

Tutorial

Wherigos mit Urwigo

von WhitePawn

Inhaltsverzeichnis

Was ist ein Wherigo?	4
Inhalt des Tutorials	4
Die Geschichte	5
Umsetzung	6
Objektorientierung - ein bisschen Theorie muß sein	7
Urwigo - der Editor	11
Einen neuen Wherigo beginnen	11
Das Hauptfenster	12
Bilder laden	13
Das erste Bild einbinden	17
Zonen anlegen	18
Wie geht's weiter?	22
Ein erster Test	27
Eine Aufgabe und ein erster Gegenstand	29
Die Aufgabe	29
So sieht's aus:	30
Der Ring	31
Eine erste eigene Funktion	32
Und so sieht's aus:	35
Betreten der ersten Zone	36
Noch ein kleiner Ausflug in die Informatik:	38
So sieht's aus:	38
Erste Interaktion	40
Neue Objekte	40
Die Unterhaltung mit dem Zwerg	42
Wie sieht's aus?	48
Zwei weitere Objekte	49
Ausprobieren	53

Das Verlassen der Zone	54
Wieder ein Test	58
Die Kapelle	59
Der letzte Akt	62
Tests vor Ort	67
Hochladen zu Wherigo.com	68
Was nicht mehr realisiert wurde	69
Tipps & Tricks	70
„Feindberührung“	70
„Schweizer Taschenmesser“	70
Komplexität	70
Von anderen „klauen“	70
Spieler in die Mitte der Zonen locken	71
Schluss jetzt	72

Was ist ein Wherigo?

Wherigo ist eine Plattform für GPS-gestützte Adventure-Games in der Realität und Cache-Typ bei Geocaching.com. Das GPS führt Spieler, die auf ihr GPS-Gerät – zum Beispiel das *Garmin Colorado 300*, die Garmin Oregon -Serie oder auch _Android- oder Symbian- Geräte – eine entsprechende sogenannte Cartridge laden, an eine gewünschte Stelle, von der sie dann mit virtuellen Objekten und Charakteren interagieren können.

Die Möglichkeiten von Wherigo sind sehr umfangreich, so kann man beispielsweise bei Annäherung an einzelne Orte bestimmte Bilder oder Texte mit Aufgaben anzeigen. Besonders reizvoll erscheint Wherigo in den Fällen, in denen nicht nur lineare Abläufe dargestellt werden, sondern der Anwender selber entscheiden kann, in welcher Reihenfolge er einzelne Stationen besucht.

Quelle: [Wikipedia](#)

Inhalt des Tutorials

Im Folgenden Tutorial soll gezeigt werden, wie mit Hilfe des Urwigo-Editors, sog. Cartridges für Wherigos programmiert werden können. Der Name Cartridge ist eine Hommage an die Spielkonsolen der 80er, als die Spiele noch in Form eines Moduls in die Spielkonsolen gesteckt wurden. Heute ist natürlich alles Software.

In diesem Tutorial soll anhand eines sehr einfachen Beispiels gezeigt werden, wie ein Wherigo programmiert werden kann.

Die Cartridge, die wir programmieren wollen, soll in Form eines kurzen Märchens präsentiert werden. Die Grundidee wurde aus dem bekannten Märchen "Das Wasser des Lebens" geklaut und ein bisschen an die Örtlichkeit angepaßt.

Das Abenteuer wurde recht kurz gehalten, um den Rahmen des Tutorials nicht zu sprengen. Wie wir sehen werden, ist aber auch für dieses kurze Szenario einiges zu tun und wollte man das Ganze soweit ausfeilen, daß es als fertiger Wherigo präsentiert werden kann, wäre noch einiges an Zusatzaufwand nötig.

Vom Stil her ist geplant, daß eher Kinder angesprochen werden.

Zusammen mit dem PDF wird ein ZIP-File mit dem Urwigo-Projekt zum Download angeboten. Dort kann man z. B. die notwendigen Bilder heraus kopieren oder die Umsetzung der einzelnen Schritte direkt nachvollziehen.

Die Geschichte

Das Abenteuer soll wie folgt aufgezogen werden:

- Ein junger Ritter zieht aus, um ein verwunschenes Schloß zu erlösen. Dazu hat er einen goldenen Ring erhalten, den er in die Kapelle der Burg bringen soll.
- Er trifft einen Zwerg, der ihn fragt, wohin er geht.
 - Gibt er dem Zwerg eine unwirsche Antwort, wird er verwünscht und das Spiel ist zu Ende
 - Gibt er dem Zwerg ehrliche Antwort, hilft ihm dieser weiter und gibt ihm einen silbernen Stab und ein Laib Brot
- Am verwunschenen Schloß angekommen, öffnet der Stab das Tor.
 - Dahinter liegt ein Löwe – dieser kann mit dem Brot besänftigt werden.
 - Geht der Spieler weiter ohne den Löwen gefüttert zu haben, frisst ihn der Löwe und das Spiel ist zu Ende.
- In der Kapelle leuchtet der Ring auf. Daraufhin soll er diesen in das Hauptgebäude tragen, um die Burg zu erlösen.
- Mit dem Ring wird dann eine Prinzessin erweckt, die den Spieler zu einem Schatz führt.

Anmerkung: In meinen "richtigen" Wherigos, die alle für Kinder konzipiert sind, vermeide ich es, daß das Spiel durch eine falsche Entscheidung beendet wird, um Frustration zu vermeiden. Zwar könnte man einen den letzten Spielstand laden und sich anders entscheiden, aber das ist ja doch eher nervig. Letzten Endes ist es eine Frage der Kreativität, Fehlverhalten zu "bestrafen" ohne daß der Spaß darunter leidet.

Hier im Tutorial habe ich es anders gehalten, so daß auch die Möglichkeit der Beendigung der Cartridge gezeigt wird.

Umsetzung

Nachdem nun die Geschichte steht und an die Örtlichkeit angepaßt wurde, sollten wir den Ablauf planen. In etwa soll der Ablauf wie folgt aussehen:



Verlauf des Wherigo

- Starten soll das Ganze auf dem Parkplatz vor der Ruine. Dort wird der Spieler in die Geschichte eingeführt und erhält erste Informationen.
- Danach wird er in Richtung eines Aussichtspunktes geführt. Dort trifft er auf den Zwerg.
- Es folgt die Unterhaltung mit dem Zwerg und bei erfolgreicher Durchführung, erhält der Spieler das Brot und den Stab.
- Damit ausgerüstet wird er zum Tor der Burg geschickt. Dieses muß unter Benutzung des Stabes geöffnet werden.
- Danach wird der Löwe sichtbar, der mit dem Brot besänftigt werden kann.
- Ist dieses vollbracht, kann der Ring zur Kapelle getragen werden, wo er "aktiviert" wird.
- Danach ist es nur noch ein Katzensprung in das Hauptgebäude, wo das Schloß erlöst wird.
- Der Ring zeigt die Koordinaten an, es erscheint eine hübsche Prinzessin und alles wird gut.
- Und wenn sie nicht gestorben sind... ☺

Dies ist das Drehbuch, nachdem wir den Wherigo gestalten möchten.

Wer sich das Ganze als Luftbild ansehen möchte, rufe via Google Maps die Koordinaten 50.027345,9.796844 auf.

Aber zuvor ist leider etwas Theorie notwendig...

Objektorientierung - ein bisschen Theorie muß sein

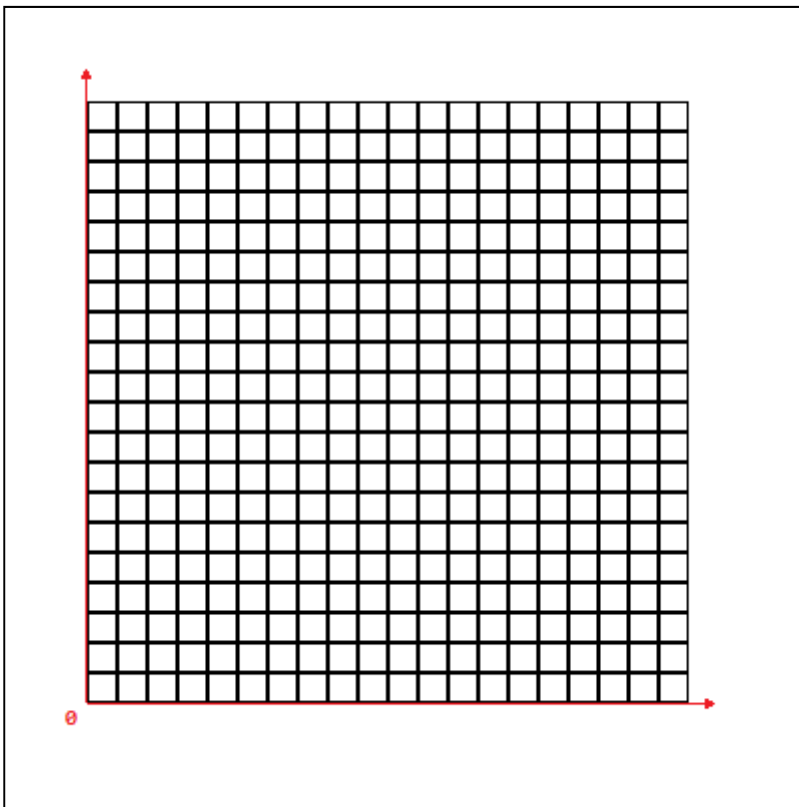
Keine Angst es wird nicht zu kompliziert. Aber ohne geht es leider nicht. Daher, bevor es losgeht, zum besseren Verständnis der Funktionsweise des Editors ist ein wenig programmiertechnische Theorie. Wer schon eine objektorientiert Programmiersprache beherrscht kann diesen Teil getrost überspringen.

Objektorientierung ist ein Konzept, das hauptsächlich in der Programmierung verwendet wird. Es ist ein Ansatz, um komplexe Sachverhalte zu vereinfachen bzw. um diese einfach abbilden zu können. Man könnte nun ein komplettes Tutorial für einen Einstieg in die Objektorientierung verwenden.

Wir wollen allerdings nur ein wenig an der Oberfläche kratzen. 😊

Dazu ein etwas abstraktes Beispiel, damit alles etwas plastischer wird:

Stellen wir uns vor, wir wollten quadratische Gebiete auf einer Karte verwalten. Der Einfachheit halber sei unsere Karte ein einfaches Koordinatensystem, wie es jeder aus dem Schulunterricht kennt:

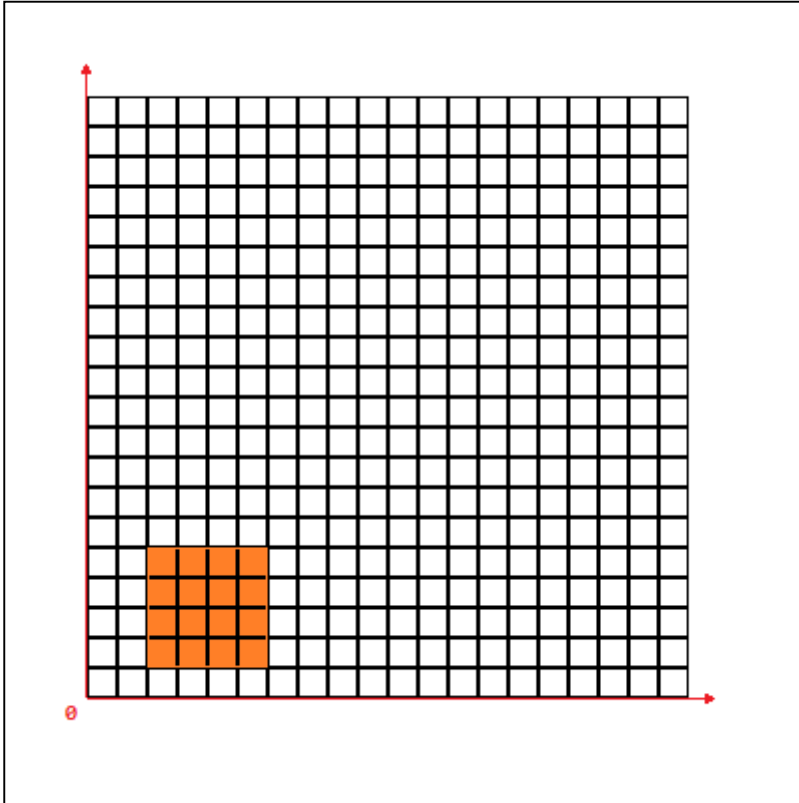


Koordinatensystem

Links unten ist der Nullpunkt und die Knotenpunkte werden über die x-Achse (waagrecht) und die y-Achse (senkrecht) definiert. Ein Kästchen ist eine Einheit. Soweit nichts Neues. Kennt man (hoffentlich) noch aus der Schule 😊

Nehmen wir weiterhin an, wir möchten die Gebiete, die wir verwalten sollen wie folgt festlegen:

- Zum einen geben wir die linke untere Ecke in Form von Koordinaten (x und y) an und
- zusätzlich noch die Breite und
- die Länge bzw. Höhe des Gebietes.



Koordinatensystem mit markiertem Gebiet

Gemäß dieser Definition hätte das oben gezeigte orangene Gebiet folgende Werte:

- linke untere Ecke = 2,1 (oder $x=2$ und $y=1$)
- Breite = 4
- Höhe = 4

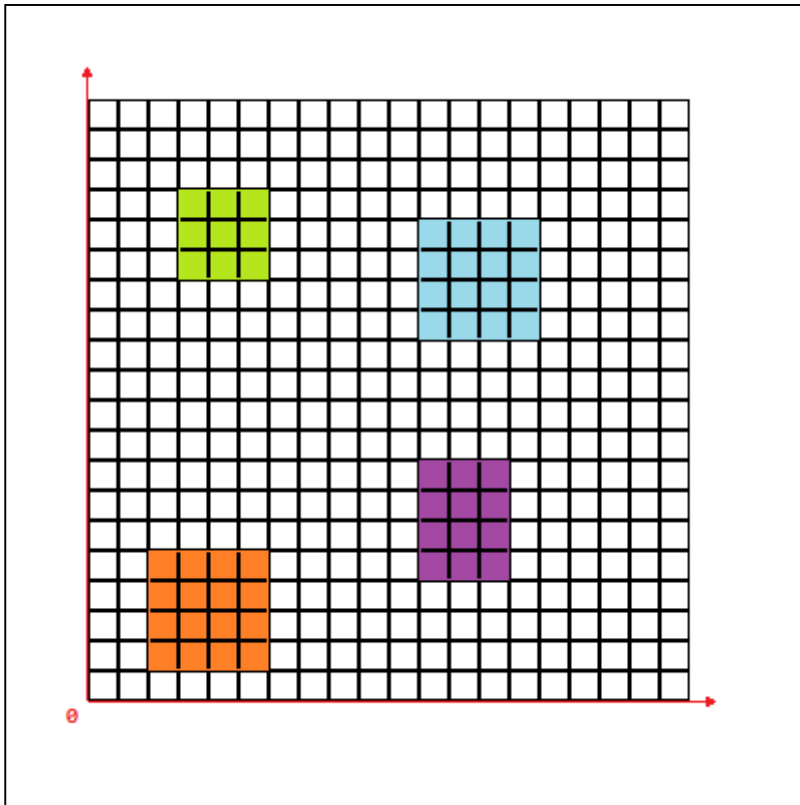
Damit wäre das oben gezeigte Gebiet eindeutig beschrieben und soweit könnten wir das auch in einem Computer so abspeichern. Wir würden 4 Variablen (bzw. Platzhalter) definieren und denen die entsprechenden Werte zuweisen. Fertig.

Wenn wir jetzt noch berechnen wollten, wie groß dieses Gebiet ist, wäre auch das verhältnismäßig einfach. In unserem Fall wäre das 4×4 (Länge \times Breite) also 16 Einheiten. Wenn wir es einem Computer beibringen wollten, würden wir ihn eben anweisen, die Variablen/Platzhalter für die Länge und die Breite zu multiplizieren und den Wert auszugeben.

Das wäre auch noch kein Hexenwerk. Unser Programm würde aus 4 Variablen/Platzhaltern für die benötigten Werte des Rechtecks bestehen und wir bräuchten noch die Möglichkeit 2 bestimmte Werte miteinander zu multiplizieren.

Solange wir nur 1 Gebiet verwalten, wäre die Sache auch noch ganz einfach.

Was aber, wenn unsere „Landkarte“ so aussieht:



Koordinatensystem mit mehreren „Ländern“

Dann wird die Sache doch gleich etwas komplexer, da etliche Werte irgendwie verwaltet werden wollen. Nach der oben beschriebenen Vorgehensweise müssen wir etliche Variablen/Platzhalter definieren, eindeutig benennen und unterscheiden. Das könnte schön etwas „frickelig“ werden.

Und hier kommt jetzt die Objektorientierung ins Spiel:

Damit wir bei all den verschiedenen Werte nicht durcheinander kommen, wollen wir die verschiedenen Gebiete als eigenständige Objekte ansehen. Dazu brauchen wir eine Art „Kochrezept“, mit dem wir diese Objekte erstellen, damit sie alle schön einheitlich sind.

Zunächst einmal haben unsere Gebiete alle die gleichen Eigenschaften:

- eine x-Koordinate für die linke untere Ecke. Die nennen wir x (wer hätte es gedacht 😊)
- eine y-Koordinate für die linke untere Ecke. Die nennen wir y. (dito)
- eine feste Breite, die wir mit dem Namen b versehen
- eine feste Höhe, für die wir den Namen h vergeben.

Da alles einen Namen braucht wollen wir außerdem das Kochrezept mit dem Namen „Land“ benennen. Die oben genannten Platzhalter x, y, b und h sind daher Eigenschaften oder Attribute unseres Objekts namens Land!

Und dann fehlt uns nur noch eine Funktion (auch Methode genannt), die aus den Eigenschaften des jeweiligen Objektes, die Fläche errechnet. Nämlich durch:

Länge x Breite – diese Methode nennen wir berechneFläche (schreibt man üblicherweise ohne Leerzeichen so). Dieser Funktion bringen wir bei, die Platzhalter für Höhe und Breite eines Objektes miteinander zu multiplizieren und auszugeben.

Mit diesem Kochrezept, welches die Eigenschaften unseres Objektes kapselt, könnten wir nun verschiedene

Ausprägungen herstellen. Diese neuen Objekte wären dann vom Typ "Land" (so wie eben unser Kochrezept heißt).

Auf die Eigenschaften der Objekte würden wir mit der sog. Punktschreibweise zugreifen, damit eindeutig ist, auf welchen Wert zugegriffen werden soll. Dies sieht so aus
<Objektname>.<Objekteigenschaft>

Wenn wir also nach dem „Kochrezept“, das wir Land genannt haben, ein Objekt erstellen, vergeben wir einen eigenständigen Namen, damit klar ist, welches gemeint ist.

Nennen wir z. B. das orangene Gebiet sinnigerweise ORANGE, dann sähe das nach Punktschreibweise so aus:

```
ORANGE.x = 2
ORANGE.y = 1
ORANGE.b = 4
ORANGE.h = 4
ORANGE.berechneFläche liefert uns den Wert 16
Der Typ (bzw. das "Kochrezept") von ORANGE wäre "Land"
```

Beispielhafte Aussage: ORANGE ist vom Typ „Land“ und der x-Wert von ORANGE ist 2.
Natürlich könnten wir auch beliebige andere Namen vergeben: Hugo, Land1, etc.

Gleiches gilt für die anderen Gebiete, die wir verwalten wollen. Zum Beispiel das violette Gebiet:

```
VIOLETT.x = 11
VIOLETT.y = 4
VIOLETT.b = 3
VIOLETT.h = 4
VIOLETT.berechnFläche ergäbe 12.
```

Auch hier der Typ wäre "Land".

Und so weiter und so fort. Über ein definiertes Objekt/Kochrezept namens Land, könnte ich also verschiedenste Ausprägungen ein und derselben Sache verwalten und einheitlich behandeln, da sie über dieses Kochrezept eine einheitliche Struktur erhalten. Im obigen Beispiel müßte ich jetzt nur noch 4 Objekte vom Typ Land irgendwie speichern und nicht 16 einzelne Werte. Durch die Objekte wird also alles schön übersichtlich.

Ein weiterer Vorteil: Wollte ich einen Computer für alle Objekte vom Typ Land die Fläche ausrechnen lassen, müßte ich ihn nur anweisen, für jedes Objekt die Funktion berechneFläche aufzurufen, da diese für alle gleich definiert und vorhanden ist. Ich müßte also auch diese Funktion nur einmal programmieren, um sie für alle Objekte automatisch zur Verfügung zu haben. Das ist doch sehr komfortabel. 😊

Ja und das ist es dann eigentlich auch schon. Objekte sind also mehr oder weniger abstrakte Konstrukte, die aus Eigenschaften/Attributen oder Funktionen/Methoden bestehen und eine Beschreibung für gleichartige „Dinge“ bereitstellen.

Tiefer wollen wir erst mal nicht graben. Etwas plastischer wird es hoffentlich im weiteren Verlauf.

Urwigo - der Editor

Zum Erstellen von Wherigos gibt es einige Editoren zum freien Download. In diesem Beispiel soll der Urwigo-Editor verwendet werden. Er kann [hier](#) herunter geladen werden.

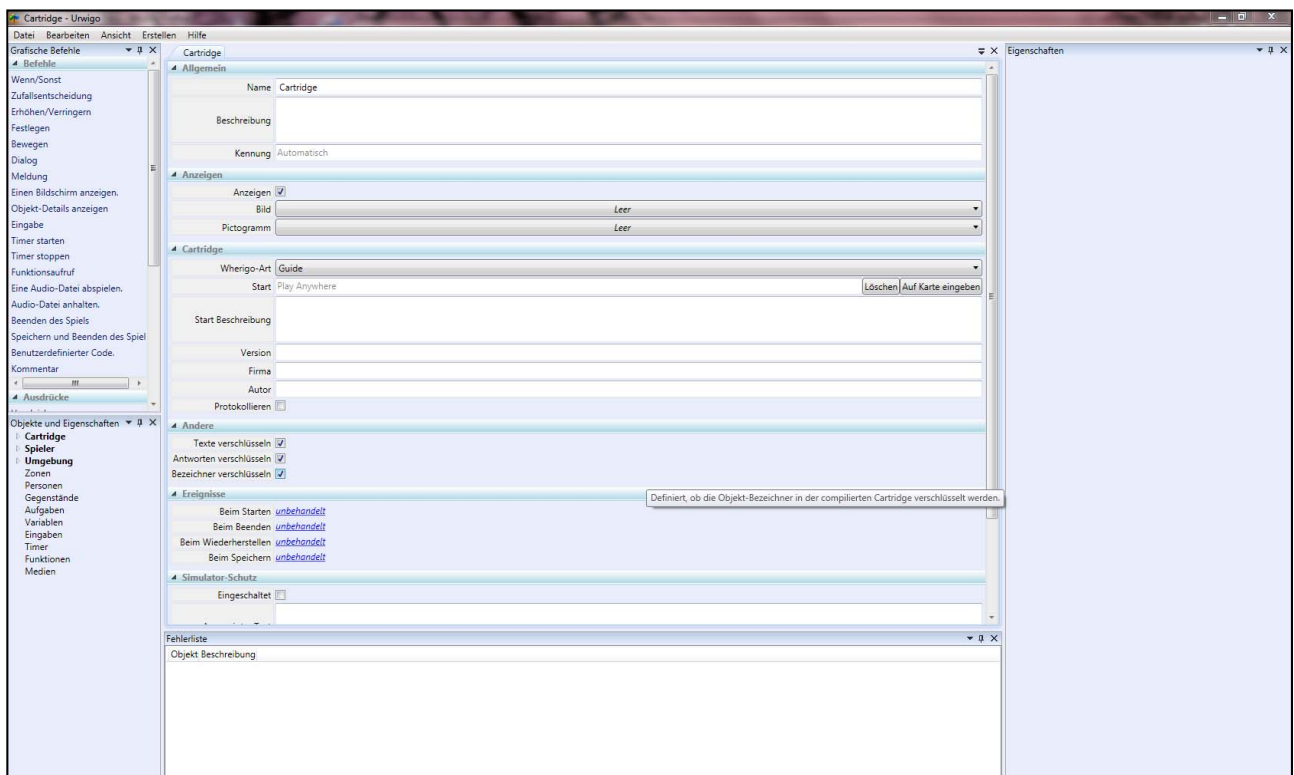
Urwigo ist sehr übersichtlich aufgebaut und wenn man sich mit dem Konzept der Objektorientierung (da war's gleich schon wieder) beschäftigt hat und ein wenig Anleitung erhalten hat, ist man durchaus in der Lage, sich den Rest selbst zu erarbeiten.

Da die Programmierung über grafische Elemente erfolgt, muß man sich auch nicht mit Syntax-Regeln und ähnlichem herumschlagen. ☺

Natürlich muß man sich ein wenig damit beschäftigen. Der Grundsatz "Programmieren lernt man durch programmieren" gilt auch hier. Es ist nicht so aufwändig, wie das Erlernen einer echten Programmiersprache (z. B.: C++ oder Java), aber auch hier ist etwas an Einarbeitung notwendig. Über sog. LUA-Skripte (nicht Teil des Tutorials) kann die Flexibilität und die Möglichkeiten des Editors noch erheblich erweitert werden.

Einen neuen Wherigo beginnen

Sobald Urwigo installiert wurde, starten wir diesen. Zunächst sehen wir den folgenden Bildschirm:



Das Hauptfenster

Nachtrag: Während der Arbeit an diesem Tutorial wurde eine neue Version des Editors frei gegeben. Die wesentlichen Merkmale blieben aber unverändert. Es gibt ein paar neue Features, die aber für dies Tutorial nicht relevant sind. Die Bildschirmsichten können daher geringfügig abweichen.

Das **Hauptfenster** in der Mitte mit dem Reiter "Cartridge" enthält Eingabemöglichkeiten für die grundlegende Definition unseres Wherigos. Damit werden wir uns gleich beschäftigen.

Das Fenster „**Befehle**“ links oben enthält grafische Befehle, wie z. B. "wenn/sonst" etc. Diese werden wir später benötigen.

Links unten ist die Objektübersicht. Hier werden die verschiedenen Objekttypen verwaltet. Wie wir schon jetzt sehen, gibt es Objekte für Zonen, Personen, Gegenstände, etc. Dies sind die „Kochrezepte“ nach denen die einzelnen Bausteine eines Wherigos erstellt werden.

