„BrabbelRP“

World of Warcraft (WoW) Mod für Rollenspieler/innen

Dokumentation

Friederike Bulka

Song Liang

Björn Teichnmann

Inhaltsverzeichnis

1. Funktionsübersicht

- Kategorien
  erstellen / löschen / bearbeiten und auswählen
- Zeilen/Gebrabbel
  erstellen / bearbeiten / löschen
- unterschiedliche Channels
  Gebrabbel kann aus normalem Text, geschrienem Text oder auch Emotes bestehen
- Timer:
  einstellbarer Zeitabstand zwischen Gebrabbel-Ausgaben
- Chance:
  einstellbare Wahrscheinlichkeit für die Ausgaben
- aktive Kategorien:
  mehrere aktive Kategorie möglich (mit jeweils unterschiedlichen Timer/Wahrscheinlichkeit)
- Standby-Modus
  Pausieren der gesamten Ausgabe des Addons
- GUI-basierte Nutzung
- Mehrsprachigkeit:
  Addon kann in 3 Sprachen verwendet werden (deutsch, englisch, chinesisch)
  Erweiterung problemlos möglich
- Mehrzeiler:
  Gebrabbel kann aus mehreren Zeilen bestehen.
- Modus:
  Jeder Kategorie kann optional ein "Modus" vergeben werden, welcher es der Kategorie ermöglicht direkt auf WoW-Ereignisse (Kampf-Treffer, etc.) zu reagieren. Alternativ kann festgelegt werden, dass die Kategorie nur zu einer bestimmten Zeit/Ort (innerhalb der Stadt, während des Kampfes, etc.) etwas ausgibt.
- Minimap-Icon:
  Zusätzlich zum BrabbelRP-Fenster befindet sich bei der Mini-Karte noch ein verschiebbares Icon um die BrabbelRP Einstellungen zu öffnen
- Standardkategorien:
  Nach dem Installieren des Addons besitzt es je nach Sprache schon vordefiniertes Gebrabbel, welches auch zu einem späteren Zeitpunkt wieder über einen Button hergestellt werden kann.
- viele Tooltips und Gimmicks, um BrabbelRP noch intuitiver und schicker zu machen

2. Verwendete Werkzeuge

Eclipse & CVS
Ermöglicht eine einfache Versionierung der Dateien und erleichtert somit das gemeinsame Arbeiten an dem Projekt. Durch den eingebauten Editor kann man auch innerhalb von Eclipse den Programmcode bearbeiten.

Blizzard Interface Kit
Blizzard stellt ein Interface Kit zur Verfügung, welches XML und LUA Dateien bereitstellt, die die Standard-Oberfläche von WoW beschreiben.

Ace
Eine Sammlung von LUA-Bibliotheken welche Funktionen bereitstellen die uns beim Erstellen des Mods sehr hilfreich waren. Auf die einzelnen Bibliotheken wird unter 3. genauer eingegangen.

Chronos
Ist ebenfalls eine Bibliothek aus der Sammlung CosmosUI[1], welche den für unser Addon benötigten Timer bereitstellt

Notepad++
Windows-Editor mit praktischen Features: Code-Highlighting, Zusammenklappen von Code-Abschnitten. Vereinfacht dadurch die Arbeit mit XML und LUA-Dateien.

Nsis Installer Maker
Ein sehr bekannter scriptbasierter Installer(exe) Maker, durch welchen für das Addon eine Setup-Exe erstellt werden konnte, welche automatisch nach dem WoW-Verzeichnis sucht und das Addon entsprechend dorthin installiert. Weiterhin ist dadurch auch eine Deinstallation des Addons über die Systemsteuerung>Software möglich!

3. Die Grundlagen eines WoW-Mods unter Verwendung von Ace2

3.1 Wie erstellt man einen Mod für WoW?

Um ein Mod für World of Warcraft zu erstellen benötigt man Kenntnisse in der Skriptsprache lua[2]. Plant man außerdem eine grafische Oberfläche für den Mod, so sollte man sich auch mit XML[3] gut auskennen, da alle GUI-Elemente im Spiel mit XML erstellt und verändert werden können.

