Die Innovationsecke von LernSax

Die Innovationsecke von LernSax ist eine Sammlung praktischer Hinweise und Tipps zur Nutzung von modernen Technologien für pädagogische Zwecke. Der Leser soll ermuntert werden, durch den Einsatz der neuen Medien den Unterricht und/oder eigene Projekte motivierend und interessant zu gestalten. Neben neuen didaktischen Konzepten steht der Spaß im Umgang mit den neuen Medien im Vordergrund.

QR-Codes im Unterricht

- Was sind QR-Codes
- Beispiel aus der Praxis
- Handreichung zur Nutzung und Erstellung

VR im Unterricht

- Was ist VR
- Beispiel aus der Praxis
- Handreichung zur Nutzung und Erstellung

AR im Unterricht

- Was ist AR
- Beispiel aus der Praxis
- Handreichung zur Nutzung und Erstellung

eigene Apps im Unterricht

- Was sind Apps
- Beispiel aus der Praxis
- Handreichung zur Nutzung und Erstellung

QR-Codes im Unterricht

Was sind QR-Codes

QR-Codes (Quick Response Code) können Informationen enthalten, die mit einer Handy-App oder einer Webcam gescannt werden können. Diese können Textnachrichten, komplette Visitenkarten oder Hyperlinks enthalten:

Text - enthält Informationen.

Visitenkarte - kann sofort zu den Kontakten hinzugefügt werden.

Hyperlink – kann sofort im Browser geöffnet werden.

Beispiel aus der Praxis

Beispiele für QR-Codes zur Anzeige von Lösungen:

Beispiel 1

Beispiele für QR-Codes zur Verlinkung auf Hilfeseiten:

Beispiel 1

Beispiel 2

Handreichung zur Nutzung und Erstellung

QR-Codes können mittels kostenloser Webtools erstellt und dann als Grafiken in Arbeitsblätter eingebaut werden. Angebote dazu findet man z.B. unter

http://www.qrcode-monkey.com
oder
http://www.visualead.com/.

Auf diesen Seiten kann man ohne Anmeldung die gewünschte Art von QR-Code auswählen, die Information eintragen und den Code erstellen lassen. Beide Webseiten bieten die Möglichkeit, die fertigen QR-Codes als Bilder abzuspeichern. Diese können dann in Arbeitsblätter etc. eingebaut werden. Der Schüler kann dann mittels Smartphone bei Bedarf den Code scannen und die Lösung überprüfen. Das geht bei den meisten QR-Apps auch offline. Im Falle einer Verlinkung auf Hilfeseiten muss das Smartphone jedoch online sein!

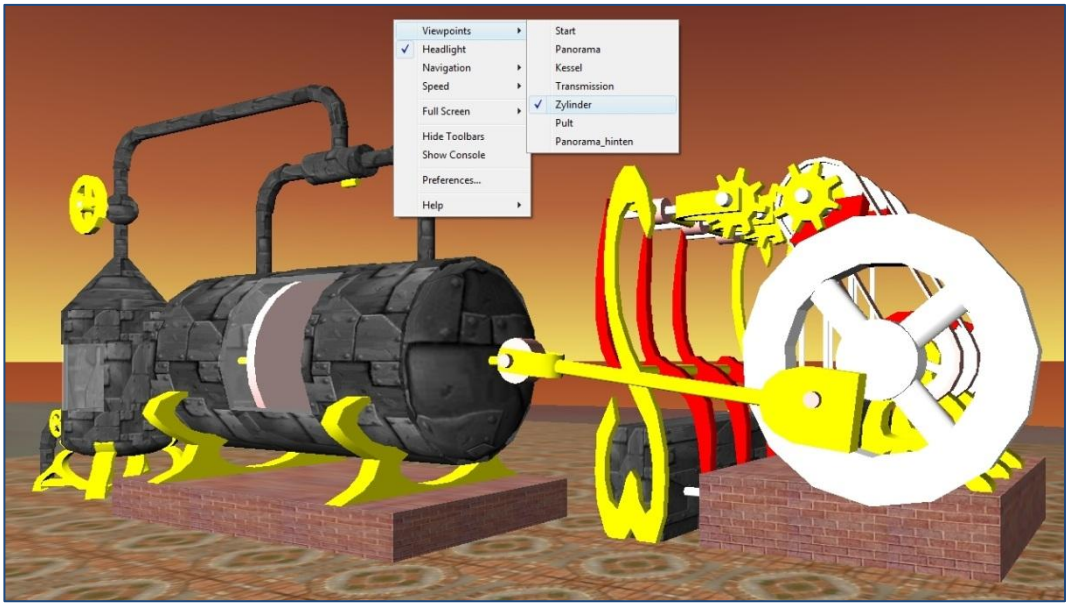
VR im Unterricht

Was ist VR

Die virtuelle Realität (kurz VR) ist eine vom Computer generierte Realität. VR-Anwendungen können dreidimensionale Inhalte enthalten und Objekte, Strukturen und komplette virtuelle Welten darstellen. Im Internet ist – neben Second Life – die Programmiersprache VRML am besten geeignet, virtuelle Realitäten darzustellen. Da mit Hilfe von VRML (Virtual Reality Modeling Language) komplexe dreidimensionale Zusammenhänge modelliert werden können eignet sich die VR hervorragend für die Simulation räumlicher Inhalte und bietet sich deshalb auch für den pädagogischen Einsatz an.

Beispiele aus der Praxis

Beispiel einer computergenerierte 3D-Animationen bzw. 3D-Simulation:



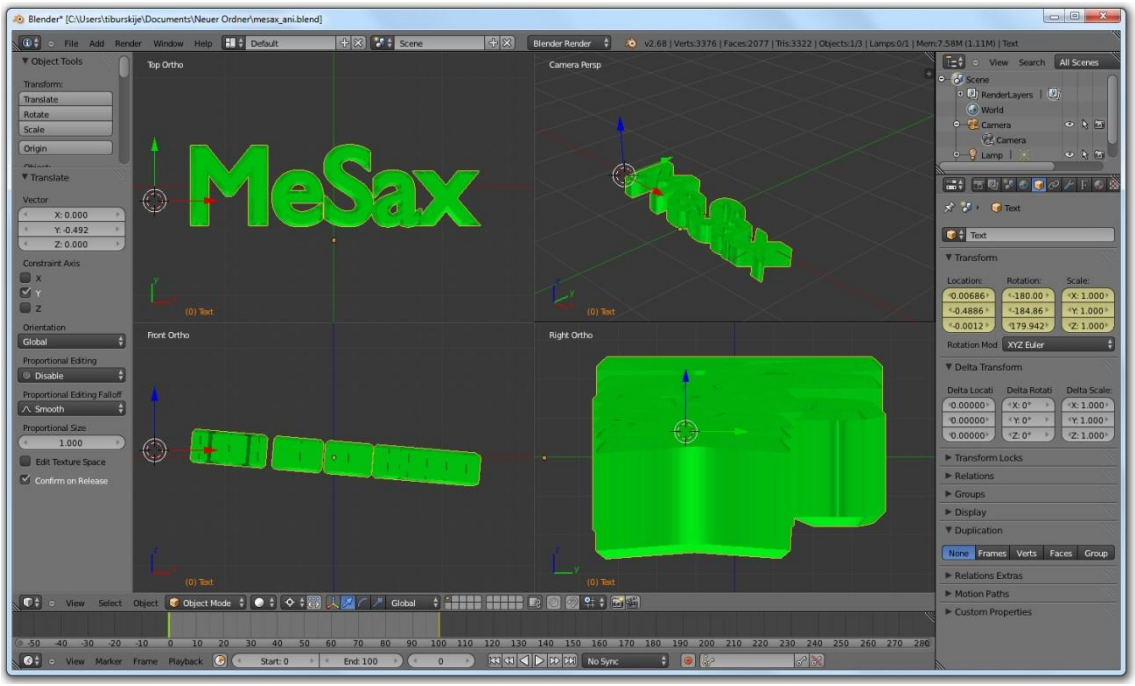
Im oben dargestellten Screenshot wird die Funktionsweise einer Dampfmaschine anhand eines virtuellen Modells erlebbar. Für die Darstellung solcher virtuellen Welten ist ein Plug-In erforderlich.

Beispiel einer komplexen 3D-Lehr-Lern-Umgebung:

Im oben dargestellten Screenshot wird die Lernumgebung 3D-Strömung dargestellt.

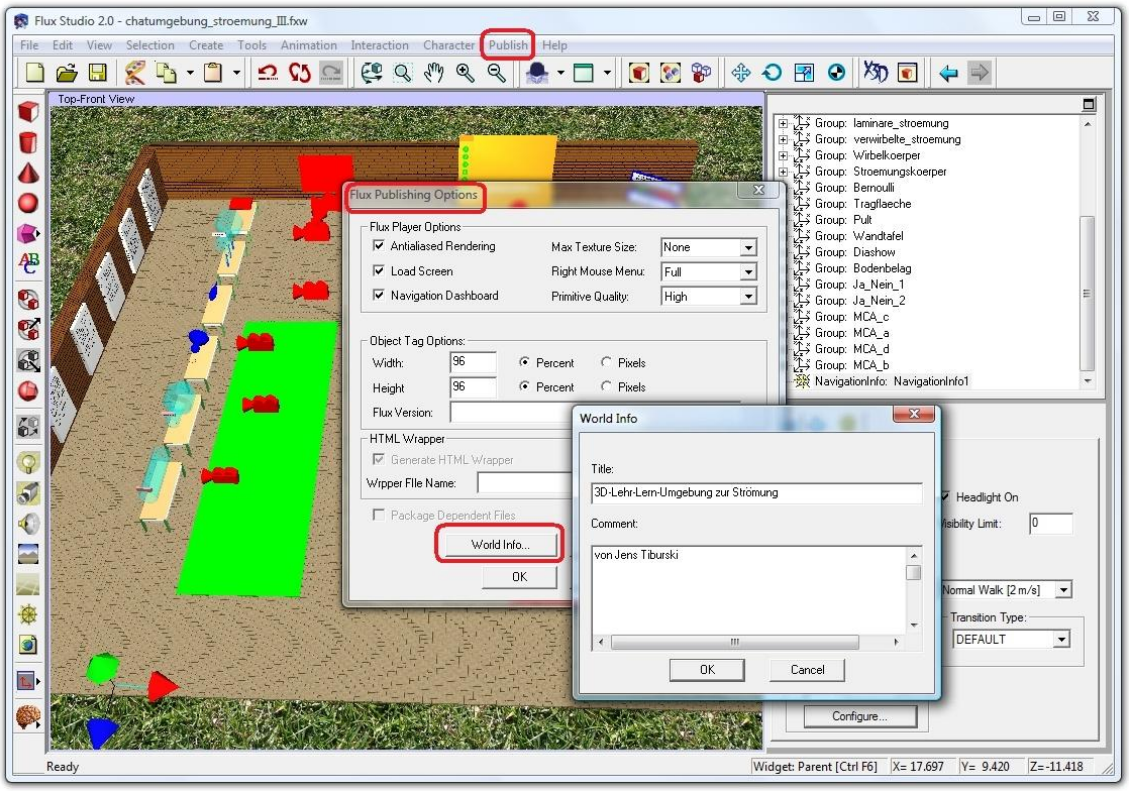
Handreichung zur Nutzung und Erstellung

Es gibt zahlreiche Freeware, die an den – sehr teuren – Marktführer in Sachen 3D-Gestaltung, **Autodesk 3ds Max,** heranreichen. Gerade das Freeware-Programm **Blender** kann mit sehr vielen Funktionen aufwarten. Daraus resultiert jedoch eine gewisse Einarbeitungszeit. Der Aufwand lohnt sich jedoch…



Screenshot des Programms **Blender Portable 2.69a**

Einfacher zu bedienen, aber trotzdem sehr umfangreich in den Funktionen ist das Programm **FluxStudio**. Auch dieses 3D-CAD-Programm ist für pädagogische Zwecke kostenlos.



Screenshot des Programms **FluxStudio 2.0**

Beide Programme können kostenlos aus dem Internet heruntergeladen und installiert werden. Deshalb eignen sie sich auch für Projektarbeiten mit Schülergruppen oder auch dem Einsatz im Unterricht.

Arbeit mit dem Freeware-Programm **Blender**

Downloadseite: http://www.blender.org/download

Anleitungen: http://blenderhilfe.de/

Arbeit mit **FluxStudio** (Für pädagogische Zwecke freigegeben.)

