Deutsches Jungforscher Netzwerk
juFORUM e.V.

MIT App Inventor

Apps entwickeln - wie DU willst!

Mit dem MIT App Inventor

Deutsches Jungforschernetzwerk – juFORUM e.V.

Julian Bopp

www.juforum.de

facebook.com/juforum

Stand: 8. März 2014

Inhaltsverzeichnis

1 Einführung **2**

2 Los geht's **3**

3 Tutorial: Pizzarechner **6**

a) Steuerelemente platzieren 6

b) Verhalten programmieren 14

4 Tutorial: Zeichnen auf Canvas **20**

a) Steuerelemente platzieren 20

b) Verhalten programmieren 23

5 Simulieren von Apps **27**

6 Ausblick **28**

1 Einführung

Der MIT App Inventor[1] bietet die Möglichkeit, eine eigene App für Android zu erstellen.

Zielgruppe:	10.-12. Klasse
Interaktiv:	ja
Vorführdauer:	ca. 15 Min.

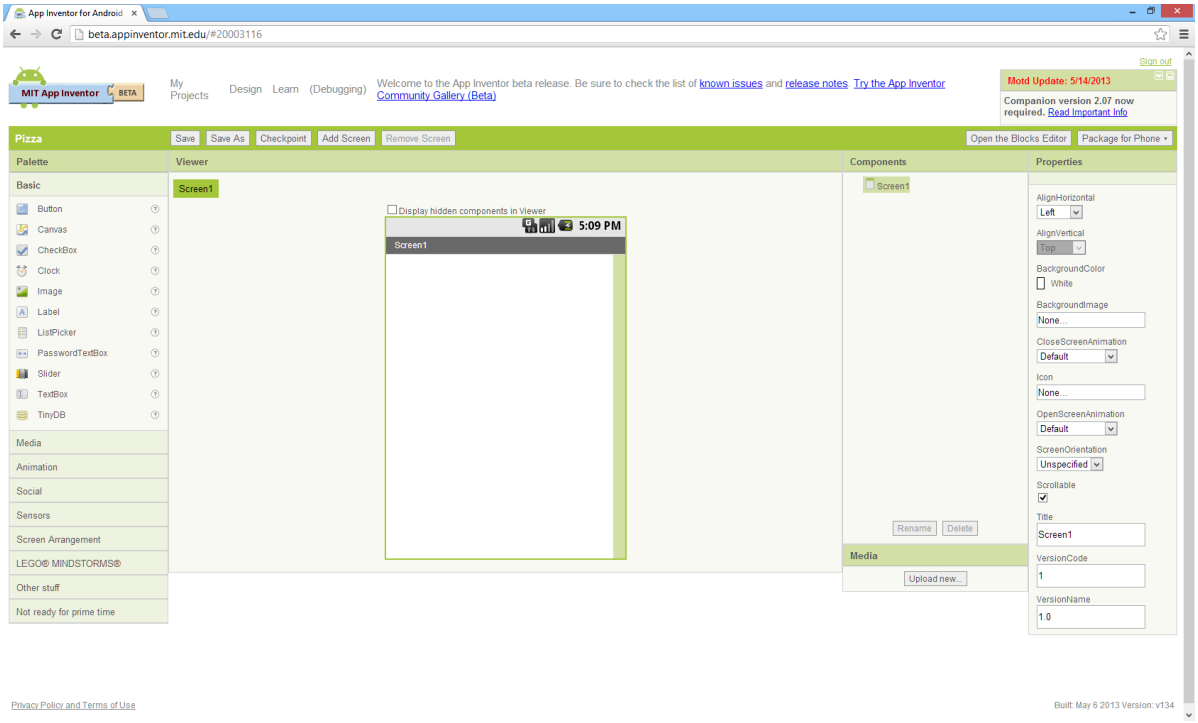
Es gibt Millionen von Apps zum Download, aber das, was man braucht, ist trotzdem nie dabei. Das Massachusetts Institute of Technology hat nun ein Programm erstellt, mit dem sich durch graphisches Programmieren und ohne Vorkenntnisse eine eigene Application für das Betriebssystem Android entwickeln lässt, ganz nach eigenem Geschmack.

[1] http://appinventor.mit.edu/

2 Los geht's

Der MIT App Inventor findet sich unter http://beta.appinventor.mit.edu/. Nach dem Anmelden über einen Google-Account, einem Klick auf „My Projects“ und dem Anlegen eines neuen Projekts über die Schaltfläche „New“ kann schon mit der Gestaltung der eigenen App begonnen werden. Um ggf. später Apps simulieren zu können, muss der „aiStarter“ installiert werden. Dieser kann unter http://appinventor.mit.edu/explore/ai2/setup-emulator.html heruntergeladen werden.

Abbildung 1: leeres Projekt



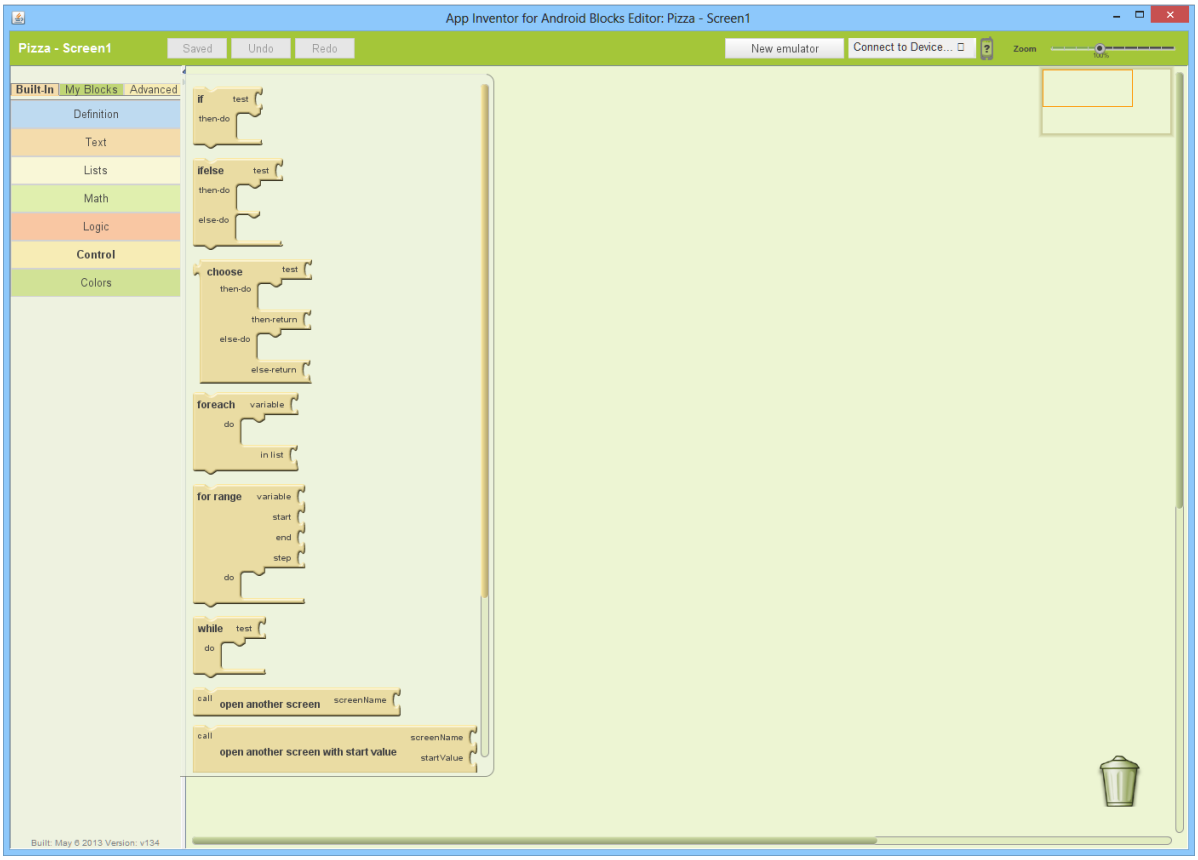
Das Fenster (siehe Abbildung 1) ist in vier Teile geteilt: In der Mitte („Viewer“) befindet sich ein fiktives Smartphonedisplay. Dieses kann mit Steuerelementen wie beispielsweise Tasten („Button“), Text („Label“), Eingabefeldern („TextBox“) oder Elementen, auf die gezeichnet werden kann („Canvas“), gefüllt werden. Diese Steuerelemente befinden sich im Bereich „Palette“ und können mit der Maus an die gewünschte Position im Viewer gezogen werden. Der Unterpunkt „Screen Arrangement“ der Palette enthält Steuerelemente, die beim zeilen- oder spaltenweisen Anordnen von Steuerelementen helfen.

Wurde ein Steuerelement platziert, ist es ratsam, ihm einen bezeichnenden Namen zu geben, der beschreibt, welchem Zweck das Element dient (soll ein Button beispielsweise

das Programm beenden, könnte ihm der Name „Beenden“ verliehen werden). Hierzu wird das jeweilige Steuerelement angeklickt. Anschließend wird in der Liste „Components“ auf „Rename“ geklickt, der neue Name eingegeben und bestätigt.

Jedes Steuerelement besitzt eine Reihe von Eigenschaften („Properties“), die sein Aussehen und Verhalten bestimmen. Wird ein Steuerelement ausgewählt, können dessen Eigenschaften auf der rechten Seite angepasst werden. Bei einem Button legt die Eigenschaft „Text“ fest, welche Beschriftung er trägt. Die Eigenschaften „Width“ und „Height“ legen die Größe eines Steuerelements fest.

Abbildung 2: Blocks Editor



Auf der rechten Seite, über den „Properties“, befindet sich eine Taste „Open the Blocks Editor“. Diese öffnet ein kleines Java-Programm, den „Blocks Editor“ (siehe Abbildung 2). Während im schon geöffneten Online-Editor nur das Aussehen der App definiert werden kann, kann im Blocks Editor das Verhalten der App programmiert werden. Um eine bestimmte Funktion zu programmieren müssen verschiedene Blöcke wie in einem Puzzle aneinandergereiht werden. Jeder Block beschreibt eine Handlung, die die App durchführt, sobald der jeweilige

Block erreicht wird (wie z.B. einen Text anzeigen, zwei Zahlen addieren, einen Ton abspielen, etc.). Nachdem die Handlung eines Blocks ausgeführt wurde, wird die des folgenden Blocks ausgeführt. Durch das Aneinanderreihen verschiedener Handlungen kann auf diese Weise eine bestimmte Funktion programmiert werden.

Im Blocks Editor sind die verfügbaren Blöcke in drei Kategorien eingeteilt (siehe linke Seite oben): „Built-In“, „My Blocks“ und „Advanced“. Die erste Kategorie enthält Blöcke, mit denen die Programmlogik definiert wird. Sie ist in Unterkategorien gegliedert: Es sind beispielsweise Blöcke vorhanden, die Text verarbeiten (Unterkategorie „Text“), Blöcke, die Zahlen verarbeiten (Unterkategorie „Math“) und Blöcke, die den Programmablauf steuern (Unterkategorie „Control“). „My Blocks“ enthält Blöcke, mit welchen auf im Online-Editor platzierte Steuerelemente zugegriffen werden kann (z.B. abfragen, welcher Text in eine Textbox eingegeben wurde oder auf Drücken eines Buttons reagieren).

Blöcke können ebenfalls mit der Maus in den Arbeitsbereich (hellgrüne Fläche) gezogen werden und dort aneinandergereiht werden. Überflüssige Blöcke werden gelöscht, indem sie auf den Mülleimer unten rechts gezogen und dort fallengelassen werden.

Durch Klick auf die Taste „New Emulator“ im Blocks Editor kann die App in einem virtuellen Smartphone, das das Betriebssystem Android emuliert, simuliert werden (siehe Abschnitt 5). Im Online-Editor befindet sich neben der Taste, über die der Blocks Editor geöffnet wird, ein Menü namens „Package for Phone“. Über dieses Menü kann die programmierte App entweder lokal auf dem Computer gespeichert („Download to this Computer“) oder direkt auf einem angeschlossenen Android-Smartphone installiert („Download to Connected Phone“) werden.

3 Tutorial: Pizzarechner

Im Folgenden soll eine kleine App programmiert werden (fertige App vgl. Abbildung 3), die berechnet, wie viele Partypizzen bestellt werden müssen, damit eine bestimmte Anzahl an Personen satt wird.

