Hochschule für Technik und Wirtschaft Dresden (FH)

Fakultät Informatik/Mathematik

Bachelorarbeit

im Studiengang Medieninformatik

Thema: Photoshop Erweiterungen – Realisierung eines Tutorials mit Hilfe des Flash Builder und Extensions

eingereicht von: Frank Schiemenz

eingereicht am: 15.08.2011

Betreuer: Prof. Dr. Teresa Merino

Inhaltsverzeichnis

Verzeichnis verwendeter Abkürzungen

AIR	Adobe Integrated Runtime
API	Application Programming Interface
AS3	ActionScript Version 3.0
CSAW	Creative Suite ActionScript Wrapper
CSS	Cascading Style Sheets
CSXS	Creative Suite Extensible Services
DOM	Document Object Model
DTD	Document Type Definition
E4X	ECMAScript for XML
ECMA	European Computer Manufacturers Association
ESTC	ExtentScript Tool Kit
GIMP	GNU Image Manipulation Program
GNU	GNU's Not Unix
GPL	GNU General Public License
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
PDF	Portable Document Format
PNG	Portable Network Graphics
RIA	Rich Internet Application
RPC	Remote Procedure Call
SDK	Software Development Kit
SWF	Shockwave Flash
UML	Unified Modeling Language
URI	Uniform Resource Identifier
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language

Verzeichnis der Begriffe und Definitionen

Awareness	Ist die Fähigkeit einer System-Komponente, die durch andere Komponenten des Systems ausgelösten Ereignisse wahrzunehmen und ggf. darauf zu reagieren, um selber Ereignisse auszulösen.
Browser	Ein Programm zum Anzeigen von Seiten des World Wide Web.
Community	Der englischer Begriff für Gemeinschaft. Ansammlung von Menschen, die sich über gemeinsame Interessen austauschen.
CSS	Sprache zur Beschreibung der Darstellung strukturierter Dokumente. Oft anzutreffen in Verbindung mit HTML und XML, welche als Lieferanten eines dazustellenden Inhalts dienen.
Desktop 2.0	Benennung der neuesten Generation der Desktop-Technologien in Analogie zu „Web 2.0“.
Endianness	Die Byte-Reihenfolge bezeichnet die Speicherorganisation für einfache Zahlenwerte im Arbeitsspeicher. Die zwei Varianten sind „Little-Endian“ und „Big-Endian“, die sich in der Position des höchstwertigen bzw. niederstwertigen Bits unterscheiden.
Eventhandler	Eine Funktion, die zur Ereignisbehandlung ausgeführt wird.
Interface	Die Schnittstelle (englisch für „Grenzfläche“) ist der Teil eines Systems, der der Kommunikation dient.
Makro	Ein Makro ist ein Programm, das eine fest vorgegebene Folge von Befehlen, Aktionen oder Tastaturcodes enthält.
Parser	Programm zum Analysieren und Umwandeln einer Eingabe in ein zur Weiterverarbeitung verwendbares Format.
Pixel	Das „Picture Element“ bezeichnet sowohl die kleinste Einheit einer digitalen Rastergrafik als auch deren Darstellung auf einem Bildschirm.
Plug-in	Auch „Plugin“. Programm, welches über eine definierte Schnittstelle in ein Anderes eingebunden wird, um dieses zu erweitern.
Screenshot	Ist abgespeicherte Form des aktuellen grafischen Bildschirminhaltes.
Stylesheet	Der englischer Begriff für „Stilvorlage“. Siehe CSS.
Tutorial	Eine „Übung unter Anleitung“. Bezeichnet auch ein Dokument, das mit Hilfe von Beispielen erklärt, wie man mit einem Programm oder einer Programmiersprache umgeht.
Workflow	Der „Arbeitsfluss“ bzw. „Arbeitsablauf“ ist eine vordefinierte, zusammenhängende Abfolge von Aktivitäten und Funktionen.

Verzeichnis der Abbildungen

Verzeichnis der Tabellen

Verzeichnis der Programmausschnitte

1 Einleitung

1.1 Motivation

Photoshop. Ein Phänomen, dass nunmehr schon seit über 20 Jahren anhält. Es ist das sich mit Abstand am besten verkaufende Bildbearbeitungsprogramm der Welt. Für das US-amerikanische Softwarehaus Adobe Systems stellt Photoshop einen gewaltigen Grundpfeiler für dessen finanziellen Erfolg dar. Zusammen mit anderen Produkten der „Creative Suite“ bescherte es Adobe zuletzt wieder Rekorderlöse: für das Jahr 2011 rechnet man mit einem Umsatzwachstum von zehn Prozent[1] im Vergleich zum Vorjahr. Ein Ende dieser Erfolgsgeschichte ist dabei noch lange nicht abzusehen. Bereits mit Version 2.0 begann Photoshop den Markt deutlich zu dominieren. Das war 1991, also schon ein Jahr bevor die erste Version für das Windows Betriebssystem und der Intel Plattform vorgestellt wurde. Heute ist Version 12.1 erhältlich und es besteht allgemeiner Konsens darüber, dass Photoshop der Industriestandard seiner Branche und eine so genannte „Killer Application“ ist [Schewe 2000].

Photoshop ist integraler Bestandteil jedes „Workflow“, wenn es um die Gestaltung und Aufbereitung digitaler Medien geht. Kreative und gestalterisch veranlagte Menschen werden deshalb im Laufe ihres Lebens irgendwann einmal mit der Frage konfrontiert werden, ob sie mit Photoshop arbeiten sollten. Im Umfeld der beliebten Grafik-Software hat sich mittlerweile eine ganze Industrie angesiedelt. Unzählige Bücher, Videos, Magazine und Internetangebote beschäftigen sich mit dem Thema der Bildbearbeitung und bieten vielfältige Informationen, die Lust am Erlernen neuer Bildbearbeitungs-techniken vermitteln. Aber auch Software-Firmen haben die Möglichkeiten für sich entdeckt: „Alien Skin Software“[2] und „Filter Forge“[3] beispielsweise entwickeln kommerziell erfolgreiche Produkte, welche die Funktionalität und damit die Attraktivität von Photoshop erhöhen.

[1] http://www.adobe.com/aboutadobe/pressroom/pressreleases/201106/Q211Earnings.html
[2] http://www.alienskin.com/
[3] http://www.filterforge.com/

Die von Adobe zum Download angebotenen Probeversionen, dienen oft als Entscheidungsgrundlage und Kaufanreiz. Doch auch wenn sich der Nutzer angesichts der Produkt- und Abonnementpreise für eine der Alternativen, wie beispielsweise das kostenlose und freie „GIMP“[4] entscheidet, kommt man an einige andere Produkte von Adobe nicht herum: „PDF“ und „Flash Player“ sind heutzutage den meisten Internet-affinen Menschen ein Begriff.

Das größte Geschenk aber, das Adobe der Computerbranche machen konnte, war die allgemeine Verbreitung von Plug-ins. Dank dem Branchenprimus Photoshop, wurden die Nutzererwartungen in Software-Produkte immens gesteigert. Das führte dazu, dass es sich heute kein Entwickler irgendeines Software-Titels mehr leisten kann auf eine Erweiterungs-Architektur zu verzichten [Kas 1999].

1.2 Zielstellung

Im Zuge des Studiums an der Hochschule für Technik und Wirtschaft Dresden wurde im sechsten Semester ein Pflichtpraktikum absolviert. Die dabei von mir gewonnenen Kenntnisse über die Technologien im Umfeld von Photoshop sollten als Grundlage für weitere Untersuchungen im Rahmen einer Bachelorarbeit dienen.

Ziel dieser Arbeit ist das Aufzeigen und Untersuchen der Erweiterungsmöglichkeiten, welche Adobe für Photoshop anbietet. Dabei werden die Vor- und Nachteile der jeweiligen Technologie aufgezeigt und eine Gesamtübersicht gegeben. Darüber hinaus soll anschließend mit Hilfe der gewonnenen Erkenntnisse eine prototypische Anwendung zur Durchführung von interaktiven Tutorials im Photoshop-Kontext umgesetzt werden. Diese Anwendung soll auf den Erkenntnissen der Arbeit fußen und von den spezifischen Vorteilen, der für die Implementation ausgewählten Technologie Gebrauch machen.

1.3 Aufbau der Arbeit

Im anschließenden zweiten Kapitel werden die unterschiedlichen Technologien und Erweiterungsmöglichkeiten von Photoshop eingeführt und aus der Sicht eines Software-

[4] http://www.gimp.org/

Entwicklers untersucht. Die im Zuge der Untersuchungen erkannten Vor- und Nachteile werden analysiert und dargestellt.

Das dem nachfolgende dritte Kapitel beschäftigt sich mit der Umsetzung des Tutorials. Es bespricht das Konzept und den Inhalt des Tutorials und zeigt das Vorgehen bei der Implementierung der Extension „MyTutor“ systematisch auf.

Im vierten und abschließenden Kapitel werden die Ergebnisse der theoretischen Untersuchungen sowie der praktischen Umsetzung zusammengefasst. Darüber hinaus wird zum Schluss ein Blick auf zukünftige Entwicklungen geworfen.

2 Untersuchung der Erweiterungsmöglichkeiten von Photoshop

Das Ziel dieses Kapitels ist es, einen Überblick darüber zu geben, welche Erweiterungsmöglichkeiten prinzipiell für das Programm Photoshop zur Verfügung stehen. Dabei werden insbesondere die klassischen Plug-ins, das Photoshop-Scripting, aber auch aktuelle Entwicklungen wie Extensions und die „Pixel Bender"-Technologie beleuchtet.

2.1 Plug-in Module

Photoshop Plug-ins sind separate Software-Programme in Form einer einzelnen Datei, welche von Adobe oder Drittherstellern entwickelt werden, um das Hauptprogramm zu erweitern. Endanwender können diese Zusatzmodule unabhängig von einander hinzufügen oder ersetzten. So können Nutzer Photoshop ihren speziellen Bedürfnissen anpassen, ohne das Programm zu modifizieren. In diesem Zusammenhang spricht man von Photoshop auch als „Plug-in Host" (Gastgeber). „Plug-in Hosts" sind solche Programme, die Plug-in Module aufnehmen können. Ihre Aufgabe besteht darin, Plug-in Module in den Speicher zu laden und diese bei Bedarf aufzurufen. Interessant dabei ist, dass ein Plug-in wiederum selbst ein „Host" für andere Module sein kann. Beispielsweise ist dies bei dem Plug-in „Photoshop Adapter" für Adobes Vektor-Grafikprogramm „Illustrator" der Fall. Damit kann man bestimmte Plug-in Typen („Format" und „Filter") laden, um innerhalb des Illustrator mit ihnen zu arbeiten. Aber auch eine ganze Reihe Produkte[5] von Drittanbietern ist in der Lage Photoshop Plug-ins zu verwenden.

2.1.1 Plug-in Typen

Wie bereits angedeutet wurde, gibt es eine Reihe unterschiedlicher Plug-in Typen (vgl. Tab. 1), die sich äußerlich nur anhand ihrer Dateiendung unterscheiden. Nachfolgend werden alle Plug-in Typen kurz charakterisiert:

- *Automation* Module haben Zugriff auf alle Ereignisse, die durch die Script-

[5] http://tml.pp.fi/gimp/pspi.html

Unterstützung anderer Module oder durch Photoshop selber statt finden. Zu finden sind diese Module unter `Datei > Automatisieren`.

- *Selection* Module nehmen Einfluss auf die ausgewählten Pixel und Pfade des aktuellen Bildes. Dieser Typ erscheint unter dem Menüeintrag `Selektion`.

- *Color Picker* bieten eine Möglichkeit zur Implementation eines weiteren Farbwählers in Anlehnung an den Photoshop-eigenen oder den des Betriebssystems an. Diese Auswahldialoge erscheinen, wenn der Nutzer auf die Vorder- oder Hintergrundfarbe innerhalb der Werkzeugleiste klickt. Der aktuelle Farbwähler wird unter `Bearbeiten > Voreinstellungen > Allgemein` eingestellt.

- *Export.* Mit Hilfe dieser Module können geöffnete Bilder in nicht-unterstützte oder unkomprimierte Formate ausgegeben werden. Zu finden sind diese unter dem Menüeintrag `Datei > Exportieren`.

- *Filter* verändern einen ausgewählten Bereich des aktuellen Bildes auf Pixelebene. Filter-Plugins erscheinen im Menü `Filter`.

- *File Format* bzw. *Image Format.* Diese Module dienen ähnlich dem *Export*-Modul dem Schreiben aber auch dem Lesen zusätzlicher Bildformate. Diese erscheinen in den `Speichern` und `Speichern unter` – Dialogen.

- *Stack Renderer* können in „Photoshop Extended“ dazu verwendet werden mathematische Operationen auf Ebenen durchzuführen, die zuvor in ein „Smart-Objekt“ konvertiert wurden. Unter `Ebene > Smart-Objekte > Stapelmodus` sind diese zu finden.

- *Import* Module dienen z.B. als Schnittstelle zu Kameras und Scannern. (unter Windows ist dies bspw. die WIA-Schnittstelle[6]). Das Menü unter `Datei > Importieren` listet diese Plug-ins auf.

[6] http://msdn.microsoft.com/en-us/library/ms630368.aspx

- *Measurement* Module definieren neue Daten-Objekte, die über `Analyse > Datenpunkte auswählen` eingebunden werden. Durch `Analyse > Messungen aufzeichnen` finden diese Verwendung.

- *3D.* Module dieser Kategorie haben in „Photoshop Extended“ Zugriff auf die Kameras, die Lichtquellen, das Polygonnetz inklusive der Texturen sowie auf die Animationsparameter einer 3D-Szene.

- *Parser* werden von Adobe dazu verwendet, um Daten, ähnlich wie Import- und Export-Module, zwischen Photoshop und anderen Programmen zu manipulieren. Dieser Plugin-Typ steht den Entwicklern nicht zur Verfügung, da die Schnittstelle dafür nicht öffentlich ist.

- *Extension* Module erweitern die Oberfläche - zum Beispiel durch neue Menüs.

- *General*. Diese unspezifizierten Module können generell die Aufgaben jedes anderen Plugin-Typs übernehmen.

Eine zusammenfassende Übersicht der Plugin-Typen gibt die nachstehende Tabelle:

Plugin-Typ	**Macintosh Dateityp**	**Windows Dateiendung**
3D	8BIF	.8BI
Automation	8LIZ	.8LI
Color Picker	8BCM	.8BC
Export	8BEM	.8BE
Extension	8BXM	.8BX
File Format	8BIF	.8BI
Filter	8BFM	.8BF
General	8BPI	.8BP
Import	8BAM	.8BA
Measurement	8MEA	.8ME
Parser	8BYM	.8BY
Selection	8BSM	.8BS
Stack Renderer	8BAM	.8BA

Tabelle 1: Plugin-Typen und die zugehörigen Dateiendungen.

