

Dokumentation

GLOBETROTTER

STEFAN MAYER | ALI GÜMÜS | PHILIPP SPIEGEL | SIMON MOOSBRUGGER

Inhalt

1. Thema.....	2
1.1. Neuausrichtung nach ÖkoSwim	2
1.1.1. Warum nicht mehr ÖkoSwim	2
1.1.2. Analyse anderer Open Data Datenquellen.....	3
1.2. Globetrotter	3
2. Analyse - Technologien.....	4
2.1. Applikation	4
2.1.1. HTML 5 mit WebGL	4
2.1.2. Blender	4
2.1.3. Unity3D.....	5
2.2. Steuerung	5
2.2.1. Android Smartphone	5
2.2.2. iOS Smartphone.....	5
3. Paperprototypes und Interaktionen.....	6
3.1. Erste Skizzen	6
3.1.1. Auswahl von Kontinenten, Ländern und Indikatoren	6
3.1.2. Vergleich von Ländern.....	7
3.2. Skizzen zum neuen User Interface	8
3.2.1. Länderauswahl	8
3.2.2. Vergleich von Ländern.....	9
3.3. Interaktionen.....	10
3.3.1. Allgemeine Aktionen	10
3.3.2. Länderauswahl	11
3.3.2. Datenvisualisierung	14
4. Realisierung	15
4.1 Verwendete Tools	15
4.2. Unity-Anwendung.....	15
4.2.1. Ländererkennung und Koordinatenumrechnung.....	15
4.2.2. Anbindung zum Weltbank-Webservice.....	20
4.2.3. Chart-Webservice	20
4.3. Android-App	21
5. Herausforderungen	22
5.1. Webservice der Weltbank	22
5.2. 502 Bad Gateway.....	22
5.3. Textencoding in Unity.....	23
5.4. Smartphone, Datenübertragung	23

1. Thema

1.1. Neuausrichtung nach ÖkoSwim

1.1.1. Warum nicht mehr ÖkoSwim

Im Laufe des Projekts stellten wir fest, dass die zur Verfügung gestellten Daten vom Land Oberösterreich teilweise veraltet sind. Die Daten sind auch nicht vollständig, sondern nur sporadisch erfasst worden. Außerdem sind die Datenformate auch nicht immer gut maschinenlesbar.

Wir betrachteten natürlich auch andere Open Data Datensätze vom Land Oberösterreich, ob eine andere Möglichkeit für eine neue Entwicklung besteht. Da die Aktualität und Vollständigkeit im Kontext von Open Data ein großes Thema ist, haben wir uns im Rahmen des Coachings für eine komplette Neuausrichtung entschieden.

1.1.2. Analyse anderer Open Data Datenquellen

Während der Neuausrichtungsphase betrachteten wir unterschiedliche Open Data Datenquellen genauer. Hier ist eine Auswahl davon mit ihren Vor- und Nachteilen:

Land Vorarlberg	<ul style="list-style-type: none">- geringe Auswahl an Datensätzen- Datensätze enthalten nicht viel Informationen- alle Formate fast nicht maschinenlesbar (PDF)
Stadt Wien	<ul style="list-style-type: none">+ gut maschinenlesbar+ Daten werden aktualisiert- viele Applikationen bereits umgesetzt
Schweiz	<ul style="list-style-type: none">+ viele Daten+ Aktualität OK- Datenformat schwer maschinenlesbar
Weltbank	<ul style="list-style-type: none">+ viele verschiedene Datensätze+ gute Aktualität+ REST Webservice zur Datenabfrage- vereinzelt fehlen erfasste Länder oder Jahre

1.2. Globetrotter

Wegen der Fülle an Datensätzen und deren hohen Vollständigkeit und Aktualität, verwenden wir nun die Daten von der Weltbank (<http://data.worldbank.org>). Das neue Projekt *Globetrotter* dient dazu, auf einem Globus Länder auszuwählen und deren Kennzahlen miteinander zu vergleichen.

Das Ziel der Applikation ist, dass man die Welt auf einem Globus erkunden kann. Dabei soll grundlegendes geographisches Wissen sowie Detailwissen zu den Datenkategorien der Weltbank aufgebaut werden können. Als Einsatzgebiet sehen wir vor allem den Bereich des E-Learning vor.

2. Analyse - Technologien

2.1. Applikation

Für die Applikation werden 3D-Fähigkeiten benötigt. Folgende Technologien beziehungsweise Game Engines stehen unter anderen zur Auswahl:

- WebGL
- Blender
- Unity3D

2.1.1. HTML 5 mit WebGL

Die Recommendations von HTML 5 und WebGL stehen frei und plattformunabhängig zur Verfügung. Da sie aber noch relativ neu sind, können mehrere Schwierigkeiten auftreten. Nicht alle Webbrowser unterstützen HTML und WebGL im nötigen Ausmaß (Microsoft Internet Explorer kann kein WebGL; Apple Safari muss WebGL zuerst umständlich aktivieren). Zudem muss eine aktuellere Grafikkarte mit aktuellem Grafiktreiber am System installiert sein. Unter diesen Umständen würden viele potentielle User die Applikation nicht nutzen können. Zusätzlich haben die Teammitglieder kaum Erfahrung mit diesen Technologien. Zudem scheint die Performance nach eigenen Tests noch nicht ganz ausgereift zu sein.

2.1.2. Blender

Blender steht für verschiedene Plattformen (Linux, Mac, Windows) frei zur Verfügung. Es enthält neben der 3D-Modellierung und -Animation die sogenannte Blender Game Engine. Für die Programmierung stellt die Blender Game Engine eine Python API zur Verfügung. Da wir im Team aber keine Erfahrungen mit der Blender Game Engine und Python haben, haben wir uns gegen die Blender Game Engine entschieden.

