Dokumentation AirSwimmer

# 1 Anforderungen

# 2 Vorbereitung

## 2.1 Versionsverwaltung

## 2.2 Android-Tutorium

# 3 Alternativen

## 3.1 Alternativen ohne Infrarot

*von Anja Hafner*

Für den Fall, dass die Ansteuerung des AirSwimmers mit Infrarot nicht realisiert werden kann, wird nach alternativen Geräten gesucht.
Diese Geräte sollten mittels Bluetooth ansteuerbar sein, nicht mehr als 100 € kosten und fliegen können.

### Warum Bluetooth?

Die Steuerung per Bluetooth ist gewünscht, da generell alle aktuellen Tablets und Smartphones Bluetooth besitzen und keine zusätzliche Hardware (wie bei der Steuerung mit Infrarot) nötig ist. Außerdem ist die Umsetztung dieser Kommunikation mit dem Fisch deutlich einfacher als mit Infrarot, da die Kommunikation mit Bluetooth besser dokumentiert ist.

### Warum soll es fliegen?

Da es sich um ein Avionik-Projekt handelt, sollte der Gedanke, etwas flugfähiges zu steuern, nicht verloren gehen.

### Suche nach Alternativen

Zu Beginn wird nach einem AirSwimmer gesucht, der mit Bluetooth gesteuert werden kann. Jedoch gibt es keinen alternativen AirSwimmer, der mit Bluetooth gesteuert wird, lediglich ferngesteuerte AirSwimmer werden angeboten.
Aus diesem Grund wird die Suche auf alle flugfähigen, mit Bluetooth und Android gesteuerten Geräte, erweitert.

**Abbildung 1**

Auch hier ist die Auswahl nicht groß, lediglich eine flugfähige Alternative der Firma BeeWi wird angeboten. Dieser Hubschrauber ist per Bluetooth 3.0 ansteuerbar und bietet bis zu 8 Minuten Flugzeit. Die Ladezeit hingegen beläuft sich auf ca. 40 Minuten und erfolgt per USB. Ein weiterer Nachteil ist die schwere Steuerbarkeit des Hubschraubers. Die Kosten des BeeWi BB/301-A0 Bluetooth Controlled Hubschraubers in schwarz belaufen sich auf 50,37 €.

## Fahrfähige Alternativen

Da nur eine flugfähige Alternative mit Bluetooth gefunden wurde, wird ebenfalls nach Bluetooth-gesteuerten Fahrzeugen gesucht, auch wenn dies nicht ganz den Anforderungen entspricht. Der Vorteil von Fahrzeugen ist die einfachere Steuerbarkeit. Das erste Modell stammt ebenfalls von der Firma BeeWi und ist ein Mini Cooper. Dieses Modellauto wird mit Bluetooth 2.0 gesteuert und hat eine Reichweite von bis zu 10 Metern. Für den Betrieb sind 3 AA Baterien erforderlich und die Kosten betragen 30,54 € bei Amazon.

**Abbildung 2**

Eine weitere Alternative sind die sogenannten Tankbots. Die kleinen Fahrzeuge werden per USB aufgeladen und besitzen 3 bereits vorhandene Fahrmodi: sie können Hindernisse selbständig ausweichen, in freien Bewegungen "tanzen" oder per Smartphone gesteuert werden. Die Kosten belaufen sich auf 24,99 € bei Amazon.

**Abbildung 3**

Für den Fall, dass die Ansteuerung per Infrarot nicht realisiert werden kann, wurde der Hubschrauber als mögliche Alternative ausgewählt.

## 3.2 Simulation/Emulation

# 4 Projektleitung

# 5 Senden

## 5.1 Irdroid-App

## 5.2 Senden

# 6 Empfangen

## 6.1 Signalanalyse

## 6.2 Lirc-File

# 7 Gehäuse

## 7.1 Recherche Gehäusemöglichkeiten

## 7.2 Gehäuse bauen

## 7.3 Gehäuse bemalen und verschönern

*Von Belgüzar Kocak*

Nachdem ein passendes Gehäuse gefunden wurde und auch der Anfang zur Gestaltung von Andreas Gerken gegeben wurde, musste dieses noch farblich gestaltet werden. Als erstes wurden die Plastikflossen gefeilt anschließend mit Tesa Sekundenkleber

angebracht. Die farbliche Gestaltung wurde mit Acrylfarbe und Naturschwamm erarbeitet.

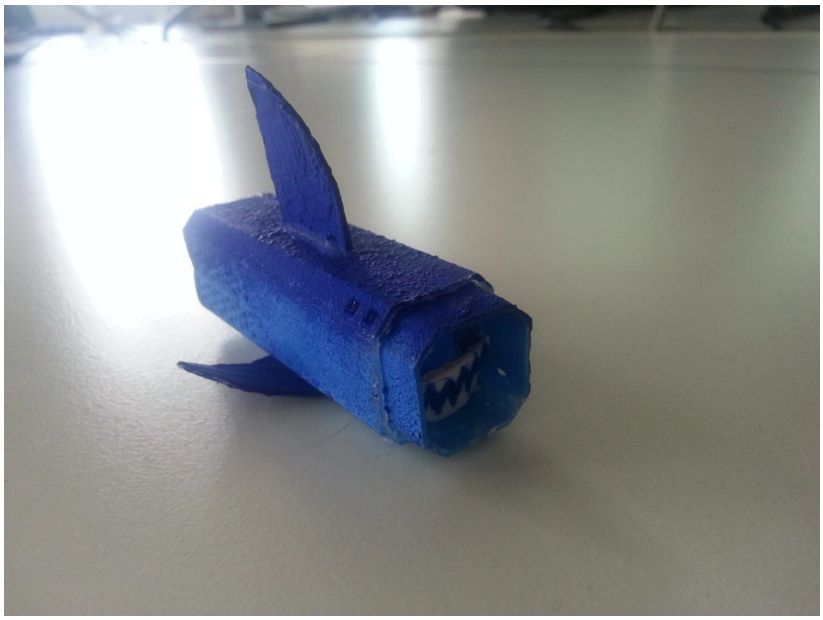


**Abbildung 4**
fertiges Gehäuse für Irdroid-Adapter

# 8 Oberfläche

## 8.1 Konzept

*Von Anja Hafner, Melanie Knappe, Belgüzar Kocak*

### *Zielsetzung*

Ziel war es mit der Oberflächen Gruppe Anja Hafner, Melanie Knappe & Belgüzar Kocak Konzepte zur Steuerung des Air Swimmers zu entwickeln.

### *Vorüberlegung und Analyse*

Die Ansteuerung des Air Swimmers erfolgt durch eine Infrarot Verbindung. Diese enthält folgende Funktionen:
1. Ansteuerung der hinteren Flosse (nach links oder nach rechts)
2. Ansteuerung des Gewichtes am AirSwimmer (Flug nach oben oder nach unten

Auf diese Funktionen soll die Oberflächensteuerung aufbauen. Somit wurden drei verschiedene Ansteuerungsvarianten zur Realisierung der Oberfläche erstellt, die wiederum die Steuervarianten Single und Permament enthalten.
Bei der Ansteuerungsart Button stellt die Bewegungsart Single die Funktion der Fernbedienung nach. Die Ansteuerungsart Permanent enthält einen zusätzlichen Button Start. Durch betätigen des Start Buttons bewegt sich der Air Swimmer aufgrund der permanenten Ansteuerung der hinteren Flusse vorwärts. Die Buttons Links & Rechts ermöglichen eine Links-/ Rechtskurve.
Beim Wischen stellt die Single Bewegungsart folgendes dar: durch eine Wischbewegung nach Links bewegt sich die Flosse einmal nach links. Durch die

Wischbewegung nach rechts bewegt sich die Flosse einmal nach rechts. Bei der Permanentfunktion fliegt der AirSwimmer durch eine Berührung der Oberfläche nach vorne. Durch das Wischen nach rechts/links fliegt der AirSwimmer die entsprechende Kurve.
Bei der Nutzung der Ansteuerungsart Kippen erscheint beim Start ein Start Button. Durch Drücken auf diesen startet der Fisch. Bei der Bewegungsart Single bewegt sich die Hinterflosse einmal nach links oder rechts, je nach Kipprichtung. Durch das Kippen nach oben/unten wird das Gewicht am AirSwimmer angesteuert. Wird der Bildschirm berührt, reagiert der AirSwimmer nicht mehr und der Startbildschirm erscheint. Bei der Bewegungsart Permanent erscheint auch zunächst der Start Button, nach Betätigung beginnt der Air Swimmer geradeaus zu schwimmen. Durch das Kippen nach links/rechts wird eine Links-/Rechtskurve geflogen. Auch hier erzeugt die Berührung an den Bildschirm ein Stoppen und der Startbildschirm erscheint.
Die Realisierung der verschiedenen Konzepte wird in den nächsten Kapiteln eindeutig erklärt.

## 8.2 Startseite

*Von Caroline Pilot*

### *Zielsetzung*

Jede App benötigt eine Startseite, die es dem Benutzer ermöglicht, eine Auswahl zu treffen, zwischen verschiedenen Seiten und Funktionen zu wechseln und somit den Benutzer empfängt und leitet.
Da die Startseite einen Knotenpunkt für alle Teilanwendungen (Tasten, Kippen, Wischen) darstellt, müssen diese hiermit verknüpft werden.

