Animationen und Microinteractions im Multi-Screen Kontext

Dominick Madden

Studienarbeit

Studiengang Informatik

Fakultät für Informatik

Hochschule Mannheim

29.02.2016

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mannheim, 29.02.2016

Dominick Madden

Inhalt

1 Einleitung

In der Welt der mobilen Applikationen spielt Design und User Experience eine immer größere Rolle. Für viele Anwendungsfälle, wie das Stellen eines Weckers oder das Abspielen von Musik, gibt es hunderte von unterschiedlichen Apps. Betrachtet man die populärsten Apps in einem Gebiet, unterscheiden sie sich im Featureumfang oft nur geringfügig. Meist sind es die Apps mit der besten User Experience, die sich von der Masse abheben. Betrachtet man die populärste Weckerapplikation (über 10 Millionen Downloads mit einer Bewertung von 4.5 Sternen) im Google Play Store, Androids offiziellem Applikationsmarkt, sind die meistgenutzten Schlagworte in den Rezensionen: „easy to use“ (3307 mal), „best alarm app“ (1862 mal), „useful“ (1293 mal). „Great features“ kommt dagegen in nur 374 Rezensionen vor. **[AVG]**

„Easy to use“ und „useful“ lässt hier auf eine gute User Experience schließen. Ein Ansatz, eine gute UX zu designen, ist es, sich auf die kleinsten Details der Interaktionen zwischen Nutzern und der App zu konzentrieren. Diese *Microinteractions* **[SAF14]** erlauben es, die teils großen Feature eines Produkts, in kleine Einzelteile zu zersetzen und diese einzeln zu betrachten.

Es gibt eine Vielzahl von Faktoren, die sich auf die Popularität von Applikationen auswirken können. Der Bekanntheitsgrad des Entwicklers, Marketing und die Anzahl der potenziellen Nutzer sind nur einige dieser Faktoren. Entwickelt man eine Applikation, die einen sehr spezifischen Nutzen hat, der zum Beispiel nur für eine bestimmte Berufsgruppe wichtig ist, schrumpft die mögliche maximale Anzahl der Nutzer. Betrachtet man jedoch die meistbenutzten unter solchen Nischenapplikationen, fällt auf, dass sich meist die mit der besten User Experience durchsetzen. Auf dem Google Play Store werben Apps oft damit, dass sie nach Googles Designsprache *Material Design* **[GOO]** implementiert wurden. Dies ist eine effektive Methode, mit der Entwickler ihre Apps ansprechender für die Nutzer machen.

Die meisten Android Nutzer haben nämlich bereits Erfahrung mit *Material Design*, da die nativen Apps in modernen Android-Betriebssystemen nach den Regeln dieser Designsprache entwickelt wurden. Diese Apps zeichnen sich meist durch ein übersichtliches User Interface mit simplen, verständnisfördernden Animationen aus.

Diese Arbeit befasst sich mit dem Design von *Microinteractions* und Animationen innerhalb einer App zum Versenden von Daten zwischen zwei Geräten.

In Kapitel 2 wird dafür zunächst auf die Grundlagen zu diesen beiden Themen eingegangen.

Kapitel 3 beschäftigt sich mit dem gesamten Interaktionskonzept. Hier wird beschrieben, wie die Applikation funktioniert und aus welchen Microinteractions sie besteht. Diese Microinteractions werden daraufhin analysiert und es werden passende Animationen zu jeder entwickelt.

Abschließend wird in Kapitel 4 auf mögliche zukünftige Verfeinerungen des Interaktionskonzepts eingegangen.

2 Grundlagen

2.1 Microinteractions

Laut **Dan Saffer (2013)** sind Microinteractions die funktionalen, interaktiven Details eines jeden Produkts. Der Vorgang der Passwortwahl bei der Registrierung eines neuen Accounts auf einer Website, ist ebenso eine Microinteraction, wie das Lösen der Tastensperre eines Smartphones oder das physische Betätigen eines Startknopfes, um ein Gerät einzuschalten.

Diese Details können entweder nur jeweils einen kleinen Teil eines Produkts ausmachen, wie in den zuvor genannten Microinteractions, oder aber die gesamte Funktionalität darstellen. Ein Beispiel für solch ein Produkt ist die Android Applikation „Blue Light Filter for Eye Care“ von Entwickler „Hardy-infinity“. **[HAR]** Diese App erlaubt dem Nutzer einen Filter über den Bildschirm des Gerätes zu legen, welcher blaues Licht vermindert und somit die Augen schonen soll. Die Microinteraction des Einschaltens des Filters ist alles, was die App können muss. Typischerweise bestehen Produkte jedoch aus einer Vielzahl von Microinteractions, wobei mehrere aufeinanderfolgend ausgeführt werden müssen um eine gewünschte Aktion auszuführen.

Eine Microinteraction selbst besteht aus vier Elementen: Einem Auslöser („Trigger“), Regeln („Rules“), Feedback und Schleifen und Modi („Loops and Modes“). Diese Elemente werden in den folgenden Unterkapiteln näher erläutert.

2.1.1 Trigger

Der Trigger ist der Auslöser für eine Microinteraction. Das heißt, sobald die Konditionen des Triggers erfüllt werden, spielt sich der Rest der Interaktion ab. Es gibt zwei verschiedene Arten von Triggern, den manuellen- und den Systemtrigger, die in jeweils verschiedenen Kontexten verwendet werden.

Ein manueller Trigger wird gebraucht wenn der Nutzer aktiv etwas tun möchte. Will er zum Beispiel die Tastensperre seines Handys lösen oder eine Email abschicken wird er einen manuellen Trigger aktivieren müssen („Swipe to Unlock“, Senden-Knopf klicken).

Ein Systemtrigger hingegen kommt zum Einsatz, wenn eine Microinteraction ohne Nutzerinteraktion gestartet werden können muss. Eine Weckerapplikation könnte zum

Beispiel Systemtrigger benutzen um zur richtigen Zeit die Microinteraction des Weckens zu starten. Dazu hört es auf die interne Uhr des Geräts und überprüft bei jedem Minutenwechsel, ob ein Wecker gestellt wurde.

2.1.2 Rules

Rules sind die Regeln einer Microinteraction. Sie bestimmen was geschehen soll, sobald die Microinteraction durch den Trigger ausgelöst wurde, wie lange sie dauert und was der Nutzer während des Ablaufs tun muss. Insgesamt verhält sich die ganze Microinteraction so, wie die Regeln es beschreiben. Eine der Regeln beim Erstellen eines Accounts auf einer Website könnte zum Beispiel lauten, dass das Passwort nicht kürzer, als sechs Zeichen sein darf.

2.1.3 Feedback

Feedback in Microinteractions hat in erster Linie die Aufgabe die Regeln der Microinteraction zu vermitteln. Wenn ein Nutzer zum Beispiel einen Knopf drückt, sollte ihm Feedback gegeben werden, das zwei Aufgaben erfüllt. Erstens, sollte der Nutzer wissen, *dass* dieser Button gedrückt wurde. Zweitens, muss der Nutzer erfahren, was das Drücken dieses Knopfes bewirkt hat.

Allgemein muss Feedback in den meisten Fällen eine dieser Nachrichten an den Nutzer übermitteln:

- Etwas ist geschehen (Es ist 8 Uhr und du hast einen Wecker gestellt)
- Du (der Nutzer) hat etwas getan (Du bist nun eingeloggt)
- Ein Prozess hat begonnen (Download wurde gestartet)
- Ein Prozess wurde beendet (Der Download ist beendet)
- Ein Prozess wird verarbeitet (Prozentanzeige des Downloads)
- Du kannst das nicht tun (Das Passwort ist zu kurz)

2.1.4 Loops and Modes

Loops („Schleifen“) in Microinteractions kommen immer zum Einsatz, wenn eine Microinteraction oder ein Teil davon mehrfach wiederholt werden oder zu einem späteren Zeitpunkt etwas aktivieren muss. Loops werden immer durch die Regeln einer Microinteraction bestimmt. Hat man zum Beispiel mehrmals sein Passwort falsch eingegeben, könnte ein Loop lauten, für eine Minute Einlogversuche zu unterbinden. Das

stellen einer Erinnerung in einer Kalender Applikation ist auch ein Loop, der erst beendet ist, wenn der Tag der Erinnerung gekommen ist.

Modes („Modi“) sind im Zusammenhang mit Microinteractions eine „Gabelung in den Regeln“. **[SAF13]** Damit ist gemeint, dass die normalen Regeln einer Microinteraction nicht mehr so gelten, wie ohne den Mode. Ein Beispiel dafür ist die Capslock Taste. Wenn sie einmal betätigt wurde, verhält sich die Eingabe von Text anders, als es normalerweise der Fall ist: alle Kleinbuchstaben werden zu Großbuchstaben und umgekehrt. **Saffer (2013)** erwähnt, dass die meisten Microinteractions ohne Modes auskommen sollten, da sie den Nutzer verwirren indem sie die bereits bekannten Regeln ändern.

2.2 Animationen

Der Begriff Animation stammt von dem Lateinischen Wort „animus“, beziehungsweise dem dazugehörigen Verb „animare“, ab. Animus bedeutet Geist oder Seele. Abgeleitet davon bedeutet animare „beseelen“ oder „zum Leben erwecken“. **[WIK16]** Etwas zu animieren heißt also, ein nicht lebendiges Objekt lebendig erscheinen zu lassen. Oder anders ausgedrückt: Etwas bewegen, dass sich nicht selbst bewegen kann.

Konkret bedeutet das für die 2D Animation, dass Bilder gezeichnet werden, die sich nur leicht voneinander unterscheiden. Wenn diese Bilder in der richtigen Reihenfolge schnell nacheinander gezeigt werden, wird dadurch eine Illusion von Bewegung erzeugt.

Diese Technik wurde zwar schon lange vor Walt Disneys Ära von Zeichentrickfilmen verwendet, jedoch revolutionierten die von Disney verwendeten Animationspraktiken die Art und Weise, wie Animationen erstellt wurden. In den Worten von Frank Thomas, einem von Walt Disneys Hauptanimateuren:

> *„… Als jeder dieser Prozesse einen Namen bekam, wurde er analysiert, perfektioniert und diskutiert. Und wenn neue Künstler angestellt wurden, wurden ihnen diese Praktiken beigebracht als wären sie schon etabliertes Handwerkszeug. Zur Überraschung aller wurden sie tatsächlich zu den fundamentalen Prinzipien von Animation.“* **[THO81]**

Im nächsten Unterkapitel werden diese 12 Animationsprinzipien aufgelistet und diejenigen, die für diese Arbeit relevant sind, erklärt. **[THO81]**

2.2.1 Walt Disneys Animationsprinzipien

Squash and Stretch

Die laut Frank Thomas bei weitem wichtigste Entdeckung in der Animationstechnik wurde mit „Squash and Stretch“ betitelt. Vor Squash and Stretch wurden animierte Figuren nicht sonderlich in ihrer Form verändert, während sie sich bewegt haben. Dies führte dazu, dass die Figuren, vor allem in Bewegung, steif und leblos wirkten. Das liegt daran, dass in der realen Welt nur wenige Objekte, wie Möbelstücke, komplett unnachgiebig und unelastisch sind. Alles Lebendige bewegt sich auch innerhalb seiner Form, nicht nur von Szene zu Szene. Betrachtet man ein Lächeln, so hat es nicht nur die Charakteristik, dass die Lippen sich nach oben neigen. Das gesamte Gesicht eines Menschen verändert sich, unter anderem werden die Wangen angehoben und dicker.

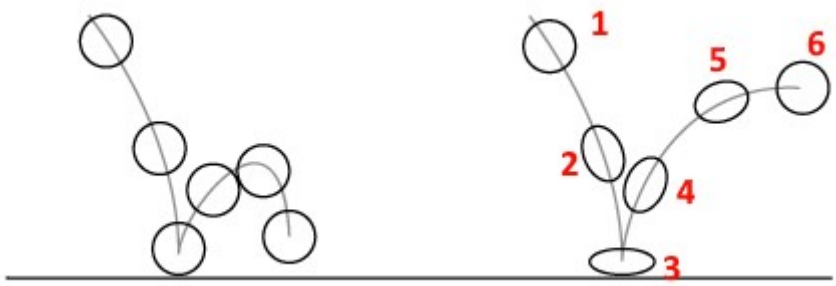

Abbildung 1: Ohne und mit Squash and Stretch **[vgl. STE]**

Ein einfaches Beispiel, um das Konzept von Squash and Stretch zu verdeutlichen sieht man in Abbildung 1. Links und rechts sieht man jeweils einen Ball, der fällt und nach Berührung mit dem Boden wieder abspringt. Die linke Animation ist hier ohne die Verwendung von Squash and Stretch realisiert, die rechte mit. Sieht man sich beide Animationen in bewegter Form an, scheint es als sei der linke Ball schwer und hart wie eine Stahlkugel. Der rechte wirkt weniger hart, eher wie ein Gummiball. Durch die

Verformung zu einem Oval an Punkt 3 in der Animation sieht man, dass der Ball elastisch ist und sich durch den Aufprall zusammenzieht. Das Oval an den Punkten 2, 4 und 5 lässt den Ball schneller erscheinen. Durch einfaches Quetschen und Strecken lassen sich also die wahrgenommenen Merkmale eines Objekts verändern.

Bevor Squash and Stretch verwendet wurde, hatten animierte Charaktere meist nur Röhren als Arme und Beine, die sich zwar bewegten, in der Länge und Breite aber gleich blieben. Dies passt immer noch in manchen Fällen, wie zum Beispiel in der kurzen Animation in Abbildung 2. Bei dem animierten Charakter handelt es sich um ein mechanisches Pferd, dass den Kopf senkt und sich darauf vorbereitet loszurennen.

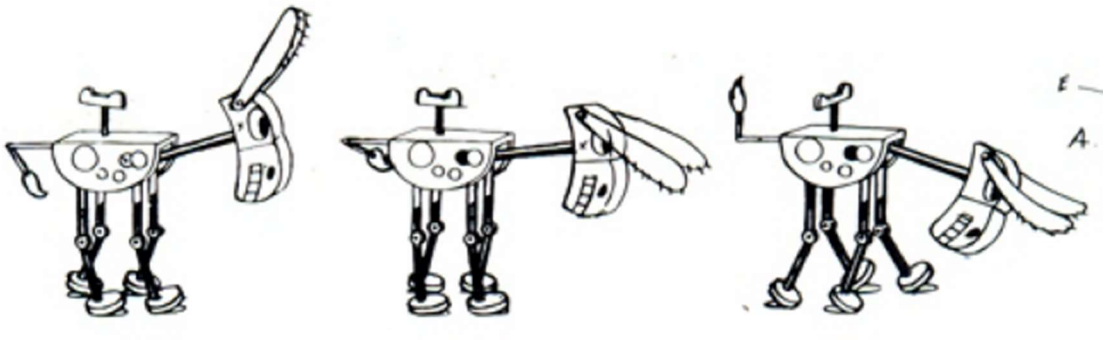

Abbildung 2: Ohne Squash and Stretch **[THO81]**

Wäre das Pferd hier jedoch nicht aus Holz, sondern aus Fleisch und Blut, würde die Animation zu steif wirken. Betrachtet man zum Beispiel einen menschlichen Arm, der sich beugt, lässt das den Bizeps schwellen und streckt den Trizeps. Der Arm gewinnt zwar nicht an Volumen, verändert jedoch seine Form, sodass es so scheinen kann.

