Deutsches Jungforscher Netzwerk
juFORUM e.V.

IdeenExpo 15

Experimentierheft

IdeenExpo

Vom 4. bis 12.07.2015 in Hannover

Deutsches Jungforschernetzwerk – juFORUM e.V.

www.juforum.de

facebook.com/juforum

Stand: 3. Juli 2015

Inhaltsverzeichnis

1 Übersicht

Apps entwickeln - wie DU willst!

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

Zielgruppe: 10.-12. Klasse / ältere
Interaktiv: ja
Vorführdauer: über 5 Min.

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

Ein eigenes PC-Spiel selbst programmiert

Entwerfe dein eigenes 2D-Spiel mit der MIT-eigenen Programmiersprache Scratch!

Zielgruppe: 5.-9. / 10.-12. Klasse
Interaktiv: ja
Vorführdauer: über 5 Min.

Programmieren – das sind viele Zeilen mit kryptischen Zeichen und verzweifelte Informatiker bei der Fehlersuche? Die Baukastensprache Scratch beweist das kunterbunte Gegenteil. Mithilfe von Funktionsblöcken, die man einfach auf den Bildschirm zieht, entsteht ein Spiel, eine Animation oder eine Geschichte – die Grenze ist die eigene Fantasie.

Unimat-Workshop: Salzstreuer drechseln

Salzstreuer selber herstellen - beim Drechsel-Workshop mit kindersicherer Unimat-Drechselbank.

Zielgruppe: 5.-9. / 10. -12. Klasse
Interaktiv: ja
Vorführdauer: über 5 Min.

Mit Unimat-Maschinen wird Technik begreifbar. Selber etwas hervorzubringen ist entscheidend - Technik und ihre Wirkung wird aktiv vermittelt. Die technische Frühbildung steht im Mittelpunkt: Training der Motorik; freier Zugang zu Maschinen, eigenständiges Arbeiten, Vertiefung technischen Verständnisses, Fertigung von Funktionsmodellen... Mehr Infos auf http://www.thecooltool.com/

Selfmade: Feuerwehr Mini-Feuerlöscher

Einen Miniatur-Feuerlöscher bauen – zum Nachmachen geeignet!

Zielgruppe: 5.-9. / 10.-12.
Interaktiv: nein
Vorführdauer: bis 5 Min.

Löschschaum ist multifunktional. Nicht umsonst setzt ihn die Feuerwehr seit langem erfolgreich zur Brandbekämpfung ein. Er dämmt, kühlt, verhindert das Ausgasen brennbarer Flüssigkeit und trennt das Feuer von der Luft und damit vom nährenden Sauerstoff. Magischer Stoff? Keineswegs! Wir zeigen euch, wie ihr euch euren eigenen Feuerlöscher bastelt.

Mobile Heizung: Latentwärmespeicher

Ein Heiz-Pad selbst bauen und ausprobieren!

Zielgruppe: 5.-9. Klasse
Interaktiv: ja
Vorführdauer: bis 5 Min.

In einer eiskalten Hofpause gibt es nichts Besseres, als ein wohlig wärmendes Pad in der Jackentasche. Hinter diesem nützlichen Gadget steckt dabei eine spannende Chemie: Sogenannte Latentwärmespeicher speichern Wärme mithilfe von Phasenübergängen. Natriumacetat lagert in einer exothermen Reaktion Wasser ein und wird zu Natriumactetat-Trihydrat.

Moderne Küchentricks: DNA aus der Erdbeere

Die Erbinformation einer Erdbeere extrahieren - mit einfachen Haushaltsmitteln realisiert.

Zielgruppe: 10.-12. / ältere
Interaktiv: ja
Vorführdauer: bis 5 Min.

Die Aufklärung der DNA-Struktur ist eine der wichtigsten Entdeckungen der Biotechnologie. An die Erbinformation einer Zelle heranzukommen scheint mühsam: Sie liegt tief im Zellkern verborgen. Mit ein paar Kniffen jedoch geht es ganz fix! In wenigen Schritten lässt sich unser Versuchsobjekt in kleinste Bestandteile trennen und die DNA isolieren.

Aus dem Garten: Kartoffelbatterie

Eine LED zum Leuchten bringen – mit einer handelsüblichen Kartoffel.

Zielgruppe: 5.-9. Klasse
Interaktiv: nein
Vorführdauer: bis 5 Min.

Ohne die Erkenntnisse der Elektrochemie könnten wir heute weder stundenlang mit dem Smartphone telefonieren noch mit der Taschenlampe nach dem Lichtschalter suchen, denn Batterien und Akkus gehören zu unserem Alltag. Aber wusstest du, dass eine Kartoffel auch eine Batterie ist? Wir zeigen es dir mithilfe von ein paar Metallplättchen und Kabeln!

Amesraum

Der Amesraum ist eine echt schräge optische Täuschung - neue Perspektive garantiert!

Zielgruppe:	5.-9. Klasse
Interaktiv:	ja
Vorführdauer:	bis 30 Min.

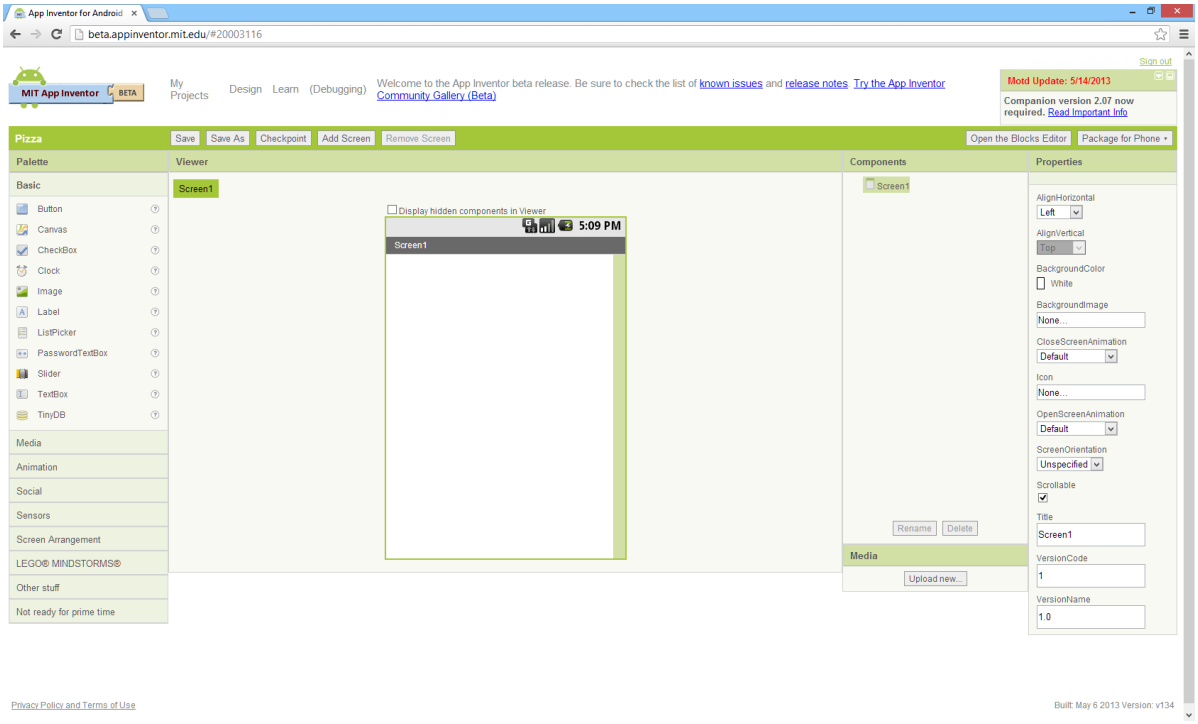
Ein Amesraum ist in der Tat meist ein Raum im herkömmlichen Sinne, also zum Beispiel ein Zimmer, wie wir es aus dem Alltag kennen. Er ist jedoch durch und durch schräg: Eine Seitenwand ist um einiges länger als die andere und auch die Decke ist nicht gerade. Doch stellt man sich an einen ganz bestimmten Punkt in diesem Raum, so erscheint plötzlich alles wieder wunderbar rechtwinklig. Wenn Ihr also wie im Film „Der Hobbit“(2012) Zauberer und Zwerge ins richtige Verhältnis setzen und euren eigenen Miniatur-Amesraum bauen wollt, ran an die Bastelscheren!

2 MIT App Inventor – Programmierung von Apps für Android

2.1 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.

Abbildung 1: leeres Projekt

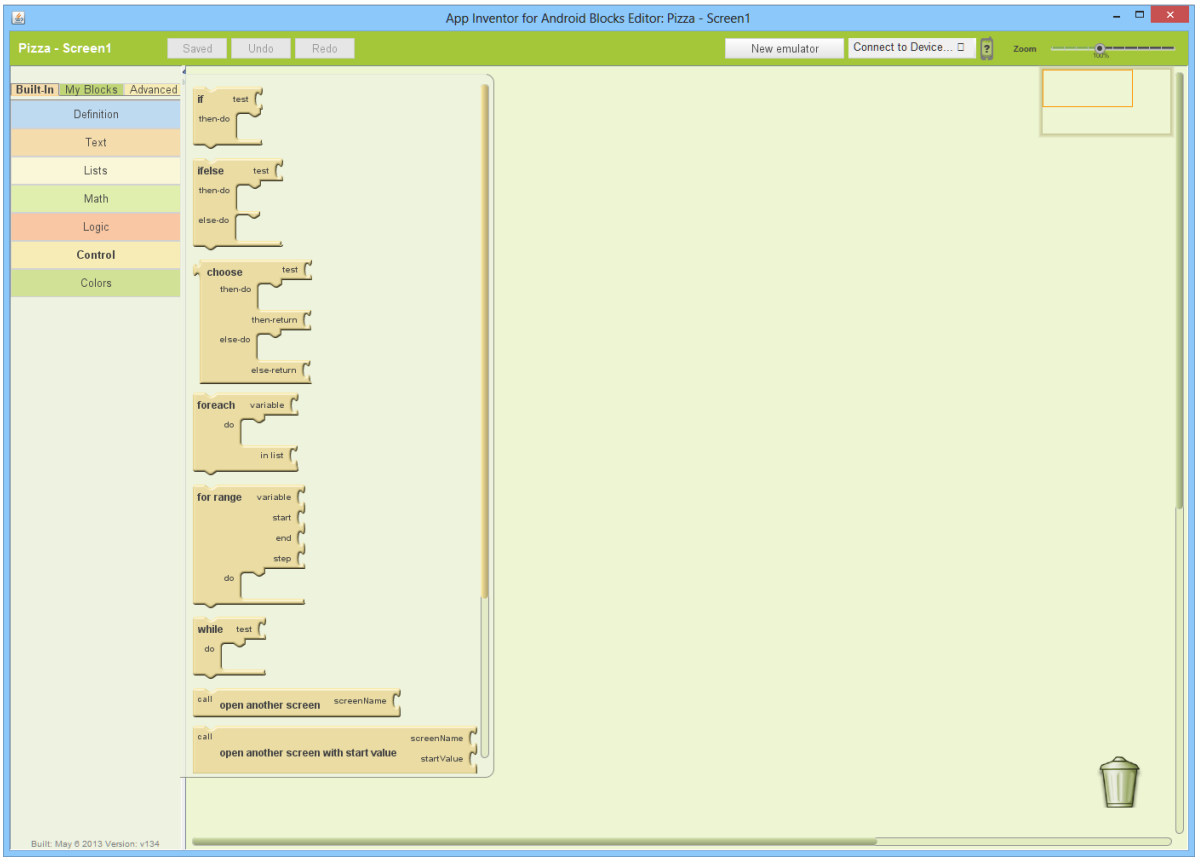


Das Fenster (siehe Abbildung 1) ist in vier Teile geteilt: In der Mitte („Viewer“) befindet sich ein fiktives Smartphonedisplay. Dieses kann mit Steuerelementen wie beispielsweise Tasten („Button“), Text („Label“), Eingabefeldern („TextBox“) oder Elementen, auf die gezeichnet werden kann („Canvas“), 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 Funk-

tion 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 Unterabschnitt 2.4). 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.

2.2 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.

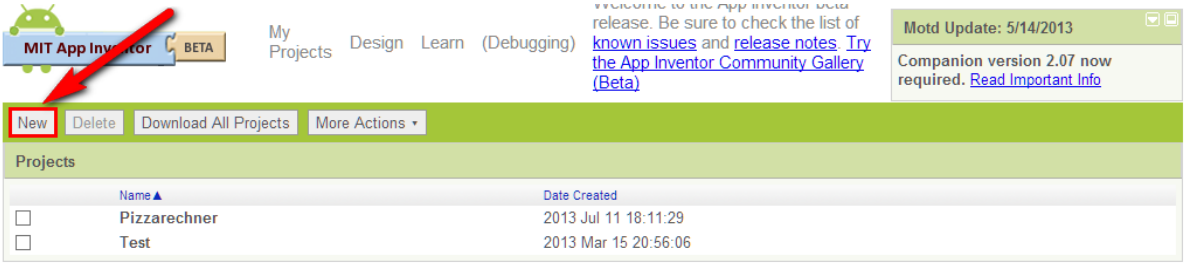
Abbildung 3: Pizzarechner

2.2.1 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).

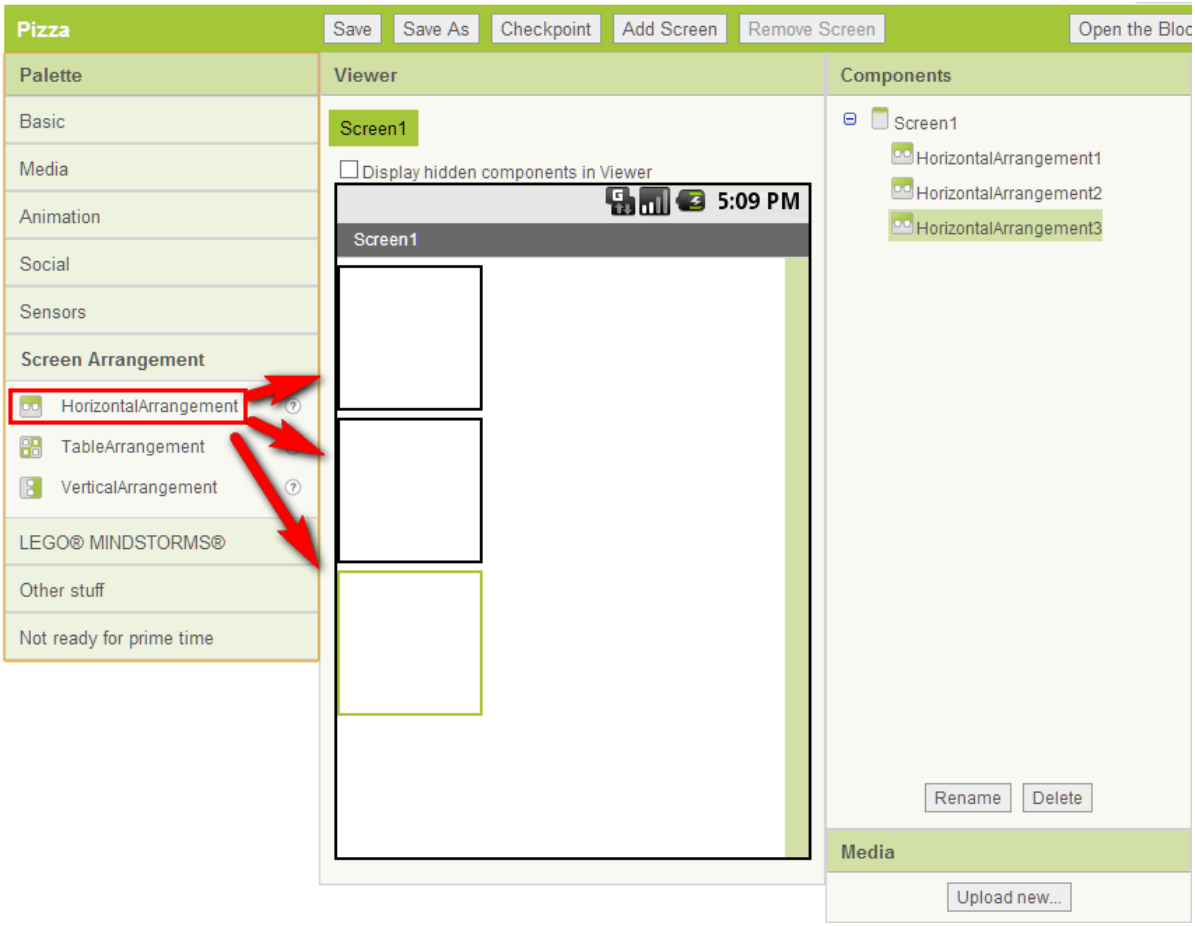
Nun können die Steuerelemente zur Eingabe der Personenzahl, zur Ausgabe der Anzahl der benötigten Pizzen und zur Steuerung der App platziert werden. Um Steuerelemente nebeneinander platzieren zu können (z.B. eine Beschreibung neben einer Textbox), werden Contai-

Abbildung 4: Neues Projekt erstellen



nerelemente 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.