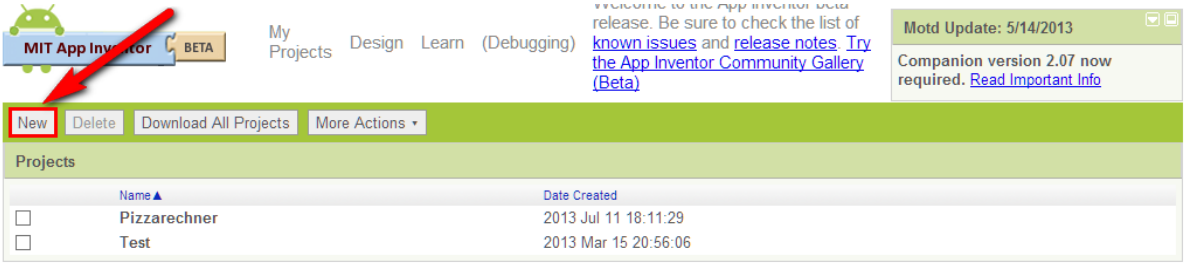
Abbildung 3: Pizzarechner

a) Steuerelemente platzieren

Nachdem der App-Inventor gestartet wurde, muss zuerst ein neues Projekt erstellt werden. Es wird auf „New“ geklickt, ein passender Name (wie „Pizzarechner“) eingegeben und bestätigt (siehe Abbildung 4).

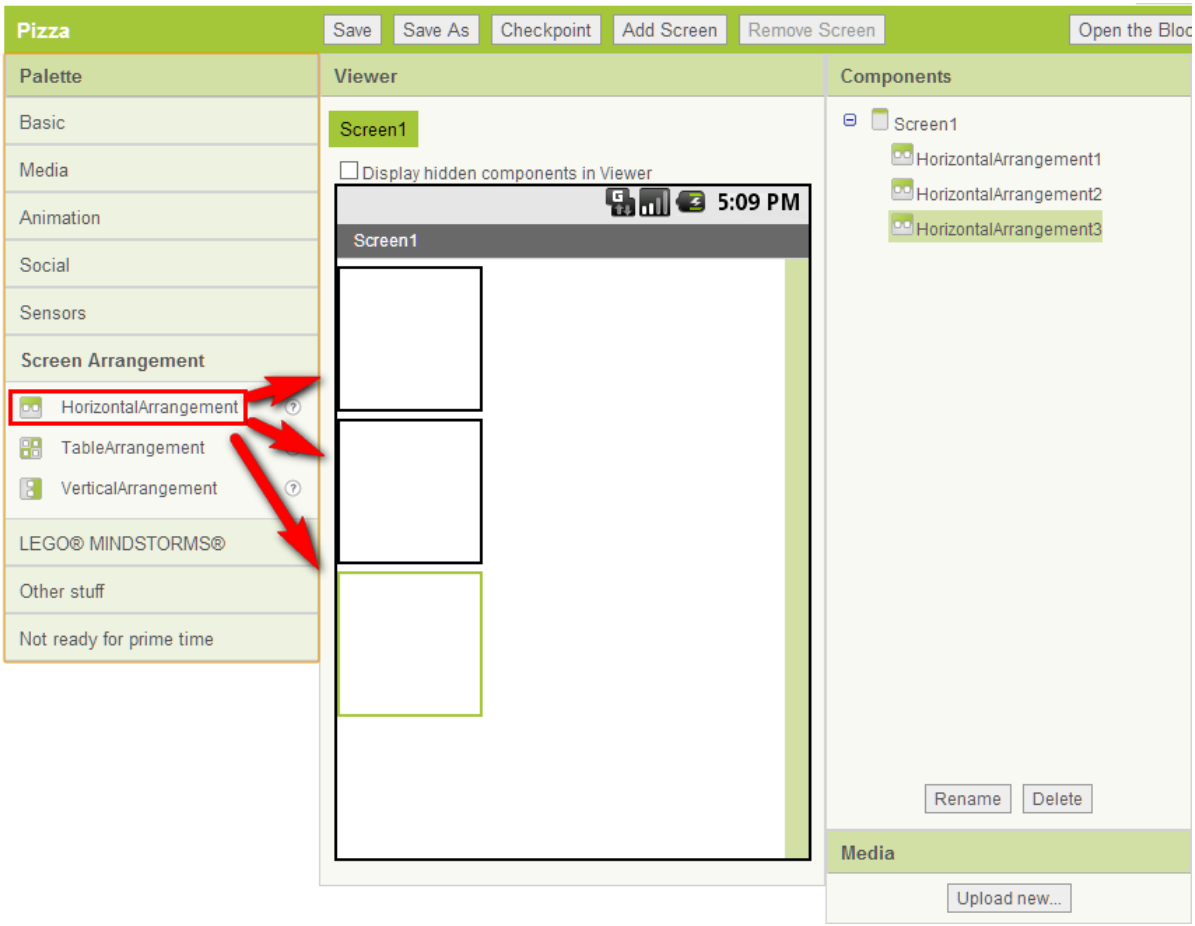
Nun können die Steuerelemente zur Eingabe der Personenzahl, zur Ausgabe der Anzahl der benötigten Pizzen und zur Steuerung der App platziert werden. Um Steuerelemente

Abbildung 4: Neues Projekt erstellen



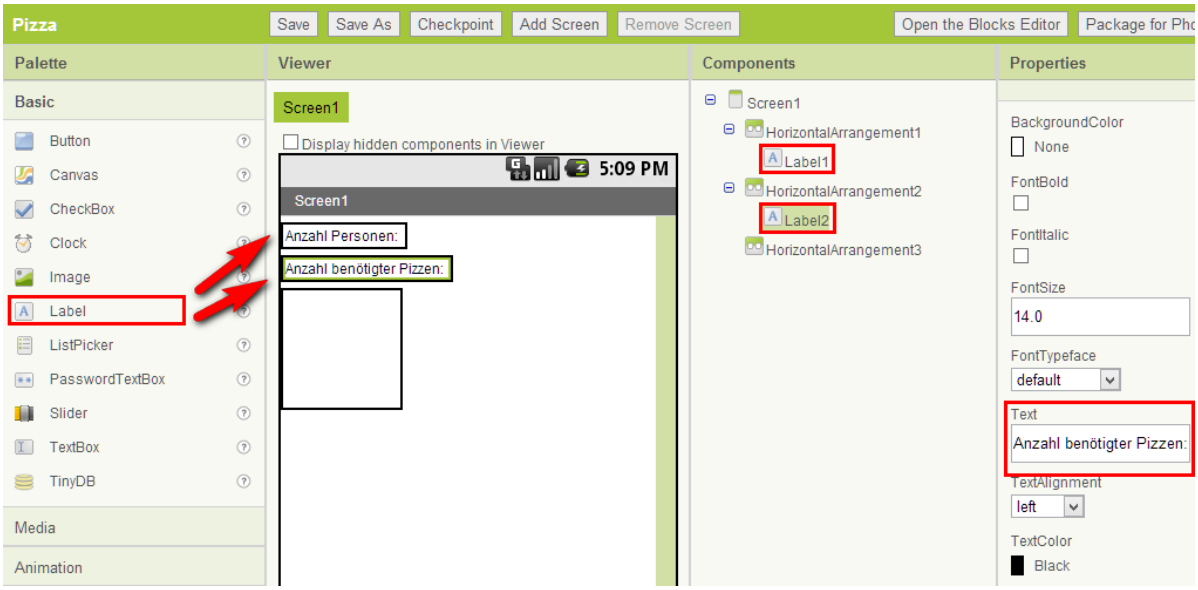
nebeneinander platzieren zu können (z.B. eine Beschreibung neben einer Textbox), werden Containerelemente des Typs „HorizontalArrangement“ benötigt. Es werden drei dieser Elemente aus der Palette (Subkategorie „Screen Arrangement“) mit der Maus auf das fiktive Smartphonedisplay (Viewer) gezogen und dort fallengelassen(siehe Abbildung 5). Die Steuerelemente sind hiermit platziert und erscheinen unter Components.

Abbildung 5: „HorizontalArrangement“ platzieren



Jeweils ein „Label“ (Subkategorie „Basic“) wird in den ersten beiden Containern platziert. Die Labels sollen kurze Infotexte darstellen. Die angezeigten Texte können, nachdem das zu bearbeitende Label angeklickt wurde, bei den Properties (Eigenschaft „Text“) bearbeitet werden (siehe Abbildung 6).

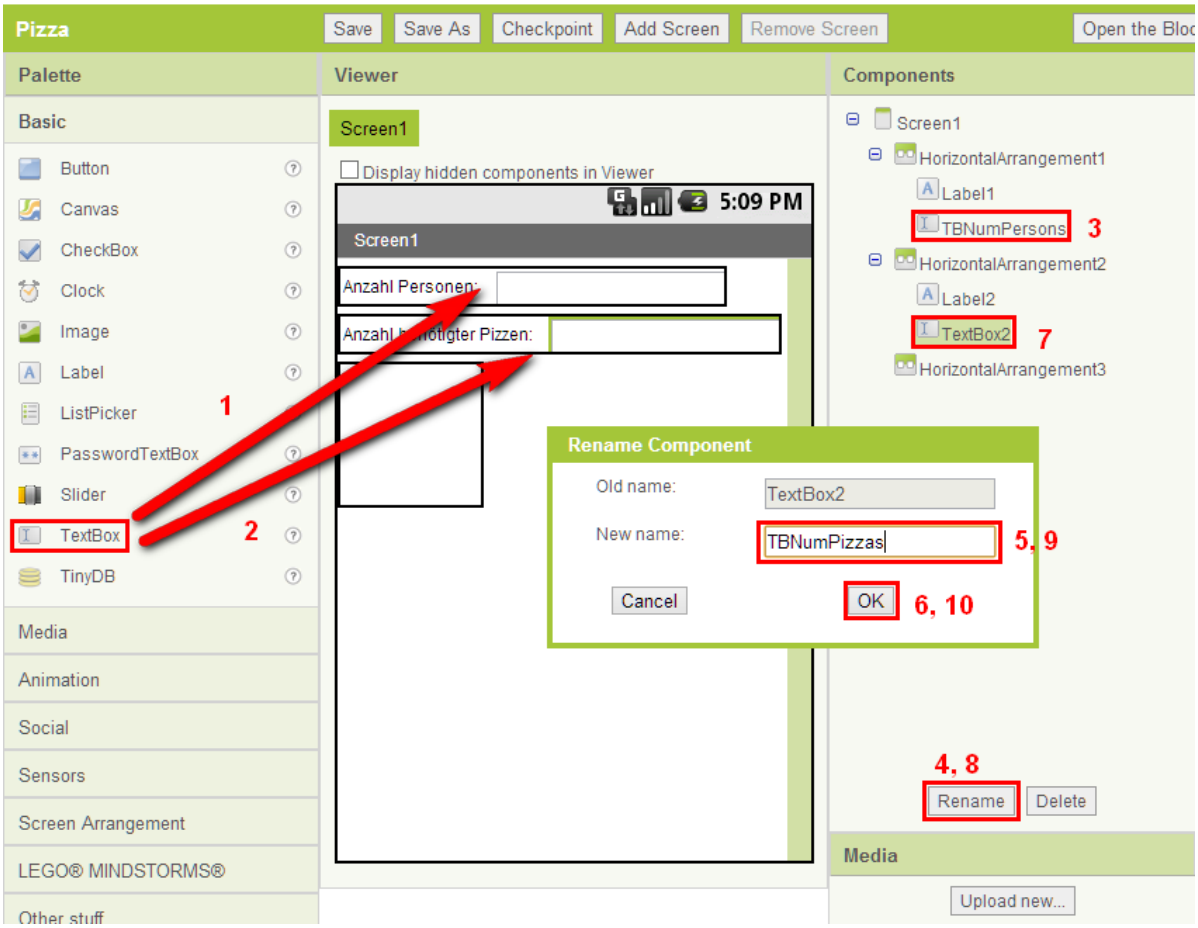
Abbildung 6: „Label“ platzieren und Text anpassen



Als nächstes werden Textboxen neben die zuvor platzierten Labels gesetzt (Schritte 1 und 2 Abbildung 7). Es ist sinnvoll, Steuerelementen aussagekräftige Namen zu geben, um später beim Programmieren zu wissen, welches Element sich unter welchem Namen verbirgt. Es wird wie folgt vorgegangen: Die erste Textbox wird ausgewählt, indem der entsprechende Eintrag der Components-Liste angeklickt wird (Schritt 3). Nach einem Klick auf „Rename“ (Schritt 4) öffnet sich eine Dialogbox. Unter „New Name“ wird der neue Name eingegeben (Schritt 5) und bestätigt (Schritt 6). Im Beispiel setzt sich dieser aus der Abkürzung des Steuerelements („TextBox“ → „TB“) und dessen Verwendungszweck (hier beispielsweise „NumPersons“ für die Personenanzahl) zusammen. Anschließend wird mit der anderen Textbox analog verfahren (Schritte 7 bis 8).

