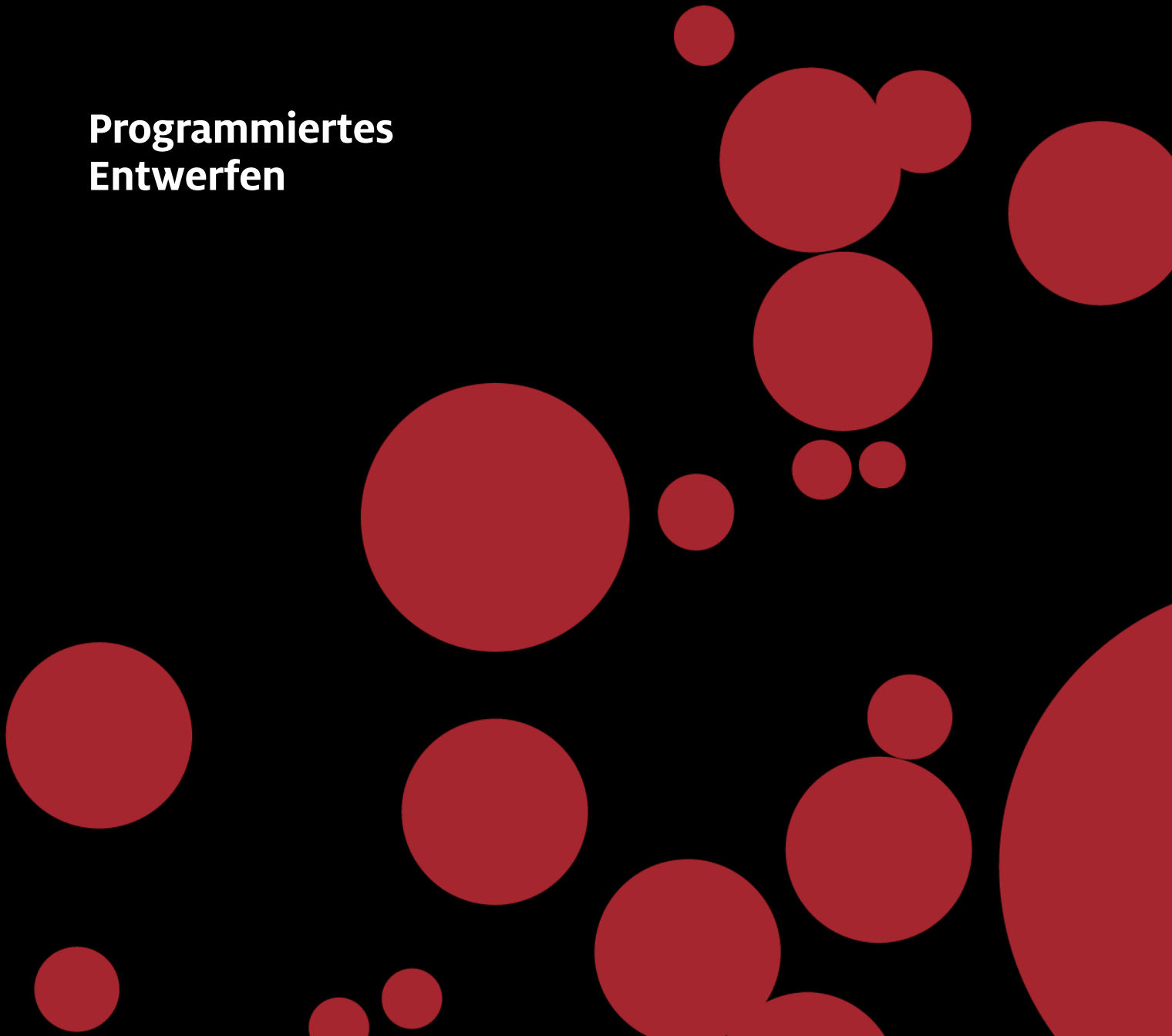
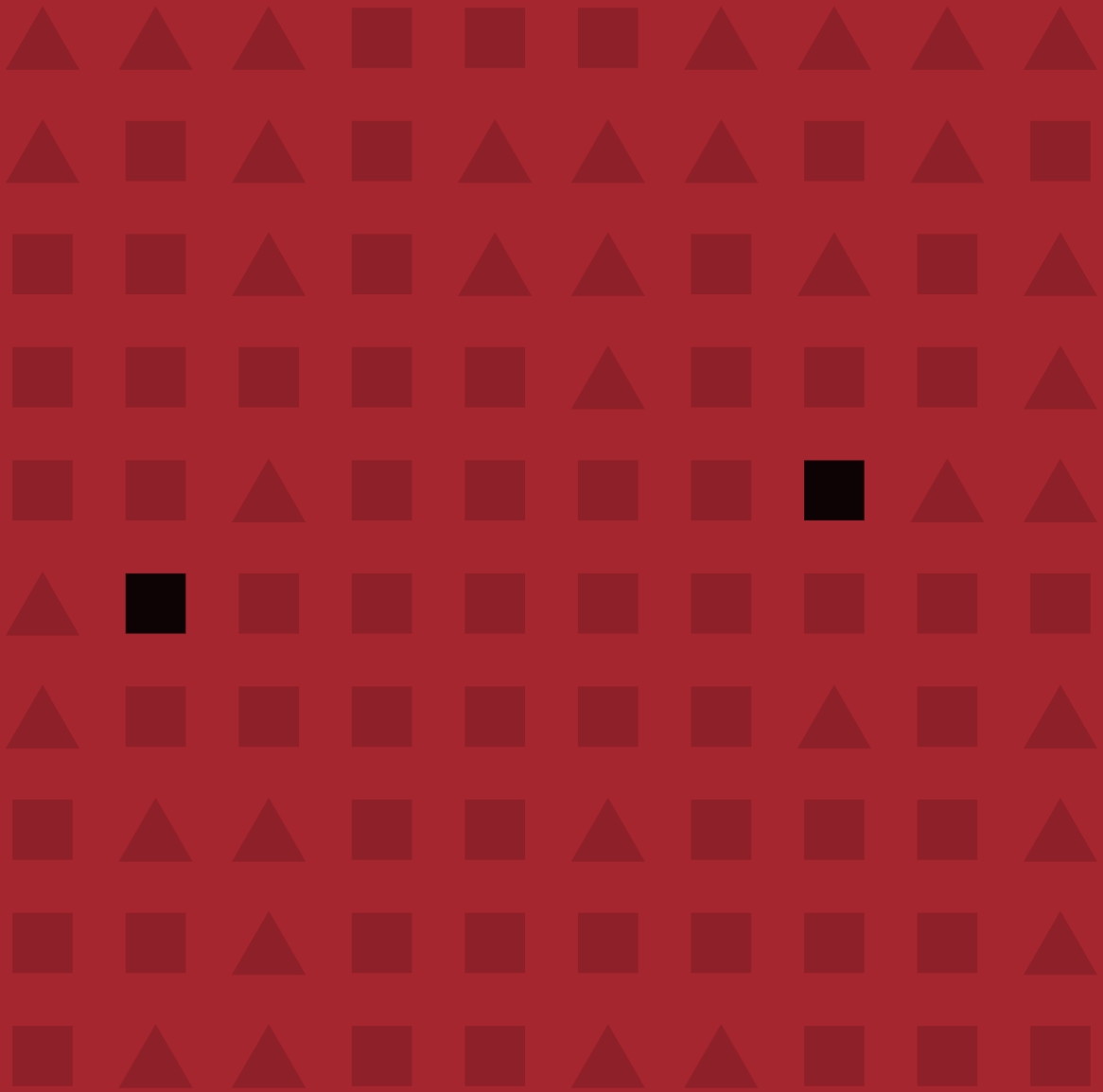
Programmiertes Entwerfen





Programmiertes Entwerfen

Christoph Labacher

Interaktionsgestaltung 2
Sommersemester 2014

Hochschule für Gestaltung
Schwäbisch Gmünd

Prof. Michael Götte
Prof. Jens Döring

Aufgabe

Ziel des Projektes war es, eine dynamische, interaktive Visualisierung zu erstellen, die in beliebiger Weise zeitabhängige Daten darstellt.

Dabei sollte das auf reine Daten reduzierte Ausgangsmaterial mit Hilfe formaler Gestaltungsparameter erfahrbar gemacht und in seinem chronologischen Zusammenhang erklärt werden. Gestaltungsparameter können zum Beispiel Form, Farbe, Ordnung oder Verhalten (im Bezug auf Interaktion oder Zeit) sein. Diese Parameter sollten systematisiert angewendet werden, jeder Parameter sollte also dazu dienen entweder Daten zu kommunizieren oder Bezüge klar zu machen.