Dies ist eine wichtige Regel zur Verwendung von Squash and Stretch - Das Volumen des Objektes sollte immer gleich bleiben.

Anticipation

Anticipation beschreibt das Konzept, dass jede größere Aktion in einer Animation vorher durch eine kleinere Aktion vorbereitet wird. Dies ermöglicht es dem Zuschauer der Handlung auf dem Bildschirm zu folgen. Wenn Anticipation richtig benutzt wird, weiß der Zuschauer nämlich schon im Voraus was ungefähr geschehen wird. Wenn man zum Beispiel Abbildung 3 betrachtet, weiß man direkt, was gleich geschehen wird: Donald wird jeden Moment losrennen. Das erkennt man an seiner ganzen Haltung. Er nimmt

mit den Armen Schwung, hat bereits ein Bein gehoben und den Blick nach vorn gerichtet.

Abbildung 3: Anticipation **[THO81]**

Obwohl der Bewegungsablauf etwas übertrieben ist, ist diese Vorbereitung auf eine Aktion direkt aus der realen Welt übernommen. Ohne Anticipation können Bewegungen nur wenig Kraft und Geschwindigkeit entwickeln. Ohne den langen Rückschwung könnte ein Golfer den Ball nicht annähernd so weit abschlagen. Anticipation ist also aus zwei Gründen essentiell:

1. Bereitet den Zuschauer auf die nächste Aktion hin, sodass er nichts verpasst.
2. Lässt das Animierte realistischer erscheinen.

Staging

Das Ziel von Staging ist recht simpel: Alles was dem Zuschauer übermittelt werden soll, muss unmissverständlich klar dargestellt werden. Gutes Staging für eine Animation zu erstellen ist jedoch keine einfache Aufgabe, da es durch das Zusammenspiel aller Elemente einer Szene entsteht. Will man zum Beispiel eine bestimmte Atmosphäre darstellen, sollte man Objekte und Hintergründe verwenden, die allgemein mit dieser Atmosphäre in Verbindung gebracht werden. Will man eine emotionale Reaktion des Charakters zeigen, sollte dies in einer Nahaufnahme geschehen, statt in einer Panoramaaufnahme.

Zusammengefasst sollte alles, was auf dem Bildschirm zu sehen ist, dafür da sein, die Aussage der Szene zu verdeutlichen. Will man in einer Szene beispielsweise eine gruselige Atmosphäre erzeugen, sollte der Hintergrund ein typisch gruseliger Ort, wie ein altes Schloss sein. Zusätzlich könnten kleinere Details wie Spinnenweben oder krabbelnde Käfer hinzugefügt werden. Man könnte den Charakter klein im Vergleich zum Umfeld darstellen, um die Bedrohung und Machtlosigkeit zu vermitteln. Die Szene sollte in diesem Beispiel relativ dunkel gehalten werden, um die Spannung aufrechtzuerhalten. Der Charakter selbst sollte nervös wirken, sich oft umschauen und zusammenzucken, wenn ein Geräusch ertönt. Dies sind nur einige Beispiele, wie man einer Szene gutes Staging verleihen kann.

Straight Ahead Action and Pose to Pose

Straight Ahead Action (SSA) und Pose to Pose (PtP) sind zwei grundlegende Methoden Animationen zu erstellen. Mit SSA zeichnet der Animateur immer von einem Bild zum nächsten. Er hat eine grobe Idee davon, was passieren soll und zeichnet die einzelnen Bilder in der gleichen Reihenfolge, in der sie auch später abgespielt werden.

PtP beschreibt eine andere Herangehensweise an den Animationsprozess: Zunächst werden die ausdrucksstarken und wichtigsten Posen der Figur gezeichnet. Anschließend werden die Zwischenbilder manuell erstellt oder durch einen Computer approximiert.

Beide Methoden haben ihre Vor- und Nachteile und werden heute noch verwendet. In der Computeranimation wird jedoch hauptsächlich mit Pose to Pose gearbeitet, weil der Computer die Bilder zwischen den Posen erstellen kann und dadurch viel Arbeit erspart wird.

Follow Through and Overlapping Action

Follow Through und Overlapping Action sind zwei weitere Konzepte, die entwickelt wurden um Animationen lebensechter gestalten zu können. Die Begriffe sind nicht eindeutig definiert und werden synonym verwendet.

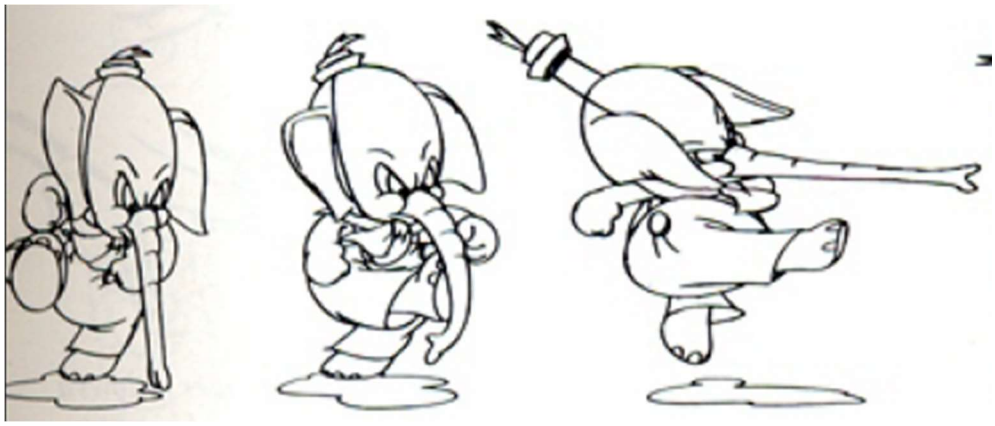

Abbildung 4: Follow Through und Overlapping Action an Dumbos Rüssel **[THO81]**

Die Funktion dieser Methoden ist es als das Gegenstück zu Anticipation zu wirken: Nach einer Aktion darf das animierte Objekt nicht schlagartig zum Stehen kommen, da das schnell abgehackt und somit unrealistisch wirken kann. Weiche Teile der Figur sollten sich weiterhin bewegen, selbst wenn die Figur ihr Ziel bereits erreicht hat. In Abbildung 4 sieht man Dumbo während der letzten Momente eines harten Trittes. Durch den Schwung, der bei dieser Aktion entsteht sieht man wie sein Rüssel mit nach vorne geschleudert wird. Würde der Rüssel in einer hängenden Position bleiben würde dies als unrealistisch empfunden werden.

Slow In and Slow Out

Slow In and Slow Out kommt zum Einsatz wenn man zum animieren die Pose to Pose Methode wählt. Statt das animierte Objekt in einer stetigen Geschwindigkeit zwischen den Posen zu wechseln zu lassen, werden viele Bilder gezeichnet, die sich nah an den Posen befinden und nur sehr wenige in der Mitte. Das hat den Effekt, dass das Objekt sich einen Großteil der Zeit in den ausdrucksstarken Positionen befindet und nur wenig Zeit mit der Transition zugebracht wird.

Arcs

Nur sehr wenige Lebewesen in der realen Welt bewegen sich in geraden Linien. Meistens folgen Gesten und selbst abrupte Bewegungen irgendeiner Art von Bogen.

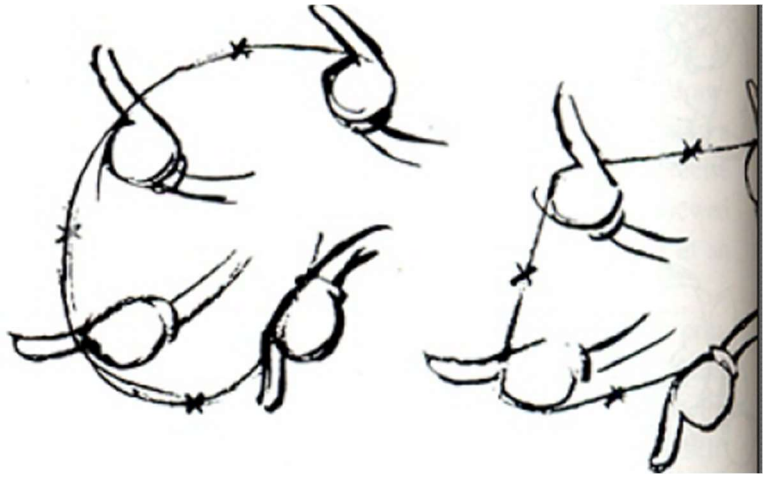

Abbildung 5: Bogenförmige Bewegungen **[THO81]**

In Abbildung 5 sieht man zwei Animationen eines Fingerzeigs. Die Linien zwischen den Grundposen zeigen an, an welchen Stellen Zwischenzeichnungen eingefügt werden sollen. Obwohl die Posen sehr ähnlich sind, würden diese beiden Animationen in Bewegung sehr unterschiedlich wirken. Die linke würde natürlich und normal wirken, die rechte jedoch abgehackt und steif.

Secondary Action

Secondary Action beschreibt das Konzept, dass man der Hauptanimation durch weitere kleinere Animationen mehr Kraft und Verständlichkeit verleihen kann. Ist ein Charakter beispielsweise sehr traurig, kann das dem Publikum durch den Gesichtsausdruck und die Haltung vermittelt werden. Was den Effekt aber noch verstärken würde, wäre wenn der Charakter sich eine Träne wegwischen würde.

Diese Methode hat sehr viele Anwendungsmöglichkeiten, jedoch muss man darauf achten, dass die Hauptanimation im Vordergrund bleibt und nicht durch die Secondary Action überspielt wird. Wenn diese andererseits zu wenig Aufmerksam auf sich zieht, wird sie vielleicht gar nicht erst wahrgenommen und kann somit weggelassen werden.

Timing

Timing ist ein sehr wichtiger Aspekt von Animationen, ist jedoch weniger Wissenschaft als Gefühlssache. Im Prinzip geht es um die Entscheidung, wie viele „Transitionszeichnungen“ zwischen den Posen liegen sollen. Je mehr Zeichnungen, desto langsamer wird die Aktion. Stellt man sich einen menschlichen Charakter vor, der seinen

Kopf nach links geneigt hat (Pose 1). In der nächsten Pose hat der Charakter seinen Kopf nach rechts oben geneigt. Hat man zwischen diesen beiden Posen nur eine Zeichnung, geschieht alles so schnell, dass es wirkt als wäre der Charakter geschlagen worden. Hat man stattdessen sechs Zwischenzeichnungen, könnte er nur auf die Wanduhr schauen.

Durch gutes Timing kann ebenfalls ein Gefühl von Masse vermittelt werden, da die Beschleunigung und Bremsung eines Objekts erst durch Ablauf der Zwischenzeichnungen entsteht. Fügt man sehr viele Zeichnungen nahe den Posen ein, beschleunigt das animierte Objekt langsamer und wirkt dadurch massiver und schwerfälliger.

Solid Drawing

Solid Drawing beschreibt, dass ein Animateur gut zeichnen können muss. Bevor man einen Charakter bewegen kann muss einem bekannt sein, wie er in einer Momentaufnahme auszusehen hat.

Appeal

Appeal, zu Deutsch Anreiz, ist eine der wichtigsten und gleichzeitig sehr vagen Qualitäten einer Animation. Benutzt man alle vorherigen Prinzipien in der richtigen Zusammensetzung, so bekommt eine Animation einen gewissen Charme, der den Zuschauer positiv beeinflusst. Diesen Charme will man heute auch in Computeranimationen einfangen um Nutzern von Applikationen eine positive User Experience zu liefern.

Exaggeration

Die meisten dieser Prinzipien sind wurden mit dem Ziel aufgestellt Animationen lebensechter, realistischer zu gestalten. Übertreibung war für die Animation ein wichtiges Stilmittel, um nicht langweilig zu werden. Animationen waren ein Medium, das schon lange vor der Entwicklung von Computergrafik unrealistische und fantastische Dinge darstellen konnte.

Die Entwicklung der hier aufgeführten 12 Prinzipien ermöglichte es den Animateuren unreale Dinge zu zeigen, die jedoch trotz allem realistisch erschienen.

2.2.2 Material Design und Animationen

Google hat einige Design Guidelines bereitgestellt, um Animationen des User Interface innerhalb von Applikationen einheitlich zum Rest des Android Betriebssystems erscheinen zu lassen. Diese Guidelines sind ein Teil von Googles eigener Designsprache. **[GOO]** Um Animationen im Kontext von dieser Designsprache zu verwenden, muss man zunächst einige zugrundeliegende Prinzipien verstehen.

Mit Android Lollipop, der 5.0 Version des Android Betriebssystems, etablierte Google 2014 eine neue Designsprache, die sie Material Design nannten. Die Metapher hinter dieser ist, dass alle sichtbaren Elemente des User Interfaces aus einem Papierähnlichen Material bestehen und auf einer Fläche aufliegen (Siehe Abbildung 6). Dieses *Material* soll sich ähnlich wie Papier in der realen Welt verhalten, ist jedoch nicht an alle physikalischen Gesetze gebunden; *Material* ist zum Beispiel streckbar, wobei Papier zerreißen würde. **[GOO]**

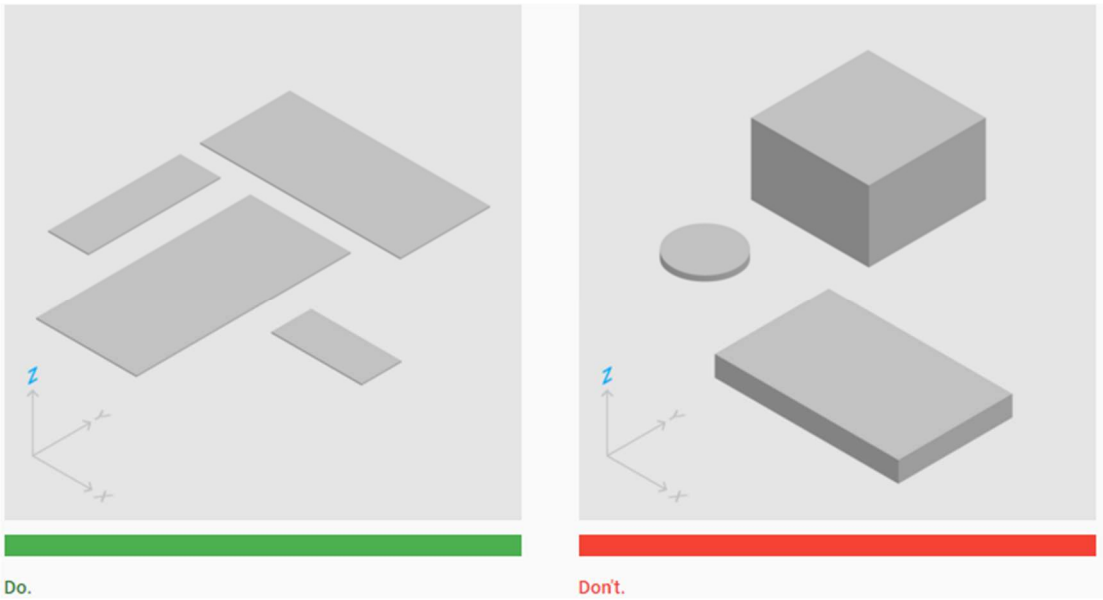

Abbildung 6: Material Design - Flache Elemente **[GOO]**

Auf den ersten Blick scheint eine Applikation die Material Design implementiert sich nicht sonderlich von anderen flach gestalteten Applikationen zu unterscheiden. Ein entscheidender Unterschied zwischen Material Design und älteren Flat-Designsprachen ist, dass Googles Sprache Dreidimensionalität etabliert, während jeder einzelne Teil der UI flach bleibt.