### *Vorüberlegung*

#### Aussehen

Da die Startseite in erster Linie die Navigation zu den verschiedenen Benutzermodi übernehmen soll, werden drei Buttons gebraucht.
Außerdem soll die erste Seite der App natürlich optisch ansprechend wirken, deshalb wäre eine Eröffnungsanimation oder eine einladende Graphik des AirSwimmers von Vorteil.

#### Programmcode

Mit Hilfe der Java-Klasse „Activity“ können die Buttons mit Aktionen belegt werden, die zu den einzelnen Teilfunktionen führen.
Der Programmcode soll in Englischer Sprache erstellt werden.
Außerdem soll für ein Tablet der Größe 10,1 Zoll mit einer Auflösung von 1280 x 800 Pixeln programmiert werden.

## *Vorgehensweise*

## Erstellen der Startseite

### *a) Konfiguration*

Um die Einbindung der Teilfunktionen einfacher zu gestalten, wird ein neues Android Appication Project aufgesetzt („AirSwimmer“) und mit folgenden Einstellungen konfiguriert:
Minimum Required SDK → API 7: Android 2.1 (Eclair)
Target SDK → API 7: Android 2.1 (Eclair)
Compile with → API 7: Android 2.1 (Eclair)

Um Probleme bei der Abwärtskompatibilität zu vermeiden, werden niedrige SDK-Versionen verwendet, damit die App auch auf Endgeräten mit älteren Android-Versionen lauffähig ist.
Die Launcher-Konfiguration wird über ein „Image“ der Form „Circle“ mit der erstellten Graphik gespeist.
Der „Activity-Name“ ist passend dazu „FrontPage“ sowie der zugehörige „Layout Name“ „front_page“ lautet.

### *b) Layout Erstellung*

Im Unterordner „Layout“ kann nun per Drag&Drop das Erscheinungsbild der Startseite, wie in der Vorüberlegung angesprochen, erstellt werden. Diese setzt sich aus folgenden Komponenten zusammen:

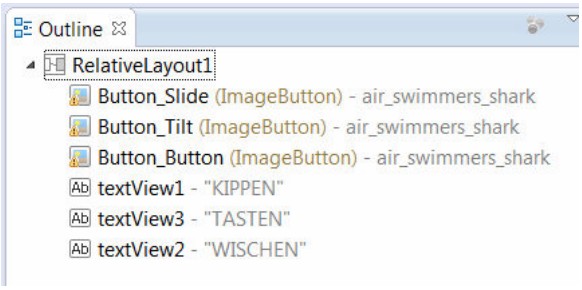



**Abbildung 5**
Komponenten des Layouts der Startseite

Die Buttons werden je nach Funktion mit einem String beschriftet.
Nach Erstellen der Buttons befindet sich für jeden Button ein Codeblock in der xml-Datei. Dieser wird jeweils um folgende Codezeile erweitert, damit der Benutzer diesen Button tatsächlich anklicken kann.

```
android:onClick="onButtonClick"
```

Außerdem wird ein passender Hintergrund gewählt, der das „AirSwimmer-Thema“ aufgreift
und die Buttons werden in ihrer Größe und Ausrichtung auf die Maßen eines Tablets ausgerichtet.

## Einbinden der Teilfunktionen

### *a) Layout und Source-Code einbinden*

Durch Erzeugen neuer xml-Dateien („activity_layout_buttons“, „activity_layout_wipe“ und „activity_layout_tilt“) im Unterordner „Layout“ und Einfügen des bereits vorhandenen Codes für die Modi, sind die Layouts der Unterfunktionen nun Bestandteil des Hauptprogramms.

Dasselbe Vorgehen findet auch für den Source-Code im Unterordner „src“ statt, mit neuen Java-Dateien („Activity_Buttons“, „Activity_Wipe“ und „Activity_Tilt“).
Diese Funktionen müssen nun im AndroidManifest unter „Application“ als Knoten hinzugefügt werden:

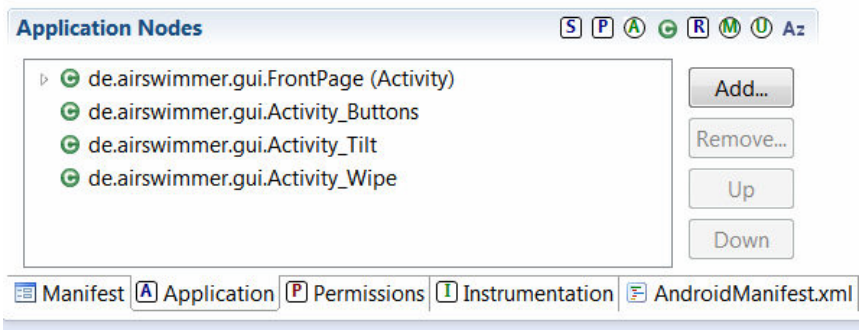



**Abbildung 6**

### *b) Funktion programmieren*

Da durch den Buttonklick zu der jeweiligen Funktion gewechselt werden soll, wird der Source-Code der Startseite um switch-case Anweisungen erweitert:

```
// What will happen when you click on a button
public void onButtonClick(View view){
	switch(view.getId()){
		case R.id.Button_Buttons:
			startActivity(new Intent(this, Activity_Buttons.class));
			break;
		case R.id.Button_Tilt:
			startActivity(new Intent(this, Activity_Tilt.class));
			break;
		case R.id.Button_Wipe:
			startActivity(new Intent(this, Activity_Wipe.class));
		default:;
}
```

Insgesamt muss beachtet werden, dass alle Namenskonflikte aufgelöst werden, alle zusätzlich genutzten Pakete ebenfalls implementiert werden und alle eingefügten Graphiken in das neue Projekt eingepflegt werden.

### *Ergebnis*

Der Benutzer gelangt mit Hilfe dieser App nun über die Startseite hin zu den Teilfunktionen (Tasten, Wischen, Kippen) und kann somit den AirSwimmer steuern.

## 8.3 Steuerung mittels Buttons

### *Ziel:*

Ziel ist es, eine Oberfläche mit Buttons zu erstellen. Diese soll vier Buttons zur Steuerung enthalten. Somit wird die Steuerungsarten rechts, links, oben und unten realisiert. Auch soll diese Oberfläche die Fernbedienung des AirSwimmers ersetzten.
Zu Beginn werden zwei Ausführungen erstellt: Einerseits mit Buttons mit Beschriftung(Variante A) und andererseits mit Buttons mit Bildern (sogenannten ImageButtons) (Variante B). Als endgültige Version der Buttonssteurung wird Variante B gewählt.

### *8.3.1 Variante A (erste Testversion)*

*von Anja Hafner*

### *Vorgehensweise:*

Die Oberfläche des Projektes „Android Fernsteuerung des AirSwimmers“ wird in der Entwicklungsumgebung „eclipse“ mit Add-Ons für die Android-Programmierung entwickelt.

Zu Beginn wird ein neues „Android Application Project“ in eclipse angelegt (zu finden unter File → New → Other... → Android → Android Application Project).
Die „main_activity“, die nach der Erstellung des Projekts erscheint, zeigt eine Darstellung eines Handybildschirms mit dem Projektnamen als Überschrift und „Hello World“ als Text im Bildschirm an.
Um nun den ersten Entwurf mit vier einfachen Knöpfen mit Beschriftung zu erzeugen, wird aus der Palette, die sich neben der Darstellung der zukünftigen App befindet, im Abschnitt „Form Widgets“ der Button ausgewählt.
Die Beschriftungen der Buttons werden zunächst in „strings.xml“ definiert (res → values → strings.xml). Für die Initialisierung wird der Button „Add...“ gedrückt. Im erscheinenden Fenster wählt man den Typ „String“ aus und gibt als Namen den Namen des Strings ein, mit dem er angesprochen werden soll. In Value wird die zukünftige Beschriftung (zum Beispiel: „up“) definiert.
Anschließend wird der Button viermal mit der Maus in den Handybildschirm gezogen. Mit dem Befehl „android:text=“@string/*““ wird der String dem Button zugeordnet, wobei der Stern für den Namen des definierten Strings steht.
Für eine einfachere Benutzung der Knöpfe während dem Programmieren, werden die Namen der vier Buttons in der „main_activity.xml“ in „button_up“, „button_down“, „button_left“ und „button_right“ geändert.

Um die Abstände der Knöpfe zueinander anzupassen, markiert man den Button, den man verschieben möchte. Danach wird in der Symbolleiste des „activity_main.xml“ Graphical Layout-Fensters der Button, der einen gestichelten Rahmen zeigt („Change Margins...“), angeklickt. In dem erscheinenden Fenster trägt man die gewünschte Position ein. Dabei ist zu beachten, dass die Abstände mit der Endung „dp“ (Density Independend Pixel) angegeben werden müssen, damit die Oberfläche richtig angezeigt wird.

