AR-Chess: Augmented – Reality – Schach

Björn Bendig
Matrikel-Nr.: 332107
bjoern.bendig@informatik.hs-fulda.de

Sebastian Brehl
Matrikel-Nr.: 731943
sebastian.brehl@informatik.hs-fulda.de

Robert Schwarz
Matrikel-Nr.: 132138
robert.schwarz@informatik.hs-fulda.de

ZUSAMMENFASSUNG
Entworfen wurde ein Schachspiel entworfen, bei dem die Figuren dreidimensional auf einem Bildschirm dargestellt werden. Genutzt wurde hierbei das „JavaScript Augmented Reality Toolkit“, sowie das X3dom Framework. Die Figuren in der realen Welt sind Damefiguren, auf welche die 3D-Figuren mit Hilfe der JSAR-Markern „projiziert“ werden. Durch klicken auf eine angezeigt Schachfigur werden Informationen über diese Bereitgestellt.

STICHWÖRTER
Augmented Reality; Schach; JavaScript; JSAR; X3dom

EINFÜHRUNG
Die Grundidee hinter AR-Chess besteht darin, Schach in die Welt der Augumented Reality(AR) zu überführen. Inspiriert von neuesten Entwicklungen auf diesem Gebiet sollte mit AR-Chess eine Anwendung geschaffen werden, die es erlaubt die reale und die digitale Welt miteinander zu verbinden und damit etwas Neues zu schaffen. Der Vorteil gegenüber einem herkömmlichen Brettspiel liegt augenscheinlich im (zumindest zu Anfang) größeren Unterhaltungswert. Durch die Möglichkeit der Interaktion mit der Anwendung können darüber hinaus zusätzliche Funktionen in ein Spiel integriert werden. So wurde in AR-Chess eine Lernhilfe integriert, bei der Informationen über eine ausgewählte Figur angezeigt werden. Um Plattformunabhängigkeit zu garantieren, wurde AR-Chess mit JavaScript entwickelt und kann somit in unterschiedlichen Browsern auf verschiedensten Endgeräten ausgeführt werden. Zusätzlich zu einem solchen Gerät, werden ein Schachbrett und 32 Objekte (z.B. Dame-Steine), die mit den verschiedenen JSAR-Markern versehen sind.

Das Grundgerüst
Der zugrunde liegende Code der Anwendung basiert zum Teil auf dem x3dom Beispiel „Augumented Reality via WebRTC und JSARToolkit“. Das Beispiel erkennt Marker und projiziert einen 3D-Globus darauf. Dafür wird zu Anfangs ein 3D-View mittels x3dom erstellt. In diesem wird hintergründig ein Video dargestellt, welches durch WebRTC direkt aus dem Stream der sich am Endgerät befindenden Kamera gelesen wird. Die Frames dieses Videostream werden dann automatisch vom JSAR-Toolkit auf diesem bekannte Marker gescannt. Wird ein Marker gefunden, wird ein Event geworfen und der Marker in die in einer Liste gespeichert. Diese Liste wird bei jedem Durchlauf der Animations-Schleife auf Existenz von Markern geprüft. Ist diese gegeben, wird ein vom Marker abhängiges Objekt auf diesen gezeichnet. Dazu wurde ein Debug Fenster angezeigt. Es stellte die Tracking Funktion des Toolkits dar und zeigte an, ob und welche Tracker im Bild gefunden wurden.

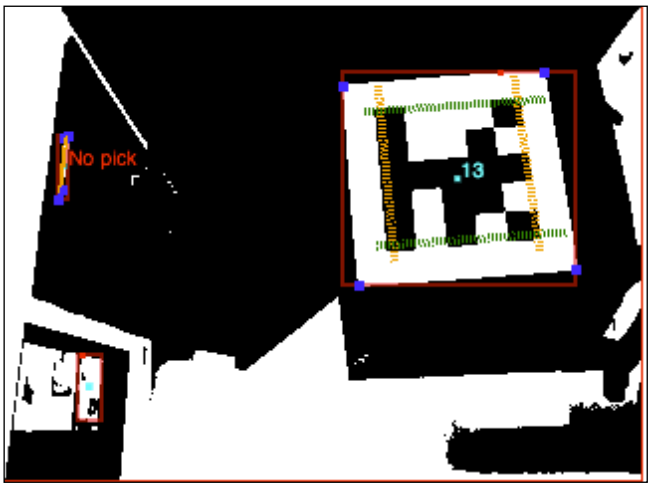

Abbildung 1. Debug Fenster mit gefundenem Marker 13.