Um einen einfachen Mod zu erstellen muss man im Addon-Ordner des Spiels */World of Warcraft/Interface/Addons* einen neuen Ordner anlegen und diesem einen aussagekräftigen Namen geben, am besten den Namen des Mods, in unserem Fall "*BrabbelRP*". In diesen Ordner legt man alle Dateien, die für den Mod benötigt werden, sowie eine Organisationsdatei. Diese Organisationsdatei muss den selben Namen haben wie der Ordner und muss auf ".*toc*" enden. Wenn das Spiel gestartet wird, durchsucht es alle Ordner in *Interface/Addon/* auf diese toc-Dateien in denen die wichtigen Informationen zum Mod stehen.

1 http://www.cosmosui.org
2 http://www.lua.org
3 http://www.w3.org/XML/

```
## Interface: 20100
## Title: BrabbelRP
## Author: Friederike Bulka, Björn Teichmann
## Version: 1.0
## Notes: Game Modding Seminar 2007
[..]
BrabbelRP.xml
BrabbelRP.lua
[..]
```

Code 1: Auszug aus BrabbelRP.toc

Zunächst wird eine ID für das Interface angegeben für das programmiert wurde. Diese Zahl ändert sich mit jedem Patch und dient zur Überprüfung ob der Mod noch aktuell ist, oder bereits veraltet. Dann werden Informationen wie Titel, Autoren, Version und Hinweise zum Mod angegeben, die in der Mod-übersicht im Spiel angezeigt werden. Zum Schluss werden dann alle Dateien angegeben, die das Spiel laden soll. In diesem Fall *BrabbelRP.xml* und *BrabbelRP.lua*. In diesen Dateien wird dann mit XML die grafische Oberfläche modeliert und mit der Skriptsprache LUA die Funktionalität implementiert. Das Spiel lädt die Dateien und arbeitet sie der Reihe nach ab, wobei man als Entwickler auf Funktionen zurückgreifen sollte, mit denen man abfragen kann ob der Mod vollständig geladen wurde.

Da viele Mods ähnliche Funktionalitäten benötigen, bietet es sich an Bibliotheken zu erstellen welche häufig genutzte Funktionen zur Verfügung stellen. Für World of Warcraft gibt es mit *Ace2*[4] eine große Sammlung von Bibliotheken, die Mod-Entwicklern das Programmieren erleichert. Einige der verwendeten Ace-Bibliotheken sollen im Folgenden kurz erkärt werden.

3.2 Ace-Addon-2.0

Die Basisbibliothek von Ace ist *AceAddon-2.0*. Die Bibliothek ist eine Art Interface mit dem man ein Objekt (den Mod) erstellt. Dadurch werden wichtige Funktionen wie *OnInitialize*, *OnEnable*, etc. bereitgestellt.[5] Ein einfaches Addon könnte so aussehen:

```
MyAddon = AceLibrary("AceAddon-2.0"):new()

function MyAddon:OnInitialize()
   -- code here, executed only once.
   self:Print("MyAddon initialized");
end

function MyAddon:OnEnable(first)
   -- code here, executed after everything is loaded.
   -- Note: AceDB-2.0 will also call this when standby is toggled.
end
```

Code 2: ein einfaches Ace2-Addon[6]

4 http://www.wowace.com
5 Für weitere Infos siehe http://www.wowace.com/wiki/AceAddon-2.0
6 Beispiel von http://www.wowace.com/wiki/AceAddon-2.0

Wenn WoW diesen Mod lädt wird zuerst die Funktion *MyAddon:OnInitialize()* aufgerufen. In der Chatkonsole des Spielers wird nach dem Einloggen dann der Text "*MyAddon initialized*" angezeigt. Ein solcher Beispielmod hat allerdings noch einige Nachteile. Der gravierendste ist, dass noch keine Interaktion durch den/die Nutzer/in möglich ist.