Nun kann die App getestet werden. Dazu wird sie zunächst gespeichert. Anschließend folgt ein rechts-Klick auf den Projektordner und unter „Run As“ wird die Option „Android Application“ ausgewählt.
Besitzt man ein Android-fähiges Handy oder Tablett, wird dieses angeschlossen und von eclipse erkannt. Sollte es nicht erkannt werden, müssen zur Erkennung zusätzliche Treiber für das Handy / Tablett installiert werden. Die .apk-Datei der App wird danach auf das Gerät installiert und es kann direkt auf dem Handy / Tablett ausgeführt werden. Eine .apk-Datei ist eine Art Ordner, die alle erforderliche Daten für die Ausführung der App enthält.
Soll die App auf dem Computer ausgeführt werden, ist ein „Virtual Device“ (ein virtuelles Handy, auf das die App gespielt wird) nötig.
Dieses wird im „Android Virtual Device Manager“ erzeugt. Dorthin gelangt man über „Window → Android Virtual Device Manager“ oder in der Symbolleiste über das Handysymbol. Per Mausklick öffnet sich ein Fenster. In diesem Fenster drückt man den Button „New...“ und ein neues Virtual Device kann erstellt werden.
Da die App für Tabletts gedacht ist, wird als darzustellendes Gerät „10.1“ WXGA (Tablet) (1280 x 800: mpdi)“ ausgewählt. Als Target (die minimale Android-Version, auf der die App laufen wird) wird die API „Android 2.2 – API Level 8“ festgelegt.
Nachdem das „Android Virtual Device“ gestartet ist, kann die App, die „AirSwimmer genannt wird, gestartet und die Steuerung mit Buttons getestet werden.

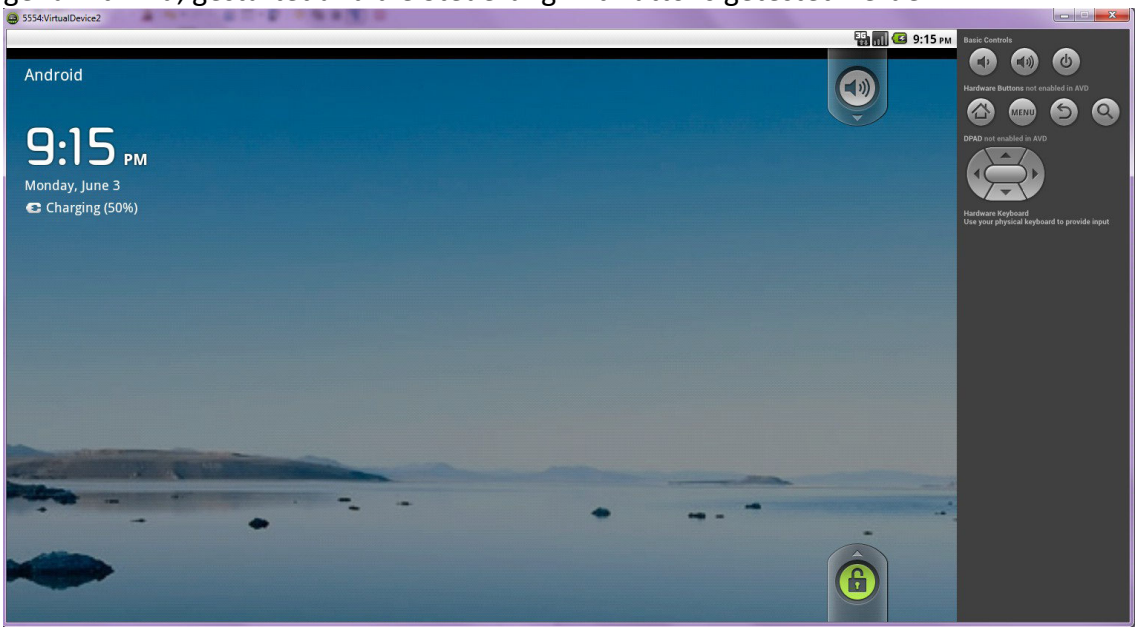



**Abbildung 7**
Android Virtual Device

### *8.3.2 Variante B (ImageButtons)*

*Von Belgüzar Kocak*

## *Vorgehen*

Zuerst stellt uns eclipse beim Erstellen von Projekten ein Standard Layout zur Verfügung.

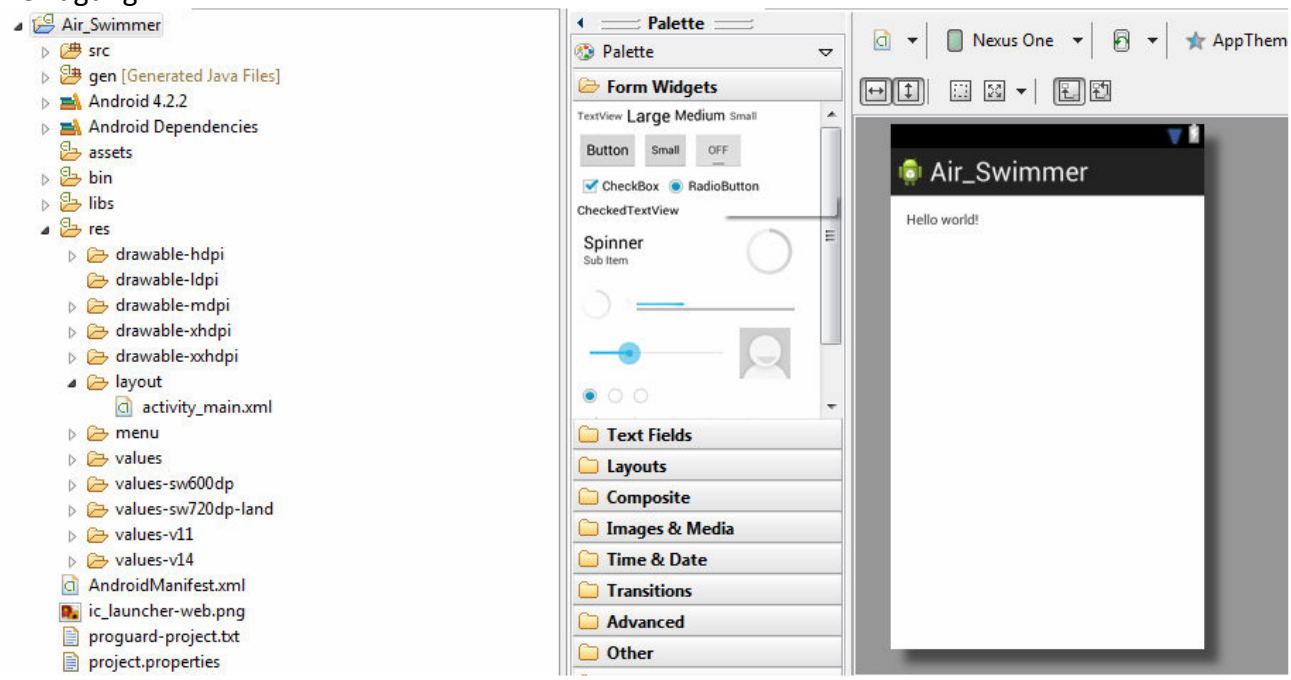



**Abbildung 8**

Dieses enthält folgende XML-Struktur:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        tools:context=".MainActivity" >

        <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

Zu allererst wird dabei die TextView entfernt. Unter dem Punkt Image & Media werden 4 ImageButtons in das Layout gezogen und durch einen Doppelklick gelangt man direkt in die XML-Datei. Anhand dieser Zeile in der XML-Datei wird die entsprechende Grafik in den Button gesetzt.

```
android:src="@drawable/up"
```

Auch wird der Hintergrund des Layouts geändert indem beim RelativeLayout die folgende Zeile hinzugefügt wurde:

```
android:background="@drawable/ic_sky"
```

Diese Grafik stellt ein Wolkenhintergrund dar und wird vom drawable-hdpi aufgerufen. Um die Buttons transparent zu gestalten, wird folgender Code editiert.

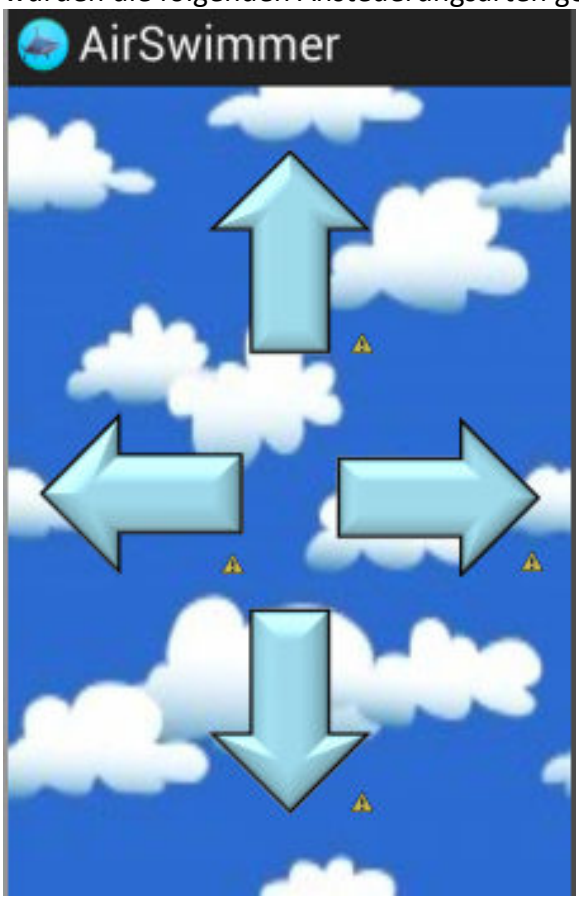
```
android:background="#00000000"
```

Um auch später die Bewegungssteuerung für die Buttons schnell und einfach zu implementieren wurde für jeden Button ein ButtonListener eingebaut. Ein Listener überprüft wann und ob ein Button gedrückt wurde. Falls ein Button gedrückt wurde kann man eine Funktion zur Steuerung implementieren.

```
button_up = (ImageButton) findViewById(R.id.imageButtonUp);
            button_up.setOnTouchListener(this);
```

### *Ergebnis*

Durch diese Vorgehensweise wurde ein Layout mit Buttons-Steuerung erstellt. Somit wurde der Start zur Oberflächenprogrammierung gegeben. Auf diese Oberflächen wurden die folgenden Ansteuerungsarten gebaut.