Inhalt

6 **Daten**

8 **Konzept**

16 **Gestaltungsparameter**

30 **Entwurfsprozess**

38 **Technologie**

Daten

Aktualität und Relevanz der dargestellten Daten

Im Augenblick leben etwa 7,2 Milliarden Menschen auf der Erde. Eine etwas genauere Schätzung der „Stiftung Weltbevölkerung“ spricht von 7.202.951.000 Menschen. Das Bevölkerungswachstum wird üblicherweise in Geburten pro Jahr und 1000 Einwohnern angegeben. Auf der ganzen Welt wären das dieses Jahr etwa 12,03 Geburten. In China sind es 14,7; in Deutschland 8,42; in Großbritannien 12,22.

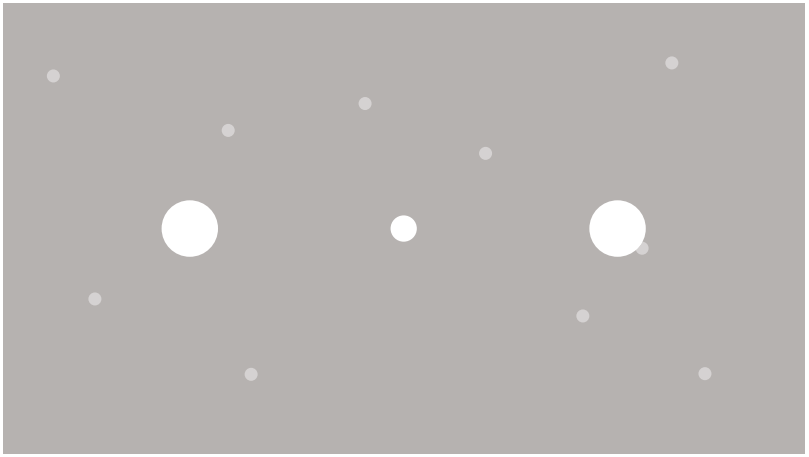
All diese Zahlen sind zwar exakt – dafür sind sie aber entweder unvorstellbar groß, oder sehr abstrakt. Was bedeutet es, wenn in einem Land 14,07 Kinder je 1000 Einwohner in einem Jahr auf die Welt kommen? Das enorme Wachstum der Weltbevölkerung, das in naher Zukunft kein Ende nehmen wird, die drohende Überbevölkerung einzelner Länder und der gesamten Erde – um die Zukunft richtig einschätzen zu können ist wichtig diese Zahlen zu verstehen und ihr Bedeutung und Relevanz zu begreifen, aber das fällt schwer.

Mit meinem System möchte ich versuchen die Daten greifbar zu machen und einen direkten Vergleich zu schaffen, um die Proportionen zu verdeutlichen. Die abstrakten Daten sollen in grafischer Form visualisiert werden und dabei durch verschiedene Interaktionen erkundbar sein.

	Deutschland	Großbritannien	USA	China
Einwohner in Millionen	81	64	319	1.356
Geburten pro Jahr je 1000 Einwohner	8,42	12,22	13,43	12,17
Kindersterblichkeit je 1000 Geburten	3,46	4,44	6,17	14,7
Verhältnis der Geschlechter Männer je Frau	1,06	1,05	1,05	1,11

Konzept

Aufbau



Auf dem Bildschirm befinden sich verschiedene Formen als Repräsentationen für die Länder. Wird in dem entsprechenden Land ein Kind geboren, so fährt aus der Form eine kleine Form heraus und bewegt sich frei über den Bildschirm. In der Mitte zwischen den Länder befindet sich eine weitere Form als Zeitindikator.

Die Anzahl der zu vergleichenden Länder ist nur beschränkt durch die Größe der Ansicht. Optimal ist ein Vergleich zwischen zwei oder vier Ländern.

Zeitlicher Ablauf

Initiierungssequenz

Zu Beginn ist der Bildschirm komplett schwarz. Der Nutzer startet die Visualisierung bewusst durch eine Interaktionen. Die „Länder“ werden eingeblendet. Langsam wandeln sie sich von einem Kreis in ihre endgültige Form (siehe Seite 18).

Entwicklung

Mit der Zeit fahren kleine Formen, die geborene Kinder repräsentieren, aus den Ländern und bewegen sich über den Screen. Ist eine bestimmte Anzahl von Kindern eines Landes erreicht, so gruppieren sie sich zu einer größeren Form. Alle Formen prallen von einander und den Seiten der Ansicht ab.

Steuerung der Zeit

Die Geschwindigkeit mit der die Kindern in den Ländern „geboren“ werden entspricht zu Beginn der Echtzeit. Durch einen Klick auf ein Land wird der Geburtenrhythmus dieses Landes zum Ein-Sekunden-Rhythmus des Systems, alle anderen Ländern verhalten sich dem entsprechend. Durch einen Klick auf den Zeitindikator in der Mitte kann das System wieder auf Echtzeit umgestellt werden.