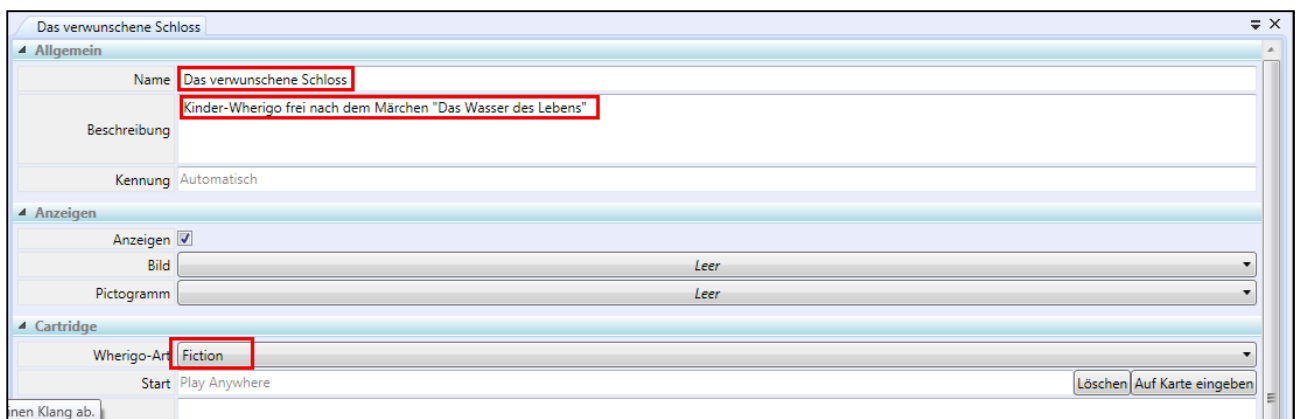
Ganz rechts gibt es noch ein Eigenschaften-Fenster. Auch dieses wird erst nachher interessant. In diesem Fenster kann man Eigenschaften und Funktionen des aktuell bearbeiteten Objektes anzeigen und anpassen.

In der Mitte unten gibt es noch ein Fenster für Fehler-Informationen. Dieses werden wir hoffentlich nur selten benötigen. Beim Erstellen von Programmteilen blitzt hier ab und an mal ein Fehler auf. Aber wenn alles fertig ist, sollte dort nichts mehr stehen.

Das Hauptfenster

Wenden wir uns zunächst dem Cartridge-Fenster zu und versorgen wir dieses mit Werten.

Wir geben den Titel und die Kurzbeschreibung ein und stellen für die Art des Wherigos "Fiction" ein.



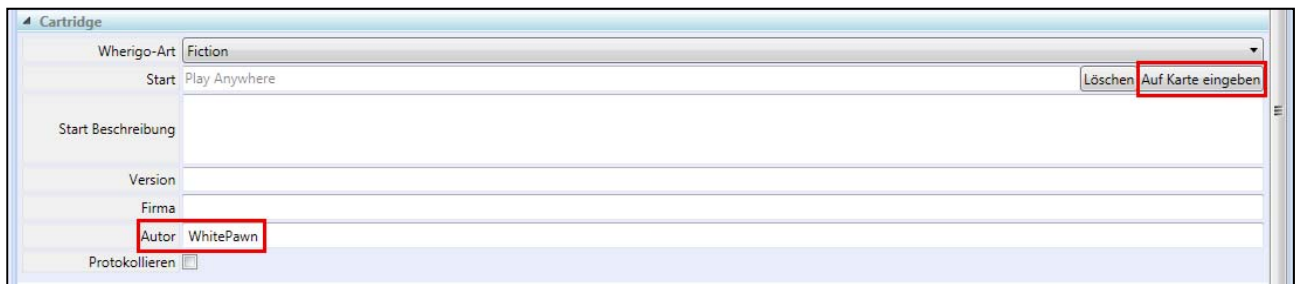
Grundsätzliche Einstellungen

Wichtig: Urwigo kann keine Umlaute darstellen! Daher sollten wir schon jetzt darauf achten, diese nicht zu verwenden.

Weiter unten gibt es noch die Option "Simulator-Schutz":

Diese Option sollten wir setzen, bevor wir den Wherigo frei geben. Damit soll verhindert werden, daß jemand den Wherigo am PC durchspielt. Allerdings kann dieser Schutz relativ leicht umgangen werden. Vorerst werden wir diese Checkbox aber frei lassen.

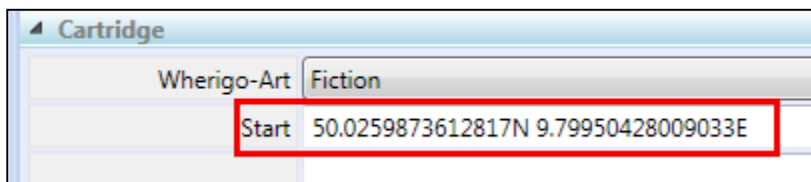
Auf jeden Fall wollen wir noch unsere Autorenschaft im Feld „Autor“ kund tun und da der Wherigo an einer fixen Position stattfinden soll, diese definieren.



Startpunkt und Autor

Um die Position festzulegen klicken wir zunächst auf den Button “Auf Karte eingeben” und wählen diese über Google Maps aus.

Danach wird im Feld “Start” die Startposition angezeigt. (Hier: 50.0259873612817N 9.79950428009033E).



Start-Koordinaten

Auch hier gilt das, was wir aus dem GeoCaching kennen: Diese Art der Positionsbestimmung kann durchaus ungenau sein. Niemand würde auf diese Art die Koordinaten seines Caches festlegen! Daher macht es durchaus Sinn einen Wherigo auch vor Ort einzumessen oder zumindest gut zu testen und vor Ort die Genauigkeit zu prüfen.

Wichtig: Speichern nicht vergessen. Auch Urwigo spinnt ab und zu einmal und nichts ist ärgerlicher, als wenn ein Programm abstürzt und man hat länger nicht gespeichert. Daher sollte man lieber einmal zu viel speichern, als zu einmal zu wenig.

Bilder laden

Als Nächstes sollten wir uns um grafische Details kümmern, damit diese zur Verfügung stehen.

Wir benötigen Bilder für:

- den Startbildschirm
- den Ritter
- den Ring
- den Zwerg
- das Brot
- den Stab
- das Tor
- den Löwen
- den Altar
- die Prinzessin

Eine wichtige Frage ist hierbei, **wie groß die Bilder** sein sollen.

Das ist eine Frage, die sich relativ schwer beantworten läßt, da die Displaygröße der Geräte sehr variiert. Das Garmin Oregon hat eine Displaygröße von 240 x 400 Pixeln. Im Vergleich dazu hat ein Samsung Galaxy S2 eine Größe von 480 x 800 Pixeln. Andere Geräte haben wieder andere Abmessungen.

Wie soll man dem gerecht werden?

Es gibt dabei mehrere unterschiedliche Ansätze:

- Manche Autoren stellen ihren Wherigo 2x mit unterschiedlichen Bildgrößen zur Verfügung: angepaßt für GPS-Geräte und angepaßt für PDAs und Smartphones.
- Andere treiben programmiertechnischen Aufwand, so daß die Bildgröße an das Gerät angepaßt werden kann. Dabei muß die Cartridge aber jedes Bild mehrfach in unterschiedlicher Größe beinhalten.
- Der einfachste Ansatz ist, die Bilder nur so groß zu machen, daß sie auch auf dem kleinsten Display angezeigt werden können, auch wenn sie dann auf anderen Geräten sehr klein erscheinen oder vergrößert werden

Diesen letzten Ansatz wollen wir hier im Tutorial verfolgen. Er spart uns einiges an Aufwand und ist für Programmier-Einsteiger die beste Lösung.

Googelt man nach der idealen Größe für ein Bild, findet man viele verschiedene Angaben.

Wir wollen uns für unser Tutorial auf eine Bildgröße mit einer max. Breite von 230 Pixeln beschränken. Damit bleibt man etwas unter der max. Breite der Garmin GPS (240 Pixel) und hat so noch etwas "Luft".

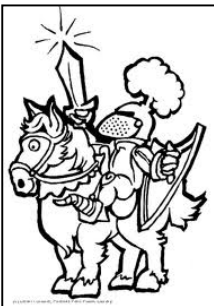
Damit einige Objekte im Inventar und der Übersicht bildhafter angezeigt werden, stellen wir für diese noch ein Bild als Icon zur Verfügung. Meiner Ansicht nach gilt: Je mehr Bilder desto besser – vor allem wenn der Wherigo für Kinder angeboten werden soll.

Die Bildbreite für die Icons sollte max. 32 Pixel betragen.

Dieses Icons erzeugen wir für alle Bilder, außer dem Ritter und dem Startbild.

Um auf dem Niveau eines Märchens zu bleiben, habe ich mich für Zeichnungen entschieden. Die entsprechenden Grafiken wurden über die Google Bildersuche gefunden und mit einem Grafikprogramm auf das richtige Format geändert. ☺

Hier z. B. der Ritter:

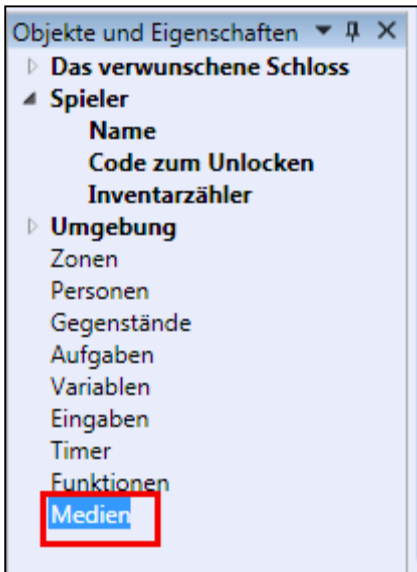


Beispielbild

Die im Tutorial verwendeten Grafiken könne aus der ZIP-Datei mit dem Projekt heraus kopiert werden.

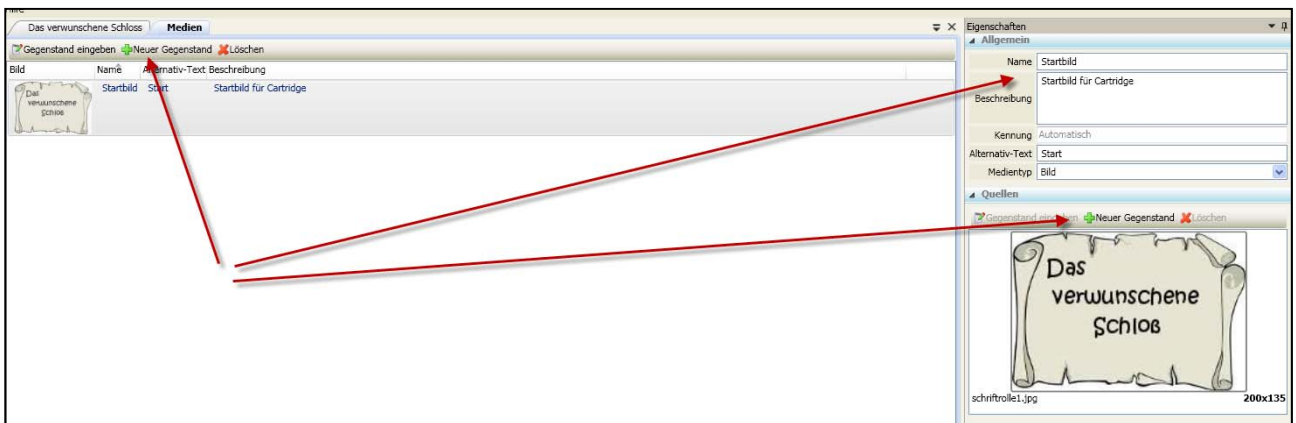
Nachdem alle Bilder zur Verfügung stehen, sollen diese in den Urwigo-Editor geladen werden:

Nach einem Doppelklick auf "Medien" im Fenster „Objekte und Eigenschaften“ öffnet sich in der Mitte ein neues Fenster:



Medien-Objekte auswählen

Hier können wir jetzt die Bilder in den Urwigo-Editor laden.

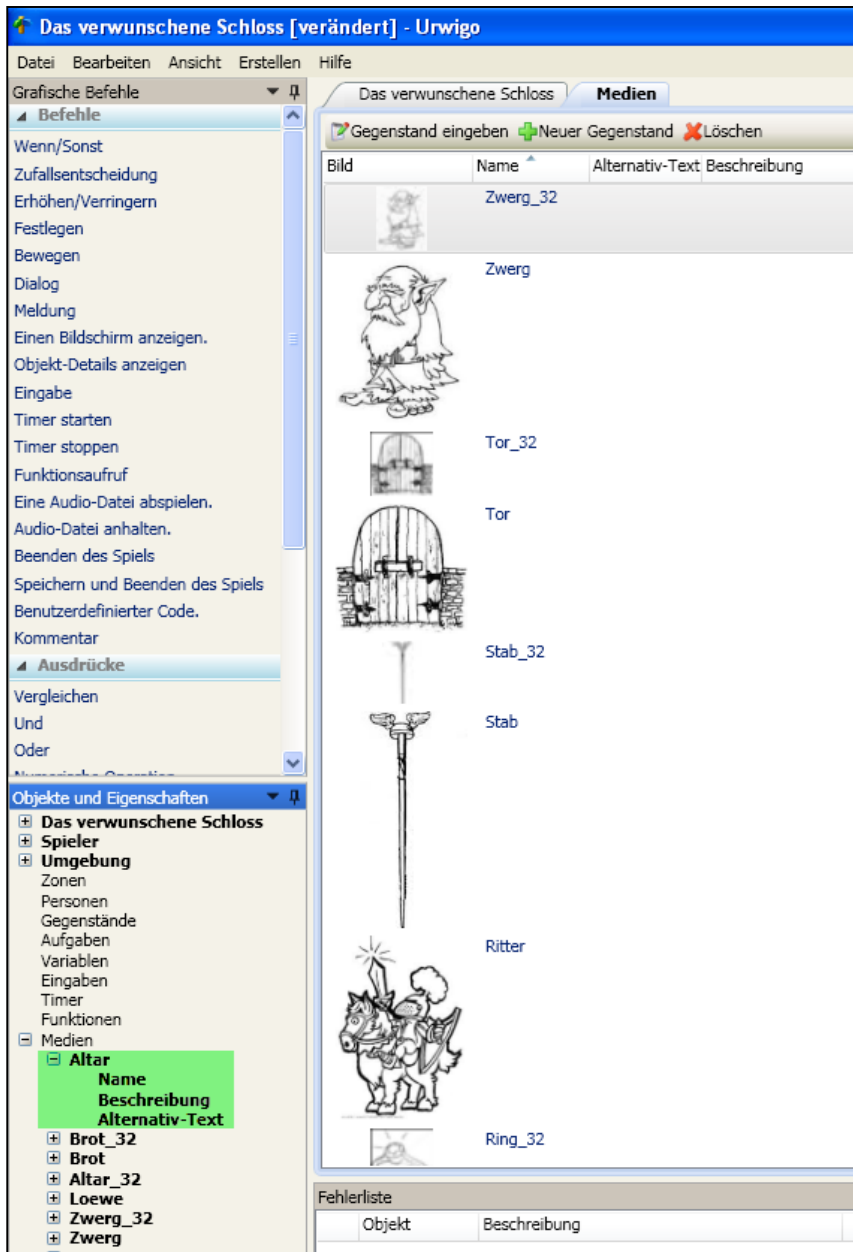


Ein erstes Medien-Objekt mit einem Bild

Wir klicken auf "Neuer Gegenstand", vergeben im Eigenschafts-Fenster rechts einen neuen Namen und suchen ebenfalls im Eigenschaftsfenster über "Neuer Gegenstand" das entsprechende Bild aus. Das Popup das nach einer Bestätigung fragt, ob das Bild in das Urwigo-Verzeichnis kopiert werden soll, bestätigen wir mit "Ja".

An dieser Stelle sei angemerkt, daß Urwigo meiner Ansicht nach einen Übersetzungsfehler enthält und "Gegenstand" wohl besser mit "Objekt" übersetzt worden wäre.

Auf die dargestellte Art und Weise laden wir nach und nach alle Bilder in den Urwigo-Editor:



Alle Bilder als Objekt vom Typ Medien

Ungefähr so, wie auf dem oberen Bild sollte das dann aussehen.

Links unten im Fenster "Objekte und Eigenschaften" kann man dann die Medien-Objekte sehen (das Objekt Altar und dessen Attribute wurde grün markiert).

Etwas eingerückt werden jeweils darunter die Eigenschaften und/oder Funktionen angezeigt, die diese Objekte jeweils haben.

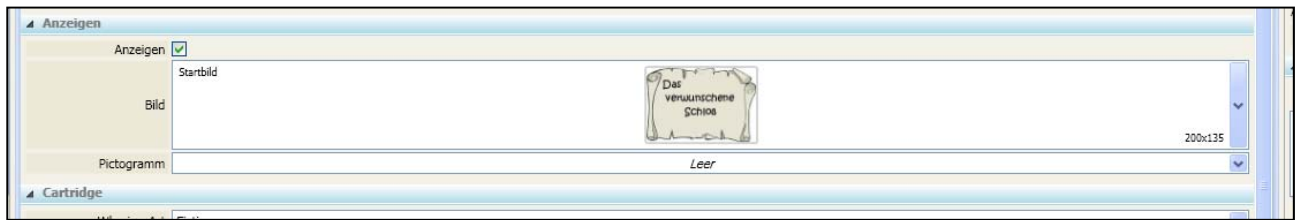
Hier sei nochmal an das theoretische Kapitel erinnert, in dem von Objekten und deren Eigenschaften gesprochen wurde. Das ist im Editor auch so umgesetzt! Die Objekte im obigen Beispiel sind vom Typ Medien und alle haben Eigenschaften wie Name, Beschreibung, etc. auf die zugegriffen werden kann, analog zu unserem Beispiel mit x, y, usw.

Nachdem die Bilder im Editor zur Verfügung stehen, können wir dem Hauptfenster vorne ein Startbild spendieren.

Aber vorher: Projekt speichern, damit nichts verloren geht!

Das erste Bild einbinden

Das geht ganz schnell:



Eingebundenes Bild im Hauptfenster der Cartridge

Im Hauptfenster mit dem Titel unseres Wherigos als Reiter, klicken wir auf das Pull Down-Menü mit der Aufschrift Bild und wählen das Startbild aus.

Fertig. Hat gar nicht weh getan. ☺

Im Folgenden werden wir die Bilder noch öfters benötigen. Jetzt aber erst mal etwas anderes.

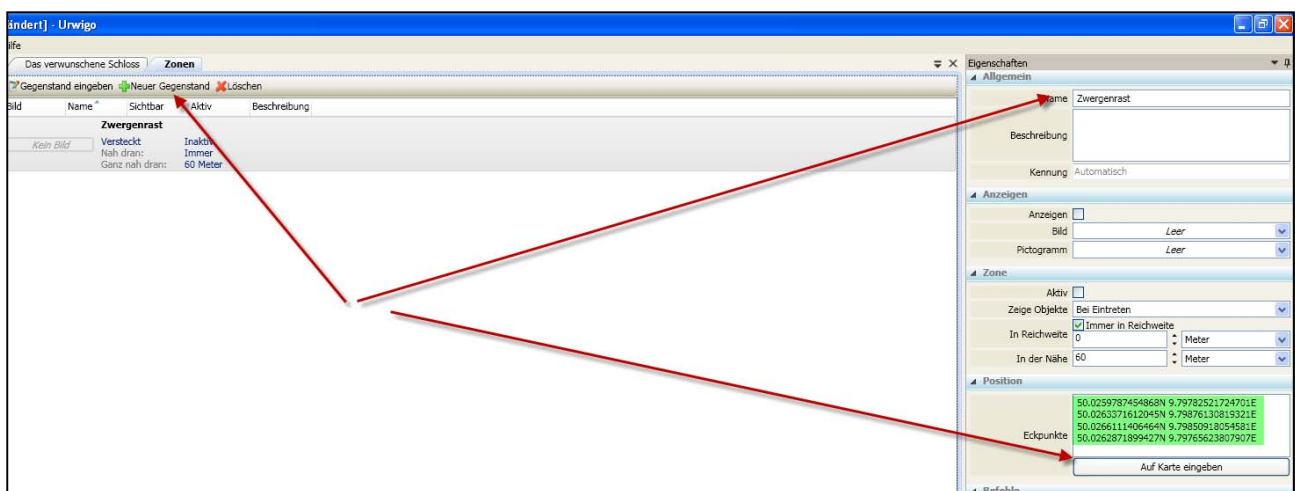
Zonen anlegen

Jetzt wollen wir uns den unterschiedlichen Zonen zuwenden. Die Zonen sind für einen Wherigo, was die Stages für den Multi-Cache sind: Das sind die Orte „wo die Action abgeht“ ☺

Über Zonen kann mit Gegenständen und Personen interagiert werden. So kann beispielsweise ein Gegenstand in einer Zone abgelegt werden und ist dann für den Spieler sicht- und nutzbar, wenn dieser die entsprechende Zone betritt. Gleiches gilt für virtuelle Personen.

Zum Anlegen klicken wir im Fenster "Objekte und Eigenschaften" doppelt auf "Zonen", so wie wir vorhin das Fenster für die Medien geöffnet haben.

Es erscheint in der Mitte ein Fenster ganz ähnlich dem, das wir eben für die Bilder bzw. Medien benutzt haben.



Die erste Zone

Hier wollen wir die erste Zone/Stage anlegen.

- Wir klicken auf "Neuer Gegenstand" (eigentlich "neues Objekt vom Typ Zone")
- Im Eigenschaften-Fenster rechts benennen wir die Zone in "Zwergenrast" um
- Im Unterpunkt "Zone" soll das Kästchen für "Aktiv" leer bleiben - dazu später mehr
- Die Koordinaten (grün markiert) ermitteln wir aus der Karte, indem wir auf den Knopf "Auf Karte eingeben" klicken



Anlegen der ersten Zone

Durch klicken in die Karte definieren wir die erste Zone. Nachdem eine Koordinaten-Ermittlung aus der Karte heraus nicht sehr genau ist, sind wir hier einfach mal großzügig und machen diese Zone sehr groß (grob geschätzt $> 200 \text{ m}^2$).

Damit haben wir alle Wege zur Burg abgedeckt. Diese Zone ist (wohlwollend betrachtet) rechteckig, es ist aber vom Dreieck bis zum Vieleck jede Form denkbar. Auch hier gilt: Gegebenenfalls die Zonen vor Ort ausmessen oder gut testen.

Wichtig: Die Zonen sollten nicht zu klein sein. Andernfalls kann es passieren, daß der Spieler bei schlechtem GPS-Empfang die Zone wieder verläßt, ohne sich bewegt zu haben. Dies muß man auch bei der späteren Programmierung berücksichtigen, sonst kann es zu ungewollten und für die Spieler unschönen Ereignissen kommen. Im o. a. Fall könnte man die Zone sogar noch größer festlegen um „Querfeldein-Läufer“ abzufangen.

Am Ende des Tutorial wird noch eine andere Möglichkeit gezeigt, die verhindert, daß der Spieler unbeabsichtigt die Zone verläßt.

Ideal wäre eine Möglichkeit, den Spieler mitten in die Zone zu locken. Evtl. durch eine reale Stage oder wenn etwas abgelesen werden muß, etc..

Auf diese Art und Weise legen wir insgesamt 4 Zonen an:



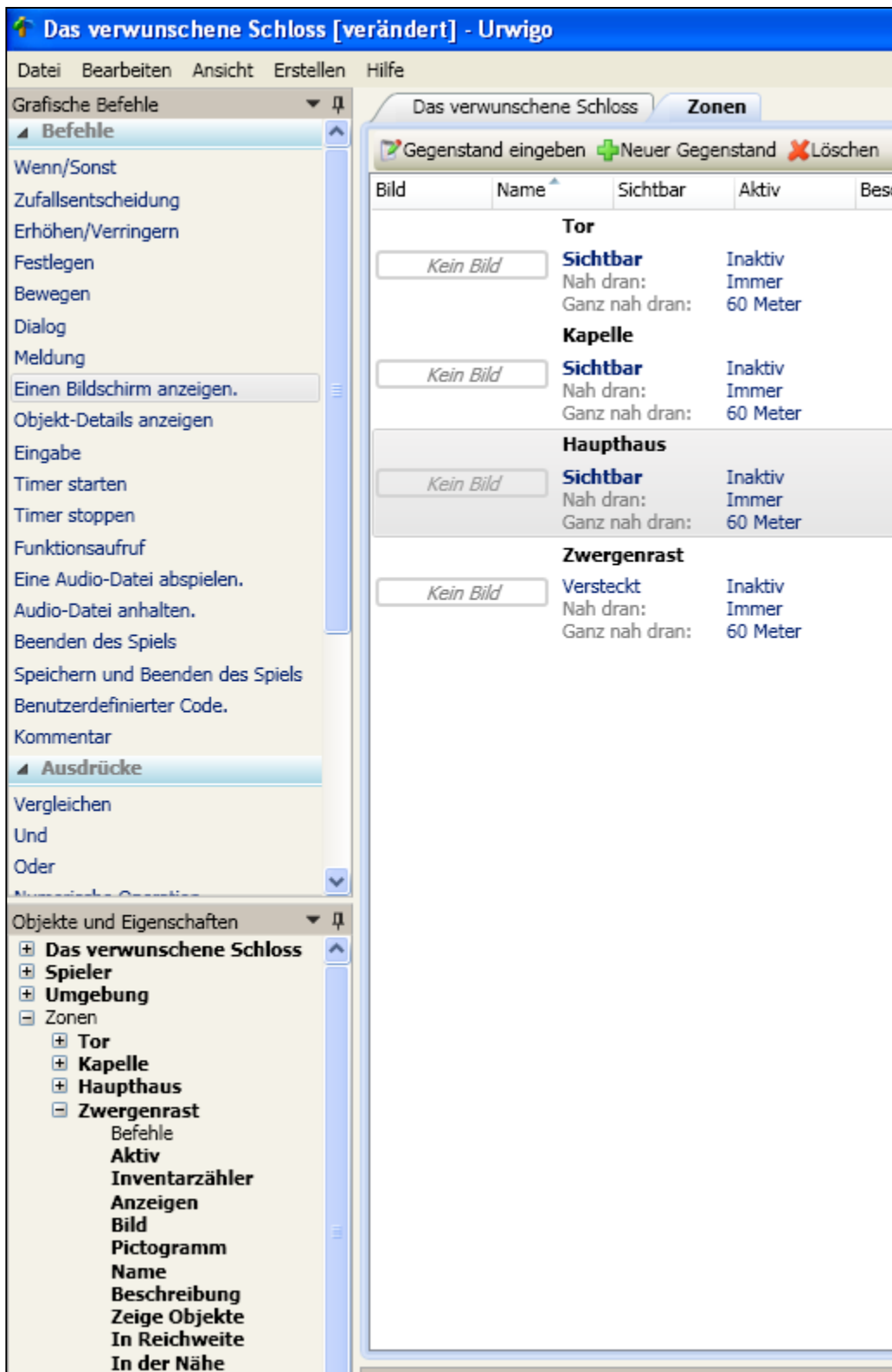
Alle Zonen des Tutorials

Ich habe diese Zonen wie folgt benannt:

1. Zwergenrast
2. Tor
3. Kapelle
4. Haupthaus

Ich habe diese hier relativ groß gewählt, damit bei ungenauem GPS-Empfang die Zone trotzdem ohne größere Probleme angelaufen werden kann.

Auf dies Art und Weise werden die 4 benötigten Zonen angelegt:



Alle Zonen in der Übersicht

Auch für die Zonen kann man im Fenster "Objekte und Eigenschaften" für die Zonen-Objekte wieder alle Eigenschaften und Funktionen anzeigen. Auch hier wurde in einem Zonen-Objekt alles Notwendige gekapselt, so daß wir für alle Zonen-Objekte einheitliche Standard-Eigenschaften und -Funktionen haben. Oben wurden die Attribute für die Zone „Zwergenrast“ beispielhaft „ausgeklappt“.

