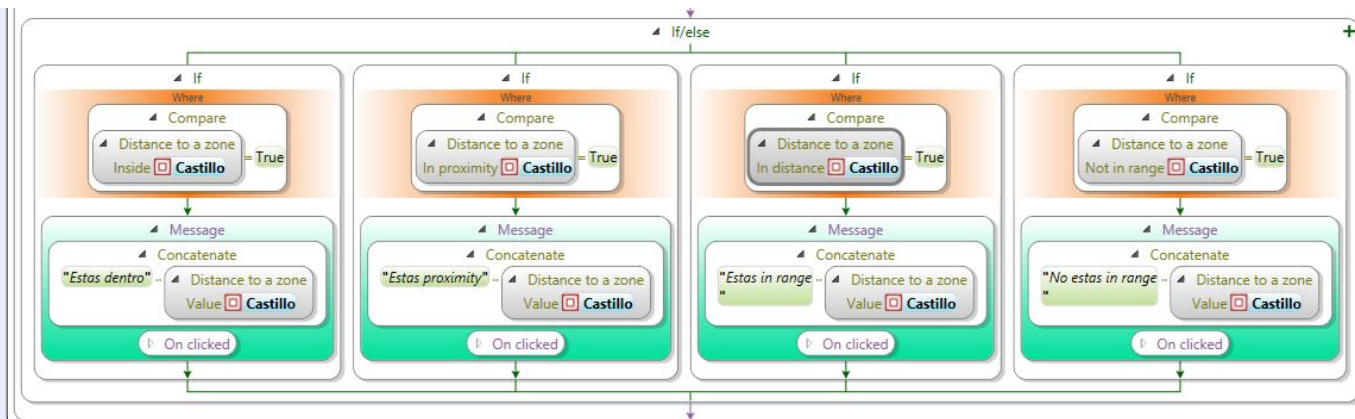


Este comando permite saber la posición del jugador con respecto a una zona.

Podemos elegir entre cinco opciones, las cuatro primeras devuelven verdadero o falso y se pueden usar junto con if compare, la otra devuelve un número:

- Inside: Devuelve verdadero si estamos dentro de la zona.
- In proximity: devuelve verdadero si estamos en la zona proximity.
- In distance: Devuelve verdadero si estamos en la zona range.
- Not in range: devuelve verdadero si no estamos en la zona range.
- Value: nos da la **distancia al centro de la zona** seleccionada. Es un valor numérico y lo podemos incluir en un mensaje

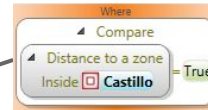
La distancia que nos muestra la pantalla de zonas nos da la distancia al borde más cercano de la zona. El comando distance to zone nos da la distancia al centro de la zona. Por eso verás diferencias entre uno y otro.



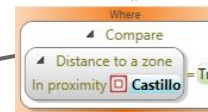
[Ejemplos](#) Comandos relacionados: [Zonas](#)



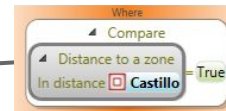
Wherigo for dummies distance to zone



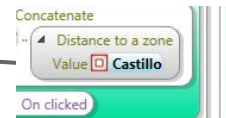
Devuelve verdadero o falso



Devuelve verdadero o falso



Devuelve verdadero o falso



Devuelve el valor numérico de la distancia **al centro** de la zona.



Devuelve verdadero o falso