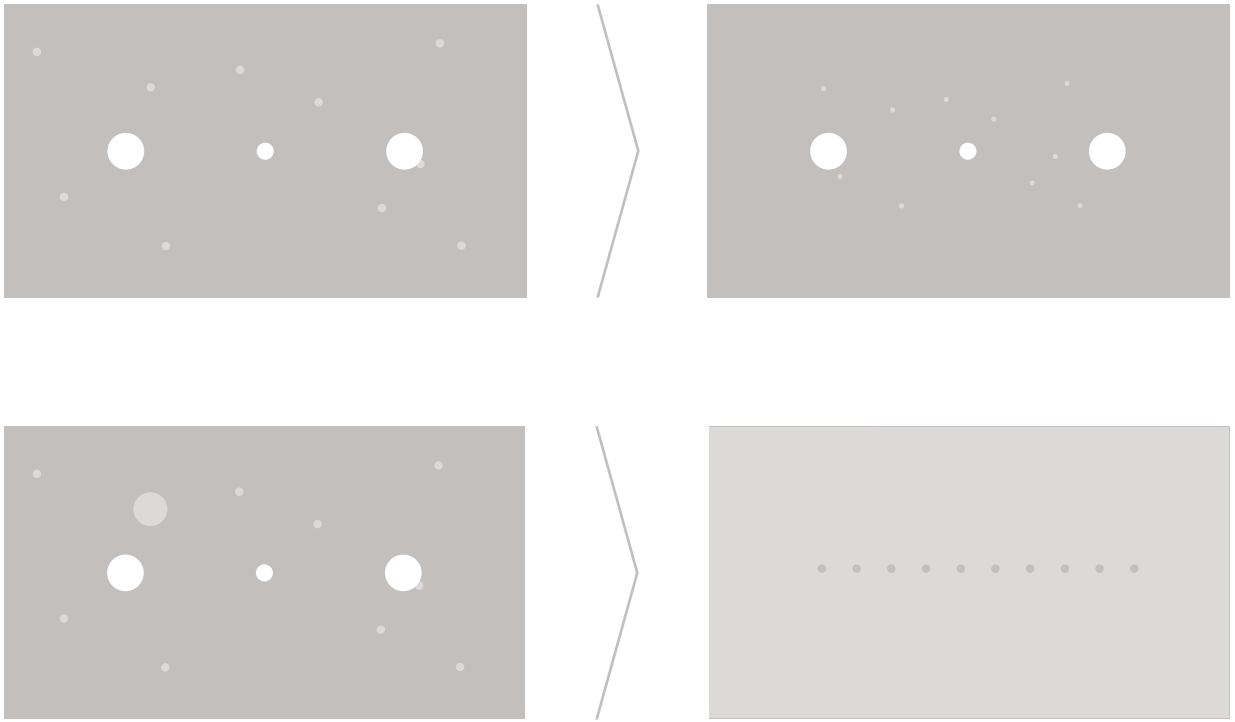
Perspektivwechsel

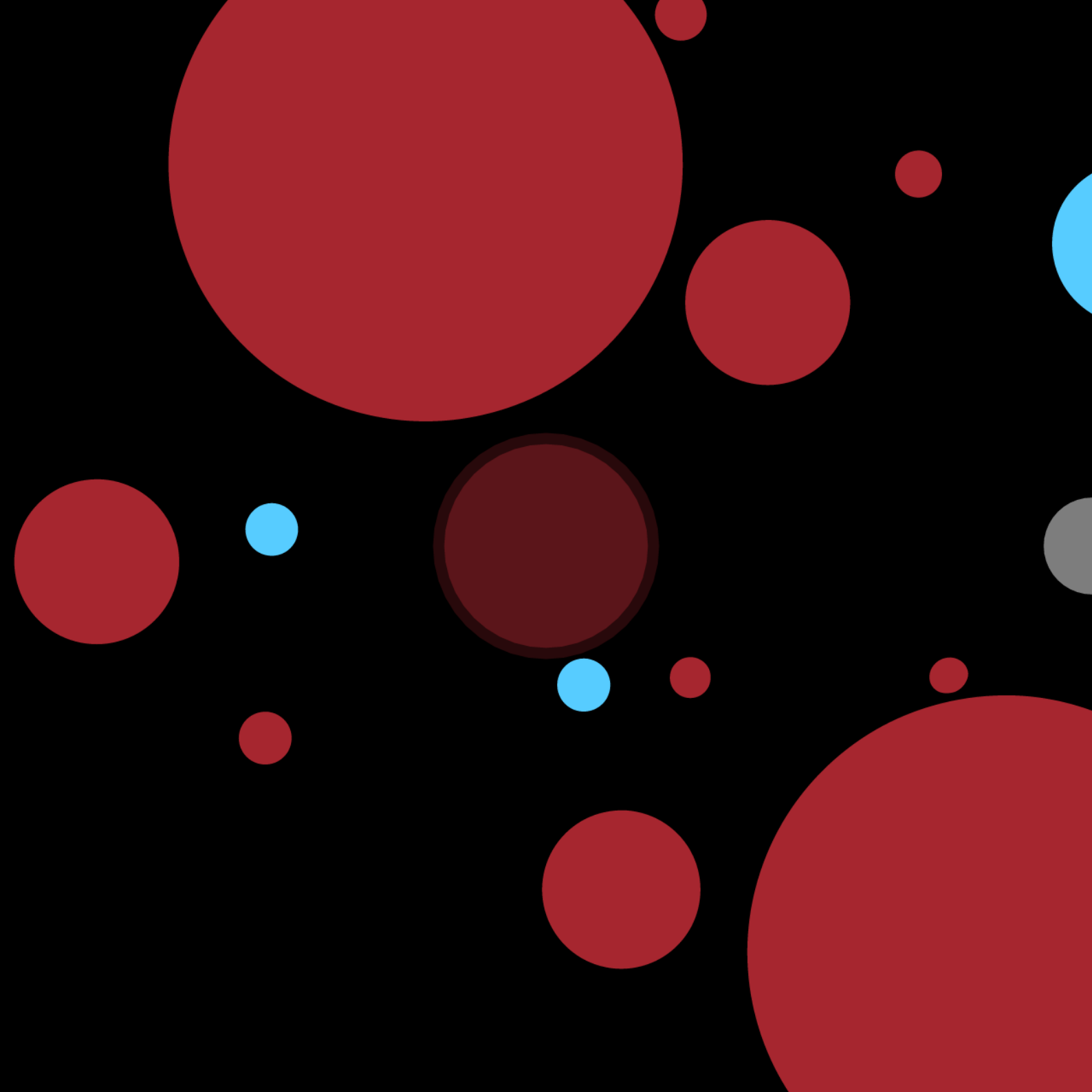
Zoom-Out

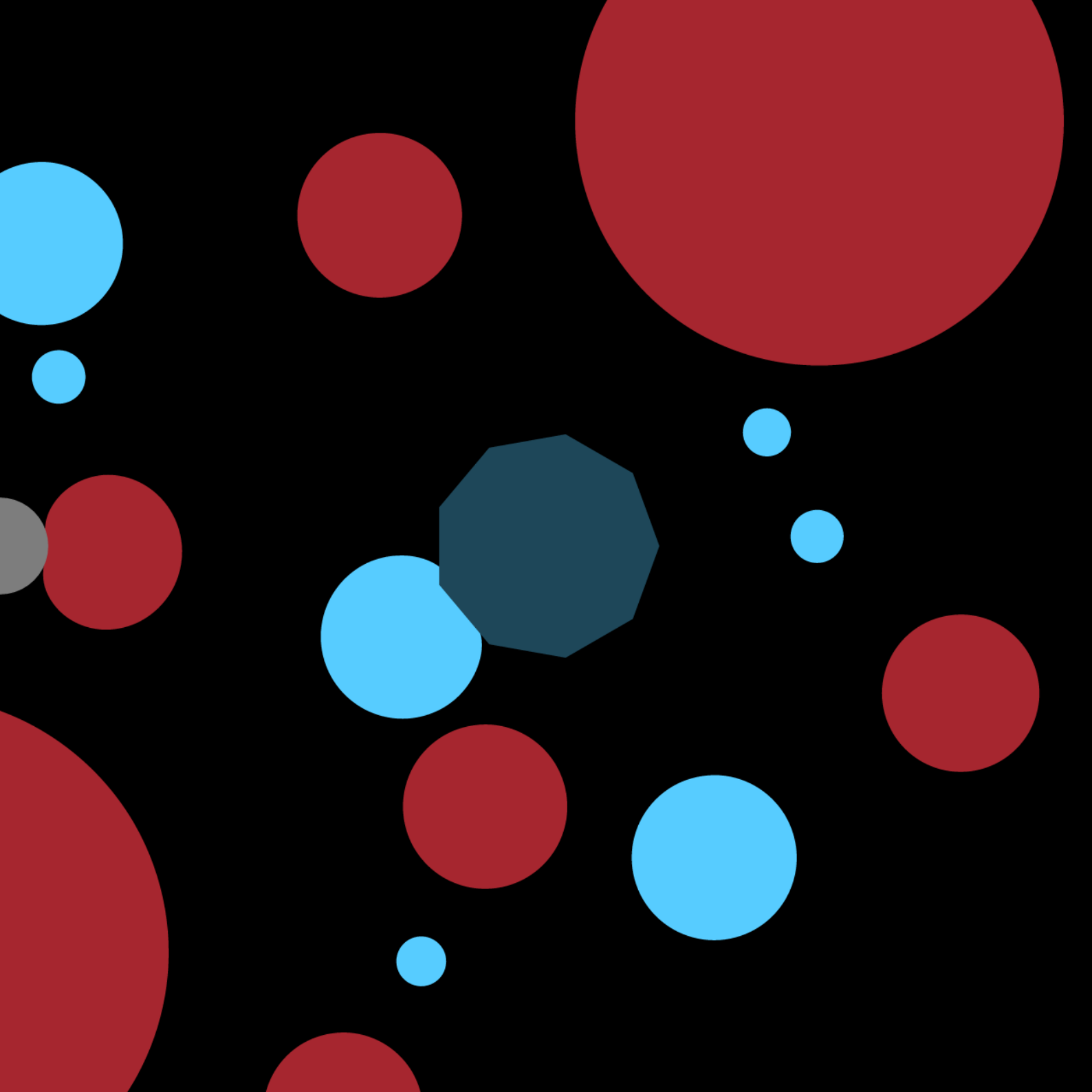
Ist eine bestimmte Anzahl von „Kindern“ auf dem Screen zu sehen, so zoomt die Ansicht nach außen um mehr Raum zu schaffen. Die Formen können sich nun wieder freier bewegen und werden nicht mehr so stark an die Seiten der Ansicht gedrängt. Nach einer gewissen Zeit ist der Bildschirm jedoch wieder gefüllt. Der „Zoom-Out“ findet mehrere Male statt – allerdings nicht unendlich oft. Nach dem Zoom-Vorgang vergrößern sich die die Länder repräsentierenden Formen ein Stück, damit sie klar erkennbar bleiben.