Und jetzt: Speichern nicht vergessen!

Wie geht's weiter?

Wir haben jetzt ein gewisses Grundgerüst geschaffen und wollen endlich mal los legen. Erfahrene Programmierer würden jetzt im Rahmen einer strukturierten Herangehensweise erst noch Personen, Gegenstände und andere Objekte programmieren.

Ich denke aber, daß es für Anfänger einerseits interessanter ist, wenn man immer wieder mal ein Zwischen-Ergebnis sieht und es andererseits weniger abstrakt ist, wenn man die verschiedenen Objekte dann programmiert, wenn man sie benötigt und so auch das Verständnis für die Funktionsweise eines Wherigos besser vermittelt wird.

Auch wenn das etwas "Kuddelmuddel"-Programmierung ist. ☺

Erfahrene Programmierer mögen mir verzeihen...

Zunächst wollen wir den Start der Cartridge ausbauen und in die Geschichte einführen!

Dazu müssen wir das Ereignis "Beim Starten" mit Leben füllen:

Das verwunschene Schloss Das verwunschene Schloss.Beim Star

Kennung: Automatisch

Anzeigen

Anzeigen: ☒

Bild: Startbild

Pictogramm:

Cartridge

Wherigo-Art: Fiction

Start: 50.0259873612817N 9.79950428009033E

Start Beschreibung:

Version:

Firma:

Autor: WhitePawn

Protokollieren: ☐

Andere

Texte verschlüsseln: ☒

Antworten verschlüsseln: ☒

Bezeichner verschlüsseln: ☐

Ereignisse

Beim Starten: [unbehandelt](#)

Beim Beenden: [unbehandelt](#)

Beim Wiederherstellen: [unbehandelt](#)

Beim Speichern: [unbehandelt](#)

Simulator-Schutz

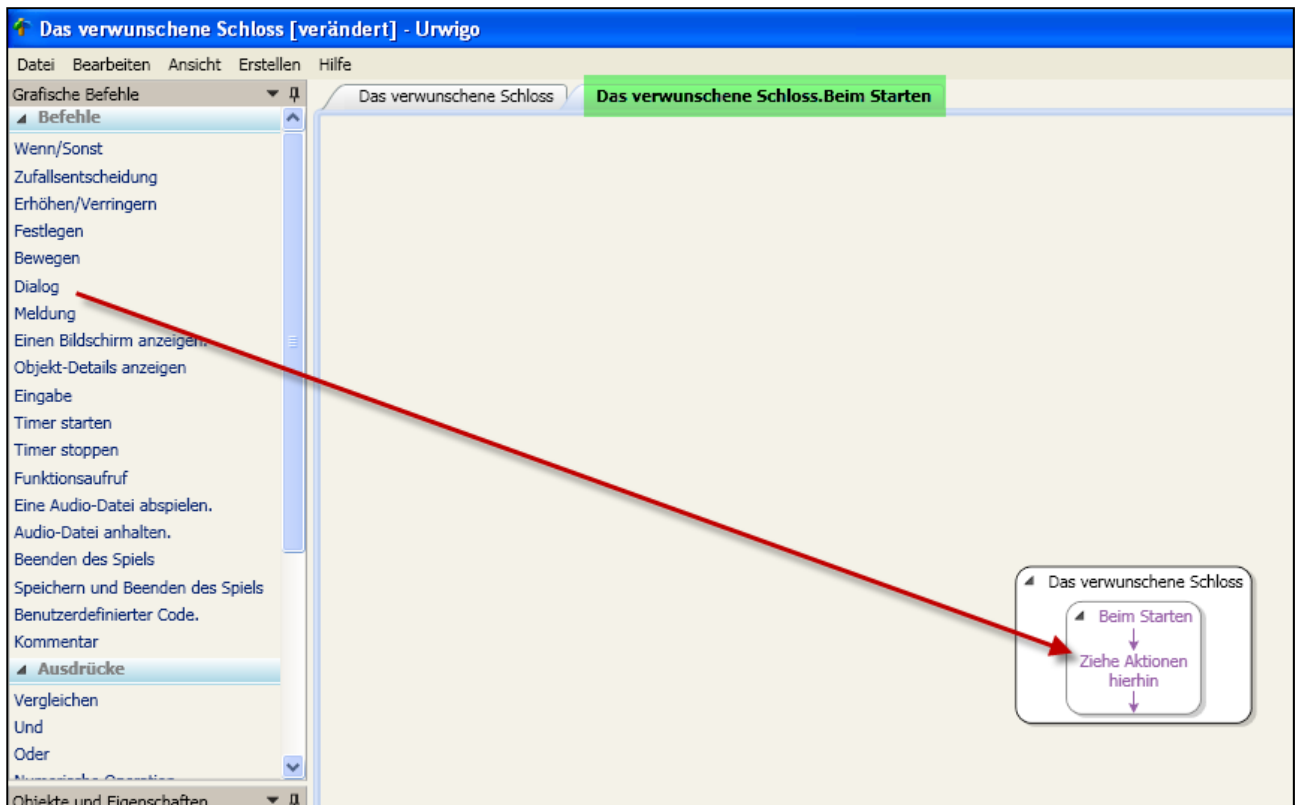
Funktion „Beim Starten“

Wie der Name schon nahelegt, wird diese Funktion (bzw. das Ereignis) beim Starten der Cartridge aufgerufen. Ähnliche Aufrufe gibt es z. B. auch beim Betreten und Verlassen von Zonen, etc. Dazu später mehr.

An dieser Stelle kann man ein paar Start-Informationen hinterlegen oder auch irgendwelche vorbereitenden Einstellungen vornehmen.

Wir wollen hier mit der Geschichte anfangen. Dazu klicken wir auf den "Link" "unbehandelt".

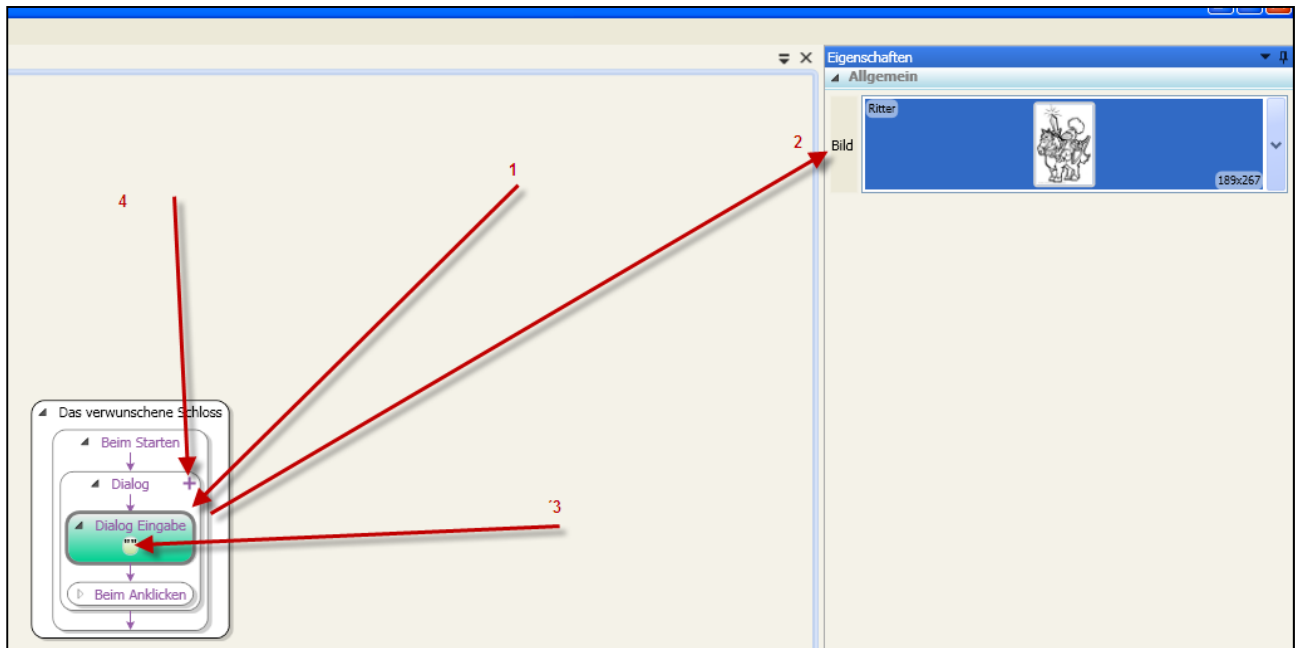
Wir erhalten ein neues Fenster. Wie wir an dem Text im Reiter (grün markiert) sehen können. Handelt es sich um die Funktion bzw. das Ereignis "Beim Starten" des Objektes "Das verwunschene Schloss". Zu erkennen ist das an der Punkt-Notation: <Objekt>.<Funktionsname> (grün markiert)



Ausprägung „Beim Starten“

Hier wollen wir jetzt ein bisschen Text zur Einführung in die Geschichte los werden und ziehen aus dem Fenster mit der Überschrift "grafische Befehle" einen "Dialog" auf das Feld "Ziehe Aktionen hierhin".

So ähnlich sieht es dann aus:



Einfügen des Dialog-Befehls

Folgende Aktionen wurden im oberen Bild schon ausgeführt:

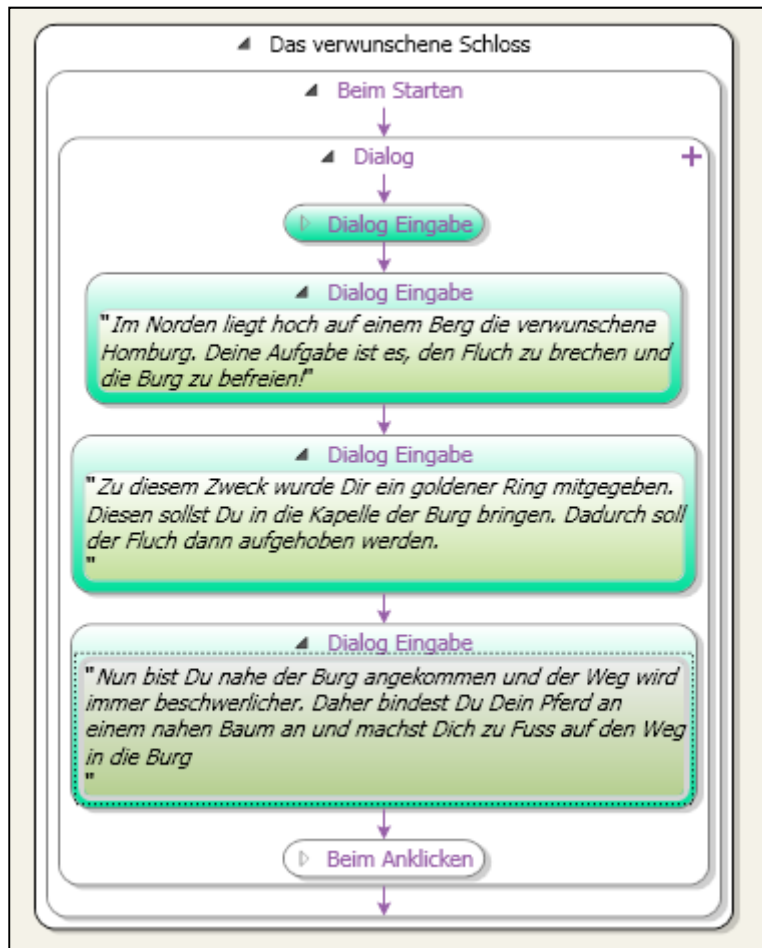
1. Das Dialogfeld wurde aus dem Befehlsvorrat herüber gezogen
2. Wir klicken auf das Dialogfeld, so daß es den dunklen fetten Rahmen erhält
3. Danach wechselt das Eigenschaftsfenster auf der rechten Seite, so daß wir das Bild des Ritters zuweisen können
4. Wir klicken auf die Gänsefüßchen und können dann im Eigenschaftsfenster rechts den Text eingeben. (Den Text dem Beispiel entnehmen oder selbst etwas schreiben ☺)
5. Danach erzeugen wir über das Plus-Zeichen im Dialog-Objekt weitere Dialog-Eingaben (diese grünen "Blasen") (vgl. nächstes Bild).

Zur Erinnerung: Urwigo kann keine Umlaute darstellen.

Ein eingegebenes „ö“ wird nicht als solches im Player angezeigt. Daher gilt: ö = oe, usw.

Es gibt durchaus Möglichkeiten Umlaute darzustellen. Im Rahmen des Tutorials soll aber darauf verzichtet werden, um das Tutorial einfach zu halten. Wen es interessiert, der findet mit den Begriffen "Urwigo" und "Umlaute" über jede Suchmaschine geeignete Hinweise.

Fertig ausgeprägt, sieht das Ganze so aus (die erste Dialogblase ist im Beispiel „zugeklappt“, sieht aber prinzipiell ähnlich aus):



Der fertige Dialog (erste „Blase“ eingeklappt)

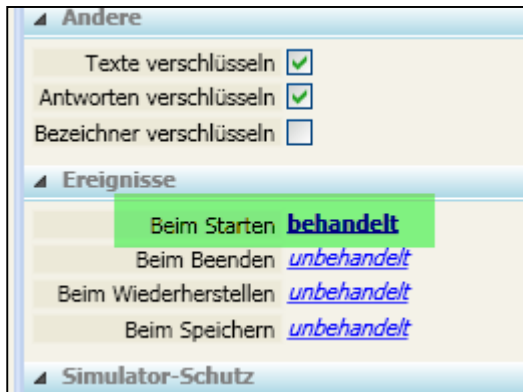
Jede der 4 Dialog-"Blasen" wird später als Text-Teilstück mit einem Button zur Bestätigung angezeigt. So können längere Texte aufgespalten werden, damit ein langer Text den Anwender nicht erschlägt.

Über entsprechende Grafiken zu jeder einzelnen "Blase" könnte auch ein Gespräch zwischen mehreren Personen simuliert werden – über unterschiedliche Bilder mit dazugehörendem Text. (Daher der Name "Dialog" für diesen Befehl).

Die einzelnen Komponenten können "minimiert" werden, wie im Bild die erste Blase. Das erhöht bei komplexen Abläufen die Übersichtlichkeit.

Damit ist die Ausprägung der Funktion "Beim Starten" für unseren Wherigo abgeschlossen und wir können das Fenster schließen.

Im Hauptfenster wird jetzt angezeigt, daß diese Funktion bzw. dieses Ereignis jetzt „mit Leben gefüllt“ ist:



Kennzeichnung, daß die Funktion ausprogrammiert wurde

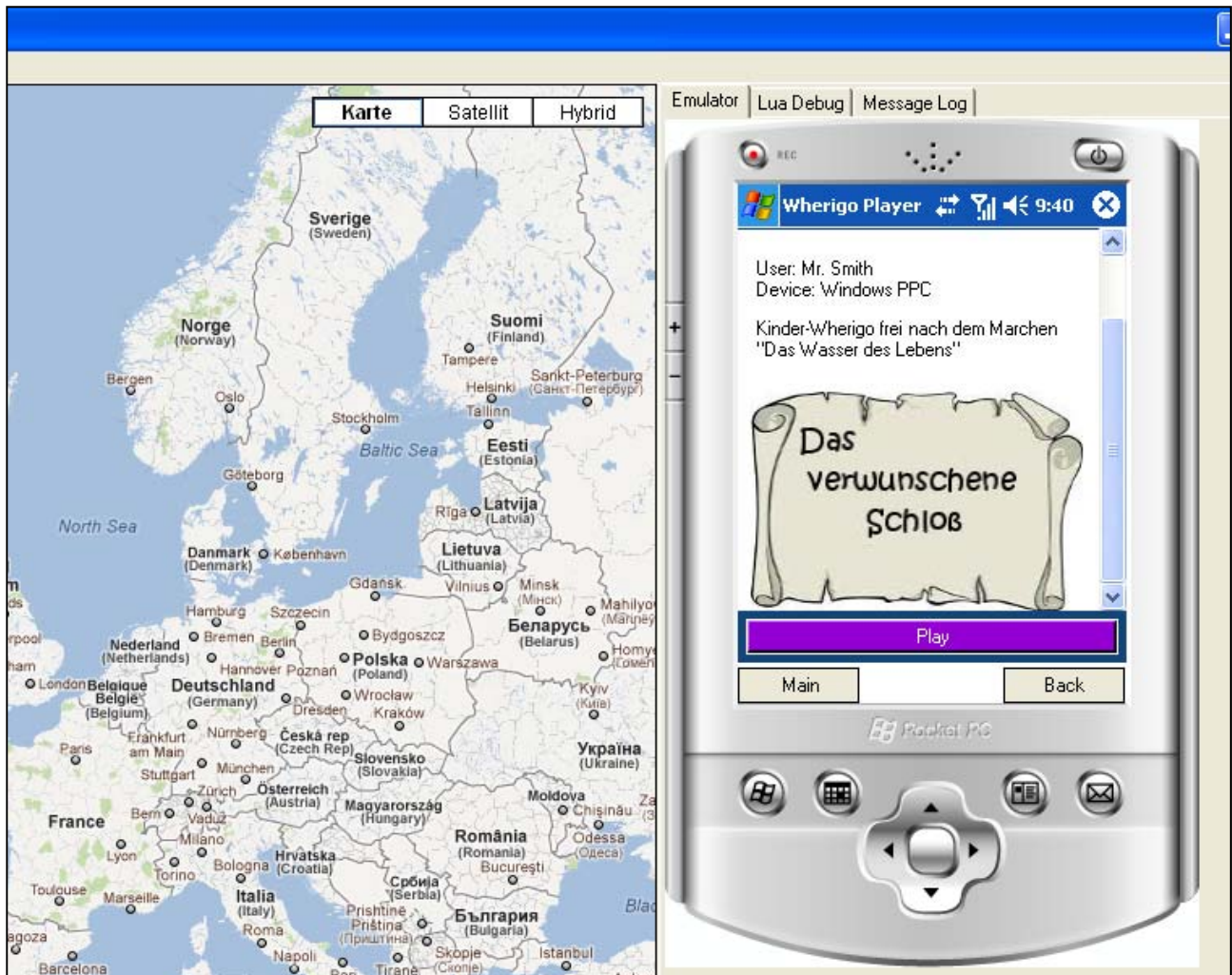
Jetzt: Sichern!

Ein erster Test

Nun wollen wir einmal ansehen, was wir bisher geschafft haben.

Dazu starten wir das Testtool über F5 oder über das Menü: "Erstellen" -> "Starten"

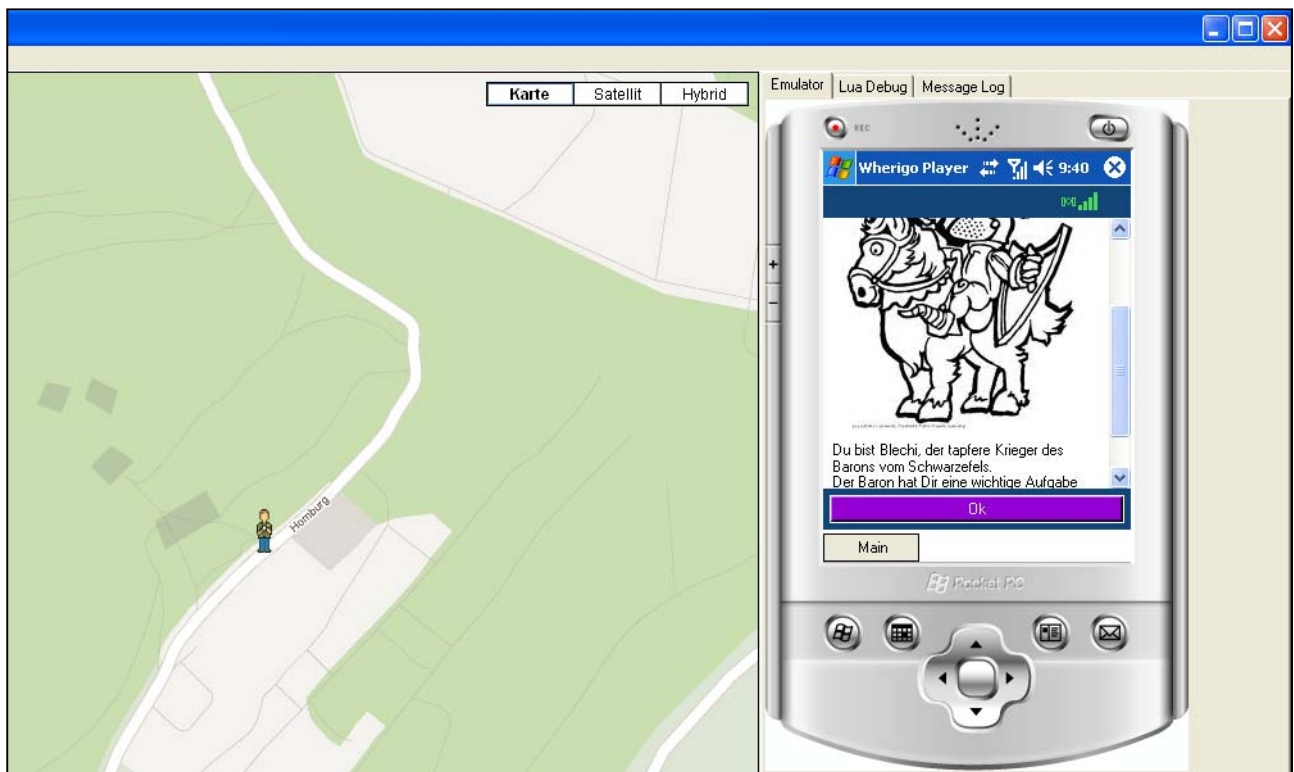
Wir sehen eine Karte und einen PDA, der unseren Wherigo enthält. Wenn wir diesen „starten“, sehen wir die Startgrafik unseres Wherigos und einen Play-Button:



Der erste Test

Sobald wir auf "Play" klicken, sehen wir den Start-Dialog, den wir eben programmiert haben im PDA. Es werden unser Bild, das wir dem ersten Dialog-Schritt zugewiesen haben und der Text gezeigt.

Nach und nach können wir uns die ganze Einführungs-Geschichte anzeigen lassen.



Das Testwerkzeug

Links sehen wir eine Karte und die noch inaktiven, unsichtbaren Zonen - damit wir für den Test wissen, wo diese genau liegen. Das Männchen simuliert den Spieler und dessen Bewegung.

In der Realität hätten wir diese Information natürlich nicht - auch nicht über den Wherigo-Player.

Leider ist dieses Testtool nicht ganz fehlerfrei. Es kann vorkommen, daß es abstürzt. Dann gibt es leider keine andere Möglichkeit, als das Tool zu schließen und neu zu beginnen.

Eine Aufgabe und ein erster Gegenstand

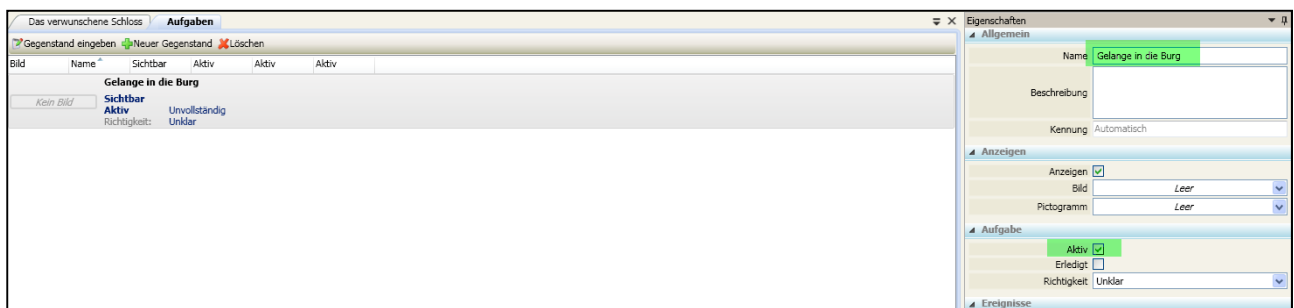
Nachdem die einführende Aufgabe durchgelesen wurde, bekommt man im Player einfach nur den Hauptbildschirm angezeigt. Außerdem wurde ja etwas von einem Ring erzählt...

Daher wollen wir jetzt den Ring erstellen und unserem Ritter „in die Tasche stecken“ außerdem wollen wir eine Aufgabe/Task erzeugen, die klar macht, wie es weitergehen soll.

Die Aufgabe

Im Fenster „Objekte und Eigenschaften“, das wir jetzt schon ganz gut kennen, klicken wir doppelt auf „Aufgaben“.

Es öffnet sich in der Bildschirmmitte ein Fenster für die Aufgaben – der Aufbau sollte mittlerweile schon etwas vertraut sein. Wir legen über „Neuer Gegenstand“ eine Aufgabe an.



Eine Aufgabe

Die Aufgabe erhält den Namen „Gelange in die Burg“ und wir setzen sie sofort aktiv, damit sie auf dem Hauptbildschirm des Players auch gleich angezeigt wird.

Wir könnten im Feld „Beschreibung“ noch detaillierteren Text zur Aufgabe hinterlegen. In diesem Fall sollte aber der Name sprechend genug sein.

So sieht's aus:



Anzeige der Aufgabe im Testwerkzeug

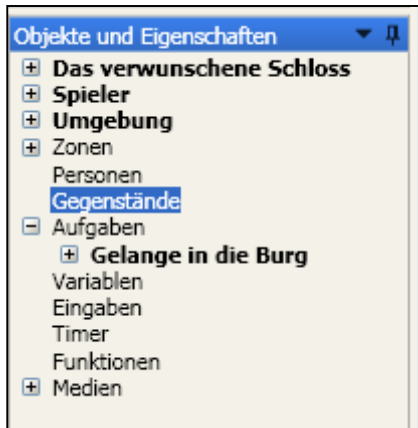
Nach dem Abspielen der Start-Geschichte sieht man jetzt den Auftrag, so daß die Richtung in der es weiter geht hoffentlich klar ist.

Man sollte bei der Erstellung eines Wherigos aber bedenken, daß dies den Spieler im Unklaren über den weiteren Verlauf lässt. Nicht jedem ist sofort klar, daß erwartet wird, daß man selbständig in Richtung Burg laufen soll.

Für uns, die wir den beabsichtigten Ablauf kennen, soll es zunächst einmal ausreichen. Es empfiehlt sich aber, den fertigen Wherigo später von Personen testen zu lassen, die den Ablauf nicht kennen. Es ist immer wieder überraschend, an welchen Stellen es dann hakt und wo ggf. Verbesserungen notwendig sind.