Eine weitere Funktion des Beispiels war der „Binarization Threshold“. Mit diesem kann die Lichtempfindlichkeit des Detektors angepasst werden. Das von der Kamera gelieferte Bild wird einfach abgedunkelt oder erhellt.

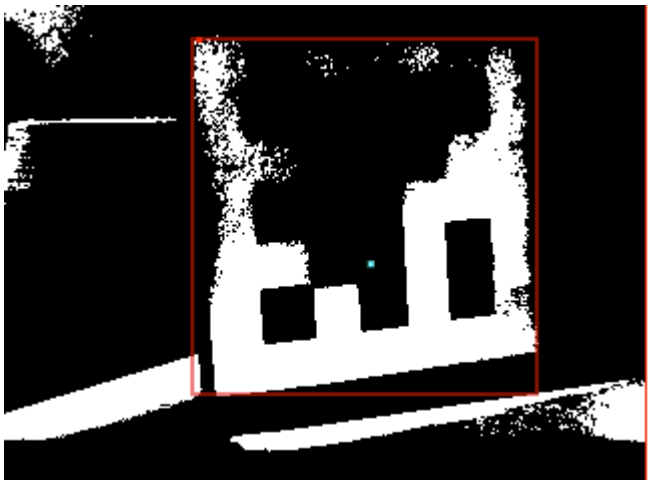

Abbildung 2. Zu niedrig eingestellter Threshold.

ANZEIGEN DER SCHACHFIGUREN
Das Anzeigen der Figuren die Zentrale Aufgabe des Projektes. Die Figuren sollten gut aussehen, an der richtigen

Stelle angezeigt werden und die Performance der Anwendung musste gewahrt bleiben.

Erstellen der 3D Modelle

Die Modelle der Schachfiguren sind von fremden Designern und wurden, da lizenzfrei, von uns genutzt. Sie wurden mit Hilfe der Open Source 3D-Modelling Software „Blender“ in .x3d Dateien gewandelt. Dies geschieht ohne weiteres direkt über die Exportfunktion von Blender. Genauere Informationen hierüber sind auch auf der x3dom-Website zu finden. Um alle Figuren darstellen zu können werden 12 verschiedene Modelle benötigt. Die Modelle können somit in eigenen Dateien abgelegt werden und mittels <inline> Element geladen werden.

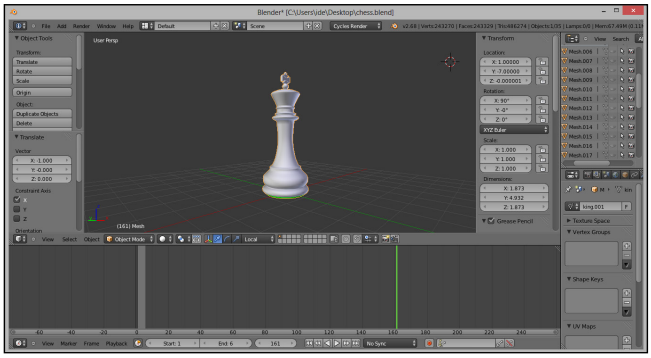

Abbildung 3. Modell des weißen Königs in Blender.

Laden der 3D Modelle

Da beim Schach zu Anfang immer alle Figuren auf dem Feld stehen, und damit auch alle Modelle geladen werden müssen, werden die Modelle über eine Funktion schon zu beginn dynamisch im HTML Dokument erzeugt. Die Funktion geht dafür ein Array durch, in dem die Daten der einzelnen Figuren gespeichert sind. Sie werden dynamisch in ein Inline Element geladen, welches dann in das HTML Dokument eingefügt werden. Von außen gesehen funktioniert dies wie bei einem Dictionary, welches die gesuchten Models zurückgibt. Die Models sind jetzt jedoch nicht sichtbar. Das werden sie erst, wenn sie detected worden sind. Bei jedem Durchlauf der Animationsschleife wird dann überprüft, ob in der Liste der Marker Einträge vorhanden sind. Diese entstehen, wenn ein Marker durch JSARToolkit gefunden wurde. Von Haus aus hat jeder Marker der vom Toolkit vorgegeben ist eine ID. Ist ein Marker erkannt worden, kann man diesen somit auf seine ID prüfen. Diese wird im Anschluss an eine Funktion übergeben, die mittels Switch-Case Anweisung einen String zurückgibt, welcher die Figur bestimmt, die auf den Marker zu zeichnen ist. Diese Figur wird dann sichtbar, indem das Attribut „render“ auf true gesetzt wird. Die Folgende Abbildung 4 zeigt, wie ein geladenes Model auf einen Marker projiziert wurde.