## *8.3.3 Reaktion auf Druck und visueller Effekt*

*Von Caroline Pilot*

### *Zielsetzung*

Da die Tasten bei einer einfachen „if-Abfrage“ in den „Activtiy_Button“ Klassen nur auf ein Loslassen der Tasten vom Benutzer reagieren, muss hier, vor allem aus Logikgründen, eine Reaktion beim Drücken der Tasten implementierten werden. Zudem soll ein visueller Effekt dem Benutzer deutlich machen, wann ein Signal gesendet wird und wann nicht.

## *Vorüberlegung*

Die Reaktion beim Drücken bedarf keiner weiteren Überlegung.

Der visuelle Effekt jedoch, kann verschieden implementiert werden:

- Ein Text mit der Aufschrift der jeweiligen Richtung besteht bereits, dieser wird allerdings nur für eine kurze Zeitdauer angezeigt
- Ein farblicher Rahmen um die Tasten beim Druck
- Die gedrückte Taste an sich, könnte die Farbe verändern

Ergebnis: Der Text soll weiterhin kurz angezeigt werden, dies soll der reinen Information für den Benutzer dienen und außerdem wird so sichergegangen, dass auch tatsächlich die Signale für beispielsweise die Aufwärtsbewegung ausgegeben werden.
Auch die Taste an sich soll keine Farbänderung erfahren, denn dies könnte möglicherweise nicht direkt gesehen werden, wenn der Benutzer seinen Finger darauf legt.
So bleibt die gebräuchliche Möglichkeit, einen dickeren Rahmen um jede Taste zu legen und diesen beim Druck einzufärben und beim Loslassen verblassen zu lassen.

## *Vorgehensweise*
## *Reaktion beim Druck der Tasten*

Durch eine einfache Erweiterung der „If-Anweisung“ in der Base-Class der Buttons kann die Reaktion nur beim Druck erreicht werden.

```
if (event.getAction() == MotionEvent.ACTION_DOWN){…}
bzw.
if(event.getAction() == MotionEvent.ACTION_UP){…}
```

## Visueller Effekt

Mit dem Befehl lässt sich der Hintergrund der Buttons einfärben:

```
button_up.setBackgroundColor(Color.BLUE);
```

Dies setzt den Hintergrund des „button_up“ Bildes auf Blau. Diese Farbe wurde gewählt, weil man sie auf jedem Hintergrundbild gut sehen kann und auch zu den Farben des AirSwimmers passt.
Da dies erst geschehen soll, sobald der Nutzer die Taste drückt (`ACTION_DOWN`), wird dieser Befehl in die erste „if-Anweisung“ eingefügt und ein passender Befehl (`Color.TRANSPARENT`), welcher den Hintergrund der Taste wieder farblich zurücksetzt, in die zweite “if- Anweisung“ eingefügt.

## *Ergebnis*

Der Benutzer erhält nun visuelle Rückmeldung, dass der Button gedrückt wird, diese Farbänderung hält die ganze Zeit an, bis der Benutzer den Finger wieder hebt.
In genau dieser Zeit sollen auch die Signale an den Fisch gesendet werden.

## 8.4 Steuerung durch Wischen

*von Anja Hafner*

### *Ziel:*

Ziel ist es, eine Ansteuerung des AirSwimmers per Drag and Drop Prinzip zu realisieren. Dabei soll ein Bild, das in dem „drawable“-Ordner im „res“-Ordner des Projektes gespeichert ist, per Drag and Drop Prinzip über den Handybildschirm bewegt werden. Unter Drag and Drop versteht man, dass mit dem Finger das Bild des Fisches in der App berührt wird und dieses in die gewünschte Position gezogen wird. Der AirSwimmer ahmt anschließend die Bewegung des Fisches in der App nach.

### *Vorgehensweise:*

Für die Steuerung der App durch Wischen wird ein neues Projekt mit dem Namen „SlideApp“ in eclipse erstellt.

Zunächst wird im „Graphical Layout“ der „activity_layout_slide.xml“ die Oberfläche erstellt. Dazu wird die Palette eingesetzt. In dem Ordner „Images & Media“ der Pallette wird die ImageView in den Handybildschirm gezogen. In dem erscheinenden Dialogfenster wählt man in den Projektressourcen das Bild des Fisches aus, das eingefügt werden soll.
Alternativ kann das Bild auch als „ImageView“ direkt in der „activity_layout_slide.xml“ Datei mit dem Befehl „android:src=“@drawable/fish““ eingefügt werden (hier ist der Name des Bildes: fish).

Wird das Bild direkt in der xml-Datei eingefügt, befindet sich das Bild im oberen linken Rand des Bildschirms. Wird es hingegen per Hand in den Bildschirm gezogen, ist die Position frei wählbar. Das zu bewegende Bild soll zentriert angezeigt werden. Dazu wird im ImageView der Befehl „`android:layout_centerInParent=“true““` hinzugefügt. In diesem Fall ist der Parent die View, in dem das Bild nun zentriert plaziert ist.

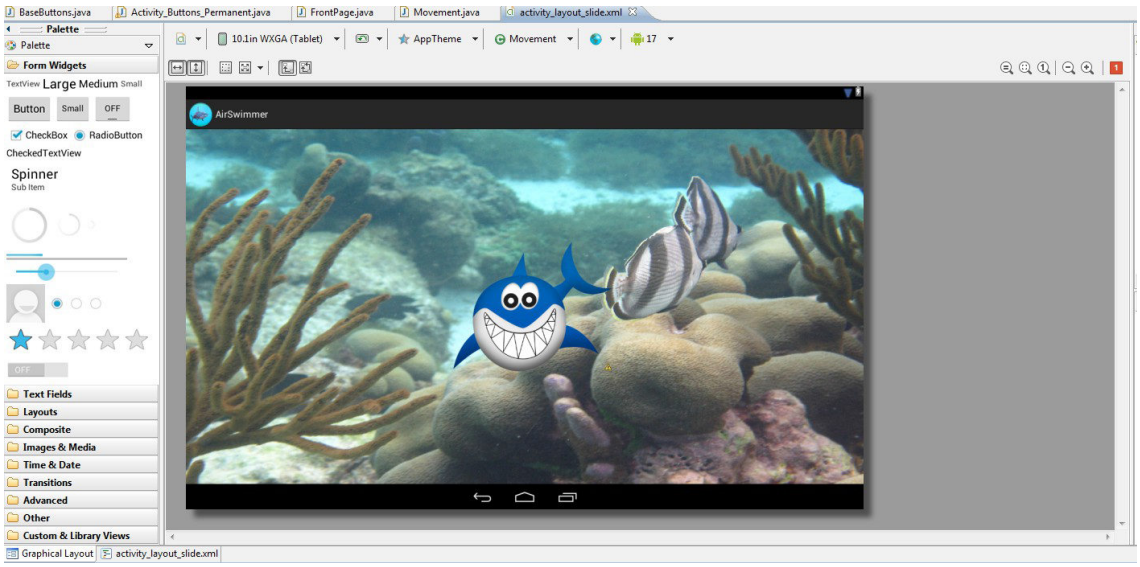



**Abbildung 9**
Bild des Graphical Layouts mit der Palette zur Erstellung der Oberfläche

Der Befehl „centerInParent“ existiert nur im „Relative Layout“, darum wird dieses Layout gewählt.
Um auch den Hintergrund dem Thema entsprechend anzupassen, wird hier ein Bild mit dem Befehl: „android:background=“@drawable/sea““ eingefügt. Dieses Bild befindet sich ebenfalls im „drawable“-Ordner des Projektes.

BaseSlide.java | *activity_layout_slide.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/sea"
    android:id="@+id/layout"
    tools:context=".MainActivity" >

    <ImageView
        android:id="@+id/img"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:src="@drawable/fish" />

</RelativeLayout>
```

**Abbildung 10**
Screenshot der „activity_layout_slide.xml“ Datei

Die Bewegung des Fisches wird in der „BaseSlide.java“-Datei implementiert. Sie wird mit Hilfe eines sogenannten „onTouch“-Listeners realisiert. Dieser ist in zwei Ausführungen im Programm vorhanden: Einmal ein „onTouch“-Listener für die ViewGroup (eine View, die andere Views beinhalten kann, in diesem Fall das Bild des Fisches) und für das Bild, welches bewegt werden soll.

Im „onTouch“-Listener des Bildes wird mit „MotionEvent.ACTION_DOWN“ überprüft, ob der Benutzer das Touch-Display berührt. Ist dies der Fall, wird ein neues Offset ermittelt. Mit diesem Offset wird im „onTouch“-Listener der ViewGroup die aktuelle Postion des Bildes berechnet.
Findet keine Berührung statt, passiert nichts.