2.1.2 Anatomie eines Plug-ins

Das ursprüngliche Plugin-Interface wurde entworfen, als Photoshop noch ein reines Macintosh-Produkt war. Entwickler für Plug-ins älterer Photoshop-Versionen haben deshalb immer noch das Problem der unterschiedlichen „Endianness“[7] der zugrundeliegenden Hardware-Plattformen. Photoshop CS5 für Macintosh unterstützt erstmals ausschließlich das Mac OS X („Leopard“ ab Version 10.5.7 bzw. „Snow Leopard“), welches nur auf Intel-Prozessoren, und damit auf „Little-endian“, läuft. Damit lösten sich einige der Probleme, die sich beim Format der für Plug-ins erforderlichen Ressourcen-Datei ergaben. Die kompilierte Form der „Plug-in Property List“ (PiPL) des Plug-ins wird beim Startvorgang von Photoshop durchsucht und legt letztendlich fest, welchem Typ (`Kind`, siehe Abb. 1) dieses zuzuordnen und wo dessen Einstiegspunkt ist. Verwirrend kann die Tatsche sein, dass die genaue Dateiendung eines Plug-ins unerheblich ist, solange es sich dabei um irgendeine unterstützte Endung handelt. So legt beispielsweise der folgende Auszug aus einer „Plug-in Property List“ einen Farbwähler (`Picker`) und einige seiner Meta-Eigenschaften fest:

```
resource 'PiPL' (ResourceID, plugInName " PiPL", purgeable){
{
   Kind{Picker},
   Name{plugInName},
   Category{vendorName},
   Version{(latestPickerVersion << 16) | latestPickerSubVersion},
   PickerID{vendorName " " plugInName},
   CodeWin64X86{"PluginMain"},
   SupportedModes{doesSupportBitmap, doesSupportRGBColor},
   EnableInfo{"true"},
}};
```

Listing 1: Auszug aus der „Plug-in Property List“ eines Farbwählers.

PICA[8] ist die anwendungsübergreifende Architektur, welche das Management aller Plug-ins im Hauptprogramm übernimmt und eine Standard-Schnittstelle für Plug-ins anbietet. Mit Hilfe der PICA ist es auch möglich, dass sich Plug-ins untereinander austauschen können. Die Funktionalität wird in der verbreiteten Programmiersprache C++ umgesetzt. Dazu stellt die „Adobe PICA API“ dem Programmierer eine ganze

[7] http://de.wikipedia.org/wiki/Byte-Reihenfolge
[8] Plug-In Component Architecture

Reihe von Rückruffunktionen nach dem Entwurfsmuster „Inversion of Control“ (IoC; deutsch: „Steuerungsumkehr“) zur Verfügung. Diese sind zu funktionalen Einheiten („Suites“) zusammengefasst und bieten Zeiger auf Funktionen und Parameterblöcke, mit denen man Zugriff auf fast alle Photoshop-Eigenschaften hat. Plug-ins können aber ihrerseits auch „Suites“ anbieten. Ein Beispiel dafür ist der „Adobe Dialog Manager“ (ADM). Dieses Plug-in stellt den Versuch dar, die Gestaltung von Dialogen einheitlich, das heißt programm- und plattformunabhängig zu ermöglichen. Ein Nachteil von ADM ist aber, dass damit in Photoshop nur modale Dialoge erzeugt werden können. Nicht-blockierende „Floating“-Dialoge werden nicht unterstützt [ADM 2003]. Ein Nachteil den es mit anderen Plug-ins gemein hat. Nach dem Erscheinen der 64-Bit Programm-versionen wird dieser Ansatz zwar inoffiziell[9] nicht mehr unterstützt, dennoch ist dieses Plug-in Bestandteil jeder Photoshop-Installation und wird im SDK dokumentiert.

Hervorzuheben ist, dass Plug-ins durch das Hinzufügen einer Terminologie-Ressource (ähnlich einer PiPL) und zusätzlicher Deskriptoren zur Ereignisbehandlung „scripting-awareness“ erlangen können. Solche Plug-ins können dann über spezielle Skript-Sprachen und den Makro-ähnlichen „Aktionen“ von außen gesteuert werden. Abschnitt 2.2 befasst sich näher mit der Thematik der Skript-Sprachen im Photoshop-Kontext.

2.1.3 Fazit

Wie gezeigt wurde, kann durch die Plug-ins eine Reihe tiefgreifender Erweiterungen realisiert werden. Durch die Verwendung einer etablierten Programmiersprache und der passenden „Suites“ ist ein schneller Zugriff auf den Speicher und die Inhalte der Pixelebenen eines Dokumentes möglich. Allerdings sollte man den hohen Einarbeitungsaufwand trotz der sehr gut dokumentierten API nicht unterschätzen. Unter Verwendung einer Entwicklungsumgebung wie bspw. „Microsoft Visual C++“[10] bzw. „Apple XCode“[11], kann man durch Setzen von Unterbrechungspunkten das Plug-in in der laufenden Host-Anwendung quasi „live“ debuggen. Die durch ein Plug-in erzeugten Oberflächen und Dialoge sind blockierend, was die Interaktivität einschränkt.

[9] http://forums.adobe.com/thread/640103
[10] http://msdn.microsoft.com/library/hs24szh9.aspx
[11] http://developer.apple.com/xcode/

2.2 ExtendScript

Ein „Skript“ ist eine Textdatei, die eine Serie von Befehlen enthält, welche Photoshop anweisen bestimmte Aktionen auszuführen: wie beispielsweise das Anwenden eines Filters auf eine Selektion des geöffneten Dokumentes. Dem Nutzer wird damit die Möglichkeit in die Hand gegeben, auf die Werkzeuge der Host-Anwendung sowie auf Funktionalitäten der Plug-ins programmatisch zuzugreifen. Photoshop unterstützt dabei verschiedene Sprachen, die in Tabelle 2 gegenübergestellt werden.

Die nachfolgenden Betrachtungen beziehen sich auf „JavaScript/ExtendScript“, da diese Sprache auf Grund ihrer Betriebssystem-Unabhängigkeit die höhere Verbreitung und somit die größte Relevanz für die weiteren Untersuchungen hat.

Skript-Typ	Dateityp	Dateiendung	Betriebssystem
AppleScript	Binär	.scpt	Mac OS
JavaScript / ExtendScript	Text	.js, .jsx (.jsxinc)	Mac OS und Windows
	Binär	.jsxbin	
VBScript	Text	.vbs	Windows
Visual Basic	Binär	.exe	Windows

Tabelle 2: Die von Photoshop unterstützten Skript-Typen.

Die Sprache „ExtendScript“ ist eine von Adobe erweiterte und umbenannte Form der im Internet verbreiteten „Websprache“ ECMAScript[12]. Als „JavaScript“ findet es dort hauptsächlich für das clientseitige DOM-Scripting in Webbrowsern Verwendung [Kriesinger 2005]. ExtendScript bietet aber als Prototyp-orientierte Sprache, genauso wie JavaScript, kein Klassenkonzept im Sinne der objektorientierten Programmierung an [Flanagan 2007].

2.2.1 Photoshop Objektmodell und ScriptUI

Die ExtendScript-Anweisungen werden mit Hilfe der „ExtendScript Engine“ zum Zwecke der Syntaxanalyse entgegengenommen. Dieser „Parser“ ist seit Photoshop Version 8.0[13] enthalten und gestattet Skripten Zugriff auf das Photoshop-Objektmodell.

[12] http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf
[13] Die Skriptsteuerung war aber schon in Version 7.0 durch ein Plug-in möglich.

Dieses „Document Object Model“ (DOM; deutsch: „Dokumentobjektmodell“) ist eine hierarchisch organisierte Struktur von Klassen, welche den funktionalen Aufbau von Photoshop repräsentieren und nach außen hin offenlegen. Die in Abbildung 1 enthaltene hierarchische Darstellung[14] bietet einen Überblick über einige der am häufigsten verwendeten Schlüsselklassen des Photoshop-Objektmodells.

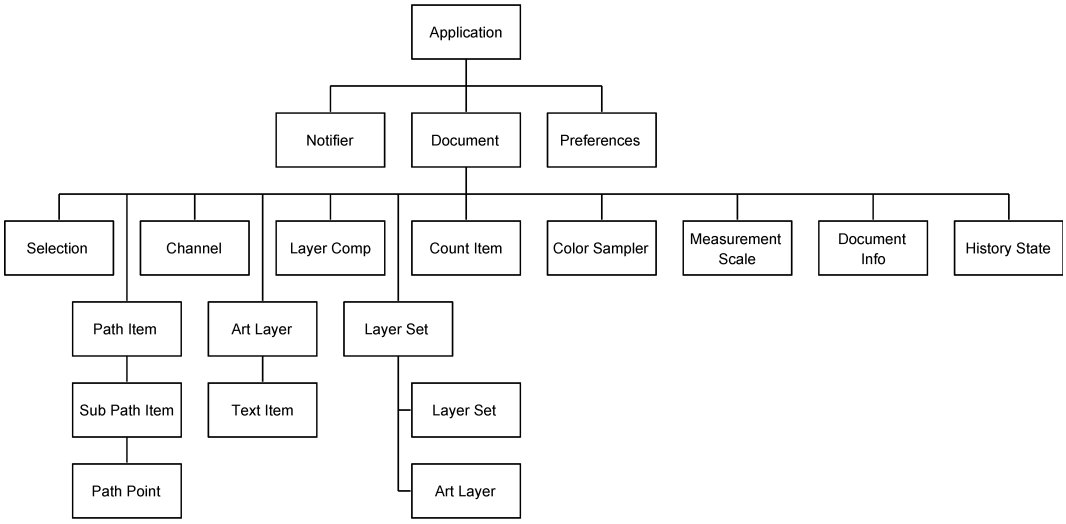

Abbildung 1: Übersicht ausgewählter Klassen des Photoshop-Objektmodells.

Da in dieser Hierarchie Klassen andere Klassen enthalten, spricht man auch von einer „containment hierarchy“ (Enthaltenseinshierarchie). Auf die Methoden und Attribute der Klassenobjekte wird, wie in anderen Sprachen üblich, durch Verwendung der Punktnotation[15] zugegriffen. Durch `app.documents.add()` wird beispielsweise ein neues Photoshop-Dokument erstellt. Das Singleton `app` repräsentiert hier die Klasse „Application“ des Photoshop-DOM.

Adobe bietet darüber hinaus mit „ScriptUI“[16], einem Objektmodell für Nutzeroberflächen, die Möglichkeit einige Elemente zur grafischen Gestaltung mit Hilfe von JavaScript zu beschreiben [ESTC 2010]. Beispielsweise legt der Quelltext in Listing 2 einen JavaScript-Funktionsprototyp fest und erweitert diesen um die Funktion `main()`. Über den Konstruktor des `Window`-Objektes erzeugt diese Funktion einen

[14] http://cssdk.host.adobe.com/sdk/1.0/docs/WebHelp/app_notes/ps_obj_model.htm
[15] http://jscheme.sourceforge.net/jscheme/doc/javaprimitives.html
[16] http://jongware.mit.edu/Sui/

Dialog in dem sich noch ein Panel mit einer Schaltfläche befindet.

```
function myJavaScriptPseudoClass(){}
myJavaScriptPseudoClass.prototype.main = function(){
   var dlg = new Window('dialog', 'MyDialog');
   dlg.pnl = dlg.add('panel', undefined,'');
   dlg.pnl.closebn = dlg.pnl.add('button', undefined, 'Neu');
   dlg.pnl.closebn.onClick = function(){
      app.documents.add();
      alert('Ein neues Dokument wurde über das PS-DOM erstellt!');
      dlg.close();}
   dlg.show();}
new myJavaScriptPseudoClass().main();
```

Listing 2: ScriptUI- und Photoshop-Objektmodell-Zugriff.

Anschießend wird eine JavaScript-Hinweisbox angezeigt, nach deren Bestätigung durch den Nutzer der Dialog beendet wird. Sowohl der Dialog als auch die `Alert`-Box sind modal, was bei deren Darstellung kein paralleles Arbeiten in Photoshop zulässt.

2.2.2 Action Manager und BridgeTalk

Eine weitere nützliche Anwendung findet ExtendScript bei der Steuerung von Plug-ins und Filter, die nicht Teil des Photoshop-DOM sind. Dazu bedient man sich des sogenannten „Action Managers“, welcher Plug-ins mittels eines bekannten und einzigartigen vierstelligen Identifikators (genannt „Event-Code“) und gegebenenfalls zusätzlich definierter Deskriptoren aufrufen und alle aufzeichenbaren Aktionen ausführen kann. Ein zusätzlicher „Action Descriptor“ dient im allgemeinen der Parameterübergabe. Die Anweisung `executeAction()` in Listing 3 ruft beispielsweise das Standard-Filter „Gaußscher Weichzeichner“ auf. In diesem Fall ist der Deskriptor leer (`undefined`), da der Nutzer weitere Einstellungen manuell tätigen muss.

```
var typeId = charIDToTypeID('GsnB');
executeAction(typeId, undefined, DialogModes.ALL);
```

Listing 3: Aufrufen des Filter „Gaußscher Weichzeichner“.

Manche Arbeiten können sich über die verschiedenen Produkte der Creative Suite erstrecken. Hilfreich ist in diesen Fällen das Objekt „BridgeTalk“, eine von Adobe geschaffene Schnittstelle zur Interprogramm-Kommunikation (auch „cross-application

scripting“ genannt). Dazu wird einem Nachrichtenobjekt, die gewünschte Zielapplikation, der Nachrichtenkörper, welcher auch ausführbaren ExtendScript-Code enthalten kann und eine Funktion zur Ereignis- bzw. Antwortbehandlung mitgeteilt:

```
function infoFromInDesign(){
   var bt = new BridgeTalk;
   bt.target = "indesign";
   bt.body = "app.version"
   bt.onResult = function(resObj){
      alert("BridgeTalk-Ergebnis: " + resObj.body);}
   bt.send();}
```

Listing 4: Zugriff auf das InDesign-Objektmodell mittels BridgeTalk.

Durch das Senden der Nachricht kann auf Eigenschaften (hier `app.version`) eines fremden Dokumentobjektmodells zugegriffen werden. Diese Methode wird auch oft genutzt, um Skripte und Aktionen im Kontext einer anderen Host-Anwendung fernzustarten.