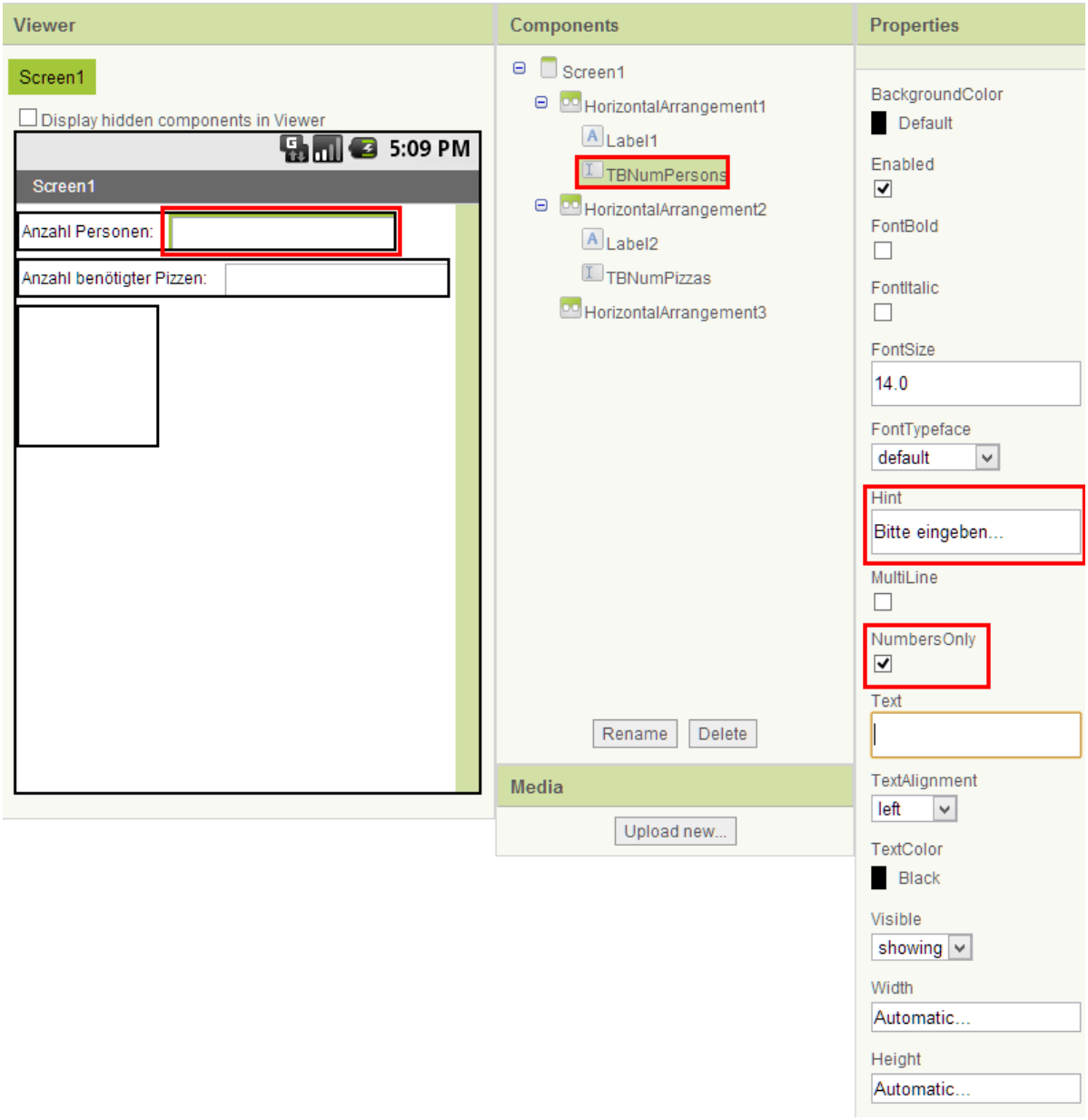
Bezüglich der Textboxen gilt es, ein paar weitere Eigenschaften anzupassen: Die Textbox „TBNumPersons“ soll der Eingabe der Personenzahl dienen. Demnach dürfen hier nur Zahlen eingegeben werden. Dies kann gewährleistet werden, indem die Eigenschaft „NumbersOnly“ bei den Properties gesetzt wird. Android wird zur Eingabe in eine mit dieser Eigenschaft versehenen Textbox eine Tastatur einblenden, die keine Buchstaben enthält. Ferner kann die Eigenschaft „Hint“ modifiziert werden. Der Hinweis (engl. Hint) ist ein klei-

Abbildung 7: „TextBox“ platzieren und umbenennen



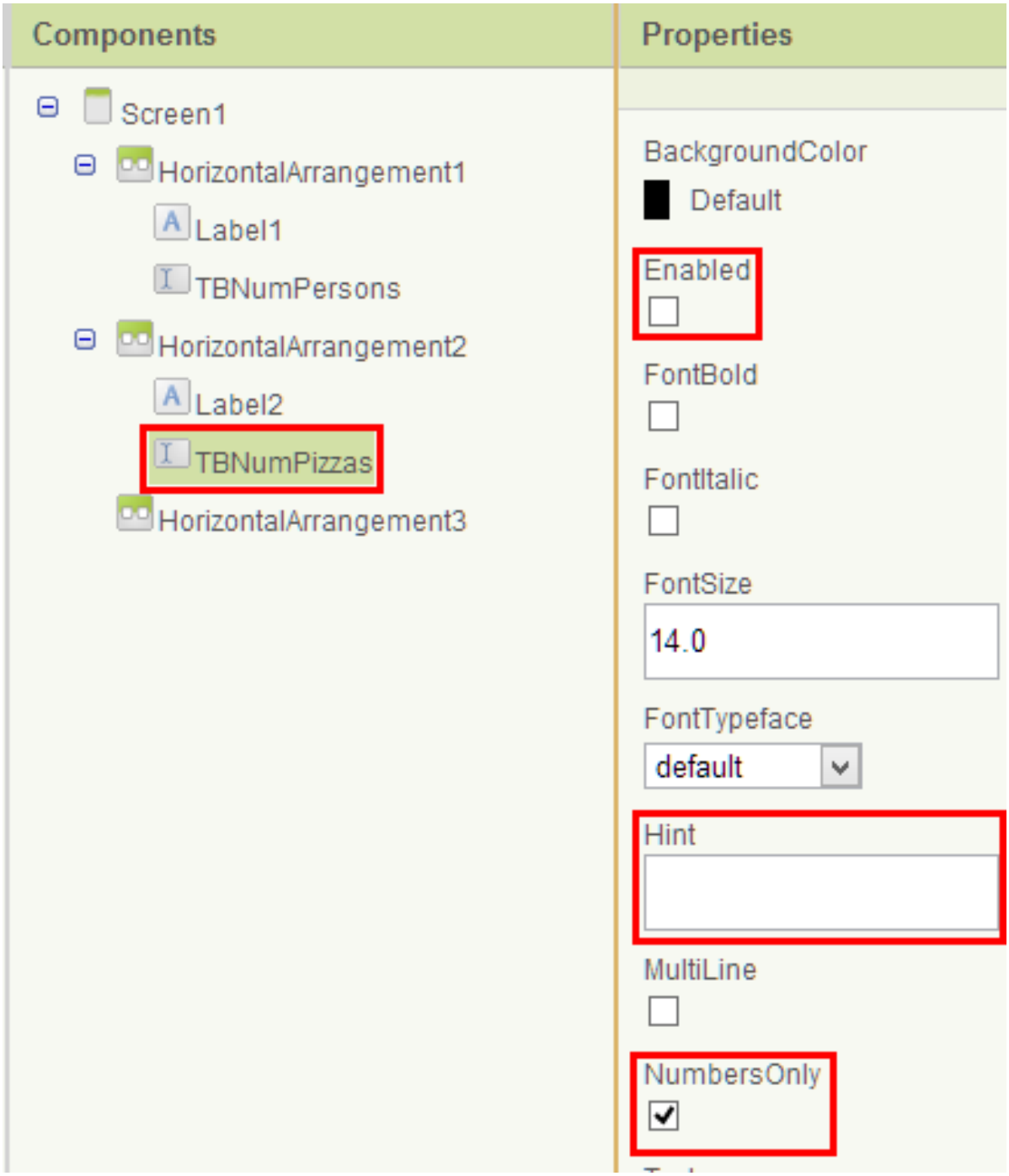
ner Infotext, der so lange in der Textbox angezeigt wird, bis eigener Text eingegeben wird (siehe Abbildung 8). Zum Ändern von Eigenschaften muss das entsprechende Steuerelement angeklickt worden sein.

Abbildung 8: TextBox 1 („TBNumPersons“) anpassen



Bei der Textbox „TBNumPizzas“ müssen im Wesentlichen dieselben Eigenschaften angepasst werden (siehe Abbildung 9). Der Hint kann hier leer bleiben, da diese Textbox nur zur Ausgabe von Daten dient. Es muss also kein Hinweistext angezeigt werden, der erklärt, welche Art von Daten eingegeben werden soll. Um die Eingabe von Daten zu verbieten, wird das Häkchen bei der Eigenschaft „Enabled“ entfernt. Hiermit reagiert das Steuerelement nicht mehr auf Benutzereingaben. Diese Eigenschaft ist bei beinahe jedem sichtbaren Steuerelement verfügbar.

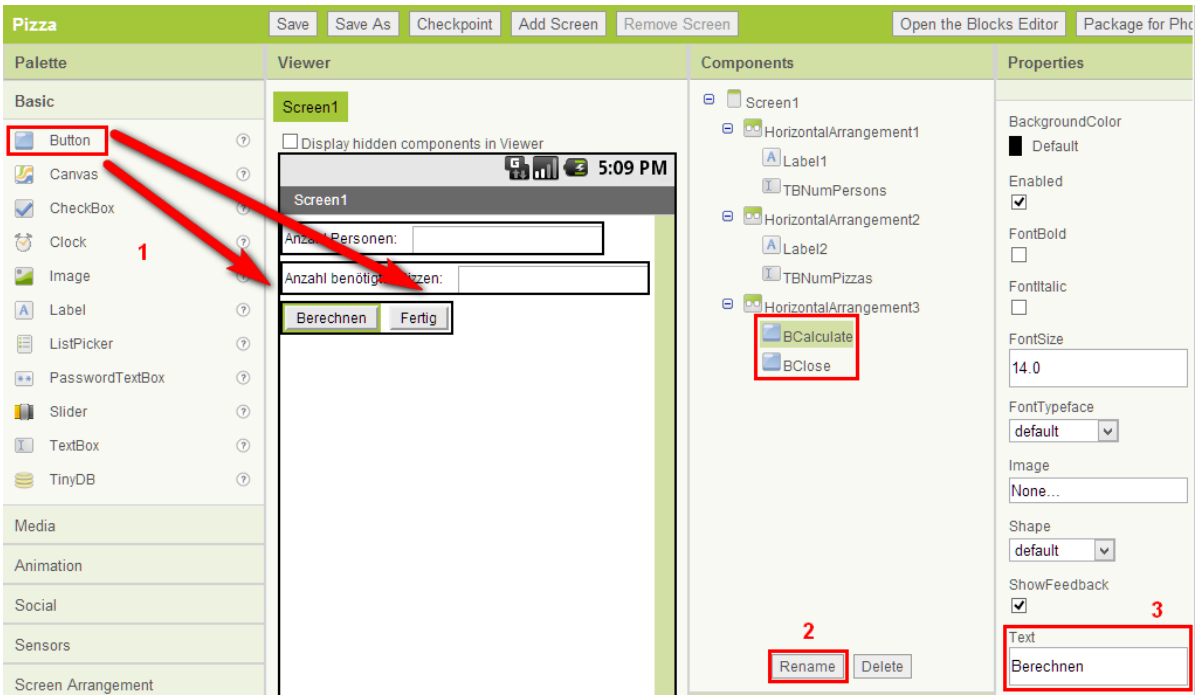
Abbildung 9: TextBox 2 („TBNumPizzas“) anpassen