2.1.3. Unity3D

Unity3D ist eine Game Engine. Neben der kostenpflichtigen Version Unity3D Pro gibt es die kostenfreie Version Unity3D Free. Seit der Programmversion 4.x ist für Unity3D Free lediglich ein kostenfreier Account beim Hersteller nötig. Mit Mac und Windows werden auch die zwei großen Plattformen unterstützt. Bei der Programmierung in Unity3D kann zwischen Boo, C# und JavaScript gewählt werden. Da Unity3D eine starke Verbreitung in der einschlägigen Branche aber auch darüber hinaus hat, bietet es sich an. Außerdem ist allen Teammitgliedern die Programmiersprache C# gut bekannt. Der Umfang der Dokumentation und eine aktive Community sorgen für viele Hilfestellungen. Leider fehlen die wichtigen Teile der .NET API zur Erzeugung und Bearbeitung der Grafik. Deshalb müssen die Diagramme selbst erstellt werden. Dennoch haben wir uns entschlossen, dass Unity3D für die Applikation zum Einsatz kommen soll.

2.2. Steuerung

Die Steuerung der Applikation soll nicht direkt am PC erfolgen. Stattdessen ist geplant, dass der User die Applikation über ein eigenes Gerät steuern kann. Neben Smartphones und Tablets könnten auch Bewegungs- und Tiefensensoren sowie Kameras zum Einsatz kommen. Dabei soll auf der Steuerungsseite keine Art der Datenvisualisierung geben.

2.2.1. Android Smartphone

Das mobile Betriebssystem Android läuft auf einer sehr hohen Anzahl an Smartphones weltweit. Somit können viele potenzielle User erreicht werden. Die nötigen Entwicklertools für Android stellt Google für die verschiedensten Desktopplattformen kostenlos zur Verfügung. Da auch eine gute Kenntnis über die Androidprogrammierung im Team vorhanden ist, ist Android die Plattform unserer Wahl.

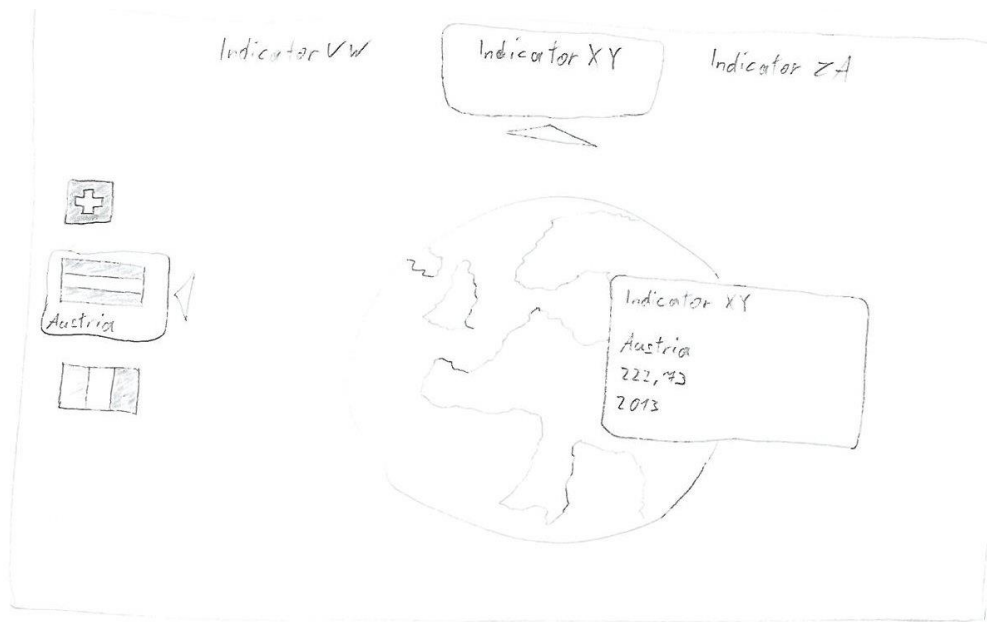
2.2.2. iOS Smartphone

Smartphones mit dem iOS Betriebssystem sind ebenfalls weit verbreitet. Allerdings ist derzeit keine Entwicklung einer entsprechenden Applikation nicht vorgesehen

3. Paperprototypes und Interaktionen

3.1. Erste Skizzen

3.1.1. Auswahl von Kontinenten, Ländern und Indikatoren



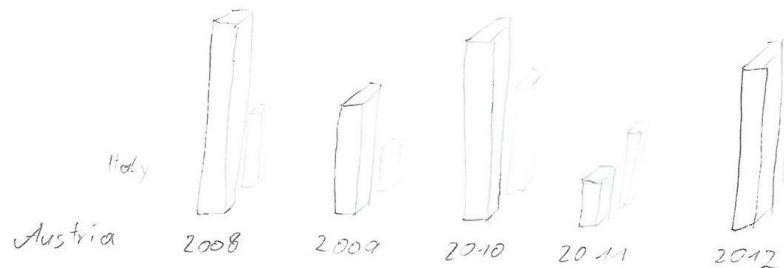
- ↑ ↓ Land / Kontinent auswählen; Erdkugel wird rotiert
- ← → Indikator auswählen; Daten werden geladen und eingeblendet

Die Auswahl von Ländern über die Kontinente und dann die Indikatoren der Weltbank enthält zu viele Ebenen. Das Auswahlverfahren basiert auf Listen. Dadurch muss der User möglicherweise sehr lange Listen „durchforsten“, bis er die gewünschten Länder erreicht. Die zwei Scroll Richtungen für verschiedene Kontexte können auch irritieren. Zusätzlich kommen hier keinerlei 3D-Fähigkeiten zum Einsatz.