2.2.3 Externe Bibliotheken einbinden

Eine weitere mächtige Eigenschaft von ExtendScript ist die Fähigkeit dynamische Bibliotheken[17] einzubinden, um ausgelagerte Funktionalitäten nutzen zu können. Möglich wird dies durch die Verwendung des `ExternalObject`, der als Container und Manager der Bibliotheken auftritt. Dem Konstruktor wird der Name der Datei und gegebenenfalls notwendige Argumente für die Initialisierungs-Routine mitgeteilt. Es wird dabei zwischen direktem und indirektem Zugriff unterschieden (vgl. List. 5). Solche Bibliotheken müssen definierte Einstiegspunkte und weitere spezielle Eigenschaften besitzen, was aber nicht Gegenstand dieser Untersuchung sein soll.

```
/* direkter Zugriff auf eine C Funktion */
myLib = new ExternalObject("lib:myLibrary.dll");
myLib.doSomething();
/* indirekter Zugriff auf eine C++ Methode */
myOtherLib = new ExternalObject("lib:myOtherLibrary.dll");
var myObject = new MyNewClass();
myObject.doSomething();
```

Listing 5: Verwenden von externen Bibliotheken mit Hilfe des ExternalObject.

[17] Unter Windows assoziiert man damit die Dateiendung `.dll` und unter Mac OS `.framework`.

2.2.4 ExtendScript Toolkit

Das ExtendScript Toolkit wird kostenlos zum Download[18] angeboten und ist seit Version CS2 Bestandteil der „Adobe Creative Suite“. Es ist ein vollwertiges Entwicklungspaket bestehend aus einer in Abbildung 2 dargestellten IDE mit Syntaxhervorhebung im Editor, rudimentärer Codevervollständigung, einem Laufzeit-Datenbrowser sowie einer Reihe elektronischer Handbücher. Mit diesem Programm lässt sich ein Skript leicht debuggen, während es in der Host-Anwendung läuft. Neben Photoshop werden unter anderem die Adobe Produkte „After Effects“, „InDesign“, „Illustrator“ und „Premiere Pro“ unterstützt. Die Programme „Dreamweaver“ und „Flash Professional“, welche die JavaScript-Engine „SpiderMonkey“ nutzen, werden hingegen nicht unterstützt. Die Objektmodelle der Zielanwendungen unterscheiden sich teilweise erheblich von einander. Beispielsweise kann die Compositing-Software „After Effects“ mit einem „Document“ des Photoshop-DOM nichts anfangen.

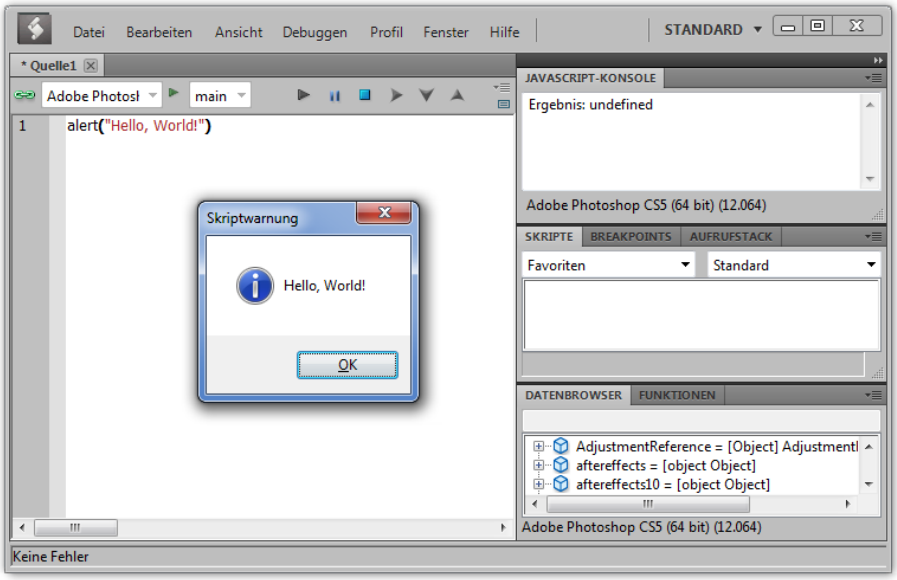

Abbildung 2: Benutzeroberfläche des ExtendScript Toolkit.

Das Programm ist in der Lage ein ExtendScript als Binärdatei zu exportieren, um den Code vor den Augen Dritter zu verbergen und kann so helfen vor Manipulationen zu schützen. Über das Menü `Datei > Skripten > Durchsuchen` werden Skripte in

[18] http://www.adobe.com/support/downloads/detail.jsp?ftpID=3205

Photoshop eingebunden. Mit den entsprechenden Dateiassoziationen des Betriebssystems genügt aber schon ein einfaches Ausführen der Skript-Datei an einem beliebigen Ort des Dateisystems.

2.2.5 Fazit

ExtendScript bieten eine günstige und einsteigerfreundliche Möglichkeit auf die vorhandene Funktionalität von Photoshop zuzugreifen. Von Vorteil sind die leicht zugänglichen Quelltext-Fragmente, die man im Internet finden kann und die Unterstützung einer stetig wachsenden Anwendergemeinde[19]. Als Nutzer sollte man sich aber der Tatsache bewusst sein, dass ein ExtendScript, anders als JavaScript innerhalb von Webbrowsern, Schreib- und Leserechte für das Dateisystem besitzt. Das kann unter Sicherheitsgesichtspunkten negativ gewertet werden. Schwierig gestaltet sich das Erstellen einer grafischen Benutzeroberfläche, da dies ohne eine visuelle Hilfe erfolgen muss. Wie schon bei den Plug-ins, fällt diese durch ihre Eigenschaft Photoshop zu blockieren negativ auf. Außerdem gestattet es das Photoshop-DOM nicht einzelne Pixel eines Bildes zu verändern. Theoretisch ist es möglich, mit einer 1x1 Pixel großen Auswahl über ein Bild zu iterieren. Die Geschwindigkeit, mit der das geschieht ist aber schon bei sehr kleinen Auflösungen nicht mehr akzeptabel.

2.3 Pixel Bender Filter

Mit der Technologie „Pixel Bender“ bietet Adobe eine einheitliche, programmübergreifende Basis zur Bild- und Videoverarbeitung in Form von Filtern an. Ursprünglich wurde Pixel Bender von Adobe in der Compositing- und Animationssoftware „After Effects“ zur Erzeugung aller OpenGL[20]-beschleunigter Effekte eingesetzt. Mit After Effects „CS4“ konnten die Anwender dann erstmals eigene Effekte laden. Da sich diese Technologie als sehr leistungsfähig und vielversprechend herausstellte, bietet Adobe Pixel Bender auch als Plug-in für Photoshop ab Version 11.0 an. Dieses Plug-in ist nicht Teil der Photoshop-Installation, kann aber kostenlos von Adobe bezogen werden. In Photoshop erscheint Pixel Bender unter dem Menü `Filter`. Im Pixel Bender-Kontext sind Filter beliebige, implementierte Algorithmen in Form von

[19] Zum Beispiel „The Photoshop Scripting Community“ unter http://www.ps-scripts.com/ .
[20] http://www.opengl.org/

Programmen, welche in der Computergrafik auch unter dem Begriff „Fragment-Shader“ bekannt sind. Diese werden vom Grafikprozessor einer 3D-Grafikkarte im Verlauf der Grafikpipeline in den „Shader-Einheiten“ ausgeführt. Dabei wird potentiell jedes einzelne Pixel eines Eingangsbildes verändert, um daraus ein Ausgangsbild zu erzeugen. Fragment-Shader werden deshalb häufig auch als „Pixel-Shader“ bezeichnet. Falls keine unterstützte Grafikhardware vorhanden ist, profitiert die Abarbeitungsgeschwindigkeit dennoch von der guten Skalierbarkeit des Pixel Bender-Plugins bei einem vorhandenen Mehrkern- und Mehrprozessorsystem. Auf Grund der hohen Nebenläufigkeit der Shader-Einheiten eigenen sich Filter am besten für Algorithmen, bei denen keine Korrelation der Ausgangspixel untereinander besteht.

2.3.1 Pixel Bender Toolkit

Pixel Bender-Filter, auch „Kernel“ genannt, werden in einem einfachen C-ähnlichen Dialekt (der „kernel language“) verfasst. Dieser orientiert sich an der Shader-Sprache „OpenGL Shading Language“ (GLSL) bzw. der „High Level Shading Language“ (HLSL), die Teil der „DirectX“-Programmierschnittstelle ist [Pixel Bender 2010]. Mit dem kostenlos erhältlichen „Pixel Bender Toolkit“ ist das Schreiben und Testen der Kernel möglich. Die Ergebnisse der Bildberechnungen, werden direkt in einem im oberen Teil der Benutzeroberfläche befindlichen Fenster dargestellt. Diese Arbeitsweise eignet sich, auch Dank der leicht erlernbaren Syntax, für das schnelle Implementieren von Algorithmen zur Bildmanipulation und der Evaluation der Ergebnisse, ohne sich in ein Framework einarbeiten zu müssen. Aufgrund dieser Art der Herangehensweise und der niedrigen Lernkurve bietet sich das Pixel Bender Toolkit auch für die Didaktik auf dem Gebiet der Medieninformatik an.

In der Pixel Bender-Sprache werden Elemente für die Nutzereingaben einfach durch Parameterdefinitionen erzeugt:

```
parameter int amount
<
   minValue : 0; maxValue : 4; defaultValue : 0;
>;
```

Listing 6: Ein Parameter in Pixel Bender-Notation.

Der Code im Listing 6 beschreibt beispielsweise einen Schieberegler für eine Ganzzahl im Bereich null bis vier. Gleichzeitig steht `amount` als Variable (vgl. List. 7) zur Verfügung. Die Funktion `evaluatePixel()` wird für jedes Pixel immer einmal aufgerufen, wobei in diesem Beispiel ein einfacher, durch Nutzereingaben gesteuerter „Weichzeichner“ programmiert wurde. Das aktuelle Pixel des Ausgangsbildes setzt sich aus den gemittelten Farbinformationen vier benachbarter Pixel zusammen.

```
...
input image4 src;
output pixel4 dst;
void evaluatePixel(){
   pixel4 col;
   float2 pos = outCoord();
   col += sample(src, pos + float2(amount, 0));
   col += sample(src, pos - float2(amount, 0));
   col += sample(src, pos + float2(0, amount));
   col += sample(src, pos - float2(0, amount));
   dst = col / 4.0;}
```

Listing 7: Ein einfacher Filter in Pixel Bender-Notation.

Die Benutzeroberfläche des Pixel Bender Toolkit ist in Abbildung 3 dargestellt. Darin wird ein Filter zur Laufzeit mittels Interaktion mit einem Steuerelement beeinflusst.

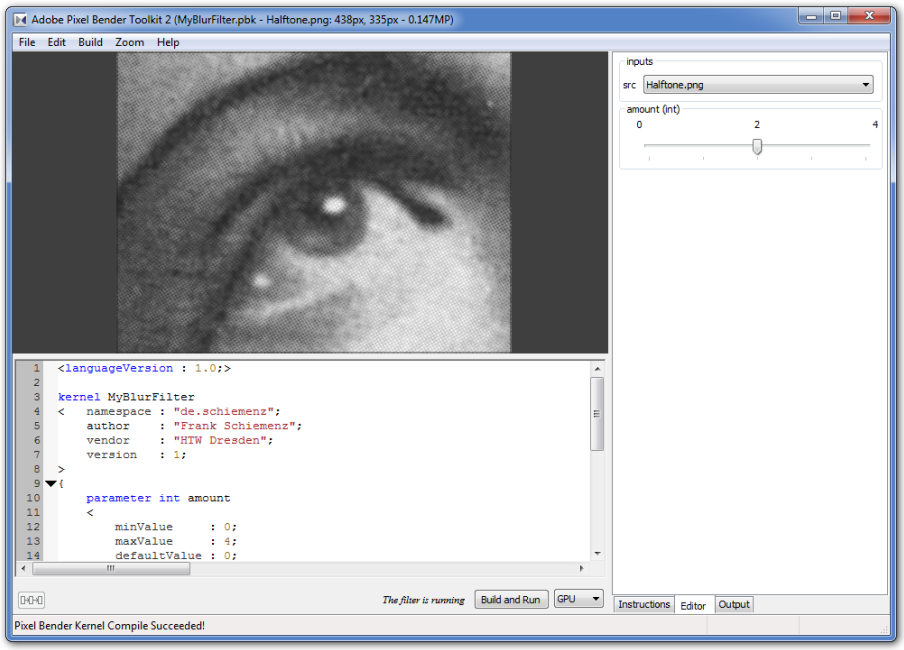

Abbildung 3: Ein Filter läuft im Pixel Bender Toolkit.

Mehrere Kernel können zu einem komplexen Filter vereinigt werden, was die Leistungsfähigkeit nochmals erhöht. Mittels der auf XML basierenden „Pixel Bender graph language“, werden die Kernel zu einem gerichteten azyklischen Graphen verbunden, welcher einen einzigen Eingang und Ausgang besitzt. Das Regelwerk dieser Graphensprache gestattet allerdings keine zyklische Abhängigkeiten der Kernel.

Die erzeugten Kernel-Dateien, werden im Ordner „Pixel Bender Files“ des Photoshop-Installationsverzeichnisses abgelegt und stehen dann dem Pixel Bender-Plugin zur Verfügung. Das Toolkit erlaubt zusätzlich den Export der einzelnen Kernel in ein kompiliertes Format. Die Kernelsprache sieht zwar die Verwendung von Schleifen zur Programmflusssteuerung vor, doch ein Export in Bytecode ist dann nicht gestattet. Auch ein Export von kompletten Graphen ist derzeit nicht möglich. Kompilierte Pixel Bender-Dateien lassen sich in Flex-Anwendungen einbetten (siehe Abschn. 2.4). Allerdings kann Adobes „Flash Player“ bzw. die Laufzeitumgebung „AIR“ diese Effekte derzeit nur mit Einschränkungen abspielen: Pixel Bender-Mischungen, -Filter und -Füllungen werden beim GPU-Rendern nicht unterstützt. Die zuvor besprochenen Kernel-Parameter können programmatisch verändert werden. Somit sind ereignis- und benutzergesteuerte Effekte möglich.