3.3 AceConsole-2.0

Um die Funktionen von BrabbelRP umzusetzen und zu testen, wurden zunächst nur Kommandozeilenbefehle verwendet. Die grafische Oberfläche folgte später. Daher soll im Folgenden kurz erklärt werden wie mit *AceConsole-2.0* solche Kommandozeilenbefehle einfach registriert werden können.

```
BrabbelRP = AceLibrary("AceAddon-2.0"):new("AceConsole-2.0");

local options = {
    type='group',
    args = {
          createCat = {
                type='text',
                name = "createCat",
                desc = "create a new category",
                usage = "<Category Name here>",
                get = false,
                set = "createCat",
          },
    [..]
    }
}

BrabbelRP:RegisterChatCommand({"/brabbelrp", "/brp","/brabbel"},
options)

[..]

function BrabbelRP:createCat(kategorie)

    -- save new category here
    [..]
end
```

Code 3: options für AceConsole-2.0

Im obigen Beispiel wird deutlich wie *AceConsole-2.0* verwendet wird. Zunächst wird die Bibliothek für das Addon registriert (*Zeile 1*). In der Variable *options* können dann mehrere Befehle festgelegt werden die dem Benutzer zur Verfügung gestellt werden sollen. Dabei gibt man einen Namen (*name*), Beschreibung (*desc*), Verwendungshinweise (*usage*), sowie *get*- und *set*-Funktion an.

Die Befehle, die wir in *options* zusammengestellt haben, müssen nur noch für das Addon registriert werden. Das geschieht mit *BrabbelRP:RegisterChatCommand([..])*. Dabei registrieren wir "*/brabbelrp*" und 2 alternative Schreibweisen mit denen der Nutzer das Addon steuern kann.

In diesem Fall wurde eine Funktion zum Erstellen einer neuen Kategorie registriert. Der/die Benutzer/in muss nur den entsprechenden Befehl sowie den Kategorienamen angeben. Der eingegebene Kategoriename wird dann der *set*-Funktion *BrabbelRP:createCat(kategorie)* übergeben. In dieser Funktion wird dann implementiert wie die Kategorie gespeichert wird.

Der/die Nutzer/in kann jetzt eine neue Kategorie erstellen indem er/sie folgende Zeile in das Chatfenster eingibt:

```
/brabbelrp createCat test
```

Code 4: Beispiel für Kommandozeilenaufruf des Mods

3.4 AceDB-2.0

Wenn man während einer Spielsitzung[7] Kategorien erstellt möchte man die erstellten Daten natürlich auch in der nächsten Spielsitzung verwenden können. Dazu müssen die Daten gespeichert werden. *AceDB-2.0* hilft dem Entwickler beim Erstellen einer spielinternen Datenbank. Um Missbrauch zu verhindern hat Blizzard starke Einschränkungen für die Skriptsprache LUA gemacht. Die verwendete Version erlaubt es Entwicklern nicht Dateien zu lesen oder zu schreiben. Dadurch soll verhindert werden, dass spielinterne Daten an das System gelangen können und von Bots, etc. ausgelesen werden könnten.

Um dennoch das Speichern von Daten zu erlauben können sogenannte *SavedVariables* angelegt werden. Diese können für einzelne Charaktere, für einen Server oder für einen Account angelegt werden. Da BrabbelRP für Rollenspieler gedacht ist, die unterschiedliche, individuelle Charaktere spielen, werden die Daten hier für jeden Charakter einzeln gespeichert. Diese *SavedVariables* liegen in einem Unterordner des WoW-Verzeichnis */WTF/user/server/charakter*, wobei *user, server* und *charakter* Platzhalter für die eigentlichen Namen sind.

Beim Start einer Sitzung liest WoW eventuell vorhandene *SavedVariables* für einen Mod ein. Während der Sitzung können diese Daten vom Mod verändert werden. Wenn die Spielsitzung beendet wird, werden die veränderten Daten aus dem Programmspeicher wieder in die Dateien im SavedVariables-Ordner geschrieben. Um einen solchen Datenspeicher für ein Mod zu registrieren, muss nur "*AceDB-2.0*" für das Addon registriert werden (äuqivalent zu *AceConsole-2.0*):

```
BrabbelRP = AceLibrary("AceAddon-2.0"):new("AceConsole-2.0",
"AceDB-2.0")


BrabbelRP:RegisterDB("BrabbelRPDB", "BrabbelRPDBPC", "char")
```

Code 5: AceDB-2.0 einbinden

Dannach kann über die registrierte Variable der Speicher direkt angesprochen werden:

7 Eine Spielsitzung beginnt mit dem Einloggen eines Charakters und endet, wenn dieser Charakter ausloggt.

```
BrabbelRP.db.char.categories = {}
```