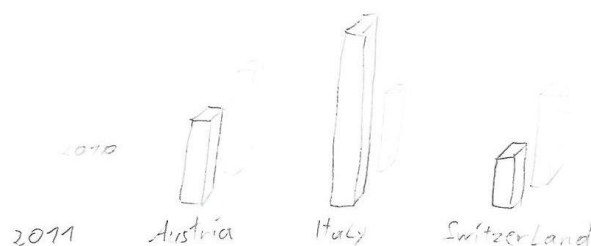
3.1.2. Vergleich von Ländern

Datacube für Historie / Vergleich von Ländern

Dimension land:



Dimension Jahr:



←→ Dimension wechseln

1D innerhalb der Dimension navigieren

? Data Cube notieren

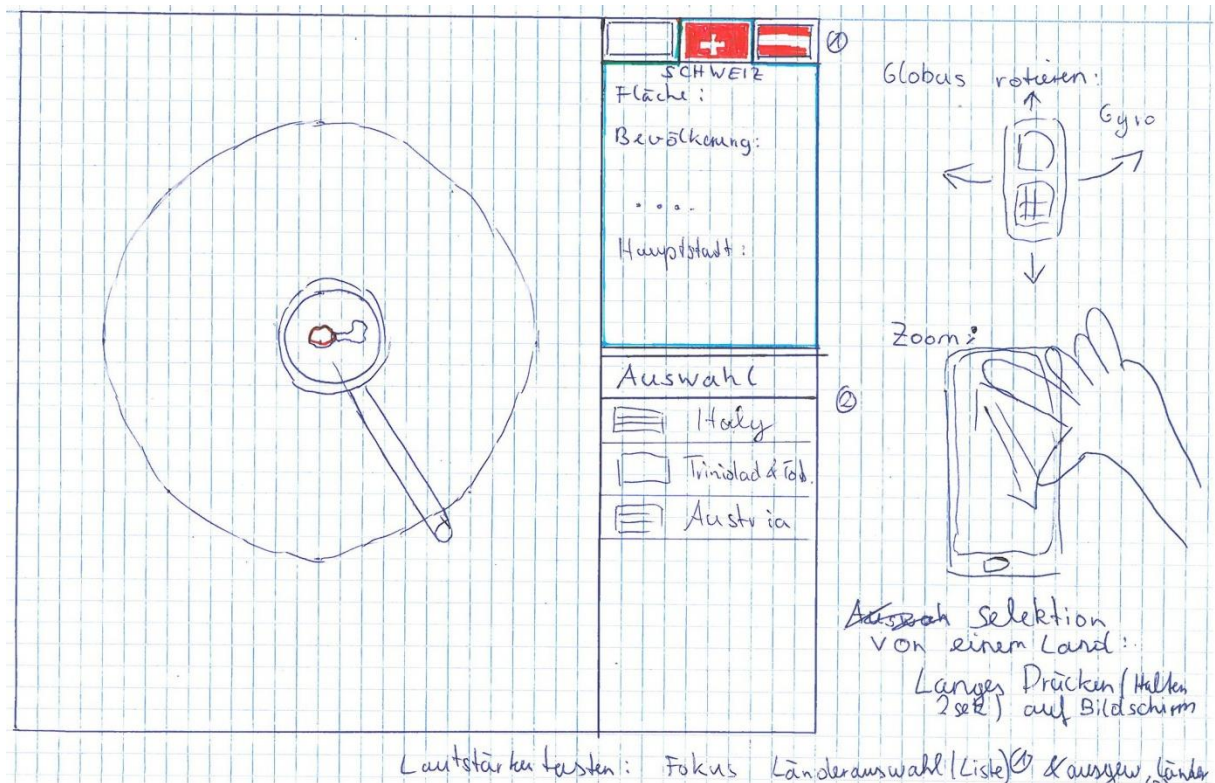
Im Datenvergleich enthielt die erste Skizze einen Data Cube. Dieser kann wie ein Würfel vorgestellt werden, auf jeder Seite eine andere Sicht auf die Daten bietet. In der Umsetzung kommen zwei Datensichten in Frage. Der User kann die Daten der ausgewählten Länder zu einem Jahr betrachten. Die zweite Datensicht zeigt die Daten eines Landes über mehrere Jahre an.

Durch Rotation kann zwischen den Datensichten gewechselt werden. Eine Bewegung in der Tiefe soll je nach Datensicht ein Navigieren zwischen Jahren beziehungsweise Länder ermöglichen.

Dieser Ansatz stellte sich aber nach ersten Versuchen als ungeeignet heraus. Durch die drei Dimensionen enthält das Bild viele Daten im Hintergrund, die für den User zu diesem Zeitpunkt nicht relevant sind, dies könnte ihn irritieren. In manchen Situationen wäre ein Liniendiagramm besser geeignet als ein Balkendiagramm.