Um die App steuern zu können, werden zwei Buttons benötigt. Einer soll die App schließen, der andere soll die Berechnung der Pizzenanzahl starten. Diese werden nebeneinander im untersten „HorizontalArrangement“ platziert (Schritt 1 Abbildung 10) und entsprechend benannt (Schritt 2). Die Beschriftung der Buttons wird festgelegt, indem ihre Eigenschaft

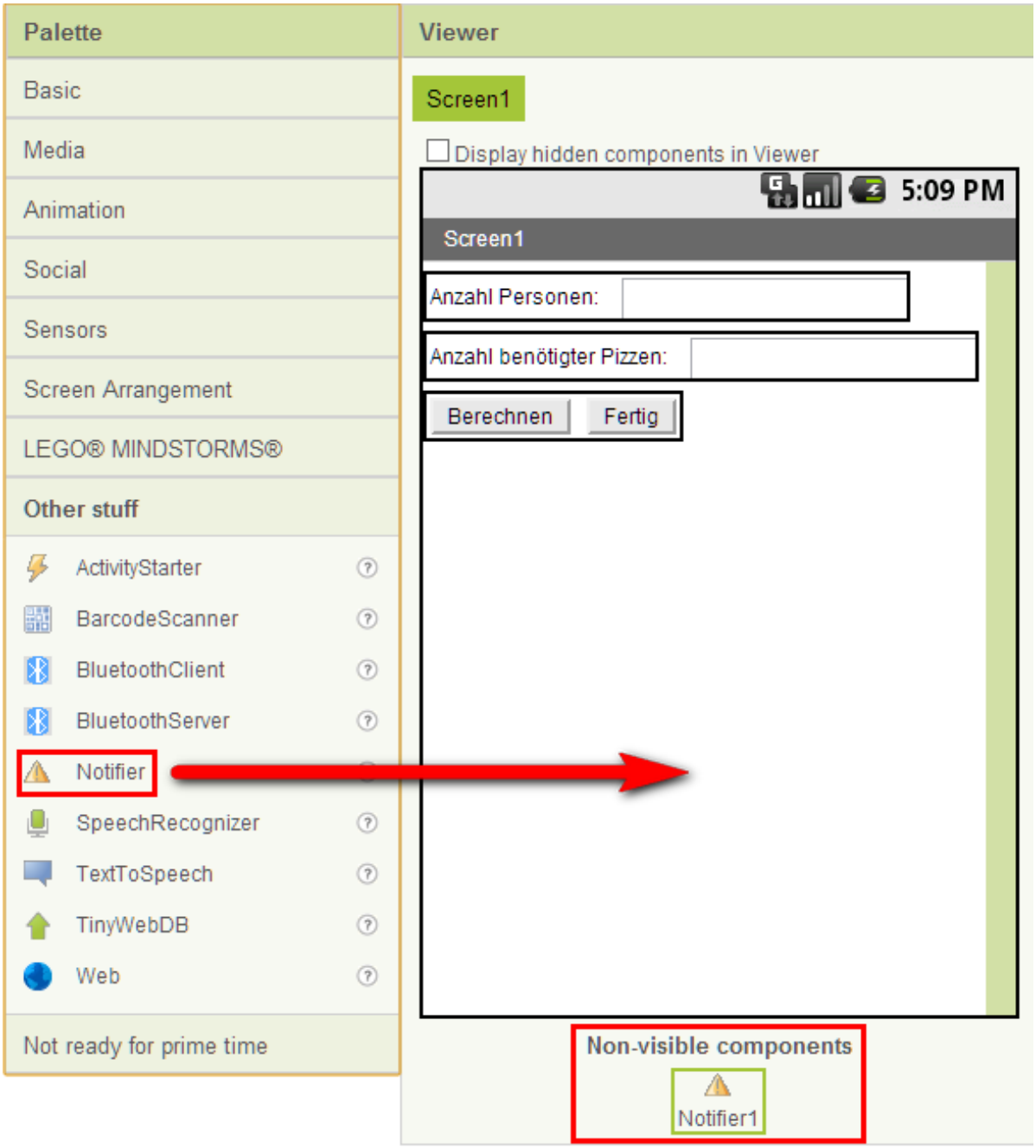
„Text“ geändert wird (Schritt 3).

Abbildung 10: „Button“ platzieren, umbenennen und anpassen



Der letzte gestalterische Schritt besteht im Hinzufügen eines unsichtbaren Steuerelements: Die „Notifier“-Komponente erlaubt das Anzeigen von aufpoppenden Meldungen (sog. Message Boxes). Sie befindet sich in der Subkategorie „Other stuff“ der Palette und wird zum Platzieren an einer beliebigen Stelle des virtuellen Displays fallengelassen (siehe Abbildung 11).

Abbildung 11: „Notifier“ einfügen



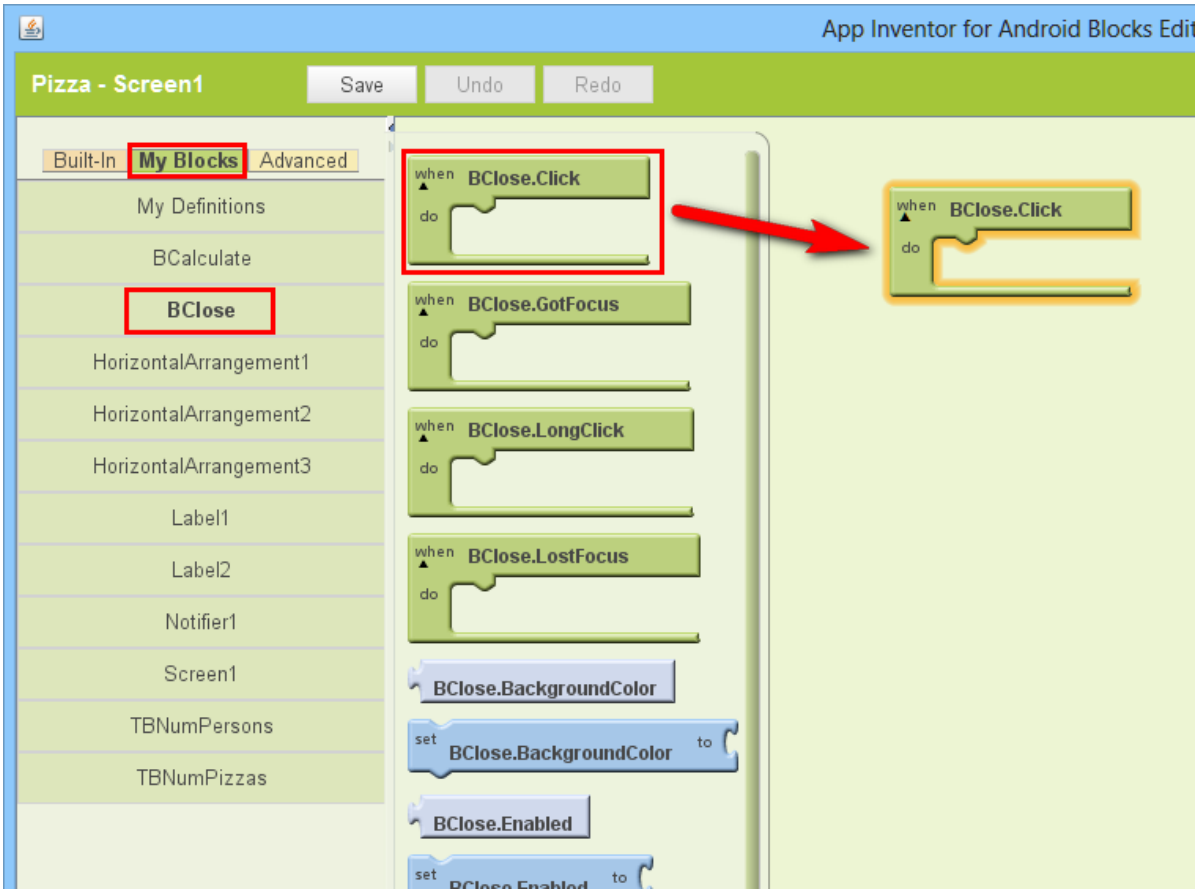
Zuletzt kann der Titel der App, der momentan „Screen1“ lautet, geändert werden, indem im Components-Bereich der entsprechende Eintrag ausgewählt und im Properties-Bereich die Eigenschaft „Title“ modifiziert wird.

b) Verhalten programmieren

Das Aussehen der App ist hiermit festgelegt. Verschiedene Steuerelemente sind vorhanden und einsatzbereit. In diesem Teil des Tutorials wird beschrieben, wie der Blocks Editor dazu verwendet wird, die Programmlogik sowie das Verhalten der App zu programmieren.

Der erste Schritt besteht darin, einen Klick auf den Button zu behandeln, der die App schließt (hier „BClose“ genannt. In der Kategorie „My Blocks“ sind die Namen aller platzierten Steuerelemente aufgelistet (siehe Abbildung 12). Darunter ist auch BClose zu finden. Wird dieser angeklickt, öffnet sich eine Liste von Blöcken. Schon der erste Block dieser Liste ist der Richtige. Mit der Maus wird er im Arbeitsbereich abgelegt. Der Block spannt eine Klammer auf und trägt den Namen „BClose.Click“. Wenn ein Klick auf BClose erfolgte, werden die vom Block eingeklammerten weiteren Blöcke von oben nach unten ausgeführt. Blöcke dieser Art folgen dem Benennungsschema „Name_des_Steuerelements.Event“ Wenn das Event eines entsprechenden Steuerelements eintritt, werden die eingeklammerten Blöcke ausgeführt. Das Event entspricht hier einem Klick (engl. Click). Beispielsweise würde das Event „LongClick“ genau dann eintreten, wenn ein lange andauernder Klick auf ein Steuerelement erfolgt.