```
        ImageView img = (ImageView) findViewById(R.id.img); // fish image is initialized
        img.setOnTouchListener(new View.OnTouchListener() { // onTouchListener for fish

            @Override
            public boolean onTouch(View view, MotionEvent event) {
                switch (event.getActionMasked()) {
                case MotionEvent.ACTION_DOWN: // if someone touches the fish
                    offset_x = (int) event.getX(); // calculate new offset
                    offset_y = (int) event.getY();
                    selected_item = view; // view is selected
                    break;

                default: // fish is not touched
                    break;
                }
                return false;
            }
        });
    }
```

**Abbildung 11**

Der „onTouch“-Listener der ViewGroup wird in der Methode „onCreate“ gezetzt. Der Listener wird in dieser Methode aufgerufen, damit der Fisch nicht bei jeder beliebigen Berührung an die berührte Stelle springt.
In diesem Listener der wird ebenfalls überprüft, ob eine Bewegung innerhalb des Bildes während einer Berührung stattfindet. Hierbei werden die Koordinaten des Bildes mit der Methode „getLocationOnScreen“ ermittelt. Diese Koordinaten werden von den Koordinaten des gedrückten Punktes abgezogen. Anschließend wird überpüft, ob die Bewegung innerhalb eines bestimmten Toleranzbereiches liegt.

```
(if (MotionEvent.ACTION_MOVE >= imageX – 70 &&
MotionEvent.ACTION_MOVE < imageX + 70 ||
MotionEvent.ACTION_MOVE >= imageY – 30 &&
MotionEvent.ACTION_MOVE< imageY + 30).
```

Wird das Bild berührt, wird die Methode zum Bewegen des Fisches („moveImage“) aufgerufen. Die Methode „moveImage“ ermöglicht die Bewegung des Fischbildes. Wird das Bild bewegt, wird die neue Position des Fisches in der x- und der y-Achse jeweils mit dem im „onTouch“-Listener ermittelten Offsets berechnet. Um zu verhindern, dass der Fisch an allen vier Seiten aus dem Bildschirm herausgezogen werden kann, wird mit „getWindowManager().getDefaultDisplay().getWith“ und „getWindowManager().getDefaultDisplay().getHeight“ die Breite und Höhe des Displays ermittelt. Mit diesen Werten ist es möglich, mit einer einfachen if-Abfrage zu ermitteln, ob der Fisch zu weit nach links/rechts oder oben/unten gezogen wurde. Wird das Bild nicht bewegt, passiert nichts.
Es fällt auf, dass der Fisch bei Annäherung an den rechten und unteren Rand der App kleiner wird.
Durch Ersetzen von „`android:layout_width=“fill_parent““` und „`android:layout_height=“fill_parent““` durch „`android:layout_width=“match_parent““` und „`android:layout_height=“match_parent““` wird das Verkleinern des Fisches verhindert.

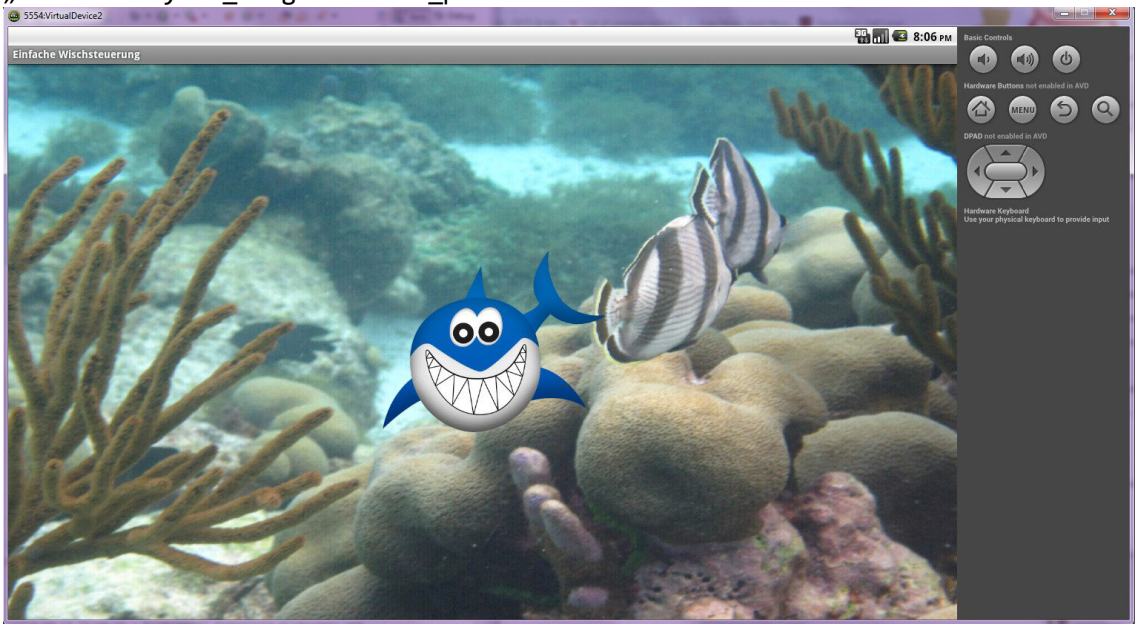



**Abbildung 12**
Bild der SlideApp

## 8.5 Steuerung durch Kippen

## 8.6 Menü

### *Steuerungsoberflächen*

*von Sabine Kressierer*

### Zielsetzung

In den verschiedenen Oberflächen zum Steuern des AirSwimmers soll ein Menü eingefügt werden, dass das Umschalten zwischen den einzelnen Steuerungsmodi sowie eine Auswahl des Hintergrund ermöglicht.

### Vorgehensweise

Ich Ordner „res“ wurde ein Ordner „menu“ angelegt, der die XML-Dateien zum darstellen der Menüs enthält. Da die Menüpunkte der verschiedenen Oberflächen zum Steuern des AirSwimmers gleich sind, wurde für sie nur eine XML-Datei „control.xml“ erstellt. Diese enthält die Menüpunkte mit allen Unterpunkten wie hier für den Menüpunkt „Hintergrund ändern“:

```
        <item android:title="@string/change_background"
android:id="@+id/change_background">
 <!-- Submenu with different background pictures -->
            <menu>
                <group android:id="@+id/submenu_changeBackground" >
                <item
                    android:id="@+id/water"
                    android:title="@string/water_picture"/>