3.2. Skizzen zum neuen User Interface

3.2.1. Länderauswahl

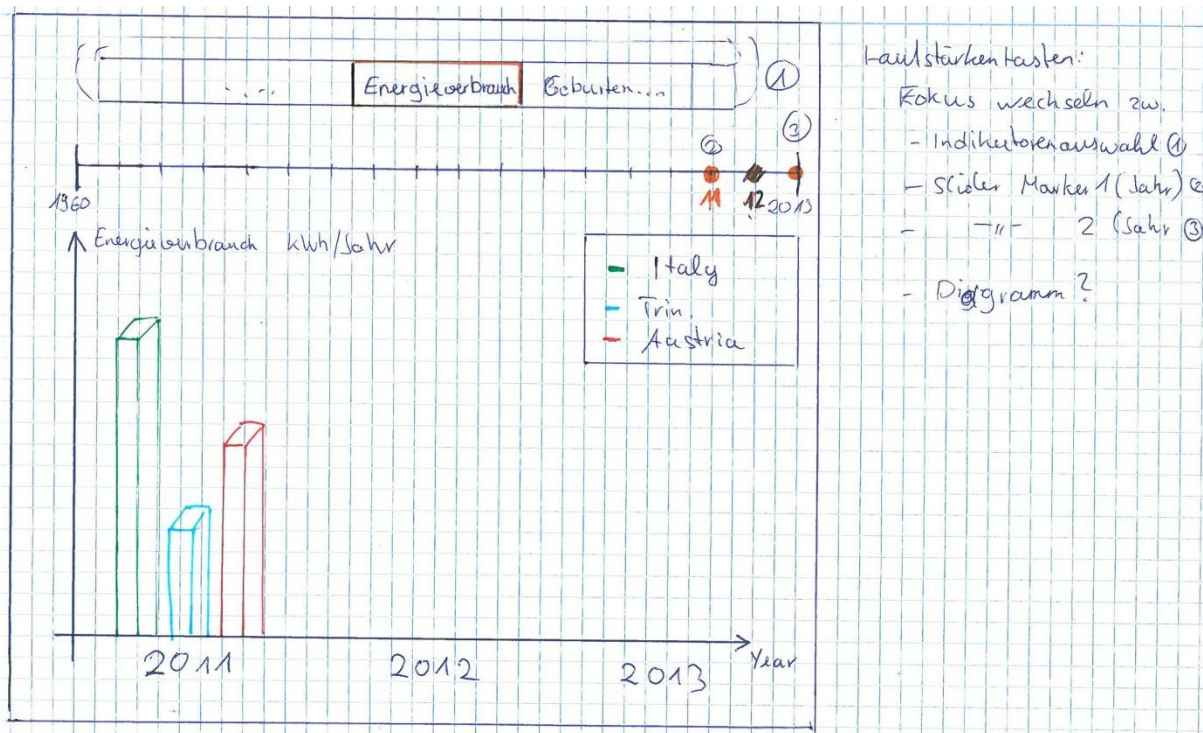


In der zweiten Version der Länderauswahl steht der Globus im Mittelpunkt. Ein Land soll mittels Rotation des Globus und Zoomen gefunden und ausgewählt werden. Der Listenansatz steht alternativ für Länder bereit, deren geographische Lage dem User noch unbekannt ist. Dieser Ansatz bietet sich aber auch zur Auswahl von kleinen Ländern an, die nicht gut über die Rotation zentriert werden können.

Ein Infocfeld rechts oben soll grundlegende Informationen zum aktuell fokussierten Land anzeigen. Dazu gehören zum Beispiel die Hauptstadt und die Bevölkerungsanzahl.

Unten auf dem Bildschirm an der linken Seite wird zusätzlich eine Liste der ausgewählten Länder für den Datenvergleich verwaltet.

3.2.2. Vergleich von Ländern



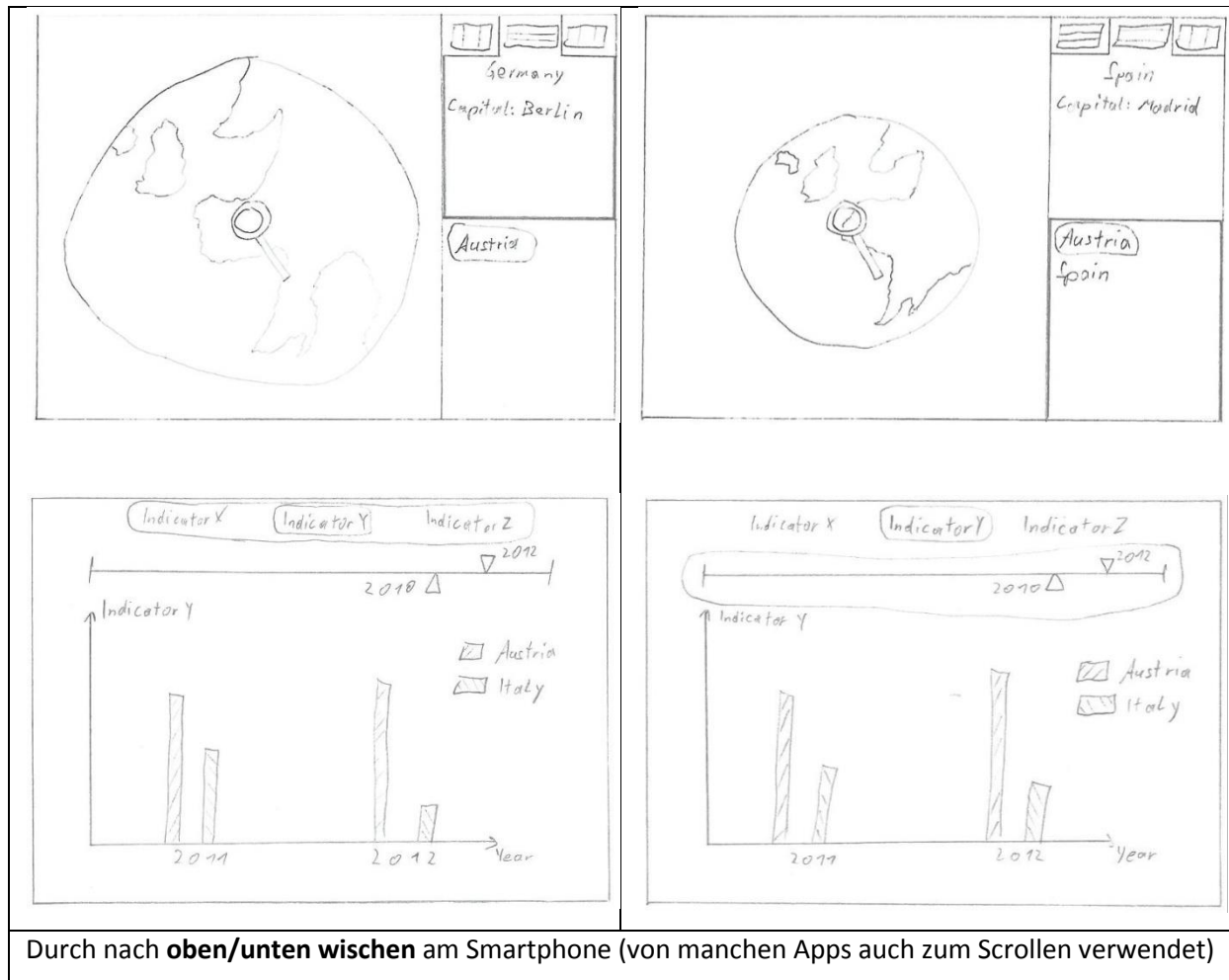
Die Auswahl des Weltbankindikators befindet sich in der neuen Variante auf dem Ländervergleich. Durch einfaches Scrollen kann in einer kurzen Liste von interessanten Weltbankindikatoren navigiert werden.

Der neue Ländervergleich bietet nun zudem einen Slider an, über den der User die Zeitspanne einschränken kann. Den größten Teil des Bildschirms nimmt ein 2D-Diagramm.