2.3.2 Fazit

Den Möglichkeiten dieser Technologie sind kaum Grenzen gesetzt. Mittlerweile gibt es neben Filter für Echtzeit-Filmeffekte auch Kernel für die Strahlverfolgung[21] („raytracing“), das Hochleistungsrechnen („number crunching“) und die Audio-verarbeitung[22], welche alle ausgiebig Gebrauch von der hohen Rechenleistung aktueller Grafikhardware machen.

Adobe arbeitet schon an einem Nachfolger: „Pixel Bender 3D“[23] soll zukünftig in der Lage sein, neben „Pixel-Shader“ auch „Vertext-Shader“ auszuführen, was dann Hardware-beschleunigte Berechnungen im dreidimensionalen Raum möglich macht. Welche Auswirkungen das auf Photoshop haben wird, bleibt abzuwarten. Doch bereits

[21] http://gingerbinger.com/2010/06/pixel-bender-raytracer/
[22] http://www.kevingoldsmith.com/labs/PBSynth-v1/
[23] http://labs.adobe.com/downloads/pixelbender3d.html

jetzt ist Pixel Bender, aufgrund der nativen Hardwareunterstützung im Vergleich zum klassischen Plug-in, die bessere Wahl für der Erstellung von Filtereffekten.

2.4 CSXS-Extensions

Extensions[24] sind im Grunde genommen AIR-Applikationen („Adobe Integrated Runtime“) und dürfen nicht mit dem gleichnamigen Plugin-Typ verwechselt werden. Die meisten Programme der „Creative Suite“ ab Version CS4 besitzen eine eingebettete Laufzeitumgebung mit einer speziellen Komponente zum Laden von CSXS-Erweiterungen („Creative Suite eXtensible Services“). In Photoshop werden diese unter dem Menü `Fenster > Erweiterungen` als Panel eingebunden, wo sie vielfältige Aufgaben übernehmen können. Bekannte Vertreter sind „Mini Bridge“ und „Kuler“[25], mit denen nach Bildern auf dem eigenen Rechner bzw. online nach Farbschemata gesucht werden kann. Die folgende Liste vermittelt einen Eindruck von den Anwendungsmöglichkeiten [PanelGuide 2010]:

- *Werkzeugpaletten.* Nutzer können eigene Panel erstellen, die den Anforderungen ihrer Arbeitsumgebung und der eigenen Arbeitsweise (dem sog. „workflow“) gerecht werden.

- *Video-Tutorials.* Anbieter können Videos auf Abruf zur Verfügung stellen, die Anwendern den Umgang mit Photoshop anhand von Beispielen und der Demonstration von Arbeitstechniken lehren.

- *Online-Dienste.* Speziell entwickelte Browser können Nutzern den Zugriff auf Dokumente gestatten, welche sich auf Webseiten mit Photogalerien oder Datenbanken befinden. Diese können direkt in Photoshop geöffnet werden.

- *Wissensaustausch.* Nutzer können mit anderen Nutzern interagieren, um innerhalb einer gemeinsamen Arbeitsumgebung Ideen zu kommunizieren und Daten auszutauschen.

[24] PatchPanel, die Vorgänger der Extensions werden in dieser Arbeit nicht besprochen.
[25] http://kuler.adobe.com/

2.4.1 Flex-Framework

Die Werkzeuge mit denen üblicherweise AIR-Anwendungen erstellt werden, stammen aus dem „Rich Internet Application"-Umfeld (RIA) und nutzen das Adobe Flex-Entwicklungsframework. Dieses findet auch bei der Entwicklung von Extensions Verwendung. Die wichtigsten Bestandteile des Frameworks sind:

- *ActionScript 3.0* (AS3) ist eine von Macromedia entwickelte imperative Programmiersprache mit klassenbasierter Objektorientierung und einer strengen Typisierung [Moock 2007]. Diese, auf dem Sprachstandard „ECMAScript 4"[26] basierende Skriptsprache, wird in einer Reihe von Adobe Produkten im Bereich Flash-Animation, Applikationsentwicklung für Mobilgeräte bis hin zur Webanwendung genutzt. Mit Hilfe spezieller Bibliotheken („Creative Suite ActionScript Wrapper"; CSAW) kann mittels ActionScript direkt auf das Objektmodell der jeweiligen Host-Anwendung zugriffen werden. Auf die Verwendung von ExtendScript ist man somit nicht mehr angewiesen.

- *MXML* ist eine auf XML basierende deklarative Auszeichnungssprache. Damit werden alle Steuer- und Datenkomponenten definiert. Dadurch versetzt es „den Entwickler in die Lage, selbst komplexe Benutzeroberflächen mit der Einfachheit von HTML zu erstellen." [Widjaja 2008] Bei konsequenter Nutzung von MXML ist eine klare Trennung der Darstellungslogik von der Programmlogik gewährleistet. So können Designer und Programmierer gleichzeitig an einem Projekt arbeiten. Beim Kompilieren der Extension wird der MXML-Quelltext in ActionScript-Bytecode überführt [Gassner 2010].

- *Creative Suite SDK* ist die auf dem Flex SDK basierende Werkzeug- und Komponentensammlung, mit der das Entwickeln von CSXS-Erweiterungen ermöglicht wird. Das Softwareentwicklungspaket kann kostenlos von den Adobe-Seiten[27] bezogen werden. Es enthält neben einem Kommandozeilen-Kompilierer die Klassenbibliotheken, die oben besprochenen Wrapper-

[26] http://www.ecma-international.org/activities/Languages/Language%20overview.pdf
[27] http://www.adobe.com/devnet/creativesuite.html

Bibliotheken für die verschiedene Programme der Creative Suite und Bibliotheken, die das Einbinden und Ausführen von ExtendScript ermöglichen. Zudem wird eine ausführliche Dokumentation mitgeliefert, welche anhand von Beispiel-Projekten in das Gebiet der „Creative Suite Extensibility" einführt [CSSDK 2011].

Im Kontext der betrachteten Extensions gehören darüber hinaus die folgenden Flash-Plattform-Technologien mit zum näheren Entwicklungsumfeld:

- *Adobe Flash Builder* (ehemals „Adobe Flex Builder") ist eine kostenpflichtige auf „Eclipse"[28] basierende integrierte Entwicklungsumgebung (IDE) zum Erstellen von CSXS-Erweiterungen aber auch „reichhaltiger" Internet-anwendungen. Das Programm besitzt einen grafischen WYSIWYG-Designmodus („What You See Is What You Get"), in dem per Drag-and-Drop und mit Hilfe von Eigenschaftsinspektoren die MXML-Dateien editiert werden, welche letztendlich die Benutzeroberfläche beschreiben. Darüber hinaus bietet der Flash Builder mit seiner automatischen Fehlererkennung, Code-vervollständigung und dem integrierten Debugger einige Features, die den Entwicklungsprozess wesentlich beschleunigen können. Er stellt somit eine bequeme Alternative zur rein Kommandozeilen-basierten Entwicklung mit dem Creative Suite SDK dar.

- *Adobe Integrated Runtime* (AIR) ist die Laufzeitumgebung zum Ausführen von Internet-Anwendungen auf dem Desktop. Diese enthält neben dem „Flash Player" auch ein auf dem WebKit[29] basierendes Webbrowser-Modul, welches der Darstellung von HTML und der Ausführung von JavaScript dient [Lott 2009]. Die schematische Darstellung in Abbildung 4 beschreibt die Hierarchie einer klassischen Adobe AIR-Architektur. In einigen aktuellen Produkten der „Creative Suite" ist bereits eine AIR-Laufzeitumgebung in Version 2.0 enthalten, die dort für das Ausführen der CSXS-Extensions und der Kommunikation dieser mit der Host-Anwendung verantwortlich zeichnet (vgl. Abb. 5).

[28] http://www.eclipse.org/
[29] http://www.webkit.org/

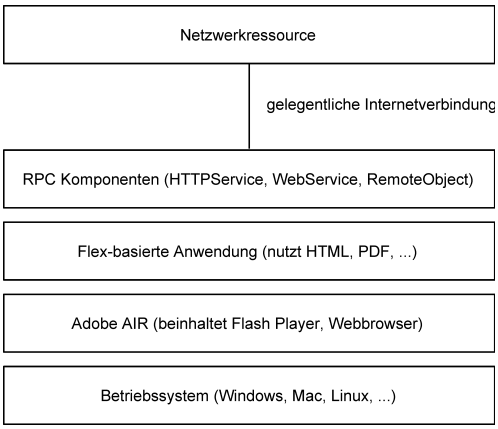

Abbildung 4: Klassische AIR Architektur.

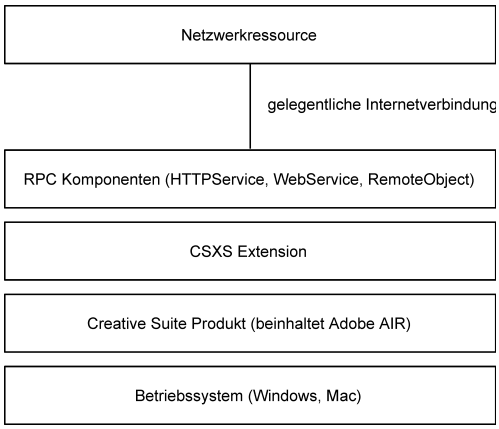

Abbildung 5: AIR und CSXS Extensions.

2.4.2 Hybrid-Extensions

Extensions können mit den klassischen Plug-ins gebündelt werden. Man spricht in einem solchen Fall von „Hybrid-Extensions“. Diese eignen sich besonders, wenn C++ Quellcode vorhanden ist, der nicht in die ActionScript-Sprache überführt werden kann oder soll. Auch CPU-intensive Berechnungen profitieren von einer Auslagerung als C++ Modul. Ein weiterer Entscheidungsgrund für eine solche Form der Extension ist gegeben, wenn eigene Menüs innerhalb der Photoshop-Oberfläche verwendet werden sollen. Diese sind nur durch die Verwendung bestimmter Plug-in Typen („Extension“) realisierbar. Hybrid-Erweiterungen bieten somit dem Entwickler die Möglichkeit, Technologien nach ihren Bedürfnissen auszuwählen und zu mischen.

2.4.3 Fazit

Bei der Nutzung des Flex-Technologieumfeldes ergeben sich für Extensions wesentliche Vorteile gegenüber der ausschließlichen Verwendung der C++ API, wie es bei dem klassischen Plug-in der Fall ist. Mit dem Flex-Framework ist es relativ leicht zusätzliche Komponenten, wie beispielsweise Web-Services einzubinden oder mit einem entfernten Server („Back-End“) im Sinne einer Datenanbindung zu kommunizieren. Darüber hinaus lassen sich ExtendScript-Dateien einbetten bzw. ausführen, was die Verwendung der in Abschnitt 2.2 beschriebenen Funktionalitäten möglich macht. Dies alles erhöht die Vielfältigkeit der Einsatzmöglichkeiten einer Extension im Vergleich zum Plug-in immens.

Ein weiterer wesentlicher Vorteil ist die gleiche Codebasis bei der Entwicklung auf unterschiedlichen Plattformen. Darüber hinaus garantieren Flex und AIR eine einheitliche grafische Darstellung in allen unterstützten Creative Suite-Produkten, wie man es von den Anwendungen aus dem RIA-Bereich gewohnt ist.

Positiv hervorzuheben ist zudem die Tatsache, dass Extensions die Gastanwendung nicht blockieren, also nicht-modal sind. Somit ist eine unkomplizierte und fließende Integration der Extensions in die Arbeitsprozesse der Nutzer möglich.

2.5 Rückblick und Bewertung

In den zurückliegenden Abschnitten dieses Kapitels wurden die aktuellen Photoshop-Erweiterungsmöglichkeiten und deren zugrundeliegenden Technologien eingeführt. Die gewonnenen Erkenntnisse werden an dieser Stelle zusammengetragen. Zur anschaulichen Gegenüberstellung ist es zunächst ratsam, die folgenden Fragen zu beantworten und als Vergleichskriterien heranzuziehen:

- *Anschaffungskosten: Wie ist hoch der zu leistende finanzielle Aufwand?* Die benötigten SDKs („Software Development Kits“) liegen Photoshop bei oder sind kostenlos verfügbar. Die Entwicklungsumgebungen können frei gewählt werden. Adobe bietet Entwicklern bei Bedarf Unterstützung in Form teurer Service-Pakete und zeitlich befristeter Entwicklerprogramme[30].

- *Einarbeitungsaufwand: Wie hoch ist der zeitliche Aufwand, um mit der Entwicklung im vernünftigen Rahmen beginnen zu können?* Aufgrund der von Adobe angebotenen Informationen in Form zahlreicher Handbücher und API-Dokumentationen, gestaltet sich der Einstieg relativ unkompliziert. Plugin-Entwickler sollten einen gewissen Programmier-Hintergrund besitzen, wenn es gilt, andere Technologien einzubinden. Pixel Bender stellt anfänglich keine großen Ansprüche, lebt aber durchweg von der Fähigkeit des Programmierers bildverarbeitende Algorithmen umsetzen zu können.

[30] http://www.adobe.com/devnet/aedp.html

- *Community-Unterstützung: In wieweit hat sich die Technologie bei den Endanwendern durchgesetzt, so dass diese Lösungen und Hilfestellungen anbieten?* Die klassischen Plug-ins besitzen zwar die größte Marktdurchdringung, dennoch fanden sich bei den Internetrecherchen überwiegend Skripte. Viele semiprofessionelle Nutzer setzen diese zur Steigerung ihrer Produktivität ein, was deren Verbreitung erklärt. Auf privaten Homepages, in Foren und auf Video-Plattformen trifft man zunehmend auf Präsentationen freier Pixel Bender-Effekte. Auch Hilfestellungen für die Entwicklung von Extensions finden sich aufgrund ihrer engen Verwandtschaft mit den weit verbreiteten Flex-Applikationen leicht. Die neuen Formen der Wissensvermittlung, wie die Arbeit so genannter „Adobe Technology Evangelists“[31] tun ihr übriges dazu.

- *Grafischer Editor: Welche Werkzeuge stellt Adobe für die Gestaltung der Benutzeroberfläche zur Verfügung?* Mit Hilfe des kostenlosen Werkzeugs „Adobe Configurator“[32] können Panel für Photoshop und das Desktop-Publishing-Werkzeug „InDesign“ erstellt werden. Der Anwender kann sich selbst eine Oberfläche zusammenstellen, indem er aus einer stark begrenzten Anzahl von Objekten auswählt. Die Gestaltung der Benutzeroberfläche im Sinne einer Flex-Anwendung ist erst mit dem kostenpflichtigen „Adobe Flash Builder“ oder dem reinen Designwerkzeug „Adobe Flash Catalyst“ möglich.