Downloadseite: http://mediamachines.wordpress.com/flux-player-and-flux-studio/

Anleitungen: http://www.hausarbeiten.de/faecher/vorschau/200240.html

AR im Unterricht

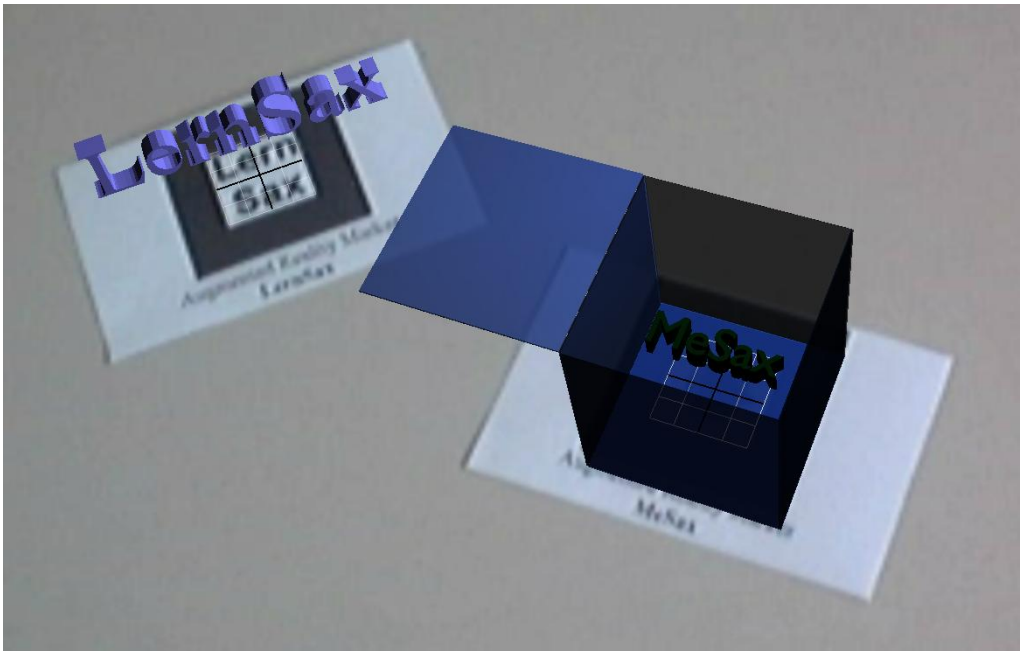
Was ist AR

Als Augmented Reality (kurz AR) wird die erweiterte Realität bezeichnet, wobei der Blick auf die reale Welt durch ein Objektiv (Webcam, Smartphone) mit virtuellen Elementen ergänzt wird. In der Praxis unterscheidet man grundsätzlich drei Methoden:

Die Marker-gestützte Methode

Bei der Marker-gestützten Methode werden vorgefertigte Standard-Bilder als Marker definiert:

Das AR-Programm erkennt diese Marker und überlagert diese dann rechnerisch mit Zusatzinhalten (Textbotschaften, Bilder, Videos oder 3D-Inhalten).

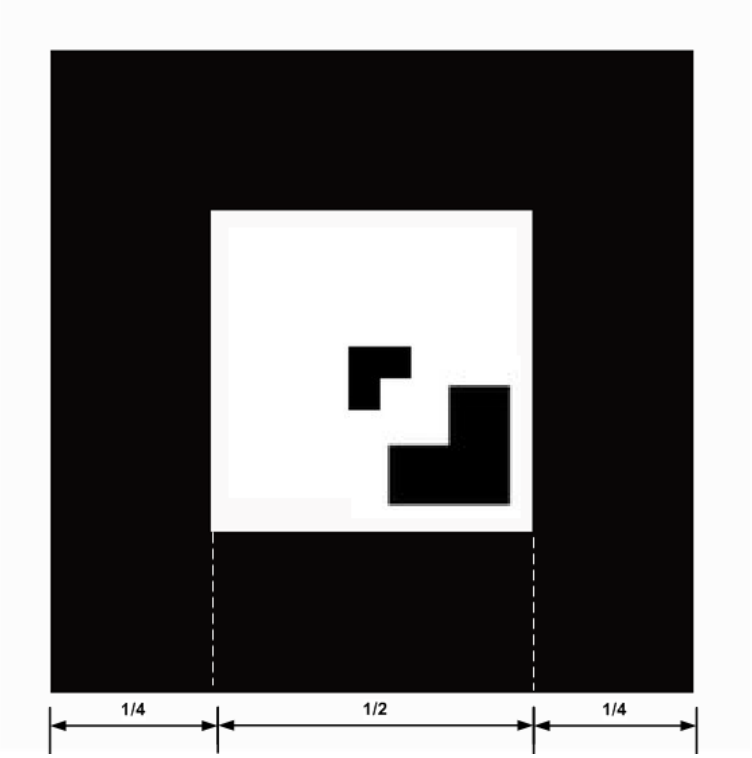


Im oberen Bild wurden die Marker mit einem 3D-Schriftzug bzw. einer kleinen 3D-Animation überlagert.

Am besten werden Marker erkannt, die folgende Dimensionen besitzen:

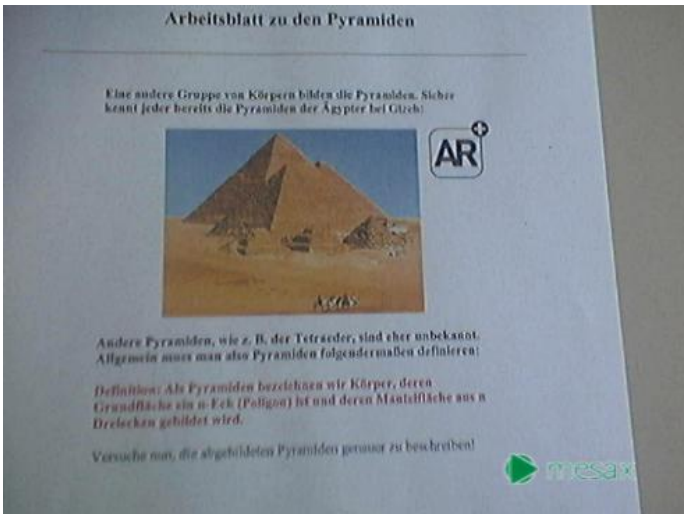
- Die Größe des Markers sollte 80 mm nicht überschreiten,

- Rand und Inhalt sollten das Verhältnis 1:2:1 aufweisen.



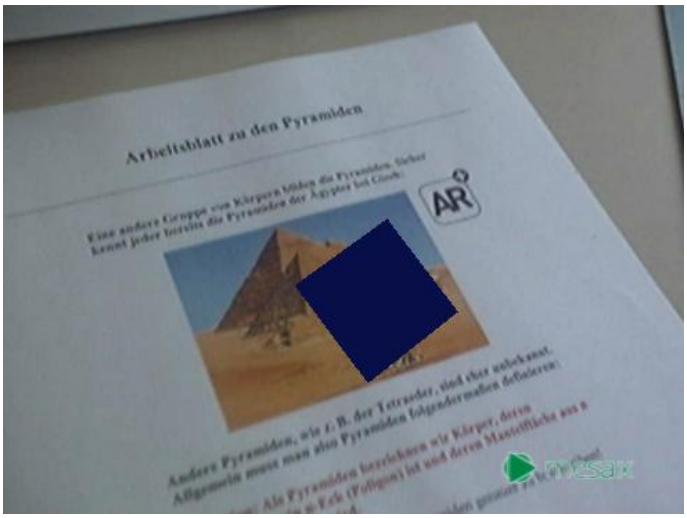
Die Image-gestützte Methode

Prinzipiell funktioniert diese Methode analog zur vorhergehenden Methode mit den speziellen Markern. Der Unterschied liegt lediglich in der Verwendung beliebiger Bilder als Marker.

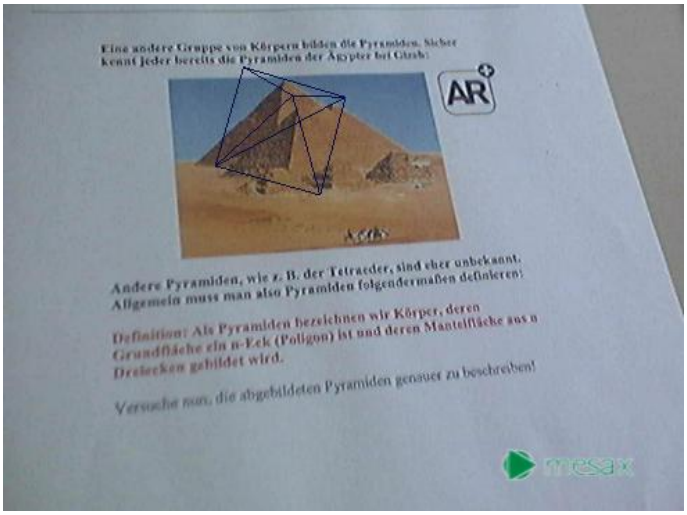


So kann beispielsweise die Pyramide des Arbeitsblattes als Image-Marker dienen. Das AR-Symbol soll lediglich darauf hinweisen, dass hier Augmented-Reality-Inhalte verborgen sind.

Nun kann die AR-Applikation das Bild mit ausgewählten Inhalten überlagern.



Oben dargestellt ist die Überlagerung mit dem animierten Modell einer quadratischen Pyramide, während unten das Drahtgittermodell angezeigt wird.



Die GPS-gestützte Methode

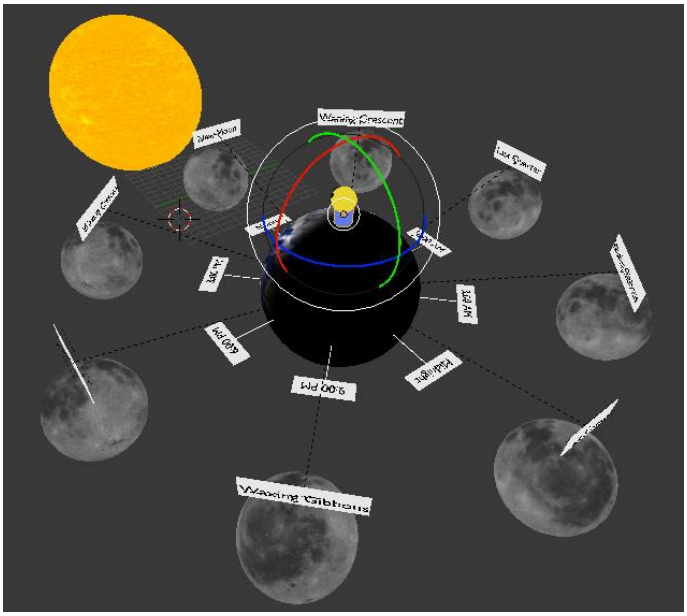
Bei der GPS-gestützten Methode errechnet der AR-Browser des Smartphones aus der Position und Richtung des Gerätes die zur Verfügung stehenden lokalen Inhalte. Diese werden dann als Icons auf dem Display mit dem Kamerabild überlagert.

Beispiele aus der Praxis

1. Das **Augmented Reality (AR) Solar System Magic Book** ist ein sehr schönes Beispiel für Marker gestützte AR, das alle Planeten nicht nur als farbige Illustrationen im Buch enthält, sondern über AR auch die dreidimensionalen Modelle zeigt.

Auf der Webseite **Augmented Reality in Education** [http://www.arined.org/?p=666] kann man alle Dateien für die Verwendung am PC herunterladen.

2. Auch die **Lunar Phases Astronomy AR Lesson** ist ein sehr gut gestaltetes Beispiel für AR-Inhalte im Bildungsumfeld. Hier wird mit Hilfe animierter 3D-Szenen die Entstehung der Mondphasen erklärt:



Die Dateien stehen ebenfalls auf **Augmented Reality in Education** zum Download bereit [http://www.arined.org/?p=855].