3.3. Interaktionen

3.3.1. Allgemeine Aktionen

3.3.1.1. Fokus wechseln

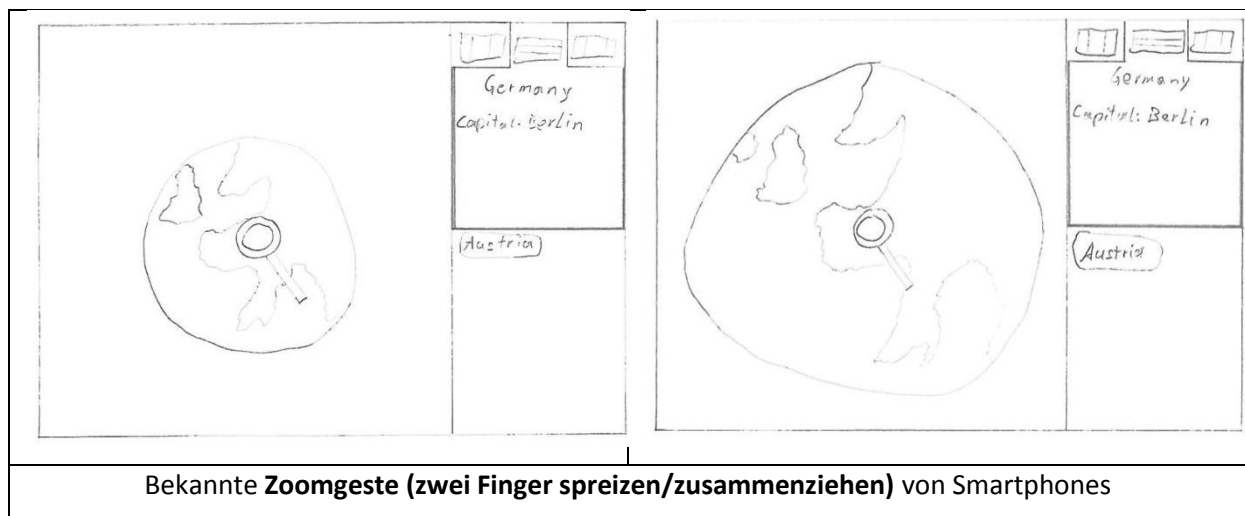


3.3.1.2. Szene wechseln

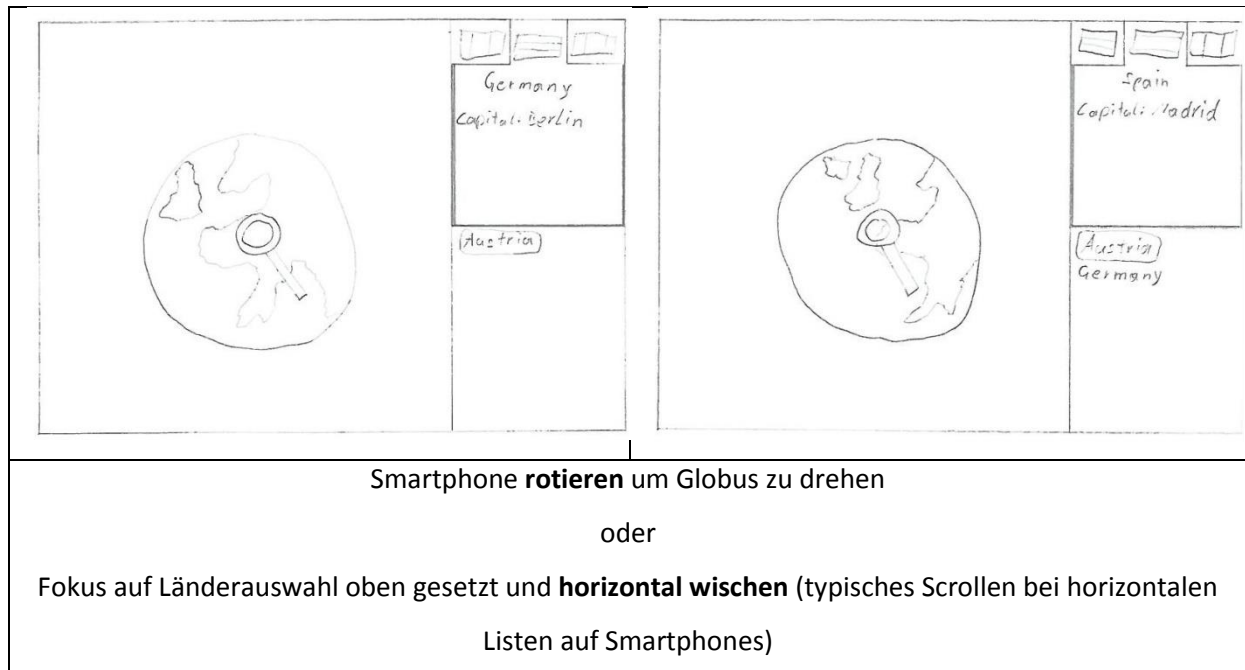


3.3.2. Länderauswahl

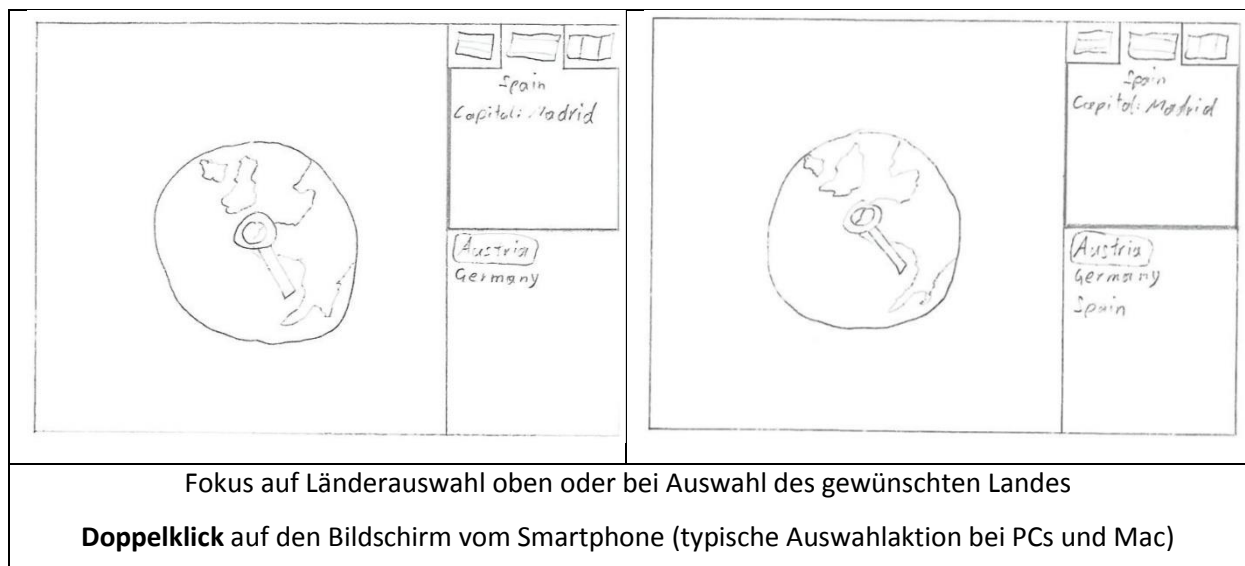
3.3.2.1. Zoom

































3.3.2.2. Land auswählen



3.3.2.3. Land zur Vergleichsauswahl hinzufügen



3.3.2.4. Land aus der Vergleichsauswahl entfernen

 <table border="1" data-bbox="619 564 790 757"><tr><td></td><td></td><td></td></tr><tr><td colspan="3">Spain Capital: Madrid</td></tr><tr><td colspan="3">Austria Germany Spain</td></tr></table>				Spain Capital: Madrid			Austria Germany Spain			 <table border="1" data-bbox="1241 295 1407 497"><tr><td></td><td></td><td></td></tr><tr><td colspan="3">Spain Capital: Madrid</td></tr><tr><td colspan="3">Austria Germany Spain</td></tr></table>				Spain Capital: Madrid			Austria Germany Spain		
																			