- *Multiplattformfähigkeit: Kann mit der Technologie plattformunabhängig entwickelt werden?* ExtendScripts und Pixel Bender-Kernel sind aufgrund der Tatsache, dass sie einfache Text-Dateien sind von Hause aus an keine Plattform gebunden. Extension-Projekte sind für Windows und Mac OS identisch, so dass auch hier plattformunabhängig entwickelt werden kann. Plug-ins werden üblicherweise in unterschiedlichen Entwicklungsumgebungen und für unterschiedliche Plattformen erstellt, was die Projekt-Verwaltung in einem gemeinsamen Repository erschwert.

[31] http://www.adobeevangelists.com/
[32] http://labs.adobe.com/downloads/configurator.html

- *Multiproduktfähigkeit: Kann man die Erweiterung applikationsübergreifend einsetzen?* Pixel Bender ist ein gutes Beispiel dafür, wie Programme ohne zusätzlichen Aufwand in anderen kompatiblen Umgebungen Verwendung finden können. Doch auch Extensions werden theoretisch von allen CSXS-fähigen Anwendungen der Creative Suite geladen, was diese Art der Erweiterung vielfältig einsetzbar macht.

- *Nutzerinteraktivität: Ist ein gewisser Grad an Interaktivität und Nutzersteuerung möglich?* CSXS-Extensions verhalten sich ganz wie Photoshop-eigene Panel und können intuitiv an andere Panel angedockt, sowie minimiert bzw. maximiert werden. In Verbindung mit interaktiven Inhalten sind diese, im Vergleich zu den anderen Erweiterungsmöglichkeiten, besser dazu geeignet die Aktivität und Konstruktivität des Nutzers anzuregen. Eine Eigenschaft, die besonders im Hinblick auf die Umsetzung des Tutorials von Bedeutung ist.

- *Objektmodellzugriff: Kann auf das Objektmodell der Host-Anwendung zugegriffen werden?* Extensions sind Aufgrund des CSAW-Konzeptes und ihrer Skript-Unterstützung dazu in der Lage. Plug-ins sind auf die „Callback"-Funktionalität spezieller Suites der PICA-API angewiesen, wohingegen Pixel Bender überhaupt keinen Zugriff auf das Host-System hat.

- *Grafikhardwareunterstützung: Ist die Erweiterungstechnologie speziell für die Verwendung vorhandener Grafikhardware konzipiert?* Einzig Pixel Bender bietet eine native GPU-Unterstützung, ohne auf zusätzliche Schnittstellen angewiesen zu sein.

Jeder Erweiterungsansatz hat seine spezifischen Vor- und Nachteile (vgl. Tab. 3). Daraus ergeben sich eher unterschiedliche Zielgruppen als klare Gewinner und Verlierer. Privatpersonen wenden sich dem schnell erlernbaren „Scripting" zu. Kommerzielle Anbieter werden weiterhin Plug-ins herausbringen, da diese noch die tiefgreifendste Integration in Photoshop möglich machen. Pixel Bender fristet, auf

Grund der kurzen Marktpräsenz ein Schattendasein. Dennoch machen sich einige Anwender aus dem wissenschaftlichen Umfeld die Leistungsfähigkeit dieser Technologie zu nutze. Der Trend geht aber in Richtung Extension bzw. Hybrid-Extension, da sich mit diesen die anderen Erweiterungsmöglichkeiten vereinigen lassen. Ein Effekt, der durchaus im Sinne von Adobe ist, trägt es doch zur weiteren Verbreitung der Flash-Plattform[33] bei.

In der nachfolgenden Tabelle 3 werden die in diesem Kapitel eingeführten Erweiterungsmöglichkeiten gegenübergestellt und hinsichtlich der oben besprochenen Kriterien bewertet.

	Plug-in Module	**Pixel Bender**	**Extend Scripts**	**CSXS Extension**
Anschaffungskosten	o	+	+	+ (o)
Einarbeitungsaufwand	–	+	o	o
Community-Unterstützung	o	o	+	+
Grafischer Editor	o	–	–	o (+)
Multiplattformfähigkeit	o	+	+	+
Multiproduktfähigkeit	o	+	–	o
Nutzerinteraktivität	o	o	–	+
Objektmodellzugriff	o	–	+	+
Grafikhardwareunterstützung	–	+	–	–

– nicht vorhanden/negativ
o zufriedenstellend
+ positiv
( ) Adobe Flash Builder

Tabelle 3: Übersicht über die Erweiterungsmöglichkeiten von Photoshop.

[33] http://www.adobe.com/flashplatform/statistics/

3 Prototypische Entwicklung der Extension „MyTutor“

Nachdem in Kapitel 2 das theoretische Fundament gelegt wurde, befassen sich die nun anschließenden Abschnitte mit den praktischen Arbeiten, welche zur Realisierung einer Extension geleistet wurden. Dabei wird die Vorgehensweise bei der Erstellung der Extension „MyTutor“ aufgezeigt, welche dazu gedacht ist dem Nutzer Tutorials im Photoshop-Kontext anzubieten.

3.1 Anforderungsanalyse

3.1.1 Inhaltliche Anforderungen

Es soll ein Tutorial erstellt werden, welches nicht darüber informiert wie Photoshop funktioniert, sondern lehrt, wie man als Nutzer einen bestimmten Effekt durch ausgewählte Werkzeuge des Grafikprogrammes erzielt. Zudem geht das Tutorial anwendungsorientiert vor und erhebt keinen Anspruch darauf, den kompletten Funktionsumfang von Photoshop abzubilden. Dabei wird in einem begrenzten Zeitfenster schrittweise durch ein Anwendungsszenario geführt, während dem Anwender mitgeteilt wird, was jeweils zu tun ist [Uhrig 2008].

Folglich sollte der Lernweg eine sequentielle und hierarchische Gliederung besitzen, die keinen offenen Interaktionsraum zulässt [Kerres 2001]. Die Einzelschritte bauen demnach aufeinander auf, ohne dem Nutzer Verzweigungen und Sprünge anzubieten. In diesem Kontext wird zudem von einer homogenen Zielgruppe ausgegangen. Das heißt, es werden ähnliche Medienkompetenzen und ein vergleichbarer Wissensstand der Lernenden vorausgesetzt. Das Vorwissen in Bezug auf das Photoshop-Umfeld wird deshalb als niedrig bis mittel eingestuft. Dies fand bei der Wahl des Lernzieles sowie bei der Gestaltung der informellen Texte Beachtung.

3.1.2 Technische Anforderungen

Die Applikation „MyTutor“ sollte in der Lage sein, einem Tutorial Zugriff auf das Photoshop-Objektmodell zu gestatten, um einen programmgesteuerten Lösungsweg aufzuzeigen und somit dem Lernenden eine Hilfestellung bieten zu können.

Des Weiteren war es wünschenswert, dass die Anwendung wiederverwendbar ist. Jedes Tutorial wird deshalb als unabhängiges Modul geladen. Somit kann man „MyTutor“ für andere Tutorials wiederverwenden, solange sich diese an einen bestimmten inneren Aufbau halten. Das heißt Aufgrund dieses Modularitätskonzeptes wird es für Lehrende leicht möglich sein, „MyTutor“ um eigene Inhalte zu erweitern.

Zusätzlich soll Anhand einer Übersetzungsfunktionalität die Einbindung eines Internet-Dienstes demonstriert werden.

Angesichts dieser Anforderungen wurden CSXS-Extensions in Verbindung mit ExtendScript für die Umsetzung der Anwendung ausgewählt. Diese Kombination verspricht eine schnelle Gestaltung der Benutzeroberfläche und die Möglichkeit der eleganten Auslagerung der Skript-Funktionalität. Als Plattform stand dabei ausschließlich Windows 7 mit Photoshop CS5 in deutscher Sprache zur Verfügung.

3.2 Auswahl des Tutorials

Zunächst wurde ein Lernziel definiert: nach einer Internetrecherche und dem Durcharbeiten mehrerer Online-Tutorials, fiel die Wahl auf das in englischer Sprache verfasste Tutorial „Reflective Liquid Type“[34] von Al Ward[35]. Das Tutorial vermittelt dem Leser die nötigen Techniken, um einer beliebigen Schrift einen metallischen oder flüssigen Eindruck zu verleihen, wie das Bildschirmfoto in Abbildung 6 illustriert.

Abbildung 6: Eine mögliche Lösung nach Bearbeitung des Tutorials.

Der „Workflow“ stellt auf der einen Seite zwar noch relativ geringe Anforderungen an den Nutzer, andererseits kommt eine Reihe verschiedener Techniken zum Einsatz, welche die Mächtigkeit der durch „MyTutor“ angebotenen Skript-Unterstützung demonstrieren helfen. Der Anwender wird auf dem Weg zur Lösung insbesondere mit

[34] http://www.photoshoplady.com/tutorial/reflective-liquid-type/340
[35] http://actionfx.blogspot.com/

den folgenden Schwerpunkten konfrontiert werden:

- Das *Text-Werkzeug* wird zum Platzieren und Verändern von Textinhalten genutzt.
- Das *Ebenen-Prinzip* ist elementarer Bestandteil von Photoshop und ein Kennzeichen moderner Bildbearbeitungsprogramme.
- *Filter* werden zum Erzeugen spezieller und manuell kaum reproduzierbarer Spezialeffekte verwendet.
- *Tonwertkorrektur* und *Gradationskurven* werden zum Anpassen des Tonwertumfanges bzw. zur Nachbearbeitung des Kontrastes benutzt.

Zunächst wurde das Online-Tutorial ins Deutsche übersetzt und sprachlich abgerundet. Dabei musste das Photoshop-typische Vokabular richtig übertragen werden. Um rechtlichen Konflikten aus dem Weg zu gehen und den unterschiedlichen Photoshop-Versionen Rechnung zu tragen, wurden alle relevanten Arbeitsschritte des Tutorials in eigenen Screenshots festgehalten. Da es sich bei „MyTutor“, hinsichtlich der verwendeten Medien, um eine lokale Anwendung handelt, wurde dazu mit PNG ein verlustfrei komprimiertes Grafikformat verwendet. Sollten zukünftig Bilder aus Internetquellen benutzt werden, ist neben einem Preloader zum Vorausladen der Bilder über die Verwendung einer verlustbehafteten Alternative (bspw. JFIF/JPEG) nachzudenken.

3.3 Wahl der Entwicklungsumgebung

Wie bereits in Abschnitt 2.4 besprochen wurde, ist der „Flash Builder“ (FB) als Werkzeug Teil der Flash-Plattformtechnologien. Adobe bietet Versionen für Windows und Mac OS X an, die jeweils als Plug-in für die Eclipse IDE oder als Vollversion vorliegen. Die Preise der aktuellen Produkte liegen dabei zwischen 250 USD für „Adobe Flash Builder 4.5 Standard“ bis ca. 700 USD für „Adobe Flash Builder 4.5 Premium“. Das Letztere enthält zusätzliche Komponenten zu Datenvisualisierung und ein „Test Automation Framework“ zum automatischen Client-seitigen Testen der Flex-Applikationen. Die Standard-Version ist für Studenten und Lehrende unter bestimmten

Voraussetzungen kostenlos. Adobe bietet darüber hinaus Programme für Entwickler (bspw. „Adobe Enterprise Developer Program“; AEDP) und strategische Partner zu besonderen Konditionen an. Diese erhalten dann unter anderem Zugriff auf Adobes „CS Extension Builder“[36], welcher das benötigte SDK und Werkzeuge für die Entwicklung, die Fehlersuche und das Verpacken von CSXS-Extensions in einem Flash Builder-Plugin vereinigt. Da vom Extension Builder zu Beginn dieser Arbeit noch keine öffentliche Testversion vorlag, wurde einzig die 60-Tage-Testversion des Flash Builder 4.5 Premium verwendet. Die Flex-IDE „Amethyst“[37] ist eine interessante Alternative zum Flash Builder, benötigt aber „Visual Studio 2010“ von Microsoft.

3.3.1 Erstellen des Flex-Projektes im Flash Builder

Nach der Installation des Flash Builder wurde zunächst das kostenlose „Creative Suite SDK“ in der aktuellen Version 1.5 eingebunden, da dieses die zur Erstellung von CSXS-Extensions benötigten Bibliotheken enthält (vgl. Abschn. 2.4.1). Das SDK basiert auf einem mittlerweile veralteten Flex 3 SDK (in Version 3.4 – aktuell ist Version 4.5.1). Dies äußert sich im wesentlichen darin, dass im Vergleich zu Flex 4 und dessen UI-Komponentenmodell „Spark“ hier ausschließlich „Halo“-Komponenten zur Verfügung stehen. Zudem fehlt die mit Flex 4 eingeführte Unterstützung für das Austauschformat „Flash XML Graphics“[38] (FXG) mit der Grafik-Primitive beschrieben werden können.

Da die Extension später auf Festplatteninhalte zugreifen muss, wurde das neu erstellte Flex-Projekt mit dem Anwendungstyp „Desktop“ auf eine AIR-Anwendung festgelegt. Folgende Flash-Komponentendateien des Creative Suite SDK wurden in das Verzeichnis `src` des Projektverzeichnisses im Flash Builder-Workspace kopiert und als Referenzbibliotheken in das neu erstellte Projekt „MyTutor“ eingebunden:

- `CSXSLibrary-2.0-sdk-3.4-public.swc` stellt die Implementation der „Creative Suite Extensible Services“ dar. Mit dieser Bibliothek werden die Kernfunktionalitäten angeboten, mit denen es möglich ist, Ereignisse an andere Extensions zu senden, ExtendScripts auszuführen oder Informationen über

[36] http://www.adobe.com/devnet/creativesuite/cs-extension-builder.html
[37] http://www.sapphiresteel.com/
[38] http://opensource.adobe.com/wiki/display/flexsdk/FXG+2.0+Specification

Photoshop bzw. der Host-Anwendung in Erfahrung zu bringen.

- `apedelta.swc` stellt die Implementation des „Adobe Player for Embedding“ (APE) dar. Die Datei wird benötigt, wenn CSAW-Bibliotheken benutzt werden.

- `csaw_photoshop.swc` diese Wrapper-Bibliothek dient dem Zugriff auf das Photoshop-Objektmodell indem es JavaScript-Anweisungen in ActionScript „verpackt“.

- `as3corelib.swc` ist eine Bibliothek des „corelib“-Projektes[39], welche einige Klassen und Hilfsmittel zur Verfügung stellt, um besser mit ActionScript 3.0 arbeiten zu können. Die Datei enthält Klassen zur Prüfsummenbildung, Bilderencoder und eine JSON-Serialisierung von der „MyTutor“ bei der Implementierung der Übersetzungsfunktionalität Gebrauch macht.