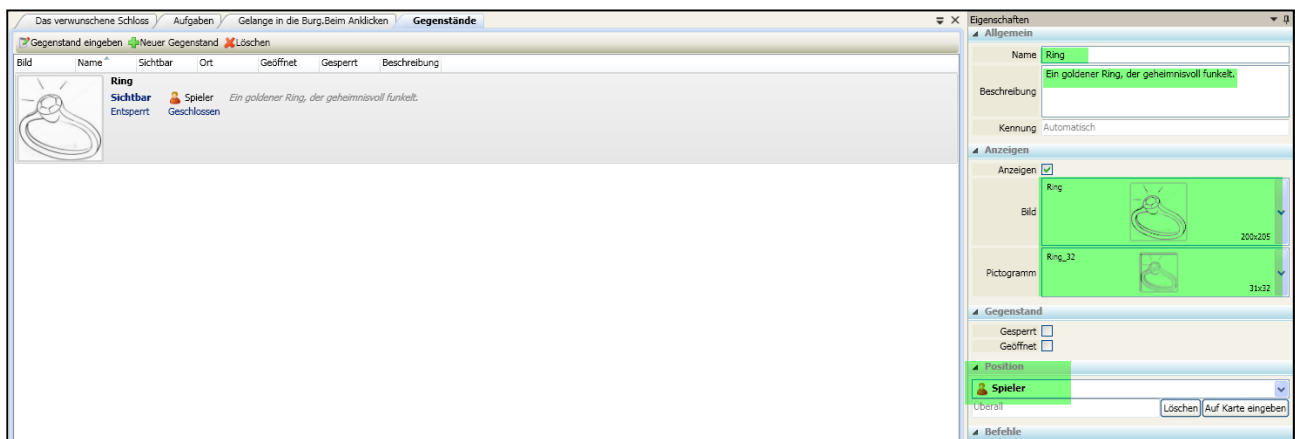
Der Ring

Jetzt wollen wir den Ring erstellen, dem Ritter „in die Tasche stecken“ und auch eine erste Funktion für den Ring selbst definieren.



Die Gegenstände im Objekt-Fenster

Auch hier ist das Fenster „Objekte und Eigenschaften“ wieder unser Freund. ☺
Ein Doppelklick auf „Gegenstände“ öffnet uns das richtige Fenster.



Der Ring – ein Objekt vom Typ Gegenstand

Wir erzeugen über „neuer Gegenstand“ (hier würde die falsche Übersetzung sogar passen) ein neues Objekt, das wir „Ring“ nennen. Über das Eigenschaftsfenster fügen wir eine Beschreibung hinzu, wählen die Bilder für die Anzeige und das Icon aus und (ganz wichtig!) geben als Position des Rings den Spieler an.

Damit hat dieser den Ring sozusagen einstecken!

Eine erste eigene Funktion

Nun wollen wir den Ring nicht nur einstecken haben, sondern es soll etwas passieren, wenn wir drauf klicken.

Wer bis jetzt mitgelesen hat, wird sich denken, daß jetzt von dieser Funktion die Rede ist:

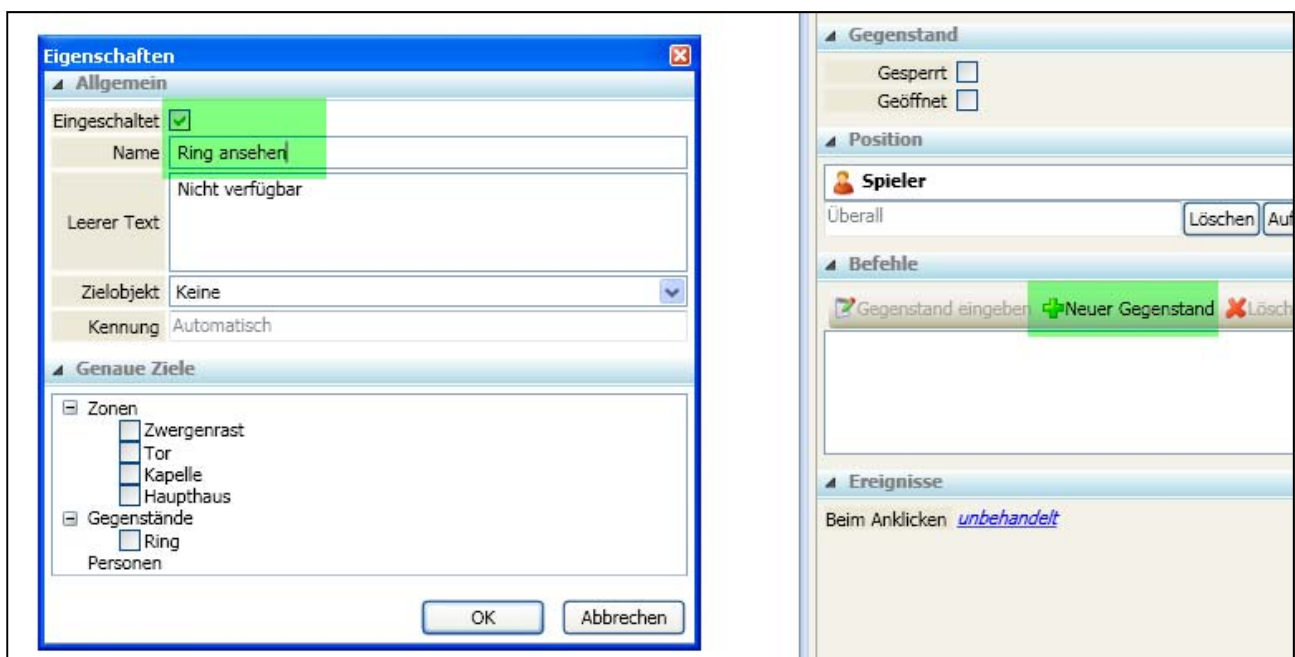


Ereignis „Beim Anklicken“

ABER: Dem ist leider nicht so! Die Funktion „beim Anklicken“ (engl. onClick) funktioniert nicht auf allen Geräten! Zum Beispiel nicht auf den Oregons von Garmin.

Daher merken wir uns: „Beim Anklicken“ = böse!!!! ☹

Das ist aber nicht wirklich schlimm, es gibt eine Möglichkeit, wie man das umgehen kann und es ist, meiner Ansicht nach, auch die elegantere Lösung!



eine individuelle Funktion für den Gegenstand Ring

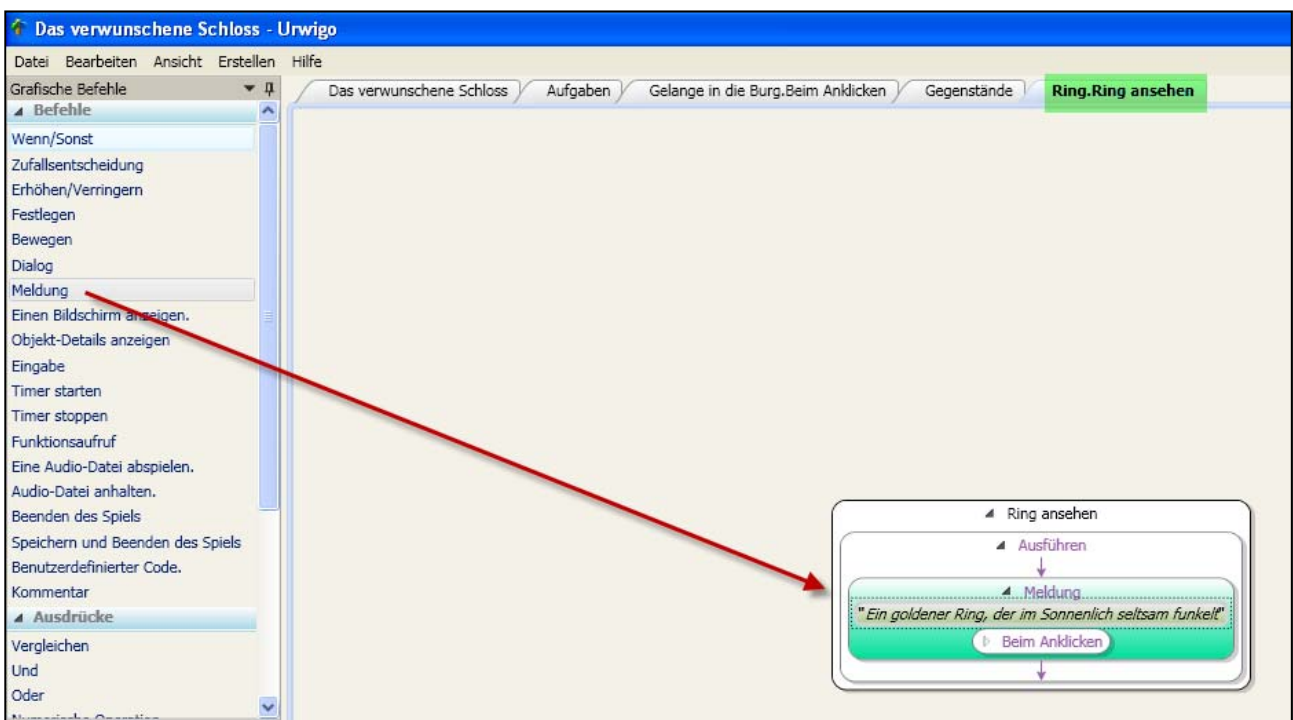
Im Eigenschaftsfenster klicken wir auf der rechten Seite unter „Befehle“ auf „neuer Gegenstand“ (eigentlich neue Funktion).

Es erscheint ein Popup mit einem neuen Befehl, den wir mit „Ring ansehen“ benennen.
Nachdem wir auf o.k. geklickt haben, wurde ein zusätzlicher Befehl erstellt:



Eine neue Funktion für den Gegenstand

Diesen neuen Befehl können wir jetzt ausprägen. Dafür wollen wir erst einmal nur ein wenig Text hinterlegen, wie wir das mittlerweile schon kennen.

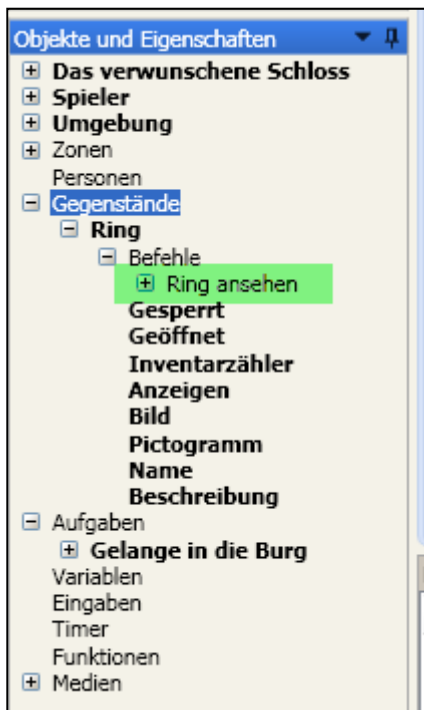


Eine Textmeldung für den Ring

Kleines Intermezzo zur Erinnerung an die Theorie:

Über die Bezeichnung des Reiters oben (grün markiert) kann man auch hier wieder die typische objektorientierte Schreibweise mit Punktnotation erkennen: <Objekt>.<Funktion oder Eigenschaft>. Hier ist der Ring das Objekt und behandelt wird die Funktion „Ring ansehen“.

Auch in unserem Fenster „Objekte und Eigenschaften“ ist jetzt ersichtlich, daß der Gegenstand Ring einen individuellen Befehl erhalten hat:



der neue Befehl für den Ring.

Und jetzt: Speichern nicht vergessen!!!

Und so sieht's aus:



Der Ring im Inventar des Testwerkzeugs

Wenn wir jetzt die „Test-Maschine“ anwerfen, sehen wir nach dem Abspielen des Intros die eben erzeugten Objekte. Eine Aufgabe und den Ring, den wir jetzt im „Rucksack“ haben. Der Ring hat zusätzlich eine individuelle Funktion, die Infos zum Gegenstand bereit stellt.

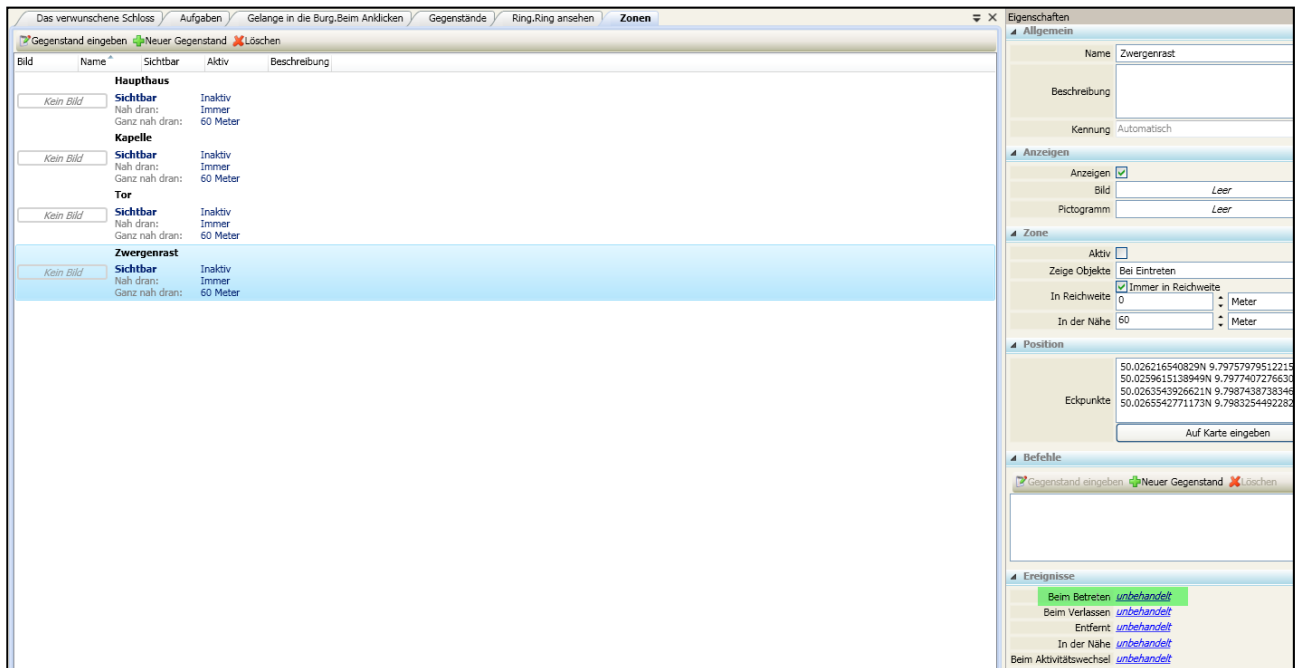
Viel Spaß beim Ausprobieren! ☺

Betreten der ersten Zone

Prinzipiell steht unser Spieler jetzt immer noch auf dem Parkplatz, wo der Whereigo startet. Er hat die Geschichte gelesen, die Aufgabe und den Ring angesehen und läuft jetzt (hoffentlich) los in Richtung Burg.

Auf dem Weg dorthin, soll er in der ersten Zone, den Zwerg treffen. Nachdem dies überraschend geschehen soll, haben wir diese Zone bis jetzt deaktiviert.

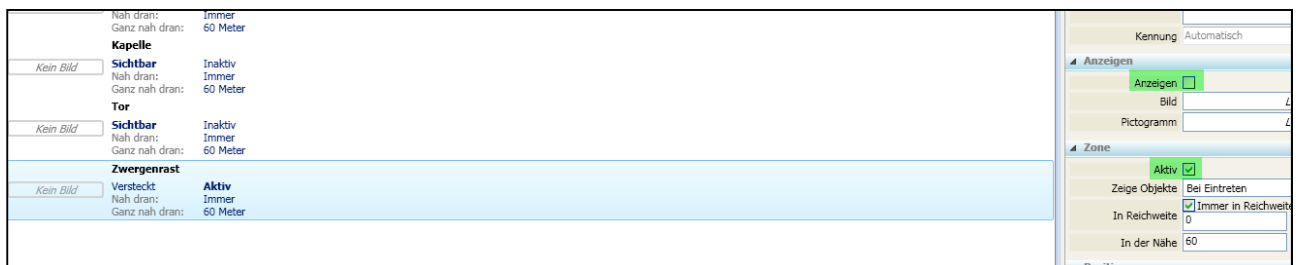
Wir wollen jetzt dafür sorgen, daß diese beim Betreten aktiviert wird.



Bearbeiten der Zone „Zwergenrast“

Das Kochrezept nachdem die Zonen erstellt werden, erzeugt auch einige „Funktions-Hülsen“. Z. B. die Funktion „Beim Betreten“, die automatisch aufgerufen wird, wenn ein Spieler die Zone betritt. Diese „Hülsen“ können wir mit individuellem Code befüllen, um die Interaktion voran zu treiben.

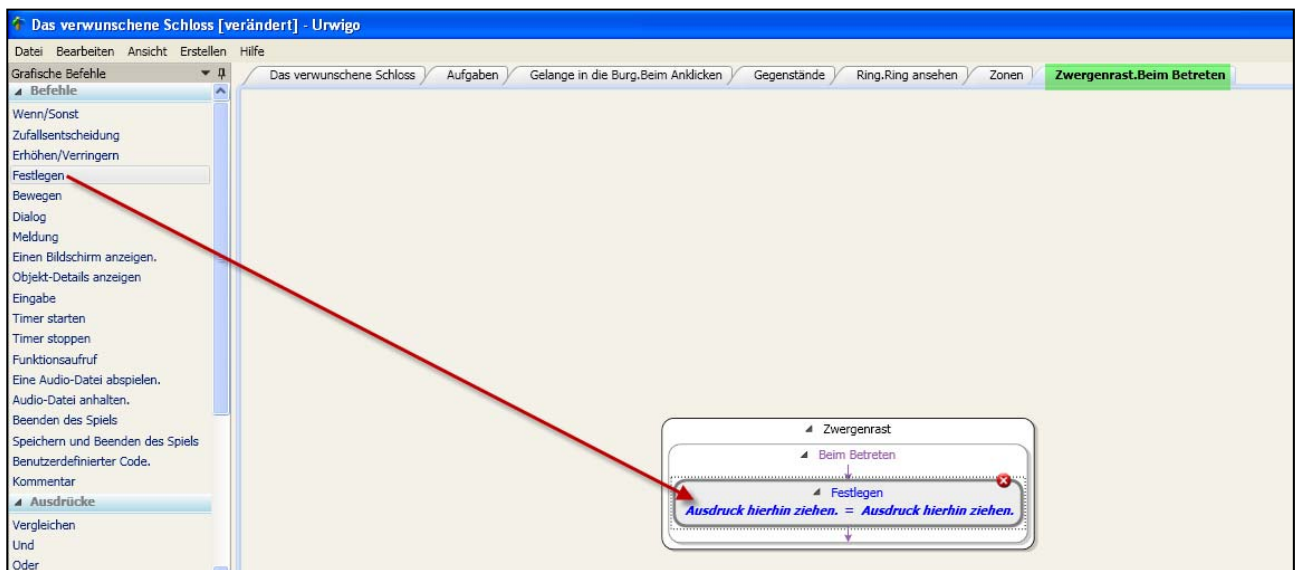
Wir rufen die Zonenübersicht über das Fenster „Objekte und Eigenschaften“ auf und bearbeiten das Objekt Zwergenrast und dort auch gleich das Ereignis bzw. die Funktion „beim Betreten“. Aber vorher ist noch eine kleine Änderung notwendig.



Modifikation der Zoneneigenschaften

Wir setzen die Zone Zwergenrast auf „aktiv“ und schalten aber die Anzeige aus. Damit stellen wir sicher, daß die Zone jetzt „reagiert“, d. h. die vorhandenen Standardfunktionen werden aufgerufen, wenn das entsprechende Ereignis eintritt. Mehr dazu im nächsten Bild.

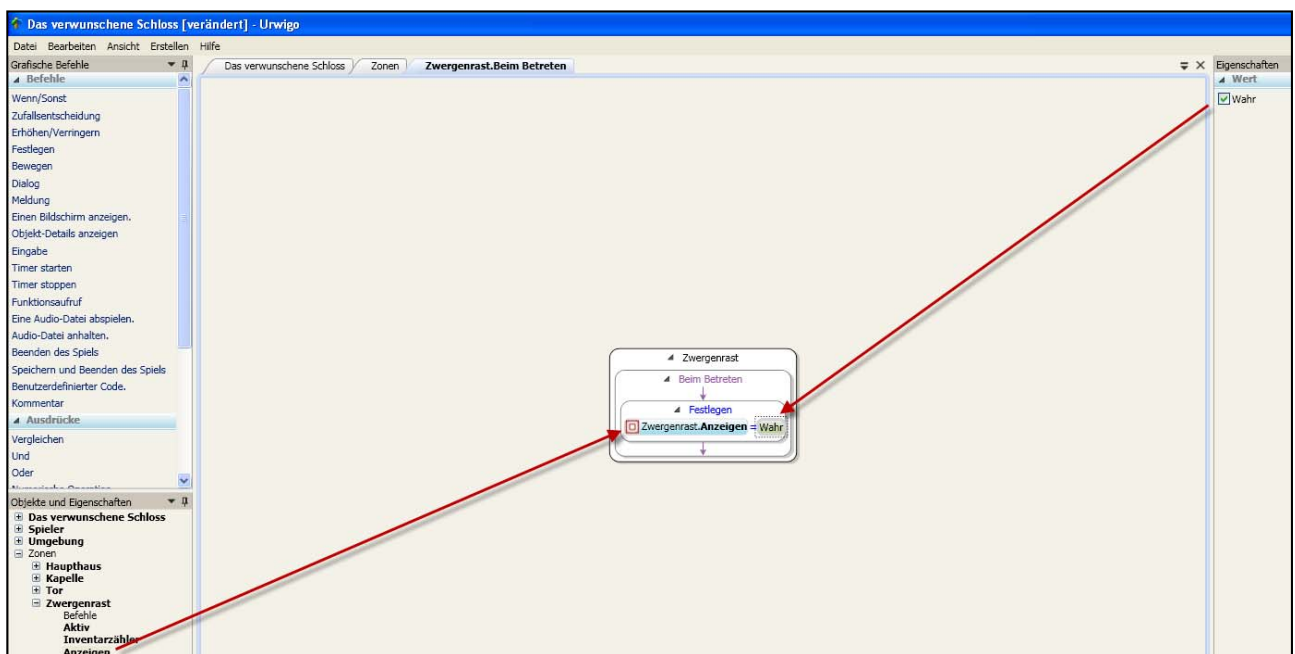
Wir wählen bei den vorhandenen Ereignissen der Zone das Ereignis „Beim Betreten“ aus und füllen dieses mit Leben. Auch hier nochmal (bzw. schon wieder) die Erinnerung an die objektorientierte Schreibweise (grün markiert) <Objekt>.<Ereignis>:



Ausprägen des Ereignisses „Beim Betreten“ für die Zone Zwergenrast

Jetzt kommt etwas Neues: Wir ziehen den Befehl „Festlegen“ aus dem Befehlsvorrat herüber. Mit diesem Befehl können wir Eigenschaften anderer Objekte verändern. Diese Eigenschaften können wir im Fenster „Objekte und Eigenschaften“ sehen, wo sie verwaltet werden.

Da wir die Zone beim Betreten anzeigen wollen, führen wir folgende Aktionen durch:



Der fertige Befehl „Festlegen“

Wir wählen die Eigenschaft „Anzeigen“ für das Objekt „Zwergenrast“ und ziehen es in die linke Hälfte des „Festlegen“-Befehls. Die rechte Hälfte ist automatisch mit „Falsch“ belegt. Dies ändern wir über die Checkbox im Eigenschaftsfenster (siehe oben).

Damit setzen wir die Zone beim Betreten von unsichtbar auf sichtbar. Erst dann können wir im Spiel die Zone „sehen“ und auch alle Objekte und Personen, die sich darin befinden!

Noch ein kleiner Ausflug in die Informatik:

Was soll dieses „wahr“ und „falsch“ im Festlegen-Befehl?

Die Eigenschaft „Anzeigen“ ist ein sog. Boolean. Dies ist ein Feldtyp in Computersprachen, der nur 2 Zustände kennt, nämlich eben „wahr“ und „falsch“ oder sehr frei übersetzt: „ein“ und „aus“ – eigentlich ja eher 0 und 1....

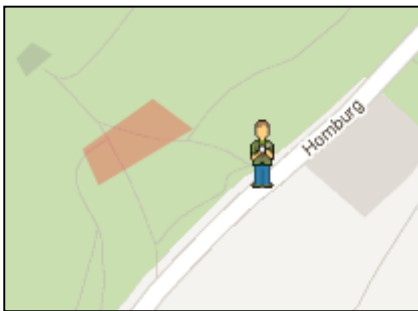
Die Lampe auf Eurem Schreibtisch ist sozusagen auch ein booleanscher Speicher: Sie kann die Zuständen „ein“ und „aus“ annehmen oder halt „wahr“ und „falsch“. 😊

In diesem Fall können wir „wahr“ mit „eingeschaltet“ und „falsch“ mit „ausgeschaltet“ gleichsetzen.

Speichern nicht vergessen!

So sieht's aus:

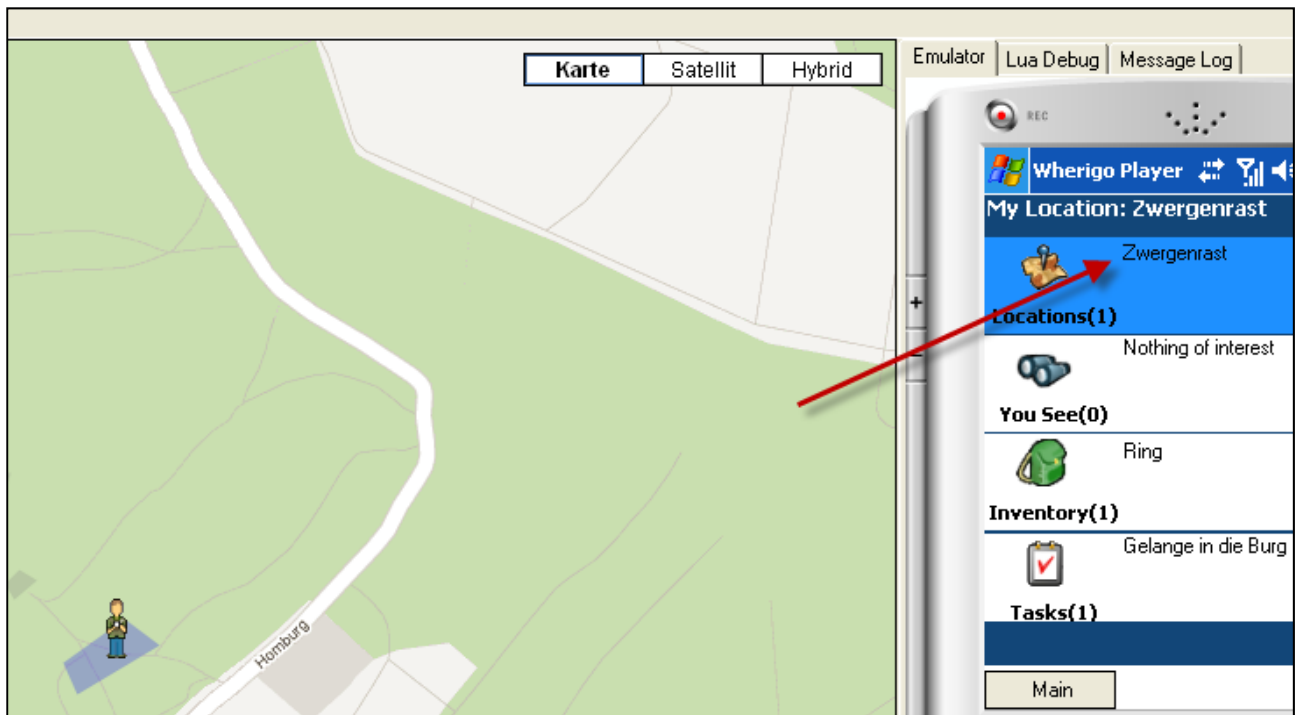
Wenn wir jetzt die Testumgebung starten, sehen wir, daß die erste Zone anders gefärbt ist:



die unsichtbare, aber aktive Zone in der Testumgebung

Dies zeigt an, daß diese zwar aktiv, aber für den Spieler zunächst noch unsichtbar ist.