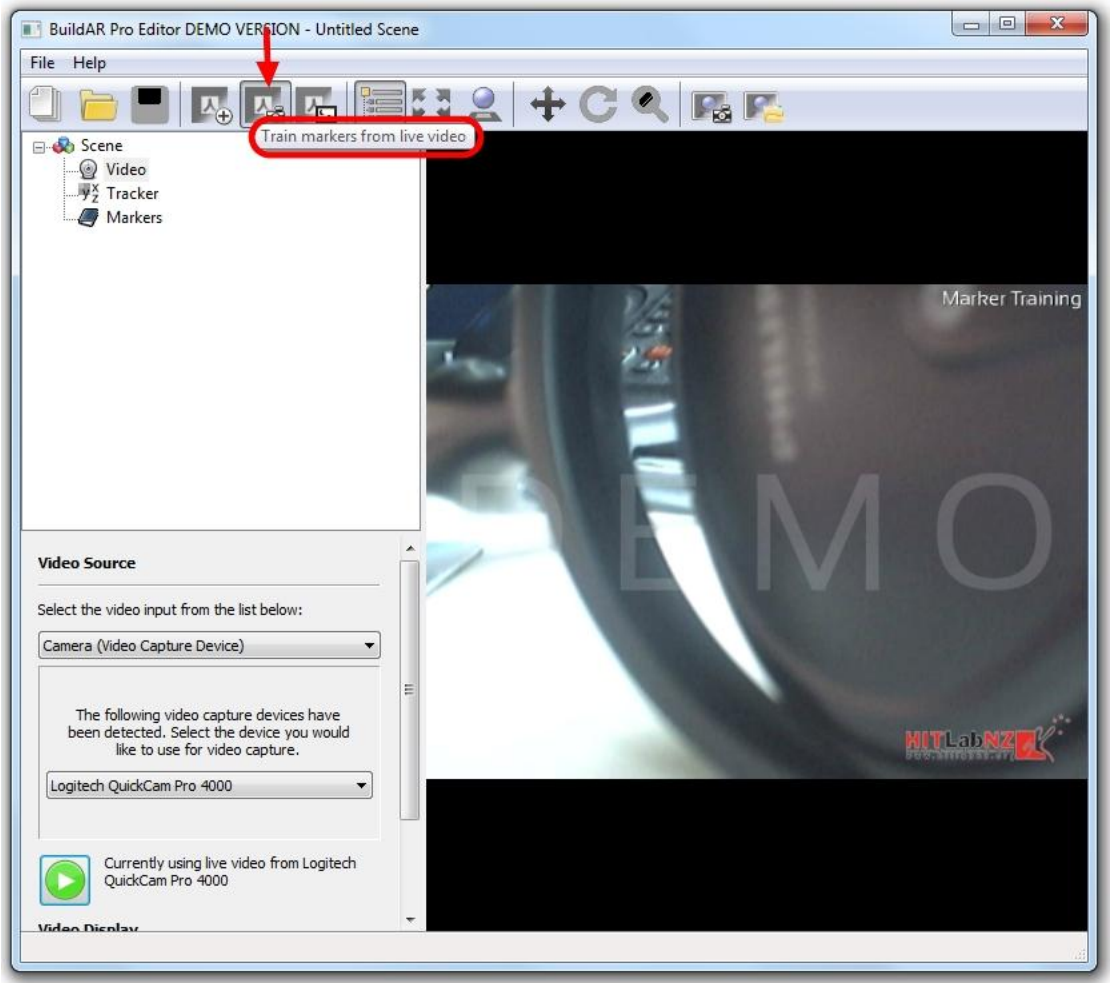
3. **Handy-Rallyes**, bei denen GPS-AR verwendet wird, um an vorprogrammierten POI (Points Of Interest) Informationen und/oder Quizfragen zu erhalten.

Unter **Surfing The Streets** [http://surfingthestreets.wordpress.com/] findet man eine Auswahl fertiger Rallyes sowie Anleitungen zur Gestaltung eigener Rallyes.

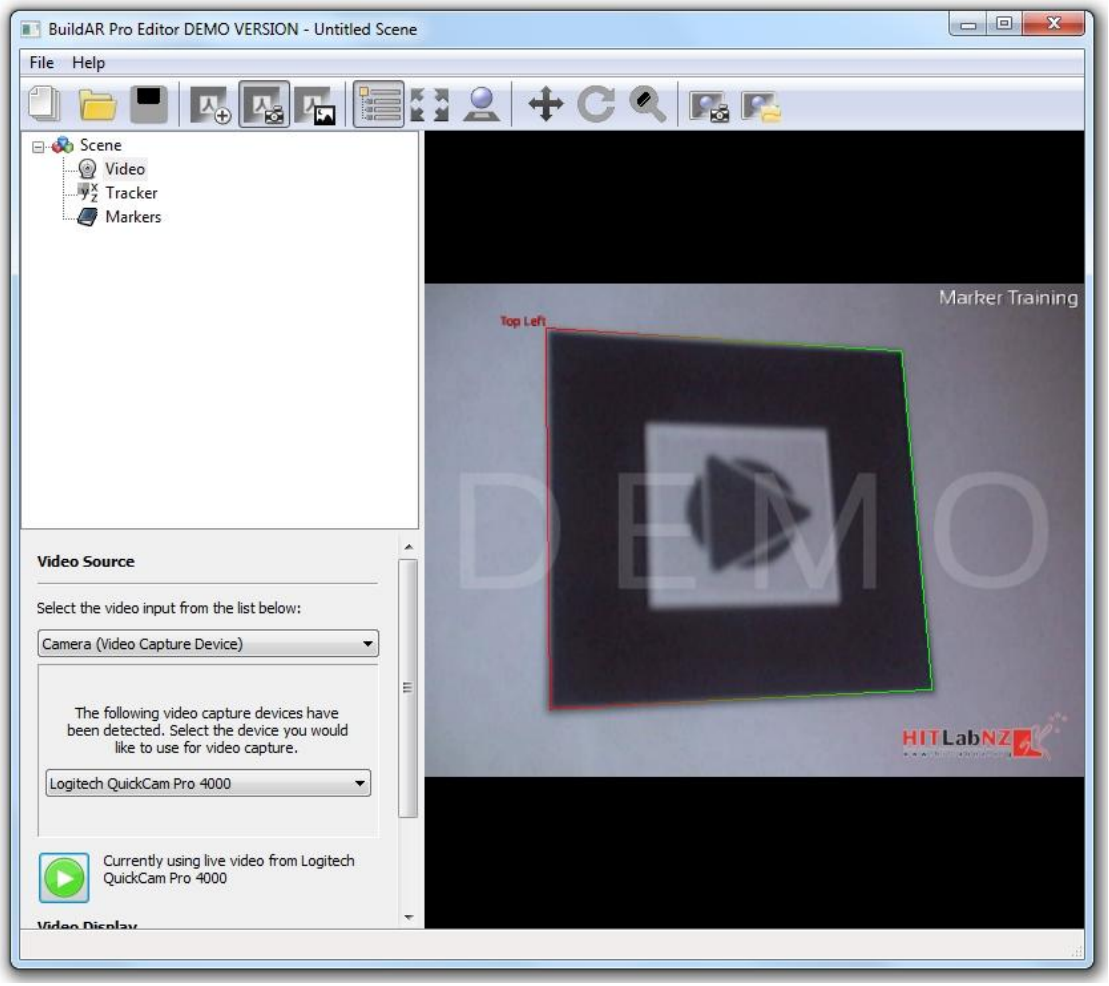
Handreichung zur Nutzung und Erstellung

Wer AR im Unterricht einsetzen möchte muss wissen, welche Art er verwenden möchte. Die Marker gestützte Methode eignet sich sehr gut für den Einsatz am PC oder für Laptop-Klassen. Externe oder fest eingebaute Webcams ermöglichen die Verbindung der gedruckten Arbeitsmittel mit den virtuellen Inhalten.

Als AR-Programm gibt es unterschiedliche Freeware von verschiedenen Anbietern. Das Programm **AMIRE v1** (unter anderem vom **Fraunhofer Institut** mitentwickelt) ist zwar am komfortabelsten, leider läuft es auf den getesteten System sehr instabil und führte sehr oft zu Abstürzen. Als stabilstes Freeware-Programm hat sich BuildAR herausgestellt. Der kostenlose Download von der Webseite der Firma **HIT Lab NZ** [http://www.buildar.co.nz/] enthält sowohl die Demo-Version des Programms zum Erzeugen von AR-Szenen **BuildAR Editor** (Leider ohne die Funktion des Speicherns!) als auch die freie Version des Betrachters **BuildAR Viewer**. Obwohl das Demo-Programm kein Speichern von Szenen zulässt, kann es trotzdem für die Erzeugung von Marker-Pattern verwendet werden. Dazu muss man zuerst mit Hilfe eines Grafik-Programms einen Marker erstellen. Dann kann dieser über die Funktion *„Train markers from live video“* mit der Webcam gescannt werden:

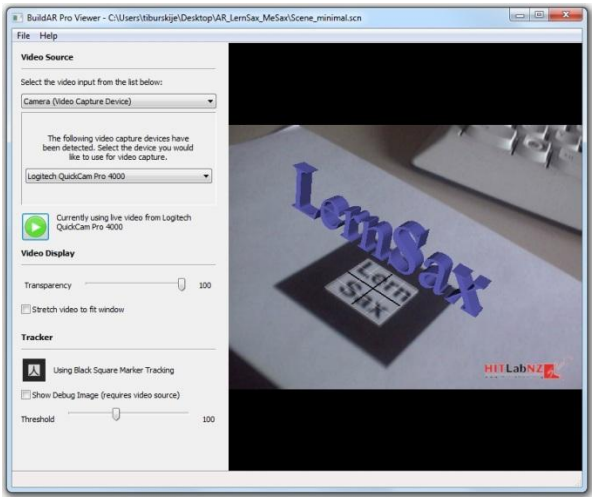


Nach dem Start des Programms muss unter dem Punkt *Video* die gewünschte Videoquelle ausgewählt und aktiviert werden. Dann kann das Marker-Training starten.

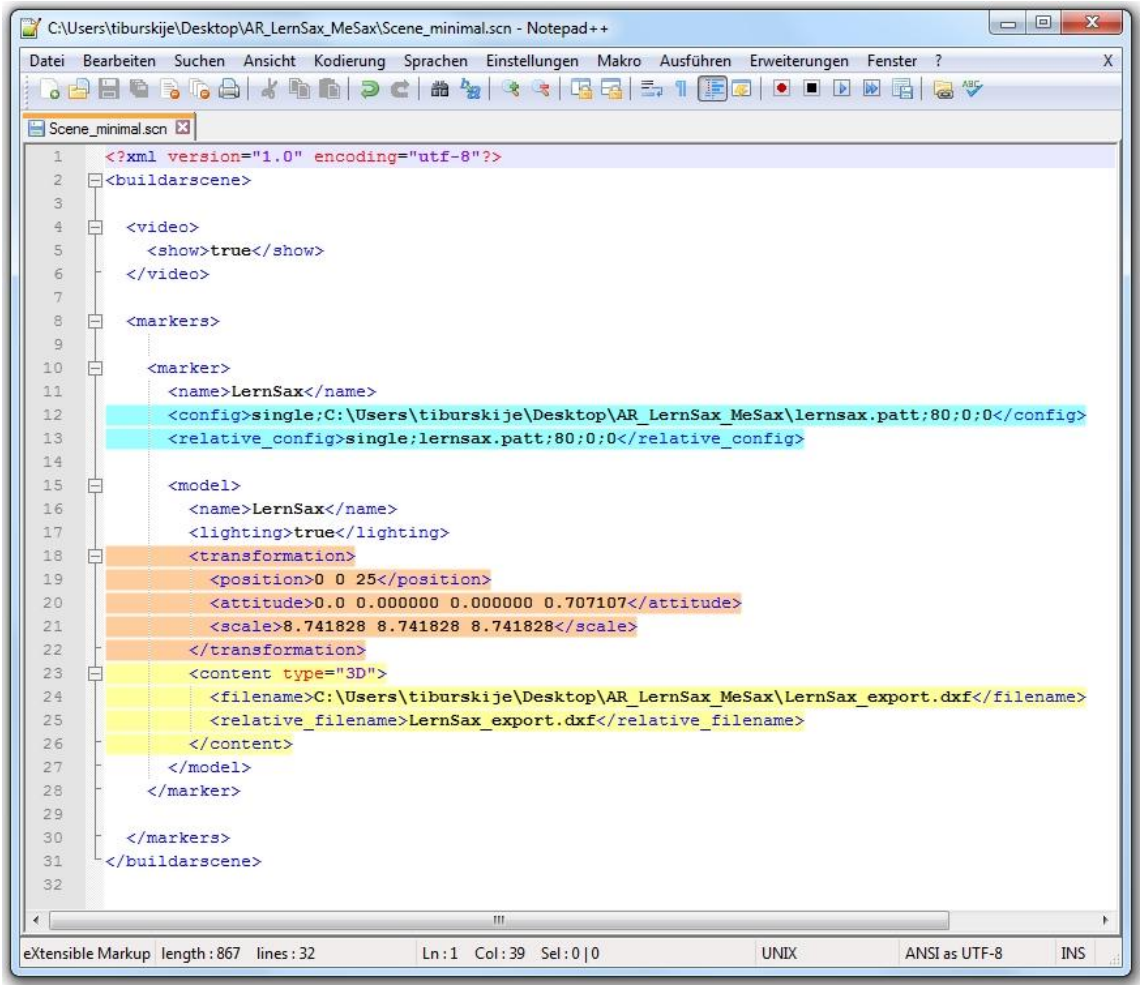


Wenn der Marker erfasst worden ist erscheint ein rot-grüner Rahmen um den Marker. Es wird die Lage des Markers an Hand der oberen linken Ecke abgefragt. Wenn man nun mit der Maus auf den markierten Marker klickt, dann öffnet sich das Speichern-Menü und der Marker wird als patt-Datei zur weiteren Verwendung abgespeichert.

Da das Programm keine Szenen abspeichert, kann man nun zwar noch ein wenig mit dem neuen Marker experimentieren [„Marker zur Szene hinzufügen“], aber man sollte nicht zu viel Energie darauf verwenden. Vielmehr sollte man sich nun den Quelltext fertiger AR-Szenen anschauen. Beispiel einer Minimal-Szene aus einem Marker mit einem Objekt:



Und hier ist der dazugehörige Quelltext dargestellt:



Die Szenen-Dateien haben die Endung *.scn, aber dahinter verbirgt sich eine einfache xml-Datenstruktur, d.h. man kann diese Dateien (wenn sie nicht komprimiert sind) in jedem beliebigen Texteditor öffnen. Im angezeigten Beispiel wird die Datei *Scene_minimal.scn* im **Notepad++** (Freeware) als xml-Datei angezeigt. Nun hat man alle Möglichkeiten, die Szene zu bearbeiten.