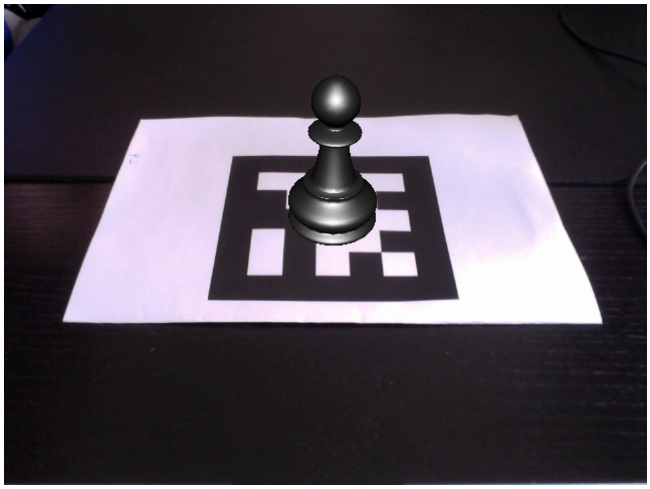

Abbildung 4. Auf Marker Projizierter Bauer.

Das „Figur-Array“

Das Array Beinhaltet Objekte, die Eigenschaften der einzelnen Figuren halten. Ähnlich wie bei einem JSON-File werden einfache Schlüssel-Wert-Paare gehalten, mit denen dann Objekte erzeugt werden. Gehalten werden der Name der Figur, der Pfad zur x3d-Datei, eine Beschreibung, die im Informationsfenster angezeigt wird, und der Pfad zum „Zugmuster“-Bild, welches im Infofenster angezeigt wird. Im Array stehen 32 Einträge, also einen für jeden Marker bzw. jede Spielfigur, die später auf dem Feld steht. Das Spiel ist dadurch leicht modifizierbar bzw. auf andere Spiele anpassbar.

Besonderheiten beim Laden

Die Figuren müssen zusätzlich transformiert werden. Da die Größe nicht in der Richtigen Relation zum Schachfeld ist muss diese angepasst werden. Des weiteren müssen Rotationen durchgeführt werden, um die Figur korrekt auf den Marker und damit auf den späteren Spielstein zu setzen. Ist eine Figur einmal geladen, so bleibt sie gespeichert. Sollte also z.B. eine Figur geschlagen werden und vom Spielfeld gelegt werden, so wird der Marker nicht mehr gefunden und die Figur wird nicht mehr gerendert. Kommt sie wieder ins Blickfeld der Kamera zurück, wird der Marker erneut erkannt und die Figur wird sichtbar. Somit gibt es den Rechenaufwand ein Model zu Laden nur bei der Initialisierung des Programms. Zur Laufzeit wird nur entschieden, welche davon angezeigt werden und welche nicht. Die Verdeckungsrechnung übernimmt das Toolkit JSAR-Toolkit. So können Figuren hintereinander angezeigt werden, ohne dass dabei Grafikfehler entstehen. In der folgenden Abbildung 5 sieht man, wie gut dies funktioniert. Das Toolkit muss zuvor jedoch die Größe des Markers kennen, um über affine Transformationen zu errechnen, welcher Marker wie weit entfernt ist

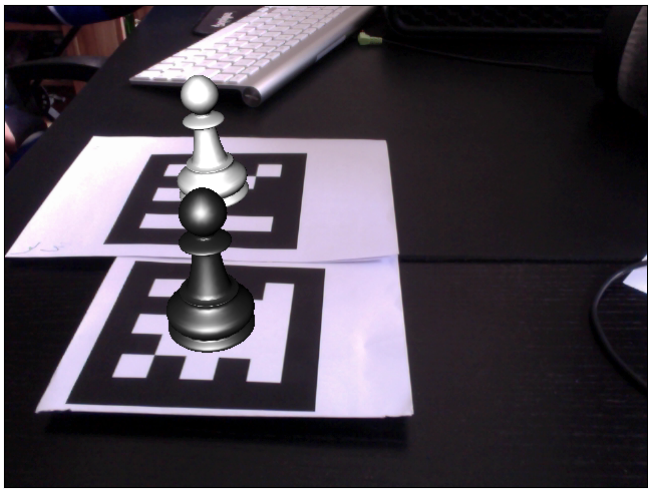

Abbildung 5. Der hintere Bauer wird korrekt verdeckt.