3.4 Gestalten der Benutzeroberfläche

3.4.1 Prinzipien der Gestaltung

Das Ziel der prototypischen Implementierung besteht zwar in erster Linie darin, einige technischen Möglichkeiten von Extensions aufzuzeigen, dennoch wurden bei der Gestaltung der Benutzeroberfläche wichtige Standards berücksichtigt. Es wurde insbesondere darauf Wert gelegt, dass die vier Prinzipien der Gestaltung eingehalten wurden [Williams 2008]:

- *Kontrast* ist die stark unterschiedliche Gestaltung von ungleichen Elementen, um Ähnlichkeiten zu vermeiden.

- *Wiederholung* zum dient der Organisation und verstärkt die Einheitlichkeit.

- *Ausrichtung* meint die saubere visuelle Verbindung mit anderen Elementen.

- *Nähe* zugehöriger Elementen dient ebenfalls der Organisation.

[39] http://code.google.com/p/as3corelib/

Darüber hinaus orientiert sich „MyTutor“ an den Photoshop-eigenen Applikationen wie „CS Review“ und „Kuler“, deren Gestaltung - selbst bei geöffnetem Panel - noch genug Raum zum Arbeiten in Photoshop zulässt.

3.4.2 Erstellen der Benutzeroberfläche mit MXML

Die Oberfläche der Extension „MyTutor“ präsentiert sich dem Nutzer mit einem oberen statischen Bereich zur Navigation und einem unteren Bereich, in dem der dynamische Inhalt in verschiedenen Ansichten angezeigt wird. Die Hauptnavigation erlaubt Zugriff auf die Ansicht „Information“ mit einleitenden Worten über den „MyTutor“, die Ansicht „Einstellungen“ mit einem Listenfeld zur Auswahl des gewünschten Tutorials und die Ansicht „Tutorial“, in welcher mittels weiterer Steuerelemente durch die einzelnen Schritte des aktuellen Tutorials navigiert und die Skript-Hilfe in Anspruch genommen werden kann. Die Screenshots in den Abbildungen 7 und 8 geben das von „MyTutor“ verwendete Konzept mit Hauptnavigations- und Inhaltsbereich wieder.

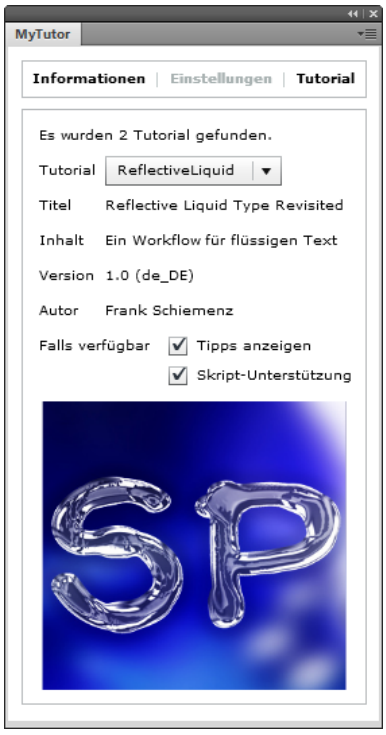

Abbildung 7: GUI – Einstellungen.

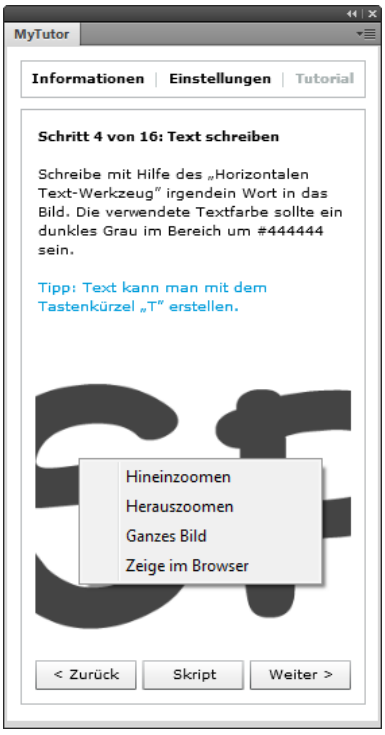

Abbildung 8: GUI – Tutorial.

MXML bietet zwei Konzepte zur Beschreibung von Ansichten an: einmal durch Status-Definitionen („view states“) und zum anderen durch Container („navigator container“).

Status-Definition eignen sich besonders für inkrementelle, das heißt aufeinander aufbauende Änderungen einer bestehenden Ansicht: das jeweils neu dazu gekommene Element einer Ansicht wird einem neuen Status zugeordnet. Nun kann von Status zu Status gesprungen werden, um eine Änderung der Ansicht zu bewirken. Da „MyTutor“ jeweils komplett verschiedene und unabhängige Inhalte in den Ansichtsbereichen hat, kommt stattdessen ein `ViewStack` (vgl. List. 8) zum Einsatz. Dieser speichert ganze Ansichten in jeweils einem `Canvas`-Container, einer normalerweise nicht sichtbaren Layout-Komponente, welche das Strukturieren von Steuerelementen ermöglicht. Zum Ändern der Ansichten muss nur noch der Inhalt dieses entstandenen „Ansichten-Stapels“ an eine Komponente weitergereicht werden, die für die Navigation zuständig ist. Dies geschieht durch das in MXML verbreitete Konzept des „Data-Binding“, bei dem, wie hier gezeigt, auf Bezeichner von zumeist dynamischen Komponenten wie auf Variablen zugegriffen wird:

```
<mx:ViewStack id="myViewStack">
   <mx:Canvas label="Informationen"></mx:Canvas>
   <mx:Canvas label="Einstellungen"></mx:Canvas>
   <mx:Canvas label="Tutorial"></mx:Canvas>
</mx:ViewStack>
<mx:LinkBar dataProvider="{myViewStack}"/>
```

Listing 8: Navigationskonzept mit ViewStack und LinkBar.

Die Extension „MyTutor“ nutzt zur Navigation eine `LinkBar`, wie der Quellcode in Listing 8 illustriert und bereits in Abbildung 7 und 8 zu erkennen war. Mögliche Alternativen dazu wären beispielsweise die untereinander austauschbaren Navigator-Komponenten `ButtonBar`, `TabBar` und `ToggleButtonBar`.

3.4.3 Erzeugen der Überblendungseffekte

Um das Wechseln von einer Ansicht zur anderen weniger abrupt wirken zu lassen, wurde auf das Effekt-Konzept von Flex zurückgegriffen. Effekte verändern Eigenschaften von visuellen Komponenten über einen festlegbaren Zeitraum hinweg. Diese lassen sich sowohl durch ActionScript-Klassenobjekte, als auch mit bestimmten MXML-Elementen und deren Attributen beschreiben. Einige der Standard-Effekte aus dem Paket `mx.effects` sind in Tabelle 4 zusammengetragen.

Effekt/Klasse	Beschreibung
Blur	Durch diesen Weichzeichnungseffekt werden die Umrisse der Komponente verwischt.
Fade	Animiert die Opazität-Eigenschaft einer Komponente, wobei die Überblendung entweder von transparent zu undurchsichtig oder umgekehrt erfolgt.
Glow	Mit dem Glühen-Effekt kann man eine Komponente aufglühen lassen.
Iris	Animiert das Effektziel durch Vergrößern oder Verkleinern einer rechteckigen Maske, die auf dem Ziel zentriert ist.
WipeLeft, WipeRight, WipeUp, WipeDown	Diese Klassen definiert einen Verwischungseffekt in die verschiedenen Richtungen.

Tabelle 4: Einige Standard-Effekte von Flex.

In „MyTutor“ kommen einfache Überblendungseffekte zum Einsatz. Der Quellcode in Listing 9 legt zwei Instanzen des `Fade`-Effektes fest, die beim Anzeigen bzw. Verbergen eines `Canvas`-Containers abgespielt werden. Dazu wird den Attributen `hideEffect` und `showEffect` des zu animierenden Containers der Bezeichner des jeweiligen Effektes mittels „Data Binding“ zugewiesen. Durch `Parallel` und `Sequence`-Elemente können mit MXML auch leicht komplexe Animationen geschaffen werden, die aus mehreren Effekten bestehen, welche zeitgleich bzw. nacheinander abgespielt werden.

```
<!-- Erzeugen von Effekten -->
<mx:Fade id="myFI" alphaFrom="0.0" alphaTo="1.0" duration="600"/>
<mx:Fade id="myFO" alphaFrom="1.0" alphaTo="0.0" duration="300"/>
<!-- Benutzen von Effekten -->
<mx:Canvas label="Tutorial" hideEffect="{myFO}" showEffect="{myFI}">
```

Listing 9: Der Überblendungseffekt in Flex.

3.4.4 Das Style-Konzept

Das aus der Web-Entwicklung bekannte Konzept der Stylesheetsprache „Cascading Style Sheets“[40] (CSS) wird auch in der Extension „MyTutor“ verwendet, um die Formatierungseigenschaften von GUI-Komponenten zu zentralisieren und von der Anwendungslogik zu trennen. Bei der Darstellung einer Komponente wird zunächst anhand eines Selektors ermittelt, ob die Definition einer Eigenschaft wie bspw. der

[40] http://www.w3.org/Style/CSS/

Schriftgröße oder der Hintergrundfarbe vorliegt. Ist das der Fall, wird diese dann anstatt der Standardeinstellung verwendet. Explizit festgelegte Eigenschaften der Komponenten haben dabei aber Vorrang. Der Flash Builder bietet dem Entwickler zudem die Möglichkeit aus den Komponenten-Eigenschaften CSS-Dateien zu extrahieren, um das Design wiederverwendbar zu machen. In Flex gibt es dabei eine Reihe verschiedener Möglichkeiten, Styles zu nutzen [Gassner 2010]:

- *Inline Styles* bezeichnen die Attribute einer MXML-Komponente.
- *Eingebettete Stylesheets* werden zentral zwischen besonderen Style-Elementen eines MXML-Dokumentes definiert.
- *Externe Stylesheets* sind in separate CSS-Textdateien untergebrachte Styles.
- *Kompilierte Styleheets* werden als SWF-Dateien erstellt und können so zur Laufzeit der Anwendung nachgeladen werden.

Für „MyTutor“ wurden einige Formatierungseigenschaften in der Datei `MyTutor.css` (vgl. List. 10) zusammengefasst und diese mit dem `source`-Attribut eines `Style`-Elementes in MXML eingebettet. Die betreffenden Komponenten verweisen mit dem `styleName`-Attribut auf den Selektor `myTut`, um die Formatierungen zu nutzen.

```
global{
   font-size:10;
   color:black;}
.myTut{
   backgroundColor:white;
   borderColor:white;}
```

Listing 10: Stylesheet-Definitionen der „MyTutor.css“-Datei.

3.5 Modellieren des Tutorial-Inhaltes mit XML

Die „Extensible Markup Language“ (XML) hat sich als eine Art Verkehrssprache im Internet etabliert und ist der De-facto-Standard, wenn Daten über verschiedene Applikationen hinweg auszutauschen sind. Bei der Strukturierung des Tutorial-Inhaltes findet XML deswegen auch hier Anwendung. Eine Forderung an die Umsetzung des

„MyTutor“ war die Modularität der einzelnen Tutorials. Diese sind in dem Ordner „Tutorials“ der MyTutor-Installation zu finden und können „wie Plug-ins“ nachgeladen werden, ohne die Extension erneut kompilieren zu müssen. Der Name der jeweiligen Tutorial-Verzeichnisse kann beliebig gewählt werden, solange sich eine in XML notierte Datei namens „tutorial.xml“ darin befindet. Ein Tutorial hat einen festgelegten hierarchischen Aufbau, der an gewisse Regeln gebunden ist:

- Das Wurzel-Element entspricht *genau einem* Tutorial, welches verschiedene Attribute besitzt, die alle vorhanden sein müssen.

- Das Tutorial kann aus *beliebig vielen* Schritten bestehen, wobei jedem Schritt eine Beschreibung per Attribut zugeordnet werden muss. Der relative Pfad zu einer ExtendScript-Datei kann optional angegeben werden.

- Ein Schritt besteht jeweils aus einem Inhalts-, Hinweis-, und Bild-Element, wobei der den aktuellen Schritt erklärende Text vorhanden sein sollte. Ein Bild-Element *muss* einen relativen Pfad zu einer Bilddatei und *kann* eine Beschriftung als Attribut besitzen.

Der mögliche Inhalt einer `tutorial.xml`-Datei sei hier wiedergegeben:

```
<?xml version="1.0" encoding="utf-8"?>
<tutorial title="HalloTutorial" version="1" author="FS"
locale="de_DE" description="" preview="\img.png">
   <step description="HalloSchritt" script="\schritt1.jsx">
      <text>Hallo, Welt!</text>
      <hint/>
      <image filename="\img.png" tooltip=""/>
   </step>
</tutorial>
```

Listing 11: Schematischer Aufbau einer „tutorial.xml“-Datei.

Die Datei `tutorial.dtd` beschreibt den schematischen Aufbau und somit das Regelwerk der XML-Datei mittels einer Dokumenttypdefinition (DTD), welche Bestandteil der XML-Spezifikation ist. Der Quelltext dieser Datei ist im Anhang der Arbeit zu finden. Mit Hilfe spezieller XML-Parser und Validatoren[41] kann überprüft

[41] Auch online möglich unter http://validator.w3.org/.

werden, ob die eigene `tutorial.xml`-Datei sowohl Wohlgeformtheit (hinsichtlich der Syntax), als auch Gültigkeit (hinsichtlich der Semantik) besitzt.

3.6 Programmieren der Extension mit ActionScript und E4X

Um mit den externen Daten in den XML-Dateien arbeiten zu können, ist es zunächst notwendig, diese der Extension zugänglich zu machen. Mit „ECMAScript for XML“ (E4X) existiert ein Spracherweiterungsstandard[42], welcher mittels einfacher Syntax erlaubt, XML-Daten auszulesen und zu verändern. Mit der aktuellen ActionScript Version 3.0 wird E4X vollständig unterstützt.