Zum Inhalt des Quelltextes:

```
<buildarscene>                          ← Beginn der Szene
    <video>
        <show>true</show>               ← Webcam Video aktivieren
    </video>
    <markers>                           ← Beginn aller Marker
        <marker>                        ← Beginn eines konkreten Markers
            <model>                     ← Beginn des angezeigten AR-Inhaltes
            …
            </model>                    ← Ende des angezeigten AR-Inhaltes
        </marker>                       ← Ende eines konkreten Markers

        <marker>                        ← Beginn des nächsten Markers
            <model>                     ← Beginn des angezeigten AR-Inhaltes
            …
            </model>                    ← Ende des angezeigten AR-Inhaltes
        </marker>                       ← Ende des nächsten Markers
    </markers>                          ← Ende aller Marker
</buildarscene>                         ← Ende der Szene
```

Im hellblau markierten Teil wird der Marker definiert. Dieser muss zuvor als patt-Datei gespeichert worden sein. Hier sind nun zwei Eintragungen erforderlich: erstens ist der vollständige Dateipfad des Speicherorts der Datei einzutragen und zweitens der Dateinamen des Marker sowie seine Größe in Millimetern.

Im gelb hervorgehobenen Teil wird die Art des AR-Inhaltes angegeben und beschrieben. type="3D" verweist auf ein zuvor erzeugtes und abgespeichertes 3D-Objekt, in diesem Falle eine mit Blender als dxf exportierte Datei mit dem Namen *LernSax_export.dxf*. Der Speicherort und der Namen der Datei müssen auch hier wieder angegeben werden.

Im orangenen Teil wird eben dieser 3D-Inhalt genau positioniert. Mit Hilfe der Punkte *position, attitude und scale* werden die Lage, die Rotation sowie die Größe des Objektes verändert. *Position* und *scale* enthalten dabei drei Koordinaten, die jeweils der x-, y- und z-Richtung entsprechen. Bei der Rotation gibt es einen vierten Wert, welcher den Drehwinkel im Bogenmaß enthält.

Nach einer gewissen Einarbeitungszeit ist damit die Erstellung von komplexen AR-Szenen mit mehreren Markern möglich. Durch die Einbettung der Marker (als Grafiken) in Textdokumente kann man somit ganze „Magic Books" mit AR-Inhalten zu einem gezielten Thema zusammenstellen.

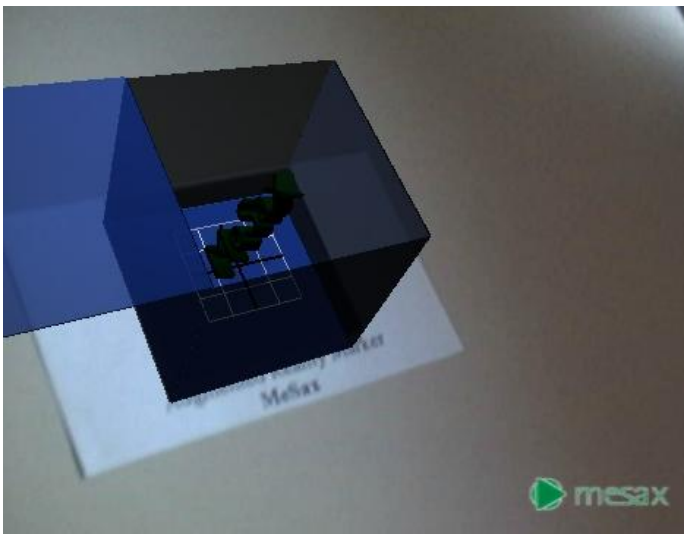
Wem das nicht genug ist, der kann die Szenen auch noch mit folgenden Mitteln erweitern:

Einbindung von Logos

Für das Einbinden von eigenen Logos muss man im Quelltext der AR-Szene lediglich einen Logo-Tag-Gruppe zwischen <video>-Tag und <markers>-Tag hinzufügen:

```
<logo show="true" width_percent="20" height_percent="7" position_x="580" position_y="80">
        <filename>\BuildAR\Logo_MeSax.png</filename>
        <relative_filename>Logo_MeSax.png</relative_filename>
</logo>
```

Nun wird das (transparente) Bild *Logo_MeSax.png* auf der gewünschten Position als Logo eingeblendet:



← MeSax-Logo

Einbindung weiterer Objekt-Typen

Wer nicht nur 3D-Inhalte anzeigen lassen möchte, der kann auch auf folgende Inhalte innerhalb des <content>-Tag zurückgreifen:

Grafik-Primitive der 3D-Gestaltung

```
<content type="PRIMITIVE">
  <color>0.000000 0.250980 0.501961 1.000000</color>
  <primitive type="CYLINDER">
    <box length="10.000000" width="10.000000" height="10.000000"/>
    <capsule height="10.000000" radius="10.000000"/>
    <cylinder height="3.000000" radius="10.000000"/>
    <cone height="10.000000" radius="10.000000"/>
    <sphere radius="10.000000"/>
  </primitive>
</content>
```

Video-Wiedergabe

```
<content type="QTVIDEO">
  <filename>C:\Users\tiburskije\Desktop\AR_LernSax_MeSax\mesaxi_home.mp4</filename>
  <relative_filename>mesaxi_home.mp4</relative_filename>
  <volume mute="false">1.003922</volume>
  <loop>true</loop>
  <on_marker_found action="PLAY" volume="1.000000"/>
  <on_marker_lost action="PAUSE" volume="1.000000"/>
</content>
```

Einbindung von Bildern als Marker

Im folgenden Beispiel ist der <markers>-Tag des Pyramidenbeispiels zu sehen. Dieses Beispiel beruht auf Image-gestützter AR. Im <marker>-Tag ist nun das Bild *pyramiden_gizeh.jpg* mit der Größe des Bildes in Pixel angegeben:

```
<markers>
  <marker>
  <name>Pyramide_Bild</name>
  <config>C:\Users\tiburskije\Desktop\Lernsax\Innovationsecke\pyramiden_gizeh.jpg;310;236</config>
  <relative_config>pyramiden_gizeh.jpg;310;236</relative_config>
  <grid show="false" size="5"/>
  <filtering>
    <translational>1.000000</translational>
    <rotational>1.000000</rotational>
    <remain_visible>5000.000000</remain_visible>
  </filtering>
  <model>
    <name>Pyramide_Modell</name>
    <render_order>0</render_order>
    <wireframe>true</wireframe>
    <phantom>false</phantom>
    <lighting>false</lighting>
    <delay>0</delay>
    <transformation>
      <position>100 -50 10</position>
      <attitude>1.0 0.000000 0.000000 0.707107</attitude>
      <scale>50 50 50</scale>
    </transformation>
    <content type="3D">
      <filename>C:\Users\tiburskije\Desktop\Lernsax\Innovationsecke\Pyramide.fbx</filename>
      <relative_filename>Pyramide.fbx</relative_filename>
      <reset_anims>false</reset_anims>
      <loop>true</loop>
    </content>
  </model>
</marker>
```

Der <wireframe>-Tag sorgt dafür, dass nur das Drahtgittermodell der Pyramide angezeigt wird. Die Positionskoordinaten beziehen sich nun auf die obere linke Ecke des Bildes.

Fertigstellung einer AR-Applikation zur Weitergabe

Wenn man eine AR-Szene erstellt und erfolgreich getestet hat, dann möchte man die Ergebnisse der Arbeit natürlich auch weitereichen können. Dies gestaltet sich auf Grund der Pfadangaben im Szenen-Quelltext jedoch problematisch. Deshalb ist die Möglichkeit, das Freeware-Programm BuildAR-Viewer weiterzureichen, eine gute Möglichkeit, zusammen mit dem Programm auch die eigenen Szenen auf Wechseldatenträgern bereitzustellen. Besonders bemerkenswert ist dabei, dass über eine Batch-Datei die Szene per Autostart im Vollbildmodus geöffnet werden kann! Doch nun der Reihe nach:

1. *Dateien übertragen*

   Um eine fertige Szene auf einem Wechseldatenträger zu speichern empfiehlt es sich zuerst das Programm BuildAR-Viewer komplett mit dem gesamten Ordner **BuildAR** auf den Datenträger zu übertragen. Danach werden alle für die eigene Szene relevanten Dateien -inklusive der Szene selbst - in diesen Ordner kopiert.
   Nun werden im Quelltext die Pfadangaben aktualisiert:
   <config>single;\BuildAR\markers\lernsax.patt;80;0;0</config>
   Das Programm BuildAR greift auf den Ordner BuildAR im Root-Verzeichnis des Datenträgers zu. Egal, ob USB-Stick oder CD – mit dieser Pfadangabe findet das Programm die Dateien. Natürlich müssen auch die Pfade für die verwendeten Objekte auf diese Weise angepasst werden.

2. *Start-Batch-Datei erzeugen*

   Nach dem Probelauf der AR-Szene kann Schritt zwei in Angriff genommen werden. Der Datenträger erhält eine Start-Batch-Datei. Diese kann man in jedem beliebigen Editor verfassen und enthält eine einzige Kommandozeile:
   BuildAR\BuildARViewer.exe BuildAR\Scene_LernSax_MeSax.scn --startscene=1
   Natürlich muss der Szenenamen durch den eigenen ersetzt werden! Die Datei wird dann im Root-Verzeichnis des Datenträgers unter dem Namen **ar_scene_launcher.bat** gespeichert. Mit einem Doppelklick auf die gespeicherte Datei sollte nun sofort des Programm BuildARViewer.exe gestartet werden und die eigene Szene im Vollbildmodus anzeigen.

3. *Auto-Start-Inf erzeugen*

   Damit der geschätzte User die Start-Datei nicht erst zu suchen braucht kann man nun noch die Autostart-Funktion von Windows nutzen. Auch dazu erstellt man mit Hilfe des Editors eine Inf-Datei mit folgenden Eintragungen:
   [autorun]
   OPEN=ar_scene_launcher.bat
   icon=BuildAR\MeSax.ico
   Die gelb unterlegten Eintragungen müssen wieder an die eigenen Dateien angepasst werden. Der Punkt mit dem icon kann auch weggelassen werden, wenn man kein eigenes Icon zur Hand hat. Die fertige Datei muss dann **als autorun.inf** ebenfalls im Root-Verzeichnis des Datenträgers abgespeichert werden.
   Wenn der Datenträger jetzt in ein Laufwerk eingelegt wird, dann greift die Autostart-Funktion von Windows auf die autostart.inf-Datei zu und zeigt das Icon im Explorer für

dieses Laufwerk an und startet über den Aufruf der Batch-Datei das Programm mit der angegeben AR-Szene.
Der fertige Datenträger kann nun der gedruckten Publikation mit den Markern beigelegt und das Ganze als eine Art „Magic Book“ weitergegeben werden.

eigene Apps im Unterricht

Was sind Apps

Als Apps werden Applikationen (Programme) auf Smartphones genannt, die meistens eine ganz spezielle Funktion erfüllen. So gibt es z.B. Wetter-Apps, Newsreader-Apps oder auch AR-Apps. Am weitesten verbreitet sind Smartphones und Tablets mit den Betriebssystemen Android oder iOS. Im Folgenden werden nur Android-Apps vorgestellt, da das MIT mit dem App-Inventor eine kostenlose Entwicklungsumgebung für Android-Apps veröffentlicht hat.

Beispiele aus der Praxis

Die besten Android Apps für die Schule
[http://www.24android.com/de/apps/die-besten-apps/die-besten-android-apps-fuer-die-schule/]

Mit dieser App haben Schüler den eigenen **Stundenplan** jederzeit griffbereit auf dem Android Smartphone oder Tablet verfügbar. Verschiedene Farben sorgen für Übersichtlichkeit und mit dem integrierten Widget kann der Stundenplan auch direkt auf dem Homescreen platziert werden. Die Schulstunden können um zusätzliche Informationen wie Notizen und das Klassenzimmer ergänzt werden. Die Stundenplan App ist kostenlos.

Die **Formelsammlung** Mathematik App bietet schnellen Zugriff auf die wichtigsten mathematischen Formeln, und kann sowohl für die Schule als auch das Studium nützlich sein. Enthalten sind die Grundrechenarten, Algebra, Analysis, Geometrie, Trigonometrie, Analytische Geometrie und Logik. Die kostenlose Version von Formelsammlung Mathematik verfügt über Werbeeinblendungen, die kostenpflichtige Variante ist werbefrei.