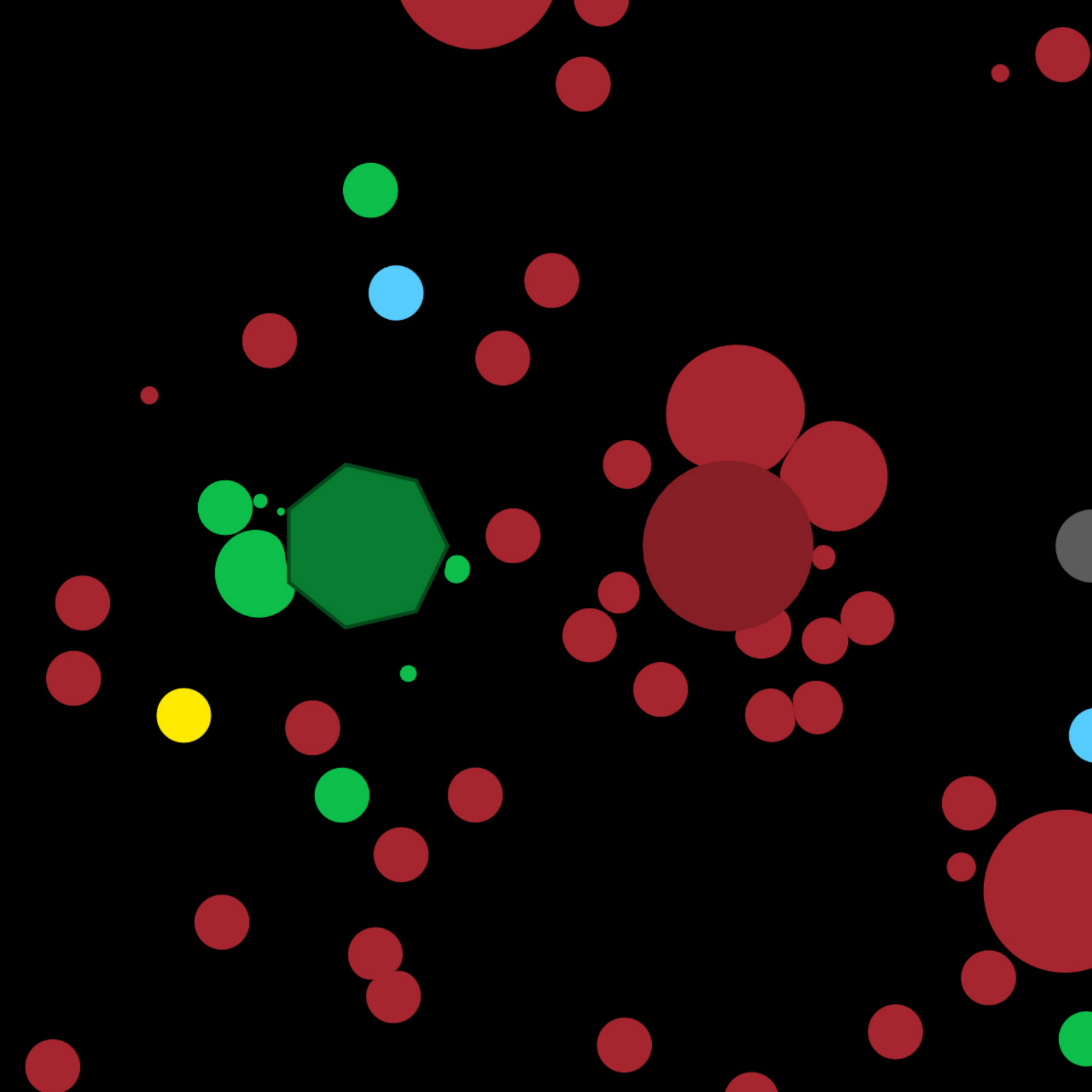
Zoom-In

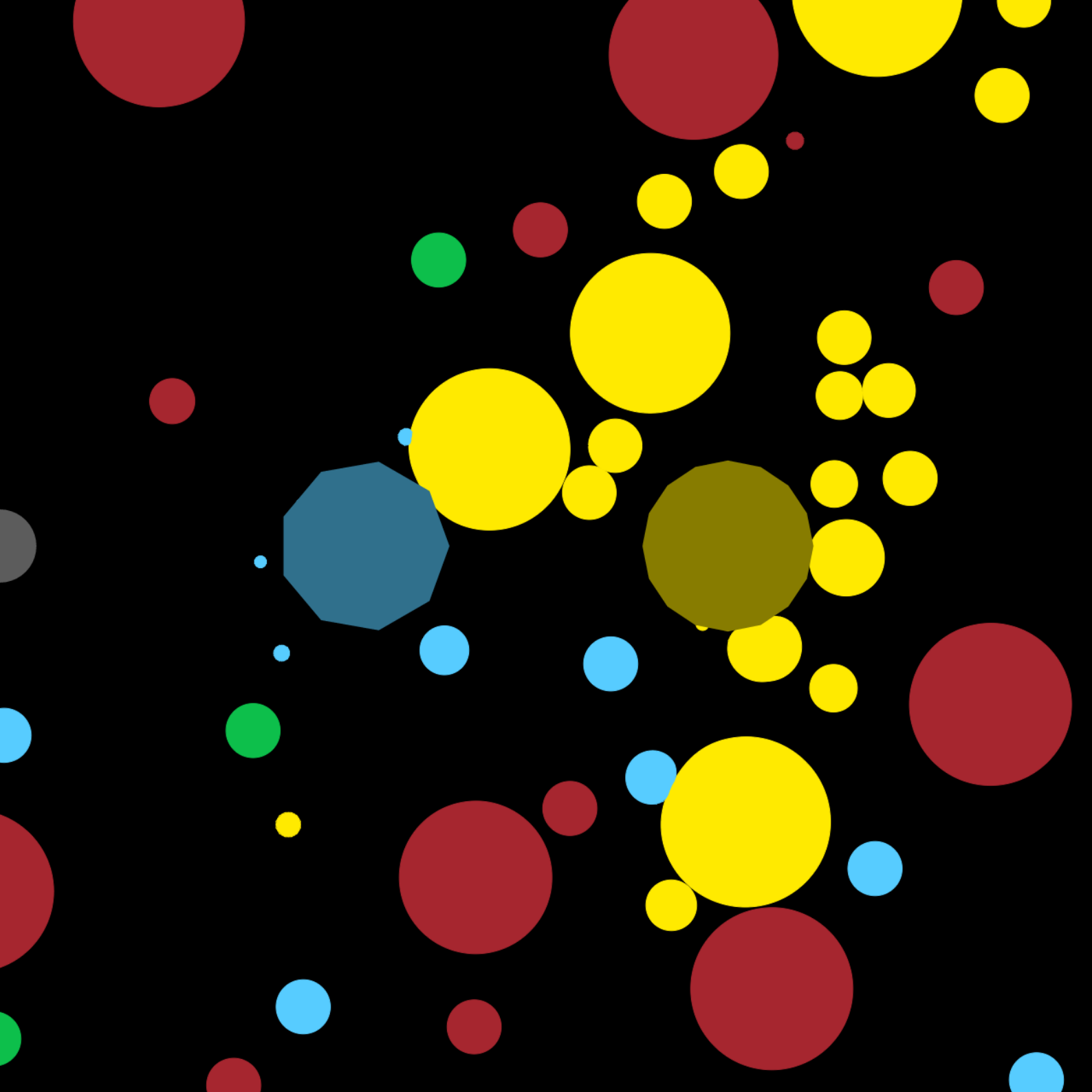
Der Nutzer kann auf die größeren Formen, die gruppierte kleinere Formen enthalten, klicken – sie klappen sich dann auf, bis sie die gesamte Ansicht ausfüllen. Dann werden kleine Formen für die in einzelnen in der Gruppierung enthalten „Kinder“, die grafisch kodiert weitere Informationen zu den einzelnen Kindern enthalten, angezeigt.











Gestaltungsparameter

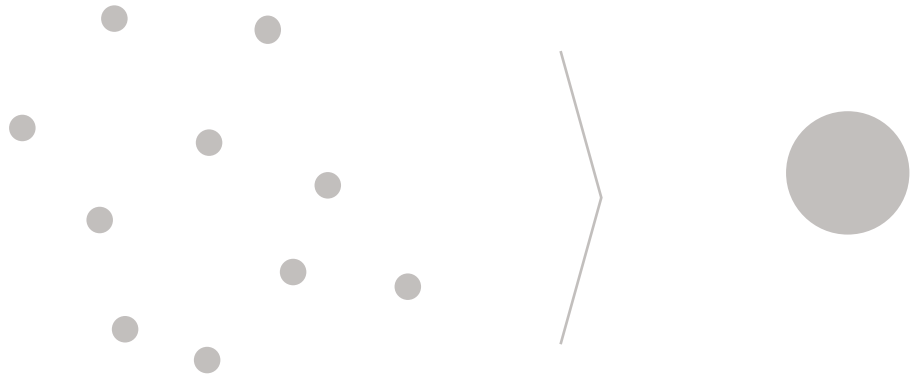
Grundprinzipien

Zeit

Der Umgang mit Zeit ist eine der grundlegenden Eigenschaften des Systems: Dargestellt werden nicht historische Informationen oder abstrahierte Verhältnismäßigkeiten, sondern konkrete Zahlen, die während der Laufzeit des Systems generiert werden. Das geschieht zwar nicht immer in Echtzeit, aber der Zeitverlauf der Visualisierung ist komplett linear und nur in eine Richtung möglich. Zeitsprünge sind ausgeschlossen, die Zeit kann allerdings pausiert werden (zum Beispiel beim Zoom-In).