Dies wird durch eine Anhebung der Elemente erreicht, die durch einen Schatten unter dem Element dargestellt wird (Siehe Abbildung 7). Das Verhalten der Schatten ist auch hier an die Realität angelehnt: Je höher etwas angehoben ist, desto näher ist es an der Lichtquelle. Der geworfene Schatten wird somit größer und weicher.

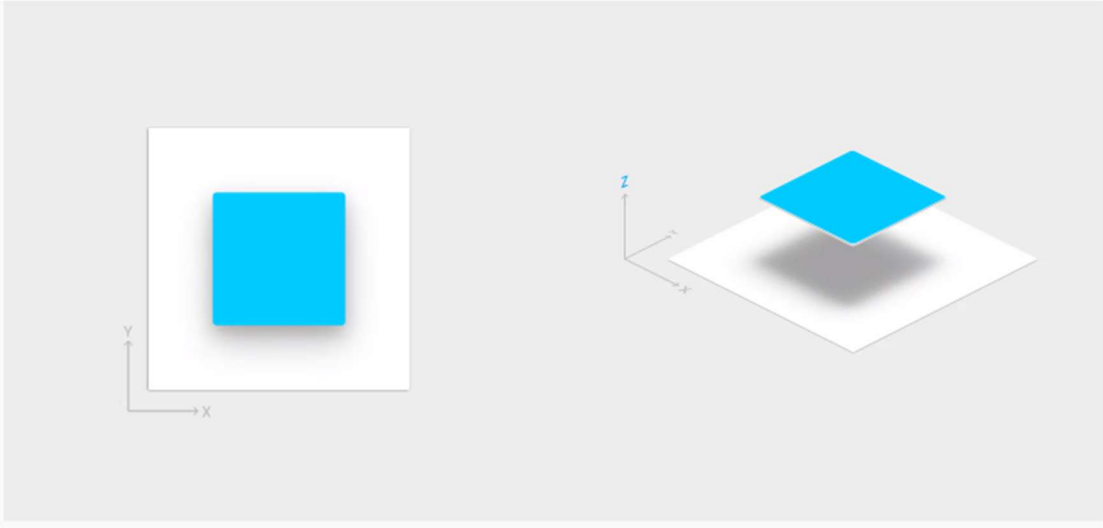

Abbildung 7: Material Design – Dreidimensionalität **[GOO]**

Dieser Unterschied fällt besonders auf, vergleicht man eine App welche Material Design implementiert mit einer App im „Modern UI“ Stil (Vormals „Metro“). Modern UI ist Microsofts Designsprache, die nur zwei Jahre vor Material Design mit der Bekanntgabe von Microsoft Windows 8 etabliert wurde.

In Abbildung 8 sind zwei Applikationen mit ähnlichem Aufbau dargestellt. Im linken Teil sieht man eine Android App im Material Design, im rechten eine Desktop App in Modern UI. Beide Apps verfügen über eine Titelleiste und einige klickbare Elemente. Trotz der ausschließlichen Benutzung von zweidimensionalen Flächen kann man deutlich den Unterschied zwischen den beiden benutzten Designsprachen erkennen: Links „schweben“ die einzelnen Elemente über dem Hintergrund, wobei rechts eine einfache Fläche zu sehen ist. In Microsofts Modern UI heben sich die interaktiven User Interface Elemente lediglich durch die Farbe vom Hintergrund ab.

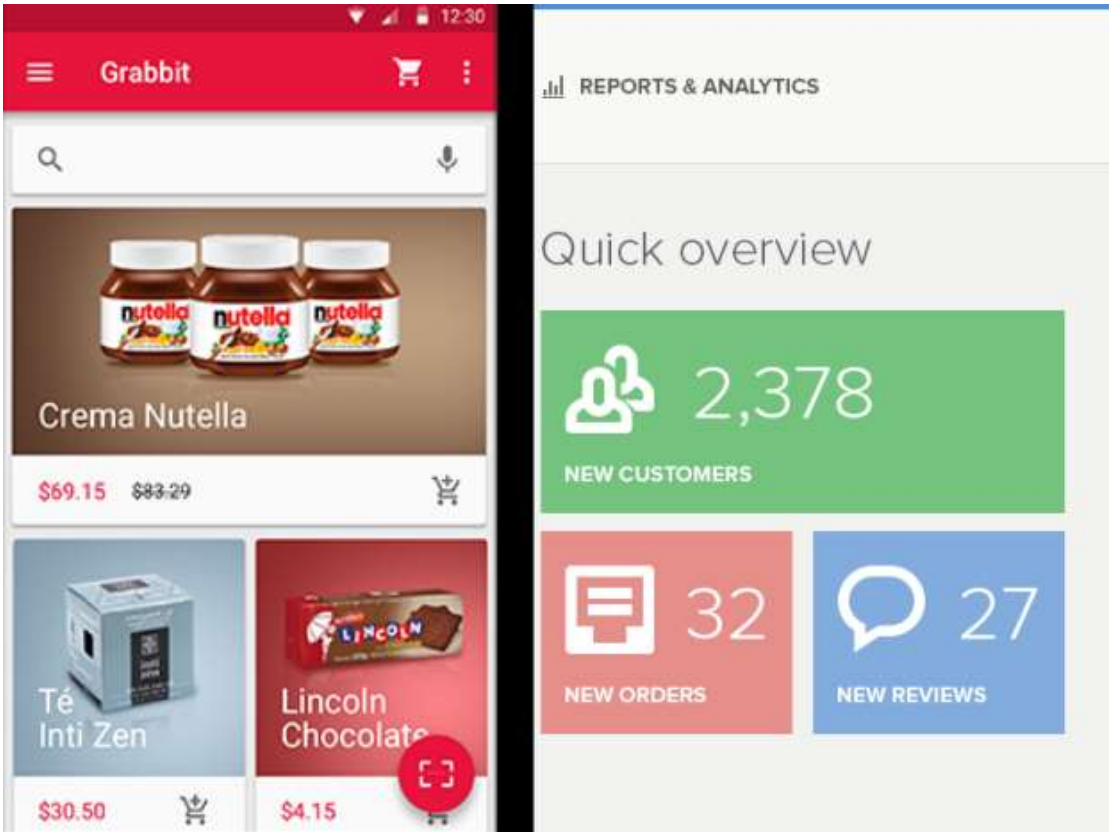

Abbildung 8: Material vs. Modern UI

Die Nutzung der dritten Dimension ist für Googles Designsprache notwendig um die Illusion von physischem *Material* zu erhalten: Zwei Blätter Papier können sich nicht an exakt der gleichen Stelle aufhalten, höchstens übereinander. Betrachtet man einen Stapel Papier von oben, sieht man nur das oberste Blatt. Nur wenn das Blatt leicht versetzt aufliegt, sieht man durch den leichten Schattenwurf, dass es sich um zwei Blätter handelt.

Ein weiterer, für diese Arbeit relevanter, Grundpfeiler von Material Design ist die Animation von User Interface Elementen. Animationen sind unverzichtbar um die Metapher von physischem *Material* aufrecht zu erhalten. Googles Vice President of Design, **Matías Duarte**, sagte folgendes in der Keynote der Google I/O 2014, auf der Material Design erstmals der Öffentlichkeit vorgestellt wurde:

„ … In der realen Welt teleportiert nichts von einem Ort zum anderen. Und deshalb ist es so wichtig, jede Änderung auf dem Bildschirm so zu animieren, dass sie Sinn ergibt.“

Diese Aussage ist zwar simpel und man könnte sie als selbstverständlich erachten, jedoch ist Material Design die einzige Designsprache bei der Animationen eine so große Rolle spielen und gleichzeitig so verständlich mitgeteilt werden, dass jeder Entwickler sie einbinden kann. Durch Einhalten der Regeln von Material Design wird es Entwicklern ermöglicht, eine kohärente User Experience zu erstellen, selbst wenn sie auf diesem Gebiet keine Experten sind.

Auf die Regeln, die Animationen betreffen, wird im nächsten Unterkapitel eingegangen.

Animationsrichtlinien

Googles Designspezifikationen zum Thema Animationen sind in vier Unterkategorien gegliedert **[GOO]**:

- **Authentic Motion** beschreibt, wie sich Objekte in verschiedenen Kontexten auf dem Bildschirm bewegen sollten.
- **Responsive Interaction** beschäftigt sich mit der Frage, wie UI Elemente auf Aktionen von Nutzern reagieren sollen.
- Mit den Richtlinien zu **Meaningful Transitions** wird beschrieben, wie man durch geschickte Übergänge Aufmerksamkeit der Nutzer lenken kann und Verständnisschwierigkeiten vermeidet.
- **Delightful Details be**sagt, dass selbst die kleinsten Animationen ansprechend für den Nutzer sein können und die Qualität des gesamten Produkts steigern.

Diese vier Prinzipien sind noch weiter untergliedert, um sie verständlicher zu gestalten.

Authentic Motion

Material Design versucht dem User Interface etwas Physisches und Realistisches zu verleihen. In der realen Welt erfährt man schon viel über die physikalischen Eigenschaften eines Objekts wenn man wenn man seine Bewegung beobachtet. Braucht etwas lange um von einer gewissen Geschwindigkeit zum Stillstand zu kommen, ist das

Objekt schwer. Springt ein Ball vom Boden ab, statt sofort liegen zu bleiben, ist er aus einem elastischen Material, etc.

Über solche physikalischen Eigenschaften sollte man sich als Entwickler Gedanken machen, wenn man ein UI Element bewegen möchte. **Authentic Motion** beschäftigt sich vor allem mit der (fiktiven) Masse von Objekten und wie diese Masse sich auf die Beschleunigung, Bremskraft und Richtungswechsel auswirkt.

Die erste und wichtigste Regel zu Bewegung ist, dass man vermeiden sollte, eine lineare Geschwindigkeit beizubehalten. Ein UI Element, das sich von einem Ort zu einem anderen bewegt, sollte immer geschmeidig beschleunigen und abbremsen – nichts in der realen Welt startet abrupt in eine bestimmte Geschwindigkeit und hält diese bis es genauso abrupt wieder zum Stillstand kommt. Bewegt sich ein UI Element in dieser Art, fällt das dem Benutzer auf, da es unnatürlich ist.

In Abbildung 9 sieht man anhand von zwei Graphen, wie Bewegung ablaufen kann. Die Graphen beschreiben die vertikale Position eines - sich auf und ab bewegenden - Objektes zu einem bestimmten Zeitpunkt. Die rote Linie zeigt eine dabei eine lineare Bewegung, wie man sie vermeiden sollte. Die Grüne zeigt ein positives Beispiel einer asymmetrischen Beschleunigung. Der Graph wurde hier in sechs Abschnitte unterteilt. In den Phasen 1 und 4 beschleunigt das Objekt. In den Phasen 2 und 5 ist die jeweilige Höchstgeschwindigkeit erreicht und wird gehalten. Die Phasen 3 und 6 zeigen den Bremsvorgang des Objekts. Man beachte, dass die vergangene Zeit für die Phasen 1-3 und 4-6 gleich, der Verlauf der Kurve jedoch nicht gespiegelt ist. In Bewegung wirkt es, als würde das Objekt schneller verschwinden als es gekommen ist. Dies entsteht jedoch nur durch die asymmetrische Beschleunigung, nicht durch die tatsächlich vergangene Zeit.

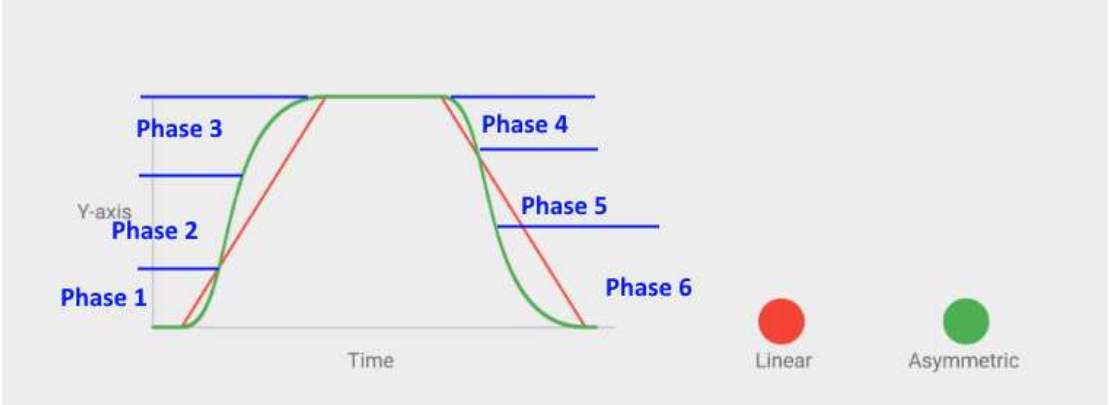

Abbildung 9: Authentic Motion **[vgl. GOO]**

Was man vermeiden sollte ist, dass alle Objekte innerhalb einer Applikation das gleiche Beschleunigungs- und Bremsverhalten haben. Größere Objekte sollten schwerfälliger sein, also langsamer Beschleunigen und Bremsen, als kleine Buttons. Außerdem können selbst kleine Veränderungen der Beschleunigung eines Objekts die Aufmerksamkeit des Nutzers lenken. Solche kleinen Veränderungen können eine Animation lebendig und reizvoll werden lassen, statt gar nicht erst bemerkt zu werden (Was auch oft gewollt sein kann).

Eine weitere interessante Methode, reizvolle Bewegung zu erreichen ist die Verwendung von asymmetrischer Beschleunigung und Bremsung. Als Beispiel nehme man einen Button, der nur den Bildschirm betritt wenn der Nutzer ihn brauchen könnte und danach auf gleichem Wege wieder verschwindet. Möchte man der Animation etwas Besonderes verleihen, könnte man den Button beispielsweise schnell beschleunigen beim Hereinkommen und vielleicht sogar etwas über das Ziel hinaus schießen lassen. Beim Verlassen des Bildschirms hingegen könnte der Button nur langsam Beschleunigen und insgesamt nicht sehr schnell werden, sodass er fast schon widerwillig wirkt unbenutzt die Szene zu verlassen.

Es gibt viele Möglichkeiten mit der Geschwindigkeit und Beschleunigung von Objekten Akzente zu setzen und Gefühle zu vermitteln und die Aufmerksamkeit von Nutzern zu lenken. Lässt man User Interface Elemente auf den Bildschirm eintreten, wie in dem Beispiel zuvor, sollte man sich immer überlegen, was wichtig für den Nutzer ist. Ist das eintretende Element wichtig für den aktuellen Kontext, sollte man vielleicht die Beschleunigungen etwas variieren, um die Aufmerksamkeit auf das Element zu ziehen. Soll die Aufmerksamkeit des Nutzers hingegen auf dem Rest der Applikation bleiben, ist vielleicht eine gleichmäßigere Beschleunigung die richtige Wahl.