Spain Capital: Madrid																			
Austria Germany Spain																			
																			
Spain Capital: Madrid																			
Austria Germany Spain																			
 <table border="1" data-bbox="1241 730 1407 931"><tr><td></td><td></td><td></td></tr><tr><td colspan="3">Spain Capital: Madrid</td></tr><tr><td colspan="3">Austria Spain</td></tr></table>				Spain Capital: Madrid			Austria Spain												
																			
Spain Capital: Madrid																			
Austria Spain																			

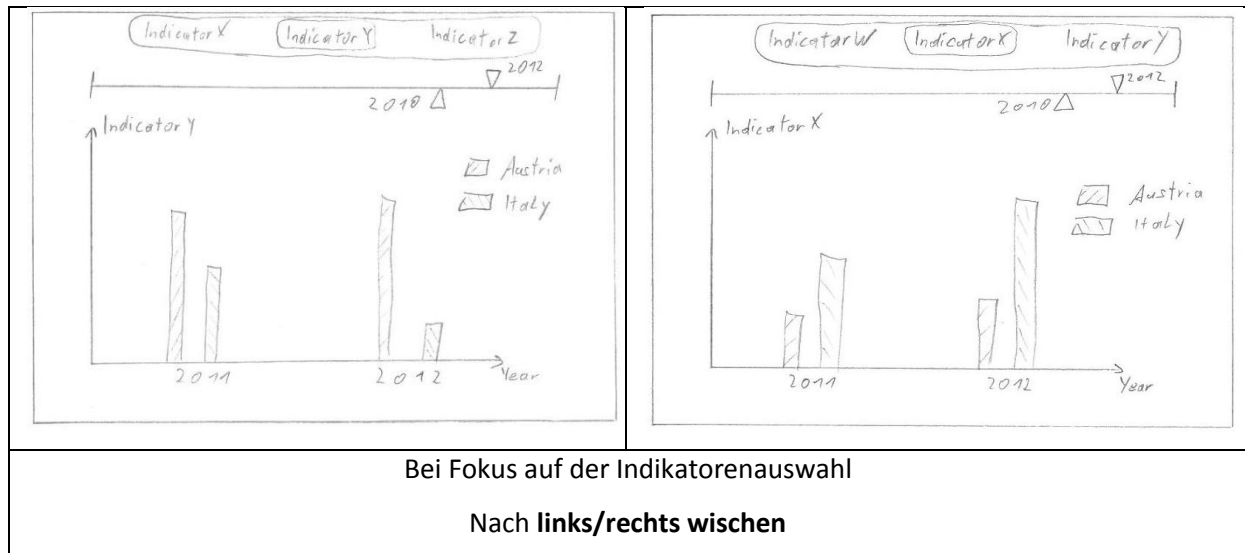
Bei Fokus auf der Liste der ausgewählten Länder unten

Mittels **Wischen nach oben/unten** wird das Land aus der Auswahl selektiert

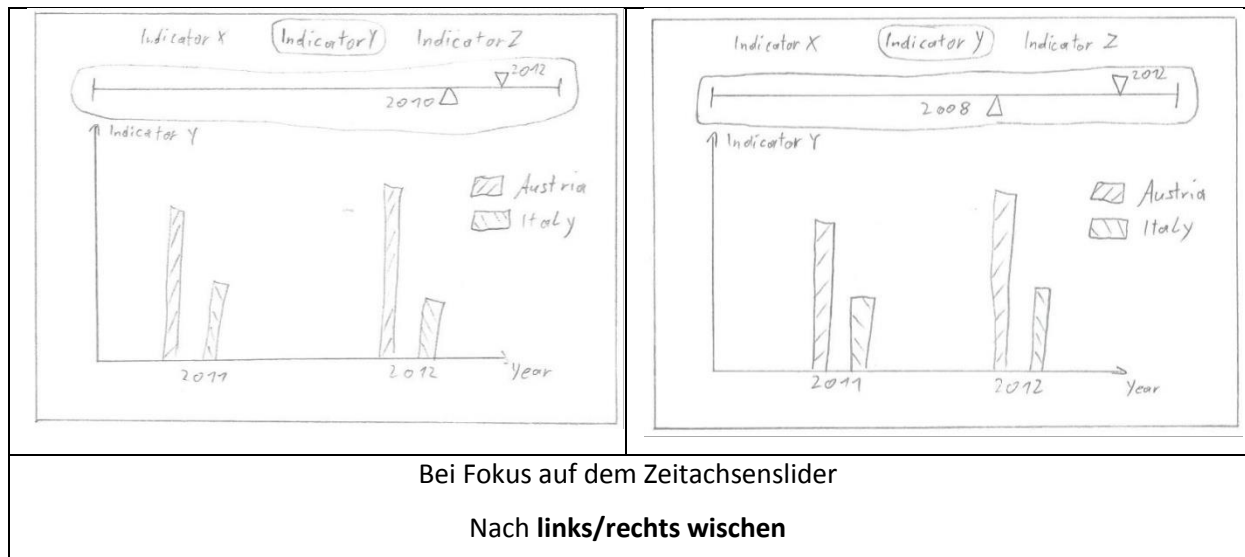
Mit einem **Doppelklick** wird das selektierte Land wieder entfernt