Gruppierung

Die Formen stellen konkrete Zahlen dar: Jeder Kreis entspricht einem Kind. Mit der Zeit nimmt die Anzahl der Kreise stark zu und die verschiedenen Ländern zugehörigen Kreise vermischen sich untereinander – die Übersicht geht verloren und auf den ersten Blick sind die Mengenverhältnisse der Länder untereinander schwer zu erkennen. Damit das System auch bei größeren Datenmengen funktioniert werden die Kreise gruppiert: zehn kleine Kreise werden zu einem größeren Kreis, der zehn Kinder repräsentiert, zehn dieser Kreise zu einem Kreis mit 100 Kindern, usw. So wird die Übersicht gesteigert und die Mengenverhältnisse sind auf den ersten Blick klar erkennbar.



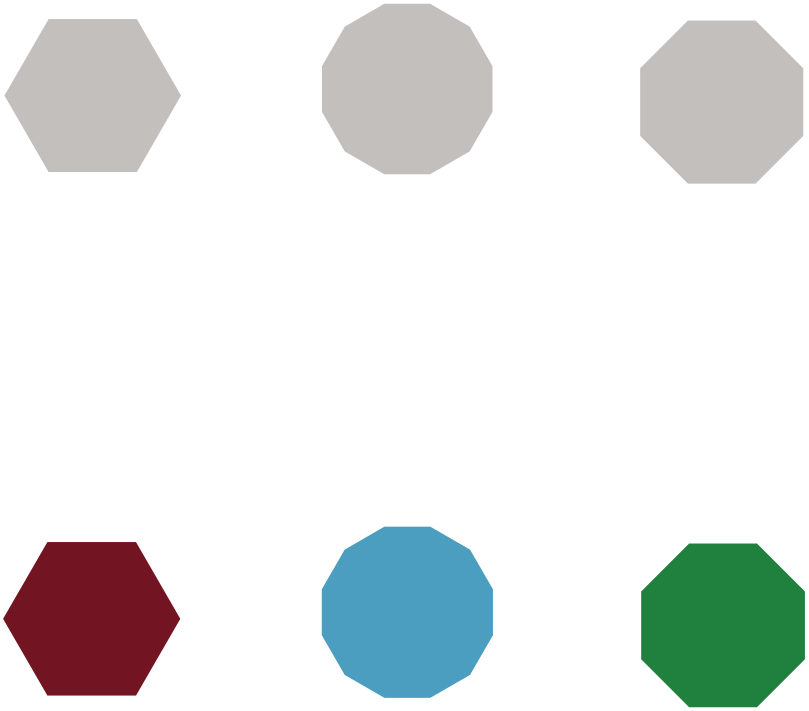
Land

Form

Jedes Land, das es zu vergleichen gilt, ist durch eine polygone Form im Zentrum der Ansicht repräsentiert. Das Land an sich ist nicht das Hauptaugenmerk der Visualisierung, es ist daher stark reduziert: Die Länder besitzen eine einheitliche Größe – sie sollen eher als Interaktionsobjekte denn als Datenpunkte wahrgenommen werden. Um dennoch eine Orientierung über die Größe des Landes zu geben entspricht die Anzahl der Ecken des Polygons einer Ecke je 10 Millionen Einwohner – in der Visualisierung ist die flächenmäßige Größe des Landes nicht relevant, alle Daten wurden auf die Einwohner beschränkt.

Farbe

Jedem Land ist eine Farbe zugeordnet. Alle mit dem Land in Zusammenhang stehenden Symbole sind in dieser Farbe eingefärbt, die Farbe variiert nur in ihrer Helligkeit. Das das Land selbst repräsentierende Symbol ist in einer etwas dunkleren Variante dieser Farbe eingefärbt. Die Farbe dient der Differenzierung der Länder untereinander und ist dementsprechend gewählt – der Farbton selbst hat keine mit dem dem Land verknüpfte Bedeutung.



Zeit

Jedes Land hat einen eigenen „Rhythmus“. Dieser entspricht der Zeit zwischen zwei Geburten in diesem Land in Echtzeit. Ist eine Phase des Rhythmus abgeschlossen, so pulsiert die Helligkeit des Landes einmal bis zum Maximum. In diesem Moment verlässt ein Symbol, das ein geborenes Kind repräsentiert, die Form des Landes. Jedes Land kann den Hauptrhythmus des Systems angeben, nach dem sich alle anderen Länder richten. Das rhythmusgebende Land (bzw. der Zeitindikator) hat eine breite, dunkle Kontur.

Zeitindikator

In der Mitte zwischen den Ländern befindet sich der Zeitindikator, der ähnlich wie ein Land aufgebaut ist. Er ist jedoch komplett rund, weiß und sein Rhythmus entspricht einer Sekunde. Der Zeitindikator hat keine Geburtenrate und keine ihm zugeordneten „Kinder“.

Person

Äußere Ansicht

Form

Die wichtigste Einheit um die herum das gesamte System aufgebaut ist, ist die einer Person, eines geborenen Kindes. Die Form dieser Einheit ist deshalb so reduziert wir möglich: Jedes geborene Kind wird durch eine kleine Kreisform in der Farbe des Landes repräsentiert.

Auch die größeren Einheiten, die entstehen wenn mehrere „Kinder“ sich gruppieren, sind Kreise. Sie unterscheiden sich von den kleinen Einheiten nur durch ihre Größe. Ausschlaggebend ist dabei der Flächeninhalt: Eine große Einheit hat den selben Flächeninhalt wie alle enthaltenen kleineren Einheiten zusammen addiert.

Person

Innere Ansicht

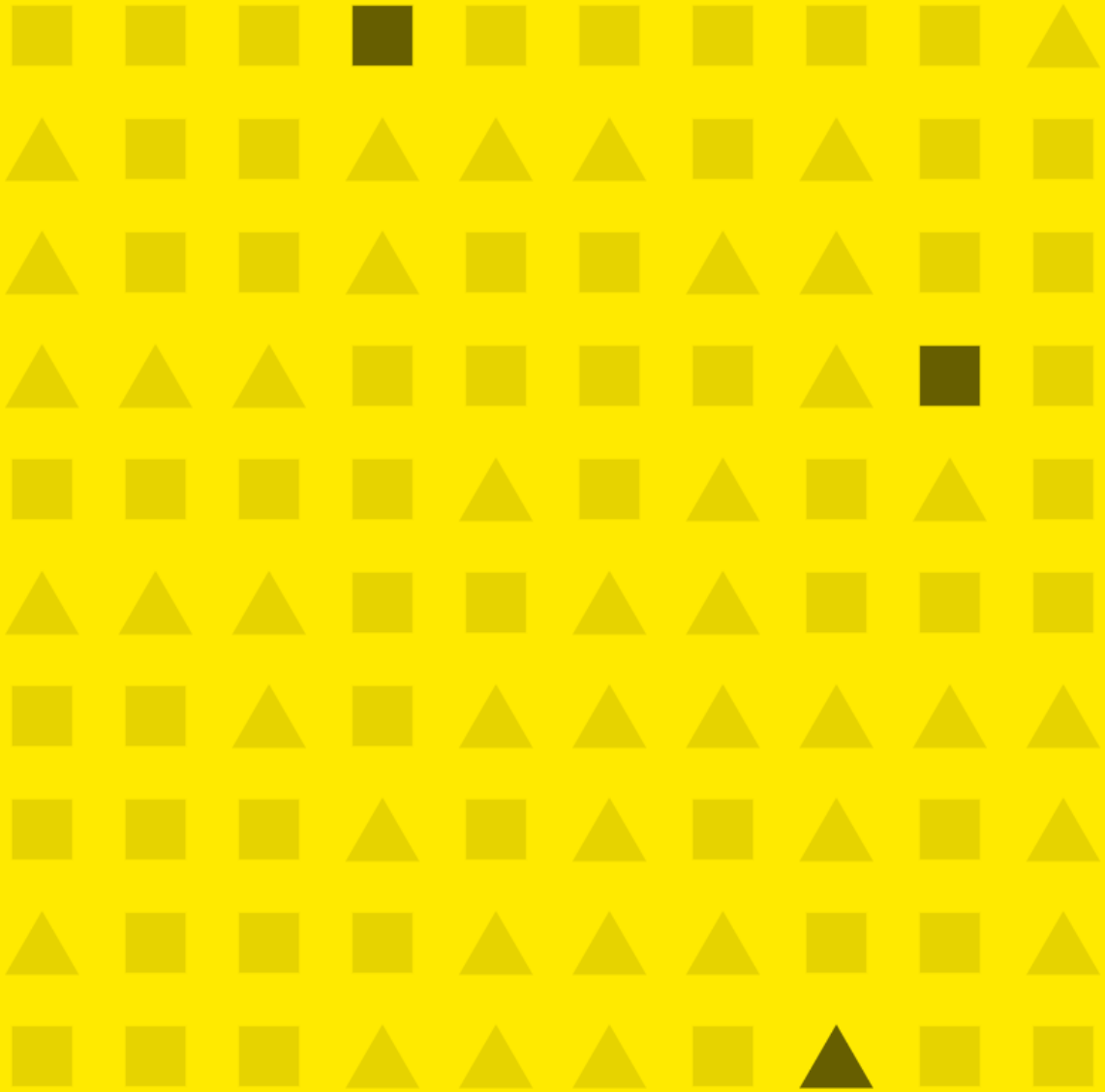
Es ist möglich in eine Gruppierung hinein zu zoomen um dann zusätzliche Daten über die enthaltenen Personen zu bekommen. Die Gruppe wird dann bildschirmfüllend angezeigt, die Hintergrundfarbe entspricht also der Grundfarbe des Landes. Darauf werden die enthaltenen Personen als kleine Formen in einem Raster angeordnet angezeigt.