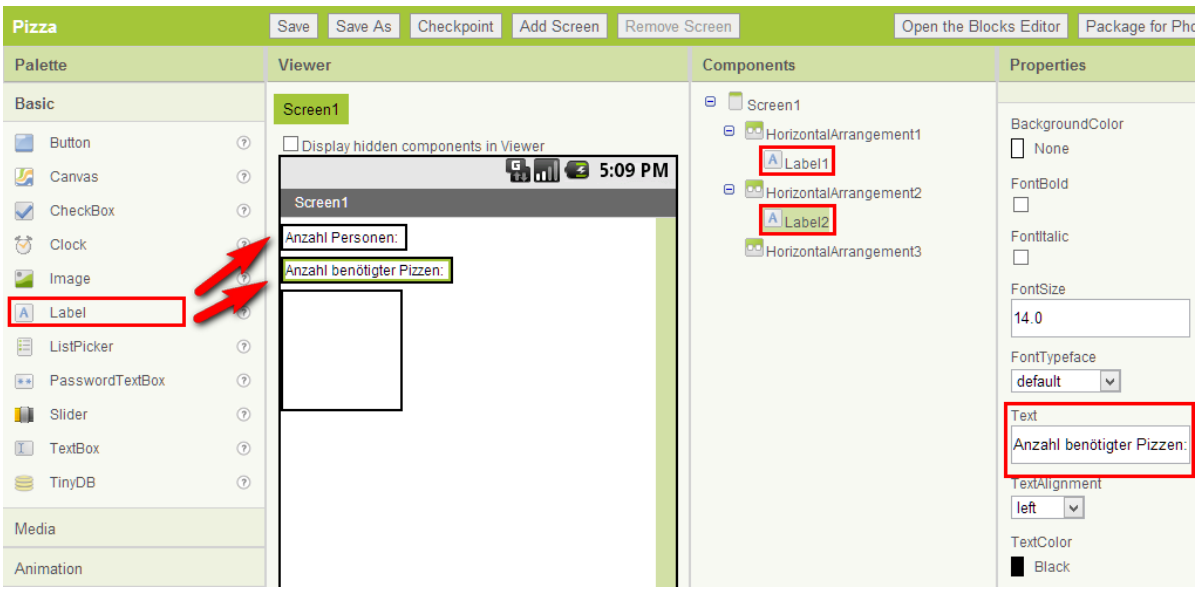
Abbildung 5: „HorizontalArrangement“ platzieren



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).

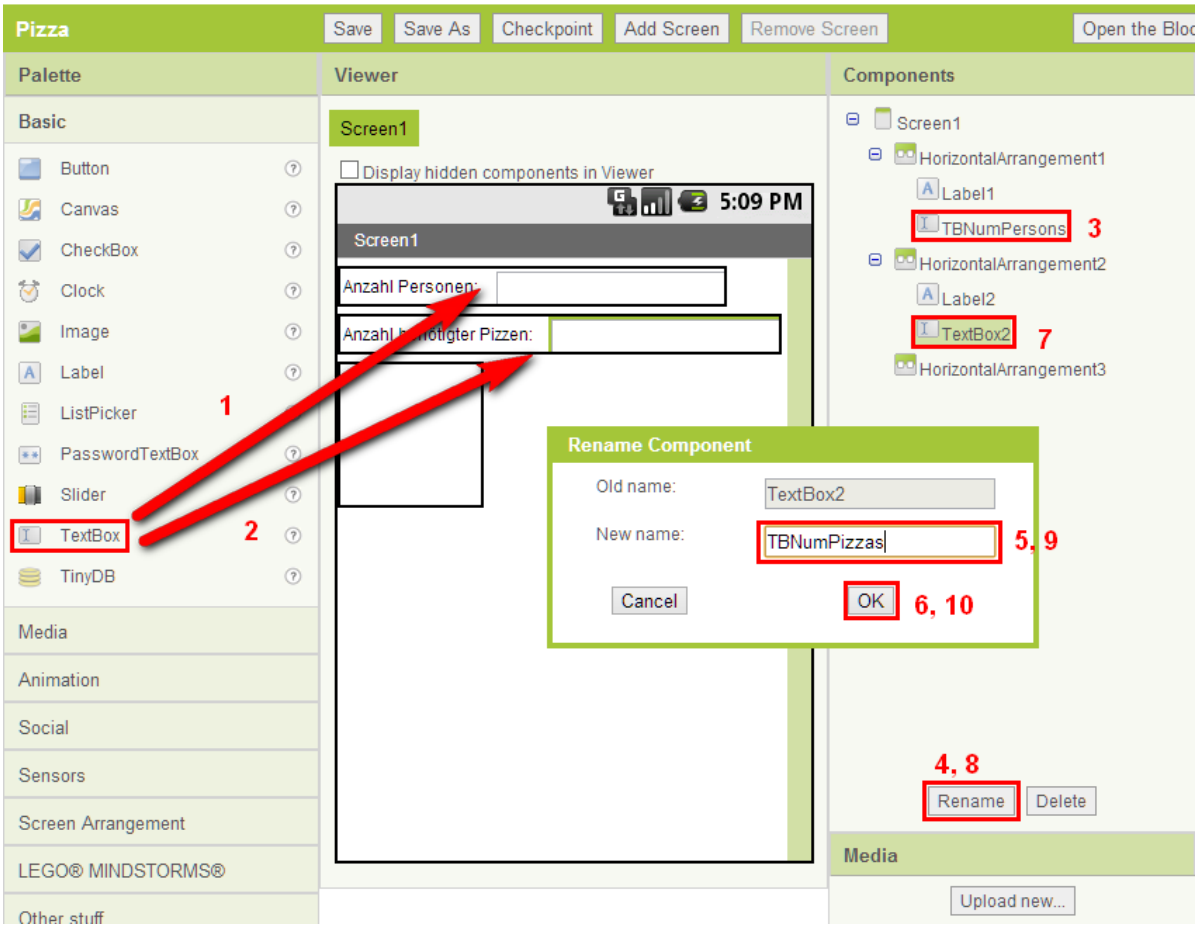
Abbildung 6: „Label“ platzieren und Text anpassen



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).

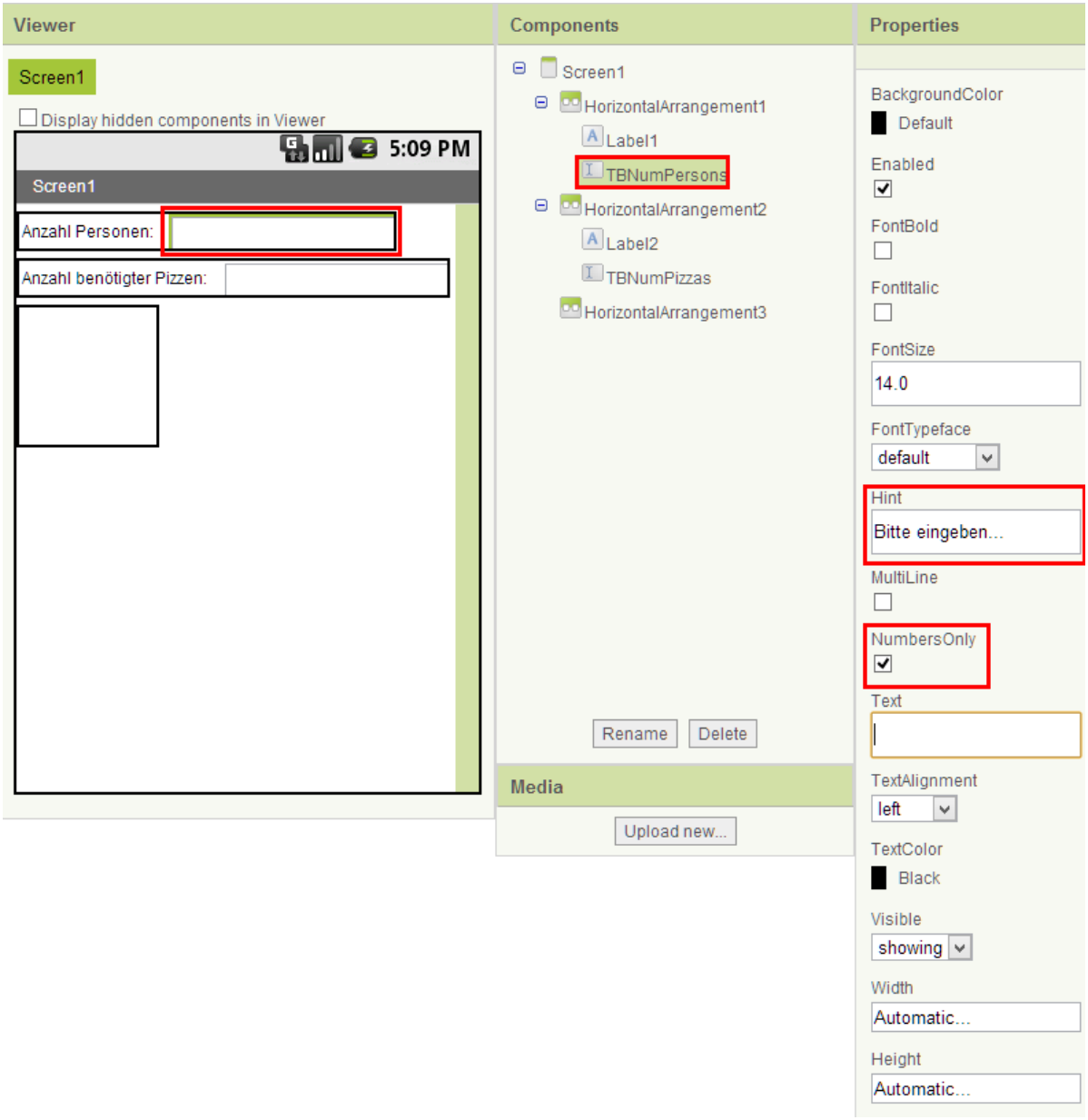
Bezüglich der Textboxen gilt es, ein paar weitere Eigenschaften anzupassen: Die Textbox „TBNumPersons“ soll der Eingabe der Personenzahl dienen. Demnach dürfen hier nur Zahlen eingegeben werden. Dies kann gewährleistet werden, indem die Eigenschaft „NumbersOnly“ bei den Properties gesetzt wird. Android 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 kleiner Infotext, der so lange in der Textbox angezeigt wird, bis eigener Text eingegeben wird

Abbildung 7: „TextBox“ platzieren und umbenennen



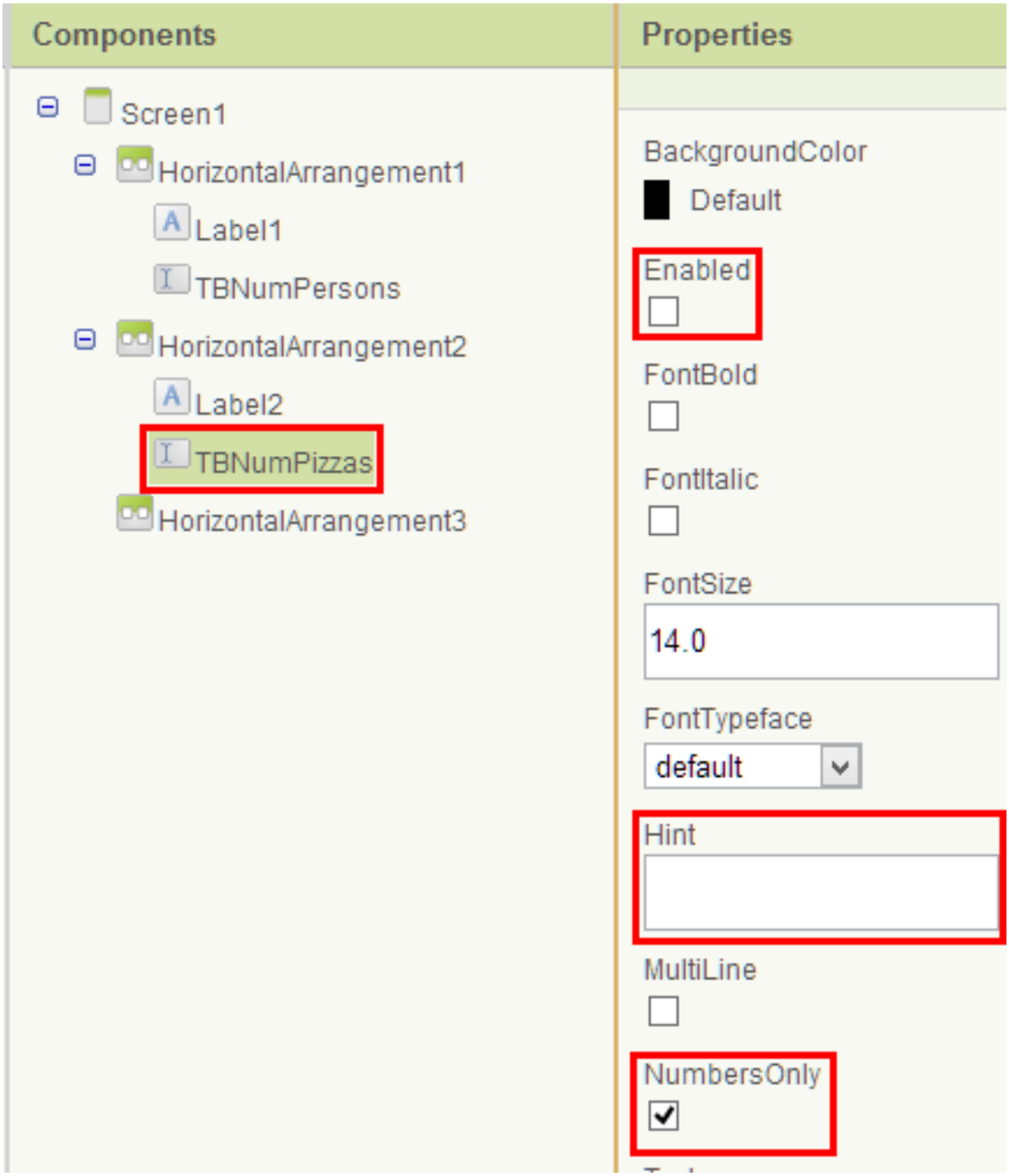
(siehe Abbildung 8). Zum Ändern von Eigenschaften muss das entsprechende Steuerelement angeklickt worden sein.