Simulieren wir jetzt Bewegung und ziehen unsere Figur in die Zone, stellen wir fest, daß diese sich violett einfärbt:



durch das Betreten aktivierte Zone

Das bedeutet, daß die Zone aktiv ist und daß der Spieler mit darin befindlichen Objekten (Gegenständen und Personen) interagieren kann bzw. diese überhaupt erst angezeigt bekommt.

Zusätzlich wird die aktive Zone auch im Player angezeigt.

Erste Interaktion

Erinnern wir uns, was hier geschehen soll:

Wir wollen den Zwerg treffen und uns unterhalten. Sind wir unfreundlich, soll die Sache unerfreulich für uns ausgehen. Sind wir höflich bekommen wir einen guten Rat, ein Brot und einen Stab.

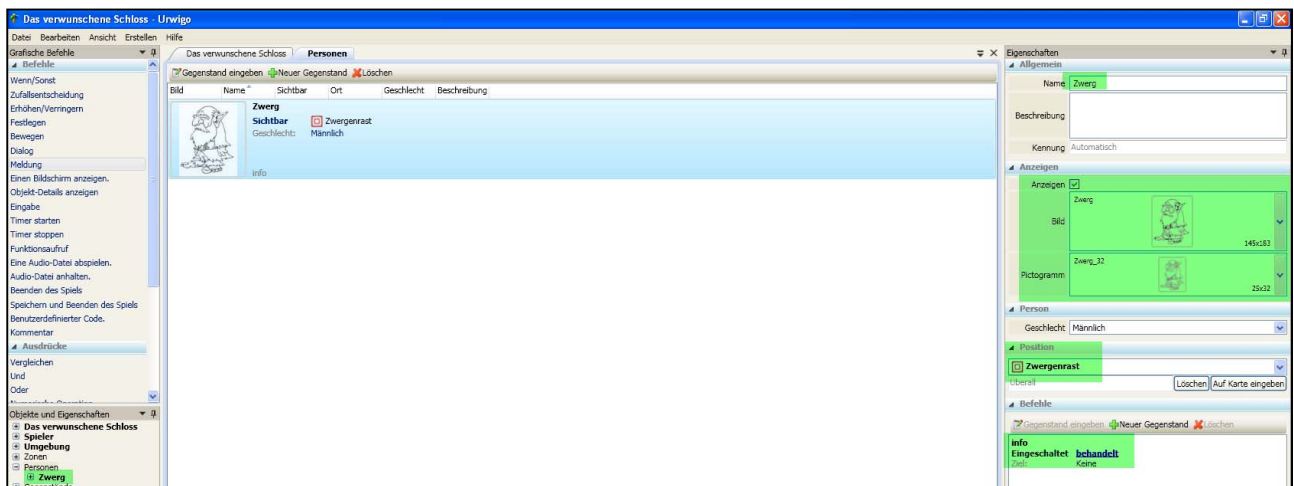
Das bedeutet, daß wir 3 neue Objekte brauchen

- den Zwerg (als Objekt vom Typ Person)
- das Brot (als Objekt vom Typ Gegenstand)
- den Stab (ebenfalls ein Objekt vom Typ Gegenstand).

Neue Objekte

Fangen wir zunächst mit dem Zwerg an.

Über das Fenster „Objekte und Eigenschaften“ legen wir für den Objekttyp Personen einen Zwerg an.



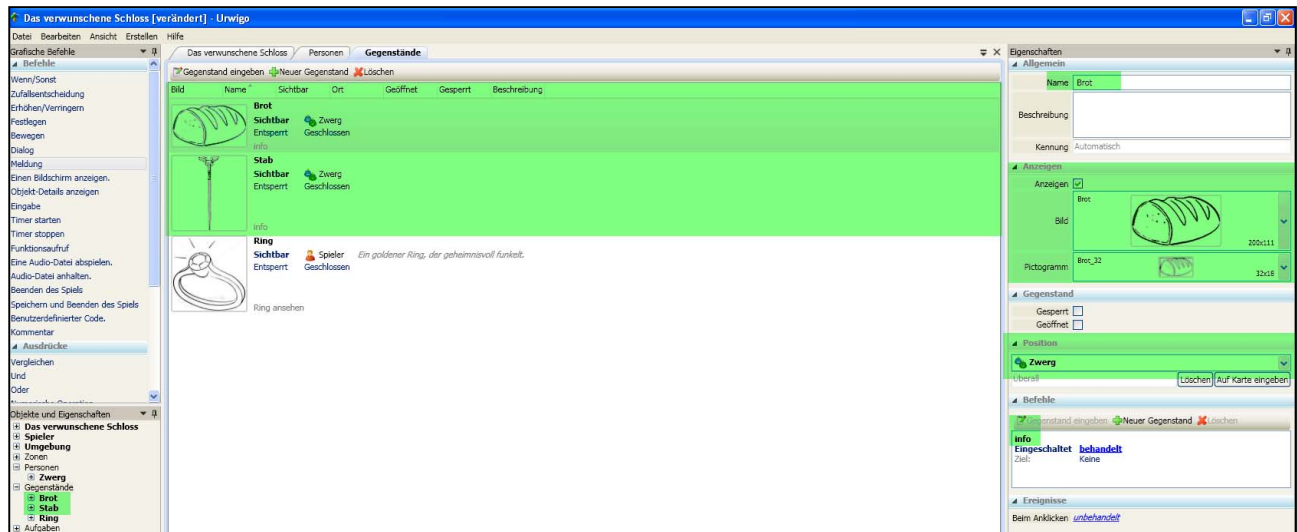
Der Zwerg (Ein Objekt vom Typ Person)

Ich werde hier nicht mehr ins Detail gehen, da die Vorgehensweise jetzt bekannt sein dürfte.

Wir benennen das neue Objekt mit „Zwerg“, ordnen Bild und Icon zu und setzen den Zwerg in der Zone „Zwergenrast“ ab. Außerdem erzeugen wir einen individuellen Befehl „info“ in dem wir eine kurze Textmeldung hinterlegen, die den Zwerg kurz beschreibt (vgl. ZIP-Datei des Projektes)

Im Prinzip nichts anderes, als wir zur Ausprägung des Rings gemacht haben.

Der Zwerg benötigt natürlich noch das Brot und den Stab. Auch diese legen wir analog an.



2 neue Objekte bei den Gegenständen

Auch hier will ich jetzt nicht mehr ins Detail gehen. Die Vorgehensweise zum Anlegen dieser noch rudimentären Objekte ist bekannt. Wichtig ist, daß beide einen zusätzlichen Befehl namens „info“ erhalten, über den eine kurze Beschreibung geliefert wird und daß beide Objekte dem Zwerg zugeordnet werden.

Die Unterhaltung mit dem Zwerg

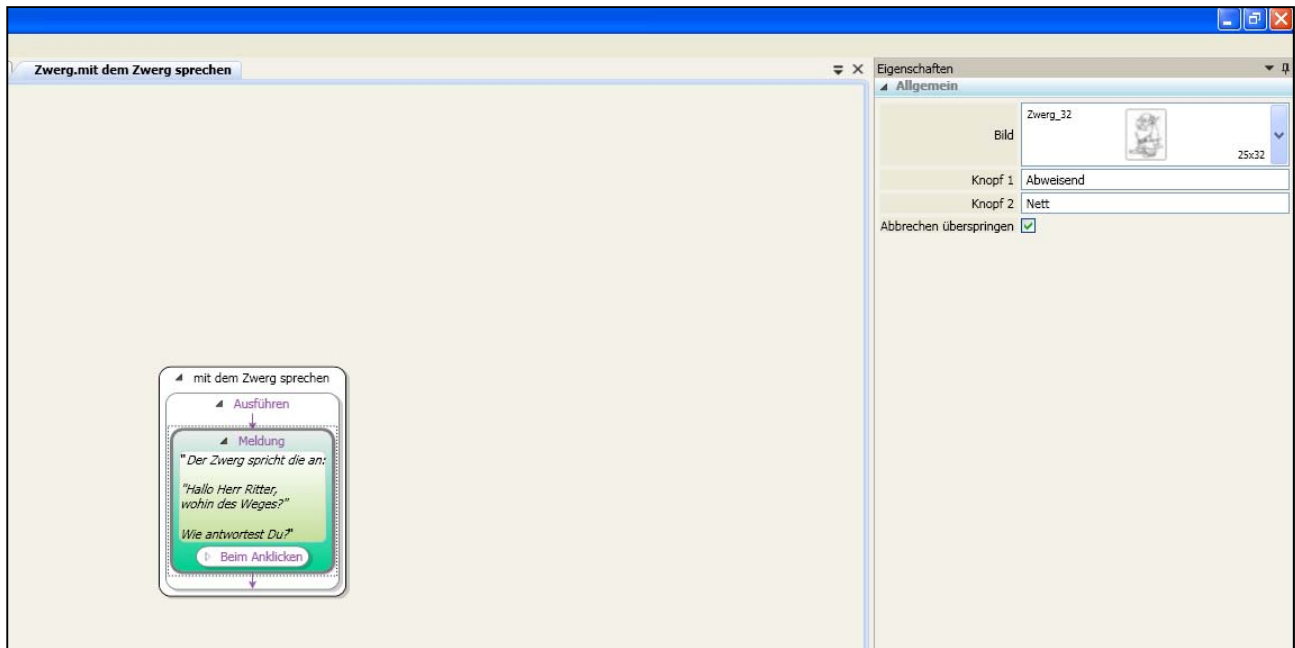
Jetzt wird es eine Spur komplexer, wir wollen die Unterhaltung mit dem Zwerg und die entsprechenden Reaktionen programmieren:

Anzeigen	
Anzeigen	<input checked="" type="checkbox"/>
Bild	<div>Zwerg</div> <div></div> <div>14</div>
Pictogramm	<div>Zwerg_32</div> <div></div>
Person	
Geschlecht	Männlich
Position	
<input checked="" type="checkbox"/> Zwergenrast	
Überall	<div>Löschen</div> <div>Auf Karte</div>
Befehle	
<div> Gegenstand eingeben  Neuer Gegenstand  Löschen</div>	
info	
Eingeschaltet <u>behandelt</u>	
Ziel: Keine	
mit dem Zwerg sprechen	
Eingeschaltet <u>unbehandelt</u>	
Ziel: Keine	
Ereignisse	
Beim Anklicken <u>unbehandelt</u>	

Die Programmierung des Zwergs

Dazu erzeugen wir den Befehl „mit dem Zwerg sprechen“

Zunächst der einfache Teil



Start der Ausprägung „mit dem Zwerg sprechen“

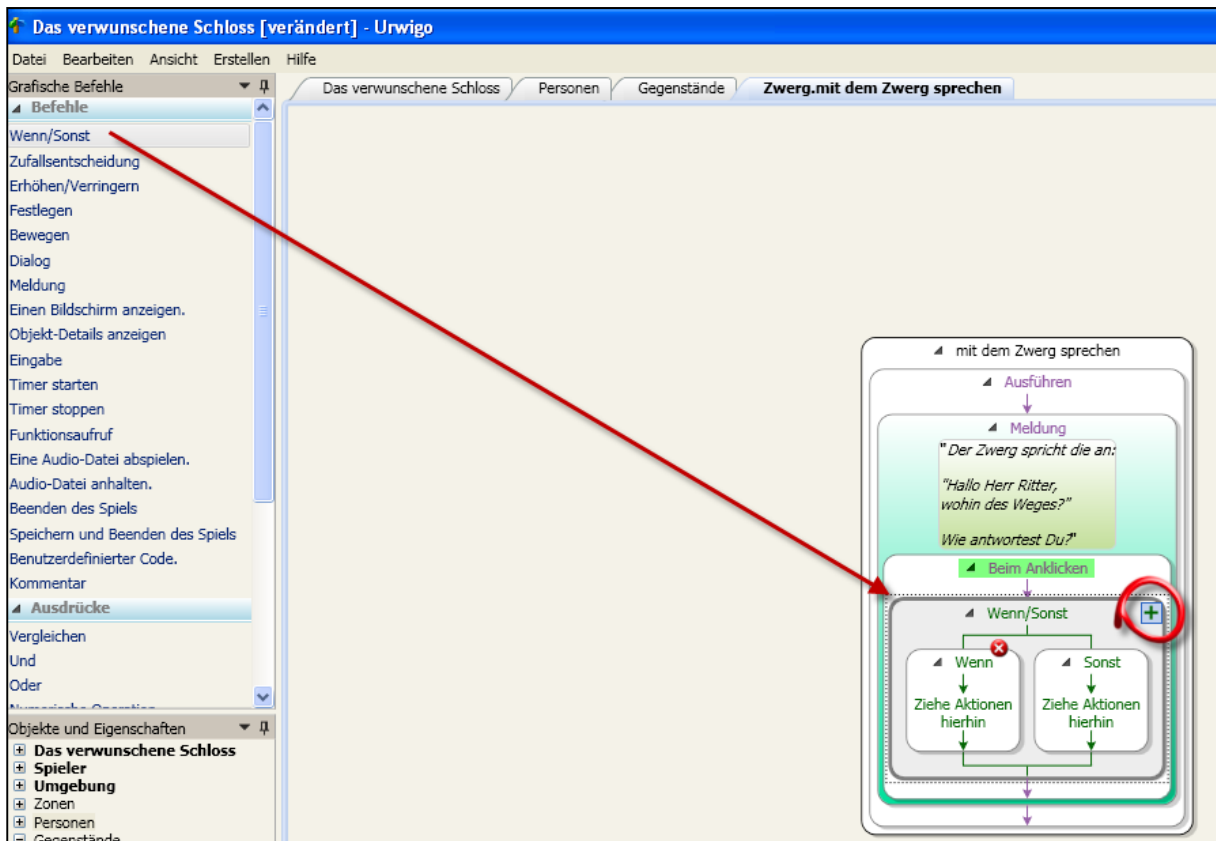
Wir ziehen eine „Meldung“ in den Befehlsfluß. Im Eigenschaftsfenster rechts weisen wir ein Bild zu und Belegen die Knöpfe im rechten Fenster mit den Texten „Abweisend“ und „Nett“.

In der Meldung hinterlegen wir ein bisschen Text, der abschließend fragt, wie wir dem Zwerg antworten möchten. Wenn wir einen kurzen Test machen, gibt es für den Zwerg nun die Option „mit dem Zwerg sprechen“. Unter dem Text finden wir 2 Buttons die mit „Abweisend“ und „Nett“ beschriftet sind.

Probiert es zwischendurch ruhig mal aus. 😊

Natürlich passiert jetzt noch nichts. Das realisieren wir im Folgenden.

Jetzt wollen wir eine Bedingung hinterlegen, damit wir unterscheiden können, wie geantwortet wurde:



Die wenn/sonst-Bedingung

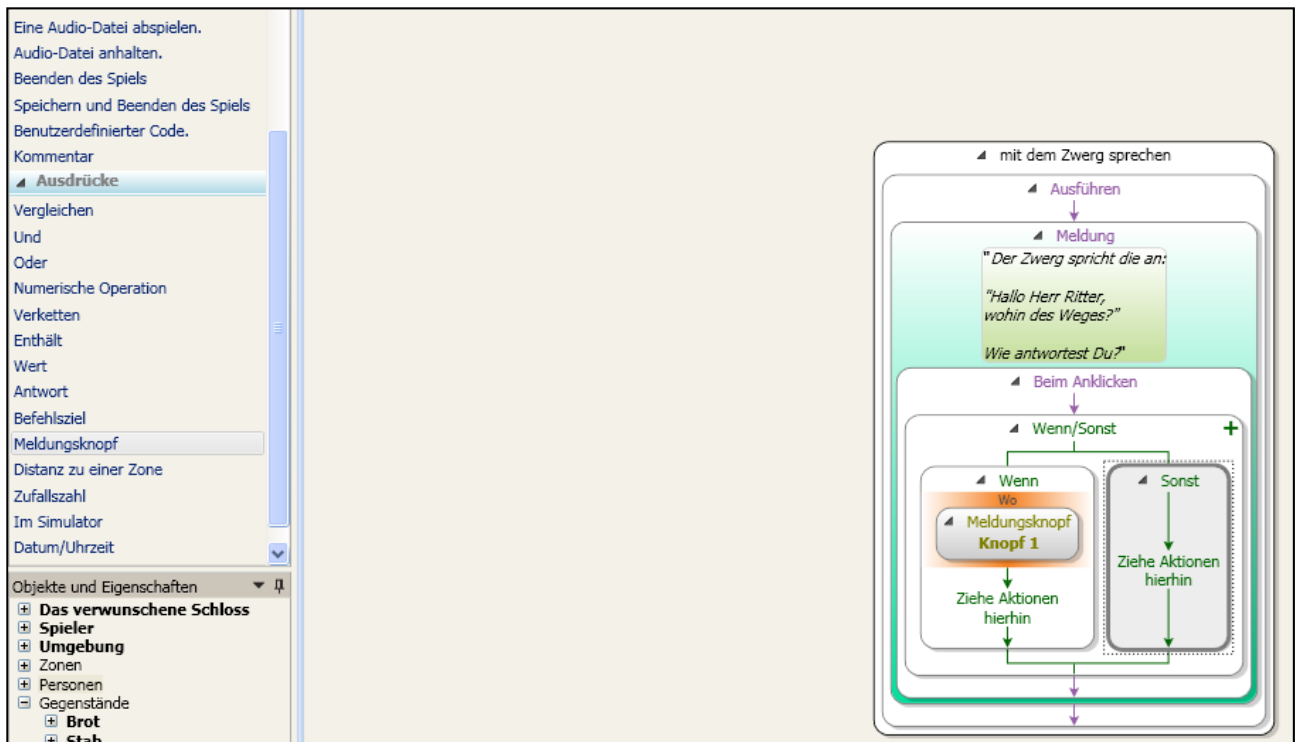
- aus dem Befehlsvorrat ziehen wir den Befehl „Wenn/Sonst“ in das Feld „Beim Anklicken“
- mit dem Plus (roter Kringel) erweitern wir den Wenn-Befehl um den Sonst-Zweig (so daß es wie oben aussieht)
- die Fehlermeldung ignorieren wir erst einmal

In diesem Wenn/Sonst-Befehl muß nun eine Bedingung hinterlegt werden. Davon ist abhängig, wie das Programm weiter fließt.

Trifft die (noch zu hinterlegende) Bedingung zu, wird der linke Strang ausgeführt (die weiße Blase, in der „Wenn“ steht). In jedem anderen Fall der rechte Strang (die „Sonst-Blase“).

Grundsätzlich kann man sagen, daß der Ausdruck in einem Wenn/Sonst-Befehl auf einen Boolean-Wert zurück zu führen sein muß. Oder einfacher ausgedrückt: Die Bedingung in Wenn/Sonst muß auf die Aussage WAHR oder FALSCH reduzierbar sein.

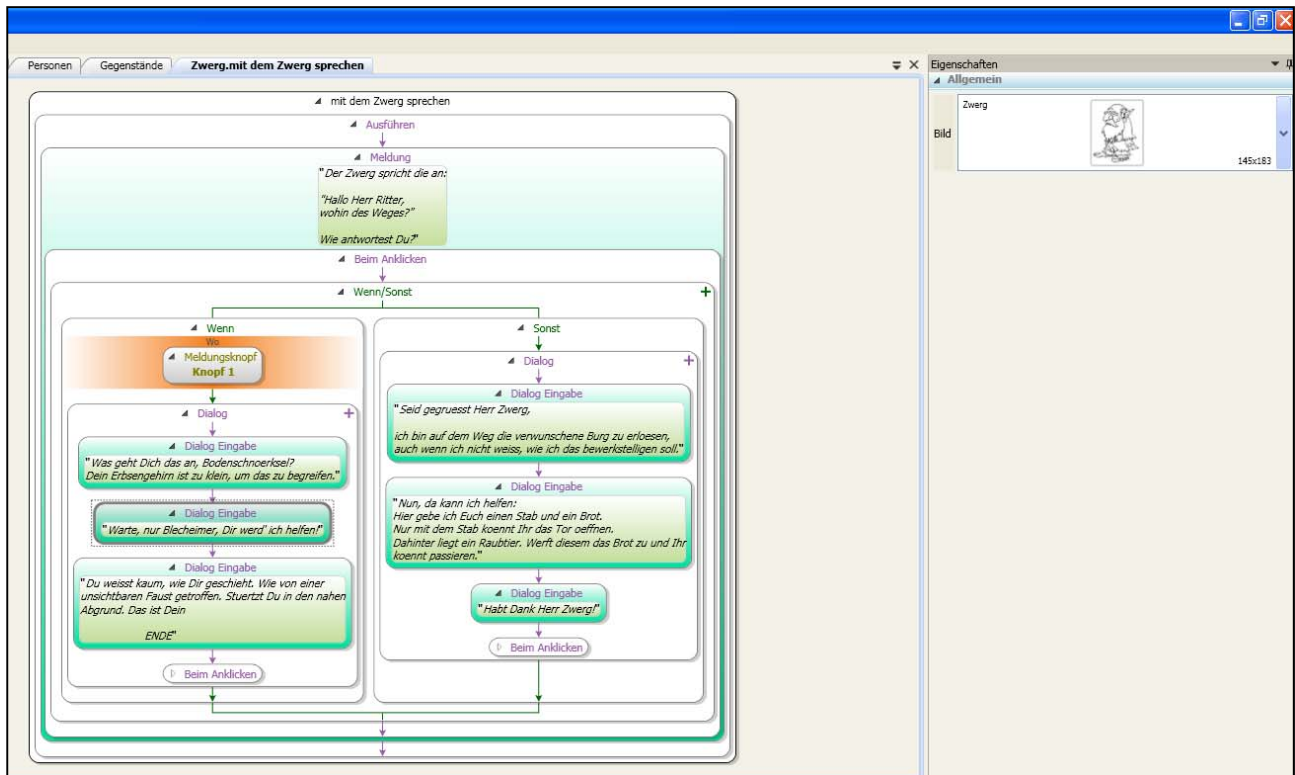
Aus dem Befehlsvorrat ziehen wir aus dem Unterpunkt „Ausdrücke“ den Ausdruck „Meldungsknopf“ in die Wenn-Bedingung:



Ausprägen der Bedingung

Der Ausdruck „Meldungsknopf“ ist schon mit „Knopf 1“ vorbelegt. Das lassen wir so. Ab jetzt, wenn in der übergeordneten Meldung der Knopf 1 („Abweisend“) gedrückt wird, kann die Bedingung mit WAHR bewertet werden und die linke Seite der Wenn/Sonst-Bedingung wird ausgeführt. In jedem anderen Fall ist die Bedingung FALSCH und es wird die rechte Seite ausgeführt.

Danach fügen wir für jede der beiden Ausprägungen etwas Prosa über einen Dialog-Strang an



Die erweiterte Wenn/Sonst-Bedingung. Textfluss abhängig von der Eingabe.

Den einzelnen Dialog-„Blasen“ sind Bilder zugewiesen, damit klar ist, ob der Zwerg oder Ritter spricht. So kann, wie schon erwähnt, eine Unterhaltung simuliert werden.

Inhaltlich erhält der Ritter auf der rechten Seite Hilfe vom Zwerg und sowohl den Stab und das Brot. Auf der rechten Seite geht die Geschichte für den Ritter weniger gut aus.... ☺

Wenn wir jetzt das Testtool anwerfen, können wir beide Versionen testen, da unsere Antwort außer einer unterschiedlichen Dialogabfolge noch keine Auswirkungen hat.

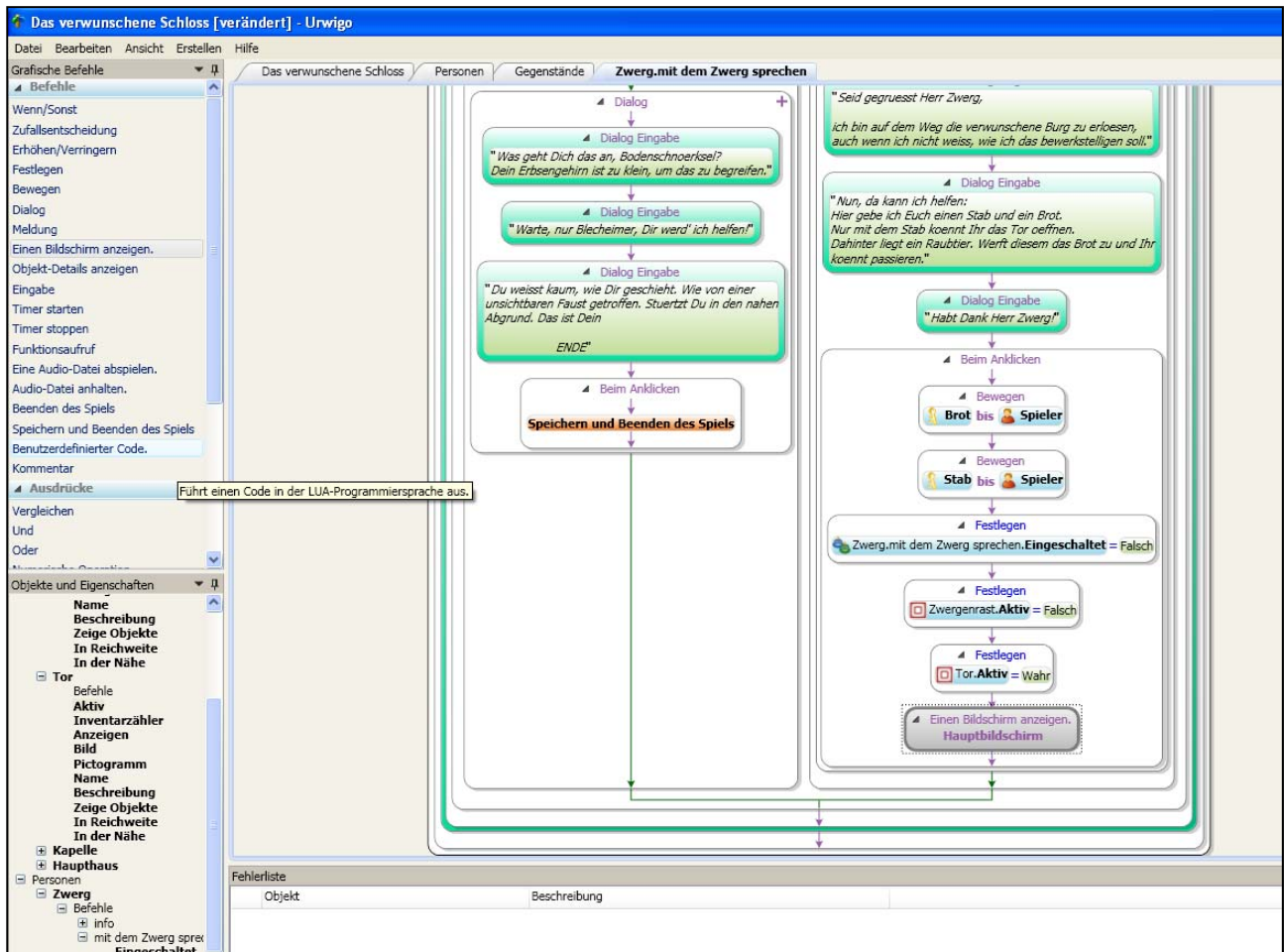
Speichern nicht vergessen!