Code 6: Beispiel für Aufruf einer SavedVariable

3.5 AceLocale-2.2

Da BrabbelRP ein Rollenspielermod den möglichst viele Spieler/innen verwenden sollen muss man sich auch über die Frage der Lokalisierung Gedanken machen. Am einfachsten lässt sich das mit *AceLocale-2.2* lösen. Mit Hilfe dieser Bibliothek erstellt man eine Art "Wörterbuch" in das man alle Übersetzungen schreibt, die man benötigt.

```
L = AceLibrary("AceLocale-2.2"):new("BrabbelRP")
```

Code 7: Anlegen eines Ace-"Wörterbuchs"

Das Wörterbuch (*L*) muss jetzt nur noch gefüllt werden. Dazu legt man für jede vorgesehene Sprache eine Datei an. Zur besseren Übersicht sollte man diese in einem extra Unterordner ablegen, bspw. */BrabbelRP/lang*. Der Dateiname muss dabei wie folgt aussehen: "*Locale-"+Sprachkürzel+".lua*". Im Fall von BrabbelRP haben wir 2 Dateien - eine für den deutschen und eine für den englischen Client - angelegt: *Locale-deDE.lua* und *Locale-enUS.lua*.

In diesen Dateien befindet sich je eine Tabelle mit allen Übersetzungen.

```
L:RegisterTranslations("deDE", function() return {
    [..]
    ["remove category "] = "Kategorie l\195\182schen",
    ["added new line to cat"] = "Neue Zeile zur Kategorie
hinzugef\195\188gt",
    ["category changed"] = "Kategorie ge\195\164ndert",
    ["No categories active at the moment."] = "Zur Zeit sind keine
Kategorien aktiv.",
    [..]
}
```

Code 8: Auszug aus Locale-deDE.lua

Diese Texte müssen für alle vorgesehenen Sprachen angelegt werden und können dann wie folgt verwendet werden.

```
self:Print(L["No categories active at the moment."]);
```

Code 9: Verwenden von AceLocale-2.2

Im deutschen Client würde an dieser Stelle der Satz "*Zur Zeit sind keine Kategorien aktiv.*" ausgegeben werden.

4. Dateiübersicht

- **BrabbelRP.toc**
  Organisationsdatei (siehe 3.1)
- **BrabbelRP.lua**
  enthält alle unsere Programmfunktionen
- **BrabbelRP.xml**
  enthält den UI-Code für unser verschiebbares BrabbelRP-Icon (mit welchem man die Einstellungen aufruft)
- **BrabbelRPForm.xml**
  enthält den gesamten restlichen UI-Code des Addons
- Ordner: **libs**
  enthält die von uns genutzten externen Libraries
- Ordner: **lang**
  enthält die Sprachdateien zur Lokalisierung des Addons
- Ordner: **defaultBrabbel**
  enthält die Informationen zum Herstellen der Standardeinträge des Gebrabbel

5. Automatisches Brabbeln - Wie funktioniert das?

5.1 Timer / Wahrscheinlichkeit / Modus

Es gibt in unserem Addon 2 Arten von Ausgaben:

- **direkte Ausgaben**
- **zeitgesteuerte Ausgaben**

Eine **direkte Ausgabe** ist eine Reaktion auf ein bestimmtes Ereignisse in WoW, wie z.B. eine bestimmte Kampfhandlung oder das Betreten eines bestimmten Bereichs. Auf welche Ereignisse eine Kategorie reagieren soll, wird in BrabbelRP über die **(direkten) Modi** eingestellt.
Um nicht bei jedem Ereignis jedes Mal eine Ausgabe zu bekommen, wird zudem eine Wahrscheinlichkeit vergeben, nach welcher berechnet wird, wann eine Ausgabe aus dieser Kategorie erfolgen soll.

Bsp: Bei einer Wahrscheinlichkeit von 25%, würde eine Ausgabe im Schnitt nur aller 4 Versuche wirklich passieren.

Eine **zeitgesteuerte Ausgabe** erfolgt dagegen über einen einstellbaren Timer statt direkt auf Ereignisse von WoW. Auch hier wird aber beim Feuern dieses Timers nur zur eingestellten Wahrscheinlichkeit eine wirkliche Ausgabe passieren.

Bsp: Bei einem Timer von 10 Sekunden und einer Wahrscheinlichkeit von 50%, würde das Addon aller 10 Sekunden eine Ausgabe versuchen, aber dann nur aller 2 Versuche wirklich reagieren.

