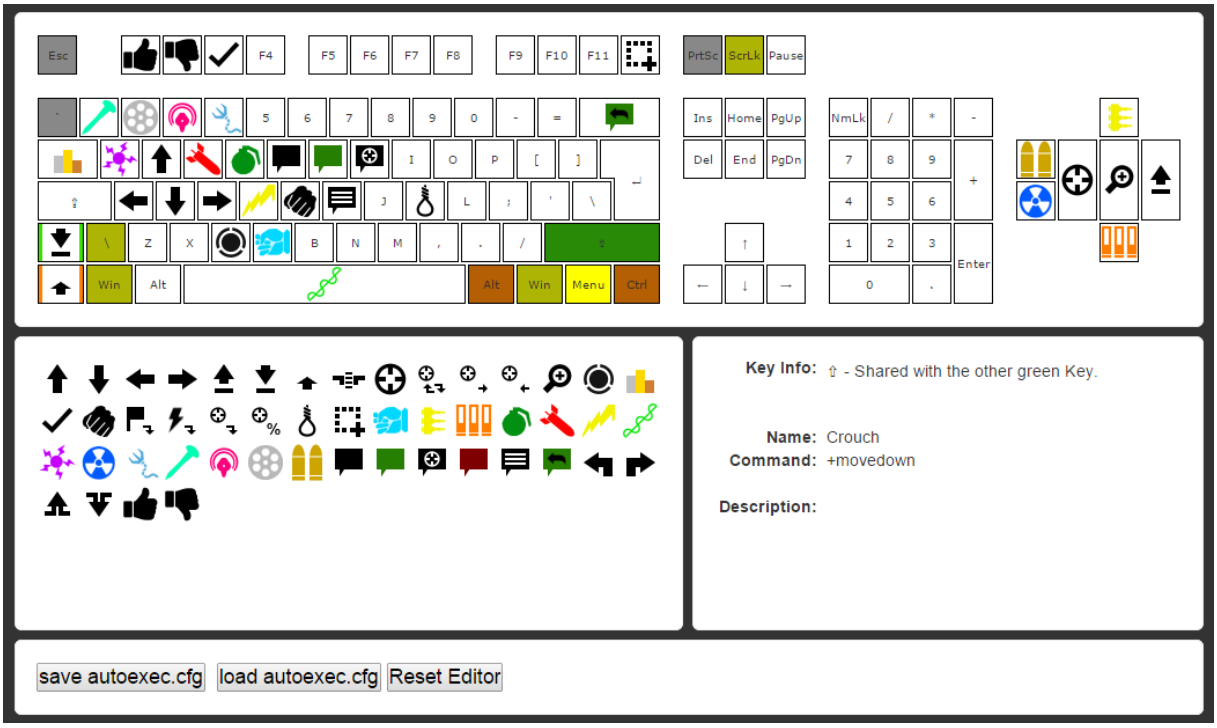
Quake Live Configuration Editor (QLCE)



Vorwort

Im Fach Scriptsprachen habe ich die Aufgabe gekriegt eine Semesterarbeit zu schreiben bei der ich eine kleines Programm entwickeln soll welches in einer für mich noch nicht so gängigen Programmiersprache geschrieben ist. Zur Auswahl standen: *PHP*, *JavaScript*, *Ruby*, *Perl* und *Python*. Bei der Arbeit soll es primär darum gehen eine Sprache besser kennen zu lernen und die Vorzüge bzw. Nachteile der Sprache zu erkennen.

Idee

Als gelegentlicher Gamer des First Person Shooter „Quake Live“ und jemanden der sich gerne seine Spieleinstellungen perfekt einrichtet kam ich auf die Idee für eben dieses Spiel einen Konfigurationseditor zu entwickeln. Das Spiel bietet sehr viele Einstellungsmöglichkeiten. Leider sind viele davon nur über die spielinterne Kommando Konsole konfigurierbar. Deshalb wollte ich etwas entwickeln mit dem man diese Konfigurations-Kommandos komfortabler editieren kann.