Nun muß die Wahl des Spielers auch Konsequenzen haben:

Geht die Geschichte schlecht aus, ist das Spiel zu Ende. Hilft der Zwerg aber dem Ritter ist jede Menge zu programmieren:

- wir müssen dem Ritter den Stab und das Brot geben
- diese Zone deaktivieren und die nächste aktivieren
- vorsichtshalber die Frage beim Zwerg deaktivieren
- und zurück zum Hauptbildschirm hüpfen, damit es schöner aussieht ☺

Die genannten Punkte haken wir mit etlichen neuen Befehlen ab:



Erweiterung der Befehlsabläufe

Zuerst die linke Seite:

Das ist fast einfach genug. Aus dem Befehlsvorrat ziehen wir „Speichern und Beenden des Spiels“ in das Feld „Beim Anklicken“ des Dialogs. Damit endet das Spiel wenn diese Option gewählt wurde.

Jetzt die rechte Seite:

Wir ziehen folgende Befehle aus dem Befehlsvorrat in das Feld „Beim Anklicken“ des Dialogs:

- 2x den Befehl „Bewegen“
- 3x den Befehl „Festlegen“
- den Befehl „Einen Bildschirm anzeigen“

Über den Befehl „Bewegen“ schieben wir das Brot und den Stab in das Inventar des Spielers. Die Objekte, die notwendig sind, um den Befehl mit Inhalt zu füllen, ziehen wir aus dem Fenster „Objekte und Eigenschaften“ herüber. Brot und Stab sind unter Gegenstände abgelegt. Der Spieler ist standardmäßig vorhanden.

Beim Zwerg klappen wir alles soweit auf, bis für den Befehl „mit dem Zwerg sprechen“ das Attribut „Eingeschaltet“ angezeigt wird. Dieses ziehen wir auf die linke Seite des ersten „Festlegen“-Befehls. Die

rechte Seite steht schon auf „Falsch“, was uns nur recht ist. Damit wir diese Eingabemöglichkeit deaktiviert und sie kann kein 2. Mal aufgerufen werden, da diese Aktion jetzt nicht mehr benötigt wird.

Auf die gleiche Art und Weise ändern wir die Attribute „Aktiv“ für die Zonen Zwergenrast und Tor. Die Zone Zwergenrast deaktivieren wir, während wir die Zone „Tor“ aktivieren.

Der Befehl „Einen Bildschirm anzeigen“ steht schon auf „Hauptbildschirm“, was das ist, was wir wollen. Dadurch springt das Programm zum Hauptbildschirm, was dem Spieler später ein paar Aktionen spart.

Jetzt: Sichern!!!

Wie sieht's aus?

Wie das Testtool funktioniert, sollte jetzt ja bekannt sein. Daher an dieser Stelle kein weiteres Bild.

Jetzt sollte die Cartridge bis hierhin getestet werden. Für eine „echte“ Cartridge muß man hier auch alle möglichen „Dummheiten“ prüfen. Was passiert, wenn die Zone zwischendurch verlassen wird, welche Fehleingaben sind möglich, etc.

Wenn alles richtig gemacht wurde, haben wir folgenden Stand:

- der Hauptbildschirm wird angezeigt
- wir haben das Brot und den Stab im Rucksack
- die Zone mit dem Zwerg ist deaktiviert und damit auch der Zwerg nicht mehr sichtbar
- die Zone mit dem Tor ist aktiviert und kann angesteuert werden

Zwei weitere Objekte

Wichtig: Mir ist, nachdem das Tutorial schon sehr gewachsen war, aufgefallen, daß es für Anfänger missverständlich sein könnte, daß sowohl eine Zone, als auch ein Gegenstand gleichlautend „Tor“ heißen. Ich werde daher jeweils immer in Klammern dahinter schreiben, ob Tor oder Gegenstand gemeint sind. Sorry, blöd von mir... ☹

Für die nächste Zone benötigen wir neue Objekte und neue Befehle:

- das verschlossene Tor (als Gegenstand)
- den Löwen – zunächst noch unsichtbar, da er ja hinter dem Tor sitzt
- eine Möglichkeit, mit dem Stab zu schlagen
- eine Möglichkeit, das Brot zu verfüttern
- außerdem müssen wir beim Verlassen der Zone prüfen, ob der Löwe satt ist

Fangen wir mit 2 neuen Objekten an, die wir als Gegenstand erzeugen.



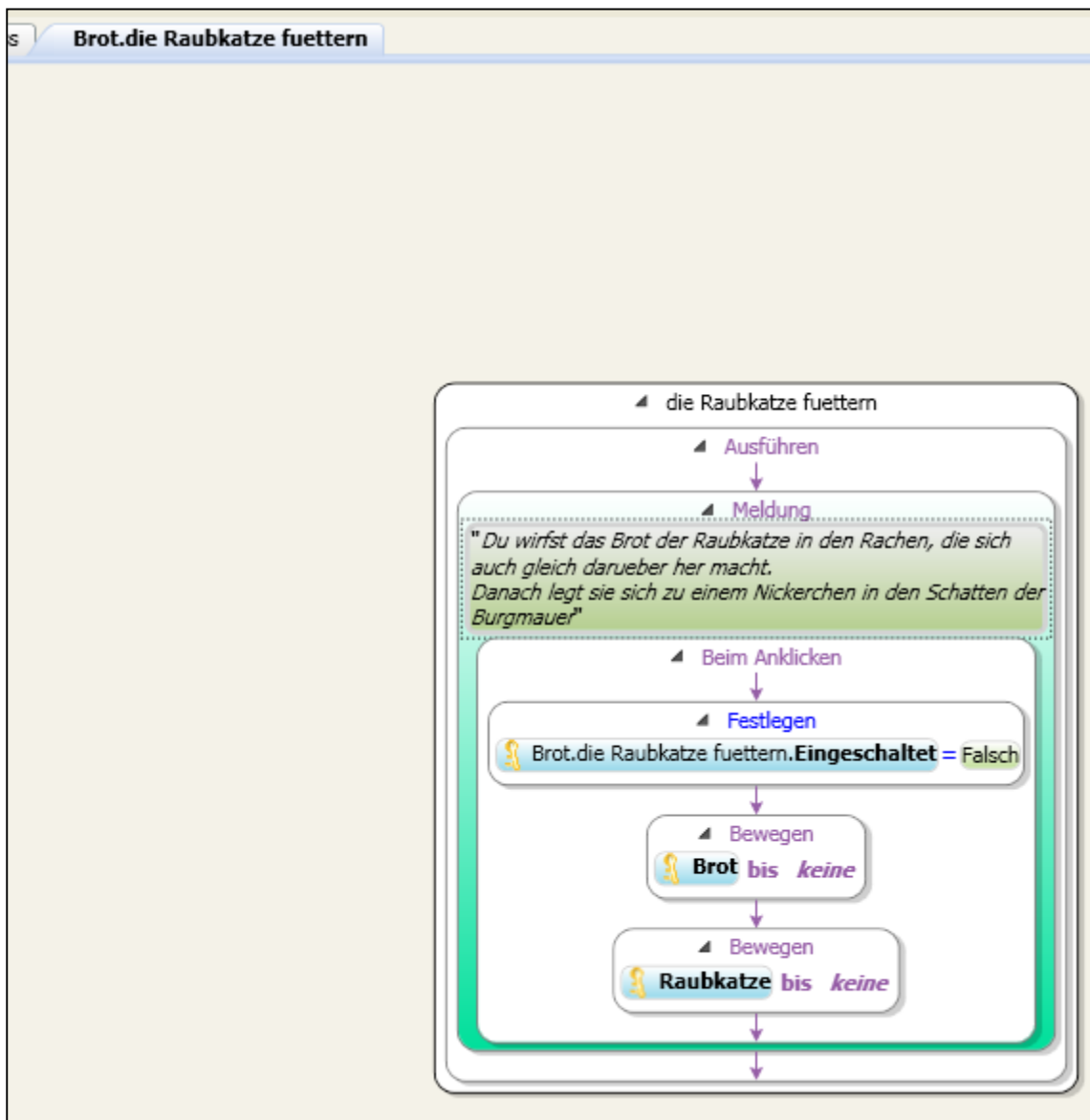
Neue Objekte

Wir legen 2 neue Gegenstände an

- beide erhalten einen zusätzlichen Befehl namens „info“, der ein bisschen Prosa enthält
- für das Tor (Gegenstand) wird als Position die Zone namens Tor gesetzt
- das „gesperrt“-Flag wird für das Tor (Gegenstand) aktiviert
- die Raubkatze erhält keine Position und bleibt erst mal im elektronischen Nirwana

(Die beiden Objekte oben enthalten noch die Befehle „streicheln“ bzw. „Sesam öffne Dich!“, aber das sind nur ein paar Späßchen, auf die ich hier nicht eingehen werde – sie sind für den Ablauf nicht zwingend notwendig).

Jetzt erhalten der Stab und das Brot noch jeweils zwei individuelle Befehle. **Beide Befehle werden aber auf inaktiv gesetzt**, da es keinen Sinn macht, wenn diese schon aktiv sind, wenn der Spieler das Tor (Zone) noch gar nicht erreicht hat!

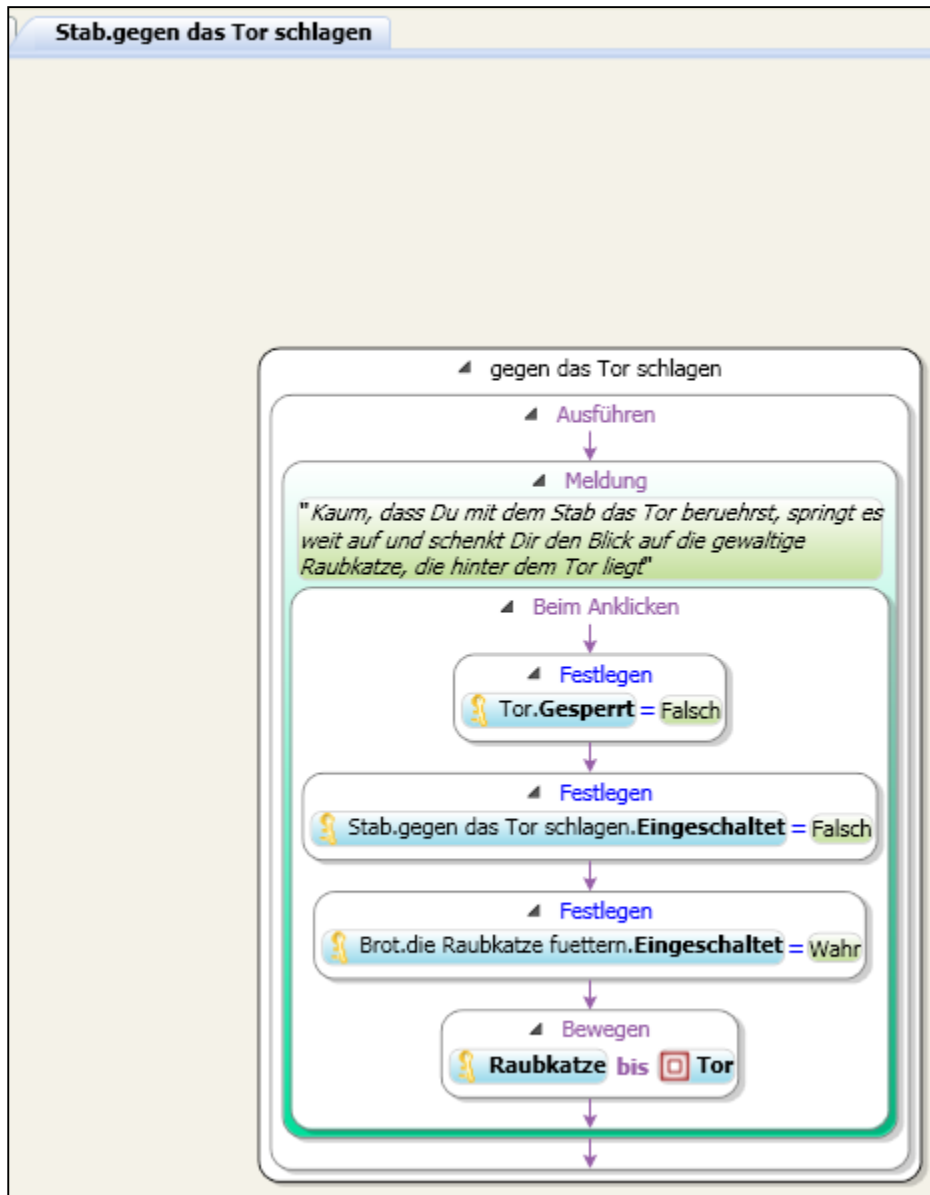


Der Befehl „die Raubkatze fuettern“ für den Gegenstand „Brot“

Beim Brot programmieren wir den, vorerst inaktiven, Befehl „die Raubkatze fuettern“. Dieser enthält ein wenig Text. Nach Aufruf wird der Befehl deaktiviert, da das dann ja erledigt ist. Das Brot wird aus dem Inventar gelöscht und der Löwe verschwindet ebenfalls in die elektronischen Jagdgründe!

So werden wir uns später den Löwen aus dem Weg schaffen, der in der Zone Tor auf uns warten wird.

Für den Stab sieht es ähnlich aus:



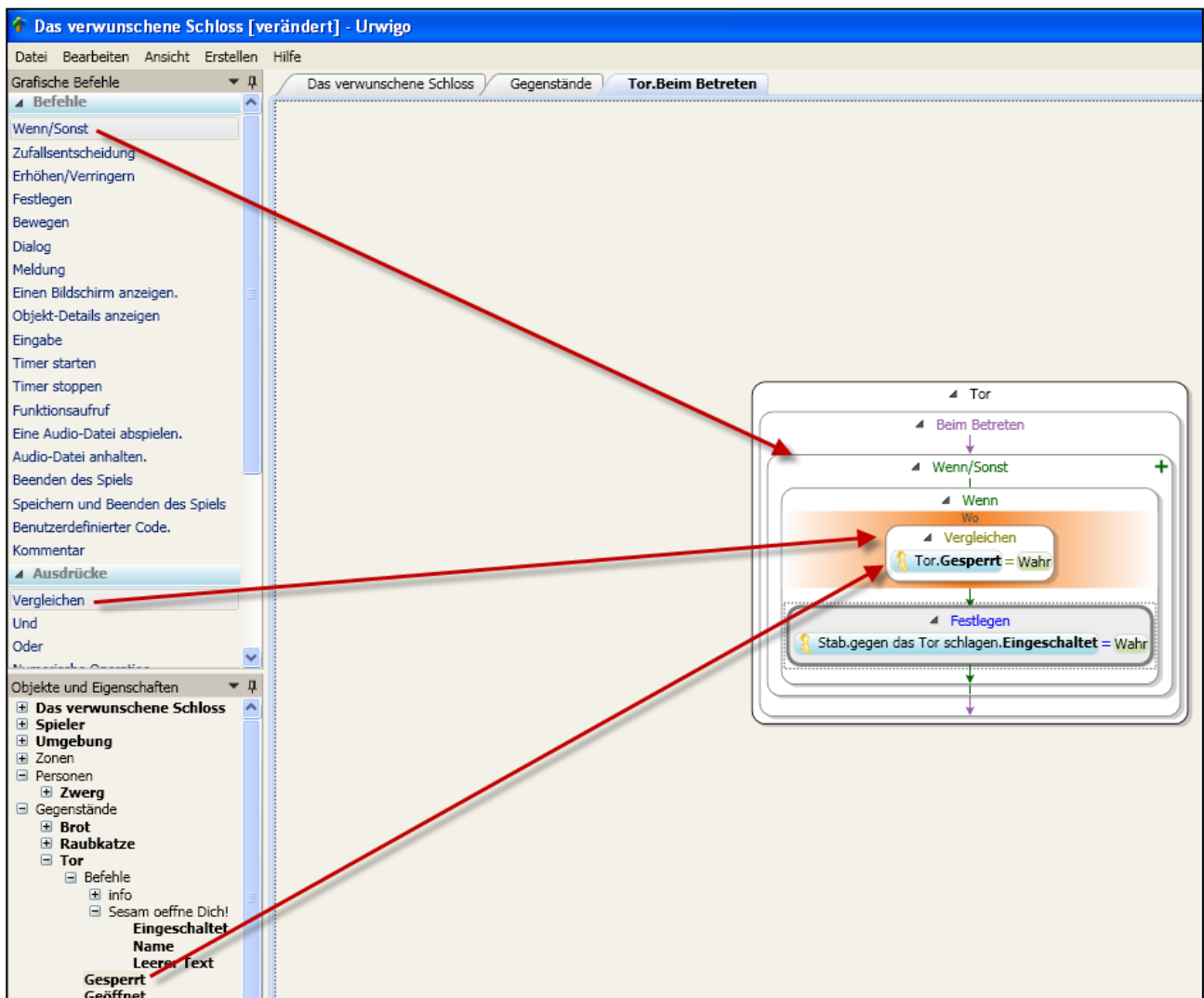
Befehl „gegen das Tor schlagen“ für den Gegenstand „Stab“

Auch hier ist der Befehl „gegen das Tor schlagen“ zunächst inaktiv, da er ja erst Sinn macht, wenn man das Tor (Zone) erreicht. Wir werden ihn erst aktivieren, wenn die Zone namens „Tor“ betreten wird.

Wird der Befehl aufgerufen passiert folgendes:

- Es wird Text angezeigt, daß das Tor (Gegenstand) offen ist und ein Löwe zu sehen ist
- Das Attribut Tor.Gesperrt wird auf „Falsch“ geändert, denn jetzt ist das Tor (Gegenstand) ja offen
- Der Befehl „gegen das Tor schlagen“ wird wieder ausgeschaltet
- Der Befehl „die Raubkatze fuettern“ für das Brot wird aktiviert, da er erst jetzt Sinn macht
- Die Raubkatze wird in die Zone namens Tor bewegt. Da die Zone Tor zu diesem Zeitpunkt aktiv sein wird, ist die Raubkatze dann für uns auf dem Hauptbildschirm sichtbar.

Damit nun alles so ablaufen kann, wie wir uns das vorstellen, muß noch im geeigneten Augenblick der Befehl „gegen das Tor schlagen“ aktiviert werden. Das machen wir, wenn der Spieler das Tor erreicht. Daher hinterlegen wir den notwendigen Befehl im Ereignis „Beim Betreten“ für die Zone Tor.



Ausprägung des Befehls „Beim Betreten“ für die Zone Tor.

Diese vermeintlich einfache Option gilt es aber abzusichern! Es könnte ja sein, daß der Spieler mehrfach die Zone betritt. Sei es wegen schlechtem GPS-Empfang oder weil er nochmal ein Stück zurück gelaufen ist. Um an dieser Ecke kein Fehlverhalten zu provozieren, prüfen wir, ob das Tor (Gegenstand) noch gesperrt ist, wenn wir die Zone betreten. Nur dann wird beim Stab die Funktion „gegen das Tor schlagen“ frei geschaltet.

Dies programmieren wir wie folgt:

- Wir ziehen einen Wenn/Sonst-Befehl in den Ablauf.
- In den Wenn/Sonst-Befehl ziehen wir den Ausdruck „Vergleichen“
- Auf die linke Seite des Vergleichen-Ausdrucks ziehen wir das Attribut „Gesperrt“ des Gegenstandes Tor. Die rechte Seite ändern wir in „Wahr“
- Damit ist die Bedingung festgelegt. Nur wenn das zutrifft (also WAHR ist), schalten wir über einen „Festlegen“-Befehl die entsprechende Funktion ein.

- Einen Sonst-Zweig gibt es diesmal nicht. In diesem Fall wäre ja „sonst“ mit „mach' nix“ zu belegen
😊😊😊

Damit haben wir verhindert, daß beim nochmaligen Betreten, der Stab nicht nochmal die Funktion „gegen das Tor schlagen“ erhält, wenn das Tor (Gegenstand) schon längst geöffnet wurde.

Hier sieht man nochmal schön, daß die Bedingung innerhalb eines „Wenn/Sonst“-Befehls auf „Wahr“ oder „Falsch“ reduzierbar sein muß.

Jetzt: Speichern!

Ausprobieren

Jetzt können wir das Testtool anwerfen und probieren, ob alles so funktioniert, wie wir uns das gedacht haben. Dabei sollte man auch gleich mal das eine oder andere Fehlverhalten ausprobieren!

Das Verlassen der Zone

Jetzt müssen wir noch prüfen, ob der Wherigo wie gewünscht weiter laufen kann, wenn die Zone verlassen wird. Wenn alles korrekt durchlaufen ist, gelten folgende Kriterien:

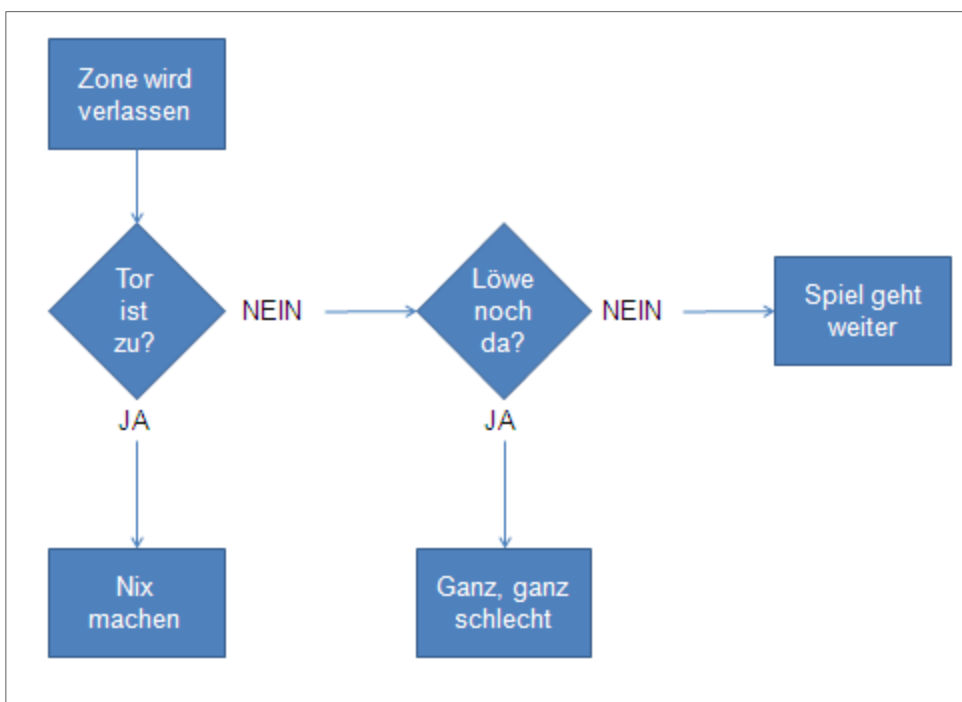
- das Tor (Gegenstand) ist offen
- der Löwe ist weg
- das Brot ist weg

Wir steuern das Verhalten des Wherigos über folgende Prüfungen

- ist beim Verlassen das Tor (Gegenstand) noch zu, passiert nichts. Wir gehen in diesem Fall davon aus, daß der Spieler aus irgend einem Grund nochmal zurück geht (statt weiter in die Burg)
- ist das Tor (Gegenstand) offen, wenn die Zone verlassen wird müssen wir folgende Prüfungen durchführen
 - ist der Löwe und/oder das Brot noch da, endet das Spiel, weil der Spieler gefressen wird
 - ist der Löwe besänftigt worden, kann es weiter gehen. Wir deaktivieren die Zone „Tor“ und aktivieren die Zone in der Kapelle. Außerdem können wir die Aufgabe „gelange in die Burg“ ebenfalls deaktivieren.

Wollte man das aufzeichnen, sähe das so aus:

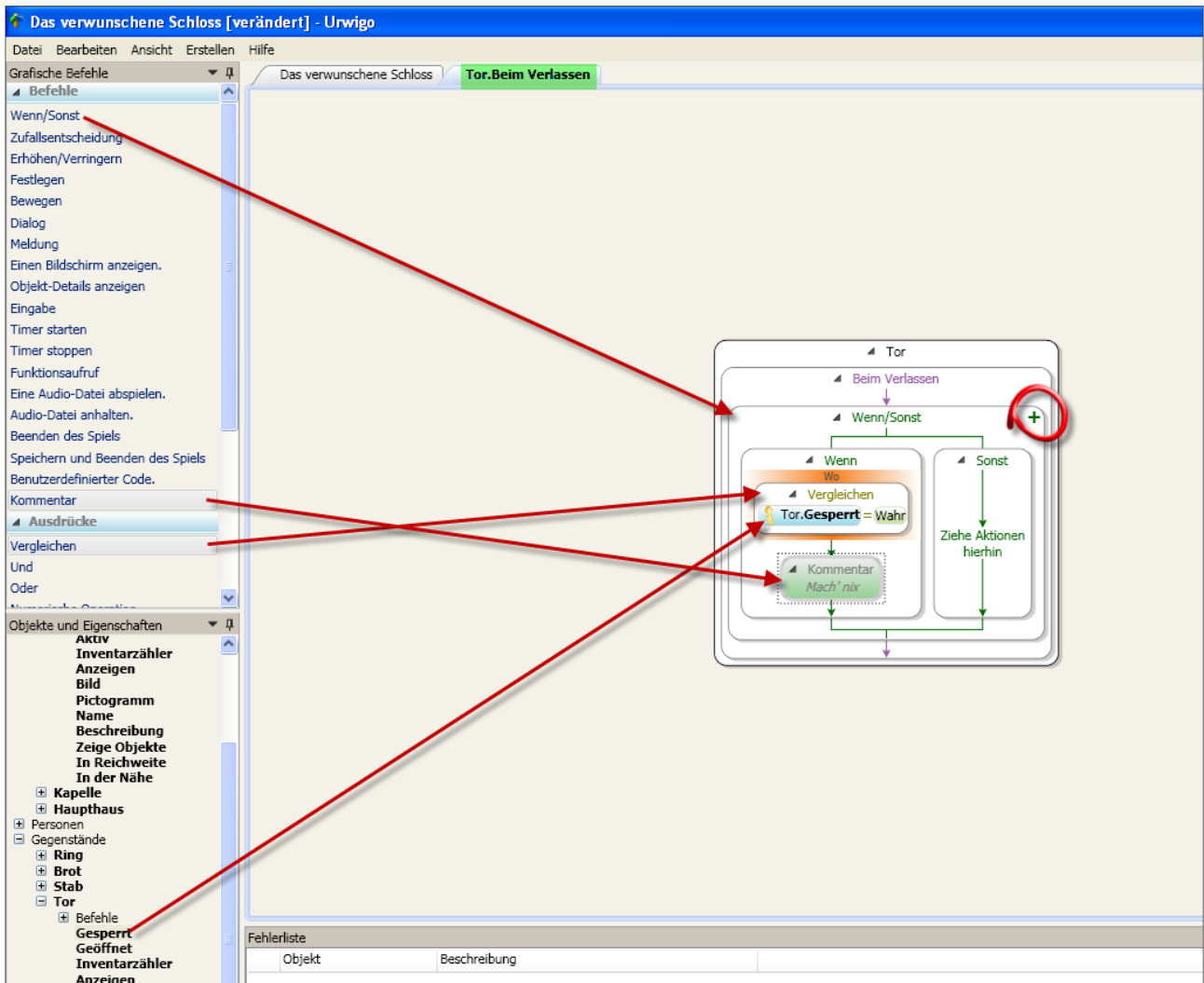
Die Rechtecke stellen Aktionen dar, die Rauten die wenn/sonst-Befehle:



Der Ablauf in grafischer Form

Diese etwas komplexere Abfrage wollen wir jetzt realisieren.