Abbildung 12: Blocks Editor: Eventhandler für Klickevent von „BClose“ hinzufügen

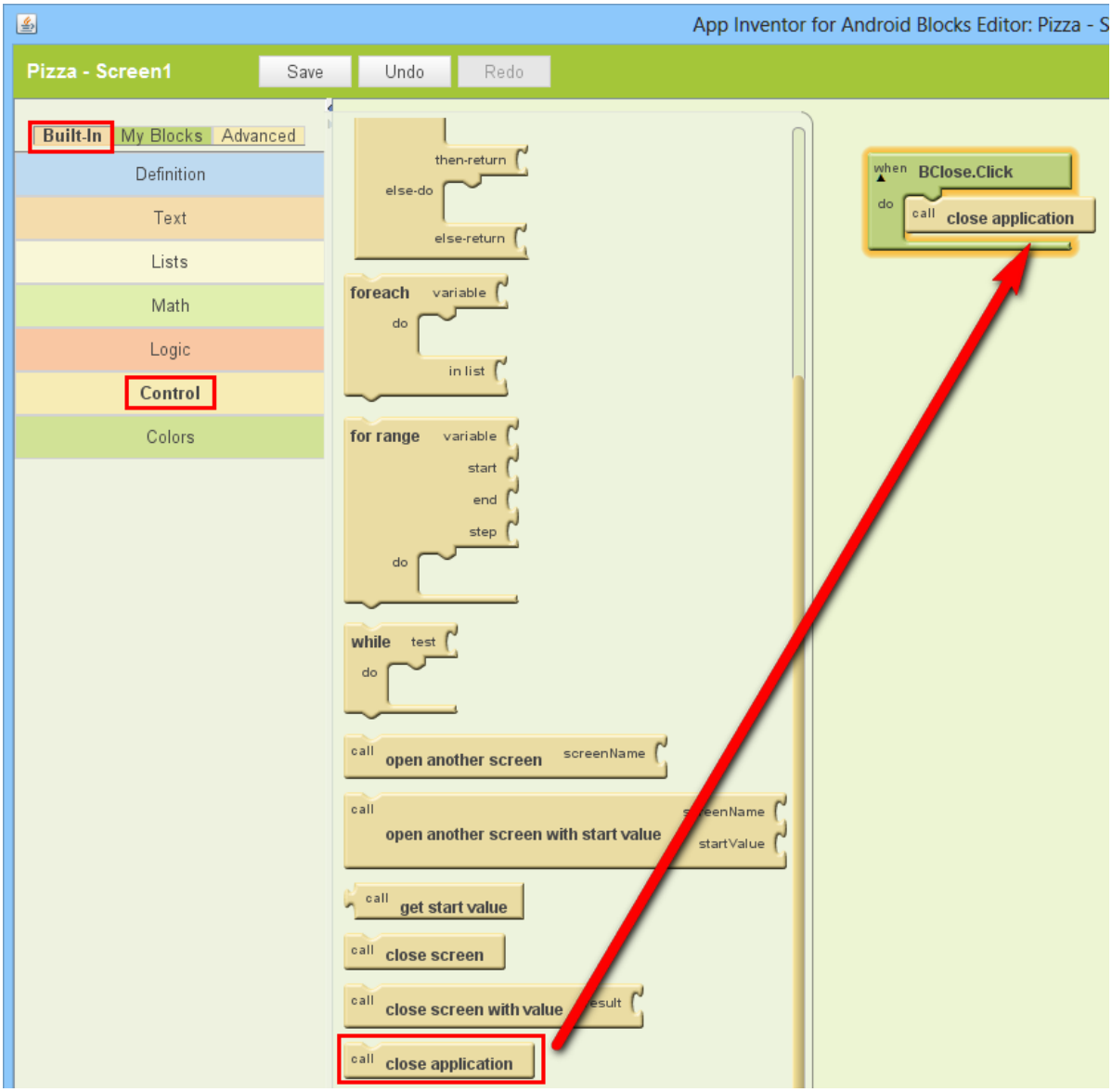


In der Kategorie „Built-In“, Subkategorie „Control“befindet sich ganz unten der Block „close application“. Während der zuerst eingefügte Block der Ablaufsteuerung des Programms dient (wie alle eine Klammer aufspannenden Blöcke), ist dieser Block ein Befehl. Er schließt die App. Der Block wird wird in die Klammer des „BClose.Click“-Blocks eingefügt, womit das Klicken auf BClose schon vollständig behandelt wäre (siehe Abbildung 13).

Um das Anklicken von BCalculate zu behandeln, wird analog zu „BClose.Click“ der Block „BCalculate.Click“ eingefügt. In diesen wird der ebenfalls in der Kategorie Built-In → Control verfügbare Block „ifelse“ gesetzt (Schritt 1 Abbildung 14). Bevor berechnet wird, wie viele Pizzen nötig sind, muss zuerst überprüft werden, ob die Personenanzahl einen sinnvollen Wert enthält. Zwar wurde bei der Textbox TBNumPersons die Eigenschaft „NumbersOnly“ gesetzt, die garantiert, dass keine Buchstaben eingegeben werden, dennoch muss überprüft werden, ob überhaupt eine Zahl eingegeben wurde und ob diese größer als 0 ist.

„ifelse“ (engl. für falls-sonst) führt Code bedingt aus: Abhängig davon, ob die Bedin-

Abbildung 13: Eventhandler für Klickevent von „BClose“ programmieren

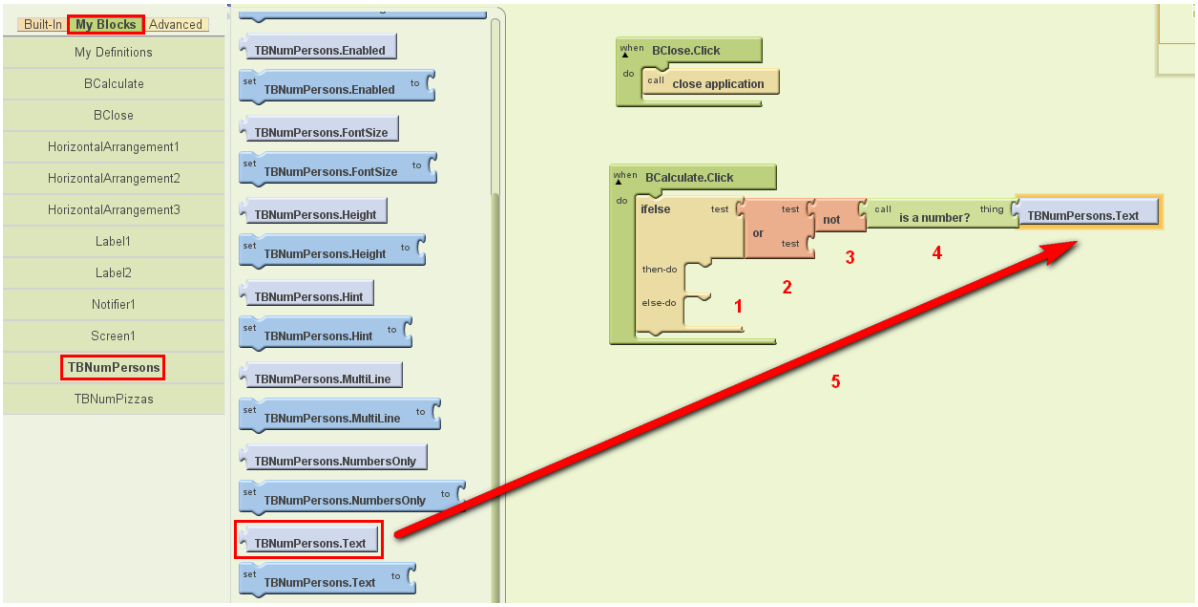


gung „test“ erfüllt ist, wird entweder die erste vom Block aufgespannte Klammer ausgeführt, sonst die Zweite. Wörtlich: Falls die Bedingung „test“ erfüllt ist, dann führe („then do“) Klammer 1 aus, sonst führe („else do“) Klammer 2 aus.

Die hier benötigte Bedingung setzt sich aus zwei Teilbedingungen zusammen. Die erste Teilbedingung testet, ob in TBNumPersons **keine** Zahl eingegeben wurde, die Zweite testet, ob die eingegebene Zahl (sofern eine eingegeben ist) negativ ist. Insgesamt soll, sobald die eine **oder** die andere Bedingung erfüllt ist, ein Hinweistext eingeblendet werden, der den Benutzer auffordert, eine gültige Zahl einzugeben. Um beide Bedingungen testen zu können

ist demnach ein ODER-Block (engl. or) erforderlich. Dieser wird an den ifelse-Block angehängt (Schritt 2 Abbildung 14). Der ODER-Block befindet sich unter Built-In → Logic. Der erste Zweig des Blocks testet, ob **keine** Nummer eingegeben wurde. Ein NICHT-Block (engl. not) gefolgt von einem „is a number?“-Block (engl. für „ist eine Zahl? “) werden also benötigt. Der erste Block befindet sich ebenfalls unter Built-In → Control (Schritt 3), der Zweite unter Built-In → Math (Schritt 4). Der „is a number?“-Block soll den in TBNumPersons eingegebenen Text überprüfen. Dieser kann über den Block „TBNumPersons.Text“ (My Blocks → TBNumPersons) verarbeitet werden (Schritt 5). Der Block folgt dem Benennungsschema „Name_des_Steuerelements.Eigenschaft“.

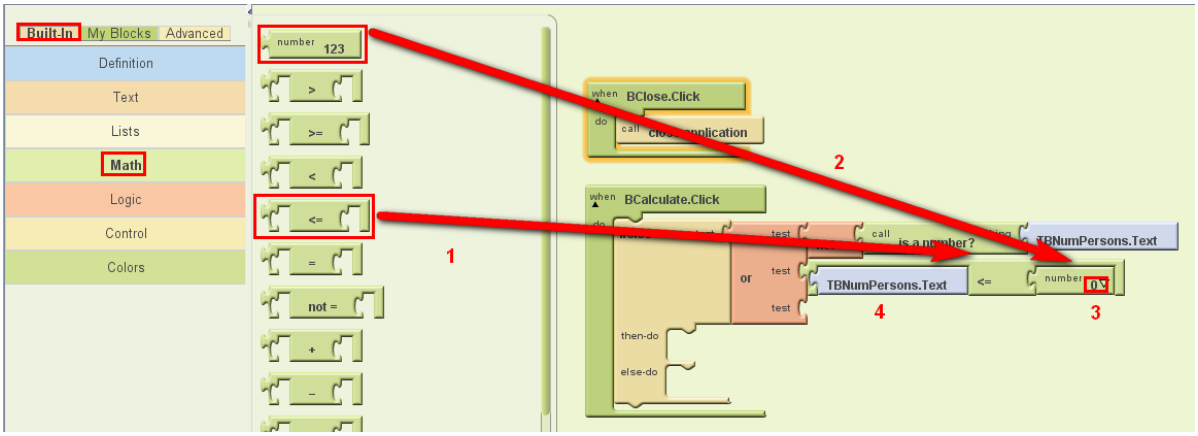
Abbildung 14: Eventhandler für Klickevent von „BCalculate“ programmieren - Schritt 1



Die zweite Teilbedingung überprüft, ob der Inhalt von TBNumPersons kleiner ($<$) oder gleich ($=$) 0 ist. Dazu wird der „$<=$“-Block (Built-In → Math) an den zweiten Testzweig des ODER-Blocks gehängt (Schritt 1 Abbildung 15). In die zweite Aussparung des „$<=$“-Blocks wird über den „number“-Block (Built-In → Math) die Zahl 0 eingefügt. Nach dem Einfügen des Blocks (Schritt 2) muss die Zahl durch einen Klick auf die zu Beginn eingestellte 123 und anschließende Eingabe der Zahl 0 angepasst werden (Schritt 3). In die erste Aussparung wird erneut der „TBNumPersons.Text“-Block eingefügt (Schritt 4).

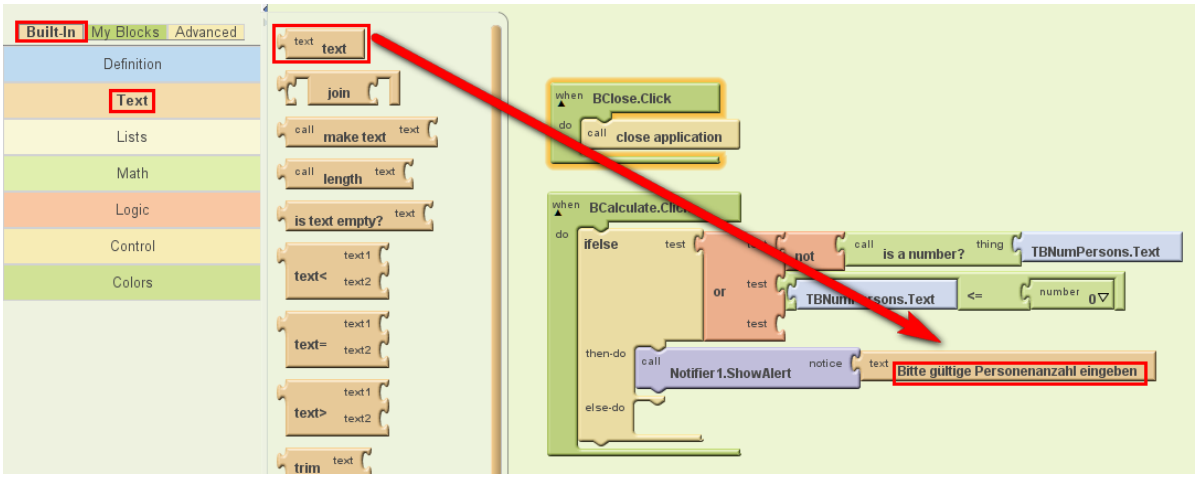
Ist mindestens eine der oben beschriebenen Teilbedingungen erfüllt, soll eine Meldung an den Benutzer ausgegeben werden. Dazu bietet sich die im letzten Schritt des Platzierens der Steuerelemente eingefügte „Notifier“-Komponente an. Unter My Blocks → Notifier1 findet sich der Block „Notifier1.ShowAlert“. Dieser wird in der ersten Klammer des ifelse-

Abbildung 15: Eventhandler für Klickevent von „BCalculate“ programmieren - Schritt 2



Blocks eingefügt. Sobald der Block ausgeführt wird, lässt er den in seinen Eingang „notice“ eingegebenen Text aufpoppen. Ein „text“-Block (Built-In → Text) wird demnach an den Eingang gehängt (siehe Abbildung 16). Der Standardtext lässt sich wie beim „number“-Block durch Klicken ändern.

