# Dynamisches Augmented Reality mittels nachladbaren 3D- und Markerdaten

Peter Wozniak*, Katharina Mehner-Heindl*, Tom Rüdebusch*, Felix Müller †

* Hochschule Offenburg
Badstr. 24
77652 Offenburg
{peter.wozniak|katharina.mehner-heindl|ruedebusch}
@hs-offenburg.de

† visionsbox GmbH
Kornstraße 1
77652 Offenburg
felix.mueller@visionsbox.de

**Zusammenfassung:** Um 3D- und Markerdaten zur Laufzeit in einer Augmented Reality App nachzuladen, wurde auf Basis von Unity3D und Vuforia eine Software-Architektur entwickelt. Diese Architektur wurde benutzt um beispielhaft eine iOS und Android App zu implementieren.

**Stichworte:** Augmented Reality, Softwarearchitektur, Unity3D, Vuforia

## 1 Problemstellung

Mittels der plattformübergreifenden Entwicklungsumgebung Unity3D [Uni] und der von Qualcomm bereitgestellten AR Programmbibliothek Vuforia [Vuf] lassen sich schnell und einfach mobile AR-Anwendungen für Android und iOS erstellen. Vuforia ermöglicht auf Mobilgeräten den Zugriff auf die Kamera und ermöglicht ID- bzw. Mustermarker basiertes Tracking [Tö10][MBRS11][WS09]. Vorgefertigte Unity3D-Skriptkomponenten minimieren den Entwicklungsaufwand für eine AR-App.

Standardmäßig werden Vuforia und Unity3D basierte AR Apps mit allen notwendigen Daten kompiliert und ausgeliefert. Spätere Änderungen an den Daten müssen als Update der App verteilt werden. Eine Internetverbindung und entsprechende Schnittstellen der Unity3D und Vuforia API ermöglichen jedoch auch Apps, die Daten direkt aus dem Internet herunterladen und dynamisch einbinden. Je nach Anwendungsfall können so Inhalte aktualisiert, ausgetauscht bzw. erweitert werden ohne die eigentliche App upzudaten.

Eine beispielhaft umgesetzte App basiert auf folgendem Szenario (Abb. 1) [Woz13] und kann als Basis für weitere Entwicklungen dienen.

- Die App startet und erwartet vom Benutzer eine einfache alphanumerische Code-Eingabe.
- Der Code entscheidet über die anzuzeigenden AR-Inhalte und die für das Tracking zu nutzenden Markerdaten.
- Der Code wird benutzt, um eine URL zu generieren.

- Diese URL verweist auf ein Webserververzeichnis, innerhalb dessen Daten hinterlegt sind.
- Eine XML-Konfigurationsdatei wird heruntergeladen und ausgewertet.
- Je nach Bedarf werden 3D-Daten und Vuforia Markerdaten heruntergeladen und auf dem Gerät gespeichert.
- Die Daten werden für die AR-App genutzt bzw. interpretiert.

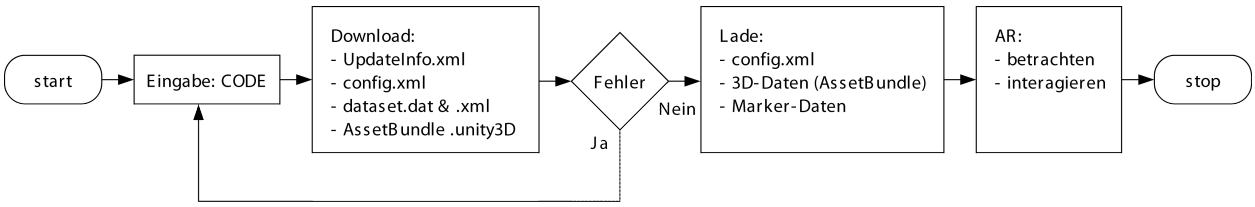


Abbildung 1: Programmablauf der App

## 2 Umsetzung

Die Umsetzung der Funktionalität basiert auf dem Unity3D Entwicklungskonzept.

- Szenen können Anwendungen funktional gliedern
- Skriptkomponenten dienen der Implementierung
- AssetBundle dienen dem Datenaustausch und ersparen die Implementierung eigener Parser

Die beispielhaft implementierte App besteht aus zwei Szenen, die das obige Szenario umsetzen. Beim Start wird die für das Abfragen des Codes sowie das anschließende Herunterladen der Daten zuständige Szene geladen. Im Anschluss wird die zweite Szene geladen. Diese interpretiert die nun lokal vorliegenden Daten, lädt die 3D-Daten und initialisiert die AR-Funktionalität (Abb. 2). Die notwendige Funktionalität lässt sich als endlicher Automat formulieren. Die Skriptkomponenten bilden den aktuellen Zustand ab und stellen Methoden für die Zustandsübergänge bereit. Für die Implementierung der Anwendungslogik wurde das Gof-Entwurfsmuster ‚Zustand' (State-Pattern) [GHJV96] gewählt. Dieses erlaubt es, Verhalten bzw. Programmanweisungen in Form von Zustands-Klassen zu kapseln. Jede dieser Klassen stellt einen Zustand und dessen spezifisches Verhalten dar.