Schritt 1:



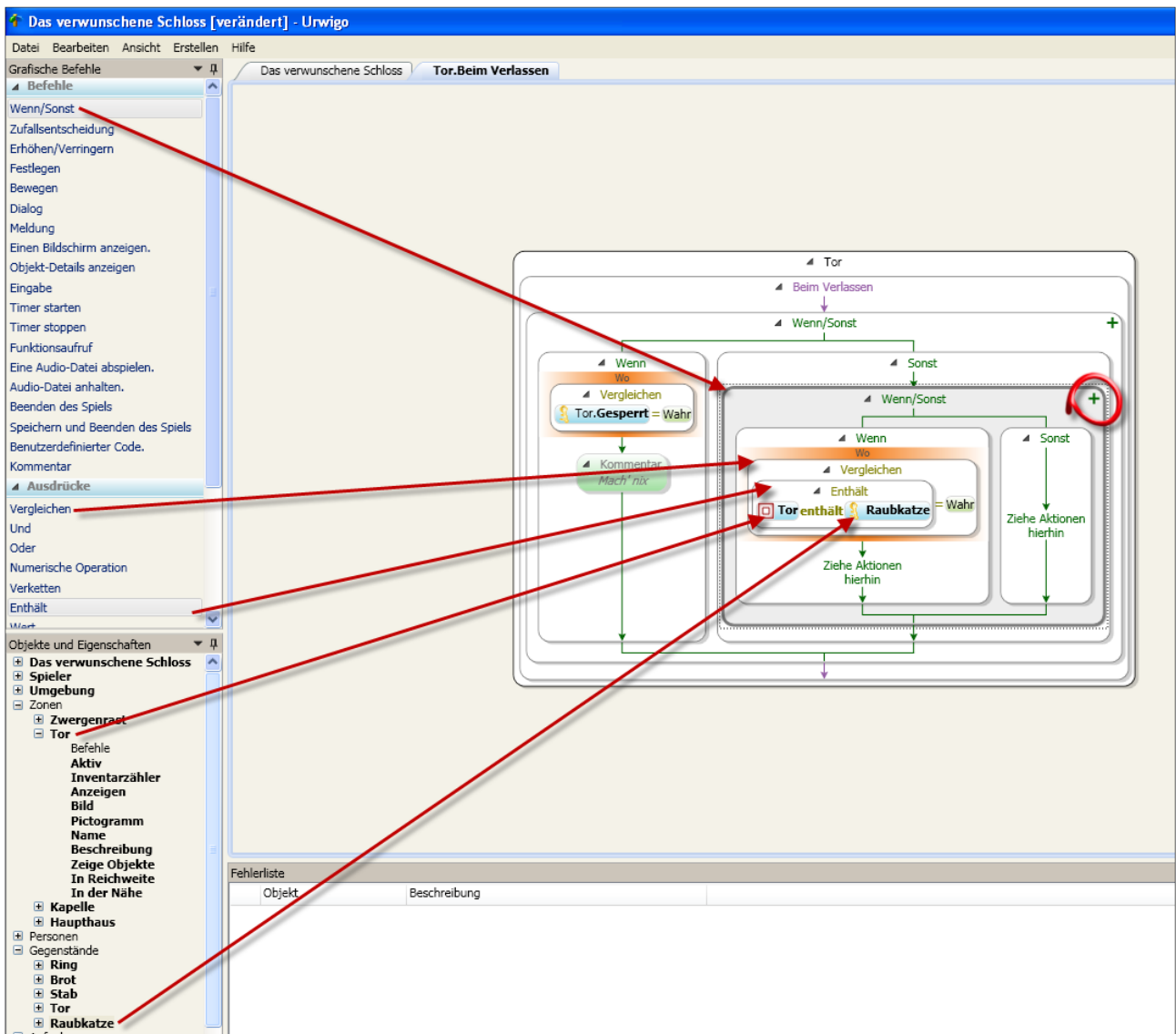
Abfrage komplexer Bedingungen – erster Schritt

Fangen wir mit der ersten Bedingung an:

- Wir gehen in das Ereignis „Beim Verlassen“ der Zone „Tor“ (vgl. Reiter (grün markiert)). Das ist auch wieder eine der vorgefertigten „Hülsen“ analog zu „Beim Betreten“.
- Wir ziehen in den Programmablauf den Befehl „Wenn/Sonst“ aus dem Befehlsfenster herüber
- Über das „Plus“ klappen wir auch den Sonst-Zweig auf (damit es so aussieht wie oben)
- Aus den Ausdrücken ziehen wir den Befehl „Vergleichen“ in den Wenn-Zweig, der Wenn/Sonst-Prüfung
- In die linke Hälfte des Vergleichen-Ausdrucks ziehen wir das Attribut „Gesperrt“ des Gegenstandes „Tor“
- Da nichts passieren soll, wenn die Zone verlassen wird und das Tor noch gesperrt ist, fügen wir hier nur einen Kommentar ein, der nur hier im Editor sichtbar ist. Im Player passiert in diesem Falle nichts.

Damit haben wir die erste Bedingung erledigt: Wenn wir die Zone „Tor“ verlassen und der Gegenstand „Tor“ noch zu ist, geschieht nichts weiter.

Wird die Zone „Tor“ verlassen und der Gegenstand „Tor“ ist nicht mehr gesperrt, müssen wir eine 2. Wenn/Sonst-Bedingung einfügen. Dies geschieht wie folgt:

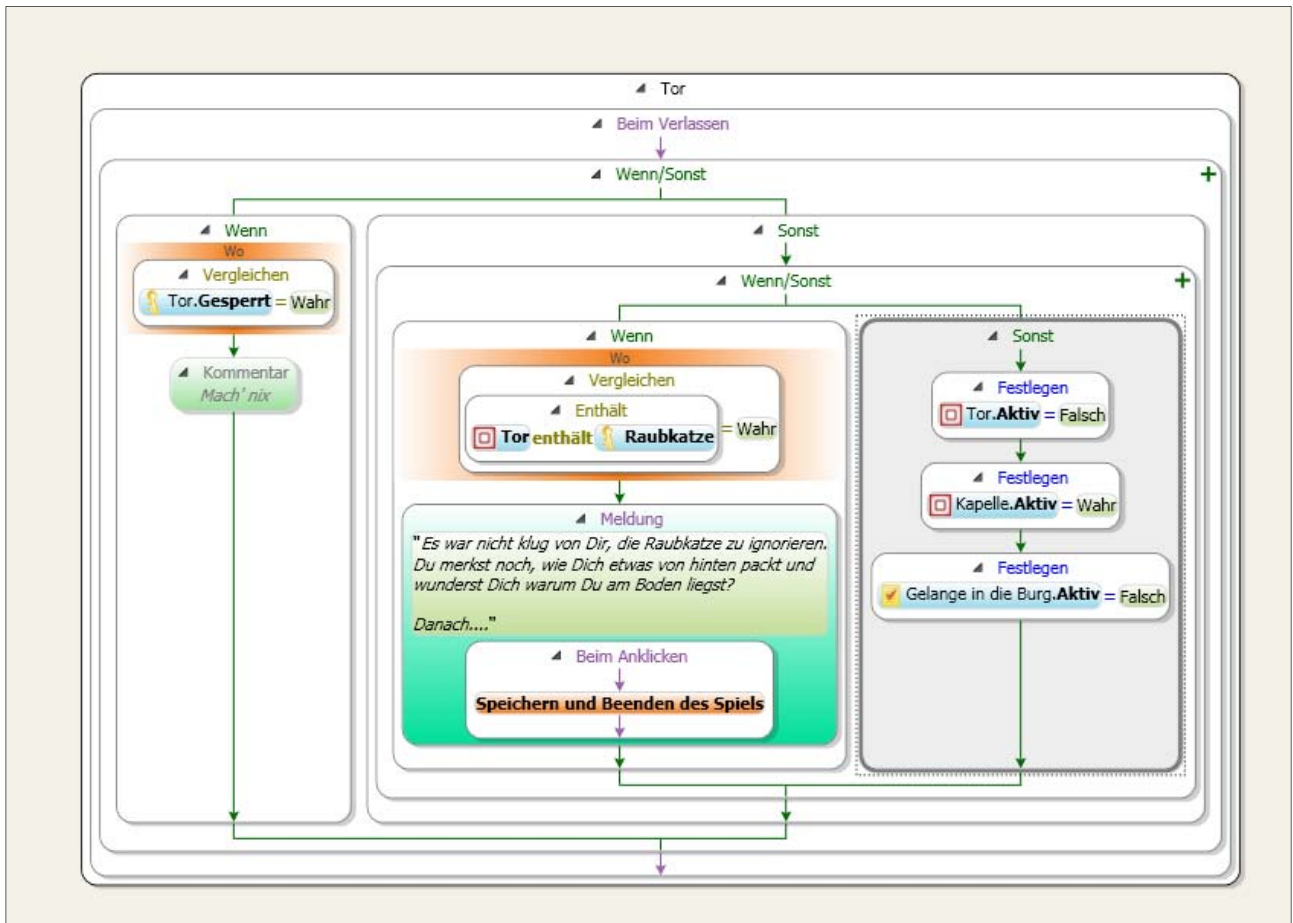


Verschachtel von Bedingungen – 2. Schritt

- Wir nehmen einen weiteren Wenn/Sonst-Befehl aus dem Vorrat und ziehen ihn in den Sonst-Zweig der ersten Prüfung. Anschließend zeigen wir auch für diesen Wenn/Sonst-Befehl über das Pluszeichen den Sonst-Zweig an. Wir haben jetzt also 2 Wenn/Sonst-Bedingungen verschachtelt: Ist die erste Wenn/Sonst-Prüfung nicht WAHR, wird über deren Sonst-Zweig eine zweite Wenn/Sonst-Prüfung durchgeführt
- Jetzt ziehen wir wieder einen „Vergleichen“-Befehl in den Wenn-Zweig der neuen Wenn/Sonst-Abfrage. Den rechten Teil der Bedingung setzen wir auf „WAHR“
- Auf die linke Seite des „Vergleichen“-Befehls ziehen wir den Ausdruck „Enthält“. Den Ausdruck „Enthält“ befüllen wir auf der linken Seite mit der Zone Tor und auf der rechten Seite mit dem Gegenstand „Raubkatze“. Damit prüfen wir also, ob es WAHR ist, daß der Löwe noch in der Zone

Tor „herumsteht“. (Die Prüfung in der Bedingung ist also wieder auf wahr/falsch bzw. ja/nein reduzierbar).

Damit ist die 2. Prüfung realisiert: Wenn das Tor offen ist, wird geprüft, ob der Löwe noch da ist. Im Idealfall ist er verschwunden, weil wir ihn mit dem Brot gefüttert haben. Ist der Löwe noch in der Zone, geht das leider nicht gut für uns aus. Im anderen Fall ist alles gut und wir können die Zone verlassen und weiter spielen!



Fertig ausgeprägte Bedingung mit verschachtelten wenn/sonst-Befehlen

Dieser letzte Schritt ist oben abgebildet:

- Ist das Tor(Gegenstand) offen und der Löwe noch da, wenn wir die Zone Tor verlassen, frisst uns der Löwe. Es kommt eine kurze Meldung und danach ist das Spiel zu Ende. Dafür haben wir eine Meldung und die „Beenden“-Option eingebaut.
- Wenn alles passt deaktivieren wir die Zone „Tor“ und die Aufgabe „Gelange in die Burg“ und aktivieren die Zone „Kapelle“. Der Spieler bekommt dann als aktive Zone die Kapelle angezeigt und kann diese anlaufen.

Jetzt: Sichern!!!

Wieder ein Test

Auch jetzt sollte man eine kleine Test-Session einlegen

Potentieller Fehler in diesem Konstrukt:

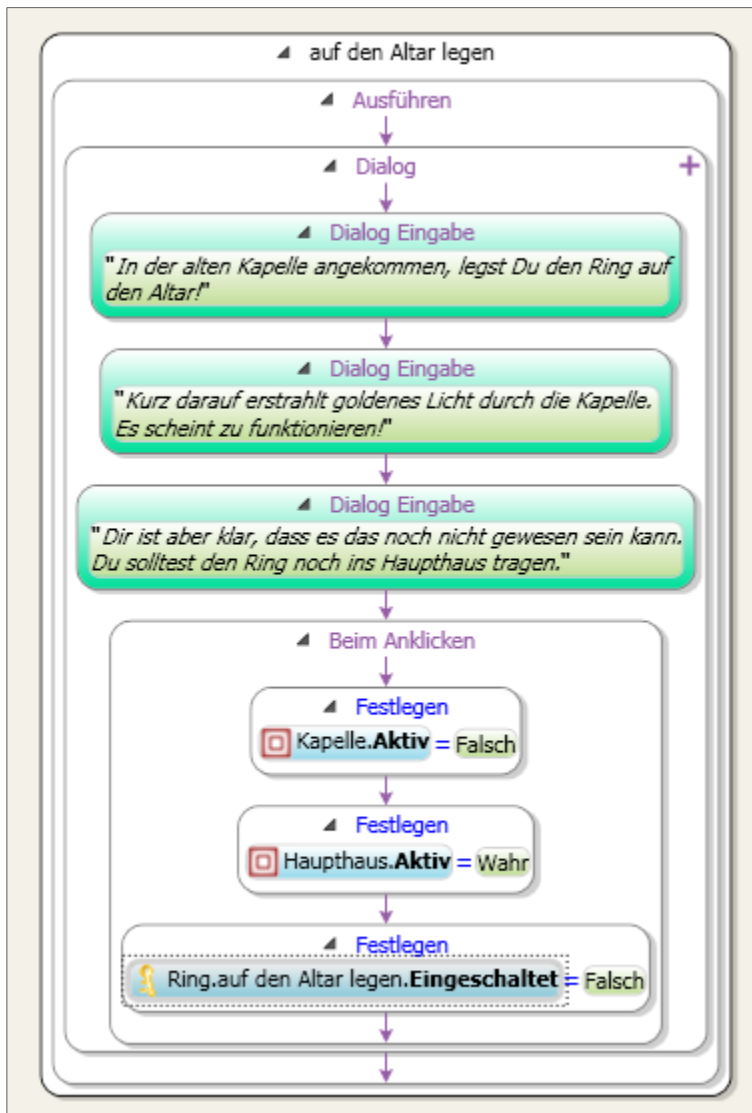
Was hier nicht abgefangen wurde: Wenn das Tor geöffnet wurde und Spieler aufgrund von schlechtem GPS-Empfang die Zone „verlässt“, zerreit ihn Lwe. Diese Mglichkeit wollen wir im Rahmen des Tutorials vernachlssigen und von einem 100%ig genauen GPS-Empfang ausgehen ☺

Vorstellbar wre z. B. ein weitere, unsichtbare Zone in Richtung Kapelle und da der Lwe erst zuschlgt, wenn man diese, fr den Spieler unsichtbare Zone betritt.

Die Kapelle

Der nächste Schritt ist wieder etwas einfacher: Der Ring soll ja in die Kapelle gebracht werden, wo er sozusagen „aktiviert“ wird. Das bedeutet für uns, wir verpassen dem Ring einen weiteren inaktiven Befehl, den wir erst aktivieren, wenn er benötigt wird: Beim Betreten der Kapellen-Zone.




Zunächst der Befehl: Wir nennen ihn „auf den Altar legen“:



Der neue Befehl für den Ring (noch nicht aktiv)

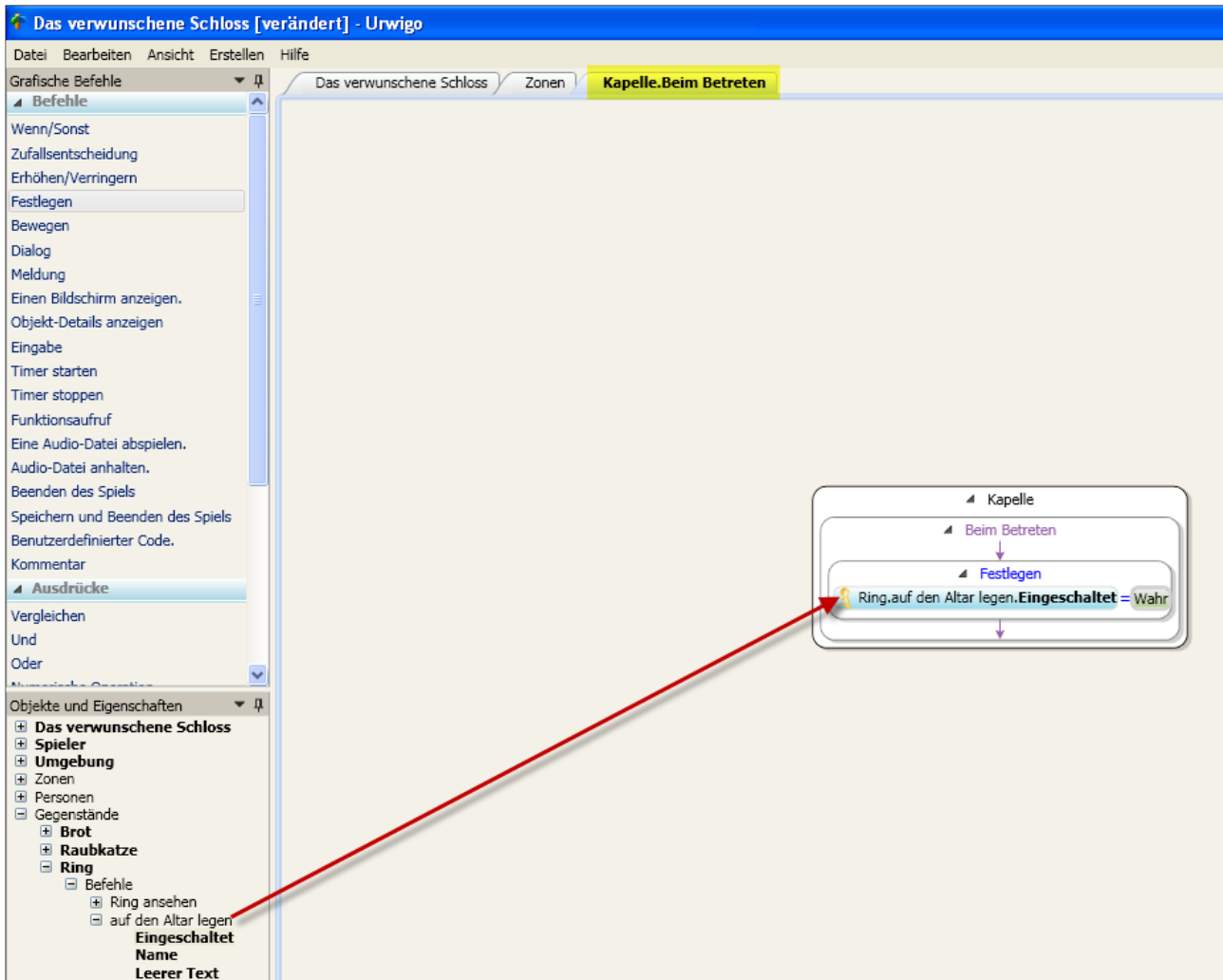
Dieser Befehl enthält nicht sehr viel: Einen auf drei „Blasen“ aufgebohrten Dialog, um den Text etwas zu entzerren. Zudem wird nach Abschluß des Dialogs die Kapellen-Zone wieder deaktiviert und die Haupthaus-Zone aktiviert. Und den Befehl benötigen wir jetzt auch nicht mehr, daher deaktivieren wir diesen ebenfalls gleich wieder.

Damit wir den Befehl, der ja momentan noch inaktiv ist, zur rechten Zeit zur Verfügung haben, müssen wir ihn zur gegebene Zeit aktivieren: Beim Betreten der Kapellen-Zone. Dafür gibt es ja in den Zonen, das bekannte Standard-Ereignis, das wir jetzt ausprägen:

Position	
Eckpunkte	50.0277725204245N 9.7965
	50.0276036568603N 9.7965
	50.0275243941662N 9.7968
	50.0276967042038N 9.7969
Auf Karte ein	
Befehle	
 Gegenstand eingeben  Neuer Gegenstand 	
<div></div>	
Ereignisse	
Beim Betreten	<u>unbehandelt</u>
Beim Verlassen	<u>unbehandelt</u>
Entfernt	<u>unbehandelt</u>

„Beim Betreten“ für die Zone „Kapelle“

Das ist dann mittlerweile einfach genug:



Ausprägung von Kapelle.Beim Betreten

Wir benötigen nur einen „Festlegen“-Befehl. In diesen ziehen wir aus den „Objekten und Eigenschaften“ das Attribut „Eingeschaltet“ für den Befehl „auf den Altar legen“ den wir beim Gegenstand Ring erzeugt haben und setzen das Attribut auf WAHR.

(O.k. das war jetzt sehr umständlich formuliert, ich wollte nur mal wieder auf die Objekt-Zusammenhänge hinweisen ☺ - macht es so wie auf dem Bild).

Sichern nicht vergessen!

Und das man das jetzt noch über das Testtool ausprobiert, muß ich ja nicht mehr explizit erwähnen, oder?

☺

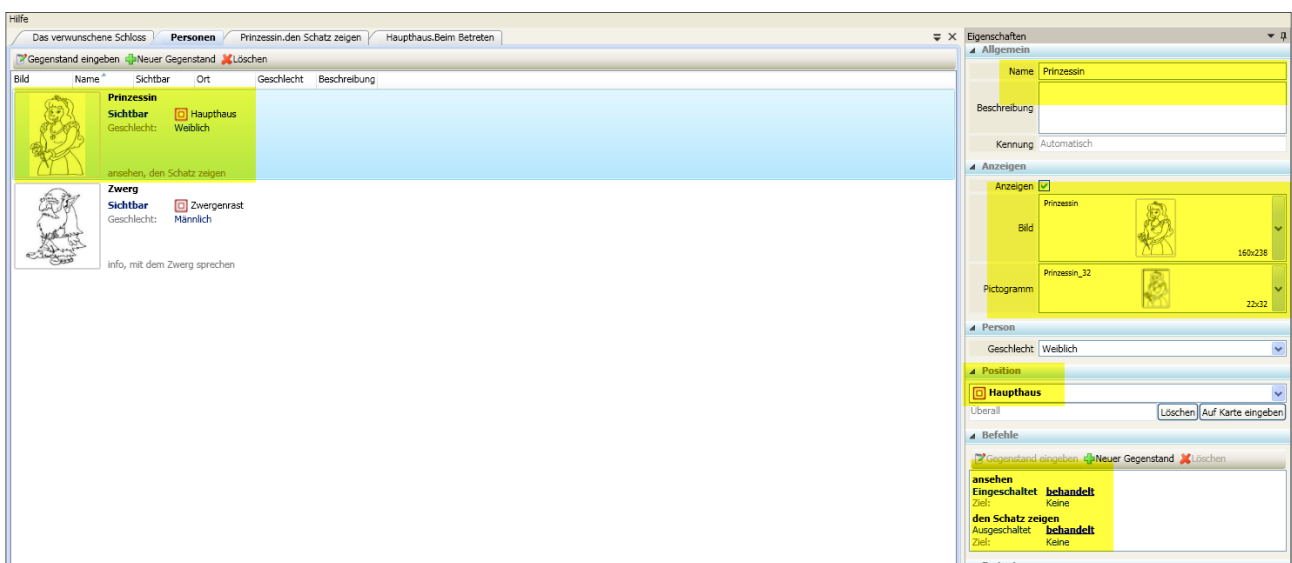
Der letzte Akt

Jetzt haben wir es fast geschafft. Wir müssen jetzt noch das Szenario zu Ende bringen und die Prinzessin auftreten lassen. Für dieses Beispiel wollen wir es so machen, daß man der „versteinerten“ Prinzessin den Ring an den Finger stecken muß, um die Aufgabe abzuschließen.

Um es nicht zu einfach zu machen, „verstecken“ wir den notwendigen Befehl beim Ring und nicht bei der Prinzessin. Wir benötigen also:

- eine Person namens Prinzessin – die wir im Hauptgebäude „abstellen“
- einen Befehl, der den Zustand der Prinzessin beschreibt
- einen noch nicht aktiven Befehl bei der Prinzessin, mit der sie einen Schatz zeigt (GC-Koordinaten und Ende des Wherigos.
- einen zusätzlichen inaktiven Befehl beim Ring, mit dem die Prinzessin erlöst wird
- Programm-Code im Ereignis „beim Betreten“ der Zone Haupthaus. Damit wird ein kurzer Text eingeblendet und der noch inaktive Befehl beim Ring aktiviert.

Fangen wir mit der Prinzessin an:

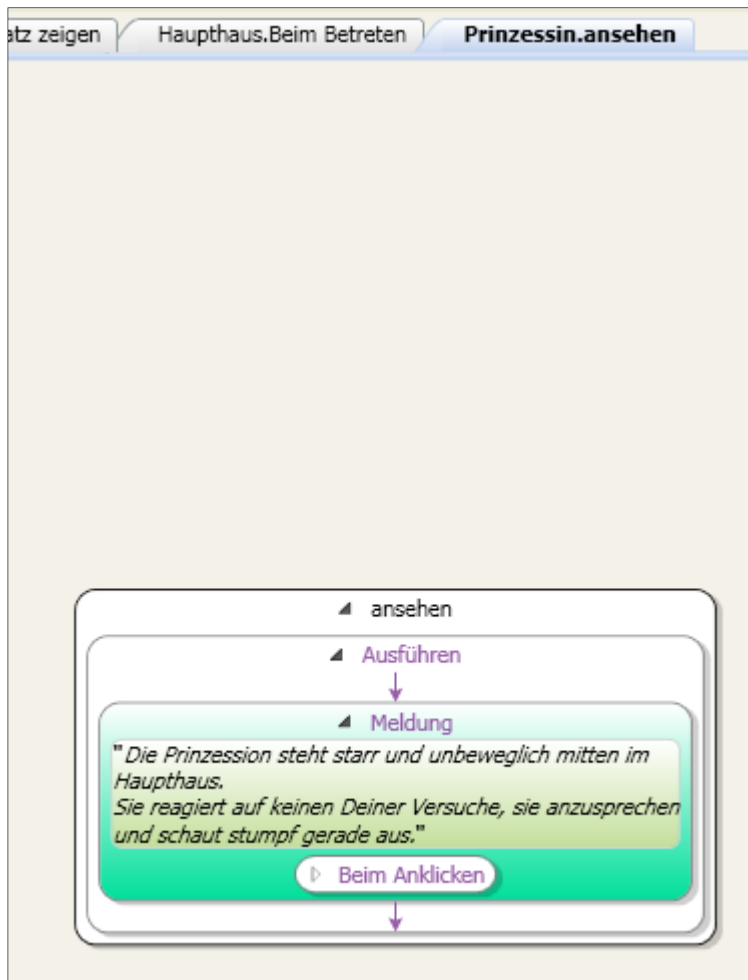


Eine neue Person: die Prinzessin.