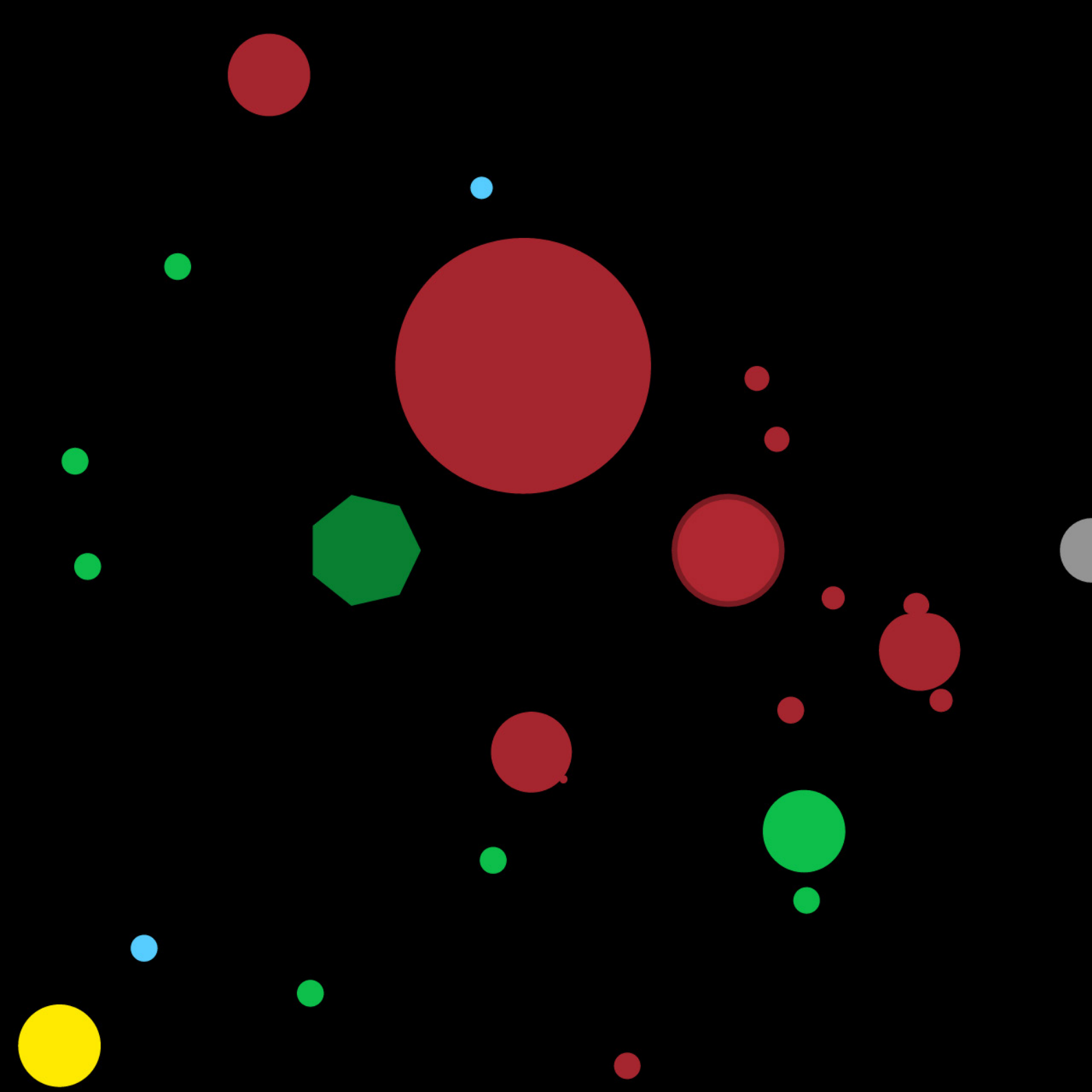
Form

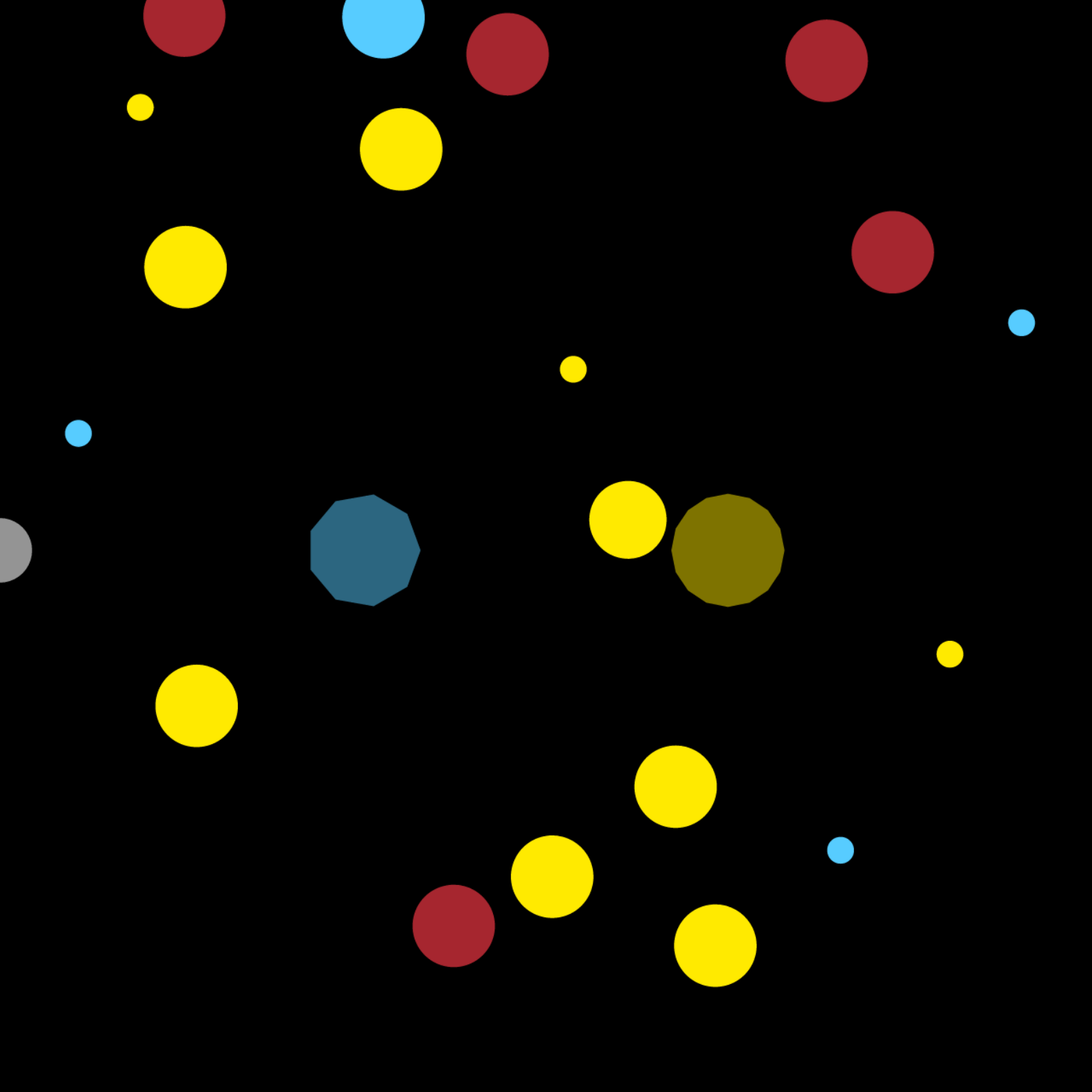
In dieser Ansicht werden die Personen nicht als Kreise dargestellt, sondern als Polygon. Die Anzahl der Ecken gibt dabei das Geschlecht an: Ein Dreieck steht für ein weibliches Kind, ein Quadrat für ein männliches.