Soll das Element für immer verschwinden, sollte es schnell aus dem Bildschirm austreten, um weit weg zu wirken. Ist es hingegen ein Element, das viel genutzt wird und immer wieder auftaucht, sollte man es vielleicht vor dem Verlassen des Bildschirms noch abbremsen lassen. Dies hat den Effekt, dass es wirkt, als sei das Element noch sehr nah und fast in Reichweite.

Durch den geschickten Einsatz von **Authentic Motion** kann schon viel erreicht werden, ohne die Form des animierten Objekts zu verändern.

Responsive Interaction

Responsive Interaction beschreibt, wie User Interface Elemente auf Input von Nutzern reagieren sollen. Allgemein gesagt sollte sich die UI dinglich und somit greifbar anfühlen. Es soll wirken, als würde man das *Material* direkt mit den Fingern manipulieren, obwohl es sich hinter dem Glas des Bildschirms befindet. Die Reaktionen der Elemente geben dem Nutzer möglichst verständliches, aber auch visuell ansprechendes, Feedback zu der von ihm ausgeführten Aktion. Diese angenehmen Details sollen die User ermutigen, die Applikation zu erforschen. Sie sollen Spaß daran haben, herumzuklicken, zu swipen und Animationen zu entdecken.

Google stellt dazu einige Interface Reaktionen vor, die die Kriterien einer guten **Responsive Interaction** erfüllen. Diese sind seit Android 5.0 in den wichtigsten UI Elementen, wie Buttons oder Karten, verbaut und lassen sich entweder unverändert übernehmen oder nach Wünschen anpassen und austauschen.

Eine dieser Feedback-Reaktionen wird *Surface Reaction* genannt und hat die Aufgabe dem Nutzer zu bestätigen, mit welchem Teil des User Interfaces er gerade interagiert.

Abbildung 10: Surface Reaction **[vgl. GOO]**

In Abbildung 10 ist ein Beispiel einer typischen *Surface Reaction* dargestellt. Wie man sieht, hat man eine Liste von fünf klickbaren Flächen. Der blaue Kreis symbolisiert hier die Stelle an der der Nutzer einen Touch ausgeführt hat. Die gesamte Schaltfläche wird augenblicklich hellgrau eingefärbt, um sofortiges Feedback zu liefern, was der Nutzer gerade berührt hat (1). Eine zweite, dunklere Fläche breitet sich vom Berührungspunkt aus kreisförmig aus (2). Man beachte, dass die dunklere Fläche nicht über die Grenzen des Elementes hinauswächst. Das gerade berührte Element soll das Zentrum der Aufmerksamkeit des Nutzers bleiben. Hat sich die gesamte Schaltfläche dunkelgrau eingefärbt, ist die **Responsive Interaction** abgeschlossen und das Element kehrt wieder zum Ausganszustand zurück. Die gesamte Interaktion dauert etwa eine halbe Sekunde, vermittelt alle wichtigen Informationen zur Eingabe des Nutzers und hat aufgrund der zirkulären Ausbreitung der dunkelgrauen Fläche etwas Ansprechendes.

Zusätzlich zu solch einer Oberflächenreaktion kann man von den Qualitäten des *Materials G*ebrauch machen. Eine häufig verwendete Interaktion ist das Anheben eines berührten Elementes, wie es in Abbildung 11 dargestellt ist. Dadurch wird dem Nutzer noch klarer vermittelt, welches Element gerade aktiv ist.

Abbildung 11: Lift on Touch **[GOO]**

Insgesamt sollte jede Aktion des Nutzers in irgendeiner Weise zur Kenntnis genommen werden und eine Reaktion hervorrufen. Selbst wenn ein Element zurzeit nicht benutzbar ist, sollte es Feedback liefern und diesen Fakt vermitteln. Der User sollte sich nie

fragen müssen, ob eine seiner Aktionen überhaupt vom Gerät registriert wurde. Dadurch können zum Beispiel versehentliche doppelte Eingaben verhindert werden, da der Nutzer sofort sieht, dass seine Aktion zur Kenntnis genommen wurde.

Solche Reaktionen dauern zwar nur selten länger als eine Sekunde, aber während dieser Zeit kann die App im Hintergrund bereits die vom Nutzer gewünschte Aktion ausführen, beziehungsweise vorbereiten. Die Zeit, die für die Animationen benötigt wird, reicht dafür oft schon aus. Somit kann der richtige Einsatz von **Responsive Interactions** eine Applikation performanter erscheinen lassen, als sie wirklich ist.

Meaningful Transitions

Gute Übergänge zwischen Zuständen einer Applikation können die Aufmerksamkeit eines Nutzers so lenken, dass er jederzeit weiß, was geschieht und was seine weiteren Interaktionsmöglichkeiten sind. Das verleiht der App einen guten Fluss und vermeidet Verwirrung des Users. Ein schönes Beispiel von **Meaningful Transitions** kann man in Abbildung 12 beobachten.

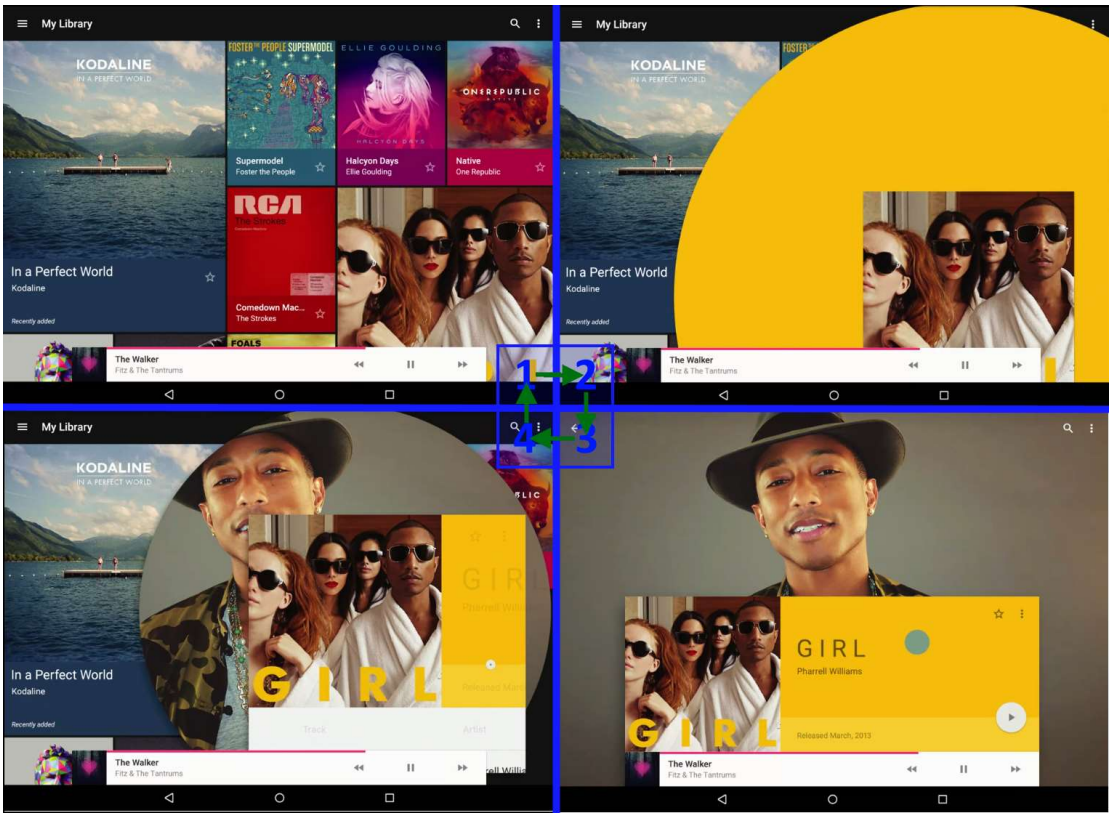

Abbildung 12: Meaningful Transitions **[vgl. GOO]**

Man sieht eine Musikplayer Applikation zu vier verschiedenen Zeitpunkten:

(1) ist der Ausgangszustand, wobei alle vorhandene Musik auf Schaltflächen zur Schau gestellt wird.

In **(2)** sieht man den Übergang nachdem das Albumcover von Pharrels Album G I R L gewählt wurde.

In **(3)** sieht man das geöffnete Fenster zum Album. In diesem Fenster sieht man die Songs aufgelistet und kann sie auswählen um sie abzuspielen.

(4) ist der Übergang wieder zurück zu **(1)**, nachdem der Zurück-Knopf betätigt wurde.

Insgesamt gibt es bei Übergängen drei verschiedene UI Elementtypen **[GOO]**:

i. Neue Elemente, die es einzugliedern gilt.
ii. Nicht mehr benötigte Elemente, die entfernt werden sollen.
iii. Gemeinsame Elemente, die in den nächsten Zustand übernommen werden sollen.

In dem Beispiel aus Abbildung 12 sind alle drei Typen vertreten. Im Übergang von **(1)** zu **(3)** muss das Fenster mit der Songauswahl und das Hintergrundfoto neu erstellt und angezeigt werden, die Albumauswahl aus **(1)** verschwinden und das Albumcover mit übernommen werden.

Durch Antippen des Covers breitet sich eine kreisförmige gelbe Fläche aus und überdeckt die Albumauswahl, womit (ii) abgeschlossen ist. Das gewählte Cover bewegt sich von seinem Ursprungsort zu der Position in **(3)**. Das Cover ist somit das übernommene gemeinsame Element (iii). Gleichzeitig bildet sich um das Albumcover das neue Fenster mit der Songauswahl (i). Sobald der gelbe Kreis den gesamten Bildschirm bedeckt, wird das Hintergrundbild eingeblendet (i).

Generell sollte man bei Übergängen auf drei Dinge achten:

Erstens, wohin und wie lenkt man die Aufmerksamkeit des Nutzers? Das wichtige Element, jenes welches der Nutzer als nächstes benötigt um seine gewünschte Aktion auszuführen, sollte in den Vordergrund gerückt werden. Das Songauswahl-Fenster aus

obigem Beispiel ist in dem Fall das wichtige Element. Die Aufmerksamkeit des Nutzers ist auf diesem Fenster, da es in die Mitte des Bildschirms gerückt wird und über dem Hintergrund schwebt.

Zweitens, sollten Übergänge visuell verbunden werden um logisch und verständlich zu vermitteln was gerade geschieht. Dies kann zum Beispiel durch übernommene Elemente geschehen, oder durch die Nutzung der gleichen Farben. In Abbildung 12 geschieht beides: Der sich ausbreitende Kreis ist von dem gleichen Gelb wie der „G I R L“-Schriftzug auf dem Albumcover, das Coverbild selbst wird zusätzlich dazu übernommen.

Drittens, sollte man sehr genau auf die einzelnen beweglichen Elemente achten um chaotische Bewegungen zu vermeiden. Dies nennt Google eine Konsistente Choreographie. Bewegen sich mehrere Objekte gleichzeitig, sollte dies in einer logischen Art und Weise geschehen. Würde das Cover zum Beispiel von oben den Bildschirm betreten um dann zu der Position in **(3)** zu gelangen, würde das weniger sinnvoll sein als die Bewegung vom Ursprungsort aus. Die richtige Choreographie für den jeweiligen Kontext zu wählen, kann die größte Herausforderung sein, die der Entwickler überwinden muss, da es nur wenige feste Regeln gibt und verschiedene Situationen verschiedene Ansätze erfordern.

Delightful Details

Das letzte Prinzip zu Animationen in Material Design besagt, dass Animationen auf allen Ebenen einer Applikation verwendet werden sollten um das Gesamtprodukt zu verbessern. Das reicht von großen Transitionen zwischen getrennten Teilen einer App bis hin zur Animation der kleinsten Bestandteile. Gerade diese kleinen Teile, wenn schön animiert, verleihen einer App den letzten Schliff. Sie können den Nutzer angenehm überraschen, und sich so von anderen Applikationen abheben. Ein schön animierter Ladebalken zum Beispiel kann die Toleranz für Ladezeiten erhöhen.

Zusätzlich zu dem visuellen Anreiz können solche Animationen benutzt werden um die App funktionaler zu gestalten. In Abbildung 13 zum Beispiel, sieht man wie sich ein Menü-Icon zu einem Zurück-Icon verändert. Dies wird durch eine Rotation nach rechts und das Verbinden der drei Linien zu einem Pfeil erreicht. Das ist eine schöne und clevere Animation, da beide Icons aus den gleichen Grundbausteinen bestehen.

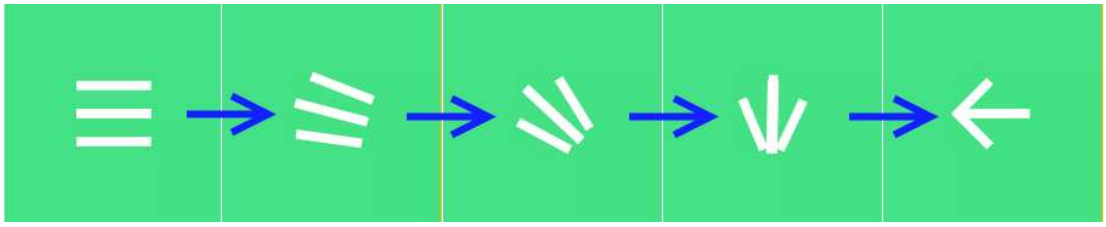

Abbildung 13: Animation von Icons **[vgl. GOO]**

Zusätzlich jedoch wird durch diese subtile Animation Platz auf dem Bildschirm gespart, da der Pfeil den Platz des Menü-Icons einnimmt. Einige Informationen werden auch vermittelt: Dadurch, dass dem Nutzer ein Zurück-Icon angeboten wird, weiß er, dass er sich auf einer niedrigeren Ebene der App befindet. Außerdem gibt es auf dieser Ebene keinen Zugriff auf das Menü, was heißt, dass er es nicht braucht um die gewünschte Aktion auszuführen.

Diese Art von doppelter Benutzung von Icons ist kann also eine elegante Methode sein Nutzern ihre Optionen darzubieten. Ein weiteres Beispiel ist in Googles Musikplayer Applikationen zu beobachten. Der Play-Button muss sich nicht auf dem Bildschirm befinden, wenn die Musik bereits läuft. Ebenso braucht man nicht auf Pause zu drücken, wenn die Musik pausiert ist. Aus diesen Gründen wechselt das Icon zur jeweils nutzbaren Aktion und teilt so dem Nutzer mit in welchem Zustand sich die App befindet.

3 Interaktionskonzept

In den vorherigen Kapiteln wurden die zugrundeliegenden Ideen und Prinzipien behandelt, die für die Konzeption der, im Zuge dieser Arbeit konzipierten, Microinteractions von Bedeutung waren. Im Folgenden wird zunächst das gewählte Pattern erklärt und auf bestehende Microinteractions überprüft. Die somit erkannten Microinteractions werden daraufhin analysiert, um geeignetes Feedback geben zu können. Wie bereits in Kapitel 2 beschrieben, sind Animationen oft eine effektive Art, dem Nutzer zu vermitteln was durch eine bestimmte Interaktion ausgelöst wurde.