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.

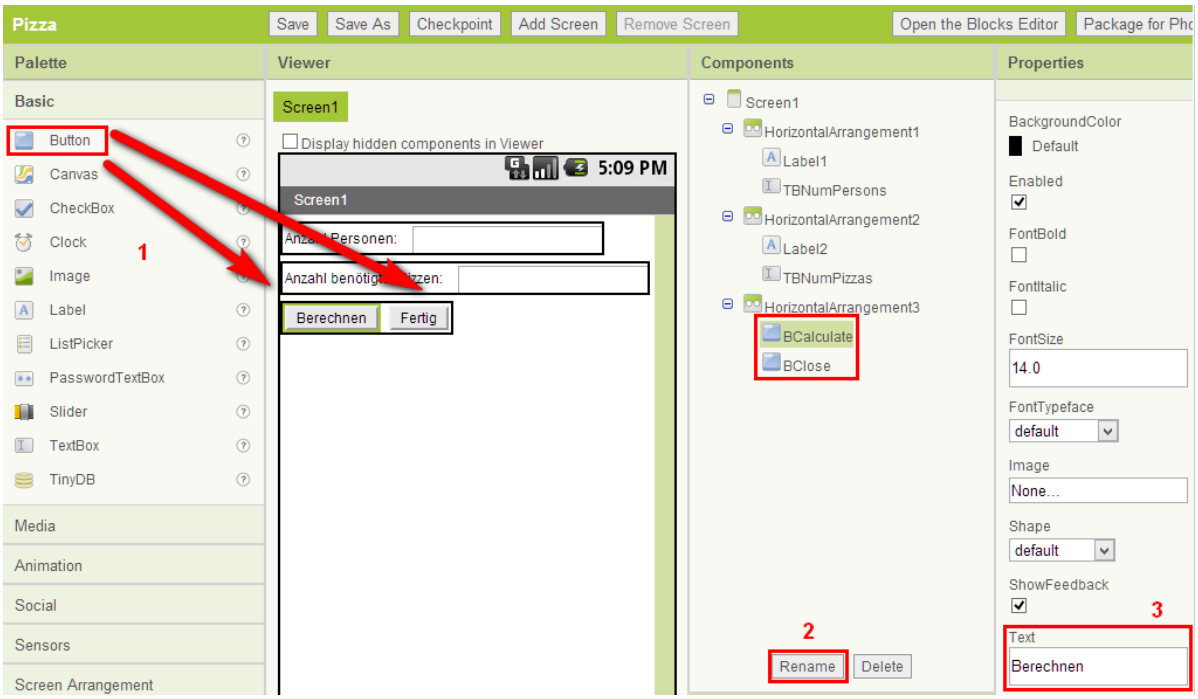
Abbildung 9: TextBox 2 („TBNumPizzas“) anpassen



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

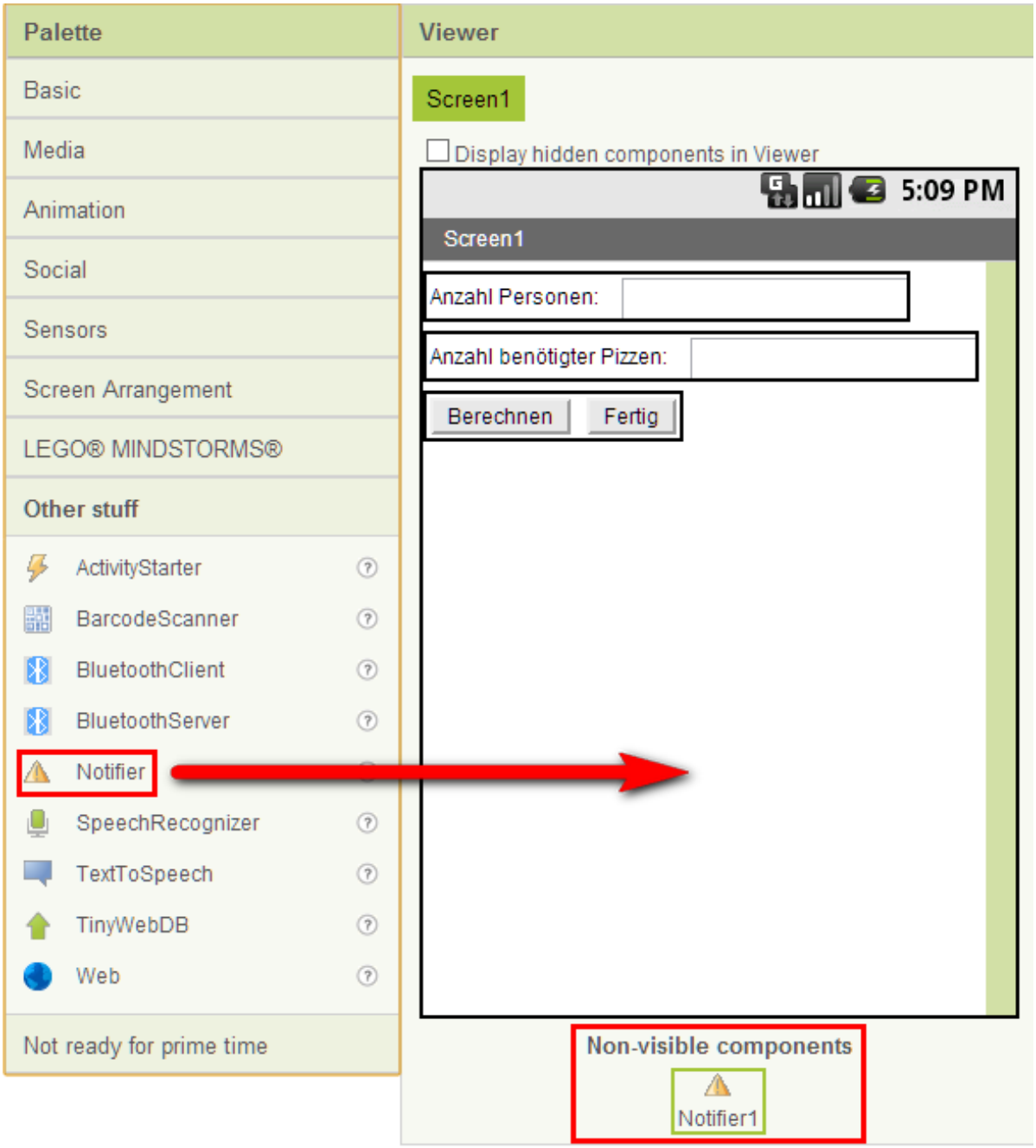
„Text“ geändert wird (Schritt 3).

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



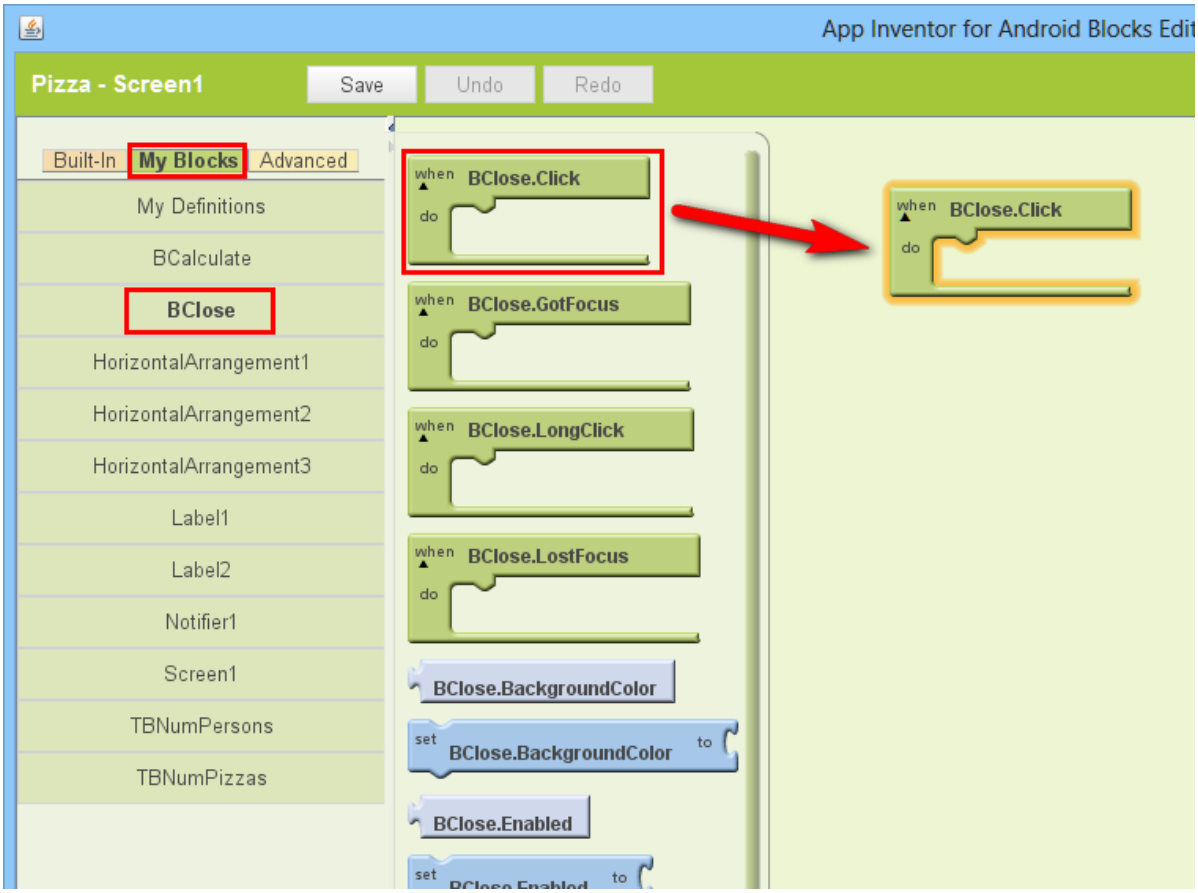
Zuletzt kann der Titel der App, der momentan „Screen1“ lautet, geändert werden, indem im Components-Bereich der entsprechende Eintrag ausgewählt und im Properties-Bereich die Eigenschaft „Title“ modifiziert wird.

2.2.2 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

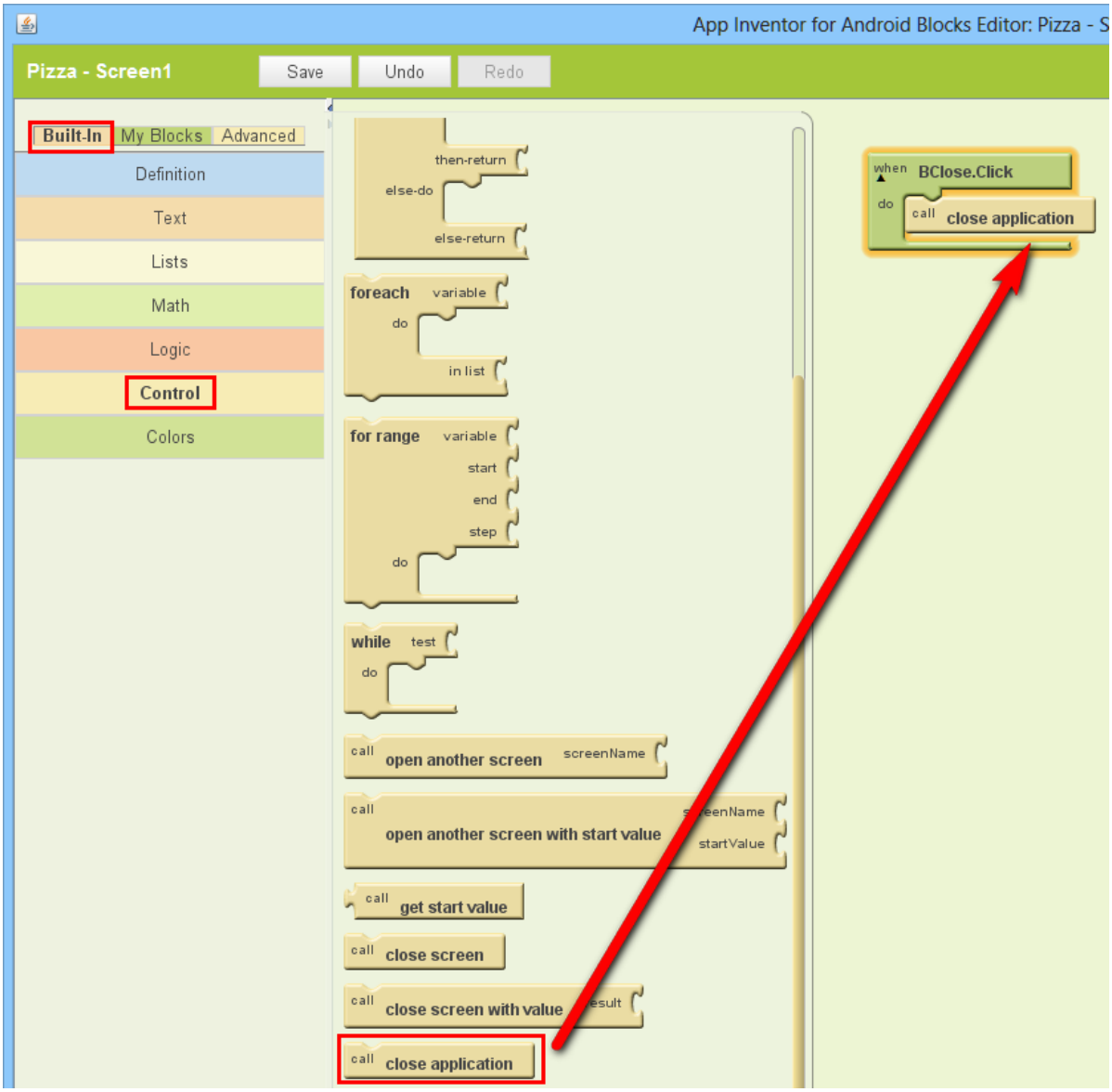


In der Kategorie „Built-In“, Subkategorie „Control“befindet sich ganz unten der Block „close application“. Während der zuerst eingefügte Block der Ablaufsteuerung des Programms dient (wie alle eine Klammer aufspannenden Blöcke), ist dieser Block ein Befehl. Er schließt die App. Der Block wird wird in 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 Bedingung

Abbildung 13: Eventhandler für Klickevent von „BClose“ programmieren

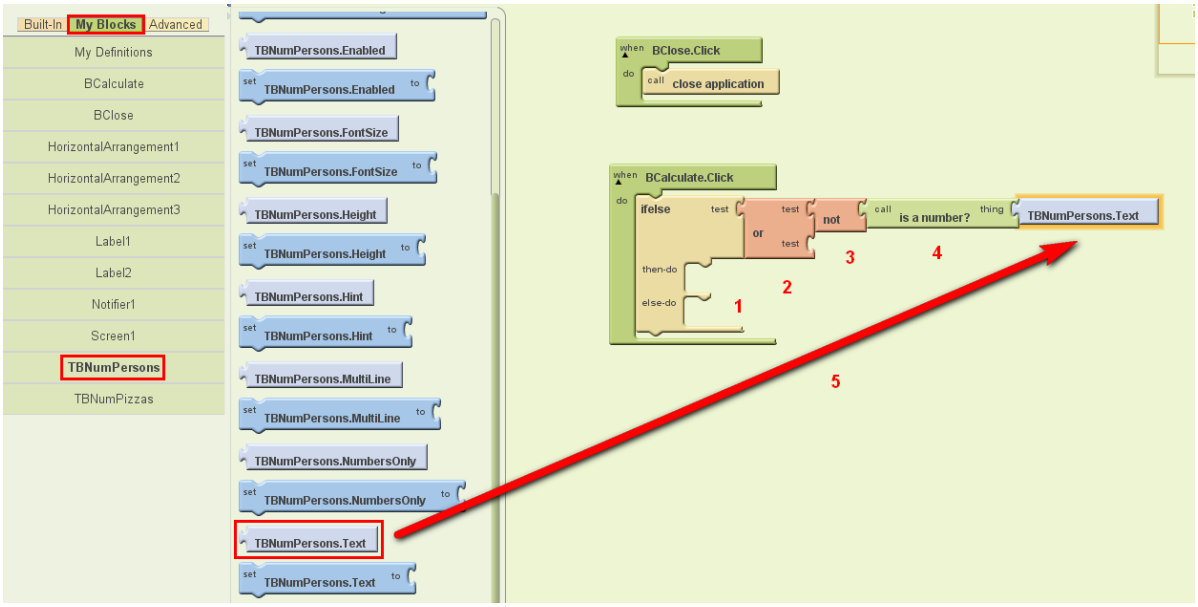


„test“ erfüllt ist, wird entweder die erste vom Block aufgespannte Klammer ausgeführt, sonst die Zweite. Wörtlich: Falls die Bedingung „test“ erfüllt ist, dann führe („then do“) Klammer 1 aus, sonst führe („else do“) Klammer 2 aus.