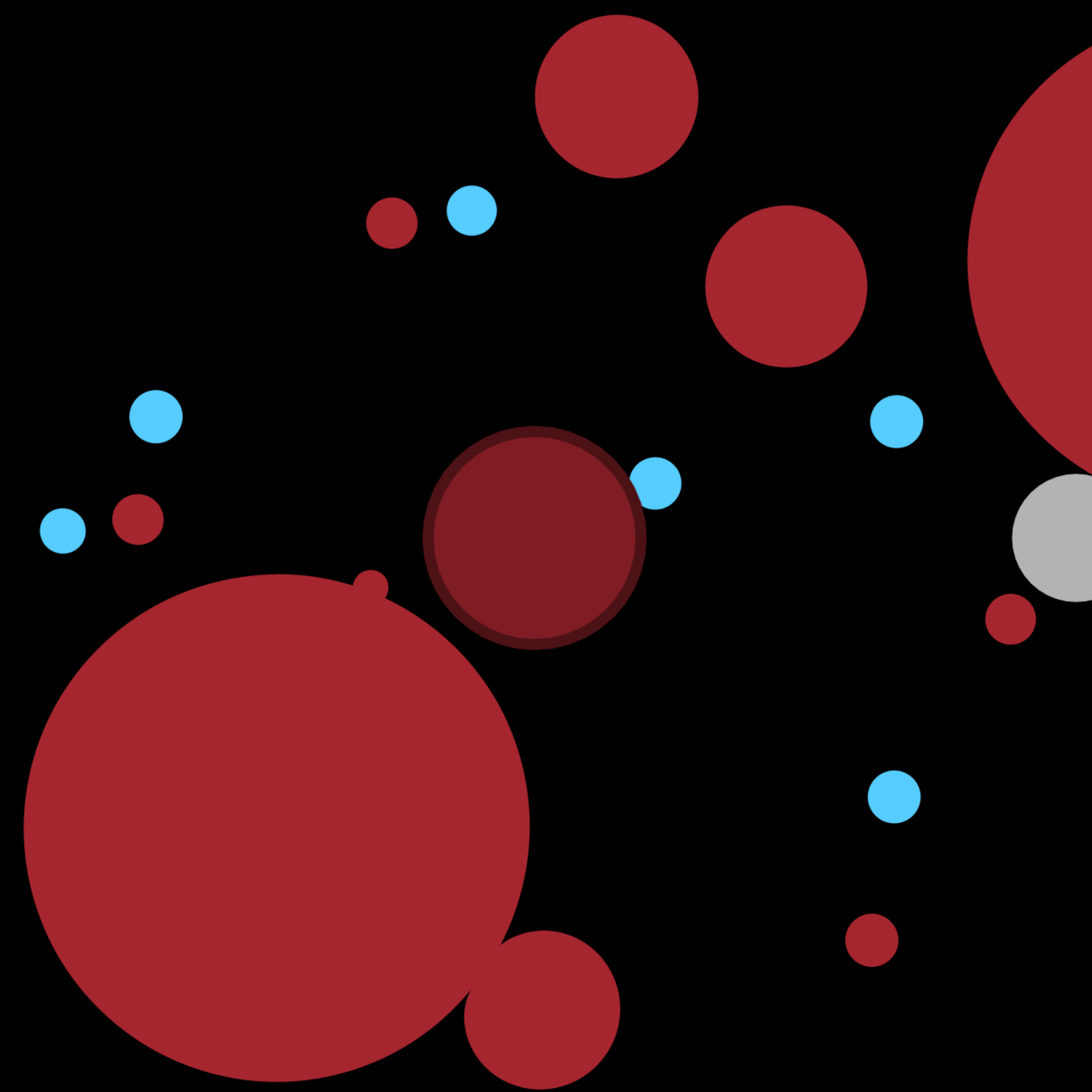
Farbe

Die Farbe der Formen ist eine dunklere Variante der Grundfarbe des Landes. Einzelne Formen werden außerordentlich dunkel dargestellt – eine solche Form repräsentiert ein Kind das vor Vollendung des ersten Lebensjahres gestorben ist.

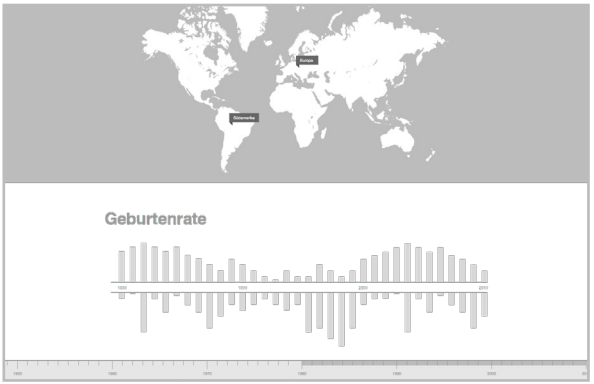




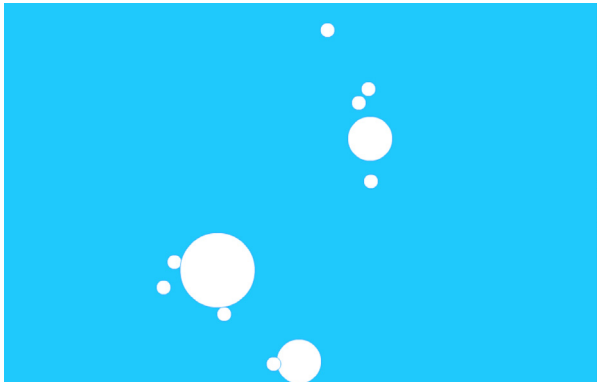




Entwurfsprozess

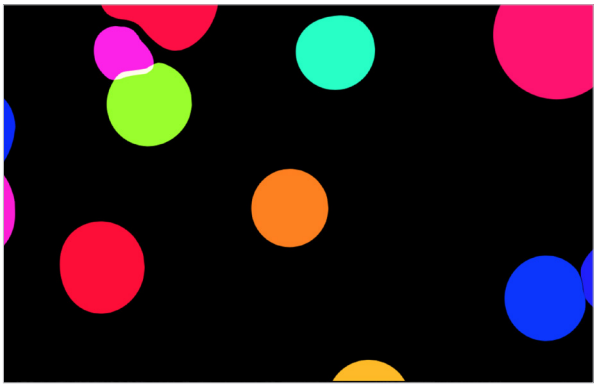


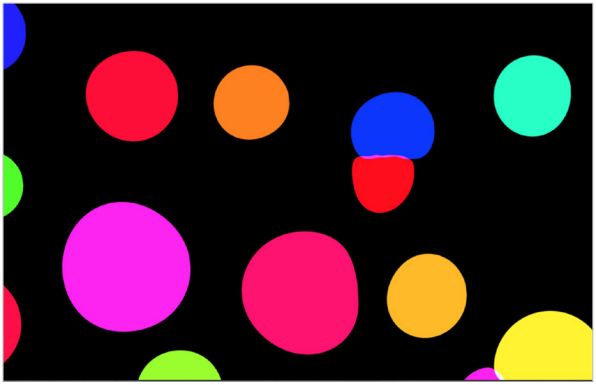
Zu Beginn des Projektes viel es mir schwer mich von eine konkreten Anwendungsstruktur zu lösen. Mein erster Storyboard-Entwurf zeigt deswegen eine sehr klassische Anwendung, bei der zuerst auf einer Karte mit Pins Länder selektiert werden, deren Geburtenrate dann als einfaches Balkendiagramm dargestellt wird. Eine Timeline erlaubt das Auswählen einer Zeitspanne. Um eine abstrakt entworfene Visualisierung zu erstellen löste ich mich komplett von diesem Konzept.