3.3.2. Datenvisualisierung

3.3.2.1. Weltbankindikator auswählen



3.3.2.2. Zeitspanne definieren



3.3.2.3. Daten laden und visualisieren

Um die Daten zu laden und visualisieren reicht ein **Doppelklick** bei beliebigen Fokus aus. Außerdem soll das Laden der Daten auch nach längerer Inaktivität des Users gestartet werden.

4. Realisierung

4.1 Verwendete Tools

Während der Entwicklung haben wir folgende Tools eingesetzt:

- Unity3D v 4.3 und MonoDevelop
- Google Android SDK (SDK 15)
- Visual Studio 2012
- Eclipse Kepler 4.3.1
- Eclipse BIRT 4.3.1
- Blender 2.69
- Java 7
- Apache Tomcat 7.0.50

4.2. Unity-Anwendung

Die Unity-Anwendung dient hauptsächlich zur Darstellung der Weltkugel und der Visualisierung von Daten.

Die Anwendung wurde komplett auf einem Schichtenmodell aufgebaut. Dadurch sind die Darstellung, die Datenverarbeitung und die Kommunikation im Code voneinander getrennt. Dies führt zur besseren Erweiterbarkeit der Anwendung.

4.2.1. Ländererkennung und Koordinatenumrechnung

4.2.1.1. Ländererkennung

Das aktuell dargestellte Land ist bei Globetrotter immer das Land auf das die Kamera zeigt. (Kamera ist auf Mittelpunkt vom Globus zentriert.

Um eine Position auf der Sphere zu bekommen haben wir den Raycast von Unity verwendet. Damit eine Kollision zustande kommen kann muss die „Sphere“ im Unity unbedingt einen **MeshRenderer** besitzen.

```
// Mitte von Kamera als Ausgangsposition
Ray ray = Camera.main.ViewportPointToRay (new Vector3(0.5f,0.5f,0));
RaycastHit hit;

// Nach Kollision mit Sphere schauen
if (collider.Raycast (ray, out hit, 10000f)) {
    // Kollision mit Sphere (MeshRenderer)

    // Umrechnung der Koordinaten auf Originalgröße der Textur
    int w = (int)Mathf.Round(hit.textureCoord.x * 4096);
    int h = (int)Mathf.Round(2048- hit.textureCoord.y * 2048);

    Vector2 v = PixelXYToLatLong(w, h);

    Country c = getCountry(v.y, v.x);
    if(c != null){
        ...
    }
}
```

4.2.1.2. Rotieren zu einem Land

Prinzipiell werden die GPS-Koordinaten mit denselben Formeln umgewandelt wie bei der Erkennung von Ländern. Damit jedoch die Kugel an die gewünschte Position rotiert werden kann muss man die Texturkoordinaten in den 3D Raum von Unity umrechnen.

```
private void rotate(){
    Country c = null;
    _countries.TryGetValue(m_countrySelectorViewModel.CurrentCountry.IsoAlphaThreeCode, out c);

    if(c != null)
    {
        Debug.Log(c.Longitude + " - " + c.Latitude);
        // Umrechnung von Latitude / Longitude auf x,y
        Vector2 v2 = LatLongToPixelXY(c.Latitude, c.Longitude);
        // Umrechnung der Koordinaten für Unity
        v2.x = v2.x/ 4096;
        v2.y = (v2.y - 2048) / 2048 * - 1;

        // Umwandlung der 2D Koordination in den 3D Raum
        Vector3 vector = UvTo3D(v2);
        Debug.Log(vector.x+"-"+vector.y+"-"+vector.z);

        lock(m_lockObj)
        {
            // Vektor der die Rotation der Kugel angibt
            m_rotationVector = vector;
            m_rotateToCountry = true;
        }
    }
}
```

4.2.1.3. UvTo3D

Diese Funktion wird verwendet um die 2D Texturkoordinaten in die Koordinaten des Spheringitters umzuwandeln.

```
Vector3 UvTo3D(Vector2 uv) {
    Vector2[] uvs = m_uv;
    Vector3[] verts = m_vertices;
    int[] tris = m_triangles;

    for (int i = 0; i < tris.Length; i += 3){
        Vector2 u1= uvs[tris[i]];
        Vector2 u2= uvs[tris[i+1]];
        Vector2 u3= uvs[tris[i+2]];
    }
}
```

```

        float a = Area(u1, u2, u3); if (a == 0) continue;

        float a1= Area(u2, u3, uv)/a; if (a1 < 0) continue;
        float a2 = Area(u3, u1, uv)/a; if (a2 < 0) continue;
        float a3 = Area(u1, u2, uv)/a; if (a3 < 0) continue;

        Vector3 p3D =
a1*verts[tris[i]]+a2*verts[tris[i+1]]+a3*verts[tris[i+2]];

        return p3D;
    }

    return Vector3.zero;
}

float Area(Vector2 p1, Vector2 p2, Vector2 p3){
    Vector2 v1= p1 - p3;
    Vector2 v2 = p2 - p3;
    return (v1.x * v2.y - v1.y * v2.x)/2;
}

```

4.2.1.4. PixelXYToLatLong und LatLongToPixelXY

Diese Funktion rechnet die Texturkoordinaten in Latitude, Longitude um. Sie berechnet die GPS-Koordinaten zwar nicht ganz genau, jedoch sind diese Fehler nur bei den Polarkappen groß. Ansonsten ist es eine gute Annäherung zur aufwendigeren Berechnung.

```

Vector2 PixelXYToLatLong(int pixelX, int pixelY)
{
    Vector2 vector = new Vector2();
    float longitude = pixelX / (4096/360.0f) - 180;
    float latitude = (pixelY / (2048/180.0f) - 90) * -1;
    vector.x = latitude;
    vector.y = longitude;
    return vector;
}

```

```

Vector2 LatLongToPixelXY(float lat, float lon){
    Vector2 vector = new Vector2();
    vector.x = (4096/360.0f) * (180 + lon);
    vector.y = (2048/180.0f) * (90 - lat);
    return vector;
}

```

4.2.1.5. *GetCountry*

Diese Funktion ermittelt ein Land anhand von Longitude und Latitude.