Der Programmcode, welcher den Zugriff auf die XML-Daten gestattet, ist als ActionScript-Klasse `TutLoader` ausgelagert. Diese fungiert somit als DAO-Klasse („Data Access Object“), da sie die technischen Details des Datenmanagements von der Programmlogik trennt. Die extrahierten Daten werden in Datenobjekte gespeichert, welche die einzelnen Tutorials und deren Schritte abbilden. Diese Datenobjekte sind die ActionScript-Klassen `Tutorial` und `Step`, welche sich gemeinsam mit der `TutLoader`-Klasse im Paket `de.schiemenz.MyTutor` und somit in einem gemeinsamen Namensraum befinden. Nachdem ein Objekt der Klasse `TutLoader` erstellt wurde, stehen ActionScript die Inhalte der Tutorials zur Verfügung. Der ActionScript-Code, welche die Programmlogik beschreibt, befindet sich innerhalb eines speziellen Script-Elements in der Datei `MyTutor.mxml` (vgl. List. 12).

```
<mx:Script>
   <![CDATA[
      /* ActionScript */
   ]]>
</mx:Script>
```

Listing 12: Einbettung von ActionScript in MXML.

Die Abhängigkeiten der Klassen im Package `de.schiemenz.MyTutor` illustriert das in der grafischen Modellierungssprache UML („Unified Modeling Language“) erstellte Klassendiagramm in Abbildung 9.

[42] http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-357.pdf

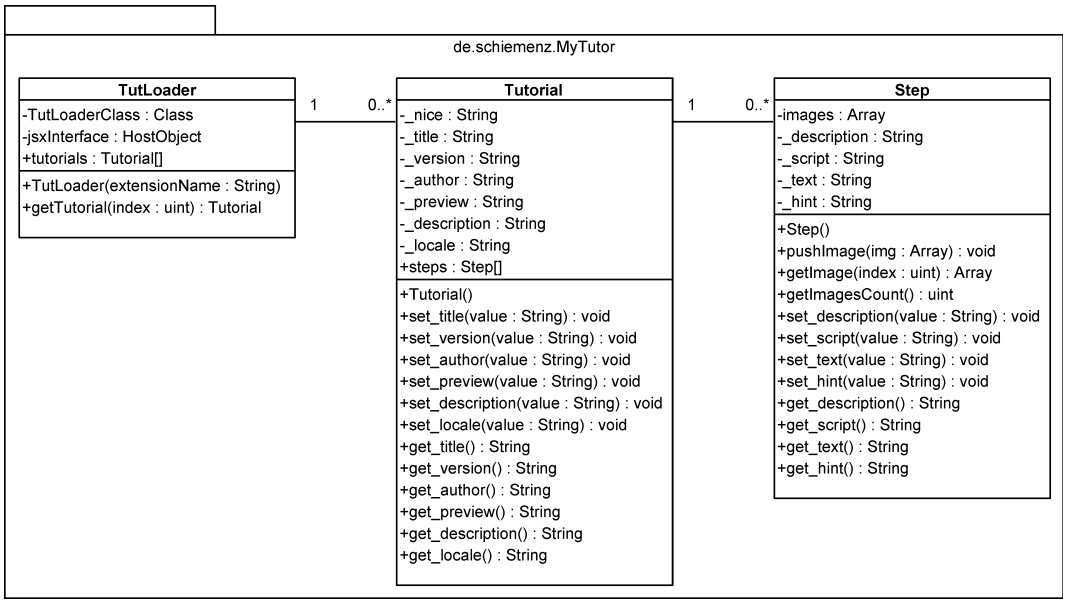

Abbildung 9: Das Klassendiagramm in UML-Notation.

Diese Klassen bilden den logischen Inhalt des Datei-Pfades „Photoshop-Installationspfad\Plug-ins\Panels\MyTutor\Tutorials“ bzw. der dort gefundenen XML-Dateien ab. Wie man erkennen kann, beinhaltet eine Instanz der Klasse `TutLoader` beliebig viele („0..*“) Tutorials, welche wiederum aus beliebig vielen Einzelschritten (`Step`) bestehen, welche in der jeweils beinhaltenden Klasse in Feldern (Arrays) organisiert sind. Die Klassen lassen sich bei Bedarf an Erweiterungen anpassen, wenn zusätzliche Medientypen bzw. mehrere Bilder pro Schritt unterstützt werden sollen.

3.7 Umsetzen der Skript-Unterstützung

Dieser Abschnitt bespricht, wie die Skript-Hilfe des „MyTutor“ realisiert wurde. Die Verwendung von CSAW, also der Zugriff auf das Photoshop-DOM mittels ActionScript, bot sich dafür nicht an. Der ActionScript-Quellcode müsste dann in die Anwendung kompiliert oder als SWF-Datei ausgelagert werden, was das Modularitätsgebot (vgl. Abschn. 3.1.2) verletzten bzw. einen Mehraufwand bei dem Schreiben von Tutorials bedeuten würde. Darüber hinaus kann ein Teil der Skript-Logik über ein Plug-in gewonnen werden, das bereits verwendbares ExtendScript erzeugt, wie weiter unten gezeigt wird. So wie schon in Abschnitt 3.5 angedeutet wurde, kann also jedem Schritt eines Tutorials eine externe ExtendScript- bzw. JavaScript-Datei zugeordnet werden.

Es gibt grundsätzlich zwei Möglichkeiten, wie Extensions JavaScript-Code durch ActionScript-Anweisungen aufgerufen können. Zum Einen werden Skripte direkt über die CSXS-Schnittstelle ausgeführt. Dabei erwartet das `CSXSInterface` eine ExtendScript-Datei mit dem Namen der Extension, welche sich im gleichen Verzeichnis wie die Extension befinden muss. Das Skript muss also im Kundenordner vorhanden sein, was evtl. hinsichtlich des Schutzes der Programmlogik nicht erwünscht ist. Das folgende Listing zeigt den Aufruf einer externen Funktion mit dieser Methode:

```
CSXSInterface.instance.evalScript("myFunction", myArgument);
```

Listing 13: Ausführen einer JavaScript-Funktion mittels CSXSInterface.

Zum Anderen kann man Skripte, ähnlich wie Bild- und Klangdateien, einbetten und mit Hilfe des `HostObject` quasi zu ActionScript-Objekten machen, mit denen JavaScript-Funktionen per Punktnotation aufrufbar sind (vgl. List. 14). Ein Vorteil dieser Methode ist, dass so der umgekehrte Weg möglich wird: JavaScript kann nun auch ActionScript ausführen[43]. Der folgende Quelltext zeigt wie ein eingebettetes Skript verwendet wird:

```
[Embed(source="myScript.jsx", mimeType="application/octet-stream")]
var myScript:Class;
var myHostObject:HostObject;
myHostObject = HostObject.getRoot(HostObject.extensions[0]);
myHostObject.eval(new myScript().toString());
myHostObject.myFunction(myArgument);
```

Listing 14: Ausführen einer JavaScript-Funktion mittels HostObject.

In beiden Fällen ist es möglich mit Rückgabewerten zu arbeiten. Für die Realisierung der Nutzer-Unterstützung des „MyTutor“ wird die zuletzt eingeführte Methode angewendet. Dazu macht sich das eingebettete Skript `MyTutor.jsx` die ExtendScript-Funktion `$.evalFile()` zunutze, welche die absolute Pfadangabe einer JavaScript-Datei als Argument erwartet. Diese Datei wird ausgewertet, d.h der enthaltene Code wird innerhalb von Photoshop ausgeführt. Relative Pfade werden, wie oben besprochen, in den einzelnen Schritten des Tutorials notiert und sind Dank der `TutLoader`-Klasse als absolute Pfade abrufbar, was die eben beschriebene Funktionalität ermöglicht.

[43] http://cookbooks.adobe.com/post_Communicating_between_JavaScript_and_the_Creative-17383.html

3.7.1 ScriptListener

Nachdem die Logik zur Skript-Ausführung programmiert worden war, konnten die Lösungen der einzelnen Schritte programmiert und in den zu referenzierenden Dateien abgespeichert werden. Für einen Teil der Aufgaben erwies sich der „ScriptListener“ als nützliches Werkzeug. Dieses „Automation“-Plugin liegt jeder Photoshop-Installation bei und muss nur noch in das dafür vorgesehene Verzeichnis kopiert werden, damit es aktiv wird. Das Plug-in „lauscht“ sodann auf Nutzer- sowie Skript-Aktionen und schreibt diese in zwei Textdateien mit. Diese befinden sich unter Windows als `ScriptingListenerJS.log` und `ScriptingListenerVB.log` auf dem Desktop. Auf einem Mac wird „AppleScript“ statt „VBScript“ erzeugt. Der in der JavaScript-Logdatei zu findende Quellcode ist so strukturiert, wie die in Abschnitt 2.2.2 eingeführten Anweisungen für den „Action Manager“. Auf diesem Wege ließen sich bspw. Photoshop-DOM-Zugriffe oder Filter-Aufrufe aufzeichnen und als eigene JavaScript-Funktionen kapseln, was den Entwicklungsprozess beschleunigte. Nachteilig sind die langen und für Menschen schlecht nachvollziehbaren Quelltexte, die generiert werden.

Wenn nötig, werden Maßeinheiten für Lineal- und Textgrößen (zu finden im `preferences`-Objekt des Photoshop-DOM) in Variablen gespeichert und auf definierte Werte gesetzt. Somit ist garantiert, dass unabhängig von den Voreinstellungen des Nutzers reproduzierbare Ergebnisse entstehen. Am Ende des Skripts müssen diese wieder auf die zuvor gespeicherten Einstellungen zurückgesetzt werden.

3.8 Einbinden der Bildbetrachtungskomponente

Eine weitere Aufgabe des „MyTutor“ ist die Darstellung der illustrierenden Bildschirmfotos bzw. Grafiken, die Bestandteil der Tutorials sein können. Eine mögliche Lösung sollte den unterschiedlichen Auflösungen der Bilder sowie dem begrenzten Platzangebot eines Panel innerhalb von Photoshop Rechnung tragen.

Als gangbarer Weg erwies sich die Nutzung der von Adobe zu Demonstrationszwecken und zur freien Verwendung angebotenen Flex-Komponente „Pan|Zoom“[44]. Diese ist, im Gegensatz zu den Standart-Komponenten, in der Lage dem Nutzer die Navigation

[44] http://www.adobe.com/devnet/flex/samples/fig_panzoom.html

räumlich organisierter Daten zu ermöglichen, wie es beispielsweise bei großen Diagrammen oder Karten notwendig ist (vgl. „Google Earth“). Dazu ermöglicht diese Komponente das „Panning“ und „Zoomen“, also das Verschieben bzw. das Vergrößern des Bildausschnittes zur Detailfindung mittels Mausinteraktion.

Die Migration des Quelltextes bzw. des Klassenpakets in das eigene Projekt erwies sich als erstaunlich einfach. Es mussten lediglich einige Stellen der in den Workspace übernommenen ActionScript-Klassedatei `ImageViewer.as` angepasst werden, die zur Darstellung nicht mehr benötigter GUI-Elemente und der hier fehlerverursachenden Ereignisbehandlung dienen. Des Weiteren wurde ein Kontext-Menü für die Mausinteraktion der Komponente übernommen und an die Bedürfnisse des „MyTutor“ angepasst (siehe Abb. 8). Zudem wurde die gefilterte Darstellung der Komponenten-Inhalte dauerhaft aktiviert und der Hintergrund durch eine `getStyle`-Abfrage auf die Hintergrundfarbe der Extension gesetzt, damit sich die Komponente besser in die bestehende Benutzeroberfläche einfügt.

3.9 Implementieren der Übersetzungsfunktionalität

Wie unkompliziert es ist, einer Extension durch Verwendung von Internetdiensten einen Mehrwert zu verleihen, soll dieser Abschnitt zeigen. Die Idee hinter der implementierten Übersetzungsfunktionalität ist relativ simpel: der Nutzer soll durch Markieren einer Stelle im Tutorial einen übersetzten Text per Tooltip präsentiert bekommen. Als Übersetzer kommt der unter der Bezeichnung „Google Translate API“[45] angebotene Dienst des US-Suchmaschinenbetreibers Google zum Einsatz. Mit den so genannten RPC-Klassen („Remote Procedure Call“), die sich in der Art unterscheiden, wie sie mit Servern kommunizieren, bietet Flex drei verschiedene Möglichkeiten an, wie auf Daten aus entfernten Quellen zugegriffen werden kann:

- `HTTPService`. Diese Klasse schickt einfache HTTP-Anfragen an Webserver, wie dies beispielsweise beim Aufrufen von Webseite oder dem Abrufen von RSS-Feeds der Fall ist, welche mit Text oder XML antworten.

[45] http://code.google.com/intl/de-DE/apis/language/translate/overview.html

- `RemoteObject`. Diese Klasse sendet und empfängt Daten im AMF-Format („Action Message Format“). Dieses von Adobe stammende binäre Format findet in vielen Server-Produkten[46] seinen Einsatz.

- `WebService`. Diese Klasse tauscht die Daten mittels XML-basierter SOAP-Nachrichten (auch „Envelopes“ genannt) aus.

Um auf den Übersetzungsdienst zugreifen zu können, wird eine wie im nachfolgenden Listing definierte Instanz der eben besprochenen `HTTPService`-Komponente benötigt.

```
<mx:HTTPService
   id = "googleTransService"
   url = "{googleTransString}"
   result = "onJSONLoad(event)"
   resultFormat = "text"/>
```

Listing 15: Definition eines HTTPService in MXML.

Die URL des Übersetzungsdienstes besteht aus einer Zeichenkette, in der unter anderem der Suchbegriff und die zu verwendenden Quell- und Zielsprachen kodiert sind. Die Antwortnachricht des Dienstes wird hier im textbasierten JSON-Format („JavaScript Object Notation“) zurückgegeben. Die Ereignisbehandlungsfunktion `onJSONLoad()` wertet diese nur noch aus und stellt das übersetzte Ergebnis als Tooltip der Text-Komponente dar, welche zuvor den Vorgang nach einem Maus-Ereignis initiiert hatte:

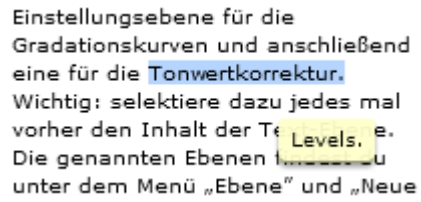

Abbildung 10: Die Übersetzungsfunktionalität in Aktion.