So wie oben angezeigt wurde die Prinzessin angelegt. Wie das im Detail funktioniert muß jetzt wohl nicht mehr erklärt werden. Wichtig ist, daß sie in der Zone Haupthaus positioniert wird und die beiden Befehle „ansehen“ und „den Schatz zeigen“ erhält (im o. a. Bild sind diese schon ausgeprägt und daher fett markiert), bei uns sind diese im Moment noch unbehandelt.

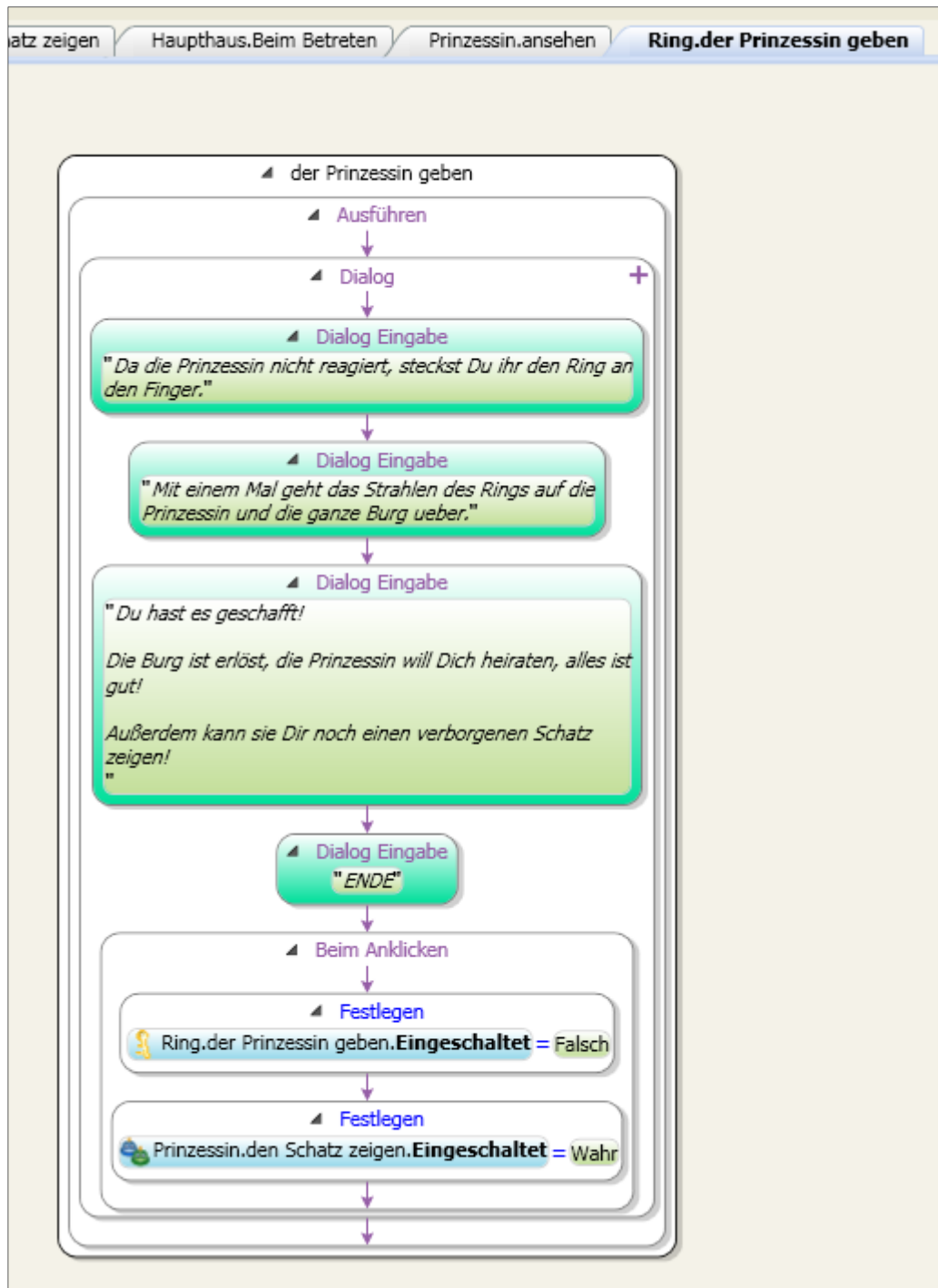
Der Befehl „den Schatz zeigen“ muß man jetzt erst mal inaktiv lassen.

Den Befehl „ansehen“ der Person Prinzessin prägen wir auch noch schnell aus:



Etwas Prosa beim Ansehen der Prinzessin

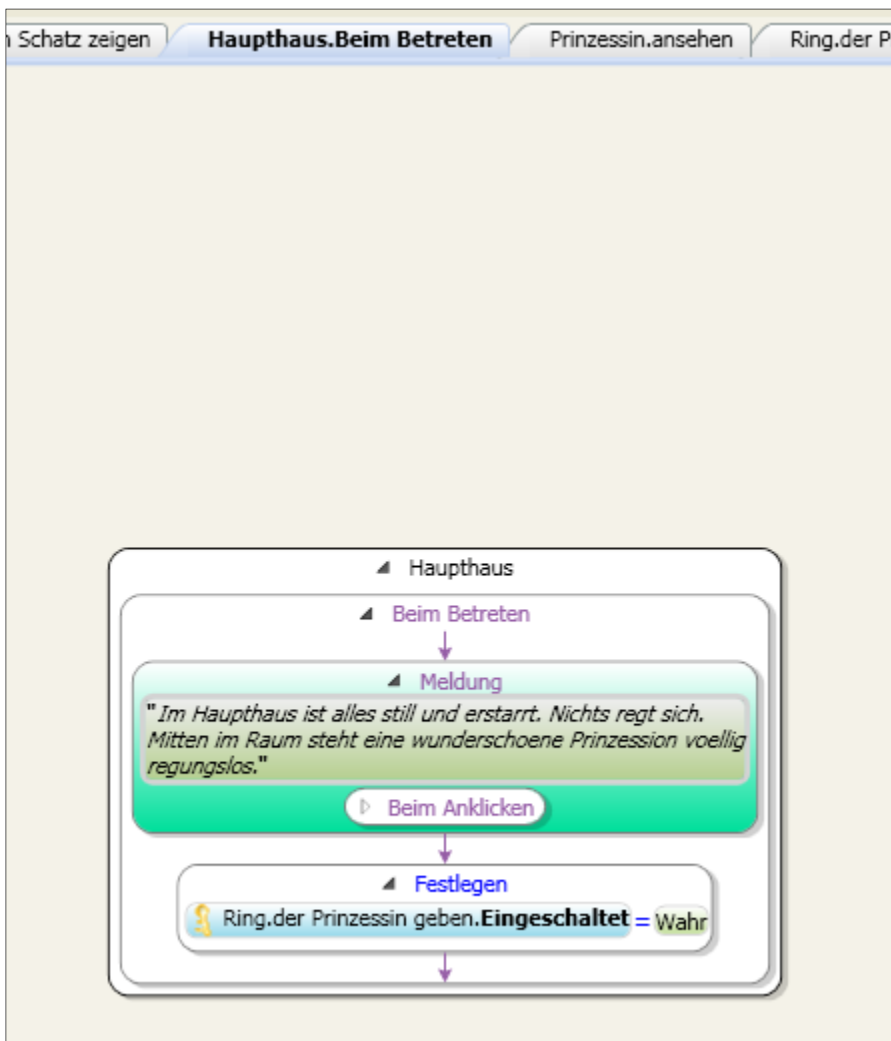
Jetzt bekommt der Ring noch eine neue Methode – wie gesagt vorerst inaktiv:



Die Ausprägung des Befehls „der Prinzessin geben“

Hier passiert eigentlich nicht viel. Der Spieler wird ein bisschen zugetextet und anschließend wird der Befehl gleich wieder deaktiviert. Dafür wird der Befehl „den Schatz zeigen“ aktiviert.

Damit dies alles so funktionieren kann, müssen wir den neuen Befehl beim Ring aktivieren, wenn der Spieler die Zone Haupthaus betritt:

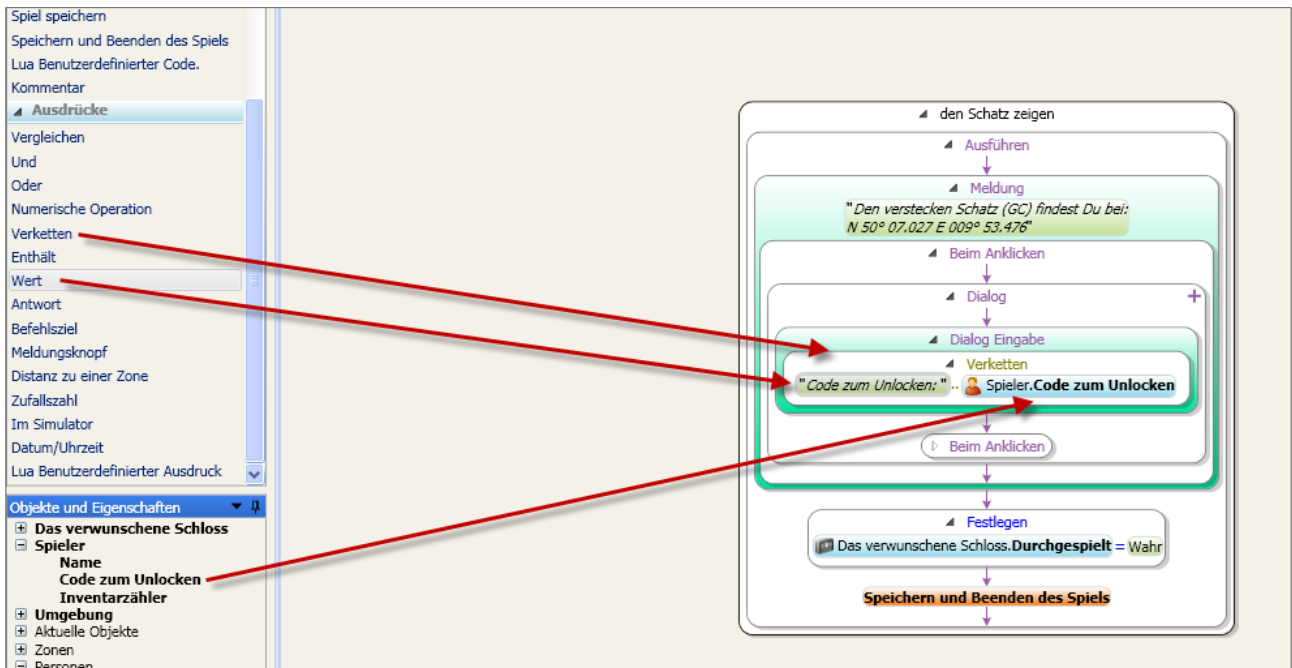


Aktivierung notwendiger Befehle beim Betreten der letzten Zone

Hier hinterlegen wir nur etwas Text zur Stimmungsmache und schalten über den „Festlegen“-Befehl die neue Option beim Ring ein. Somit steht diese dann in der letzten Zone zur Verfügung!

Zwischendurch: Sichern nicht vergessen!

Das war's dann fast. Den Befehl „den Schatz zeigen“ müssen wir aber noch mit Leben füllen:



Die Informationen zum Schatzzzzz!

Hier hinterlegen wir die Koordinaten des Finals und geben noch den Code zum Unlocken der Cartridge frei. Dieser ist dazu da, um auf wherigo.com den Wherigo als durchgespielt zu markieren. Alternativ dazu kann man auch eine gws-Datei vom Player hochladen. (Viele verzichten aber darauf und loggen nur den Cache. Wie Ihr das handhabt bleibt Euch überlassen.)

In dem Befehl oben zeigen wir auf jeden Fall den Code zum Unlocken noch an und setzen das Attribut „durchgespielt“ auf wahr.

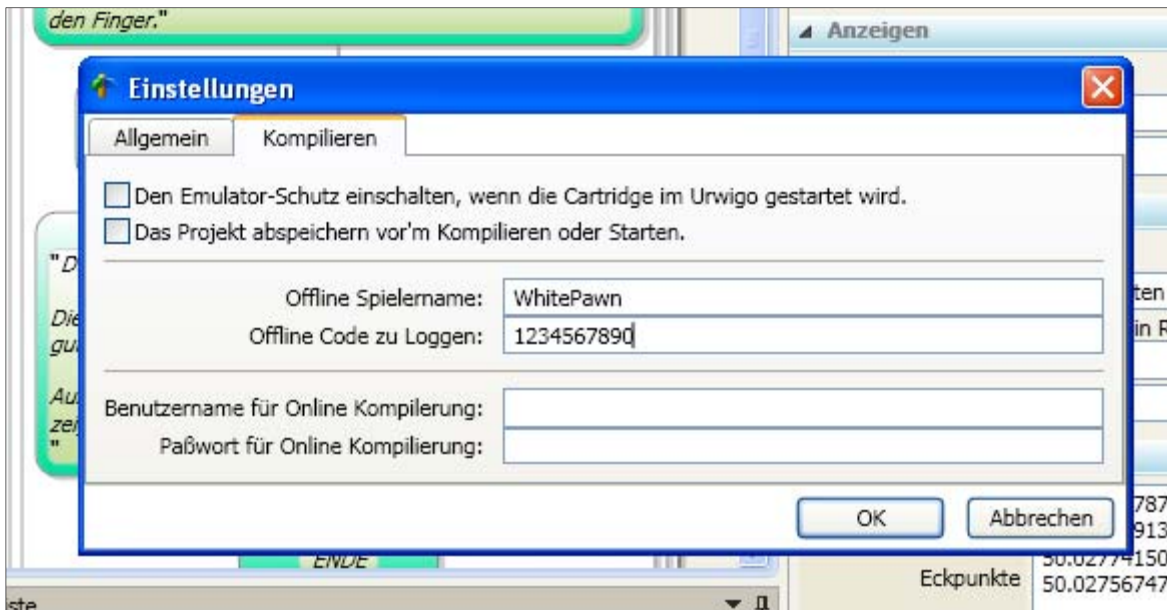
Hierzu benutzen wir einen neuen Befehl: verketten

Dieser Befehl verknüpft mehrere „Stücke“ zu einem Ganzen.

- Wir ziehen den verketten-Befehl in den Dialog
- Danach einen „Wert“ in den Verketteten Befehl und
- Dem Code zum Unlocken aus dem Spieler-Objekt
- Den Wert definierten wir über das Eigenschaftsfenster als Text und hinterlegen „Code zum Unlocken: “. (Nach dem Doppelpunkt ein Leerzeichen einfügen, sonst klebt alles aneinander ☺)

Zum Schluß noch Speichern und Beenden, damit es eine gws-Datei gibt, die hochgeladen werden kann.

Den Code zum Unlocken kann im Menü „Datei“ unter dem Unterpunkt „Einstellungen“ hinterlegen.

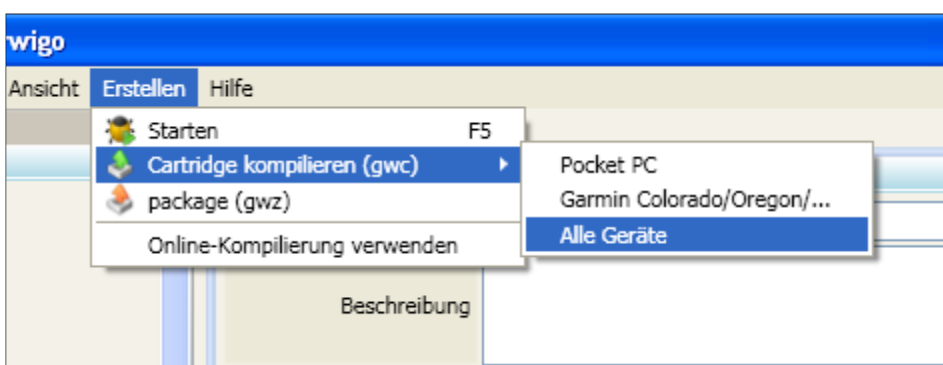


Der Unlock-Code im Editor

WICHTIG: Der hier hinterlegte Code ist nur zum Testen, wenn man direkt kompiliert und ausprobiert, damit man sehen kann, ob es richtig funktioniert. Für die Spieler, die später die Cartridge bei wherigo.com herunterladen, wird jedes Mal ein individueller Code generiert!

Tests vor Ort

Natürlich sollte man sein Machwerk auch vor Ort ausgiebig testen – wie erwähnt, am besten mit verschiedenen Geräten.



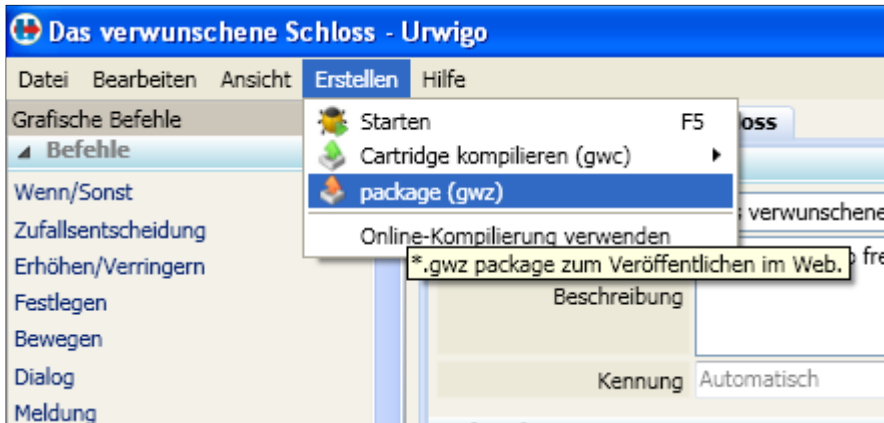
Eine testfähige Cartridge erstellen

Über das Menü „Erstellen“ -> „Cartridge kompilieren“ kann nun eine Cartridge generiert werden, d. h. es wird eine Datei erzeugt, die als Programm in diversen Playern abgespielt werden kann.

Diese kann für Tests vor Ort verwendet werden.

Hochladen zu Wherigo.com

Hat man die Cartridge ausgiebig getestet (auch durch Dritte), kann man sie bei wherigo.com hochladen. Dazu benötigt man ein gwz-package



Erstellen eines Packages für den Upload.

Dieses erstellen wir aus dem gleichen Menü, wie die gwc-Datei zum Testen. Dieses gwz-File enthält alles Notwendige für den Upload nach wherigo.com. Aus dieser Datei generiert wherigo.com die Download-Files für die verschiedenen Gerätetypen und versieht die gwc-Datei, die man herunter lädt, mit einem individuellen Freischalt-Code (vgl. weiter oben).

Damit ist unsere Aufgabe erledigt und die Spiele können beginnen 😊

Was nicht mehr realisiert wurde

Wollte man diesen Wherigo live stellen, wäre man natürlich noch längst nicht fertig. Es gäbe noch einiges zu verbessern. Das würde dieses Tutorial aber sprengen. Wer sich bis zum Ende durchgekämpft hat, sollte aber jetzt in der Lage sein, einiges zur Not selbst heraus zu finden. Eigentlich ist ja alles ganz sprechend ☺

Was könnte man jetzt noch verbessern?

- Das Erreichen einer Zone könnte durch Abspielen eines Sound-Files angezeigt werden. Dies ist vor allem bei unsichtbaren Zonen sinnvoll. Benutzer eines GPS müssten natürlich auf diese Option verzichten, da die meisten keine Lautsprecher haben.
- Die Aufgaben könnten intensiver genutzt werden, um den Spieler anzuleiten. Gerade an den Stellen in dieser Cartridge, wo der Spieler erst einmal allein gelassen wird, z. B. wenn er erfolgreich das Tor durchschreitet
- Das Auffinden des realen Caches könnte über eine eigene Zone und Hinweise oder Spoiler-Bilder gesteuert werden. Dann wäre der Spieler nicht gezwungen, die finale GPS-Koordinaten zu notieren
- Es könnten (soweit vorhanden) bei diesem Beispiel noch historische Informationen zur Verfügung gestellt werden, z. B. über einen speziellen Gegenstand oder einen virtuellen Fremdenführer. Infos und Position (im Falle des Fremdenführers) würden über die „Beim Betreten“-Funktionen der Zonen gesteuert
- Man könnte dem Spieler noch einen Objekt ins Inventar legen über das er jederzeit den Spielstand speichern kann. Dies kann durchaus etwas zum Spiel passendes sein, wie z. B. eine magische Schreibfeder
- Alles was mir hier jetzt nicht mehr eingefallen ist und von dem Ihr sagt: „Das wär doch cool, wenn...“. Dies ist ein Programm und auf virtueller Ebene ist (fast) alles möglich.

Ihr seht, dieses Medium bietet erhebliche Möglichkeiten, aber irgendwann muß man einfach mal einen Schlussstrich ziehen und das Ganze zum Abschluss bringen.

Es gibt von der hier vorgestellten Cartridge eine „aufgebohrte“ Version, die ich als GeoCache veröffentlicht habe: [GC3H9GP](#). Die dort verlinkte Cartridge spielt auch auf der Homburg, ist aber wesentlich komplexer. Auch die Handlung ist natürlich abgeändert, damit niemand anhand dieses Tutorials den Cache loggen kann.

Abschließend noch ein paar Tipps und den Rest überlasse ich dann Euch. ☺

Tipps & Tricks

„Feindberührung“

Wie heißt es so schön: „Kein Plan überlebt die erste Schlacht“. So ähnlich ist es mit dem erstellten Programm. Da hier noch viel komplexere Logiken hinterlegt werden können, als bei einem Multi-Cache, sollte man den Wherigo auf jeden Fall durch Dritte testen lassen.

Ideal wäre, wenn man die Möglichkeit hätte, auf mehreren Geräten zu testen, z. B. auf einem GPS und einem Smartphone.

Schon beim ersten Selbsttest findet man meist einige Fehler. Sei es das ein Button auf dem Player anders angezeigt wird, als im Testtool, sei es weil man Kleinigkeiten bemerkt, z. b. daß ein Rücksprung auf den Hauptbildschirm das Ganze noch komfortabler machen würde, etc.

Allerdings kennt man natürlich die Lösung. Daher ist es wichtig, andere testen zu lassen. Nur so bekommt man heraus, ob der Unfug, den man sich ausgedacht hat, für andere nachvollziehbar ist. ☺

„Schweizer Taschenmesser“

Bei komplexen Cartridges, die man nicht für jeden Test erneut von Anfang an durchspielen möchte, kann es Sinn machen, sich ein „Schweizer Taschenmesser“ zu bauen. Das ist ein beliebiger Gegenstand, den man dem Spieler während der Entwicklungsphase von Anfang an ins Inventar packt.

Diesem Gegenstand kann man dann entsprechende Funktionen zuweisen, die das Testszenario dadurch abkürzen, daß sie die notwendigen Zonen aktivieren und das Inventar anpassen. Als Beispiel zum Tutorial könnte man beispielsweise einem Gegenstand den Befehl „geh gleich zur Kapelle“ zuordnen. Dieser Befehl aktiviert dann die Kapellen-Zone und deaktiviert alle anderen und man kann so ein ganzes Stück überspringen, weil man diesen Teil ja schon ausführlich getestet hat.

Oder man ordnet diesem Gegenstand eine „Speichern“-Funktion zu und springt über einen gespeicherten Spielstand an die richtige Stelle. Mit einem „Schweizer Taschenmesser“ kann man aber mehrere Befehle hinterlegen und ggf. mehrere Problempunkte „anspringen“, während eine gespeicherter Spielstand nur einen bestimmten Punkt beinhaltet-

Komplexität

Im Prinzip könnte man mit dem Wherigo auch komplexeste Abenteuer kreieren. Die Frage ist, ob diese jemals durchgespielt würden. Niemand möchte stundenlang an einer Ecke stehen und Optionen durchprobieren. Daher sollte man erst einmal ein kurzes Abenteuer realisieren. Man wird überrascht sein, wie schnell auch kleine Konstrukte von der Komplexität her anwachsen.

Von anderen „klauen“

Im Internet findet man diverse Urwigo-Projekte, die man herunterladen und ansehen kann. Evtl. kann man auch bei Programmierern Anfragen, die Wherigos veröffentlicht haben. Aus fremdem Programmcode kann

man einiges lernen! Wie hat jemand eine bestimmte Aufgabenstellung gelöst? Ist der Ansatz gut oder hat er Nachteile? Nicht nur aus gut erstellten Programmen kann man einiges lernen. Wie heißt es so schön? Niemand ist unnütz, er kann immer noch als schlechtes Beispiel dienen. ☺

Spieler in die Mitte der Zonen locken

Um Fehlfunktionen von vorne herein zu vermeiden, ist es vorteilhaft, wenn die Möglichkeit besteht, den Spieler in die Mitte einer Zone zu locken, weil es etwas abzulesen gilt oder weil an dieser Stelle etwas in der wirklichen Welt gefunden werden muß (z. B. eine tatsächliche Stage), um weiter spielen zu können.

Für rein virtuelle Zonen möchte ich ein besonderes Konstrukt vorstellen: Ich habe meinen Cache mit jeweils 2 Zonen realisiert, die gleiche Namen haben. Das Ganze sieht dann z. B. so aus:



Ereignis-Zone(groß) mit „Zünder-Zone“ (klein)

Um während der Entwicklungsphase nicht wahnsinnig zu werden benenne ich die Zonen erst einmal unterschiedlich: Die kleine Zone z.B. „Tor Start“ und die große Zone ganz normal „Tor“.

Die große Zone ist erst einmal deaktiviert und unsichtbar. Der Spieler bekommt zunächst die kleine Zone angezeigt. Sobald er die kleine Zone betritt, verschwindet diese und die große Zone wird angezeigt. Diese ist ganz normal ausgeprägt – so wie hier im Tutorial gezeigt, nur daß sie eben durch die kleine Zone „gezündet“ wird. Die kleine Zone enthält in der Funktion „beim Betreten“ daher nur ein paar Festlegen-Befehle, welche die kleine Zone verschwinden lassen und die große „herzaubern“

Der Vorteil dieser Vorgehensweise ist, daß der Spieler mitten in einer großen Zone steht, aus der er garantiert nicht mehr „raus fällt“ (oder heraus laufen kann) ☺

Nach dem Abschluss der Tests habe ich die Zonen gleich benannt. Die kleine und die große Zone heißen dann „Tor“ und so fällt dem Spieler in seinem Player nicht mehr auf, daß eigentlich 2 Zonen im Spiel sind.

Schluss jetzt

So, das war's dann aber wirklich. Ich hoffe es hat Spaß gemacht und ein neuer Wherigo geht demnächst online 😊

Viel Spaß wünscht

WhitePawn

Version: 1.01 13.03.2012 diverse Tippfehler korrigiert