Abbildung 16: Eventhandler für Klickevent von „BCalculate“ programmieren - Schritt 3



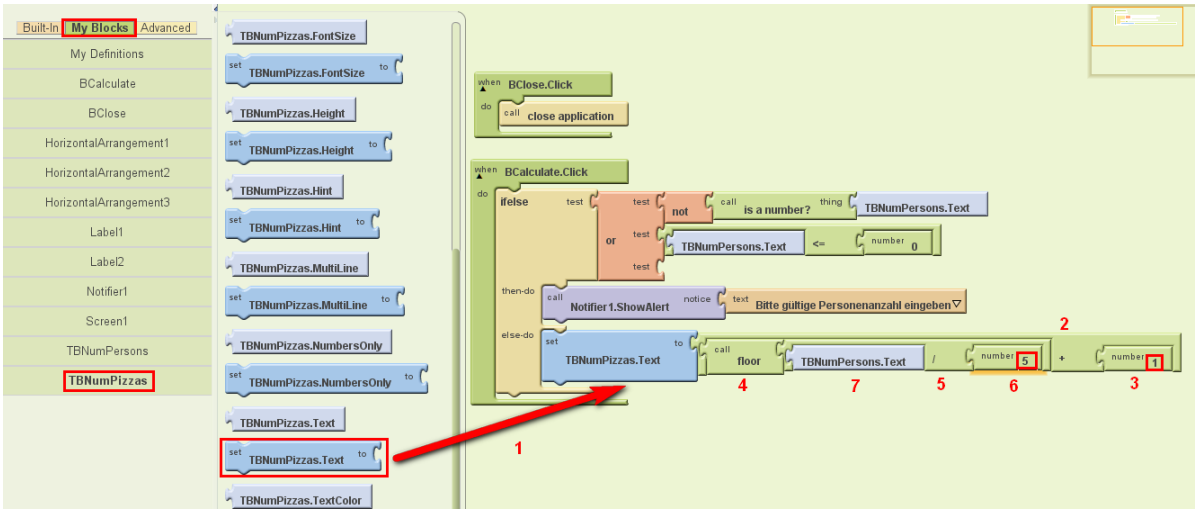
Ist keine der Teilbedingungen erfüllt, kann endlich die Anzahl der benötigten Pizzen berechnet und in TBNumPizzas ausgegeben werden. Zuvor wurde die Eigenschaft „Text“der Textbox TBNumPersons gelesen. Nun soll die gleichnamige Eigenschaft von TBNumPizzas geschrieben werden. Hierzu steht unter My Blocks → TBNumPizzas der Befehl „set TBNumPizzas.Text“ zur Verfügung. Dieser wird in die zweite Klammer des ifelse-Blocks eingefügt (Schritt 1 Abbildung 17). Mit entsprechenden Blöcken aus Built-In → Math wird

folgende Formel zur Berechnung der Pizzenanzahl umgesetzt:

$$AnzahlPizzen = floor\left(\frac{AnzahlPersonen}{5}\right) + 1$$

Die Funktion $floor$ rundet dabei immer ab. Es wird ein „+“-Block an den zuvor eingefügten Block angefügt (Schritt 2). In dessen zweite Aussparung wird ein „number“-Block der Zahl 1 eingesetzt (Schritt 3). In die linke Aussparung wird ein „floor“-Block gelegt (Schritt 4). Dessen Aussparung wird mit einem „/“-Block gefüllt, der wiederum selbst zwei Aussparungen besitzt (Schritt 5). Ein „number“-Block der Zahl 5 wird in die rechte Aussparung (Schritt 6) und der „TBNumPersons.Text“-Block in die linke Aussparung (Schritt 7) gelegt.

Abbildung 17: Eventhandler für Klickevent von „BCalculate“ programmieren - Schritt 4



Nun ist der Pizzarechner fertiggestellt. Wie die fertige App simuliert werden kann, ist in Abschnitt 5 zu finden.

Obwohl das Überprüfen der Gültigkeit von Benutzereingaben oft einen großen Teil des eigentlichen Programms ausmacht, gehört dies zum guten Programmierstil. Es gewährleistet, dass eine App durch fehlerhafte oder manipulative Benutzereingaben nicht abstürzt oder gar Schaden auf dem Smartphone anrichtet.

4 Tutorial: Zeichnen auf Canvas

Dieses Tutorial soll einen kurzen Ausblick geben, wie mit weiteren Steuerelementen (insbesondere mit dem Canvas-Steuerelement) gearbeitet wird, indem auf ein Canvas (engl. Leinwand) gezeichnet wird. Das erste Tutorial wird als Grundlage betrachtet. Ziel wird es sein, einen kleinen Ball durch Kippen des Smartphones zu bewegen und die Spur des Balls aufzuzeichnen.

a) Steuerelemente platzieren

Zuerst wird der Titel der App auf „Canvas“ geändert und die Eigenschaft „ScreenOrientation“ auf „Portrait“ festgelegt (Eigenschaften der Komponente „Screen1“). Letztere Anpassung verhindert, dass sich die App dreht. Weiterhin werden vier Buttons nebeneinander in einem HorizontalArrangement platziert, von links nach rechts mit den Aufschriften „Rot“, „Grün“, „Blau“ und „Reset“ beschriftet und passend benannt (z.B. „BRed“ usw.). Die ersten drei Buttons werden zum Wählen einer Zeichenfarbe dienen, der Letzte zum Löschen des Gezeichneten. Allen Buttons sollen bezeichnende Namen verliehen werden. Durch Anpassen der Eigenschaft „TextColor“ kann die Farbe der Aufschriften geändert werden (siehe Abbildung 18).

Im folgenden Schritt werden je eine Checkbox (Name: „CBMove“, Aufschrift: „Ball“, Breite: 80 Pixel, Checked: ja), ein Label (Text: „Geschwindigkeit:“, Breite: 120 Pixel) und ein Slider (Name: „SSpeed“, Breite: 80 Pixel, MinValue: 0, MaxValue: 30, ThumbPosition: 10) in einem weiteren HorizontalArrangement platziert und mit den genannten Eigenschaften versehen (siehe Abbildung 19). Die Checkbox soll später dazu dienen, den Ball anzuzeigen bzw. zu verbergen, über den Slider soll die Geschwindigkeit des Balls eingestellt werden können. Die Eigenschaften Min- bzw. MaxValue des Sliders legen dabei einen Grenzbereich für den Wert (ThumbPosition) des Sliders fest, den dieser minimal bzw. maximal annehmen kann. Der Bereich unter den HorizontalArrangements wird mit einem Canvas-Steuerelement gefüllt (PaintColor: rot, Breite: 300 Pixel, Höhe: 300 Pixel).

Abbildung 18: Buttons platzieren und anpassen

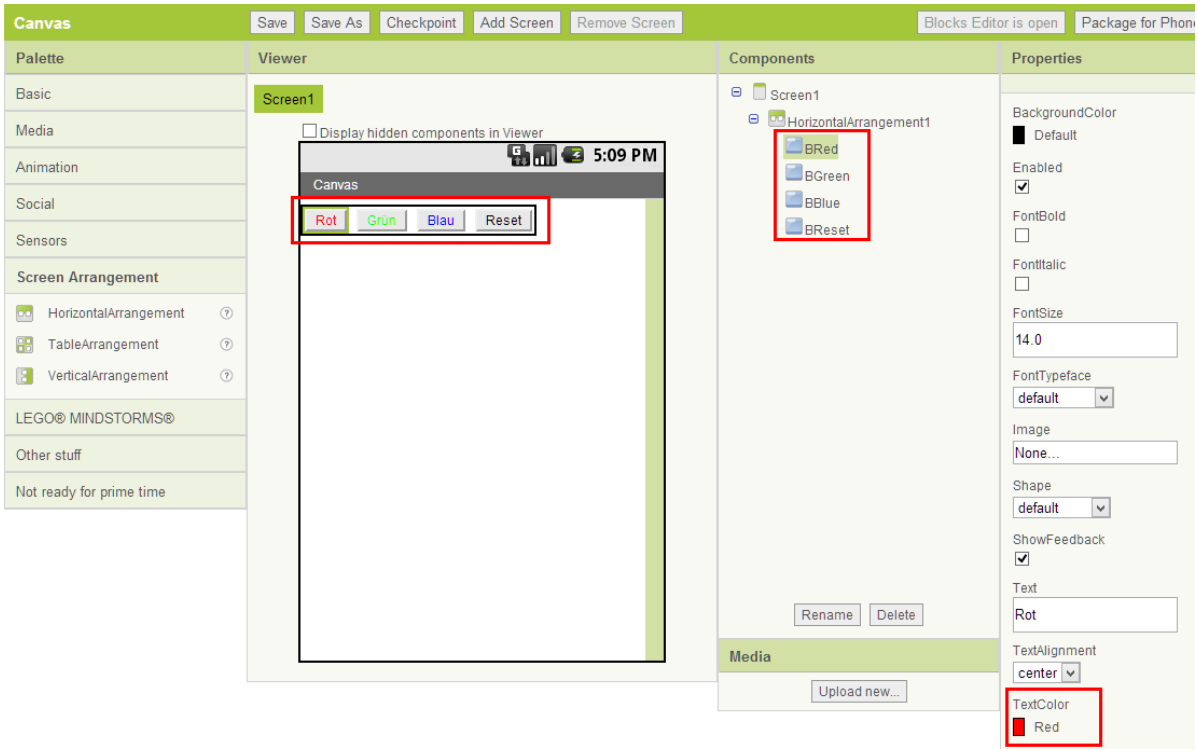
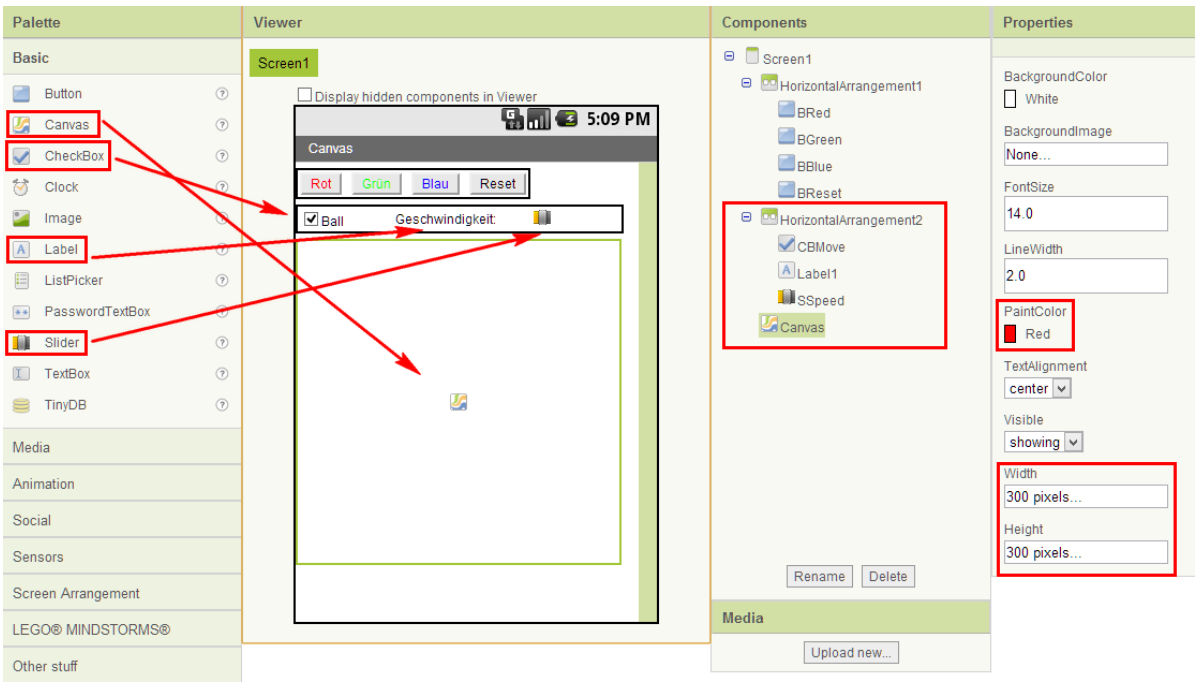
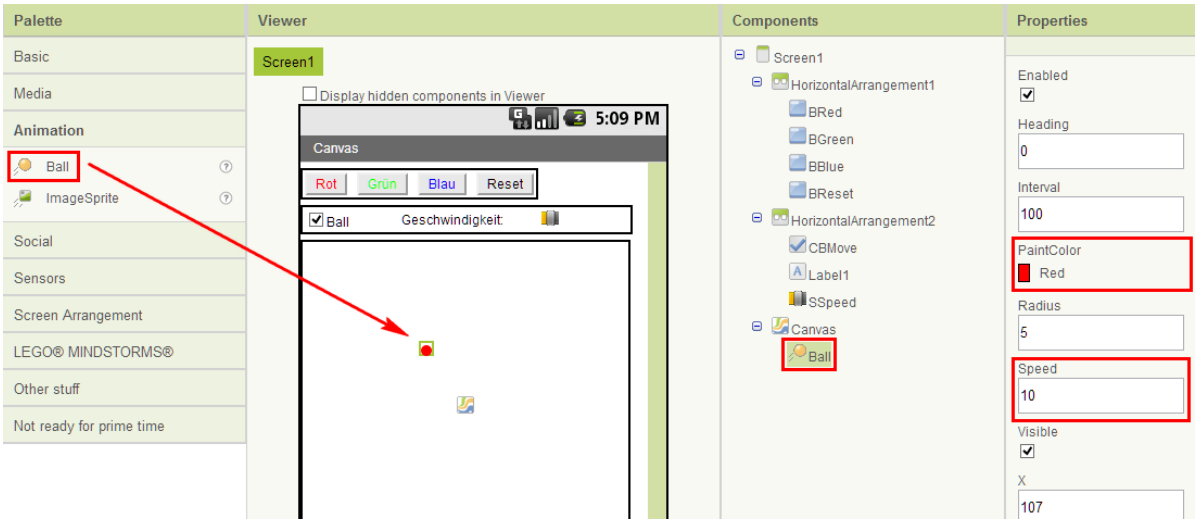