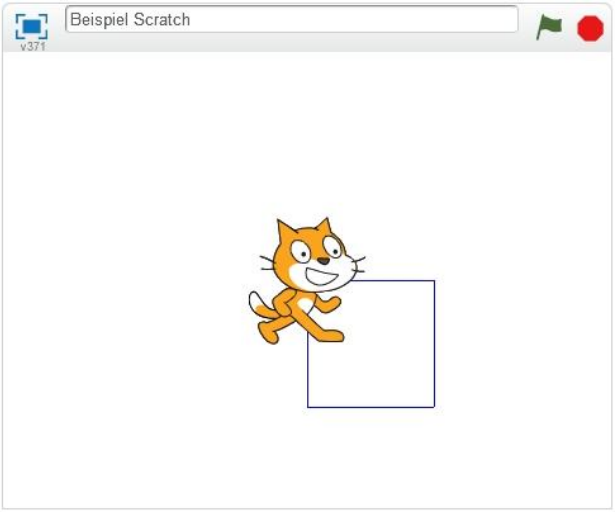
Wurden Vokabeln früher zum Lernen meistens in ein Heft geschrieben und anschließend von einer weiteren Person abgefragt, reicht dafür heute ein Android Smartphone und die **Vokabel-Heft** App. Da fast jeder sein Smartphone sowieso immer bei sich trägt, ergeben sich dadurch auch neue Möglichkeiten, sich immer mal wieder mit den Vokabeln zu beschäftigen. Und das integrierte Lernsystem macht eine weitere Person zum Abfragen überflüssig. Die Vokabel-Heft App steht kostenlos zum Download im Google Play Store bereit.

Handreichung zur Nutzung und Erstellung

Um eigene Apps entwickeln zu können benötigt man eine Entwickler-Umgebung ähnlich wie bei einer Programmiersprache. Das **Massachusetts Institute of Technology** hat mit dem **App-Inventor** eine solche geschaffen und im Internet freigegeben.

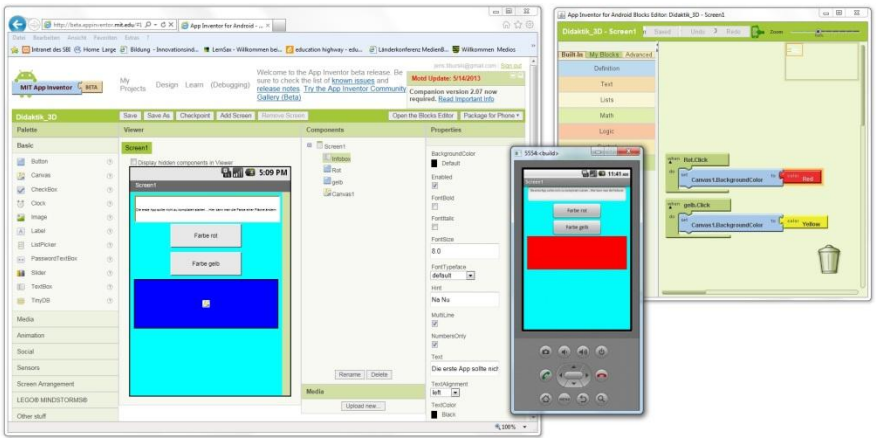
Das Beste daran ist, dass der App-Inventor sich an dieselbe Programmier-Philosophie anlehnt, wie auch schon die Programmierumgebung Scratch [http://scratch.mit.edu]. Das Prinzip der grafischen Programmierung durch Anordnung von bildhaften Modulen wurde dabei von Scratch auf den App-Inventor übertragen:

Beispiel für grafische Programmierung mit Scratch

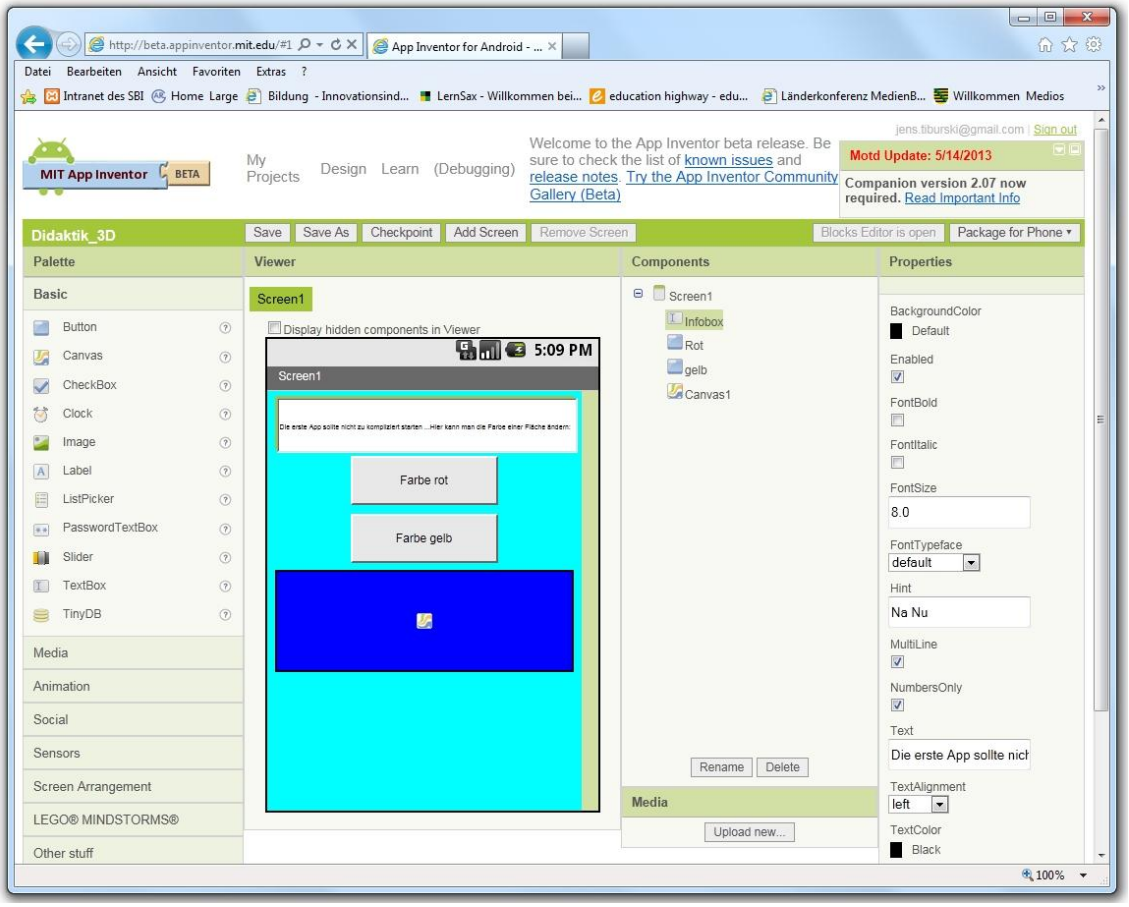


Wenn die grüne Fahne (Start-Symbol) angeklickt wird, dann läuft der – natürlich frei wählbare – Darsteller 10 x 10 Pixel geradeaus, dreht sich 3 x 30°, läuft wieder geradeaus und so weiter bis er das Quadrat beendet hat. Die einzelnen Programmierbefehle müssen nicht mehr in einen Quelltext eingegeben werden, sondern können bequem aus einer Reihe von Menüs ausgewählt und per Drag & Drop zusammengefügt werden. Damit können auch schon kleine Kinder eigenständig nette „Programme“ erstellen. Mehr Informationen über Scratch findet man auf der deutschsprachigen Webseite [http://scratch-dach.info/wiki/Hauptseite].

Im App-Inventor sieht das dann ganz ähnlich aus:



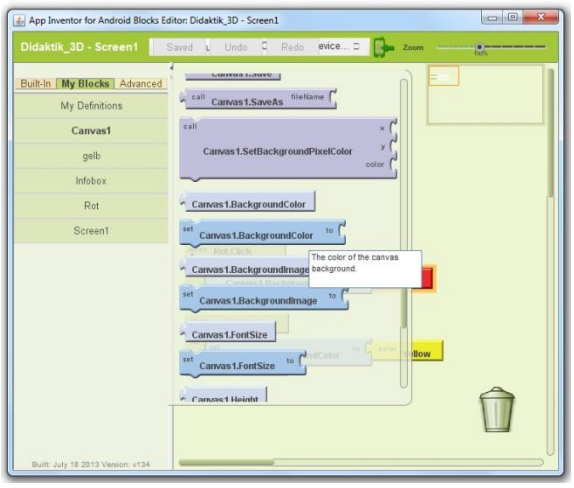
Im ersten Schritt wird - nach der Anmeldung - ein neues Projekt begonnen. Auf dem Android-Screen können jetzt Objekte aus dem linken Auswahlmenü durch Drag & Drop hinzugefügt werden. Im rechten Teil des Fensters können die Objekteigenschaften verändert werden.



Im oberen Bild befinden sich auf den Screen ganz oben eine Infobox, zwei Button sowie unten eine Canvas-Zeichenfläche.

Die erste App soll nun ganz simpel mittels Knopfdruck die Farbe der Zeichenfläche ändern. Beim Start der App ist die Farbe (Backgroundcolor) blau. Durch Drücken der beiden Buttons soll diese Farbe zwischen „rot“ und „gelb“ wechseln.

Dazu muss man den „Blocks Editor“ öffnen, indem man den entsprechenden Button in der Button-Leiste drückt. Es folgt ein Java-Download, den man bestätigen muss und schon öffnet sich ein separates Java-Fenster mit dem Blockeditor. Die mittlere Auswahl enthält nun die Objekte, welche im App-Inventor auf dem Android-Screen positioniert worden sind. Hier kann man nun alle Objekteigenschaften mittels Events beeinflussen.



Aus der Auswahl für das Objekt **Rot** (Button) kann der Event **Rot.Click** ausgewählt werden. Aus der Auswahl für das Objekt **Canvas1** (Zeichenfläche) kann **Canvas1.BackgroundColor** ausgewählt und bei **Rot.Click** eingesetzt werden. Dies bewirkt in der App, dass ein Druck auf den Button **Rot** tatsächlich die Farbe der Zeichenfläche auf „rot“ geändert wird.

Analog dazu wird nun der Methode **gelb.Click** des Buttons **gelb** die Attributsänderung der Zeichenfläche **Canvas1. BackgroundColor** in „gelb“ zugeordnet:

Wenn das im Blockeditor dann so aussieht startet man einen Android-Emulator, indem man auf den Button „*New emulator*“ drückt.

Nach einer kurzen Ladezeit ist der Emulator gestartet und kann mit dem Blockeditor verbunden werden. Das geschieht durch das Drücken des Buttons „*Connect to Device …*“. Jetzt werden die Daten unserer App auf den Emulator übertragen und die App gestartet.

Wenn alles problemlos geklappt hat, dann sollte die App nun auf Knopfdruck die Farbe des unteren Kastens ändern.

Nach dem erfolgreichen Probelauf soll die App natürlich auf das Smartphone übertragen werden. Auch das ist mit dem App-Inventor kein Problem. kehrt man zur Webseite mit dem App-Inventor zurück. Hier drückt man nun den Button „*Package for Phone*“. Es öffnet sich ein Kontxt-Menü mit drei Optionen.

Der Anwender hat nun folgende Auswahl:

Die fertige App kann als QR-Code verlinkt werden. Das erfordert auf dem Handy dieselbe Google-Anmeldung wie auf dem App-Inventor. Die zweite Möglichkeit ist das Herunterladen der App als apk-Datei auf den Computer. Nach dem Abspeichern kann diese dann auf das Handy übertragen werden. Die dritte Möglichkeit ist die sofortige Übertragung der App auf das (am Computer angeschlossene) Smartphone.

Wenn alles prima geklappt hat, dann kann man nun anfangen eigene Apps für den Unterricht zu entwickeln.

Beispiel einer App, die im Unterricht eigesetzt werden kann:

Bruchrechnung02.apk

Die App-Serie **Bruchrechnung** wurde mit dem MIT-App-Inventor für den Unterrichtseinsatz in Klasse 5/6 entwickelt, damit die Lernenden unabhängig vom Computerkabinett auf ihren Android-Smartphones die wichtigsten Übungen zur Bruchrechnung interaktiv ausführen können.

Bruchrechnung01.apk

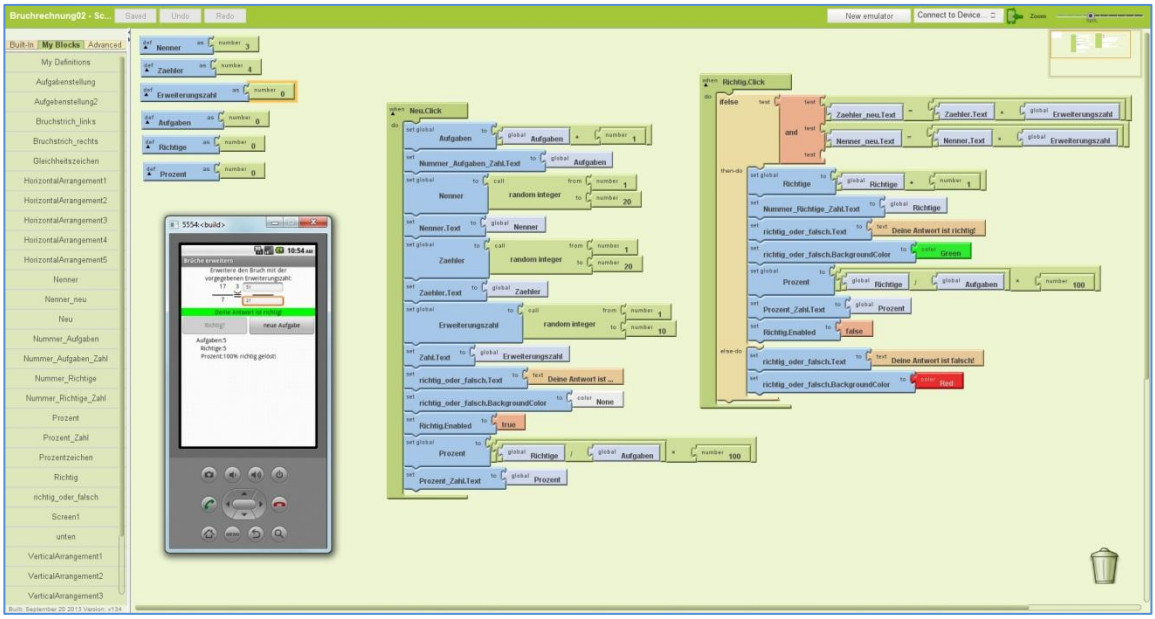
Mit der App „**Bruchteile erkennen“** können die Lernenden ihre Grundkenntnisse zu Bruchteilen trainieren.