                <item
                    android:id="@+id/sky"
                    android:title="@string/sky_picture"/>
                <item android:id="@+id/th_picture"
android:title="@string/th_picture"/>
                </group>
            </menu>
        </item>
```

In einer Basisklasse „BaseActivity.java“, von der alle Activities zum Steuern des Airswimmers ableiten, wird die Logik für das Menü implementiert. Das Erzeugen des richtigen Menüs erfolgt durch überschreiben der onCreateOptionsMenu-Methode, in der den Activities mit Hilfe des Befehls `getMenuInflater().inflate(R.menu.control, menu)` das Menü der Datei „control.xml“ zugewiesen wird Die Reaktion auf die Auswahl der Menüpunkte erfolgt in der Funktion `onOptionsItemSelected`, die ebenfalls in der Basisklasse überschrieben wird. Auf diese Weise werden die nachfolgenden Menüpunkte realisiert.

### a) Hintergrund ändern

*von Sabine Kressierer*

### Zweck

Dem Benutzer soll die Möglichkeit zur Verfügung stehen zwischen verschiedenen Hintergründen für die Steuerungsoberflächen zu wählen.

### Umsetzung

Der Menüpunkt „change_background“ enthält ein Untermenü mit den verschiedenen Hintergrundbildern. Wird einer dieser Punkte ausgewählt wird mittels `background.setBackgroundDrawable(source.getDrawable(R.drawable.ic_sky));` das entsprechende Bild (in diesem Fall ein Himmel) als Hintergrund verwendet. „background“ ist in diesem Fall das Layout der Activity. Zur einheitlichen Verwendung hat das Layout bei allen Activities die Id „layout“.
Besonderheit der Oberfläche Kippen: Da beim Kippen der Hintergrund anstatt einem einfachen Drawable eine BitMap ist wird hier das ändern des Hintergrunds überschrieben. Hier wird der Hintergrund geändert, indem das gewünschte Bild in einer Variablen „background“ des Threads zum Zeichnen der Oberfläche gespeichert wird. Das dort gespeicherte Bild wird in der run()-Methode des Threads gezeichnet.

## b) Modus auswählen

*von Sabine Kressierer*

### Zweck

Um die Bedienung zu vereinfachen soll man nicht nur über die Startseite sondern auch über das Menü zwischen den einzelnen Steuerungsmodi (Kippen, Wischen, Tasten) wechseln können

### Umsetzung

Der Menüpunkt „change_mode“ enthält ein Untermenü mit den Namen der einzelnen Steuerungsarten. Wird ein Punkt ausgewählt, wird mittels `startActivity(new Intent(this, Activity_Buttons.class));` die entsprechende Activity gestartet. Um ein erneutes Auswähler der bereits laufenden Activity zu verhindern, wird bei Erzeugung des Menüs, der entsprechende Menüpukt ermittelt und unsichtbar gemacht.

Besonderheit der Oberfläche Kippen: Da in dieser Oberfläche ein Thread verwendet wird, muss dieser bei Verlassen der Oberfläche gestoppt werden, da sonst auf einigen Geräten eine Fehlermeldung auftritt. Dies geschieht durch überschreiben der Methode `onOptionsItemSelected` in der entsprechenden Activities. Damit bei Rückkehr zur Oberfläche durch Betätigen des Zurück-Knopfes die im Thread realisierte Animation der Oberfläche wieder stattfindet, wird in der Methode `onResume()` dieser Thread wieder gestartet.

## c) Steuerungsart ändern

*von Caroline Pilot*

### *Zielsetzung*

Der Benutzer soll die Möglichkeit haben das Versenden von Signalen an den AirSwimmer über verschiedene Steuerungsarten zu realisieren.

Zum einen soll genau ein Signal (ein einziger kompletter Bewegungsbefehl) je Tastendruck ausgeführt werden (→ Einfach), zum anderen soll eine permanente Bewegung mit nur einem Klick ebenfalls möglich sein(→Permanent).
Diese Steuerungsart soll sowohl im Tastenmodus als auch im Wisch- und Kippmodus möglich sein.
Im Folgenden wird das Einfügen des Menüpunkts für die Tastensteuerung beschrieben, da die anderen Steuerungsmodi analog ablaufen.

## Vorüberlegung

Da es sich zwar bei der Optik jeweils um die gleichen Klassen handelt (Einfache Tastensteuerung und Permanente Tastensteuerung), jedoch die Logik dahinter anders ist, muss die vorhandene Klasse geklont werden, damit ein Menüpunkt später dorthin verweisen kann.
Dieser Menüpunkt („Steuerungsart ändern“) muss als ein weiterer in die Liste eingetragen werden, wobei ein Untermenü zum Wechseln zwischen „Einfach“ und „Permanent“ ebenfalls implementiert werden muss.
Damit der Benutzer auch weiß, in welchem Modus er sich befindet, muss der Android Action Bar Text (Wird am oberen Bildschirmrand während jeder Aktivität angezeigt) angepasst werden.

## Vorgehensweise

### Erstellen des Menüpunkts „Steuerungsart ändern“

In der bereits vorhandenen *control.xml* Datei werden im Layout folgende Punkte hinzugefügt:

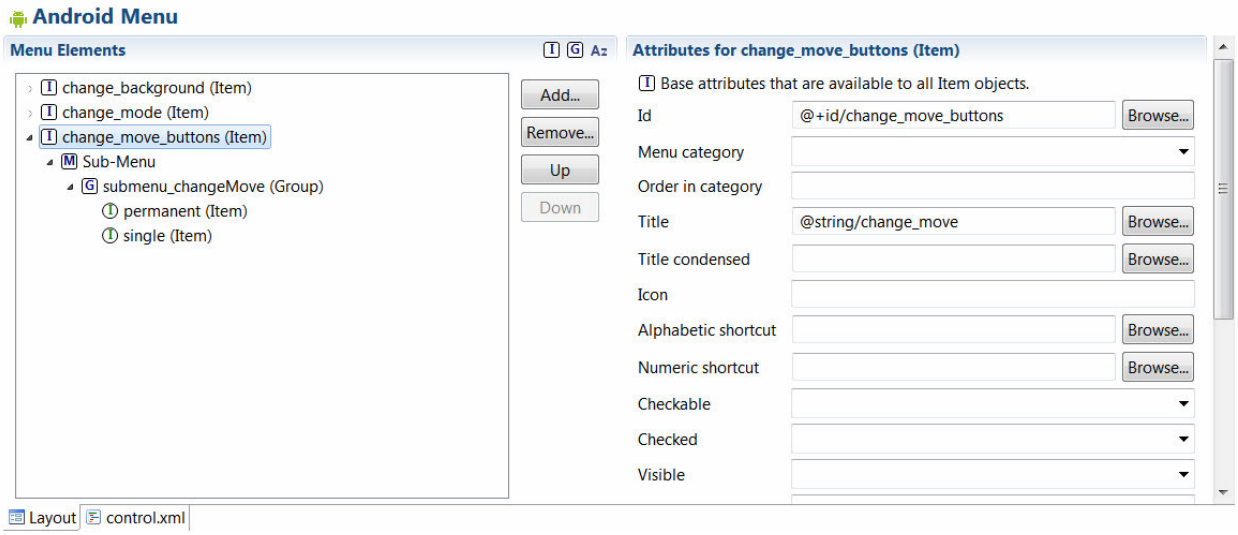



**Abbildung 13**
Menüpunkt für Steuerungsmodus hinzugefügt

Passende „id‘s“ und „strings“ werden eingetragen. So findet sich unter *@string/change_move* der Text „Steuerungsart ändern“.
Hiermit hat man nun im Menü diesen Punkt, sollte dieser durch Klicken aufgerufen werden, so öffnet sich ein Untermenü mit den Unterpunkten „Einfach“ und „Permanent“.

Um auch die Logik dahinter zu realisieren wird in der BaseActivity Klasse strukturiert abgefragt, um welche Klasse es sich aktuell handelt:

```
if (currentActivityName.equals(getResources().getString(
    R.string.title_activity_activity__buttons))){…}
```

und ob der „Steuerungsart ändern“ Menüpunkt aufgerufen wurde:

```
if (item.getGroupId() == R.id.submenu_changeMove){…}
```

Sollte dies der Fall sein, so wird die Nutzereingabe über `switch (item.getItemId())` eingelesen und zur jeweils zugehörigen Klasse gesprungen:

```
case R.id.permanent:
     startActivity(new Intent(this,Activity_Buttons_Permanent.class));
      return true;
case R.id.single:
      startActivity(new Intent(this, Activity_Buttons.class));
      return true;
default:
      return false;
```

Ändern des Android Action Bar Texts
Der Android Action Bar Text kann im *AndroidManifest.xml* verändert werden.
Da hier ohnehin schon zu jeder Activity ein Codeblock mit einem Label (default: Namen der Klasse) generiert wurde kann dieser mit gebräuchlicheren und verständlicheren Namen ersetzt werden.

```
<activity
            android:name="de.airswimmer.gui.Activity_Buttons"
            android:label="Einfache Tastensteuerung"
            android:screenOrientation="Landscape" >
</activity>
```

Hier wird unabhängig von definierten Strings gearbeitet und macht ein Modifizieren und Erweitern einfach.

## Ergebnis

Der Benutzer gelangt nun mit Hilfe eines „Steuerungsart ändern“ Menüpunkts zu der Auswahl, ob der aktuelle Modus in einer Permanenten oder Einfachen Steuerungsart bedient werden soll.
Optisch unterscheiden sie sich kaum, jedoch ist die Logik dahinter jeweils eine andere. Für eine bessere Orientierung sieht der Benutzer am oberen Bildschirmrand nun in welchem Modus und welcher Steuerungsart er sich befindet.

## *Startseite*

## Abfrage Aux-Anschluss

*Von Sabine Kressierer*

**Zielsetzung**
Die Ausrichtung der Oberfläche soll so erfolgen, dass sich der Aux-Anschluss immer an der oberen Seite des Tablets befindet. Somit kann der IrDroid Sender direkt auf den AirSwimmer gerichtet werden.

Umsetzung
Bei erstmaligem Öffnen der App wird zuerst ein Dialog geöffnet, in dem angegeben werden muss wo sich der Aux-Anschluss befindet. Dies geschieht mit Hilfe eines AlertDialog, der in der onCreate()-Methode der FrontPage-Activity erzeugt und angezeigt wird. Die Reaktion auf die Auswahl eines Punktes erfolgt mittels onClick-Listener, in dessen onClick-Methode der Index des gewählten Punktes bekannt ist. Entsprechend dieser Auswahl wird die ActivityInfo-Konstante für das entsprechende Layout gespeichert.