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“.

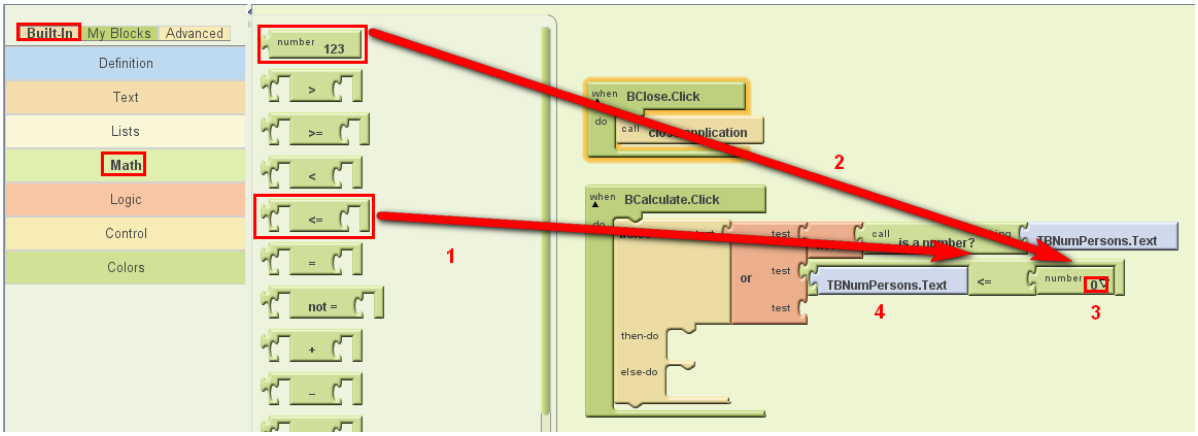
Abbildung 14: Eventhandler für Klickevent von „BCalculate“ programmieren - Schritt 1



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).

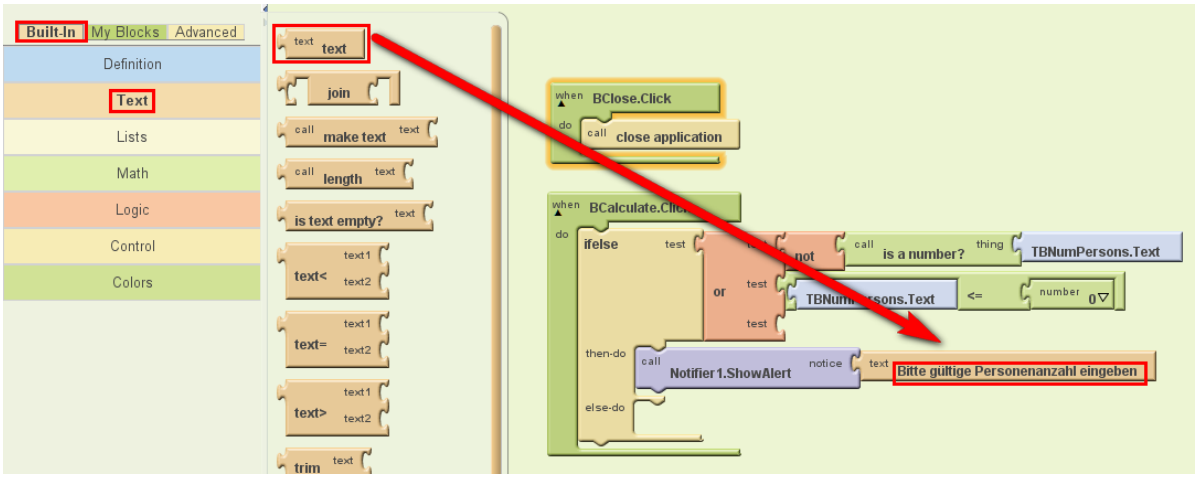
Ist mindestens eine der oben beschriebenen Teilbedingungen erfüllt, soll eine Meldung an den Benutzer ausgegeben werden. Dazu bietet sich die im letzten Schritt des Platzierens der Steuerelemente eingefügte „Notifier“-Komponente an. Unter My Blocks → Notifier1 findet sich der Block „Notifier1.ShowAlert“.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



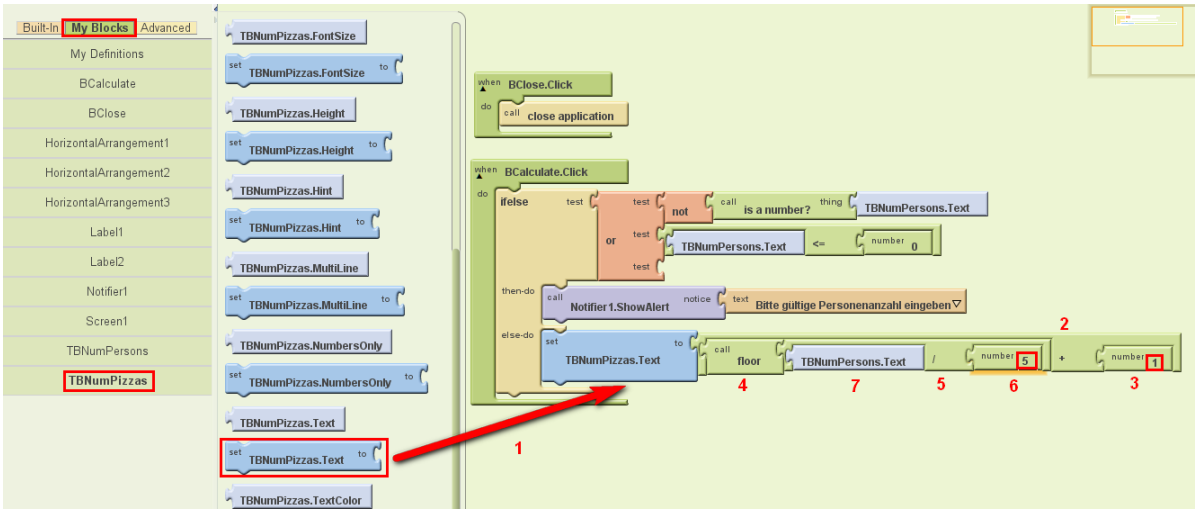
Ist keine der Teilbedingungen erfüllt, kann endlich die Anzahl der benötigten Pizzen berechnet und in TBNumPizzas ausgegeben werden. Zuvor wurde die Eigenschaft „Text“der Textbox TBNumPersons gelesen. Nun soll die gleichnamige Eigenschaft von TBNumPizzas geschrieben werden. Hierzu steht unter My Blocks → 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:

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

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 Unterabschnitt 2.4 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.

2.3 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.

2.3.1 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

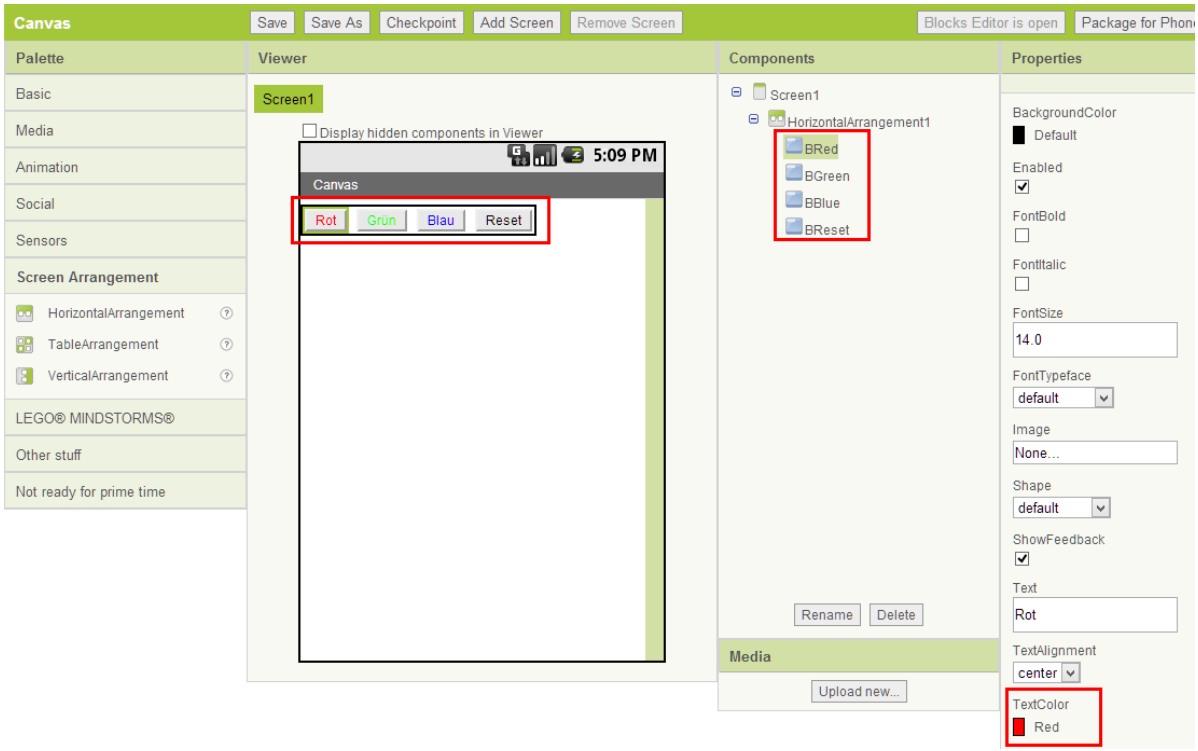
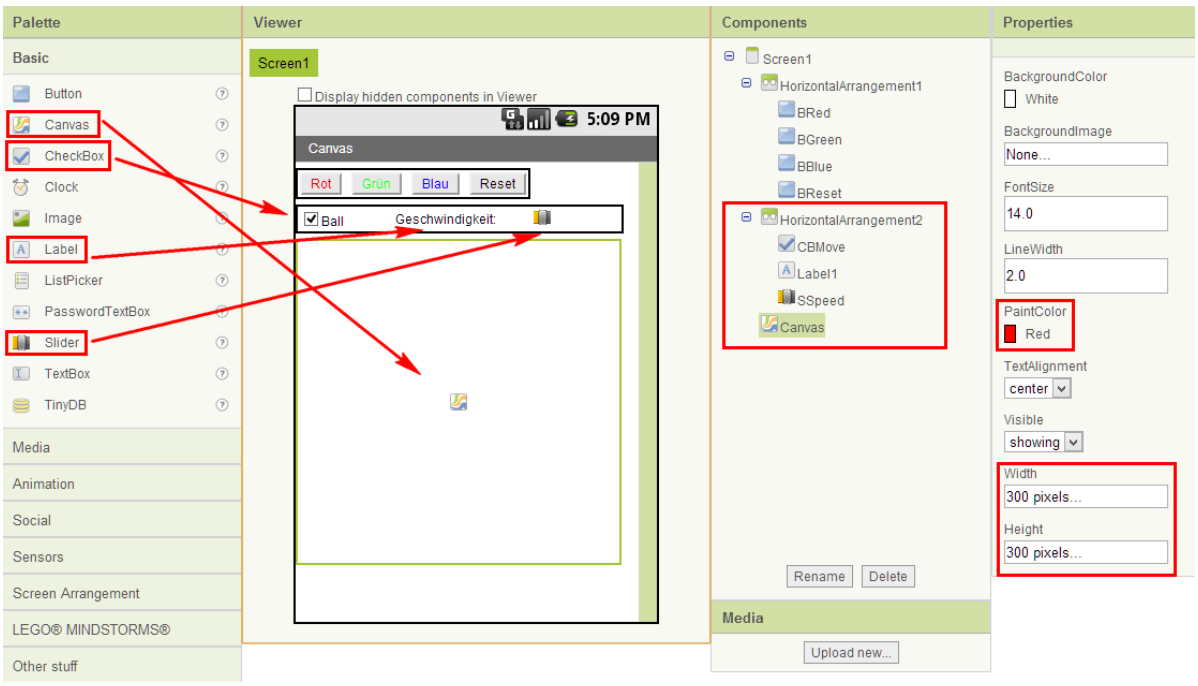
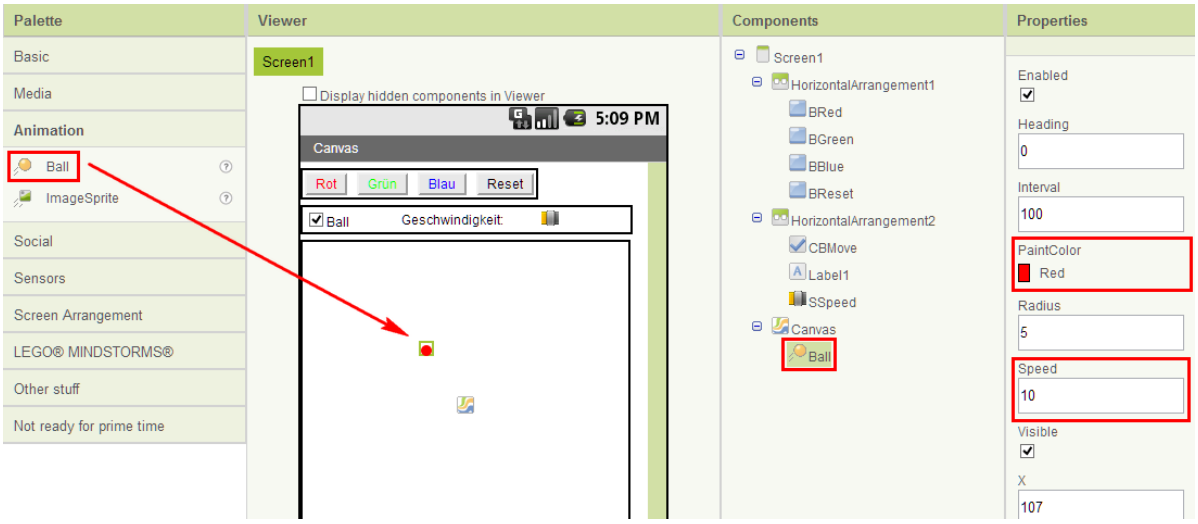