Auch in diesem Fall wird auf Animationen zurückgegriffen, deren Design detailliert, mit Bezug auf die in Kapitel 2.2 beschriebenen Animationsprinzipien, erklärt wird.

3.1 Das Swipe-to-Give Pattern

Das Swipe-to-Give Pattern wurde im Zuge des Projekts „Sysplace“ von Alexander Hahn entwickelt und prototypisch umgesetzt. **[HAH15]** Hierbei handelt es sich um eine gestenbasierte Interaktion, die es Smartphone-Nutzern ermöglicht, per Swipe Daten an ein anderes Gerät zu übertragen.

In dem entwickelten Prototyp musste der Nutzer erst auswählen, welche Dateien er versenden wollte. Sobald diese Auswahl getroffen wurde und die Geräte miteinander verbunden waren, wurde durch einen einfachen Swipe über den Bildschirm die Sendung der gewählten Dateien ausgelöst. Die Metapher für dieses Pattern ist die des Übergebens eines Objektes an eine andere Person. Man kann es sich vorstellen wie das hinüberschieben eines Blattes Papier über einen Schreibtisch. Der leere Teil des Bildschirms (Siehe Abbildung 14, Grauer Hintergrund) dient in dieser Metapher als Schreibtisch, über den das Papier geschoben wird.

Für die Konzeption der Microinteractions innerhalb von Swipe-to-Give wird der Auswahlvorgang übergangen, beziehungsweise mit dem Sendevorgang vereint. Das heißt, das sich zum Zeitpunkt des Swipes auf dem Bildschirm befindende Objekt wird versandt. Die Verbindung der beiden involvierten Geräte wird ebenfalls als gegeben angenommen, da diese eine Voraussetzung für eine Kommunikation zwischen Geräten ist und mit der eigentlichen Geste nicht direkt etwas zu tun hat.

Um die Microinteractions innerhalb von Swipe-to-Give zu identifizieren, muss man zunächst den Ablauf der gesamten Interaktion betrachten.

Es werden zwei Geräte benötigt: ein Sendergerät und ein Empfängergerät, die miteinander verbunden sind. Diese Verbindung kann auf verschiedene Arten hergestellt werden. Man könnte die Geräte direkt miteinander verbinden, zum Beispiel über WiFi-Direct oder Bluetooth. Eine andere Möglichkeit die Nutzung eines Vermittlungsservers. Bei dieser Methode sind die Geräte nicht direkt miteinander verbunden, sondern jeweils mit dem Server. Das Sendergerät würde in diesem Fall die Daten an den Server schicken, der sie dann an das Empfängergerät weiterleiten würde.

Auf dem Sendergerät befindet sich eine Datei, welche versendet werden soll. Der Einfachheit und dem Verständnis halber wird angenommen, dass es sich bei dieser Datei um ein Bild handelt, das auf dem Bildschirm des Sendergeräts angezeigt wird (Siehe Abbildung 14).

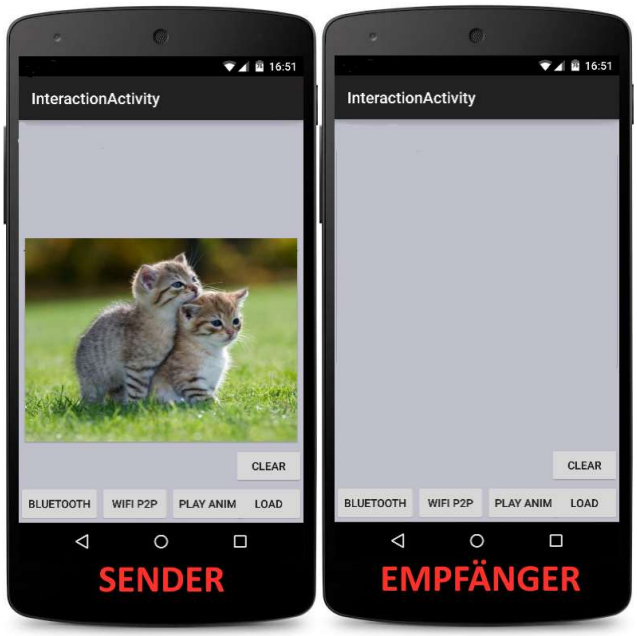

Abbildung 14: Ausgangszustand

Will der Nutzer des Sendergeräts nun das Bild an das Empfängergerät versenden, muss er einen Swipe in Richtung des oberen Bildschirmrandes ausführen.

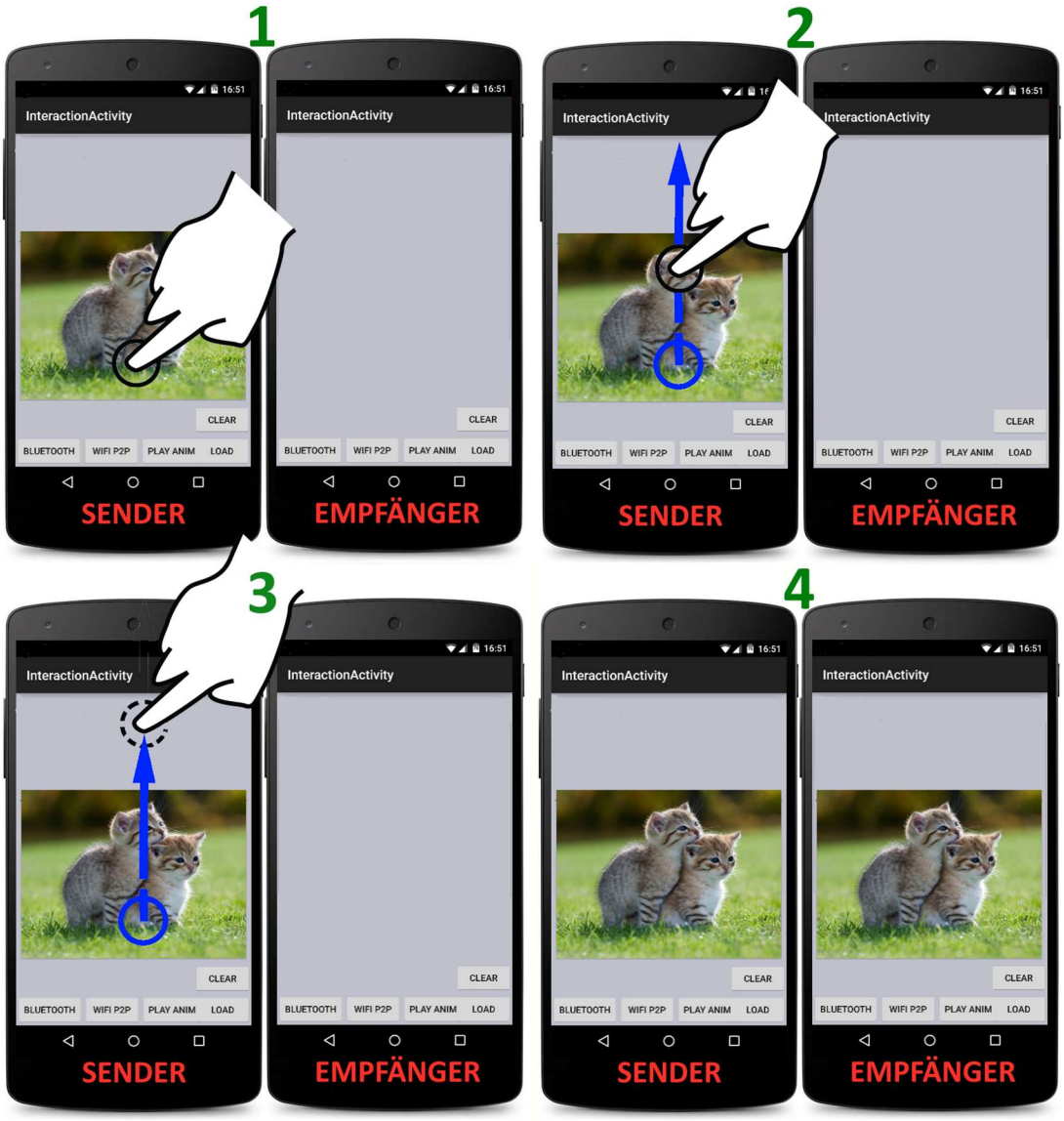

Abbildung 15: Gesamter Ablauf ohne Animationen

Dazu legt der Nutzer erst seinen Finger auf den Bildschirm über dem Bild (Abbildung 15, 1). Dann bewegt er seinen Finger in Richtung des oberen Bildschirmrandes (Abbildung 15, 2). Schließlich löst es den Kontakt mit dem Bildschirm (Abbildung 15, 3). Dies kann entweder geschehen, indem er seinen Finger vom Bildschirm hebt, wie es in der Abbildung gezeigt wird, oder indem er seinen Finger aus dem oberen Bildschirmrand hinausführt.

An dieser Stelle wird geprüft, ob die Aktion des Nutzers alle Anforderungen erfüllt, um als Swipe anerkannt zu werden. Diese Anforderungen werden vom Entwickler der Applikation definiert und umfassen Informationen wie die minimale und maximale

Entfernung zwischen Beginn und Ende des Swipes, wieviel Zeit während des Swipes vergehen darf und in welche Richtung der Swipe ausgeführt werden darf.

Erfüllt der ausgeführte Swipe alle Anforderungen, wird das Senden des Bildes initiiert. Das Bild wird auf dem Empfängergerät angezeigt, sobald es angekommen ist (Abbildung 15, 4). Somit ist die Interaktion beendet.

Wird mindestens eine der Anforderungen nicht erfüllt, so wird das Bild nicht versendet um versehentliche Sendungen zu vermeiden – Die Applikation befindet sich wieder im Ausgangszustand.

3.1.1 Microinteractions in Swipe-to-Give

Da nun der gesamte Ablauf der Swipe-to-Give Interaktion aufgezeigt wurde, können die einzelnen Microinteractions identifiziert und definiert werden. Dabei gehört zu jeder Aktion in Abbildung 15 eine Microinteraction.

Microinteraction 1: Touch

Die erste Microinteraction, die der Nutzer ausführt ist das Berühren des Bildschirms. Wird die Berührung auf dem Bild ausgeführt, zählt sie als erfolgreich.

Microinteraction 2: Move

Die zweite Microinteraction ist das Bewegen des Fingers über den Bildschirm.

Microinteraction 3: Release

Die dritte Microinteraction entsteht durch das Beenden des Swipes. Hier wird durch die Regeln der Microinteraction entschieden ob der Swipe korrekt ausgeführt worden ist. Ist dies der Fall, wird positives Feedback gegeben und die Sendung ausgeführt. Wurde mindestens eine der Anforderungen an einen Swipe nicht erfüllt, wird negatives Feedback gegeben und es wird der Ausgangszustand wiederhergestellt.

Microinteraction 4: Receive

Die vierte und letzte Microinteraction ist das Empfangen und Anzeigen des gesendeten Bildes auf dem Empfängergerät. Diese ist auch die einzige Microinteraction, die sich auf dem Empfängergerät abspielt.

Man kann also sehen, dass ein Swipe, eine so einfache und schnell ausgeführte Interaktion mit einem Gerät, in drei unterschiedliche Microinteractions mit ihren eigenen Auslösern, Regeln und Feedbacks aufgeteilt werden kann.

Es folgt eine Tabelle, die diese in Kapitel 2 definierte Struktur der vier Microinteractions übersichtlich darstellt. Loops und Modes werden für die hier verwendeten Microinteractions nicht benötigt, weshalb sie auch nicht in der Tabelle vertreten sind.

Tabelle 1: Struktur der Microinteractions

N°	Name	Trigger Type	Trigger	Rules	Feedback
1	Touch	Manual	Touch Event on Screen (Down)	-Touch must be on the image	Animation 3.2
2	Move	Manual	Touch Event on Screen (Move)	-N°1 has occured -N°3 has not occured	Animation 3.2.2
3	Release	Manual	Touch Event on Screen (Up)	-Swipelength OK -Swipeduration OK -Swipeorientation OK	Animation 3.2.3
4	Receive	System	Data Received	-Data is an image	-Animation 3.2.4 -Vibrate device

Wie schon erwähnt, muss der **Touch** auf dem Bild ausgeführt werden, um einen korrekten Swipe ausführen zu können. Das heißt, man hat zwei verschiedene Möglichkeiten:

1. Man kann den Swipe normal ausführen, wenn der **Touch** auf dem Bild war.
2. Es passiert nichts und man muss nochmal den **Touch** ausführen.

Man muss bedenken, dass man für beide Möglichkeiten Feedback liefern muss, damit der Nutzer vermittelt bekommt was er richtig, bzw. falsch gemacht hat. Wie auch schon in Kapitel 2 beschrieben, ist die primäre Aufgabe von Feedback die Vermittlung der Regeln der Microinteraction.

Das Besondere an **Move** ist, dass die Microinteraction nur ausgelöst wird, wenn zuvor ein **Touch** geschehen ist aber noch kein **Release.**

Wie man sieht, werden die Swipeanforderungen überprüft, sobald **Release** getriggert wird. Wird mindestens eine der Anforderungen durch den Swipe nicht erfüllt, so wird der Ausgangszustand wiederhergestellt. Auch hier sollte das Feedback vermitteln, was der Nutzer falsch gemacht hat - also welche der Anforderungen nicht erfüllt wurde. Zusätzlich muss noch entschieden werden was geschieht, wenn mehrere Anforderungen gleichzeitig nicht erfüllt wurden.

Es fällt auf, dass die Receive-Microinteraction die einzige ist, die durch einen System Trigger ausgelöst wird. Das liegt daran, dass die Nummern 1-3 zusammen den Swipe auf dem Sendergerät bilden und somit direkt vom Nutzer ausgeführt werden. **Receive** hingegen ist die einzige Microinteraction, welche sich auf dem Empfängergerät abspielt. Sie wird lediglich durch das Empfangen von Daten ausgelöst. Das bedeutet auch, dass **Receive** komplett abgekoppelt von den anderen drei Microinteractions, also dem gesamten Swipevorgang, ist. Jede Art von Datenübermittlung kann **Receive** auslösen, womit diese Microinteraction ohne weitere Anpassung auch in anderen Applikationen verwendet werden könnte.

Außerdem ist **Receive** die einzige Microinteraction, die mehr als Animationen als Feedback gibt. Das Vibrieren des Geräts soll dem Nutzer vermitteln, dass etwas empfangen wurde. Dies ist besonders effektiv, da der Nutzer dieses Feedback schon kennt, vor allem beim Empfangen von SMS. Um diesen Effekt noch zu verstärken sollte darauf geachtet werden, dass die Art und Länge der Vibration denen eines SMS-Empfangs gleicht.

Wie bereits erwähnt, wurden Animationen als primäres Feedback gewählt. Auf das Design dieser Animationen wird im folgenden Unterkapitel ausführlich eingegangen, wobei an passenden Stellen auf die benutzten Prinzipien aus Kapitel 2.2 verwiesen wird.

3.2 Design der Animationen

In jedem der folgenden Unterkapitel wird zunächst beschrieben, welche Informationen durch die jeweilige Animation vermittelt werden sollen. Daraufhin wird die Animation anhand von Screenshots genau gezeigt. Zuletzt wird auf eventuell genutzte Methoden aus Disneys Animationsprinzipien und Material Design (Siehe Kapitel 2.2) verwiesen und auf sonstige Besonderheiten des Designprozesses eingegangen.