Optional kann einer Kategorie mit **zeitgesteuerte Ausgabe** ein **(indirekter) Modus** vergeben werden. Dieser steuert, wann eine Ausgabe erfolgen soll, und wann nicht.

Bsp: Eine Kategorie mit dem Modus "Stadt" ist außerhalb der Stadt vollkommen still, und wird nur innerhalb der Stadt versuchen etwas auszugeben.

5.2 Chronos

Chronos ist ein Addon bzw. vielmehr eine Bibliothek der CosmosUI Addon-Sammlung[8]. Diese Bibliothek bietet uns die Möglichkeit auf einfache Art und Weise Timer einzubinden, welche für unsere automatische Ausgabe wie oben zu lesen nötig ist. Dabei vergeben wir einem Timer eine Funktion, einen Namen und eine Zeit, in welcher er feuern soll. Über den Namen des Timers haben wir zudem zu jeder späteren Zeit Zugriff auf diesen Timer und können ihn notfalls anhalten oder neustarten. Die Funktion ist in dem Fall natürlich der Versuch einer zeitgesteuerten Ausgabe.

5.3 Channels

In WoW gibt es mehrere Channels über die die Spieler miteinander kommunizieren können. Einige Channels sind Zonen-übergreifend:

Bsp: Weltverteidigung-Channel, Hauptstadt-Channel, Handel-Channel

Andere Channels sind auf die nähere Umgebung des Charakters beschränkt.

Bsp: /say, /yell, /emote

Letztere sind die Channels auf die wir uns bei der Erstellung des Mods konzentriert haben. Rollenspieler verwenden hauptsächlich diese Channels um ihren Charakter sprechen (oder schreien) zu lassen und um zu beschreiben, was der Charakter tut.

Die einzelnen Channels können im Spiel mit /s, /say, /y, /yell, etc. aufgerufen werden, ebenso wie die überregionalen Channels, die mit ihren Nummern angesprochen werden können[9]. Dieses Prinzip ist für die Spieler sehr einfach zu handhaben und erleichtert die Kommunikation. Wenn man mit Hilfe eines Mods Ausgaben in den Chat geben möchte, verwendet man die Funktion *SendChatMessage("msg" [, "chatType" [, "language"] [, "channel"]])*[10].

Genau wie die Spieler müssen auch Programmierer den entsprechenden Channel angeben, in dem der Text ausgegeben werden soll. Wird der Channel nicht angegeben wird automatisch der Sprachchannel /s benutzt. Um die Syntax des Spiels beizubehalten ist also ein Parser nötig, der die einzelnen Zeilen auf Channelangaben überprüft.

Bsp: Ein Spieler gibt „/e kratzt sich.“ ein. BrabbelRP würde ohne Parser diesen Text an den /s Channel leiten und der Charakter würde „/e kratzt sich.“ sagen. Gewünscht ist aber eine Ausgabe an den Emote-Channel.

Um herauszufinden welchen Channel der Spieler ansprechen möchte parst BrabbelRP den /e-Befehl am Anfang der Zeile. Handelt es sich um einen von BrabbelRP unterstützten Channel (/say /yell oder /emote) so wird die Zeile akzeptiert und gespeichert. Versucht der Spieler aber in andere Channels zu schreiben, wie z.B. Handels-Channel, Hauptstadt-Channel, etc. so bekommt der

8 http://www.cosmosui.org
9 Hat ein/e Spieler/in den Hauptstadt-Channel auf Nr. 1, so kann er/sie mit /1 in den Hauptstadt-Channel schreiben.
10 Für genauere Hinweise zur Funktion siehe http://www.wowwiki.com/API_SendChatMessage

Spieler eine Fehlermeldung und die Zeile wird nicht gespeichert. Dadurch soll unter anderem verhindert werden, dass Spieler das Addon missbrauchen um in Zonen-übergreifenden Channels zu spamen.

5.4 Event-Handling

Um - wie oben beschrieben - mit direkten Ausgaben auf bestimmte Ereignisse in WoW zu reagieren, nutzen wir die Events, welche WoW in vielen Fällen feuert. Dafür wird für ein UI-Objekt (in unserem Fall für einen bestimmten Frame) ein Event registriert, wodurch dieser Frame beim Feuern dieses Events benachrichtigt wird.

```
BrabbelRP:RegisterEvent(„PLAYER_LEVEL_UP");
```

Code 10: Registriere Event, das gefeuert wird, wenn der Charakter einen neuen Level erreicht.