3.10 Erzeugen des Installationspaketes

Der letzte Abschnitt dieses Kapitels soll sich mit der Frage beschäftigen, wie die Extension zur Auslieferung an einen möglichen Nutzer vorbereitet wurde. Extensions

[46] Das sind zum Beispiel die Adobe Produkte LiveCycle Data Services, BlazeDS und ColdFusion.

wie auch Plug-ins können in eine ZXP-Datei („Zip Extension Package“) komprimiert werden. Das Packen, die Installation und die Verwaltung dieser übernimmt dabei der „Adobe Extension Manager“, welcher der Creative Suite beiliegt oder kostenlos von Adobe bezogen werden kann. Zum Verpacken wurde eine Textdatei mit der Endung MXI („Adobe Extension Information“) durch den Extension Manager ausgeführt, welcher das fertige Installationspaket erzeugte. Eine MXI-Datei besteht aus definierten XML-ähnlichen Sprachelementen, die unter anderem den Namen der Erweiterung, das Zielprodukt in dem die Extension laufen soll und die zu verpackenden Dateien bzw. das Quellverzeichnis festlegen. Der Inhalt, der für das Verpacken von „MyTutor“ verwendeten MXI-Datei, ist als anschauliches Beispiel im Anhang zu finden.

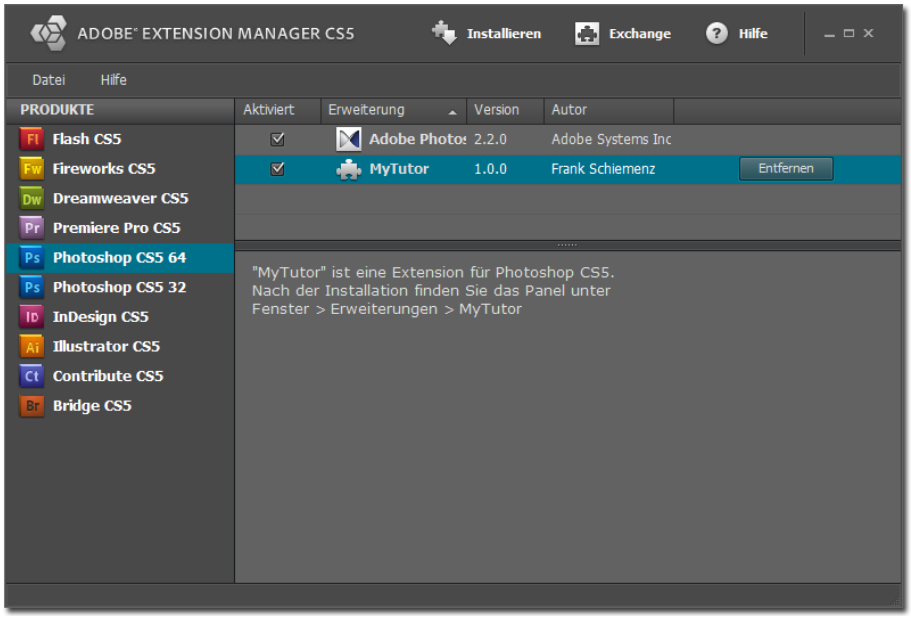

Abbildung 11: Der Extension Manager verwaltet MyTutor.

Adobe empfiehlt seinen Entwicklern, die Inhalte der Installationspakete mittels UCF (Universal Container Format), ein von Adobe spezifiziertes Container-Datei-Format, zu verpacken. Darauf wurde hier verzichtet. Ein Vorteil dieser Methode wäre das Verwenden von Zertifikaten zum Signieren der Anwendungen, um dem Nutzer ein gewisses Maß an Sicherheit zu bieten. Die Zertifikate können selbst erstellt oder von Adobe und bestimmten vertrauenswürdigen Anbietern, den sog. „Trusted Certificate Authorites“ (TCA), erstellt werden, was diese sich vergüten lassen.

4 Zusammenfassung und Ausblick

4.1 Zusammenfassung der Untersuchung der Erweiterungsmöglichkeiten

Ziel des theoretischen Teils dieser Bachelorarbeit war es, die verschiedenen Erweiterungsmöglichkeiten vorzustellen, wie sie Adobe für Photoshop anbietet. Dazu wurden diese anhand vieler anschaulicher Beispiele untersucht und so die jeweiligen Stärken und Schwächen herausgearbeitet, um sie am Ende einordnen zu können. Ein wesentlicher Teil der Untersuchungen widmete sich der Frage, welche Technologien von den einzelnen Erweiterungen verwendet und welche Werkzeuge in deren Umfeld dafür eingesetzt werden. Es wurde festgestellt, dass Plug-ins die tiefste Integration in Photoshop besitzen, was anhand der vielen, teils hoch spezialisierten Plugin-Typen ersichtlich wurde. Pixel Bender offenbarte ein großes Potential auf dem Gebiet der Bildverarbeitung. Doch letztendlich empfahlen sich einzig die auf der Flash-Technologie basierenden Extensions für die Umsetzung einer Tutorial-Anwendung. Mit Extensions sind, aufgrund einer modernen und flexiblen Nutzeroberfläche, der Möglichkeit der Integration anderer Erweiterungsansätze sowie der ExtendScript-Unterstützung leistungsfähige Erweiterungen möglich. Das Ergebnis der Untersuchungen spiegelt im gewissen Maße die Entwicklungen des letzten Jahrzehnts wieder: Adobe hat es verstanden, die nach der Übernahme von Macromedia[47] im Jahre 2005 erworbenen Flash-Technologien in das bestehenden Portfolio einfließen zu lassen.

4.2 Zusammenfassung der Entwicklung des „MyTutor“

Der praktische Teil der Arbeit beschäftigte sich mit der Realisierung des Applikations-Prototypen „MyTutor“, welcher als Photoshop-Extension umgesetzt wurde. Der Softwareentwicklungsprozess durchlief mehrere Phasen, welche das Vorgehen von der Konzeption des Tutorials bis hin zur Auslieferung als Installationspaket aufzeigten. Dabei bestätigten sich die in den Untersuchungen erkannten Vorteile der Extensions

[47] http://www.adobe.com/aboutadobe/invrelations/adobeandmacromedia.html

gegenüber den anderen Erweiterungsmöglichkeiten. Allerdings stellte sich bei der Implementation der Skript-Hilfe heraus, dass nicht alle Nutzeraktionen durch den „ScriptListener“ aufzeichenbar oder durch das Photoshop-DOM reproduzierbar sind. So kann man beispielsweise mit dem Pinsel-Werkzeug nicht programmgesteuert zeichnen: die Kreativität bleibt also noch dem Menschen vorbehalten. Mit Hilfe des Flash Builder konnte „MyTutor“ rasch und unkompliziert umgesetzt werden, womit die Zielsetzung als erreicht gelten kann. Auf die Unterstützung des „CS Extension Builder“ ist man also nicht unbedingt angewiesen. Herausfordernd und zugleich spannend war die Tatsache, dass eine Anzahl unterschiedlicher Programmiersprachen zum Einsatz kam und so verschiedene Herangehensweisen an die Realisierung der Extension ermöglichte.

4.3 Selbstreflexion

Der vorliegende Prototyp stellt kein fertiges Produkt im Sinne einer marktreifen Anwendung dar. Die im Folgenden genannten Ansätze für Verbesserungen bzw. mögliche Erweiterungen des Funktionsumfanges wären denkbar:

- Test der Extension unter einer Macintosh-Umgebung: Dazu müssen sicherlich die Pfadangaben angepasst werden. Rückblickend wäre es sicher sinnvoll gewesen, von Beginn an URIs anstatt der Windows-Pfadnamen zu verwenden.

- Ausbau der Übersetzungsfunktion: Die vorhandenen `locale`-Attribute der Tutorials und die Photoshop-Informationen könnten als Sprachpaare für den Übersetzungsdienst herangezogen werden. In diesem Zusammenhang empfiehlt es sich, dass Programm zu „lokalisieren“, d.h. den lokalen sprachlichen Gegebenheiten bspw. einer englischsprachigen Photoshop-Version anzupassen.

- Die Skripte der Tutorial-Schritte werden technologiebedingt unabhängig voneinander ausgeführt. Die eingebettete ExtendScript-Datei könnte globale Objekte zum Datenaustausch oder sogar eine eigene API zur Kommunikationen der Skripte untereinander anbieten. Unter Einbeziehung von Events ist dann u.U. eine neue Qualität der Interaktion und Nutzersteuerung möglich.

4.4 Ausblick

Mit dem unaufhaltsamen Wachstum des Mobilgeräte-Marktes unterliegt auch die Software-Branche einer rasanten Entwicklung. Durch das Erscheinen des „iPad“ von Apple sind, Dank eines für Privatanwender relativ großen Multi-Touch-Bildschirms, neue Bedienkonzepte möglich geworden, wie beispielsweise die Gestensteuerung mit bis zu fünf Fingern. Dies spiegelt sich auch in der Produktstrategie von Adobe wieder. So verwundert es kaum, dass erst kürzlich mit dem „Photoshop Touch SDK“ eine Werkzeugsammlung veröffentlicht wurde, die es den Entwicklern gestattet an völlig neue Formen der Photoshop-Erweiterungen zu denken. Android-, Blackberry- oder iOS-Geräte können nun mit Photoshop interagieren, sofern diese eine Internet- oder WiFi-Anbindung besitzen. Die ersten iPad-Apps[48] sehen schon sehr vielversprechend aus und demonstrieren was zukünftig alles möglich ist. So sind beispielsweise eBooks für Mobilgeräte denkbar, die es neben der Vermittlung von Lerninhalten gestatten, dass der Nutzer das Erlernte parallel in einer entfernten Photoshop-Umgebung umsetzt. Einen ähnlichen, wenngleich lokalen, Ansatz verfolgt schon die in dieser Arbeit entwickelte Extension „MyTutor“.

[48] Das sind die Titel „Eazel“, „Nav“ und „Color Lava“ von Adobe.

Literaturverzeichnis

[ADM 2003]
Adobe Systems: *Adobe Dialog Manager Programmer's Guide and Reference*.
http://partners.adobe.com/public/developer/en/acrobat/sdk/ADMReferenceGuide.pdf
Zuletzt aufgerufen am 31.07.2011.

[CSSDK 2011]
Adobe Systems: *Using the Adobe CreativeSuite 5.5 SDK*.
http://www.adobe.com/content/dam/Adobe/en/devnet/creativesuite/pdfs/CS_SDK_Guide.pdf
Zuletzt aufgerufen am 31.07.2011.

[ESTC 2010]
Adobe Systems: *JavaScript Tools Guide*.
http://www.adobe.com/content/dam/Adobe/en/devnet/scripting/pdfs/javascript_tools_guide.pdf
Zuletzt aufgerufen am 31.07.2011.

[Flanagan 2007]
Flanagan, David: *JavaScript: Das umfassende Referenzwerk*.
Köln. O'Reilly Verlag, 2007.

[Gassner 2010]
Gassner, David: *Adobe Flash Builder 4 and Flex 4 Bible*.
Indianapolis. Wiley Publishing, 2010.

[Kas 1999]
Thomas, Kas: *How to Write a Photoshop Plug-In*.
Ausgabe 4. MacTech Magazine, 1999.

[Kerres 2001]
Kerres, Michael: *Multimediale und telemediale Lernumgebungen: Konzeption und Entwicklung*. München. Oldenbourg Wissenschaftsverlag, 2001.

[Kriesinger 2005]
Kriesinger, Petra: *Photoshop mit JavaScript steuern*.
Poing. Franzis Verlag, 2005.

[Lott 2009]
Lott, Joey; Rotondo, Kathryn; Ahn, Sam ;Atkins, Ashley: *Adobe Air im Einsatz*.
München. Hanser Verlag, 2009.

[Moock 2007]
Moock, Colin: *Essential ActionScript 3.0*.
Sebastopol. O'Reilly Media, 2007.

[PanelGuide 2010]
Adobe Systems: *Photoshop Panel Developer's Guide*.
http://download.macromedia.com/pub/developer/photoshop/sdk/photoshop_panel_developers_guide_cs5_win.zip
Zuletzt aufgerufen am 31.07.2011.

[Pixel Bender 2010]
Adobe Systems: *Pixel Bender Developer's Guide*.
http://www.adobe.com/content/dam/Adobe/en/devnet/pixelbender/pdfs/pixelbender_guide.pdf
Zuletzt aufgerufen am 31.07.2011.

[Schewe 2000]
Schewe, Jeff: *10 Years of Photoshop*.
http://www.schewephoto.com/pei/pshistory.pdf
Zuletzt aufgerufen am 31.07.2011.

[Uhrig 2008]
Uhrig, Martin: *Entwicklung eines Standardkonzepts zur Erstellung interaktiver Tutorials für Viamedici Software GmbH*. Hochschule Karlsruhe – Technik und Wirtschaft. Diplomarbeit, 2008.

[Widjaja 2008]
Widjaja, Simon: *Rich Internet Applications mit Adobe Flex 3*.
München. Hanser Verlag, 2008.

[Williams 2008]
Williams, Robin P.: *Design und Typografie: ... Für Dich!- Die überraschend einfachen Gesetze*. München. Addison-Wesley Verlag, 2008.

Anhang

A.1 Dokumenttypdefinition der Tutorial-Datei

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT hint (#PCDATA)>
<!ELEMENT image EMPTY>
<!ATTLIST image
      filename CDATA #REQUIRED
      tooltip CDATA #REQUIRED
>
<!ELEMENT step (hint|image|text)*>
<!ATTLIST step
      description CDATA #REQUIRED
      script CDATA #IMPLIED
>
<!ELEMENT text (#PCDATA)>
<!ELEMENT tutorial (step+)>
<!ATTLIST tutorial
      author CDATA #REQUIRED
      description CDATA #REQUIRED
      locale NMTOKEN #REQUIRED
      preview CDATA #REQUIRED
      title CDATA #REQUIRED
      version NMTOKEN #REQUIRED
>
```

A.2 Adobe Extension Information-Datei

```
<?xml version="1.0" encoding="UTF-8"?>
<macromedia-extension name="MyTutor" version="1.0.0" type="object">
   <author name="Frank Schiemenz"/>
   <products>
      <product familyname="Photoshop" version="12.0" primary="true"/>
   </products>
   <description>
      <![CDATA[
         "MyTutor" ist eine Extension fuer Photoshop CS5.
      ]]>
   </description>
   <ui-access>
      <![CDATA[
         Nach der Installation finden Sie das Panel unter
         Fenster > Erweiterungen > MyTutor
      ]]>
   </ui-access>
   <license-agreement>
      <![CDATA[
      ]]>
   </license-agreement>
   <files>
      <file source="MyTutor" destination="$panels"/>
   </files>
   <configuration-changes>
   </configuration-changes>
</macromedia-extension>
```

Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Bachelorarbeit zum Thema „Photoshop Erweiterungen – Realisierung eines Tutorials mit Hilfe des Flash Builder und Extensions“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Dresden, am 15. August 2011

…….....….

Frank Schiemenz