Abbildung 19: weitere Steuerelemente platzieren und anpassen



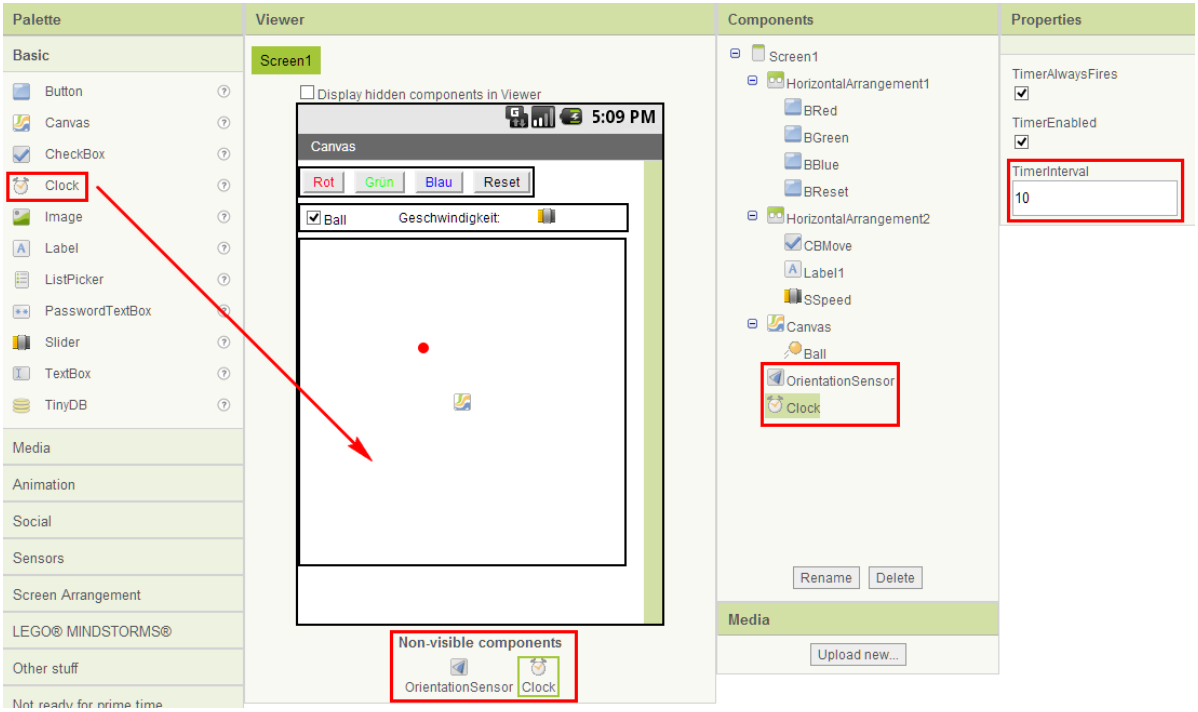
Der Ball, der später durch Drehen des Smartphones gesteuert wird, befindet sich in der Palette, Unterkategorie „Animation“. Er wird an einer beliebigen Stelle auf dem Canvas-Steuerelement platziert. Im Wesentlichen entspricht das Ball-Steuerelement dem ImageSprite-Steuerelement mit der Besonderheit, 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).

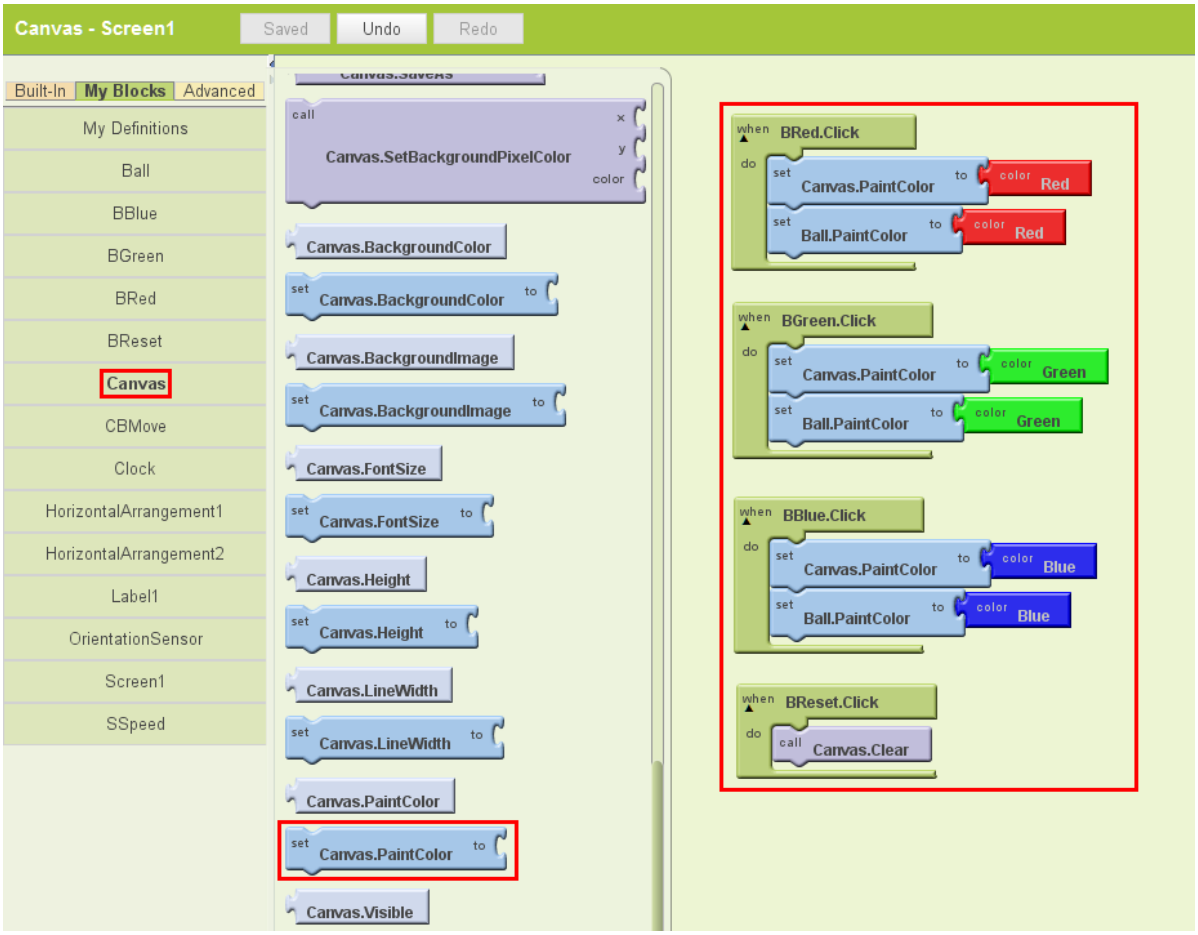
Abbildung 21: Clock und OrientationSensor einfügen



2.3.2 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).

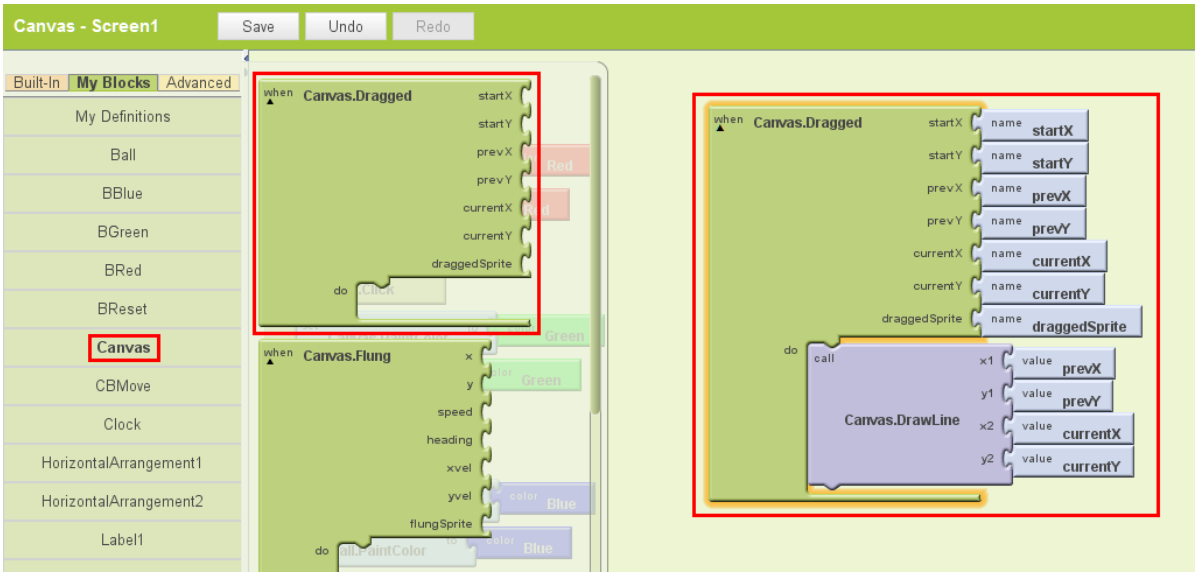
Abbildung 22: Eventhandler für Buttons zur Farbauswahl programmieren



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 Orientierungssensors

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



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.

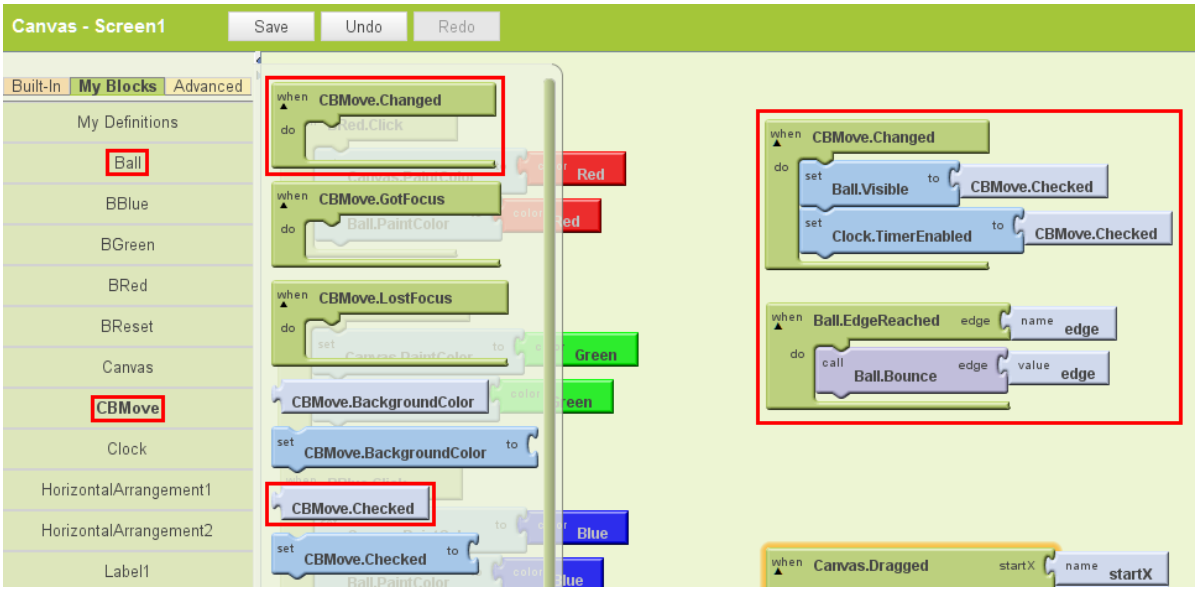
Ferner soll der Ball an den Kanten des Canvas-Steuerelements abprallen. Dies geschieht, indem im „Ball.EdgeReached“-Event „Ball.Bounce“ aufgerufen und der Funktion die entsprechende Ecke in Form des „edge“-Parameters des Events übergeben wird (siehe Abbildung 24).

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:

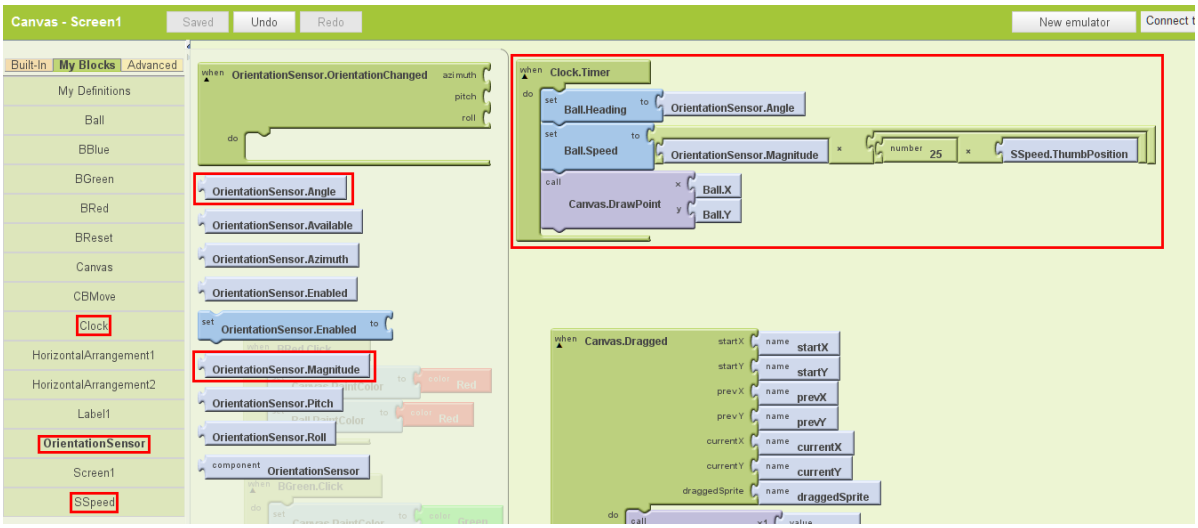
$$\text{Ball.Speed} = \text{OrientationSensor.Magnitude} \cdot 25 \cdot \text{SSpeed.ThumbPosition}$$

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

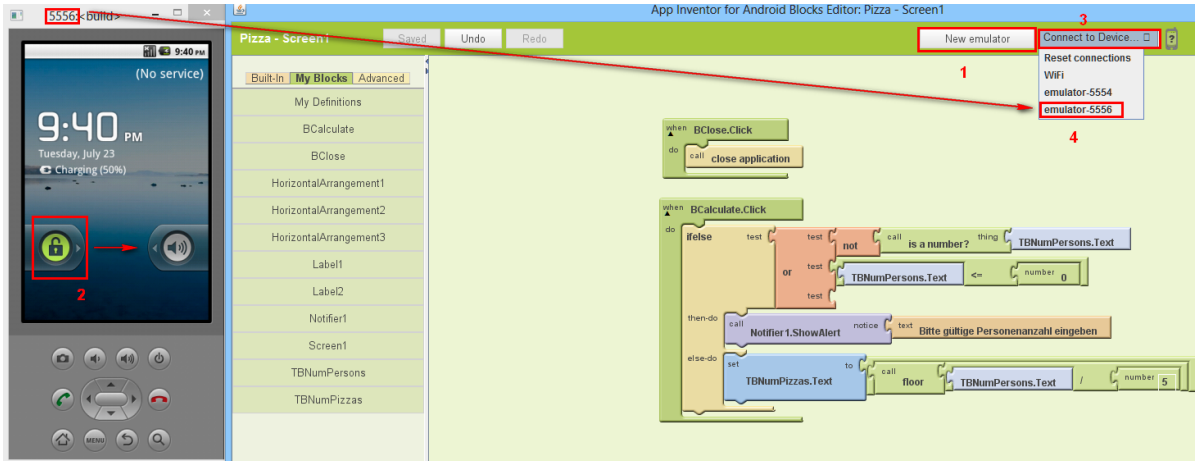


Mit diesem Schritt ist die App fertiggestellt. Sie eignet sich als kleines Spiel: Es können Labyrinthe gezeichnet werden, durch die der Ball gelenkt werden muss ohne die Wände zu berühren.

2.4 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. 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



2.5 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.