Allgemein wurde beim Design der Animationen darauf geachtet, die in Kapitel 2.2.2 beschriebenen Material Design Guidelines einzuhalten. Dadurch wird versucht, eine familiäre User Experience zu schaffen, da viele der meistgenutzten Android Apps Material Design verwenden.

3.2.1 Microinteraction 1: Touch

Die erste Animation wird durch die Berührung des Bildschirms ausgelöst, sofern der Touch auf dem Bild stattfindet, welches der Nutzer versenden möchte. Diese Feedbackanimation soll dem Nutzer vermitteln, dass er ein sendbares Bild berührt hat. Sollten sich mehrere Bilder auf dem Bildschirm befinden, erhält er so auch die Information, welches der Bilder aktiviert wurde, da die Animation nur auf dem aktivierten Bild abgespielt wird. Zusätzlich sollte die Touch-Animation andeuten, was im weiteren Verlauf zu tun ist, um das Bild zu versenden. Folgend wird die gesamte Touch-Animation beschrieben.

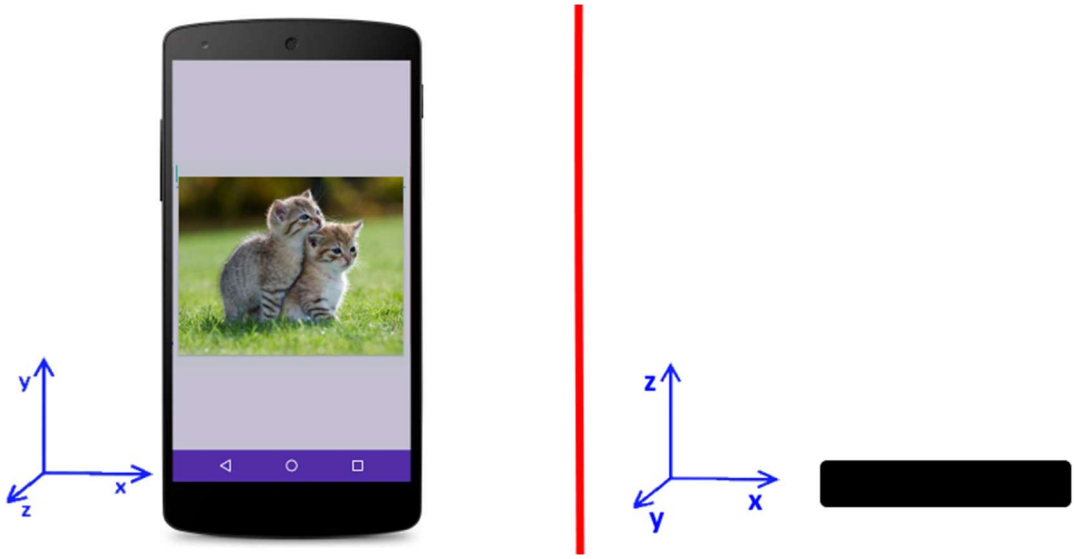

Abbildung 16: Ausgangszustand

Vor Beginn der Nutzerinteraktion befindet sich ein Bild auf dem Display, welches man versenden kann (Siehe Abbildung 16, links). In blau wurden zum besseren Verständnis die Bewegungsachsen eingezeichnet.

Das Bild ist leicht in Z-Richtung angehoben (Siehe Abbildung 7), um zu vermitteln, dass es sich um ein interaktives Element der UI handelt; nicht um einen Teil des Hintergrundes.

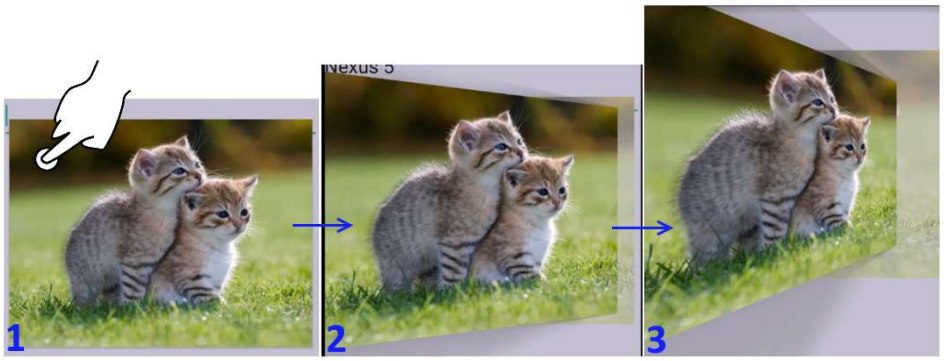

Abbildung 17: Touch Animation (Teil 1)

Berührt der Nutzer nun den Teil des Displays, auf dem sich das Bild befindet (Siehe Abbildung 17, **(1)**), wird die Animation abgespielt. Das Bild hebt weiter in Z-Richtung ab und beginnt eine Drehung um die Y-Achse (Siehe Abbildung 17, **(2)** u. **(3)**). Man beachte, dass der Schatten von Android automatisch mitanimiert wird. In **(2)** und **(3)** kann man im Hintergrund eine ausgegraute Kopie des animierten Bildes sehen. Diese Kopie wird nicht mitanimiert, sondern bleibt immer am ursprünglichen Platz des Originals.

Hat sich das Bild um 90° Grad um die Y-Achse gedreht, so verschwindet es für einen kurzen Moment und es ist nur die ausgegraute Kopie des Bildes zu erkennen.

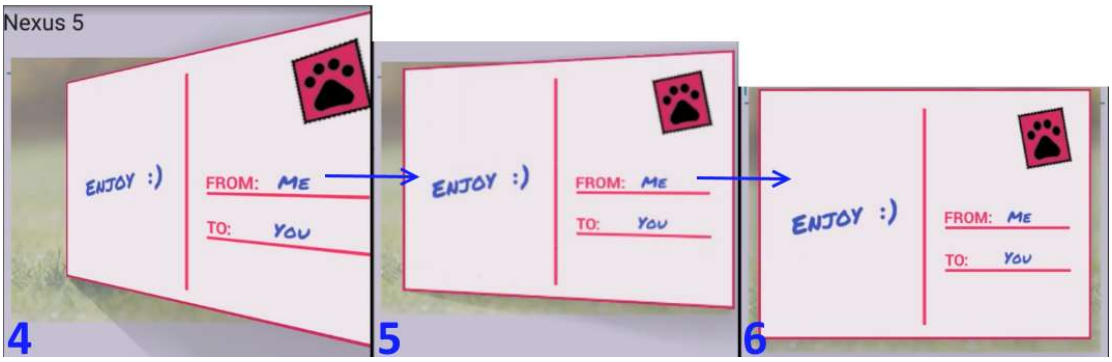

Abbildung 18: Touch Animation (Teil 2)

Dreht sich das Bild nun weiter, erscheint dessen Rückseite. In Abbildung 18 sieht man, dass die Rückseite des Bildes eine Postkarte ist. Sie dreht sich um weitere 90° um die Y-Achse und schwebt viel höher über dem Hintergrund als das inaktive Bild in Abbildung 16. Sobald die gesamte Rotation abgeschlossen wurde, ist die Touch-Animation beendet.

Wie zuvor erwähnt, soll diese Animation Material Design anwenden. Da *Material* einigen Regeln unterliegt, musste eine Animation gewählt werden, die keine dieser Regeln verletzt. Eine frühere Version der Touch-Animation sollte nur die in Kapitel 2.2.2 aufgeführten Animationen **Lift on Touch** und **Surface Reaction** benutzen. In der gesamten Swipe-to-Give Interaktion würde dies jedoch etwas zu wenig wirken.

Eine Verwandlung des Bildes in eine Postkarte sollte zwei Probleme lösen: Erstens soll die Swipe-to-Give Interaktion etwas Besonderes sein und dem Nutzer, über die angenehme User Experience einer Material Design App hinaus, Spaß machen. Zweitens sollte das Feedback der Touch-Microinteraction schon darauf hinweisen, dass etwas versendet werden wird.

Streng nach Material Design Guidelines, kann sich *Material* zwar frei verformen und bewegen, nicht jedoch biegen oder falten. Ohne Verlassen des Bildschirms oder des Hinzufügens eines weiteren Elements, das die Sicht verdeckt, hat man dadurch nur wenige Möglichkeiten, die Verwandlung eines Bildes zu einer Postkarte darzustellen.

Postkarten in der realen Welt haben oft Fotographien auf einer Seite. *Material* ist ein dünnes, an Papier erinnerndes Blatt. Man kann sich vorstellen, dass das Bild, welches der Benutzer versenden möchte (In unserem Beispiel das Katzenfoto), auf einem Blatt *Material* abgebildet wird. Dieses *Material* hat dann auf einer Seite das zu versendende Bild und auf der anderen eine stilisierte Zeichnung einer Postkarte. Das Wenden des Bildes zu der Postkarte ergibt in diesem Kontext Sinn.

Diese Animation soll dem Nutzer Spaß machen und ihn überraschen, da Wende-Animationen relativ selten verwendet werden. Außerdem ist eine Postkarte etwas, das man im realen Leben versendet. Damit verstärkt man die Metapher von Swipe-to-Give und gibt dem Nutzer einen Hinweis auf die nächste Microinteraction, die er ausführen muss: **Move**. Um die Postkarte zu versenden, muss er sie vom Bildschirm verschwinden lassen. Dafür muss er sie jedoch zuerst bewegen.

Die gesamte Touch-Animation ist eine Form von **Responsive Interaction**, da sie dem Nutzer sofortiges Feedback gibt. Würde sich das Bild schlagartig in eine Postkarte verwandeln, könnte das den Nutzer verwirren. Die Wende-Animation erklärt, dass das Bild und die Postkarte ein einziges Objekt sind. Das Versenden einer Postkarte ist eine verständliche Metapher, da es nah an der Realität ist.

Eine Besonderheit der Wendeanimation wurde zuvor kurz erwähnt: das Bild dreht sich um 90° und verschwindet dann kurz. Abbildung 19 zeigt diesen Moment aus drei Perspektiven um besser erklären zu können was geschieht.

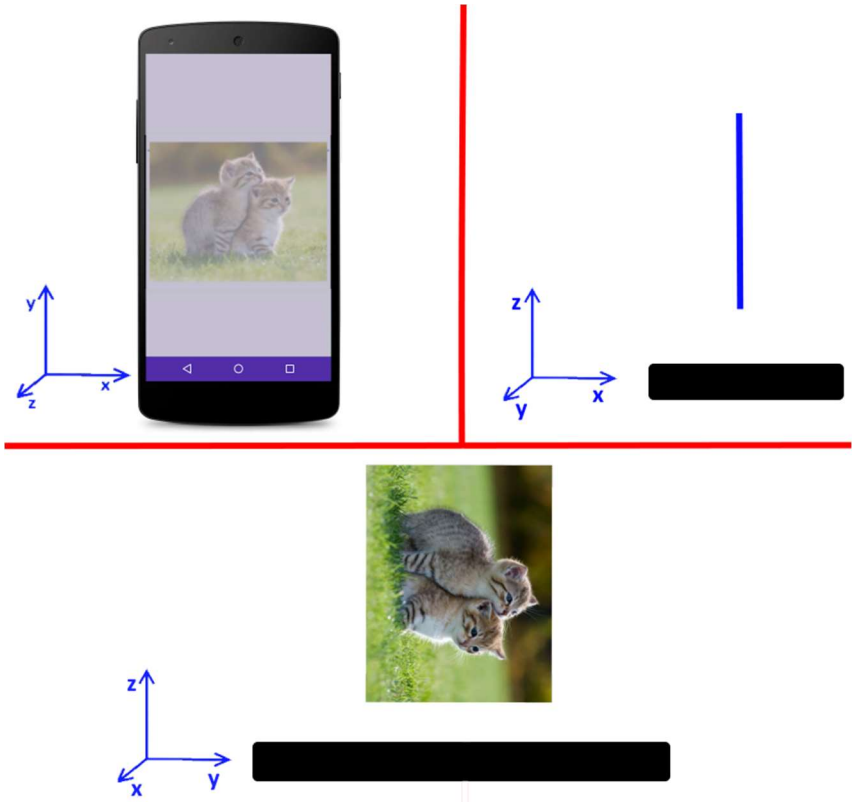

Abbildung 19: Perspektiven zum Zeitpunkt des Verschwindens des Bildes

In der Draufsicht **(oben links)** kann man das originale Bild nicht mehr sehen, da es in diesem Moment parallel zur Z-Achse steht. Wie bereits in Kapitel 2.2.2 beschrieben, hat *Material* keine Tiefe und kann aus diesem Grund von der Seite nicht gesehen werden. Schaut man sich das Handy jedoch von der Seite an und stellt sich den virtuellen Raum über dem Display vor **(unten)**, erkennt man, dass sich das Bild um 90° Grad

gedreht hat. Die Sicht von unten auf das Gerät **(oben rechts)** zeigt nochmals, dass das Bild senkrecht zum Display steht.

3.2.2 Microinteraction 2: Move

Die Move-Microinteraction wird nur ausgelöst, wenn der Nutzer zuvor die Touch-Microinteraction ausgeführt hat und sein Finger sich nach Abspielen der Animation noch immer auf dem Bildschirm befindet. Also wurde **Touch** ausgeführt, nicht jedoch **Release**. Das heißt, zu Beginn der Feedback-Animation von **Move**, befindet sich die Applikation im letzten Zustand der Touch-Animation (Siehe Abbildung 18, **6**; Abbildung 20).

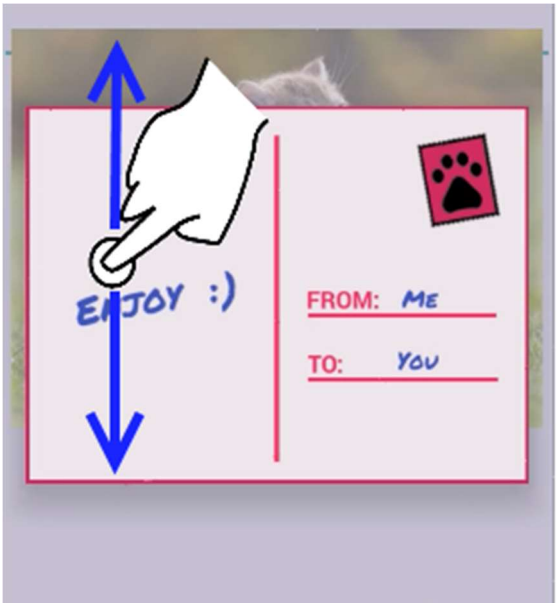

Abbildung 20: Move-Animation

Beginnt der Nutzer nun seinen Finger auf dem Bildschirm zu bewegen, wird die Move-Animation abgespielt. Das Ziel dieses Feedbacks ist es, dem Nutzer zu vermitteln, dass das Bewegen des Fingers die nächste richtige Aktion ist, um einen Swipe-to-Give auszuführen. Zusätzlich soll es dem Nutzer das Gefühl geben, die Postkarte in der Hand zu halten. Dadurch hat er die Kontrolle über die Animation, was zum herumexperimentieren anregen und somit den Spaß an der Swipe-to-Give Interaktion erhöhen soll. Dafür muss für jede Bewegung des Fingers die Postkarte mitanimiert werden, um sofortiges Feedback zur Nutzerinteraktion liefern zu können und somit eine **Responsive Interaction** darzustellen.