Bruchrechnung02.apk

Die zweite App „**Brüche erweitern**“ trainiert das Erweitern nach vorgegebenen Erweiterungszahlen.

Bruchrechnung03.apk

Mit Hilfe der dritten App „**Brüche kürzen**“ kann das Finden einer Kürzungszahl beim Kürzen von Brüchen geübt werden.

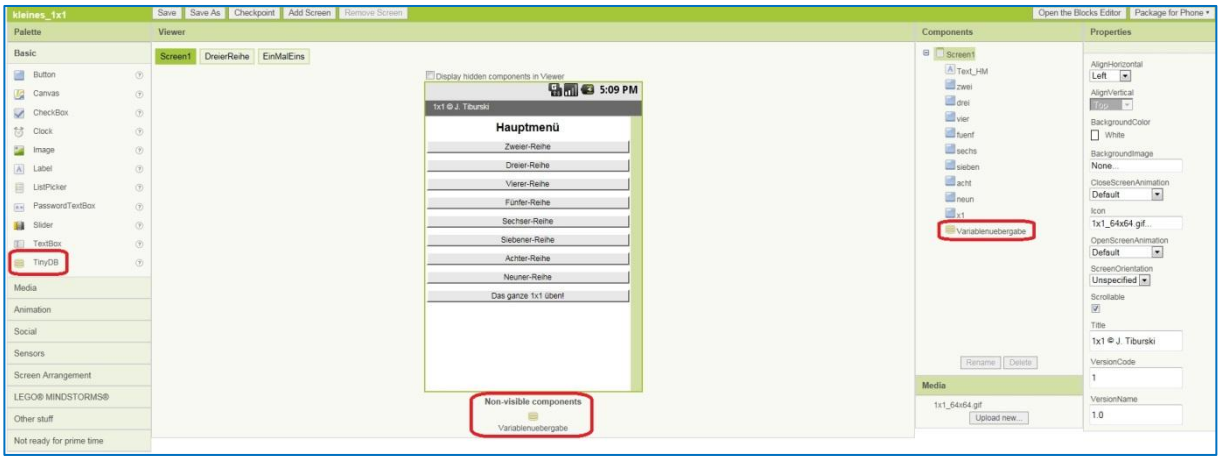


Oben zu sehen ist der Block-Editor des App-Inventors für die App **Bruchrechnung02.apk** mit integriertem Android-Emulator.

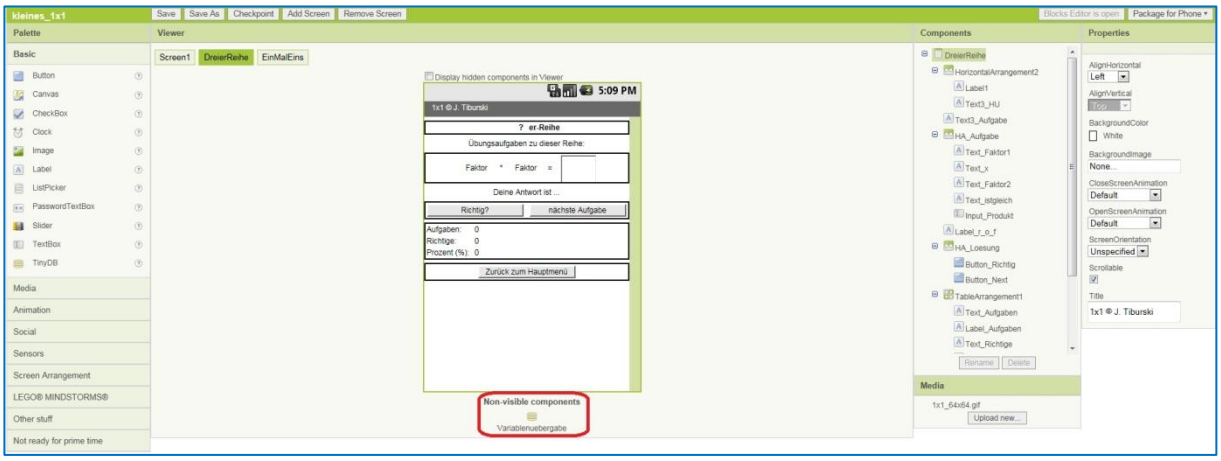
Ein nächster Schritt beim Programmieren von App's mit dem App-Inventor ist die Verwendung von Datenbanken. So ist es z.B. nur über die Definition einer Datenbank „TinyDB" möglich, Daten zwischen verschiedenen Screens einer App zu übergeben. Auch die Gestaltung von Quiz-App's basiert auf Datenbanken.

Im Folgenden wird nun erst einmal die Verwendung von DB's bei App's mit mehreren Screens erklärt. Dazu dient als Beispiel die App „kleines 1x1":

Im Startbildschirm wird ein Menü aus 9 Button erstellt. Hier soll der Anwender die Übungsreihe auswählen. Zusätzlich wird nun noch das Element **TinyDB** aus den Basis-Elementen ausgewählt und dem Screen hinzugefügt. Der Name „*Variablenuebergabe*" wurde aus naheliegenden Gründen gewählt:

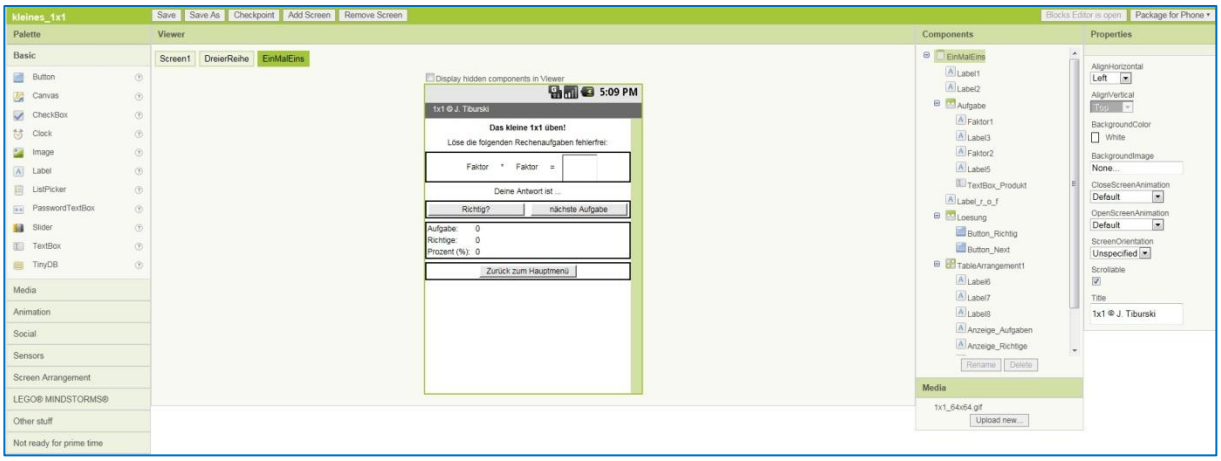


Der zweite Screen wird per Button „Add Screen" hinzugefügt und mit den bekannten Basiselementen ausgestattet. Damit die Variablenübergabe problemlos erfolgen kann muss auch dieser Screen die TinyDB „*Variablenuebergabe*" erhalten:

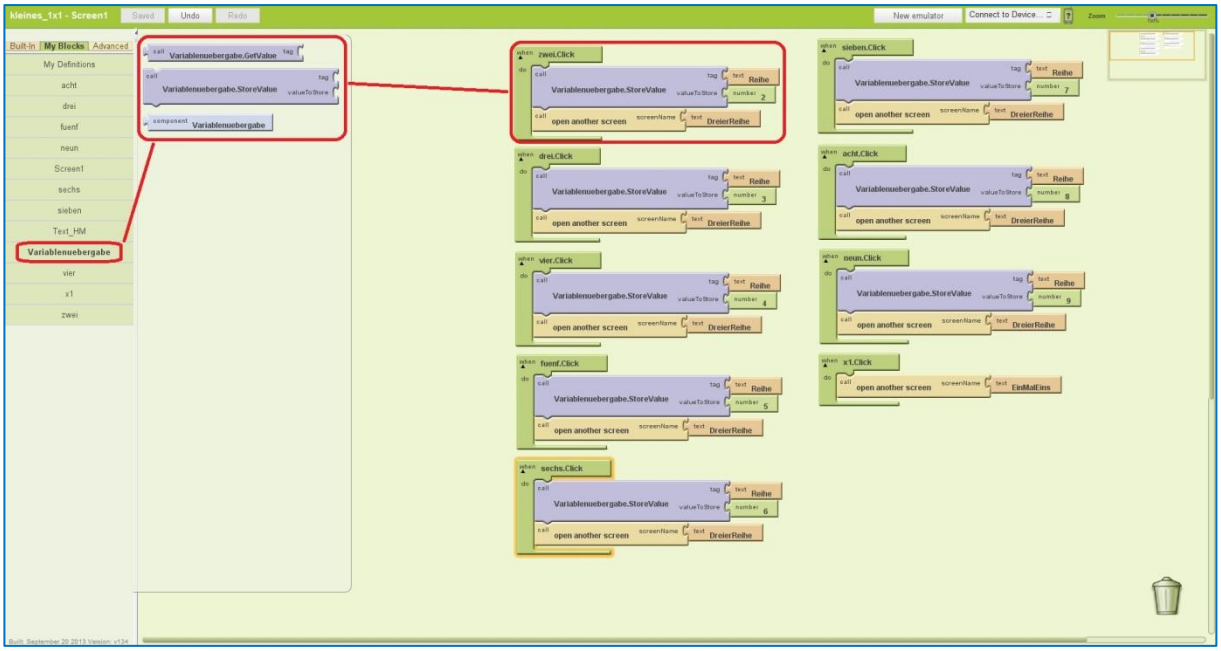


Nur wenn in beiden Screens dieselbe Datenbank definiert ist, dann greifen sie auf eine gemeinsame Datenbasis zu.

Im dritten Screen soll das gesamte „kleine 1x1" geübt werden. Es werden also beide Faktoren über den Zufallsgenerator erzeugt. Folglich braucht man für den dritten Screen auch keine DB hinzuzufügen.