3 Scratch - Erstellung kleiner 2D-Computerspiele

3.1 Kennenlernen der Oberfläche

Die Entwicklungsumgebung Scratch kann direkt gratis unter `http://scratch.mit.edu` online verwendet werden. Hierfür ist eine aktuelle Java Version und der Adobe Flash Player notwendig. Alternativ steht das Programm auch als Download zur Verfügung. Zum Starten der Onlineversion klicke auf der Website auf den Button „Probiere es aus“ (Abb. 27).

Abbildung 27: Ausschnitt aus der Scratch Homepage

Anschließend erscheint die Programmoberfläche (Abb. 28) mit den Hauptbereichen: *Bühne*, *Blockpalette*, *Programmbereich* und *Spriteliste*.

Die Scratch-Projekte setzen sich aus Objekten, den *Sprites* (hier: Grafikelemente) zusammen. Dies können Personen, Tiere oder sonstige Elemente mit verschiedenen Kostümen sein. Es ist sogar möglich eigene Bilder zu importieren. Jedem einzelnen *Sprite* kannst du Befehle für Bewegungsabläufe, Töne oder sogar Interaktionen mit anderen *Sprites* ausführen lassen. Dazu werden grafische Blöcke aus der Blockpalette in den Programmbereich geschoben und dort zu Stapeln zusammengefasst. Mit dem Programmstart über die grüne Flagge im Bühnenbereich erfolgt dann die Abarbeitung der einzelnen Befehlen in der jeweilig angeordneten Reihenfolge.

Bühne Dies ist der Ort, an dem später dein Spiel, Animation oder deine Geschichte abgespielt wird. Die Bühne ist in einem kartesischen Koordinatensystem (x- und y-Koordinaten) eingeteilt und hat eine Größe von 480 $\times$ 360 Bildpunkten.

Blockpalette Hier befinden sich die Blöcke mit den Programmanweisungen, welche in unterschiedlichen Kategorien mit verschiedenen Farben untergeordnet sind. Die üblichen Kontrollstrukturen befinden sich unter `Steuerung` und `Ereignisse`. Blöcke mit Textfeldern können einfach editiert werden. Manche Blöcke weisen sogar ein Untermenü auf, bei denen du einfach aus einer vorgegebenen Auswahl eine Eigenschaft festlegen kannst.

Programmbereich Lege hier die Blöcke aus der Blockpalette ab und verschachtele sie mit anderen Blöcken so zusammen, dass ein in sich geschlossenes Programm entsteht.

Spriteliste Innerhalb dieses Bereiches werden die Vorschaubilder deiner *Sprites*, die du in deinem Programm verwendet hast, angezeigt. Beim Anklicken eines *Sprites* wird dieser für einen kurzen Moment auf der Bühne eingerahmt und du erhälst im Programmbereich die zum *Sprite* zugehörigen Anweisungen. Du kannst nun ebenfalls die Kostüme wechseln, indem du oberhalb von der Blockpalette den Button *Kostüme* betätigst.

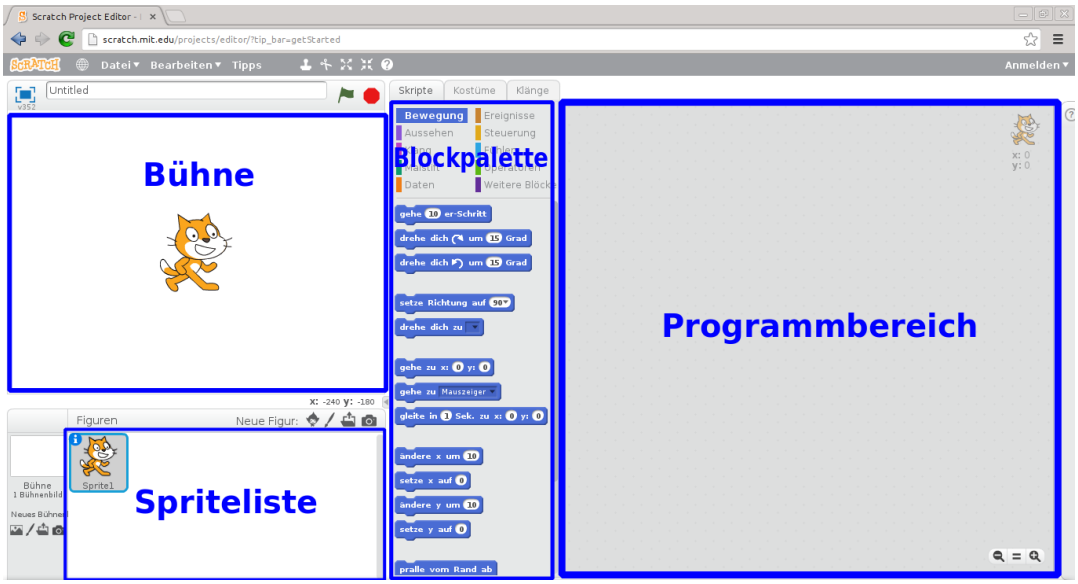


Abbildung 28: Programmoberfläche

3.2 Tutorial: Zeichnen von Blumen

Das folgende Programm eignet sich gut als Einstieg in die Materie der Unterprogrammaufrufe. Ziel ist es, sein Programm in mehrere Teilaufgaben zu unterteilen, die miteinander durch Botschaften kommunizieren können. Nach dem „teile und herrsche“-Prinzip findet sich dies

in der Informatik als grundsätzliche Denkweise beim Entwurf von komplexen Programmieraufgaben wieder.

Bühnenreset

Damit bei jedem Neustart des Programms später eine leere Bühne vorhanden ist, sollten wir mit einem Programm beginnen, dass uns die Bühne freiräumt. Setze dazu das Programm aus der Abb. 29 zusammen. Um eine neue Botschaft in die Kontrollstruktur `Wenn ich ...empfange` einzutragen, öffne die Auswahlliste und klicke auf `neue Nachricht` und gebe dort eine Botschaft ein. Wie wäre es mit der Botschaft „`Clear`“?

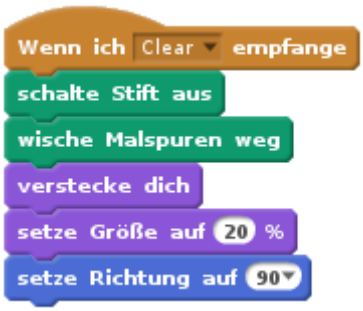


Abbildung 29: Programm für Bühnenreset

Kreis zeichnen

Wir können nun unseren Sprite dazu auffordern einen Kreis zu zeichnen. Dazu startet das Programm aus der Abb. 30 beim Aufruf über die grüne Flagge unser Bühnenresetprogramm und zeichnet danach einen vollständigen Kreis.

Abbildung 30: Programm für das Zeichnen eines Kreises

Der Sprite dreht sich hierbei 360 mal um 1°, sodass dieser insgesamt eine volle Umdrehung (360°) durchführt. Damit der Kreis schneller gezeichnet wird, kann auch 36 mal um 10° in 15 Schritten gedreht werden.

Teilkreis

Nun soll ein Teilkreis gezeichnet werden. Zeichne hierzu mit dem Sprite 12 mal 10°. Ändere hierzu einfach die Parameter im vorherigen Programm. Das Programm ist in der Abb. 31 vorzufinden.

Abbildung 31: Programm für das Zeichnen eines Teilkreises

Blatt

Nun können wir endlich aus zwei Teilkreisen ein Blatt zeichnen. Die Botschaft des ersten Programmblocks aus dem vorherigen Programm kann hierzu einfach abgeändert werden. Der zweite Programmblock muss beibehalten werden, damit dieser vom folgenden Programm aufgerufen werden kann. Wir fügen nun einen weiteren Programmblock (Abb. 32) hinzu, welches zunächst einen Teilkreis zeichnet, dann den Sprite um 60° dreht und darauffolgend einen zweiten Teilkreis zeichnet.

Abbildung 32: Programm für das Zeichnen eines Blattes

Blume

Wir haben es fast geschafft, denn eine Blume besteht aus mehreren Blättern. Wir modifizieren daher nun unser Startprogramm mit der grünen Flagge wie in der Abb. 33, um wiederholt Blätter die im Kreis angeordnet sind zu zeichnen.

Abbildung 33: Programmblock für das Zeichnen einer Blume

Wenn du alles richtig gemacht hast solltest du ein ähnliches Ergebnis wie in Abb. 34 erhalten.

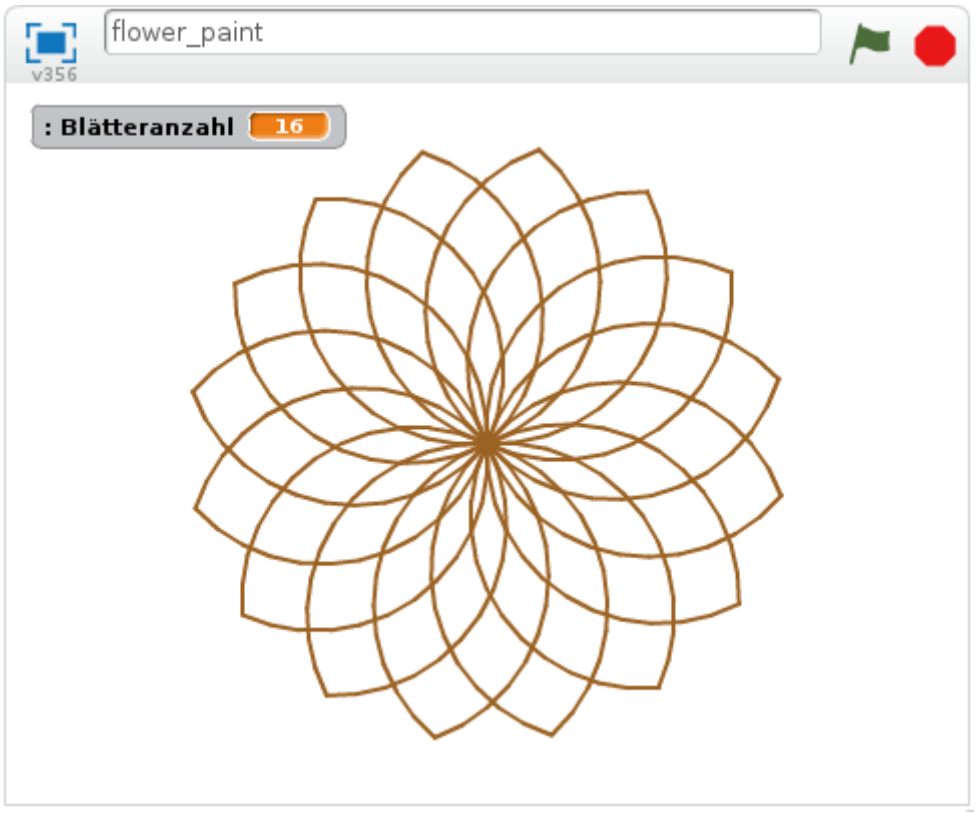


Abbildung 34: Darstellung der Blume

In der Abb. 35 ist nochmal das komplette Programm dargestellt.

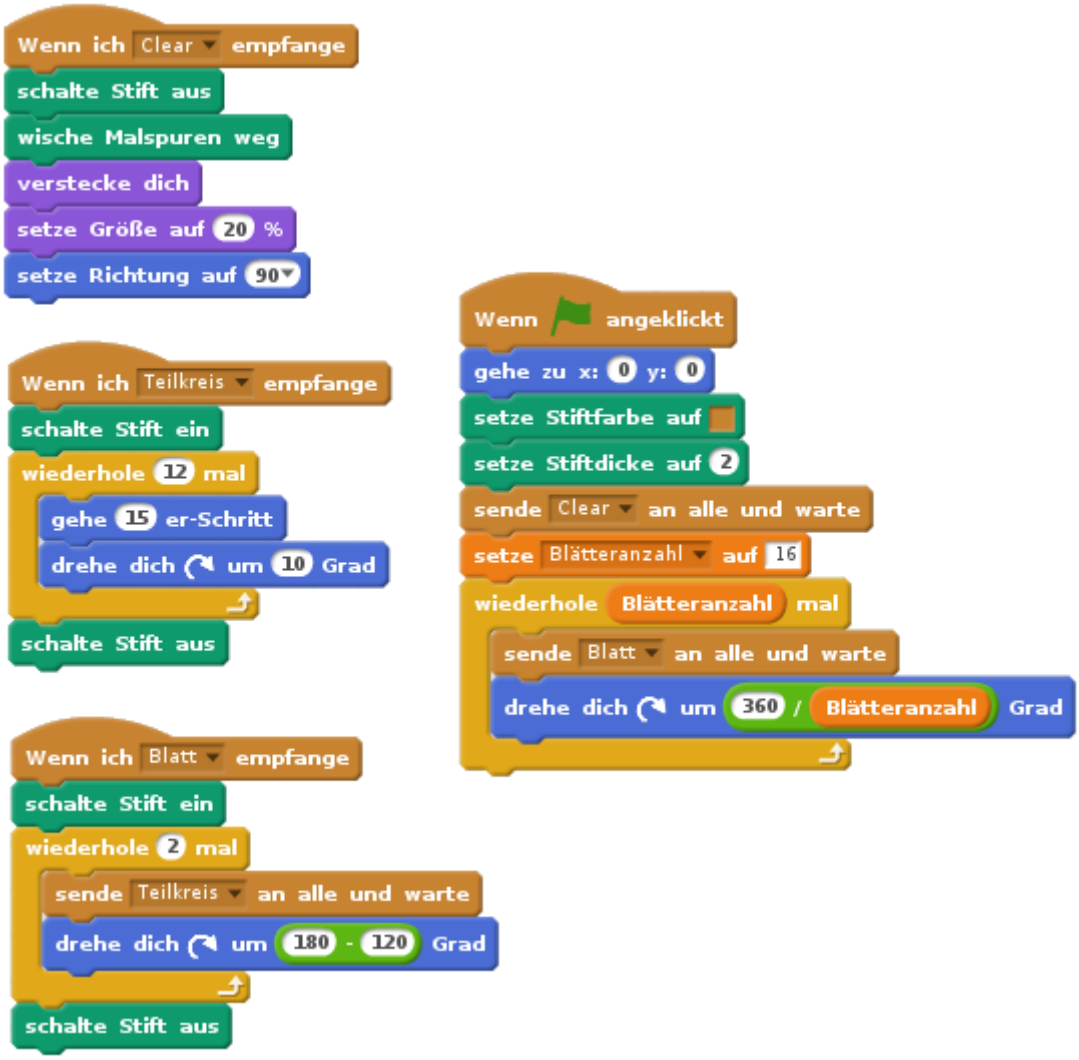


Abbildung 35: Vollständiges Programm für das Zeichnen einer Blume

Zusatz: Blattlänge und -breite

Die Blattlänge wird von der Schrittweite (`gehe 15er-Schritt`) der Teilkreise bestimmt. Definiere hierfür einfach eine Variable `Blattlänge` und weise dieser im Startprogramm einen Wert zu.

Die Blattbreite wird über die Anzahl der Wiederholungen im Teilkreis eingestellt. Definiere auch hierfür eine Variable `Blattbreite` und weise dieser im Startprogramm einen Wert zu. Es muss ebenfalls noch die Drehung im Blattprogramm angepasst werden. Siehe hierzu in Abb. 36 nach.