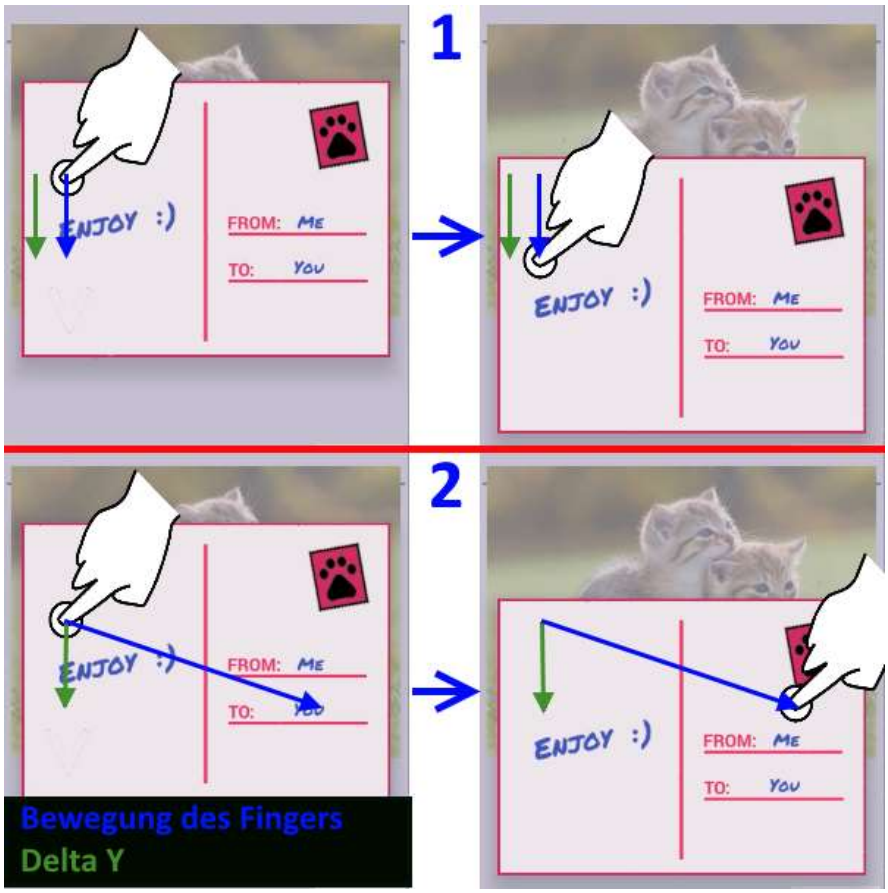

Abbildung 21: Drag-and-Drop in Y-Richtung

Als Move-Feedback wurde deshalb eine Drag-and-Drop Animation gewählt, welche auf die Y-Richtung beschränkt ist. Das heißt, die Postkarte folgt dem Finger, wobei Bewegungen in X-Richtung ignoriert werden. Betrachtet man Abbildung 21, so sieht man, dass das Resultat in **(1)** und **(2)** identisch ist, obwohl die jeweilige Bewegung des Fingers sehr unterschiedlich ist. Die grünen Pfeile zeigen jeweils die Bewegung in Y-Richtung an. Da diese in **(1)** und **(2)** gleich ist, kommt es zum identischen Resultat.

Die Beschränkung der Drag-and-Drop Animation auf die Y-Achse teilt dem Nutzer mit, dass er die Postkarte entweder nach oben oder unten bewegen muss um einen Swipe-to-Give auszuführen. Wenn man die Metapher des Versendens betrachtet, ergibt die Bewegung nach oben, also weg vom Nutzer, mehr Sinn: Versendet man eine Postkarte in der Realität, schmeißt man sie in einen Briefkasten. Die Bewegungsrichtung des Arms zeigt vom Körper weg. Ein Swipe nach unten hingegen führt den Finger in Richtung des Nutzers, die Bewegungsrichtung zeigt zum Körper.

Somit wird mit der Beschränkung der Bewegung auf die Y-Achse, im Zusammenspiel mit der Metapher des Versendens einer Postkarte, dem Nutzer mitgeteilt was zu tun ist, um das Bild zu versenden: Es muss ein Swipe nach oben ausgeführt werden.

3.2.3 Microinteraction 3: Release

Die Release-Microinteraction wird ausgelöst, sobald der Nutzer seinen Finger wieder vom Bildschirm entfernt. Wie bereits erwähnt, kann dies entweder durch ein Hochheben des Fingers oder durch das Verlassen der Bildschirmgrenze geschehen. **Release** kann also direkt nach **Touch** aktiviert werden, indem das Bild einfach angetippt wird, ohne dass der Nutzer seinen Finger anderweitig bewegt. Die zweite Möglichkeit ist, dass der Nutzer sowohl die Touch- als auch die Move-Microinteraction zuvor ausgeführt hat, wie es bei einem Swipe der Fall wäre.

Eine Besonderheit der Release-Microinteraction ist, dass sie die einzige ist, die zwei verschiedene Feedback-Animationen benötigt. Das liegt daran, dass nach dem Auslösen von **Release** erst überprüft wird, ob die ausgeführte Kette von Microinteractions (**Touch -> (Move) -> Release**) den Anforderungen eines Swipes entspricht.

Swipe-Anforderungen nicht erfüllt

Wurde mindestens eine der Anforderungen nicht erfüllt, zum Beispiel wenn die Entfernung zwischen Touchpunkt und Releasepunkt zu kurz war, begibt sich die App wieder in den Anfangszustand (Siehe Abbildung 14) zurück. Dies wird mit zwei parallel ablaufenden Animationen erreicht (Siehe Abbildung 22). Die erste ist das Gegenstück zu der Touch-Animation. Das heißt, die Postkarte dreht sich in die entgegengesetzte Richtung um die Y-Achse. Gleichzeitig bewegt sich das Bild wieder an seine ursprüngliche Position zurück, falls zuvor die Move-Microinteraction benutzt wurde.

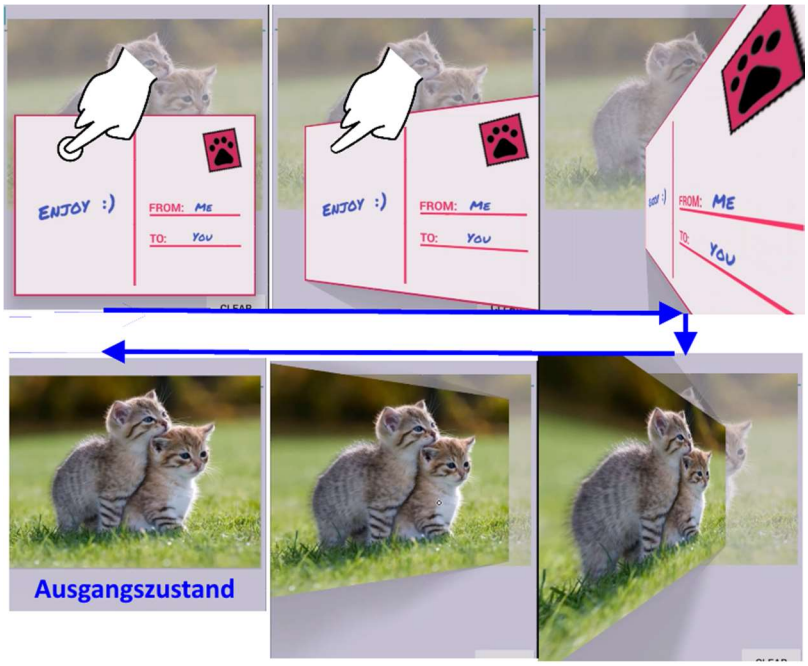

Abbildung 22: Release-Animation bei Nichterfüllung der Swipe-Anforderungen

Durch die Rückkehr zum Ausgangszustand wird dem Nutzer mitgeteilt, dass er den Swipe inkorrekt ausgeführt hat und dass er es nochmals versuchen kann. Für diesen Schritt wurden bewusst keine weiteren Animationen eingeführt, sondern nur die bisherigen rückgängig gemacht. Würden neue Animationen auftreten, so würde dies dem Nutzer signalisieren, dass er Fortschritt macht. Das wäre das genaue Gegenteil dessen, was man dem Nutzer mitteilen muss.

Swipe-Anforderungen erfüllt

Wurden alle Swipe-Anforderungen erfüllt, wird die Sende-Animation ausgeführt. Diese Animation soll einige Informationen vermitteln und gleichzeitig dynamisch wirken, um dem Nutzer eine angenehme User Experience zu bieten, die er möglichst wiederholen möchte.

Die erste und wichtigste zu vermittelnde Information ist, dass der Nutzer den Swipe korrekt ausgeführt hat und das Bild nun versendet wird. Die zweite Information ist, dass das versendete Bild nicht vom Gerät des Nutzers verschwunden ist, sondern lediglich eine Kopie versendet wurde. Außerdem soll es nach Abschluss der Animationen möglich sein, das Bild nochmals zu versenden. Im Folgenden wird der gesamte Ablauf der Sende-Animation beschrieben.

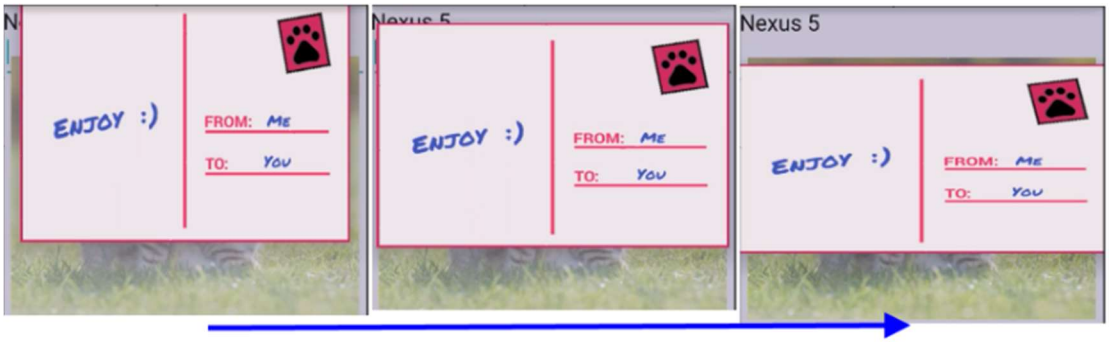

Abbildung 23: Sende-Animation Teil 1: „Stauchung“

Nachdem die Release-Microinteraction ausgelöst wurde und die Swipe-Anforderungen erfüllt wurden, beginnt der erste Teil der Sende-Animation. Die Postkarte beginnt sich in Y-Richtung zu stauchen. Das heißt, sie wird weniger hoch und dafür noch breiter (Siehe Abbildung 23). Man beachte, dass der untere Rand der Postkarte ungefähr auf einer Höhe bleibt.

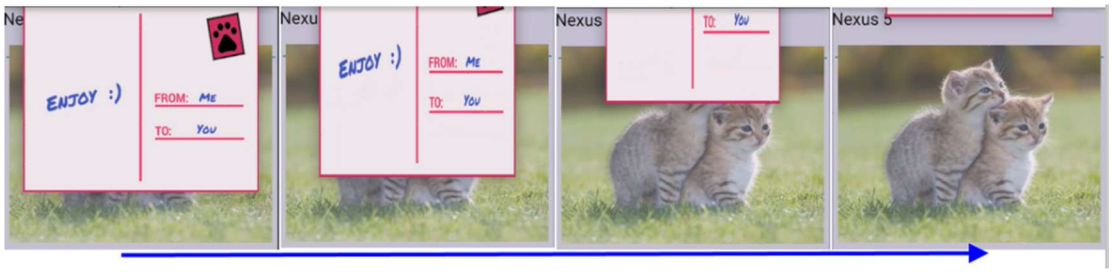

Abbildung 24: Sende-Animation Teil 2: „Streckung“ und Verlassen des Bildschirms

Nachdem die maximale Stauchung der Postkarte erfolgt ist, beginnt die gegenteilige Aktion: Die Karte streckt sich in Y-Richtung und beginnt nach oben den Bildschirm zu verlassen (Siehe Abbildung 24). Die Bewegung nach oben folgt einer spürbaren Beschleunigungskurve, das heißt sie fängt langsam an und wird immer schneller, bis die Postkarte nicht mehr zu sehen ist. Nachdem sie den Bildschirm vollständig verlassen hat, beginnt der dritte Teil der Sende-Animation: Die Rückführung der App in den Ausgangszustand.

Abbildung 25: Sende-Animation Teil 3: Wiederherstellung des Ausgangszustands

Nachdem die Postkarte den Bildschirm verlassen hat, ist nur noch die ausgegraute Kopie des originalen Bildes auf dem Display zu sehen (Siehe Abbildung 25, links). Zu diesem Zeitpunkt entfernt sich der graue Schleier, der über dem Bild liegt, langsam. Sobald dieser Prozess abgeschlossen ist, befindet sich alles wieder im Ausgangszustand. Das heißt auch, dass der Nutzer die Swipe-to-Give Interaktion nochmals von vorne ausführen kann, wieder angefangen mit der Touch-Interaction (wobei wieder eine ausgegraute Kopie des Bildes erstellt werden würde).

Dadurch, dass die Postkarte den Bildschirm in die gleiche Richtung verlässt, in die der Nutzer sie bewegt hat, bekommt er die Bestätigung, dass die Interaktion korrekt ausgeführt wurde. Somit wurde die wichtigste Aufgabe dieser Feedback-Animation erfüllt.

Um die Swipe-to-Give Interaktion etwas reizvoller zu gestalten, wurde auf einige Prinzipien und Methoden aus Kapitel Animationen2.2 zurückgegriffen. Die auffälligsten Teile der Animation sind die, an der die Postkarte gestaucht und gestreckt wird. Hierfür wurden gleich zwei von Disneys Animationsprinzipien benutzt: Erstens sollte der Nutzer nicht verpassen oder missverstehen, wenn die Postkarte den Bildschirm verlässt. Aus diesem Grund staucht sie sich zuerst und signalisiert somit, dass sie sich gleich in die entgegengesetzte Richtung zur Stauchung bewegen wird. Dies ist eines der 12 Prinzipien der Animation: **Anticipation**. Um diesen Teil der Animation, der den Nutzer auf die nächste Aktion vorbereitet, umzusetzen, wurde ein weiteres der Animationsprinzipien benutzt. Die Stauchung und Streckung der Postkarte verhält sich nach den Regeln von **Squash and Stretch**. Die Metapher hinter der Stauchung war die Vorbereitung auf einen hohen Absprung: Es soll so scheinen, als würde die Postkarte in die Knie gehen um dadurch mehr Kraft für den Absprung zu haben. Aus diesem Grund wurde versucht, die Unterkante der Postkarte an der gleichen Stelle zu behalten. Dies verstärkt das Bild des Absprungs vom Boden, da sich der Boden in der Realität ebenfalls nicht bewegt.