Um mit unserem Addon dann auf dieses Event zu reagieren, wird zusätzlich für diesen Frame festgelegt, was bei einem Event damit geschehen soll. Dies geschieht mit dem Frame-Handler *OnEvent*. Wir geben das Event dabei inkl. seiner Argumente (*arg1-arg9*) an eine Funktion weiter, welche dann anhand der Art des Events entscheidet, ob eine direkte Ausgabe passieren soll oder zb. eine zeitgesteuerte Ausgabe aktiv geschaltet werden soll.

5.5 Kategorie - Status

Ganz kurz möchte ich vorbereitend auf 5.6 noch auf den Status einer Kategorie eingehen, da wir für unser Addon statt **aktiv** und **inaktiv** noch einen weiteren Status (**semi-aktiv**) benötigen. Dieser Status wird für Kategorien mit zeitgesteuerten Ausgaben[11] verwendet, und definiert dabei eine Pause der Ausgabe, während wir uns z.B. bei einer Kategorie mit dem Modus "*Stadt*" außerhalb einer Stadt befinden.

5.6 Modi-Format

Um die in 5.1 beschriebenen 2 Arten von Ausgaben zu ermöglichen, haben wir uns eine Art eigene Modus/Event-Klasse geschrieben. Jeder '*BrabbelModus*' besitzt einen Namen, einen Tooltip, eine boolean Variable und 2 Listen von WoW-Events.

- Der Tooltip (**tooltip)** wird angezeigt, während sich der Mauszeiger des Benutzers später in der UI über dem Modus-Eintrag befindet.
- Die boolean Variable (**isInstantEvent)** gibt an, ob es sich bei diesem Modus um eine direkte Reaktion handelt[12].
- Die erste Liste (**activating_events)** enthält alle WoW-Events, welche die Kategorie, die diesen Modus besitzt, bei *isInstantEvent == true* direkt feuern sollen bzw. bei *isInstantEvent == false* von semi-aktiv[13] auf aktiv schalten sollen.

11 siehe 5.1
12 siehe 5.1
13 siehe 5.5

- Die zweite Liste (**deactivating_events**) enthält für Modi mit *isInstantEvent == false* die WoW-Events, welche die Kategorie wieder auf semi-aktiv[14] schalten, also pausieren sollen.

Über die Verknüpfung mehrerer WoW-Events können wir gezielt Modi "beschreiben", so dass der Nutzer auf einfachste Weise bestimmen kann wann seine erstellen Kategorien/Gebrabbel auf welche Weise reagieren soll.

14 siehe 5.5

6. Quellen- und Literaturverzeichnis

- BrabbelRP-Download: http://www.derflash.de/blog/?page_id=115
- Ace 2: http://www.wowace.com
- Eclipse: http://www.eclipse.org/
- Skriptsprache LUA: http://www.lua.org
- Blizzards WoW Interface Addon Kit: http://www.blizzard.com/support/wow/?id=aww01671p
- WoW API: http://www.wowwiki.com/World_of_Warcraft_API
- WoW Events: http://www.wowwiki.com/Events_%28API%29
- WoW HowTos: http://www.wowwiki.com/Category:HOWTOs
- WoW Widget Reference: http://update.multiverse.net/wiki/index.php/Widget_Reference
- WoW UI XML Definition: http://fara.webeddie.com/ui/
- WoW Frames Tutorial: http://fara.webeddie.com/frames/
- XML Grundlagen: http://www.w3.org/XML/
- Cosmos (inkl. Chronos): http://www.cosmosui.org
- Nsis: http://nsis.sourceforge.net
- Notepad++: http://notepad-plus.sourceforge.net

Anhang: Anteil der einzelnen Team-Mitglieder

Friederike Bulka

- 1/2 Programmierung
- 1/2 Dokumentation schreiben
- Testen des Addons

Song Liang

- Handbuch schreiben
- Übersetzung des Addons und der Standardeinträge ins Chinesische
- Testen des Addons

Björn Teichmann

- 1/2 Programmierung
- 1/2 Dokumentation schreiben
- Testen des Addons