DAS INFORMATIONSFENSTER

Das Informationsfenster wird rechts neben dem Hauptfenster angezeigt. Beim Klick auf eine Figur werden hier Informationen über diese angezeigt.

Funktionalität

Die Funktion des Informationsfensters liegt darin, vor allem unerfahrenen Spielern eine Lernhilfe zu sein. Ein großes Problem beim Erlernen von Schach ist es, die Art und Weise zu lernen, in der bestimmte Figuren bewegt werden dürfen. Dies wird dem Spieler in Textform beschrieben, aber auch mit einem Bild dargestellt, wodurch er nicht andauend in Hilfestellungen blättern muss, sondern einfach auf eine Figur klicken kann, über die er informiert werden möchte. Mit AR-Chess ist es also möglich, die Vorteile einer digitalen Lernhilfe für Schach mit einem realen Spielfeld auf dem man realen Gegnern gegenübersteht zu verbinden.

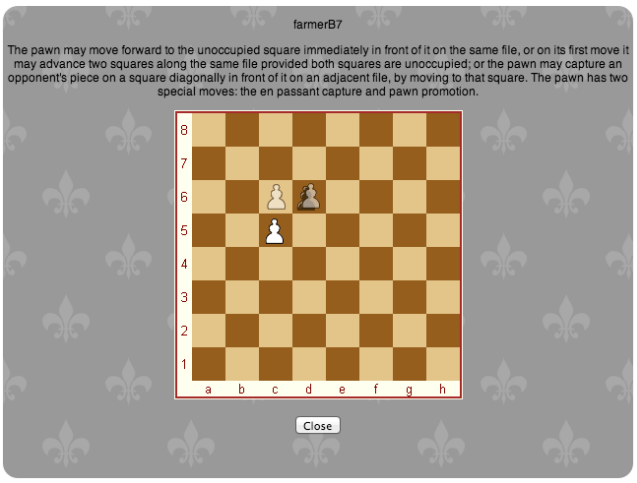

Abbildung 6. Informationsfenster für Bauern.

Aufbau des Informationsfensters

Wie schon erwähnt, werden der Text und der Pfad des Bildes im Figuren-Array vordefiniert. Da alle mit x3dom geladenen 3D-Objekte über OnClick-Eventhandler verfügen, können die Daten, mittels Klick auf eine bestimmte Figur, abgerufen und angezeigt werden. Hierfür wird die getFigureInfo Funktion aufgerufen, welche den vordefinierten Text und das Bild mittels innerHTML einfügt. Auch hier wird der Vorteil des Arrays mit vordefinierten Werten deutlich. Beschreibungen können leicht abgeändert und Bilder direkt über das Array ausgetauscht werden. Am Hauptdokument muss dafür nichts verändert werden.

DAS DESIGN

Das Design ist schlicht gehalten und an das von anderen Beispielen aus der x3dom Website angelehnt, da AR-Chess als ein weiteres Beispiel für Augumented Reality Anwendungen eventuell dort veröffentlicht werden soll. Das Hintergrundbild des Informationsfensters ist an alte 2D-Schachsimulationen angelehnt. Position und Größe des Hauptfensters sowie der Threshold Fader sind aus dem Grundgerüst übernommen worden um eine Konsistenz innerhalb der Beispiele von x3dom zu erhalten.

DAS FERTIGE AR-CHESS

Abbildung 7. Fertiges AR-Chess Feld.

Wie hier zu sehen ist, funktioniert die Anwendung zwar, sie ist jedoch stark von der Qualität der Kamera abhängig. Das Foto entstand mit einer Notebook Kamera, deren Auflösung nicht hoch genug war und deshalb manche Marker gar nicht bzw. in einem falschen Winkel erkennt. Als Lösung könnte eine Bessere Kamera benutzt oder die Marker vergrößert werden. Beim testverfahren kam auch eine Full-HD Webcam zum Einsatz, bei denen das Ergebnis besser war.

Abbildung 8. Ingame Screenshot AR-Chess.