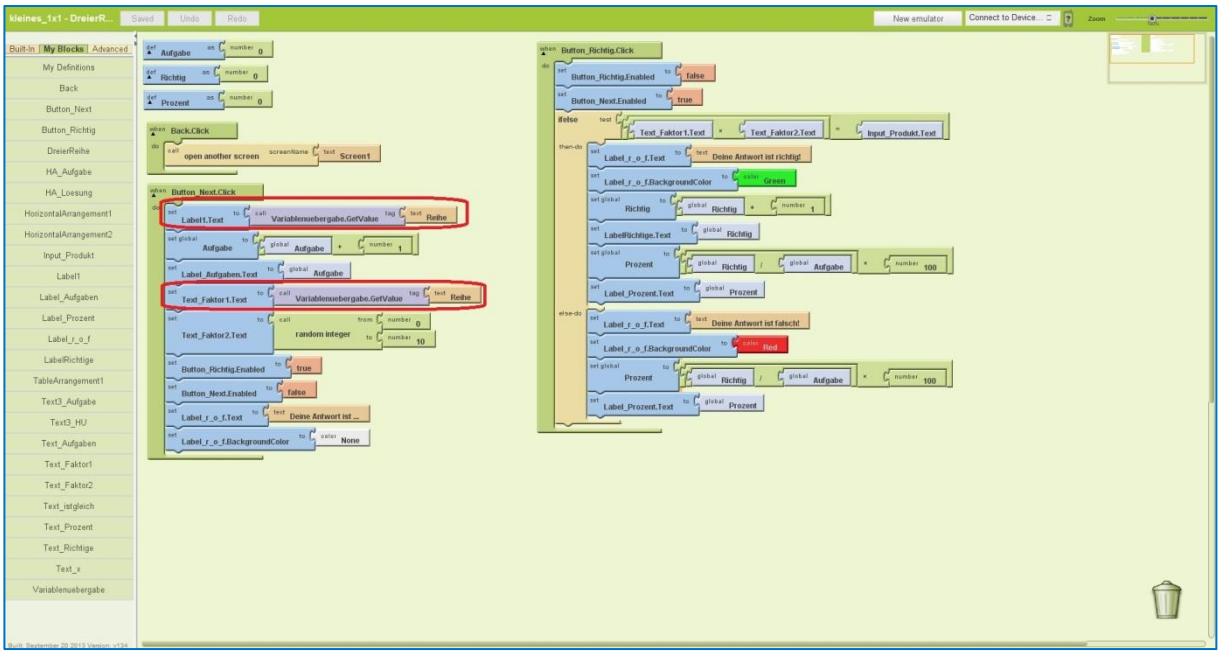


Wenn die drei Screen entworfen worden sind beginnt die Programmierung. Für die neun Button wird der Klick-Modus gewählt. Bei den Button 1 bis 8 wird mit Anklicken ein Datenbank-Eintrag in die DB „*Variablenuebergabe*“ getätigt. Unter dem Tag „Reihe“ (Der Tag-Name ist dabei frei wählbar!) wird der entsprechende Eintrag gespeichert. Danach geht es zum zweiten Screen. Der neunte Button braucht keinen DB-Eintrag sondern verweist direkt auf den letzten Screen:



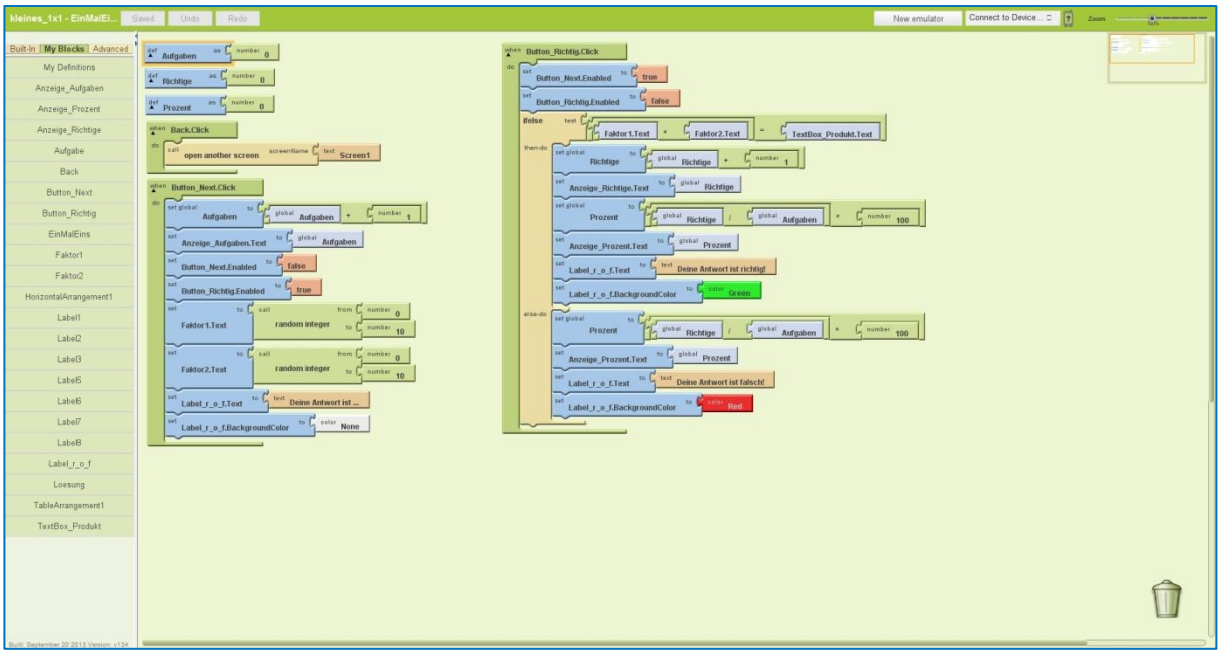
Im Grunde wird die Datenbank per „*Variablenuebergabe.StoreValue*“ mit nur einem einzigen Element gefüllt. Trotzdem ist sie unerlässlich, da der zweite Screen sonst nicht die Information bekommen würde, welche Reihe geübt werden soll.

Der zweite Screen ruft eben diese Information mittels „*Variablenuebergabe.GetValue*“ wieder ab. Beim ersten Aufruf wird die Screen-Überschrift aktualisiert, so dass der Anwender weiß, welche Reihe ausgewählt worden ist und beim zweiten Aufruf wird der erste Multiplikations-Faktor auf genau diesen Wert gesetzt. Der zweite Faktor wird mittels Zufallsgenerator „*random.intger*“ erzeugt. Somit werden auf diesem Screen nur Rechenaufgaben aus der ausgewählten Reihe erzeugt:



Nach dem Lösen der Aufgabe erfolgt der Klick auf den „Richtig?“-Button. Es erfolgt die Überprüfung des eingegebenen Ergebnisses sowie die Berechnung des Punktestandes und der Prozentangabe der richtig gelösten Aufgaben.

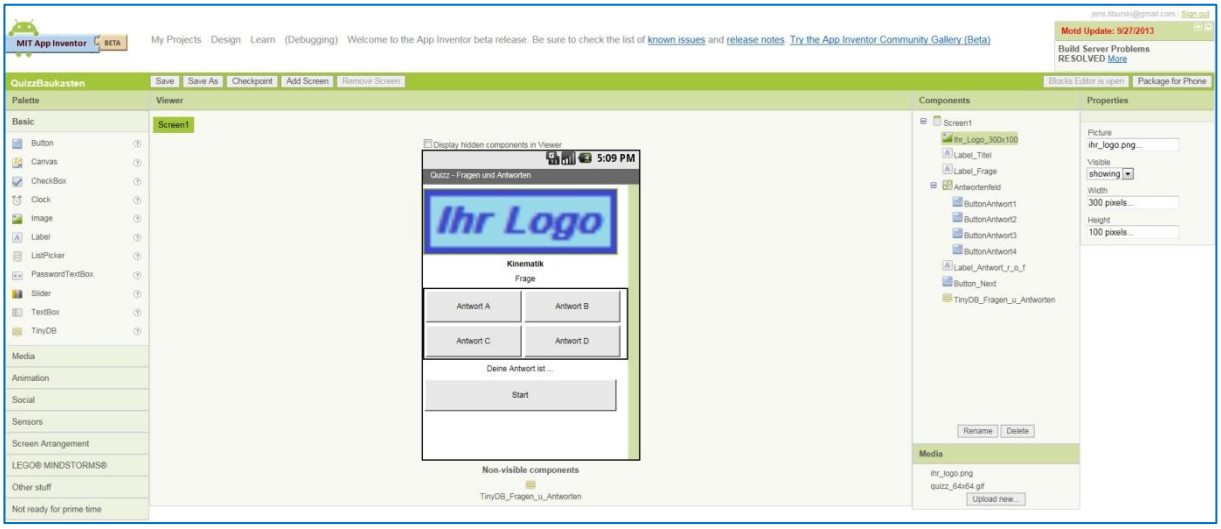
Entscheidet man sich im Hauptmenü jedoch dafür, das gesamte „1x1“ zu üben, gelangt man in den dritten Screen. Dafür ist keine Informationsübergabe notwendig – es entfällt die Einbindung der TinyDB für diesen Screen. Beide Faktoren werden nun per „*random.intger*“ erzeugt:



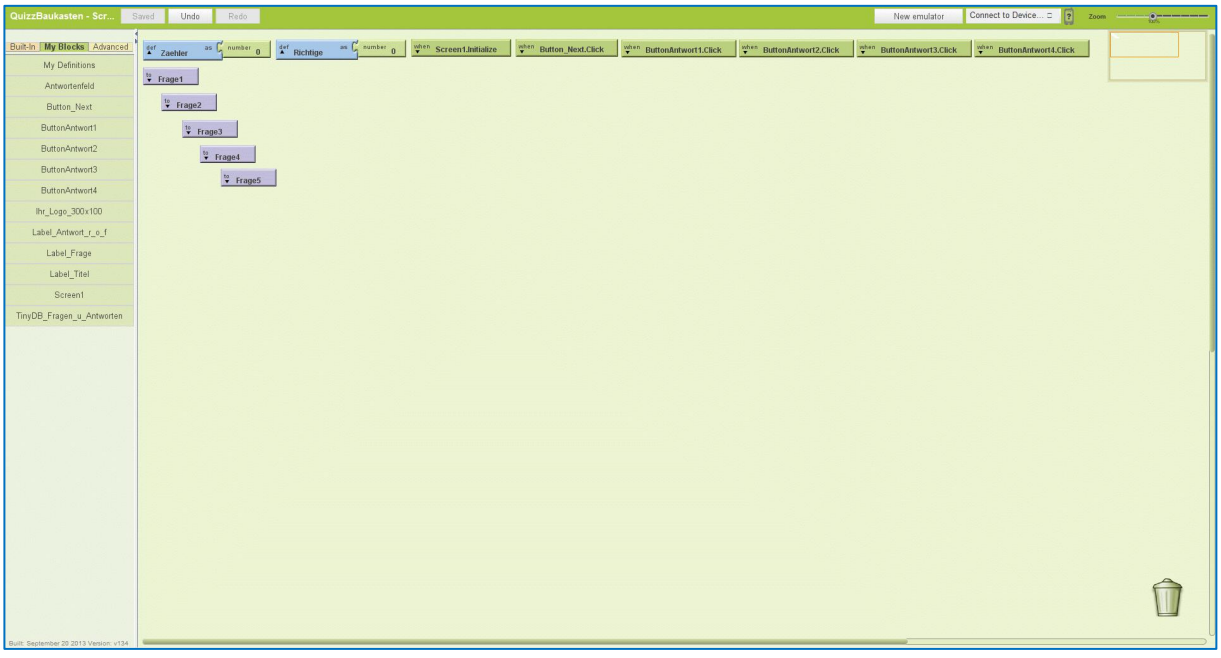
Die Auswertung erfolgt analog zur Auswertung der Eingabe im zweiten Screen.

Auch für Quiz-App's spielen Datenbanken eine entscheidende Rolle. Zum besseren Verständnis wird das anhand eines Quiz-Baukastens erläutert, der zum Download bereitsteht und von jedem Interessenten adaptiert werden kann.

Die Oberfläche ist sehr schlicht gehalten und verfügt bereits über alle Objekte, die das Quiz benötigt – inklusive einer TinyDB. Diese soll die Fragen und Antworten enthalten. Das Logo mit den Abmessungen 300 x 100 Pixel kann natürliche während der Adaption des Baukastens vom Anwender geändert werden:

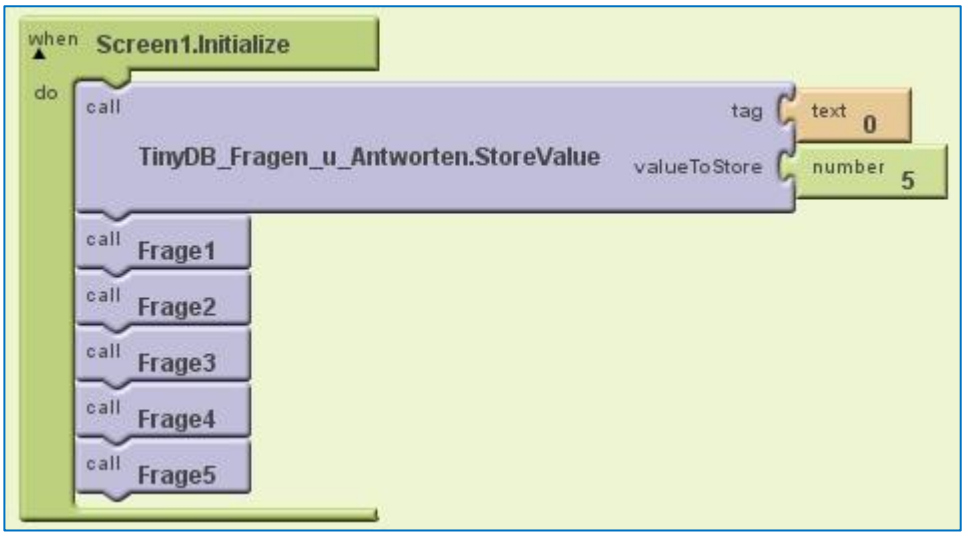


Im Block Editor sieht man die den Objekten zugeordneten Programmierinhalte:

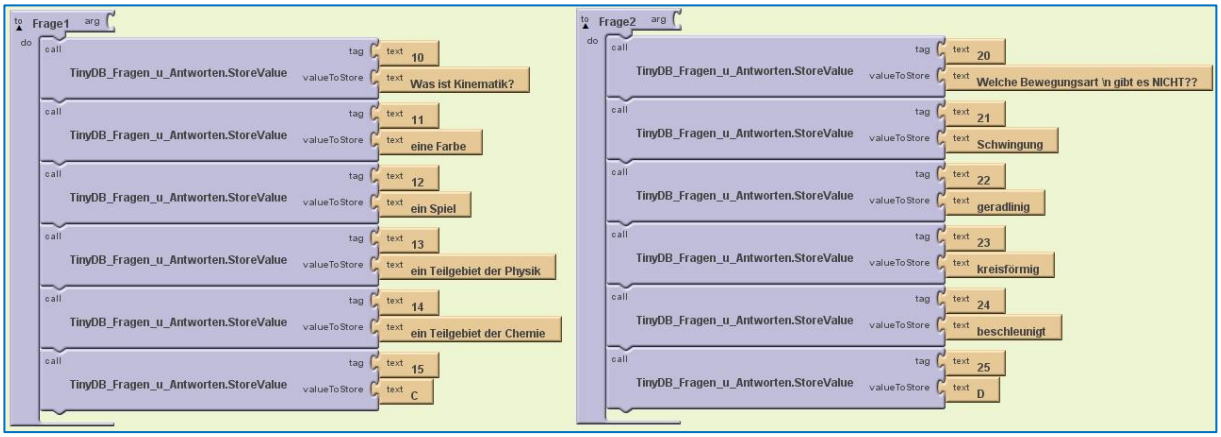


Das sind zuerst die Definition zweier Variablen. „*Screen1.Initialize*" legt die Anzahl der Quizfragen fest und ruft die Prozeduren auf, in denen die Fragen an die Datenbank übergeben werden. Es folgen die fünf Buttons. Der erste Button führt zur Auswertung der Eingabe bzw. zur nächsten Frage. Die anderen vier Buttons stehen für die Antworten 1 – 4.