```
if (item == 0) { //get value for requested screen orientation
	layout = ActivityInfo.SCREEN_ORIENTATION_PORTRAIT;
} else if (item == 1) {
	layout = ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE;
}
```
[1]

Da sich die Position des Aux-Anschlusses nicht verändert, wird dieser Wert dauerhaft gespeichert. Dazu werden shared-preferences verwendet, die es ermöglichen innerhalb der gesamten App auf den Wert zuzugreifen. Mit `Preferences = getSharedPreferences("AirSwimmerPrefs",Context.MODE_WORLD_READABLE);` werden die preferences in der Datei „AirSwimmerPrefs“ ermittelt. In diesen Preferences wird dann mit dem `ShardPreferences.Editor` der int-Wert zum Schlüssel „layout“ auf die oben ermittelte Konstante gesetzt. Am Ende wird in der onCreate-Methode der Startseite mittels `setRequestedOrientation(layout);` die Orientierung entsprechend verändert. Mit der selben Funktion wird in der onCreate-Methode der „BaseActivity“ die Orientierung gesetzt. Um dort den in den Preferences gespeicherten Wert zu erhalten, werden wie oben die Preferences ermittelt und anschließend `preferences.getInt("layout", -1)` aufgerufen.
Bei erneutem Starten der App wird der Dialog nur noch gestartet, wenn in den Preferences kein Wert gespeichert ist.

## Menü der Startseite
*Von Sabine Kressierer*

**Zweck**
Die Ausrichtung der App soll nachträglich veränderbar sein.
**Umsetzung**
In der Datei „start.xml“ wurde ein Menüpunkt „change_layout_orientation“ eingefügt, mit einem Untermenü für die beiden Orientierungsmöglichkeiten. Das Erzeugen des Menüs erfolgt in der FrontPage-Aktivity analog zum control-Menü. Bei Auswahl einer Orientierung im Untermenü, wird dieser Wert wie beim AlertDialog in den Preferences gespeichert und die Ausrichtung der FrontPage angepasst.

[1] 0=Aux-Anschluss an kurzer Seite→Portrait / 1= Aux-Anschluss an langer Seite → Landscape

# 9 Graphische Gestaltung der Oberfläche

## 9.1 Konzept

*Von Caroline Pilot*

### *Zielsetzung*

Ziel eines Prototyps für das Design der Oberfläche[2] ist es, der Gruppe einen ersten visuellen Eindruck für die App zu geben. So bekommt das gemeinsame Ziel des Teams ein Gesicht, auf das motivierter hingearbeitet werden kann.
Außerdem bieten Modelle eine bessere Diskussionsgrundlage für Wünsche, Ideen und Verbesserungsvorschläge. So kann die Oberflächenentwicklung mit konkreten Designs ausgeführt werden.

### *Vorüberlegung*

Zunächst muss ein Graphikprogramm gefunden werden, das den Anforderungen entspricht.
Hierbei ist folgendes zu beachten:

- Vielerlei Möglichkeiten Bilder zu bearbeiten (Kontraste, Layer, Transparenz, …)
- Erstellen von Graphiken in gängigen Bildformaten
- Einfache Bedienung, damit keine lange Einarbeitungszeit nötig ist
- Kostengünstig, am besten ein Freeware Download Programm

Des Weiteren muss der Kreativität bei folgenden Punkten freien Lauf gelassen werden:

- Wie soll der Hintergrund aussehen?
- Wie soll das Objekt dargestellt werden?
- Welche Besonderheiten benötigen die einzelnen Benutzermodi?

[2] Hiermit sind im Kapitel 5.1 „Konzeptentwicklung für das Oberflächendesign“ die unterschiedlichen Benutzermodi zum Steuern des AirSwimmers gemeint, nicht die Startseite der App

## *Vorgehensweise*

Unter http://www.chip.de/downloads/PhotoFiltre_13012070.html ist das Freeware Graphikprogramm als Download verfügbar. Es entspricht unseren Anforderungen und enthält zudem sehr gute Nutzerrezensionen.

Aus dem Brainstorming zum Thema Oberflächendesign ergibt sich folgendes:

- Hintergrund:
  - Himmel
  - Unterwasserwelt
  - Stadt/Skyline
- Objekt
  - Bild des AirSwimmers
  - Bild eines Fisches
  - Bild eines Köders, dem der echte AirSwimmer „hinterher schwimmt“
- Besonderheiten einzelner Modi
  - Tastensteuerung
    - Tasten in gewöhnlicher Buttonform
    - Tasten in Form eines Fisches
    - Kontrastreiche Farben <–> Gedeckte Farben
  - Kippsteuerung
    - Angabe von Himmelsrichtungen (north, south, west, east)
    - Angabe von Bewegungsrichtung (up, down, left, right)
    - Keine Angabe
  - Wischsteuerung
    - Angabe von Himmelsrichtungen (north, south, west, east)
    - Angabe von Bewegungsrichtung (up, down, left, right)
    - Keine Angabe

## *Ergebnis*

Bilder zum Verwirklichen dieser Ideen und zum Erstellen von Designprototypen lassen sich im Internet finden oder mit dem Graphikprogramm erstellen.
In verschiedenen Kombinationen, umgesetzt mit dem Photofiltre7 ergeben sich folgende Entwürfe:

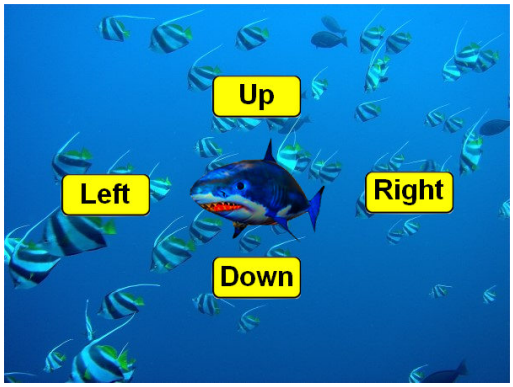



**Abbildung 14**:
Tastensteuerung;
Unterwasserwelt, Kontrastreiche Buttons


**Abbildung 15**
Tastensteuerung;
Himmel, Gedeckte Buttons


**Abbildung 16**
Kippsteuerung;
Unterwasserwelt, Angabe von Himmelsrichtungen

**Abbildung 17**
Kippsteuerung;
Skyline, ohne Angaben

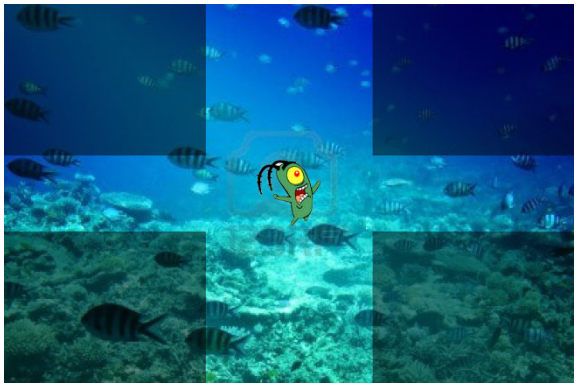


**Abbildung 18**
Wischsteuerung
Benutzergrenzen, für definierte Bewegungen, Köder als Objekt

**Abbildung 19**
Wischsteuerung
Benutzergrenzen, für definierte Bewegungen

Die Gruppendiskussion bringt hervor, dass es in der App einen Punkt für einen Wechsel des Hintergrunds zwischen Himmel und Unterwasserwelt geben wird, je nach Wunsch des Benutzers.

Des Weiteren werden nicht die Himmelsrichtungen im Kippmodus angegeben, denn dies könnte zu Verwirrungen führen, stattdessen werden keine  Angaben platziert, ebenso im Wischmodus.
Das Objekt in der Mitte soll ein Fisch sein.

### *Ausblick*

Da es sich bei den verwendeten Bildern um Eigentum Dritter handelt, könnte man sich durch dessen Verwendung rechtlich Probleme einhandeln. Aus diesem Grund ist es wünschenswert, eigene Graphiken, vielleicht auch dynamische, zu erstellen oder die Urheber schriftlich um eine Genehmigung zu bitten.

## 9.2 Launcher-Icon

*von Caroline Pilot*

### *Zielsetzung*

Um einen einheitlichen Auftritt der App zu ermöglichen, wird ein Launcher-Icon benötigt. Dieser repräsentiert die App auf dem Bildschirm und ist auch bei der Ausführung am Rand zu sehen.

### *Vorüberlegung*

Da bereits das Graphikprogramm PhotoFiltre vorhanden ist, kann dieses verwendet werden, um den Launcher-Icon zu erstellen.
Des Weiteren muss der Kreativität bei folgenden Punkten freien Lauf gelassen werden:
Welche Form soll der Icon bekommen?
Welches Objekt soll darauf zu sehen sein?
Welche Farben könnten passen?

### *Vorgehensweise*

Aus dem Brainstorming ergeben sich folgende Punkte:

<u>Form</u>
Rund, da es einer Wasserblase entspricht, in dem sich der AirSwimmer fortbewegt

<u>Objekt</u>
Da es eine App für den AirSwimmer sein soll, liegt es nahe, dass auch auf dem Icon der Fisch abgebildet ist. Somit wäre es eindeutig auf dem Desktop zu identifizieren

<u>Farben</u>
Die Farben müssen herausstechen, sodass die App nicht auf dem Desktop untergeht und man sie nicht sehen kann (Beispiel schwarz), dennoch müsste die Farbe zum Thema Wasser/Himmel passen

### *Ergebnis*

Das Bild des AirSwimmers lässt sich im Internet finden.
In verschiedenen Kombinationen mit Farben und Hintergründen, umgesetzt mit dem Photofiltre7 ergeben sich folgende Entwürfe:

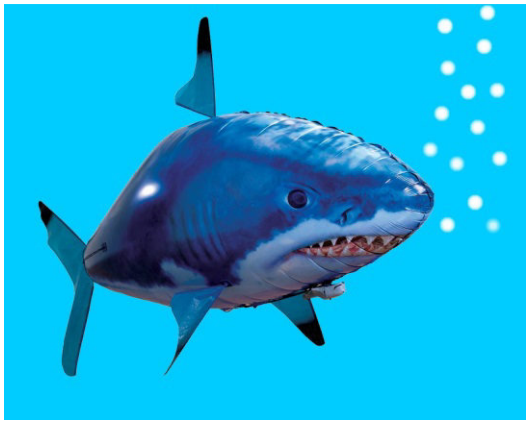


**Abbildung 21**
Launcher-Icon mit hellem Hintergrund

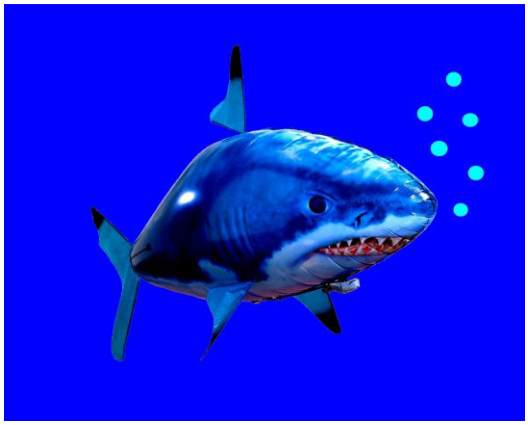


**Abbildung 21**
Launcher-Icon mit dunklem Hintergrund

Nach dem Testen auf dem Tablet wird festgestellt, dass der dunkle Hintergrund bei einem schwarzen Desktop untergeht, daher fiel die
Entscheidung auf einen runden Launcher-Icon mit hellem Hintergrund (Abbildung 19).

## 9.3 Animierter Hintergrund

## 9.4 Hintergrundbilder

## 9.5 Graphiken

### *9.5.1 Startseite (Buttons)*

*Von Belgüzar Kocak*

#### *Zielsetzung*

Die Aufgabe bestand darin, die Startseite zu verschönen und die Standard Buttons auszutauschen, da diese wie Balken über dem Bildschirm lagen. Aufgrund dessen waren größere Buttons und auch für die Startseite passende Buttons notwendig. Ziel war es Hai-Buttons für die Startseite zu erstellen.

#### *Vorgehen*

Bevor die Buttons ausgetauscht werden konnten, mussten sie in der XML-Datei in ImageButtons umgewandelt werden. In diese ImageButtons wurde eine Hai Grafik eingefügt und der Hintergrund der Buttons ausgeblendet damit diese die Form eines Haies haben.
In der XML-Datei wurden die normalen Buttons rausgelöscht und neue ImageButtons in die Oberfläche gezogen und die entsprechenden Images an die Buttons zugewiesen.
Vorab wurde die Grafik im res Ordner unter drawable-hdpi als air_swimmer_shark gespeichert.
Diese wurden dann mittels folgender Anweisung den einzelnen Buttons zugewiesen:

```
android:src="@drawable/air_swimmers_shark"
```

Der Hintergrund wurde transparent gestaltet um einen Hai-Förmigen Button zu erstellen

```
android:background="#00000000"
```

```
<ImageButton
    android:id="@+id/Button_Slide"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:background="#00000000"
    android:onClick="onButtonClick"
    android:scaleType="center"
    android:src="@drawable/air_swimmers_shark" />
```

Die Beschriftung der einzelnen Buttons wurde anhand einer TextView dargestellt. Dabei wurde die Schriftgröße, Schriftart und der Text der View Implementiert.

```
<TextView
        android:id="@+id/textView2"
        android:layout_width="1189dp"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/textView3"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/Button_Slide"
        android:layout_toRightOf="@+id/Button_Button"
        android:text="@string/button_wischen"
        android:textSize="30sp"
        android:textStyle="bold"
        android:typeface="sans" />
```

Auch wurde der Hintergrund der Startseite verändert.

```
  android:background="@drawable/frontpage_background"
```

Am Ende hatten die Buttons der Startseite folgendes Aussehen.


**Abbildung 22**

### *9.5.2 Steuerungsseiten (Buttons und Hai)*

*Von Belgüzar Kocak*

## *Zielsetzung*

Ziel war es für die Oberfläche Kippen und Wischen eine Grafik von einem Hai zu erstellen und auch Buttons für die Buttons Oberfläche zu zeichnen. Genutzt wurden hier folgende Tools: Microsoft Word, Paint.Net und Photoshop.

## *Vorgehen und Erstellen*

Verlangt wurde, dass der Hai direkt den Benutzer der Oberfläche anschaut. Aufgrund dessen wurden vorerst Zeichnungen per Hand erstellt genutzt wurden dabei ein einfaches Papier so wie Bleistift. Somit war der erste Schritt der Zeichnung getan. Jetzt war es wichtig die Zeichnung vom Papier auf den Bildschirm als Grafik zu verwirklichen. Dies gelang mit dem Photoshop Programm.
Somit entstand folgende Grafik:

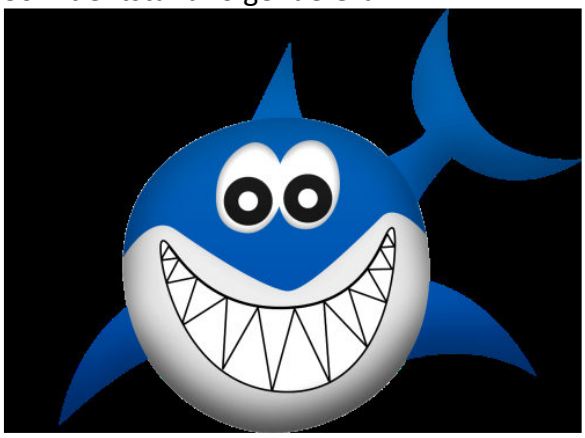


**Abbildung 23**

Die Pfeile wurden in Microsoft Word erstellt, die Pfeilform wurde in die passenden Größe skaliert. Nach der Skalierung wurde unter „Form formatieren“ in 3D-Format passende Werte eingegeben sowie Farben geändert. Anschließend in Paint.Net der Hintergrund mithilfe der Zauberstabfunktion entfernt.

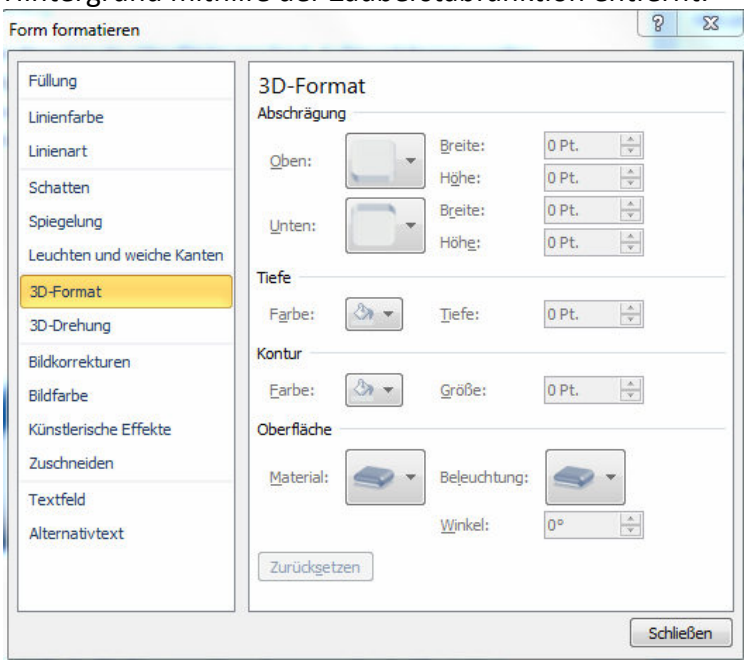



**Abbildung 24**

Nach dem Erstellen der Grafiken wurden diese in die Oberfläche eingebaut. Wie zum Beispiel in die Buttonoberfläche.

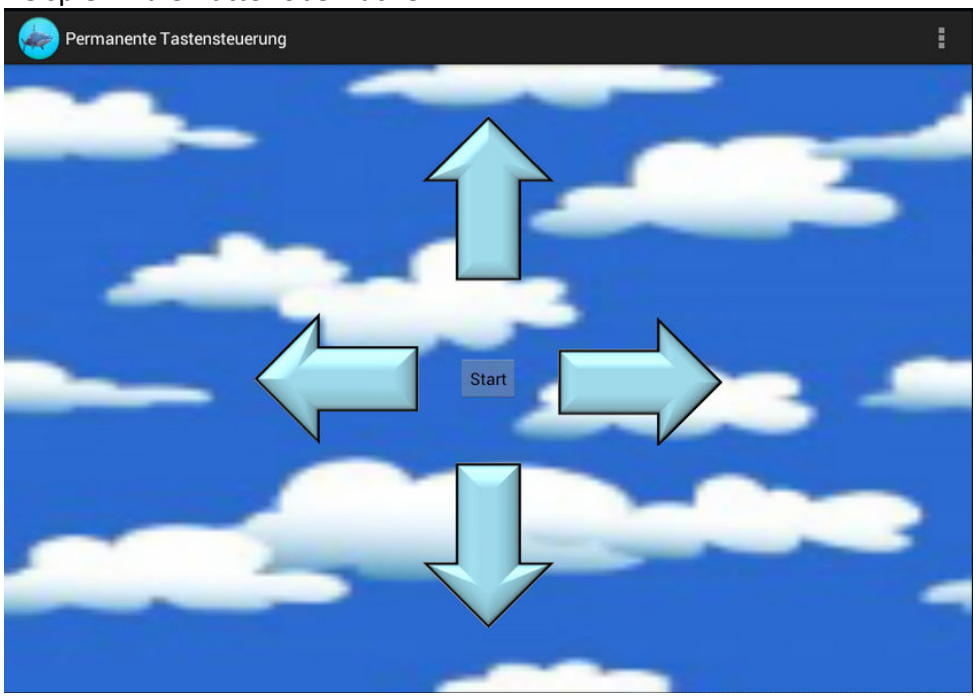



**Abbildung 25**