Abbildung 19: weitere Steuerelemente platzieren und anpassen



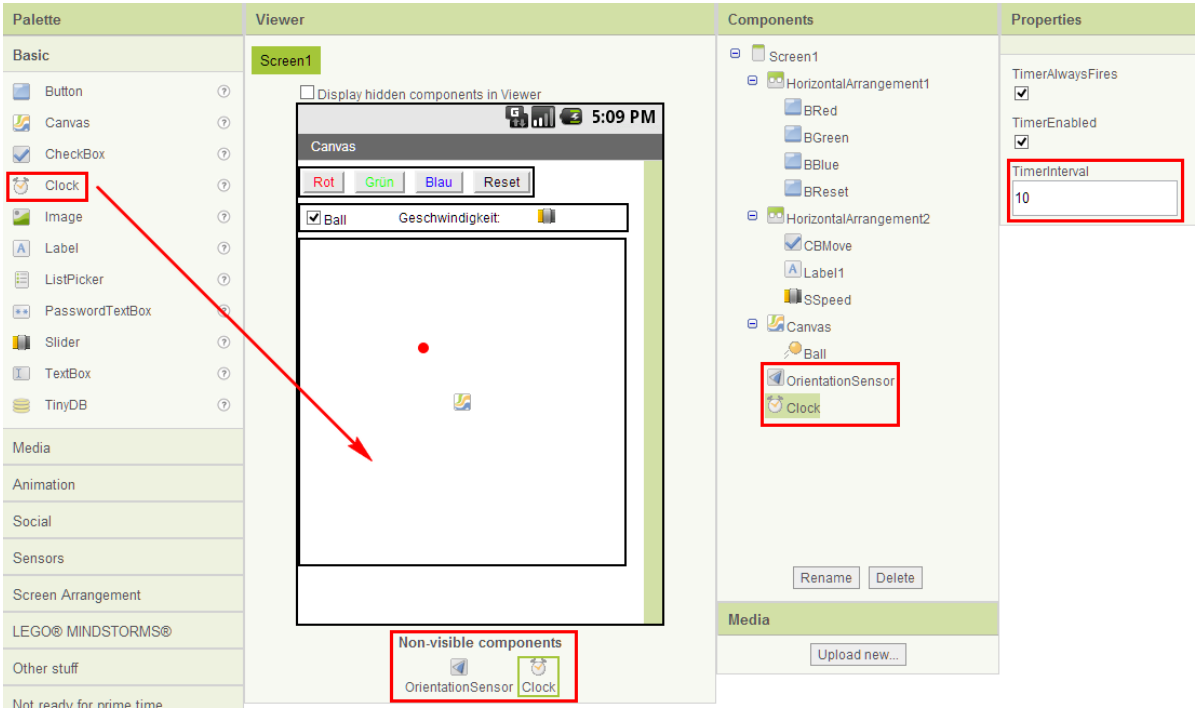
Der Ball, der später durch Drehen des Smartphones gesteuert wird, befindet sich in der Palette, Unterkategorie „Animation“. Er wird an einer beliebigen Stelle auf dem Canvas-Steuerelement platziert. Im Wesentlichen entspricht das Ball-Steuerelement dem ImageSprite-Steuerelement mit der Besonderheit, dass das Bild (Sprite) des letzteren individuell festgelegt werden kann, während das Sprite des Ball-Steuerelements ein Kreis ist, dessen Farbe angepasst werden kann. Die Farbe (PaintColor) wird hier ebenfalls auf die rot festgelegt. Die Eigenschaft „Speed“ beschreibt, um wie viele Pixel der Ball pro Bewegungsschritt seine Position ändert. Diese wird auf den Wert 10 gesetzt (siehe Abbildung 20).

Abbildung 20: Ball platzieren und anpassen



Auch diese App benötigt zwei unsichtbare Steuerelemente: Ein OrientationSensor (Palette, Subkategorie „Sensors“) wird zum Auslesen der Neigung/Drehung des Smartphones im Raum benötigt, ein Clock-Steuerelement dient als Timer und ermöglicht das periodische Ausführen eines bestimmten Funktionsblocks. Das Timerintervall legt dabei fest, nach welchen Zeitabständen (in Millisekunden) der Timer „feuert“, d.h. der Funktionsblock ausgeführt wird. Es wird auf 10 Millisekunden gesetzt (siehe Abbildung 21).

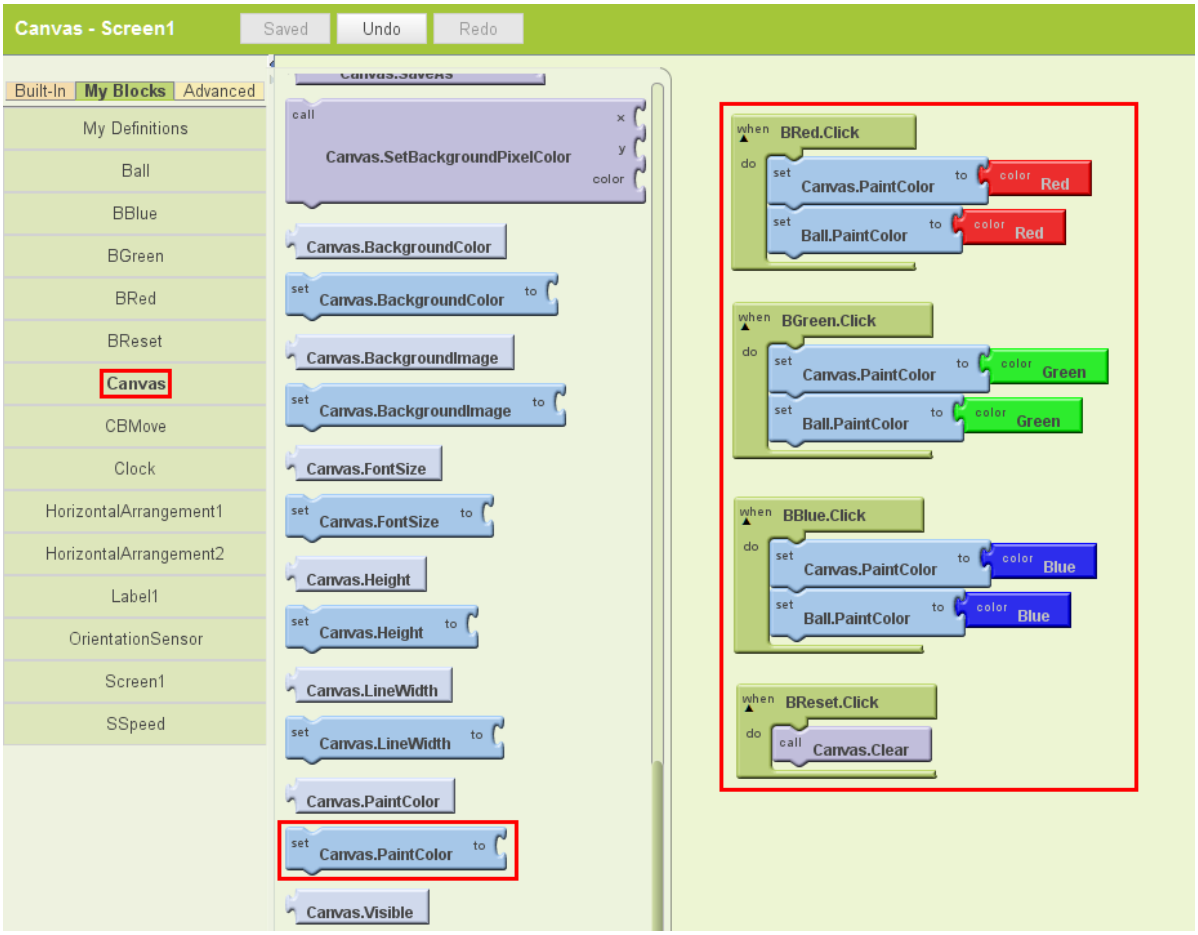
Abbildung 21: Clock und OrientationSensor einfügen



b) Verhalten programmieren

Nachdem alle benötigten Steuerelement platziert sind, folgt die Programmierung im Blocks Editor: Um Klicks auf die vier Buttons behandeln zu können, wird zu jedem Button ein Eventhandler hinzugefügt (My Blocks → [Buttonname] → [Buttonname].Click). Die Zeichenfarbe des Canvas- und des Ball-Steuerelements wird jeweils über die Eigenschaft „PaintColor“ festgelegt. Dementsprechend werden die Blöcke „set Canvas.PaintColor“ und „set Ball.PaintColor“ zum Setzen einer Farbe (Built-In → Colors) verwendet. Der Block „Canvas.Clear“ löscht das Gezeichnete (siehe Abbildung 22).

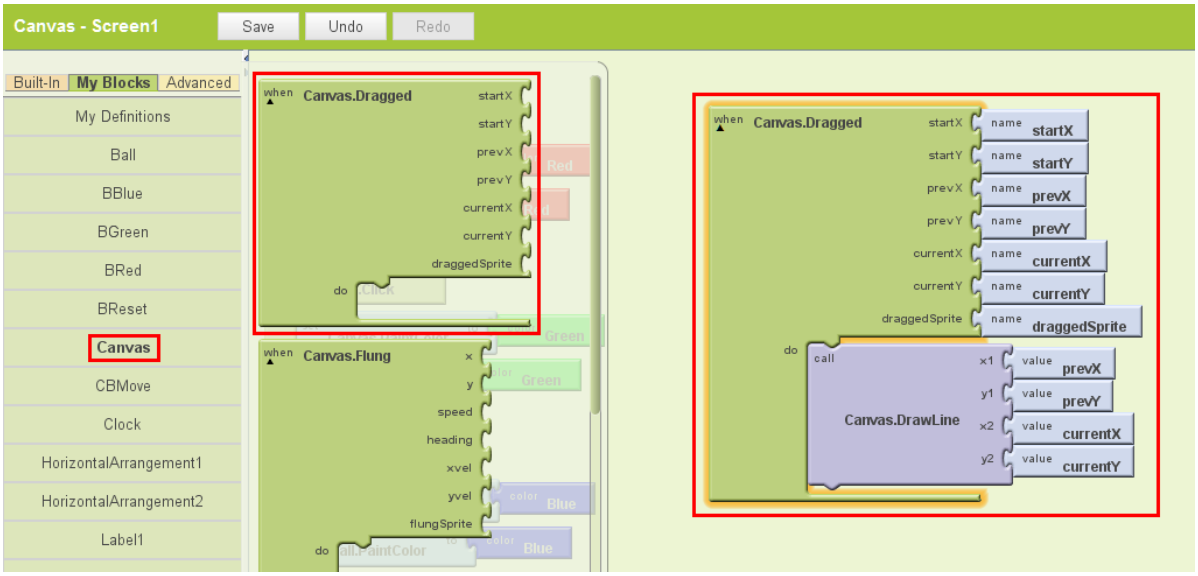
Abbildung 22: Eventhandler für Buttons zur Farbauswahl programmieren



Mit dem Finger auf das Canvas-Steuerelement zu zeichnen, ist nicht weiter schwer: Ein „Canvas.Dragged“-Event wird aufgerufen, wenn der Finger über das Steuerelement gezogen (engl. to drag) wird. Das Event stellt im Gegensatz zu den bisher genutzten Events eine Reihe von Parametern zur Verfügung: „prevX“ und „prevY“ sind die Koordinaten des Punktes an dem der Finger auf den Touchscreen aufgesetzt wurde, „currentX“ und „currentY“ sind die Koordinaten des Punktes, an dem sich der Finger aktuell befindet. Es reicht nun, beide Punkte mit einer Linie zu verbinden („Canvas.DrawLine“). Da das Dragged-Event sehr häufig aufgerufen wird, können flüssige Freihandzeichnungen ohne Weiteres erfolgen (siehe Abbildung 23).