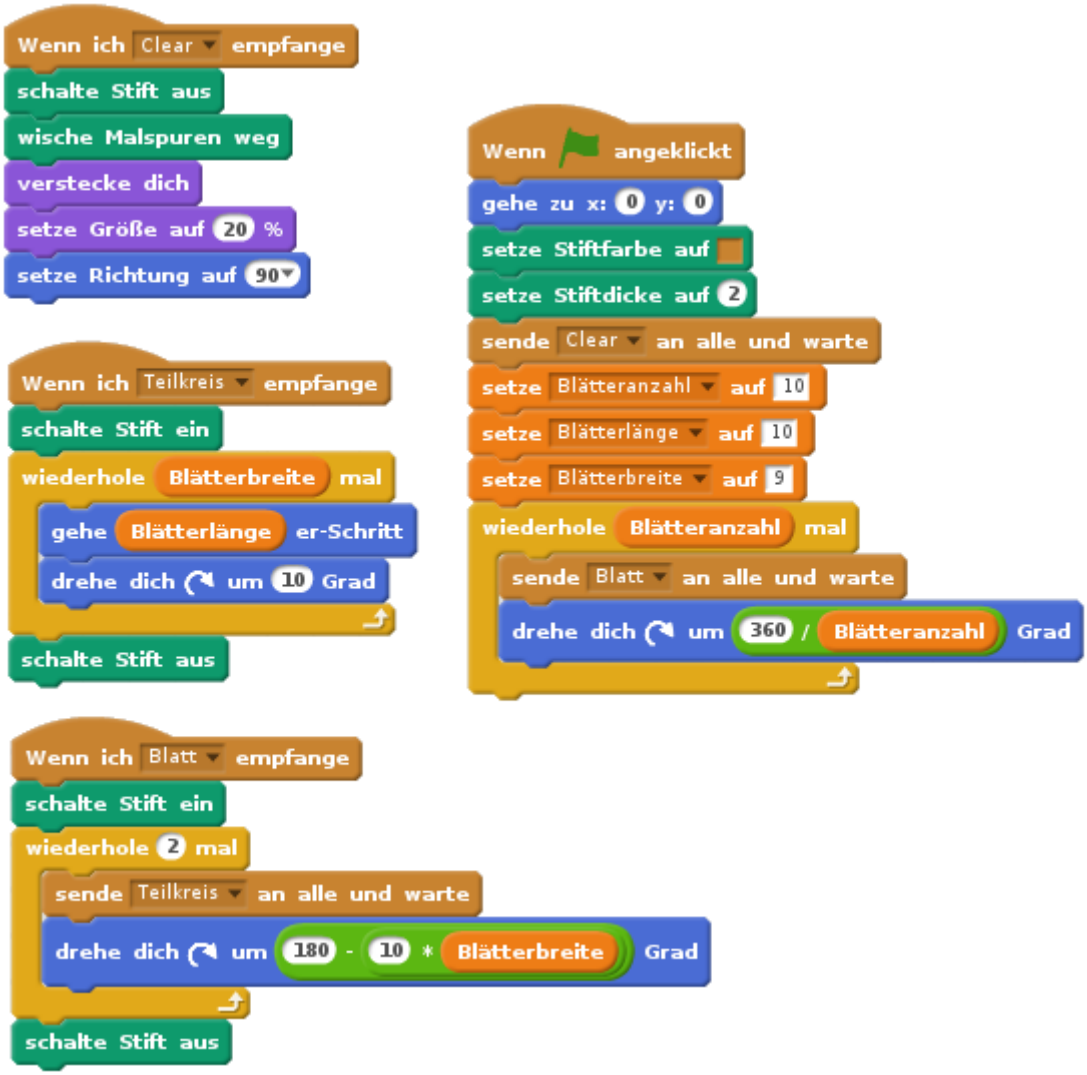


Abbildung 36: Vollständiges Programm mit Angabe von Blattlänge und -breite

3.3 Tutorial: Das Spiel Pong

Wer kennt es nicht, das Kult-Videospiel „Pong“. Ein Ball bewegt sich hierbei willkürlich auf dem Bildschirm hin und her und du steuerst mit der Maus deinen Schläger so, dass dieser Ball nicht auf den Boden trifft. Berührt der Ball dennoch mal den Boden, wird dir ein Leben abgezogen. In unserem später programmierten Spiel setzen wir 5 Leben voraus.

Um das bekannte Spiel auch in Scratch umzusetzen, sind bereits Sprites und zwei Bühnenbilder erstellt worden, damit dir dieser Aufwand erspart bleibt und du dich sofort an die Programmierung machen kannst. Selbstverständlich kannst du bei Bedarf auch eigene Grafiken ver-

wenden. Du findest in der Spriteliste (Abb. 37) ein orangenes Zahnrad (symbolisiert unseren Ball), einen senkrechten Balken (Schläger) und noch eine lange senkrechte Linie die unseren Boden darstellen soll. Als Bühnenbilder (Abb. 38) stellen wir dir einen Sternenhimmel und einen „Game over“-Schriftzug zur Verfügung.

Abbildung 37: Spriteliste

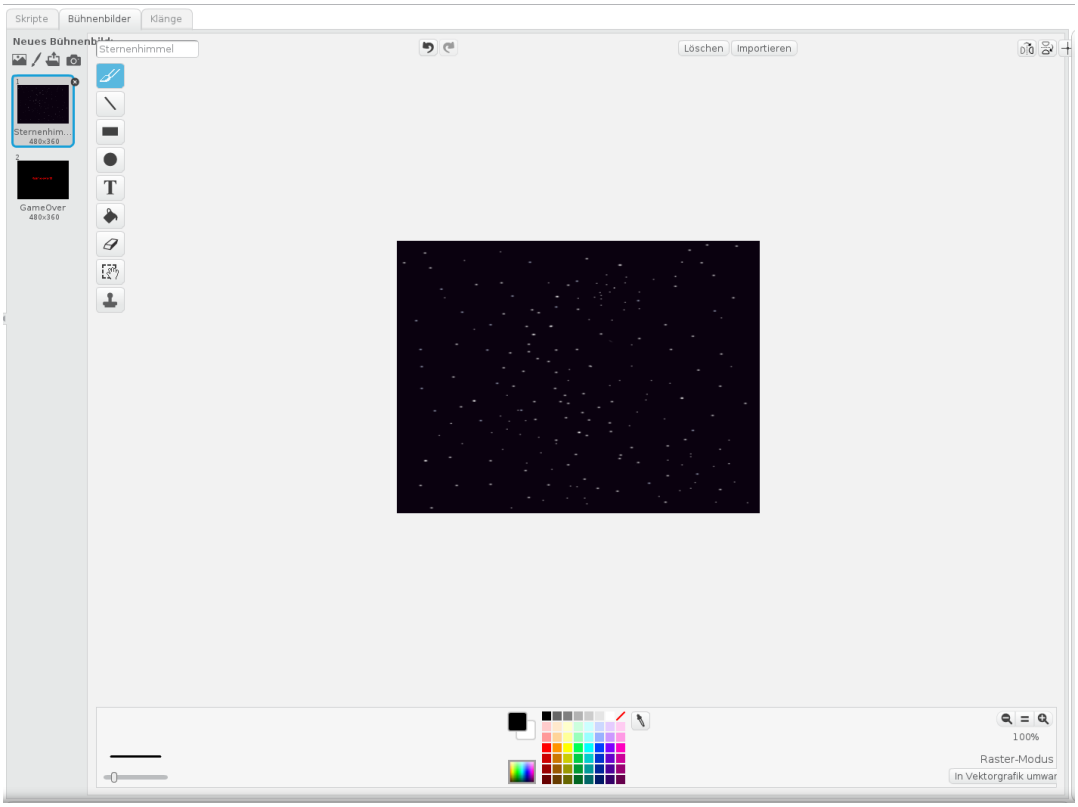


Abbildung 38: Bühnenbilder

Wie bereits in der Einführung erwähnt, kann jeder Sprite und auch Bühnenbilder im Programmbereich eigene Anweisungen erhalten. Wir werden also für unsere Problemstellung ein

Bühnen-, Schläger- und Ballprogramm erstellen. Die Variable, die später die Anzahl der verbleibenden Leben zählt, wurde bereits für dich angelegt und wird dir später in der Auswahlliste angeboten. Das Bühnenbild sieht dann wie in Abb. 39 aus.

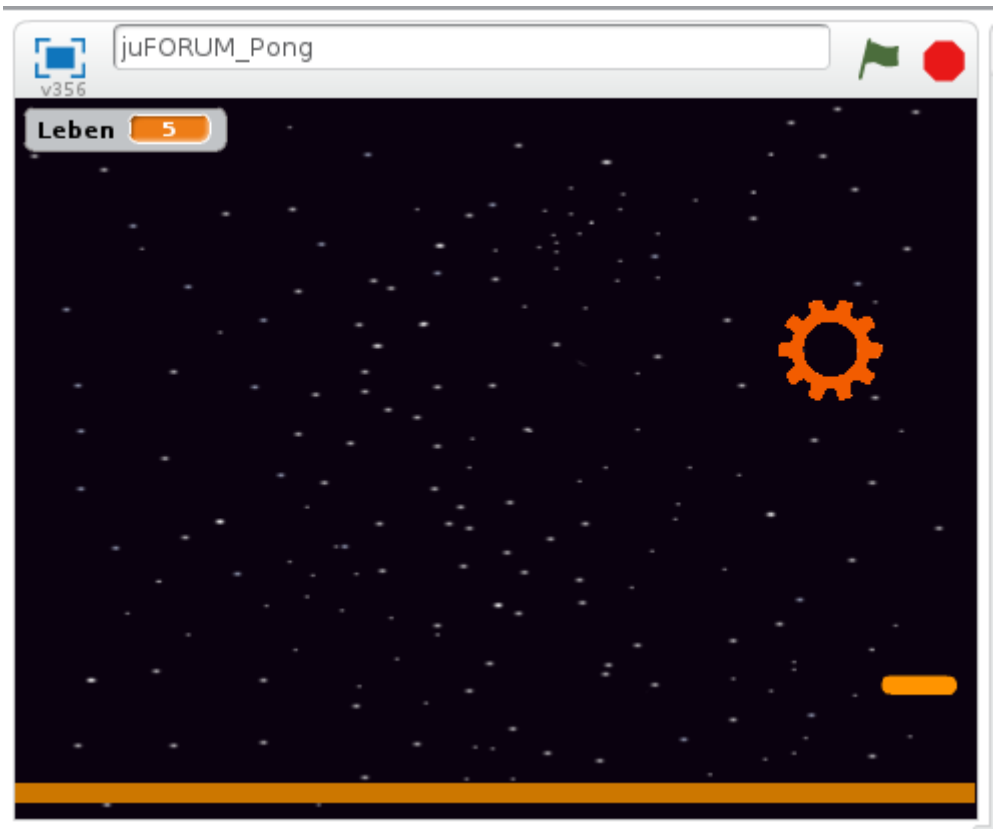


Abbildung 39: Bühne

Bühnenprogramm

Klicke links neben der Spriteliste auf die Vorschau der Bühnenbilder und wechsele mit der Ansicht in den Skriptemodus (Programmbereich). Wir möchten nun gerne, dass wenn das Programm mit der grünen Flagge gestartet wird, der Sternenhimmel in der Bühne verwendet wird. Ebenfalls soll die Variable `Leben` den Wert 5 erhalten. Sobald wir alle Leben verbraucht haben, soll der Schriftzug „Game over“ erscheinen. In der Abb. 40 wurde das gewünschte Verhalten umgesetzt.

Abbildung 40: Bühnenprogramm

Schlägerprogramm

Damit wir den Schläger mit der Maus bewegen können, setzen wir in einer Dauerschleife die x-Position des Schlägers als x-Position der Computermaus. Wechsele zum Sprite `Schläger` und setze die eben beschriebene Eigenschaft mit den Blöcken um (Abb. 41).

Abbildung 41: Schlägerprogramm

Ballprogramm

Wähle nun das Zahnrad in der Spriteliste aus, damit wir die eigentliche Logik des Spiels programmieren können. Beim Programmstart wird die Startposition des Zahnrads auf (0,0) gesetzt und anschließend die Richtung durch Zufallszahlen (± 80) festgelegt. Fortlaufend wird nun Folgendes wiederholt durchgeführt:

- Der Ball bewegt sich in 8er Schritten fort und prallt bei Bedarf vom Rand ab.
- Falls der Boden berührt wird, verlieren wir ein Leben und es wird erneut die Startposition eingenommen. Die Richtung wird ebenfalls wieder mit Zufallszahlen festgelegt.
- Falls der Rand oder der Schläger berührt wird, bewegt sich das Zahnrad in 8er Schritten zurück und dreht sich in eine zufällige Richtung.

- Falls wir keine Leben mehr besitzen, erscheint der Schriftzug „Game over“ und das Zahnrad bewegt sich zur Mitte hin und führt eine Abschlussanimation aus.

Das Programm ist in Abb. 42 einsehbar.

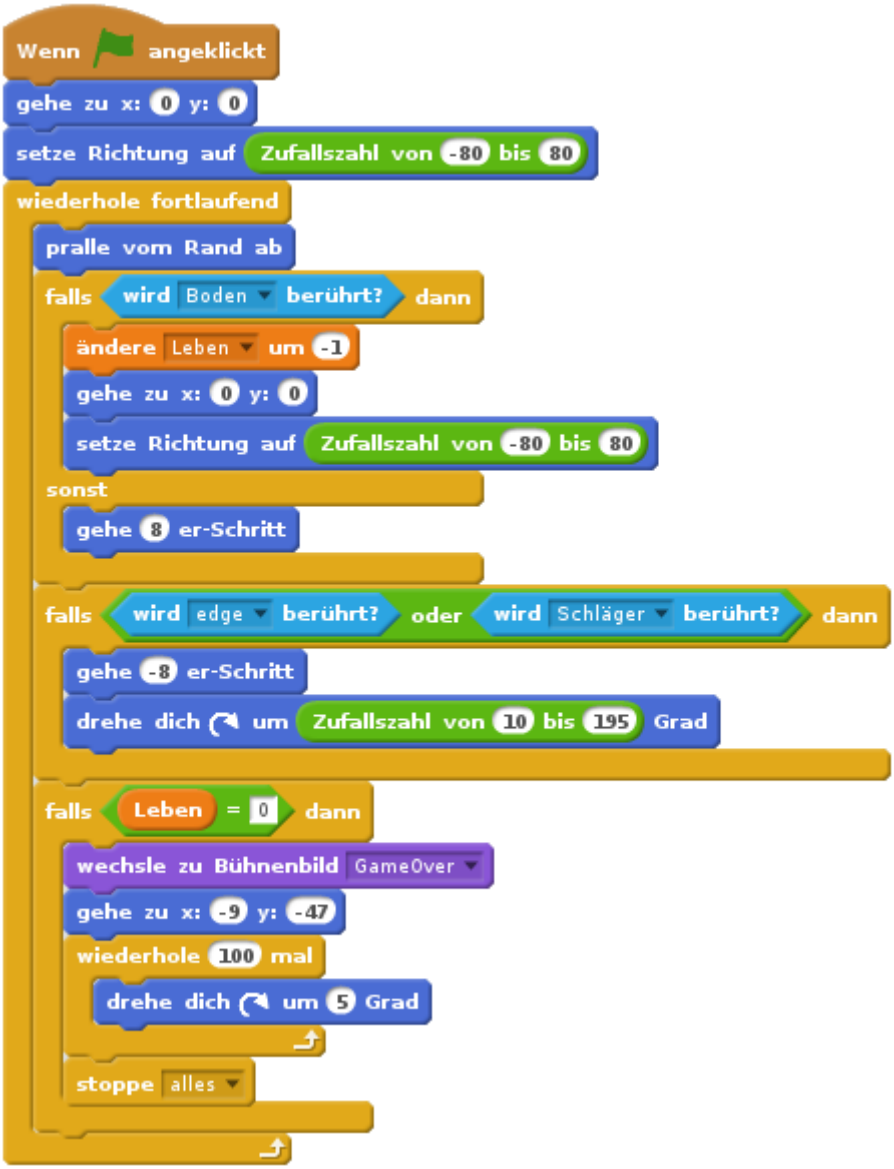


Abbildung 42: Ballprogramm

4 Selbstbau-Feuerlöscher

Der Feuerlöscher[1] - eine Black-Box mit magischem Inhalt? Das wirst du nun ändern.