Sobald die App gestartet wird, startet „*Screen1.Initialize*“ ebenfalls, d.h. Die Datenbank wird für die entsprechende Anzahl von Fragen eingerichtet und der Aufruf der – in diesem Fall fünf – Prozeduren erfolgt:



Alle Prozeduren haben hierbei denselben Aufbau:

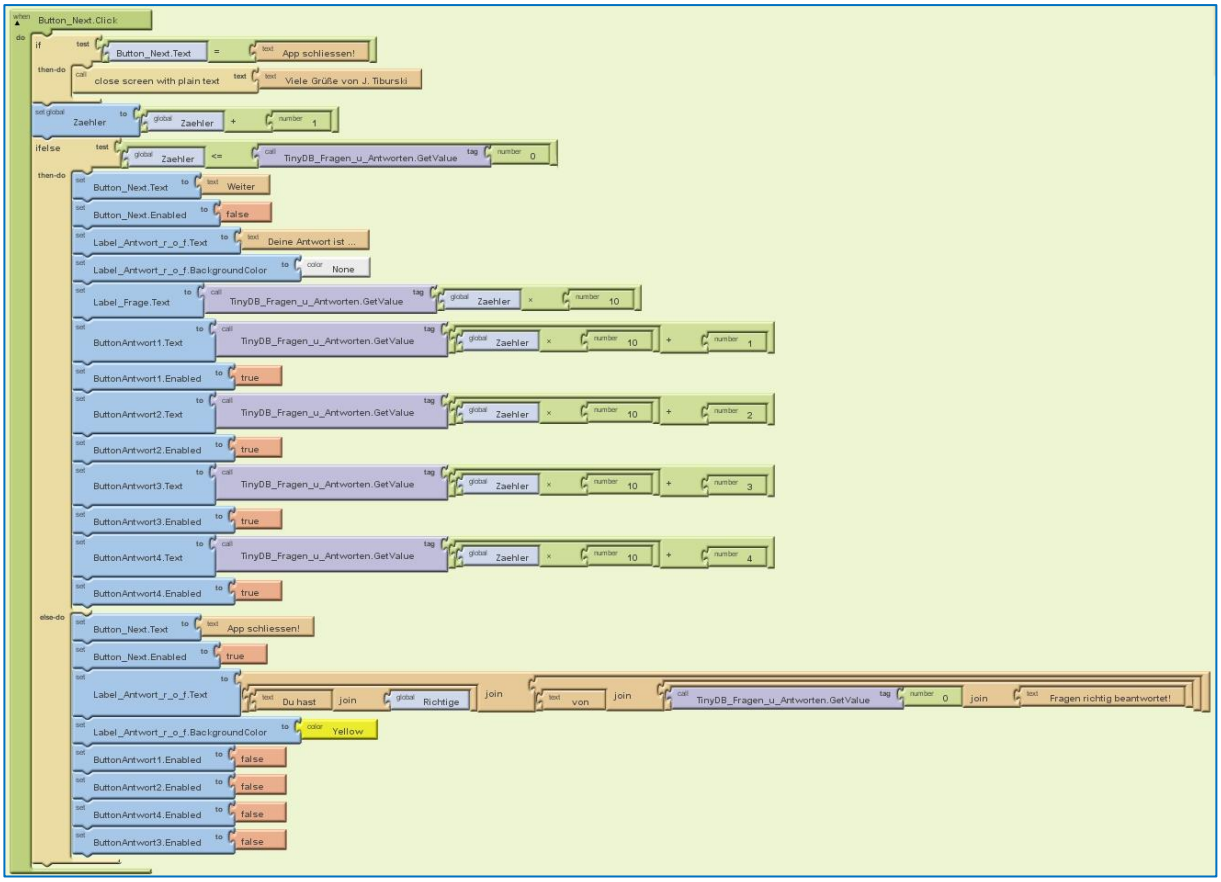


Die Tag-Angabe beinhaltet sozusagen den Namen, unter dem der eingegebene Wert (valueToStore) wieder aufgerufen werden kann. In diesem Beispiel richtet sich der Tag nach dem Fragen-Zähler. Die vorher definierte Variable „Zaehler“ enthält die Nummer der jeweils aktuellen Frage. Dieser Wert wird mit zehn multipliziert, so dass die Tags der ersten Frage mit 10 anfangen, die der zweiten Frage mit 20 u.s.w.!

Tag 10 enthält also die erste Frage. In den Tags 11 bis 14 stehen die vier Antwortmöglichkeiten und der Tag 15 enthält die Angabe der richtigen Lösung. Bei langen Fragen kann mit \n ein Zeilenumbruch erzwungen werden. Bei Zeilenumbrüchen bei den Antwortbuttons sollte man darauf achten, dass alle Buttons dieselbe Zeilenzahl haben. Sonst wirkt die Ansicht auf dem Screen nicht mehr symmetrisch …

In dieser Art kann man nun beliebig viele Fragen erzeugen. Wenn man im Block-Editor eine Prozedur markiert hat kann man mit den Tastenkombinationen STRG+C und STRG+V diese vervielfältigen. Somit entfällt das stupide Neuerstellen von Blöcken.

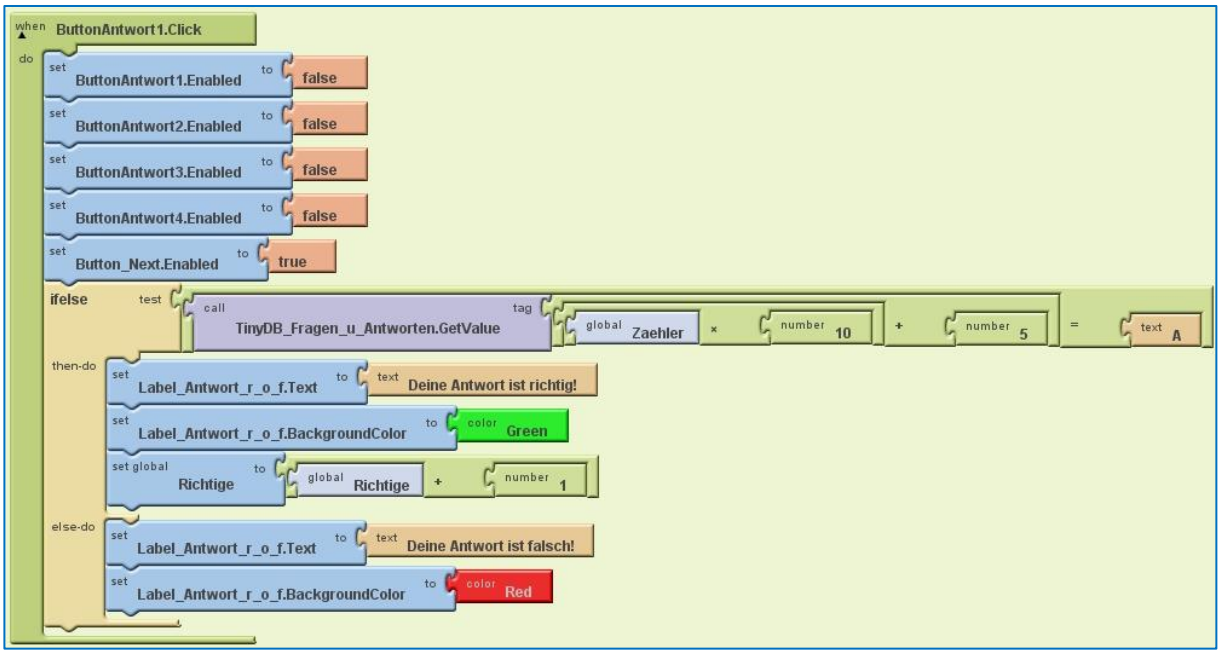
Damit wäre die Datenbank erzeugt und mit Daten gefüllt. Das Quiz startet nach dem Klick auf den „Next“-Button:

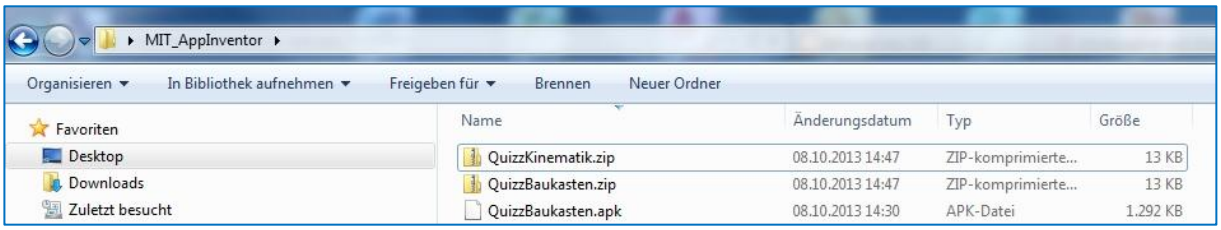


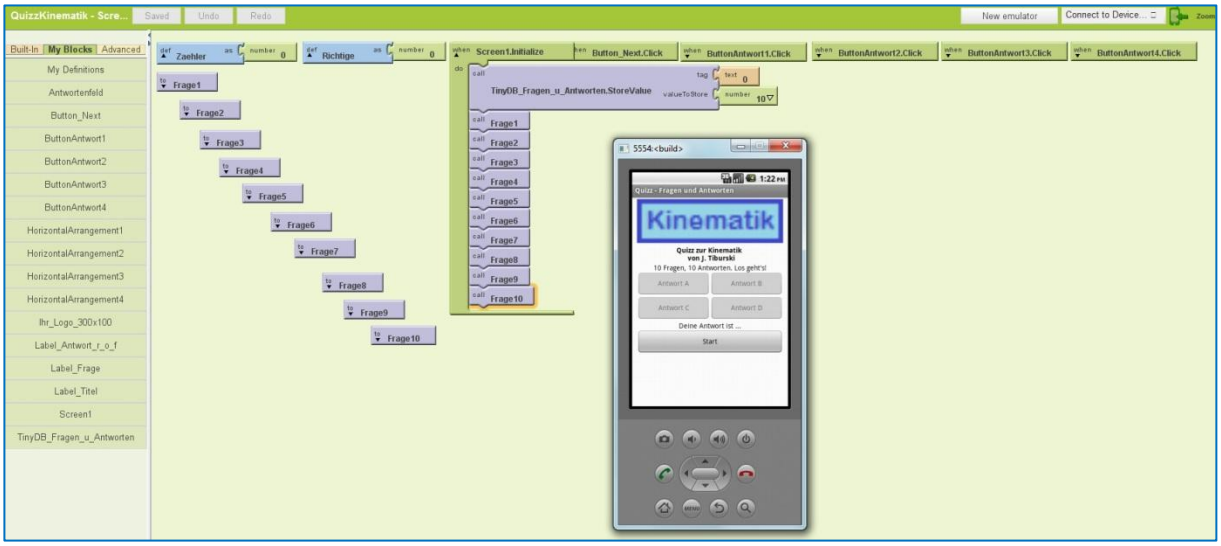
Der Zähler wird um eins erhöht, also beim Start von 0 auf 1 gesetzt. Es folgt ein Test, ob noch Fragen übrig sind. Fällt dieser Test positiv aus werden die Daten für die neue Frage mittels „Get.Value“ aus der Datenbank abgerufen und auf dem Screen angezeigt.

Fällt der Eingangstest negativ aus – sind also keine Fragen mehr verfügbar – dann bekommt der Next-Button die Beschriftung „App schliessen!“ und es erfolgt die Berechnung der Auswertung.

Die Auswertung wird gelb unterlegt angezeigt und beim nächsten Klick auf den Next-Button wird die App geschlossen.







Viel Spaß beim weiteren Probieren!