Konzept

Primär geht es bei dem Editor um die Tastenbelegungen. Sprich die Einstellungen dafür mit welcher Taste man beispielsweise den Waffenwechsel zum „Rocket Launcher“ vollführt. Die Zuweisungen erfolgen per *Drag & Drop*. Alle verfügbaren Kommandos werden als grafische Icons in einer Box dargestellt. Die zu belegenden Tasten sind in Form eines virtuellen Keyboards dargestellt. Die Belegung erfolgt durch ziehen der Icons auf die Tasten des virtuellen Keyboards.

eingesetzte Technologien

Ich habe mir für diese Arbeit ein paar Anforderungen gestellt:

- Plattform übergreifend
- Web-basiert
- kein Webserver benötigt (funktioniert auch auf lokalem Rechner)
- speichern und laden von lokalen Dateien
- Einsatz von *Drag & Drop*

Aufgrund dieser Anforderungen habe ich mich für die Scriptsprache *JavaScript* entschieden. Das führte natürlich auch dazu dass die Struktur für die grafische Darstellung mit *HTML* umgesetzt ist. Für die Darstellung des Tools werden *CSS* Stylesheets verwendet.
Zusätzlich zu *JavaScript* nutze ich *jQuery*. Diese JavaScript-Library vereinfacht vorallem das Implementieren von Code für *DOM Manipulationen* erheblich. Ausserdem ist *jQuery* Voraussetzung für das *jQuery UI* Paket. Die *Drag & Drop* Funktionalität des Tools wurde mit Hilfe von *jQuery UI* umgesetzt.
Eine weitere kleine Library die ich verwendet habe ist *hoverIntent.js*. Diese bietet eine verbesserte hover Funktion welche ich für das Anzeigen der InfoTexte verwendet habe.
Damit mit dem Tool Dateien vom lokalen Rechner geladen werden können ist der Einsatz der *File-API* nötig. Diese wurde mit *HTML5* eingeführt und ist in den gängigen modernen Browsern integriert. Um ein Browser-übergreifendes Speichern der Files zu ermöglichen kommt zusätzlich die Library *FileSaver.js* zum Einsatz.

Schwierigkeiten

Die Entwicklung von Webseiten war für mich nicht etwas ganz Neues. Aber es war doch schon etwas länger her seit ich das letzte Mal in Berührung mit all den Web-Technologien kam. Insbesondere die Darstellung hat mich ein wenig aufgehalten. Deshalb habe ich mich ein wenig länger als erwartet durch die Stylesheets gewurstelt bis die Seite so aussah wie ich es mir gewünscht habe.
Im Zusammenhang mit der Struktur des *JavaScript* Codes begleitete mich eine Unsicherheit bis zum Schluss. Ein grosser Teil des Codes sind *DOM Manipulationen* sowie auch zum Teil Style Eigenschaften welche per *JavaScript* verändert werden. Ausserdem ist vieles *Event basiert*. Ich weiss nicht wie man solchen Code sauber voneinander trennt. Ich stellte mir die Frage ob es sich lohnt und überhaupt möglich ist eine Art *MVC Architektur* für ein Tool dieser Art zu verwenden.

Erkenntnisse

JavaScript ist von der Syntax her eine einfach zu lernende Sprache die aber schnell ein wenig kompliziert werden kann. Ein Vorteil der gleichzeitig auch ein Nachteil darstellen kann ist die grosse Verbreitung und das breite Einsatzfeld der Sprache. Es gibt unzählige Libraries und Frameworks die auf *JavaScript* basieren. Die wichtigen sind auch gut dokumentiert, aber die grosse Masse kann einen auch überfordern. Ausserdem sind Konzepte wie beispielsweise die *Loopback Funktionen* am Anfang eher verwirrend. Das macht es zum Teil auch schwierig beim recherchieren geeignete Lösungen für bestimmte Probleme zu finden. Eine weitere Herausforderung beim entwickeln für den Browser ist die Tatsache dass man sich nicht nur mit einer Scriptsprache beschäftigt. Man muss neben *JavaScript* auch *HTML* und *CSS* beherrschen um zufriedenstellende Lösungen entwickeln zu können. Man sieht aber trotzdem schnell Ergebnisse seiner Arbeit im Browser Fenster. Man muss nichts kompilieren oder kopieren um seinen Code zu testen. Ein Vorteil von interpretierten Sprachen.
Ich glaube das in Zukunft noch viel mehr auf *JavaScript* und Web-Technologien gesetzt wird wenn es um Client-seitige Applikationsentwicklung geht.