4.1 Materialien

- Erlenmeyerkolben mit Stopfen und Winkelrohr
- Spatellöffel
- Wasser (300 mL)
- Chemikalien:
  - Natriumhydrogencarbonat $\mathrm{NaHCO_3}$ (Backpulver)
  - Spülmittel
  - Zitronensäure

4.2 Durchführung

- 300 mL Wasser in Erlenmeyerkolben füllen
- 5 Spatellöffel Natriumhydrogencarbonat hinzugeben und bis zum Lösen rühren
- 15–20 Tropfen Spülmittel hinzugeben
- 2 Spatellöffel Zitronensäure hinzugeben und den Kolben sofort mit dem Stopfen verschließen. Dabei ist darauf zu achten, dass das Winkelrohr ungefähr 1 cm über der Flüssigkeitsoberfläche ist.

Hinweis für den Experimentator:
Es findet sofort eine exotherme Reaktion statt (*Schutzbrille verwenden*). Mit dem Steigrohr auf keine Personen zielen. Steigrohr in Richtung eines Eimers oder Ähnlichem richten, da es zu einer Sauerei führen könnte.

[1] Modifiziert nach: http://www.lehrer-online.de/feuerloescher.php?show_complete_article=1

5 Latentwärmespeicher

5.1 Materialien

- Natron
- Essigessenz (25 % Säure)
- dicht verschließbares Plastikbeutelchen
- Wassertopf und Heizplatte

5.2 Durchführung

1. **Herstellung des Latentwärmespeicherssalzes:**

   40 g Natron und 120 g Essigessenz werden langsam unter Rühren zusammengegeben, wobei sich CO_2 entwickelt. Dann wird das Produkt solange gekocht, bis sich die Masse etwa halbiert hat. Bei dieser Ansatzgröße sollte die Endmasse etwa 60 g–80 g betragen. Die erhaltene Lösung lässt man nun abkühlen und stellt sie dann in den Kühlschrank. Nach einigen Stunden sollten aus der Lösung Kristalle ausgefallen sein. Die überschüssige Lösung wird abgegossen und die Kristalle werden geerntet.

2. **Bau und Aktivierung des Latentwärmespeichers:**

   Die Kristalle werden nun in einen dicht verschließbaren Plastikbeutel gegeben. Hierbei sollte darauf geachtet werden, dass keine Fremdstoffe in den Beutel gelangen. Staub oder ähnliches in dem Latentwärmespeicher sorgt für eine vorzeitige Kristallisation und verhindert somit die Speicherung der Wärme. Anschließend wird der dicht verschlossene Beutel in heißem Wasser solange erwärmt, bis das gesamte Salz geschmolzen ist. Danach wird der nun aktivierte Latentwärmespeicher vorsichtig auf Raumtemperatur abgekühlt. Dabei muss der Inhalt des Wärmespeichers flüssig bleiben.

3. **Freisetzung der gespeicherten Wärme:**

   Soll die Wärme nun wieder freigesetzt werden, so muss der aktivierte Wärmespeicher durch mechanische Belastung (quetschen, biegen, etc.) gestartet werden. Sobald der Inhalt anfängt auszukristallisieren, wird die Wärme freigesetzt. Die Temperatur steigt dabei auf 55 °C–60 °C an.

6 Isolierung von DNA aus Erdbeeren

Viele von euch werden sich vielleicht fragen: „Geht das denn so einfach? "- Ja es geht, und dass nur mit Hilfe von Materialien die ganz bestimmt jeder zu Hause hat.

6.1 Materialien

Für den Versuch braucht ihr folgendes:

- 1 Erdbeere, da seht ihr den Effekt am besten
- 1 Gefrierbeutel oder ein wasserdichter Beutel zum verschließen
- 1 EL Salz
- 1 EL kaltes verd. Spülmittel (1 Teil Spülmittel und 2 Teile Wasser)
- 1 EL eiskalter 90 % Alkohol (Ethanol oder Isopropanol; es geht auch 70 % Alkohol)
- 1 großes Reagenzglas oder ein anderes durchsichtiges Gefäß
- 1 raues Holzstäbchen (z.B. Schaschlikspieße oder Zahnstocher)
- Evtl. ein Mixer und ein Sieb

Hinweis: Den Alkohol bekommt ihr ganz einfach in jeder Apotheke für ein paar Euros.

6.2 Los geht's

Wenn ihr alles her bereitet habt, kann's nun auch los gehen:

1. Schritt: Um an die DNA in den Zellen des Obstes oder des Gemüses ran zukommen, müsst ihr dieses so gut wie möglich zerkleinern und zerdrücken bis ihr eine homogene Flüssigkeit vorliegen habt. Je nachdem, aus was ihr die DNA isolieren wollt, müsst ihr das Obst oder das Gemüse mit dem Mixer zerkleinern.

2. Schritt: Anschließend gebt ihr das Salz und die kalte Mischung aus Spülmittel und Wasser dazu. Je nachdem wie viel homogene Substanz ihr habt müsst ihr die Menge des Salzes und des verd. Spülmittels anpassen. Um die Zellmembran und die Kernmembran der Zelle zu zerstören, mixt ihr die so erhaltene Mischung gut durch.

3. Schritt: Nachdem ihr die Mischung ungefähr 5 min gemischt habt, gebt ihr den Beutelinhalt in das Reagenzglas oder in ein anderes Glasgefäß. Je nachdem, aus was ihr die DNA isoliert, müsst ihr die Mischung zuvor filtrieren.

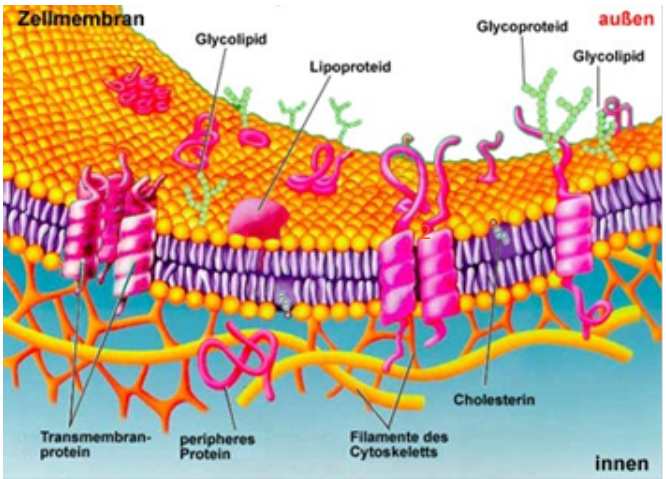
4. Schritt: Um die so isolierte DNA von den übrigen Zellbestandteilen zu trennen, überschichtet ihr die homogene Substanz ungefähr 1 cm hoch mit Alkohol. Dabei ist es sehr wichtig, dass der Alkohol eiskalt ist. Dazu legt ihr ihn am besten auf Eis oder in ein Gefrierfach. Durch die unterschiedlichen Dichten der Mischung und des Alkohols bilden sich zwei Phasen aus.

5. Schritt: Nach kurzer Zeit gehen die DNA-Moleküle aus der unteren wässrigen Phase in die obere Alkoholphase über und fallen aus. Die DNA-Moleküle liegen nun als langer Strang vor und man kann sie mit Hilfe eines Holzspießes aufwickeln.

6.3 Was ist da passiert?

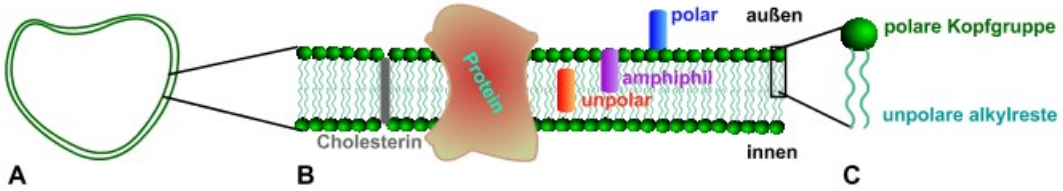
So, nun habt ihr es geschafft ...

Für die unter euch, die verstehen wollen, was genau bei den einzelnen Schritten passiert, erklären wir euch nun kurz die Theorie des Versuches.



Die DNA eines Organismus befindet sich in jeder Zelle im Zellkern, der, wie die Zelle selbst, von einer Zellmembran umgeben ist. Diese Membran besteht hauptsächlich aus Lipiden und Proteinen.
Die Lipide haben einen Wasser anziehenden bzw. in Wasser löslichen Kopf (hydrophiler Teil / polare Kopfgruppe) und einen Fett anziehenden bzw. in Fett löslichen Schwanz (lipophiler Teile / unpolarer Alkylrest). Da sich die Lipide in einer wässrigen Umgebung befinden, bilden sie eine Lipiddoppelschicht aus, bei der die lipophilen Schwänze zueinander und die hydrophilen Köpfe zu der wässrigen Umgebung zeigen.

[2] http://www.bioserve.info/Ablage/Verfahrensbeschreibung



[3]

Aufbau der Zellmembran:

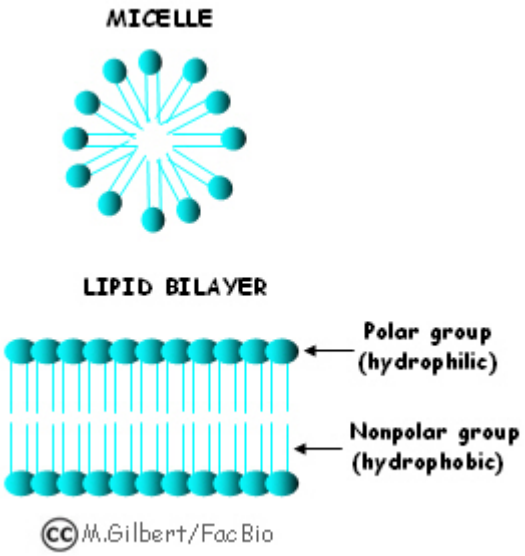
A. Lipiddoppelschicht um eine Zelle.

B. Vergrößerter Abschnitt mit eingelagertem Protein und Cholesterol - polare Wirkstoffe lagern sich an der Membranoberfläche an, amphiphile Stoffe richten sich entsprechend ihrer Struktur innerhalb der Membran aus, und unpolare Drogen reichern sich innerhalb der Membran an.

C. Vergrößerung eines Membranlipids mit der nach außen zeigenden polaren Kopfgruppe und den nach innen gerichteten unpolaren Alkyl-Resten. Um die DNA einer Zelle isolieren zu können, muss also zuerst die Membran aufgebrochen werden und die Zellen aus ihren Zellverbänden gelöst werden. Dies erreicht ihr, indem ihr euer Obst bzw. euer Gemüse zerkleinert. Als nächstes müsst ihr nun die Zellmembran und die Kernmembran der Zelle zerstören, indem ihr das Wasser-Spülmittel-Gemisch und das Salz hinzu gebt. Das Spülmittel wirkt dabei als Detergenz. Detergenzien sind waschaktive Substanzen, die die Oberflächenspannung von Wasser herabsetzen und die Fette im Gemisch binden. Durch die ähnliche chemische Struktur der Detergenzien und der Lipide in den Biomembranen können Detergenzien die Membranen „aufbrechen“ und sogenannte Micellen bilden.

[3] http://www.chemgapedia.de/vsengine/vlu/vsc/de/ch/8/bc/vlu/zellbio/zellaufbau.vlu/Page/vsc/de/ch/8/bc/zellbio/membrane2.vscml.html

Nun liegen der Zellinhalt und der Zellkern frei vor. Da ihr aber nun neben dem Zellkern mit der DNA auch noch all die andern Zellorganellen vorliegen habt (v.a. Proteine und Kohlenhydrate), müsst ihr diese nun von dem Zellkern trennen. Dies erreicht ihr durch eine Ethanol- bzw. Isoprapanol-Fällung.
Durch die unterschiedlichen Dichten von Wasser ($1\,\mathrm{g\,cm^{-3}}$) und Alkohol (Ethanol $0.79\,\mathrm{g\,cm^{-3}}$, Isopropanol $0.78\,\mathrm{g\,cm^{-3}}$) bilden sich zwei Phasen aus, wobei das Wasser mit dem Zellgemisch unten und der reine Alkohol oben liegt.



[4]

Nach kurzer Zeit erkennt man, dass die DNA von der Wasserphase in die Alkoholphase übergeht. Durch den Alkohol wird die stabilisierende Hydrathülle um die Phosphatreste der DNA verdrängt und das Molekül so destabilisiert. In Folge dessen fällt die DNA aus und liegt nun frei in Form von langen Fäden in der Alkoholphase vor. Mit dem Holzspieß könnt ihr diese nun aufwickeln und näher betrachten. Sie hat eine schleimige, gelartige Konsistenz.

6.4 Videomaterial

Unter folgendem Link finden ihr ein Video, das den Versuchsablauf nochmals verdeutlicht: http://www.youtube.com/watch?v=g3LTeaiVMbI

[4] http://www.chem.fsu.edu/chemlab/chm1046course/solnprocess.html

7 Kartoffelbatterie

Ja, du hast richtig gelesen. Du kannst eine Batterie aus Kartoffeln herstellen[5]. Es ist ein etwas kniffeliges Experiment, aber mit etwas Geduld funktioniert es bestimmt. Bitte lies dir die Bauanleitung genau durch. Es gibt ein paar Kleinigkeiten, die wichtig sind! Die Tipps sind sehr nützlich!

7.1 Material

- 1 Kartoffel
- 1 Messer
- 2 Kabel mit Krokodilklemmen
- 1 LED
- 1 Kupferscheibe
- 1 Zinkscheibe

7.2 Durchführung

1. Zuerst die Kartoffel mit dem Messer zweimal einschneiden.
2. Dann Zinkscheibe und Kupferscheibe in nicht zu großem Abstand in die Schlitze stecken.
3. LED vorsichtig auseinander biegen und mithilfe der Kabel jeweils mit der Kupfer- und der Zinkscheibe verbinden.
4. Bei Bedarf: LED umdrehen! (Sie könnte in Sperrrichtung geschaltet worden sein)

Kupferscheiben sind leider schwer zu finden. Du kannst auch einige Cent-Münzen benutzen, sie sind nämlich auch aus Kupfer.

[5] Quelle: http://www.physikfuerkids.de/lab1/versuche/kartoffel/

8 Amesraum

Sich und seine Wahrnehmung reinzulegen kann tatsächlich Spaß machen! Mit einigen wenigen Materialien und etwas Geschick kannst du mit der Vorlage auf unserer Website[6] eine optische Täuschung basteln, die bis heute als Filmtrick zum Einsatz kommt.

8.1 Material

- 1 Bastelvorlage
- 1 Bastelschere
- 1 Klebestift
- 2 gleich große Gegenstände mit weniger als 3 cm Höhe, etwa Lego-Figuren
- 1 Fotoapparat (freiwillig)

8.2 Durchführung

1. Schneide die beiden Hälften der Vorlage und das Guckloch vorsichtig mit der Schere aus.
2. Setze den Amesraum an den Laschen zusammen und verklebe alles bis auf eine Wand.
3. Stelle die zwei Gegenstände rechts und links in den Amesraum, sodass du sie durch das Guckloch sehen kannst.
4. Schließe die Wand (du kannst sie mit einem Klebefilmstreifen fixieren oder einfach festhalten) und blicke durch das Guckloch.
5. Fotografieren erlaubt!

[6] Zu finden unter: https://www.juforum.de/veranstaltungen/messen/juforum-messen/