Wird der Zustand der Checkbox CBMove geändert („checked“ zu „unchecked“ bzw. umgekehrt), wird das Event „CBMove.Changed“ aufgerufen. Hier soll der Ball angezeigt werden, wenn die Checkbox angekreuzt (checked) ist, ansonsten soll er verborgen werden. Ebenfalls wird der Timer aktiviert bzw. deaktiviert, um unnötiges Auswerten des Orien-

Abbildung 23: Canvas.Dragged-Event zum Zeichnen auf Touchscreen



tierungssensors und unnötiges Berechnen der Richtung und Geschwindigkeit des Balls zu vermeiden. Hierzu werden die Eigenschaften „Ball.Visible“ und „Clock.TimerEnabled“ auf den Zustand der Checkbox („CBMove.Checked“) gesetzt.

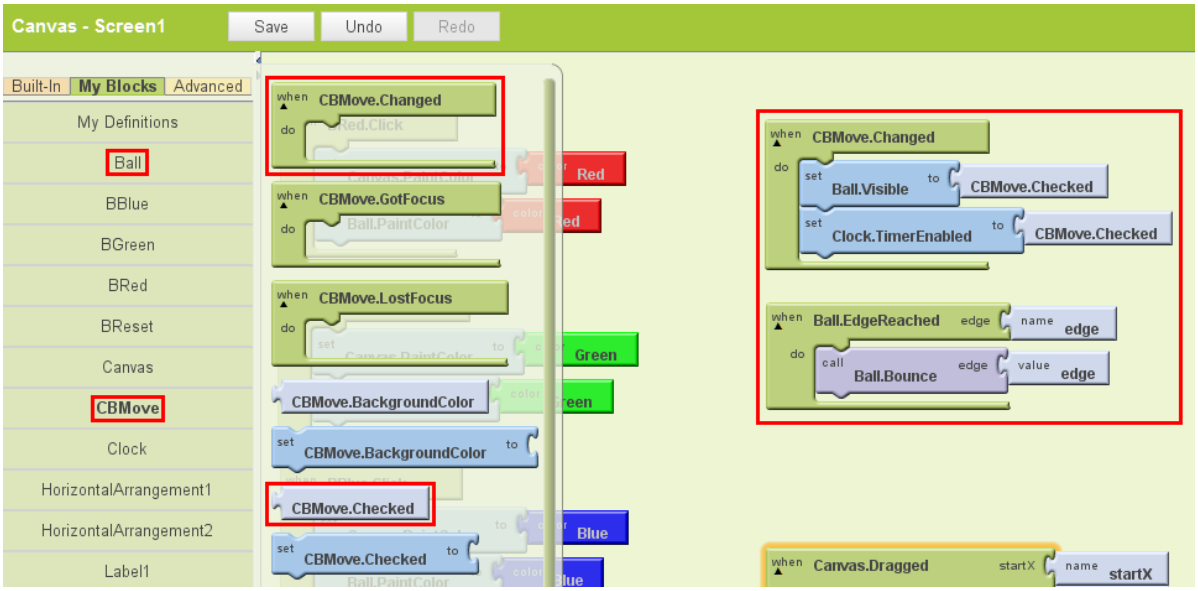
Ferner soll der Ball an den Kanten des Canvas-Steuerelements abprallen. Dies geschieht, indem im „Ball.EdgeReached“-Event „Ball.Bounce“ aufgerufen und der Funktion die entsprechende Ecke in Form des „edge“-Parameters des Events übergeben wird (siehe Abbildung 24).

Zuletzt muss das „Clock.Timer“-Event ausgewertet werden. Es wird periodisch alle 10 Millisekunden (wie über die Eigenschaft Clock.TimerInterval festgelegt wurde) aufgerufen und soll dazu dienen, die aktuelle Geschwindigkeit und Richtung des Balls festzulegen und dessen Spur zu zeichnen.

„Ball.Heading“ bezeichnet die Richtung, in die sich der Ball bewegt. Sie wird auf die Richtung gesetzt, in die das Smartphone geneigt ist. Diese Richtung wird mittels „OrientationSensor.Angle“ ermittelt. Die Geschwindigkeit des Balls „Ball.Speed“ berechnet sich aus „OrientationSensor.Magnitude“ (wie sehr wurde das Smartphone geneigt) und „SSpeed.ThumbPosition“ (dieser Wert stellt einen über den entsprechenden Slider regelbaren Empfindlichkeitsfaktor dar). Konkret lautet die Formel:

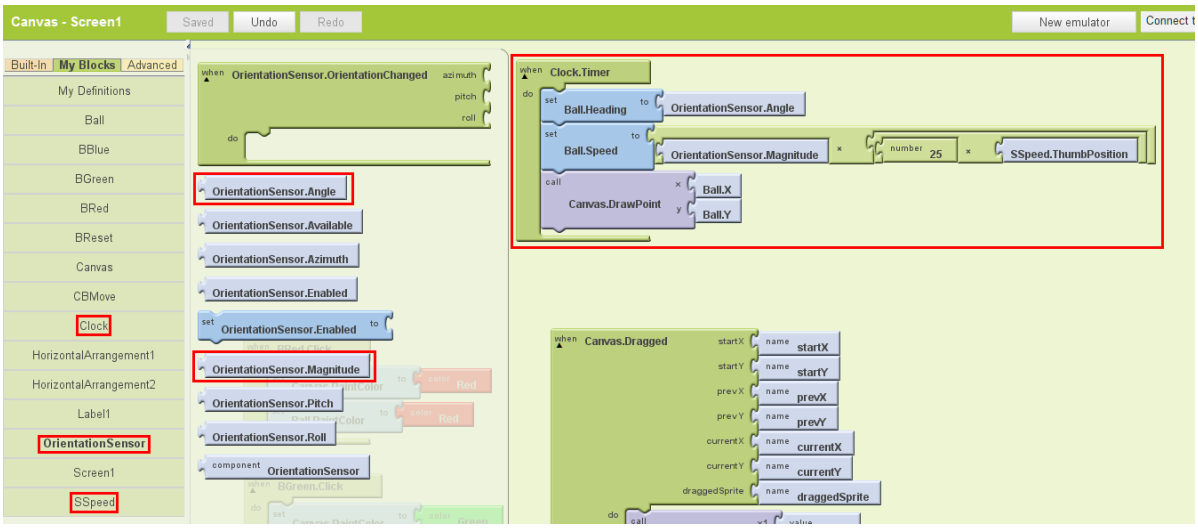
$$Ball.Speed = OrientationSensor.Magnitude \cdot 25 \cdot SSpeed.ThumbPosition$$

Abbildung 24: Geschwindigkeit und Bewegung des Balls regeln



Die Spur des Balls wird gezeichnet, indem jedes Mal, wenn das Timerevent aufgerufen wird, ein Punkt an die Stelle gesetzt wird, an der sich der Ball momentan befindet. Dies erledigt der Aufruf von „Canvas.DrawPoint“ mit den Koordinaten („Ball.X“ und „Ball.Y“) der aktuellen Position des Balls (siehe Abbildung 25).

Abbildung 25: periodisch Orientierungssensordaten auswerten

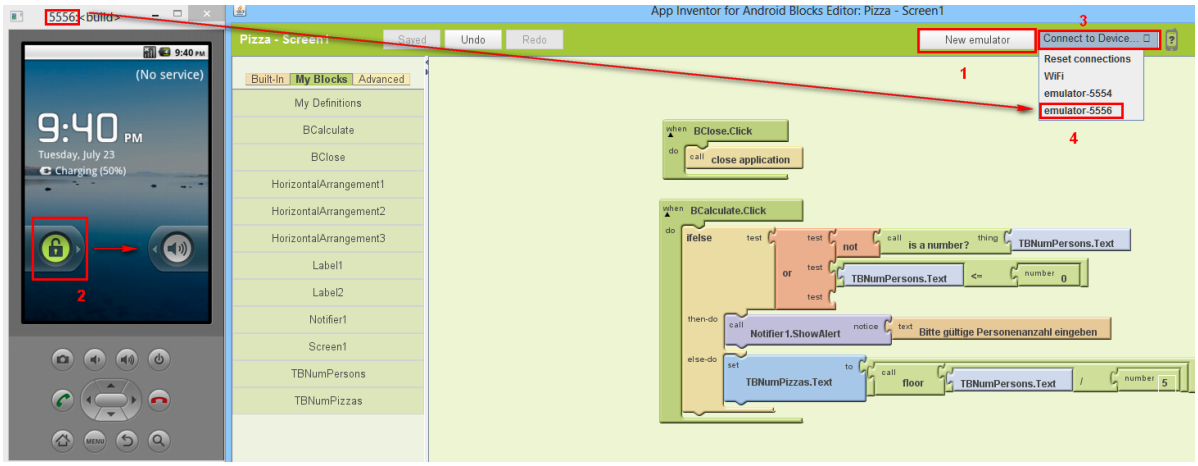


Mit diesem Schritt ist die App fertiggestellt. Sie eignet sich als kleines Spiel: Es können Labyrinthe gezeichnet werden, durch die der Ball gelenkt werden muss ohne die Wände zu berühren.

5 Simulieren von Apps

Durch Klick auf die Taste „New Emulator“ im Blocks Editor (Schritt 1 Abbildung 26) kann eine programmierte App in einem virtuellen Smartphone, das das Betriebssystem Android emuliert, simuliert werden. Zuvor muss allerdings der Emulator „aiStarter“[2] heruntergeladen und installiert werden. Nachdem der erwähnte Button angeklickt wurde, öffnet sich ein neues Fenster, in dem nach kurzer Zeit ein virtuelles gesperrtes Smartphone erscheint. Um das virtuelle Smartphone zu entsperren, wird das Schlosssymbol mit der Maus nach rechts gezogen (Schritt 2). Neben der Schaltfläche „New Emulator“ befindet sich der Button „Connect to Device“ Dieser wird angeklickt (Schritt 3). Im erscheinenden Menü wird der Eintrag „emulator-xxxx“ ausgewählt, bei dem „xxxx“ der Ziffernfolge entspricht, die sich in der Titelzeile des Fensters befindet, in dem das virtuelle Smartphone dargestellt wird (Schritt 4). Nun wird die programmierte App auf dem virtuellen Smartphone installiert, was einen Augenblick dauern kann. Im Appmenü des Smartphones sollte kurz darauf die programmierte App zu finden sein. Diese kann durch einen Mausklick gestartet werden.

Abbildung 26: Simulation der programmierten App starten



[2] http://appinventor.mit.edu/explore/ai2/setup-emulator.html

6 Ausblick

Die hier vorgestellte App nutzt die Möglichkeiten des MIT App Inventors lange nicht aus. Der App Inventor kann beinahe die gesamte Hardware eines Smartphones ansteuern. Das Ermitteln des aktuellen Standorts über den im Smartphone integrierten GPS-Empfänger, das Versenden und Auswerten von SMS und selbst das Steuern von LEGO® MINDSTORMS® ist ohne große Einarbeitung möglich. Tutorials hierzu finden sich im Internet unter http://appinventor.mit.edu/explore/tutorials.html. Ebenso ist eine Referenz vorhanden, die die Funktionsweise aller verfügbaren Blöcke und Steuerelemente erklärt.