Die Demo-App basiert maßgeblich auf der Unity3D und Vuforia API. Der Einsatz anderer 3D- und AR-Funktionsbibliotheken sowie Datenformate ist prinzipiell möglich.


Abbildung 2: Demo App mit dynamisch herunterladbaren Inhalten

## 3 Verwandte Arbeiten

Dynamisch zur Laufzeit einer Anwendung heruntergeladene AR-Inhalte sind keine Neuheit. Neben den inhaltlich statischen Apps existieren Apps mit dynamischen Inhalten. AR-Browser Apps erlauben ebenfalls dynamisch zur Laufzeit geladene Inhalte. Vergleichbar mit einem Web-Browser, können Inhalte geladen werden. Eine einheitliche Beschreibungssprache wie HTML existiert jedoch derzeit noch nicht und so unterscheiden sich die Möglichkeiten der einzelnen Apps deutlich. Eine Erweiterung der Grundfunktionalität, durch nachladbare und ausführbare Programmanweisungen, kann nur in den engen Grenzen erfolgen, in denen diese vorhergesehen und ermöglicht wurden. Vergleichbar mit JavaScript in Browsern können somit nicht beliebige Funktionalitäten umgesetzt werden. Beispiele für AR-Browser-Apps sind Wikitude, Layer, Argon oder Aurasma [ARB]. Im Gegenzug zu bestehenden AR-Browser-Apps bietet die Implementierung einer eigenen AR-App den Vorteil eines prinzipiell beliebig wählbaren Funktionsumfangs. Insbesondere die Anwendungslogik und Interaktion mit dem Benutzer kann so frei gestaltet werden. Der Anwendungsfall kann also selbst definiert werden und muss sich nicht im Rahmen der Möglichkeiten eines AR-Browser abspielen.

## 4 Ergebnisse

Dynamische Inhalte für AR-Apps auf Basis von Unity3D und Vuforia sind möglich. Das Herunterladen von Inhalten zur Laufzeit erlaubt es, Anwendungen jederzeit mit neuen Inhalten

zu versehen. Verschiedene Anwendungsfälle mit dynamischen AR-Inhalten sind denkbar:

- Inhalte werden erst geladen wenn sie tatsächlich benötigt werden und müssen nicht als Teil des Programmpaketes ausgeliefert werden.
- Inhalte können bei jedem Programstart neu geladen bzw. aktualisiert werden. Kurzfristige Aktualisierungen sind so möglich.
- Die Art der geladenen Inhalte kann vom Benutzer gewählt.

Unity3D und Vuforia ersparen viel Entwicklungsarbeit und ermöglichen es AR-Apps für Android und iOS zu erstellen. Das Nachladen von Programmcode ist unter iOS leider nicht möglich. Zugunsten einer einheitlichen Implementierung für beide Plattformen wurde deshalb auf diese Möglichkeit verzichtet. In diesem Zusammenhang wäre ein Einlenken Apples notwendig und wünschenswert, um dynamisch nachladbaren Programmcode auf der iOS Plattform zu ermöglichen. Für die Zukunft wäre die Etablierung einer einheitlichen Architektur wünschenswert. Anstelle von Insellösungen einzelner Hersteller wären funktional standardisierte und erweiterbare AR-Browser mit einer gemeinsamen Inhaltbeschreibungssprache zu bevorzugen. Ähnlich wie beim World Wide Web müssten sich Anbieter von Inhalten nicht auf eine spezialisierte Softwarelösung festlegen.

## Literatur


[ARB] *Wichtigste AR-Browser Apps*. http://www.direktplus.de/archiv/dialogmarketing-news/dialogmarketing-news-2012/augmented-reality-browser/.

[GHJV96] Gamma, Erich, Richard Helm, Ralph Johnson und John Vlissides: *Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, 1996.

[MBRS11] Mehler-Bicher, Anett, Michael Reiss und Lothar Steiger: *Augmented Reality. Theorie und Praxis*. Oldenbourg Verlag, 2011.

[Tö10] Tönnis, Marcus: *Augmented Reality. Einblicke in die Erweiterte Realität*. Springer Verlag Berlin Heidelberg, 2010.

[Uni] *Unity3D*. http://unity3d.com/.

[Vuf] *Vuforia*. http://www.qualcomm.com/solutions/augmented-reality.

[Woz13] Wozniak, Peter: *Dynamisches Augmented Reality für Mobilgeräte auf Basis von Unity3D und Vuforia*, 2013. Masterthesis.

[WS09] Wagner, Daniel und Dieter Schmalstieg: *History and Future of Tracking for Mobile Phone Augmented Reality*. In: *International Symposium on Ubiquitous Virtual Reality (ISUVR 2009)*, 2009.