Die Grenzen aller Länder der Welt sind in einer CSV-Datei hinterlegt, die beim Start eingelesen werden. Der Algorithmus untersucht ob Werte in einem Polygon liegen oder nicht. Bei Globetrotter wird jedes Land daher als ein einzelnes Polygon oder als mehrere Polygone definiert.

Der dazugehörige Algorithmus ist hier erklärt:

http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html

4.2.2. Anbindung zum Weltbank-Webservice

Für die Verwendung des Services gibt es eine ausführliche Dokumentation der Weltbank:

<http://data.worldbank.org/developers/api-overview>

Für Klimadaten:

<http://data.worldbank.org/developers/climate-data-api>

4.2.3. Chart-Webservice

Während der Entwicklung stellten wir fest, dass das Erstellen von Diagrammen mit Unity3D nicht zielführend ist. Da Unity die System.Drawing API von .NET nicht enthält, kann von Unity3D aus keine Chart Library direkt aufgerufen werden. Deshalb ist die Chartgenerierung in einen REST Webservice ausgelagert, der anhand von Parametern die entsprechenden Daten von der Weltbank lädt und mit dem Tool Eclipse BIRT ein Chart generiert, welches zurückgeschickt wird. Als Parameter müssen dem Webservice die gewünschten Länder, die Zeitspanne in Jahren, der Weltbankindikator und den Typ des Charts übergeben werden.

4.3. Android-App

Die Globetrotter Android-App liest Sensorwerte aus und erkennt Gesten. Diese Werte bzw. Informationen zu Gesten werden über eine UDP-Datenübertragung an die Unity-Anwendung gesendet.

Zur Initialisierung der Verbindung muss der Benutzer lediglich den QR-Code am Startbildschirm von Globetrotter mit der Android-App scannen.

5. Herausforderungen

5.1. Webservice der Weltbank

Am Anfang hat die Datenabfrage an den Webservice der Weltbank ganz gut funktioniert. Aber über Weihnachten stellten wir fest, dass die Weltbank scheinbar einige Änderungen am Webservice vorgenommen hat.

Es fiel uns auf, dass der Webservice nach der Datenabfrage die Connection bei HTTP/1.1 offen hält. Daraus resultierte, dass unser Code das Ende der Datenübertragung nicht mehr erkennen konnte und weiter auf Daten wartete. Durch die Berücksichtigung der *Content-Length* im Header konnten wir aber die genaue Datenmenge herausfinden und nur solange Daten einlesen, bis wir die übertragene Datenmenge erreichten.

Den HTTP-Request mussten wir ebenfalls umstellen. Der Webservice zur Abfrage von Weltbankindikatoren schickte bei unserer alten Implementierung auf einmal den Fehler *400 Bad Request* zurück. Der Webservice zur Abfrage der Länder nahm den HTTP-Request aber als korrekt an. Dieses Problem lässt sich aber beheben, indem jede Zeile im HTTP-Request konsequent mit `\r\n` beendet wird. So werden die HTTP-Request von den Webservices richtig angenommen.

5.2. 502 Bad Gateway

Nun tritt manchmal ein neuer Fehler auf. Gelegentlich passiert es, dass der Webservice den Fehler *502 Bad Gateway* zurückschickt. Bei einer erneuten Übertragung desselben HTTP-Requests wird dieser vom Webservice wieder als korrekt angenommen. Dieses merkwürdige Verhalten vom Webservice können wir aber mit einer Prüfung auf diesen Fehler und einer erneuten Übertragung des HTTP-Request handhaben.

5.3. Textencoding in Unity

Am Anfang eines Datenstroms befindet sich ein spezielles Zeichen namens Byte Order Mark. Grob gesagt gibt dieses an, ob die Bytes vorwärts oder rückwärts gelesen werden müssen. Unity löscht für sich dieses Zeichen am Anfang des Datenstroms und nimmt an, dass sich dieses spezielle Zeichen auch nicht in übergebenen Datenströmen befindet.

Nach dem Laden der Daten vom Webservice der Weltbank befindet sich aber das Byte Order Mark im Datenstrom. Dies führt dazu, dass beim Parsen die Ausnahme „*System.Xml.XmlException: Text node cannot appear in this state. Line 1, position 1.*“ Auftritt. Dieses Problem wird aber behoben, wenn wir den Datenstrom in einen In-Memory-Datenstrom umleiten und ihn aus diesem wieder auslesen. Wir nehmen an, dass die Implementierung von Unity hier das Byte Order Mark für sich wieder richtig behandelt.

5.4. Smartphone, Datenübertragung

Testgeräte:

- Samsung Galaxy S2
- HTC One

Sensoren unterschiedlicher Geräte liefern unterschiedlich viele Werte, beispielsweise sendet der Beschleunigungssensor des „Samsung Galaxy S2“ ungefähr 60-mal so viele Werte in der Minute wie ein „HTC One“.

Das unterschiedliche Verhalten zeigt sich auch bei der Gestenerkennung. Beispielsweise werden bei der Zoomgeste ständig Informationen gesendet um damit den Faktor des Zooms festzustellen.

Als Folge der unterschiedlichen Anzahl an Werten gab es bei der Performance (ruckelfreie Darstellung, Stabilität, etc.) der Unity-Anwendung je nach Smartphonetyp große Unterschiede.

Um die Anzahl der Werte zu beschränken bzw. zu normalisieren wird die Anzahl von der Android Applikation beschränkt. (20 Werte / Sekunde)

Durch die Beschränkung der Werte wurde auch die Performance der Datenübertragung stark verbessert.