Da die Karte für **Anticipation-Animation** weniger hoch werden musste, war es wichtig sie gleichzeitig breiter werden zu lassen, da **Squash and Stretch** besagt, dass das Volumen des animierten Objekts unverändert bleiben sollte.

Die Streckung der Postkarte soll ein Gefühl von Beschleunigung und Geschwindigkeit erzeugen, wie es auch in Abbildung 1: **(2)**, **(4)** und **(5)** zu sehen ist. Sie einfach tatsächlich schneller beschleunigen zu lassen würde nicht den gewünschten Effekt erbringen, da die Postkarte dadurch zu kurz auf dem Bildschirm zu sehen wäre.

Bildlich gesehen verlässt die Postkarte das Display des eigenen Geräts und soll bis hin zu dem Empfängergerät fliegen. Um dieses Bild zu verstärken, wird die Karte immer weiter beschleunigt, bis sie komplett den Bildschirm verlassen hat. Dieses Konzept wurde bereits in Material Designs Unterkapitel **Authentic Motion** beschrieben.

3.2.4 Microinteraction 4: Receive

Die Receive-Microinteraction wird auf dem Empfängergerät ausgelöst, sobald das Gerät Daten empfängt und diese Daten eine Bilddatei darstellen. Wie bereits erwähnt, ist dies die einzige Microinteraction innerhalb von Swipe-to-Give, die keine Nutzerinteraction benötigt. Sie wird durch einen System Trigger ausgelöst.

Das Feedback dieser Microinteraction soll in erster Linie dem Nutzer vermitteln, dass er eine Bildnachricht empfangen hat. Auch auf der Empfängerseite ist jedoch eine gute User Experience und Spaß an den Animationen gewünscht.

Abbildung 26: Ausgangszustand Empfängergerät

Zu Anfang der Microinteraction befindet sich das Empfängergerät in dessen Ausgangszustand (Siehe Abbildung 26). Die Buttons im unteren Teil des Bildschirms sind Teil des Prototypen und können an dieser Stelle ignoriert werden.

Die Feedback-Animation zu **Receive** dauert etwa 2,7 Sekunden und kann in vier Teile unterteilt werden: 1. Bewegung nach unten, 2. Bewegung nach oben, 3. Wenden und 4. Absenken.

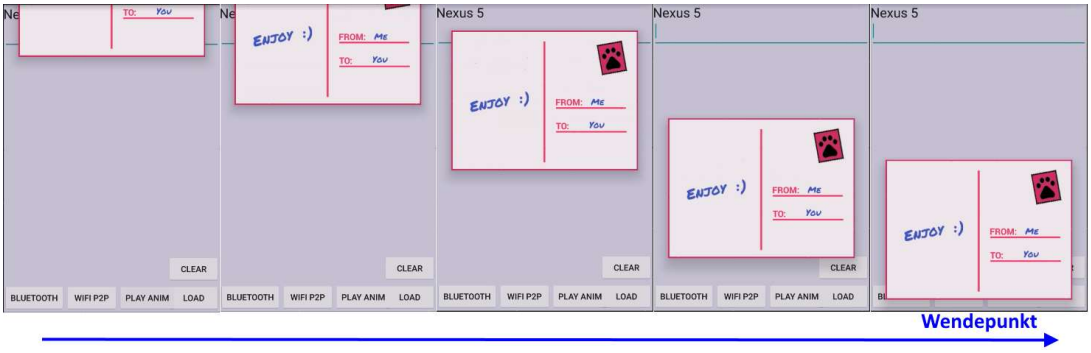

Abbildung 27: Receive-Animation Teil 1

Beim Empfangen eines Bildes, wird eine Postkarte mit hoher Geschwindigkeit von oben in den Bildschirm eingeschoben und dann stark abgebremst (Siehe Abbildung 27). Die Karte wird immer langsamer, bis sie in der Nähe vom unteren Bildschirmrand kurz zum Stillstand kommt (**Wendepunkt**).

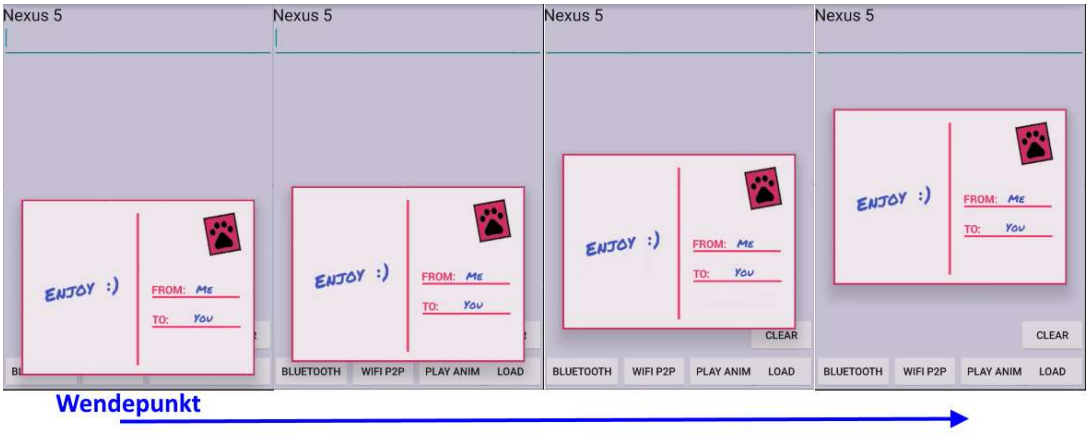

Abbildung 28: Receive-Animation Teil 2

Von diesem Moment an bewegt sich die Postkarte wieder nach oben, jedoch viel langsamer als zuvor (Siehe Abbildung 28). Sie kommt in der Mitte des Bildschirms zum Stillstand und wird gewendet, wie es auch schon bei der Touch-Animation der Fall war (Siehe Abbildung 29).

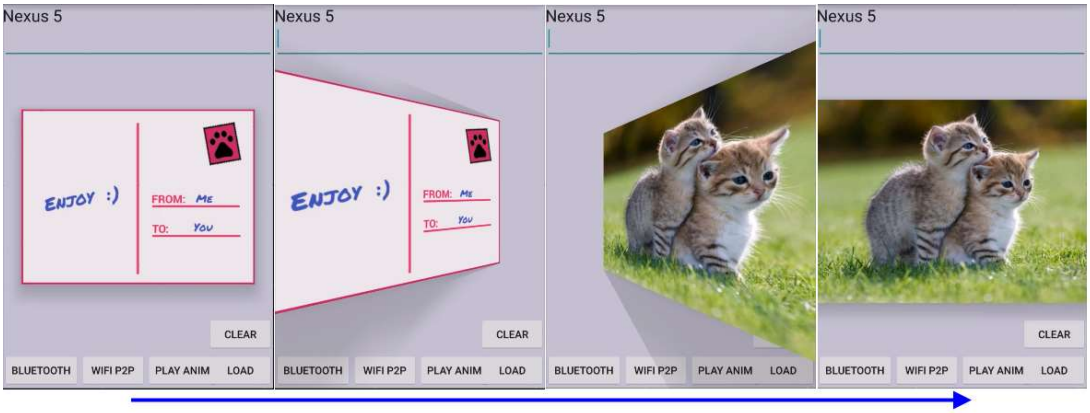

Abbildung 29: Receive-Animation Teil 3

Nachdem die Postkarte sich zu dem empfangenen Bild gewendet hat, verbleibt das Bild für einen Moment in dieser Position und senkt sich dann langsam ab um dann in seiner finalen Position zu verharren (Siehe Abbildung 30).

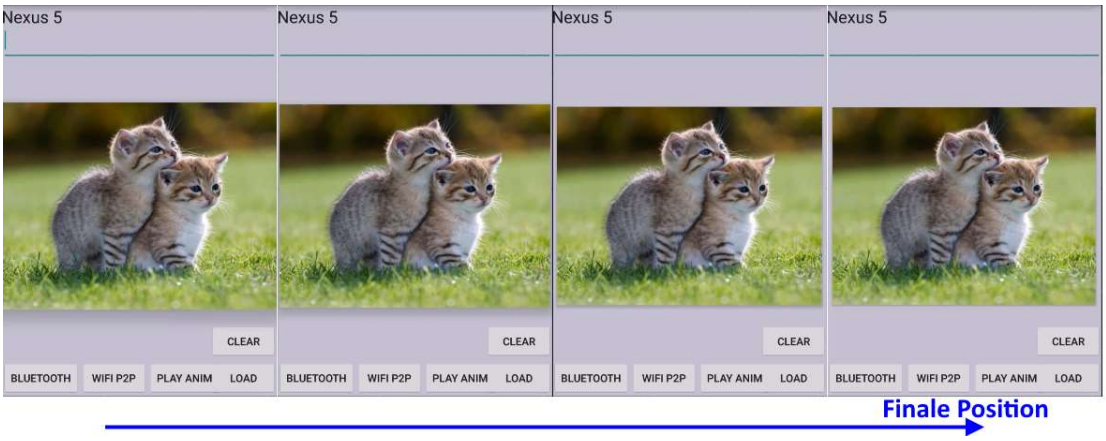

Abbildung 30: Receive-Animation Teil 4

Wie in Kapitel 3.1.1 erwähnt wurde, hat die Receive-Microinteraction zwei Arten von Feedback: die Animationen und die Vibration des Empfängergeräts. Die Vibration soll schon die Assoziation wecken, dass etwas empfangen wurde. Dieser Effekt soll noch mit zwei Teilen der Animation verstärkt werden.

Dass eine Postkarte den Bildschirm betritt, während das Handy vibriert, ist der erste dieser Teile. Die hohe Geschwindigkeit, mit der die Postkarte ankommt, soll zusätzlich signalisieren, dass diese von weit weg kommt und nicht nur ein Teil der lokalen Applikation ist (zum Beispiel eine Fehlermeldung). Das Überfliegen des Zieles, also der zu lange „Bremsweg“ der Karte lässt die Receive-Animation noch schneller wirken, da man sich vorstellen kann, dass sie bereits vor Betreten des Displays angefangen hatte zu bremsen.

Das kurze Verharren des Bildes nach der Wendung baut Spannung auf, bevor es sich langsam absenkt und letztendlich wie ein natürlicher Teil der Applikation aussieht. Dies soll eine angenehme Wirkung auf den Nutzer haben und so den Spaß beim Benutzen der Applikation erhöhen.

An dieser Stelle zeigt sich auch, dass es für das Verständnis und das Gefühl, welches Animationen vermitteln sollen, sehr wichtig ist, die richtigen Timings zu finden (Siehe Kapitel 2.2.1, Timing). Dieser letzte Teil der Animation wirkte anfangs etwas zu unspektakulär. Durch Experimentieren wurde der kurze Stillstand des Bildes und die viel langsamere Absenkung entdeckt. Dadurch fühlt sich die gesamte Animation reizvoller an.

4 Ausblick

Das Kernstück der Swipe-to-Give Interaktion ist eine scheinbar simple und schnell ausgeführte Geste. Im Zuge dieser Arbeit konnten vier *Microinteractions* identifiziert werden, die daraufhin spezifisch ausgearbeitet wurden. Durch diesen Prozess wurden viele Details aufgedeckt, die durch das einfache Betrachten eines Swipes verborgen bleiben. Diese Details erlauben eine nuancierte Analyse und somit einen strukturierteren Designprozess. Animationen sollten nicht einfach nur schön zu betrachten sein, sondern erfüllen einen wichtigen Zweck in einer Applikation: Sie liefern zur richtigen Zeit Feedback und erklären dem Nutzer auf diese Weise, was gerade geschieht, was er zu tun hat oder ob er etwas falsch gemacht hat. Diese Erkenntnis ist eines der wichtigsten der hier erarbeiteten Konzepte.

Abschließend werden einige mögliche zukünftige Verfeinerungen des in Kapitel 3 entwickelten Interaktionskonzeptes vorgestellt.

Die wichtigste Änderung, die man vornehmen könnte, liegt beim Feedbackdesign der Touch-Microinteraction (Siehe Kapitel 3.1.1, Microinteraction 1: Touch). Nach einigen vorläufigen Testläufen mit Probenutzern stellte sich heraus, dass die Animation allein nicht ausreichend Erklärung bietet, was der nächste Schritt ist, den der Nutzer ausführen muss. Eine Möglichkeit wäre das einblenden eines, nach oben zeigenden, Pfeiles im Hintergrund, damit der Nutzer sieht in welche Richtung er die Postkarte bewegen muss.

Eine weitere Verbesserung der User Experience könnte in der Release-Microinteraction erreicht werden. Bei einem nicht erfolgreichen Swipe werden die beiden zuvor ausgeführten Feedback-Animationen wieder rückwärts abgespielt. Besser wäre, wenn die Release-Animation so modifiziert werden würde, dass sie dem Nutzer vermittelt warum genau der Swipe nicht erkannt wurde. Dies könnte beispielsweise durch eine Meldung wie „Swipe faster next time!“ erreicht werden. Um störende Fehlermeldungen zu vermeiden, könnte dieser Text auf der Postkarte erscheinen, bevor sie sich wieder wendet.

Alle weiteren Änderungsvorschläge sind kosmetischer Natur und würden sich nur indirekt auf die User Experience auswirken. Zum Beispiel könnte man zu den Sende- und Empfangsanimationen Geschwindigkeitslinien oder Wind hinzufügen, um das Gefühl von Geschwindigkeit weiter zu verstärken (Siehe Kapitel 2.2.1, Secondary Action).

Eine weitere Idee wäre die Namen der Geräte auf die Postkarte zu schreiben, anstelle von „Me“ und „You“.

Zusätzlich könnte man bei allen Animationen noch etwas am Timing feilen, um die bestmögliche Erfahrung zu schaffen. Wie bereits in Kapiteln 2.2.1 und 3.2.4 erwähnt, ist das Timing ein sehr wichtiger Faktor bei der Wahrnehmung von Animationen.

Abbildungsverzeichnis

Literaturverzeichnis

[AVG] AVG Labs: Alarm Clock Xtreme Free +Timer. [ONLINE] https://play.google.com/store/apps/details?id=com.alarmclock.xtreme.free&hl=en (Zuletzt besucht: 29.02.16)

[GOO] Google, Inc.: Material Design Guidelines. [ONLINE] https://www.google.com/design/spec/material-design/introduction.html (Zuletzt besucht: 29.02.16)

[HAH15] HAHN, Alexander: Konzeption und Implementierung der Swipe-Geste zum Datentransfer im Multi-Screen Kontext. Bachelorarbeit an der HS Mannheim, Deutschland, 2015

[HAR] Hardy-infinity: Bluelight Filter for Eye Care. [ONLINE] https://play.google.com/store/apps/details?id=jp.ne.hardyinfinity.bluelightfilter.free&hl=en (Zuletzt besucht: 28.02.16)

[SAF14] SAFFER, Dan: Microinteractions – Designing with Details. Sebastopol, CA, 2014

[STE] DE STEFANO, Ralph A.: The Principles of Animation. [ONLINE] https://www.evl.uic.edu/ralph/508S99/index.html (Zuletzt besucht: 29.02.16)

[THO81] THOMAS, Frank; JOHNSTON, Ollie: The Illusion of Life: Disney Animation. New York, NY, 1981.

[WIK] Wikipedia: Animation. [ONLINE] https://de.wikipedia.org/wiki/Animation (Zuletzt besucht: 19.02.16)