EVALUATION

Schwierigkeiten bei der Entwicklung

Die erste Schwierigkeit bestand darin, das schon vorhandene Beispiel zur Benutzung des JSAR-Toolkit zu verstehen. Es setzt relativ stark fortgeschrittene Kenntnisse im Umgang mit JavaScript voraus. Dies und das technische Verständnis des x3dom Framework bzw. des JSAR-Toolkit Framework waren die ersten Meilensteine, die erreicht werden mussten und kosteten relativ viel Zeit. Als nächstes musste die Frage nach einer geeigneten Architektur gelöst werden, was nach einigen Fehlversuchen mit einer Dictionary-Artigen Lösung realisiert wurde. Das WebRTC Framework machte trotz Kompatibilität zu Google Chrome und Mozilla Firefox oftmals Probleme, da auf verschiedenen Rechnern mal der eine und auf anderen wieder der andere Browser Probleme hatte, die Kamera anzusprechen.

Plattformunabhängigkeit

Da die Anwendung Webtechnologien basiert, ist sie zumindest theoretisch auf allen Maschinen mit Web-Browsern ausführbar. Voraussetzung ist eine angeschlossene bzw. integrierte Webcam. Da bei integrierten Webcams in Notebooks die Auflösung und der Winkel der Kamera zur Unterlage meist nicht immer ideal sind, wird für ein perfektes Spielerlebnis eine externe Full-HD Webcam empfohlen. Auf Mobilen Endgeräten wird automatisch die Front-Kamera des Geräts ausgewählt, was das Spielen sehr schwierig macht. Somit ist es für Smartphones leider nicht sonderlich gut geeignet.

Tests

Getestet wurde die Anwendung in 2 kompletten Partien, die jeweils ca. 20 Minuten dauerten. Dabei blieb AR-Chess insgesamt stabil. Problem ist jedoch die Detection, die über längere Zeit bei wechselnden Lichtverhältnisse Probleme macht. Das zeigt sich dann durch ein willkürliches „umherzucken“ der Figuren, dass sie kurzzeitig gar nicht angezeigt werden oder sie in einem Falschen Winkel angezeigt werden. (Siehe Abbildung 7) Die Tests haben die Vermutung bestätigt, dass die Anwendung sehr stark von der Lichteinstrahlung und der Qualität der Kamera abhängig ist.

Erweiterbarkeit

AR-Chess kann sehr leicht modifiziert werden um damit ganz neue Spiele mit neuen Figuren zu kreieren. Da die Spiellogik weiterhin in der realen Welt vollzogen wird, muss hier nichts weiter verändert werden, als die z.B. andere x3d Files für die Figuren einzusetzen. Zudem können mehr Informationen zu den Figuren angezeigt werden oder die Modelle der Figuren angepasst. Eine Optimierung für Smartphones ist auch möglich, bedarf aber weiterer Modifikationen. Das Spiel könnte auch für AR-Brillen wie die Oculus Rift optimiert werden.

Zusammenfassung

Alles in allem kann festgehalten werden, dass mit x3dom und dem JSAR-Toolkit überzeugende Augumented Reality Anwendungen realisiert werden können. AR-Chess ist performant genug um auf nahezu allen Rechnern flüssig zu laufen. Es wurde erreicht, ein alt eingesessenes Spiel in einem neuen Kontext wiederzugeben. Beim Spielen mit AR-Chess macht auf alle Fälle Spaß, auch wenn hier und da Wackler in der Darstellung der Figuren sind. Die Probleme liegen jedoch weniger bei den Algorithmen um die Marker zu finden, als bei der Kamera. Hier werden Fortschritte in Auflösung und Kontrast das Verhalten von AR-Anwendungen in den nächsten Jahren verbessern.

DANKSAGUNG

Besonderer Dank gilt Frau Prof. Dr. Yvonne Jung, die uns auf das JSAR-Toolkit aufmerksam machte, unsere Idee unterstützte und uns bei der Entwicklung zur Seite stand. Wir danken auch Ilmari Heikkinen, ohne dessen Toolkit dies nicht möglich gewesen wäre.

QUELLENANGABE

- http://x3dom.org/x3dom/example/jsar/index.html
- http://www.html5rocks.com/en/tutorials/webgl/jsartoolkit_webrtc/#toc-webrtc
- http://www.html5rocks.com/en/tutorials/getusermedia/intro/
- https://github.com/samdutton/simpl/blob/master/getusermedia/sources/js/main.js