Bei meinem Kurzprojekt zum Thema Chronometer im Fach „Programmiersprachen“ hatte ich bereits ein Partikelsystem mit Gruppierung zur Visualierisung verwendet. Hier wurden die Daten nicht in Echtzeit generiert: Die Kreise tauchen immer dann auf, wenn eine volle Minute oder Stunde der Computerzeit erreicht ist – unabhängig von der bisher vorhanden Zahl der kleineren Einheiten. Es formen also nicht zwingend je 60 Minuten eine Stunde, sondern die vorhanden Minuten die aus dieser Stunde stammen. Die Größe der Einheiten gibt also keine Information über die darin vereinten kleineren Einheiten. Die Flash-Datei dieses Entwurfes liegt bei.

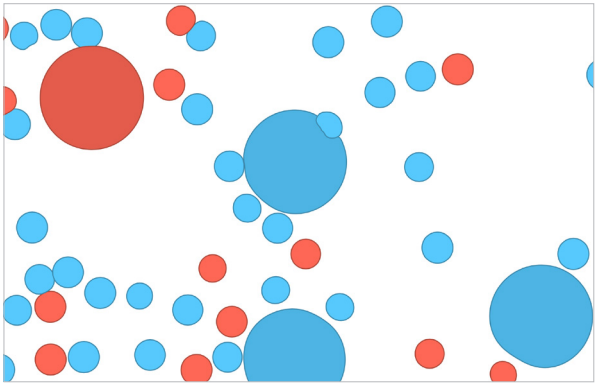
Das Konzept von Partikeln, die sich gruppieren, schien mir für meine Visualisierung als sehr passend: Es wird ein passendes Gefühl für vergehende Zeit geschaffen und die Gruppierung erlaubt einen schnellen Überblick. Ich entschied mich also diese Form der Darstellung für meine Visualisierung zu verwenden.

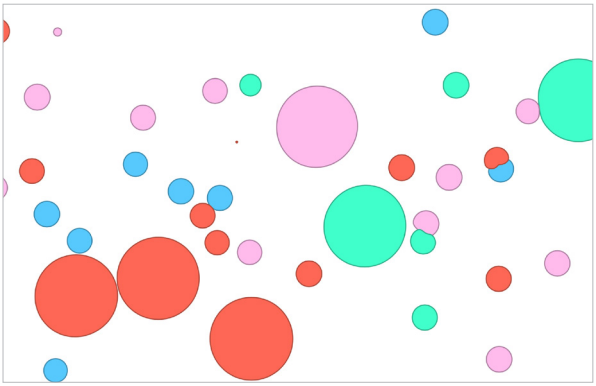




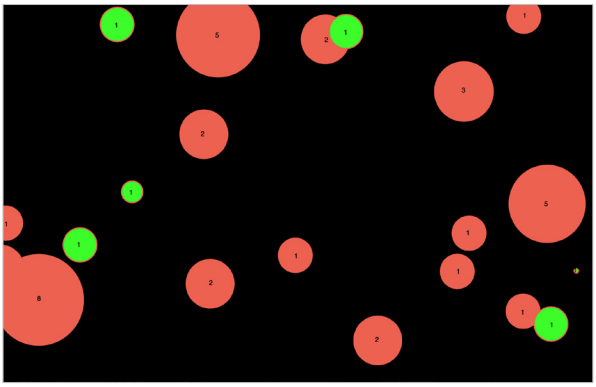
Ich beschloss mich schon sehr früh von statischen Entwürfen zu lösen und begann stattdessen mit der programmatischen Umsetzung meines Entwurfs. Die Daten, zeitliche Entwicklung und die Interaktionen des Systems hielt ich zwar zuvor schriftlich fest, setzte sie aber nicht in einem Grafikprogramm um, sondern entwickelte einen Workflow des „Designing in Browser“, der es mir erlaubte Änderungen direkt im Code umzusetzen und sie so unter realen Bedingungen (mit Interaktion und animiert) zu testen. In verschiedenen Iterationen verfeinerte ich so die Gestaltungsparameter und kam zu meinem finalen Produkt. Das Arbeiten mit agil weiterentwickelten, dynamischen Entwürfen half mir die Visualisierung von Anfang an als großes Ganzes zu betrachten und die Feinheiten Stück für Stück herauszuarbeiten.

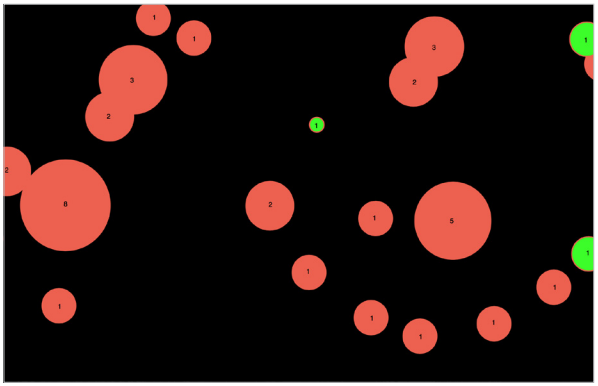
Hier dargestellt ist der Ausgangssketch auf dem ich meine Entwürfe aufbaute: Es handelt sich um ein modifiziertes Beispiel der verwendeten Libary. Dabei werden zu Beginn dynamische Formen zufällig erzeugt, die dann über die Ansicht fahren und von einander abstoßen.



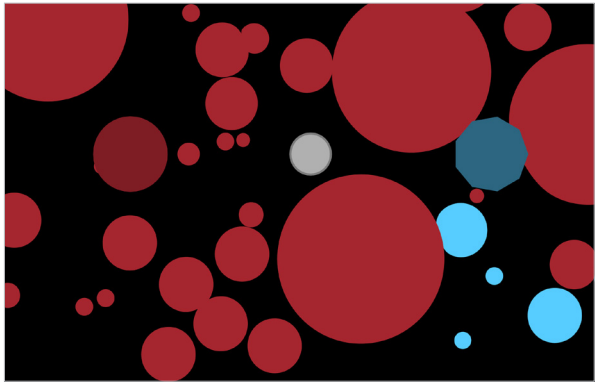


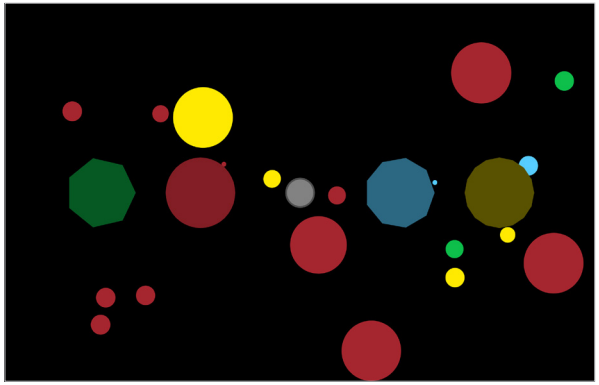
Ich erstellte einen Entwurf, in dem sich die neuen Formen bereits in Abhängigkeit von vergehender Zeit generierten. Die Formen hatten außerdem einen einfachen Gruppierungsalgorithmus: Jede zehnte Form ist größer als alle anderen und nimmt diese in sich auf (die Formen wachsen nicht, wie bei späteren Entwürfen, dynamisch zusammen, die Größen sind frei definiert). Die farbliche Gestaltung dient hier nur der Unterscheidung und ist optimiert für Debugging. Der Code dieses Entwurfes liegt bei.



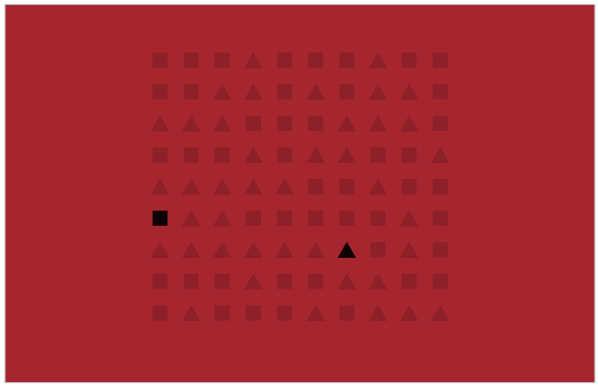


Im nächsten Schritt entwickelte ich den finalen Gruppierungsalgorithmus, damit die Formen „zusammenwachsen“ und sich die Fläche der größeren Einheiten aus den Flächen aller enthaltener Einheiten ergibt. Der globale Rhythmus konnte in diesem Entwurf bereits programmatisch auf ein Land eingestellt werden, eine Interaktion war aber noch nicht möglich.

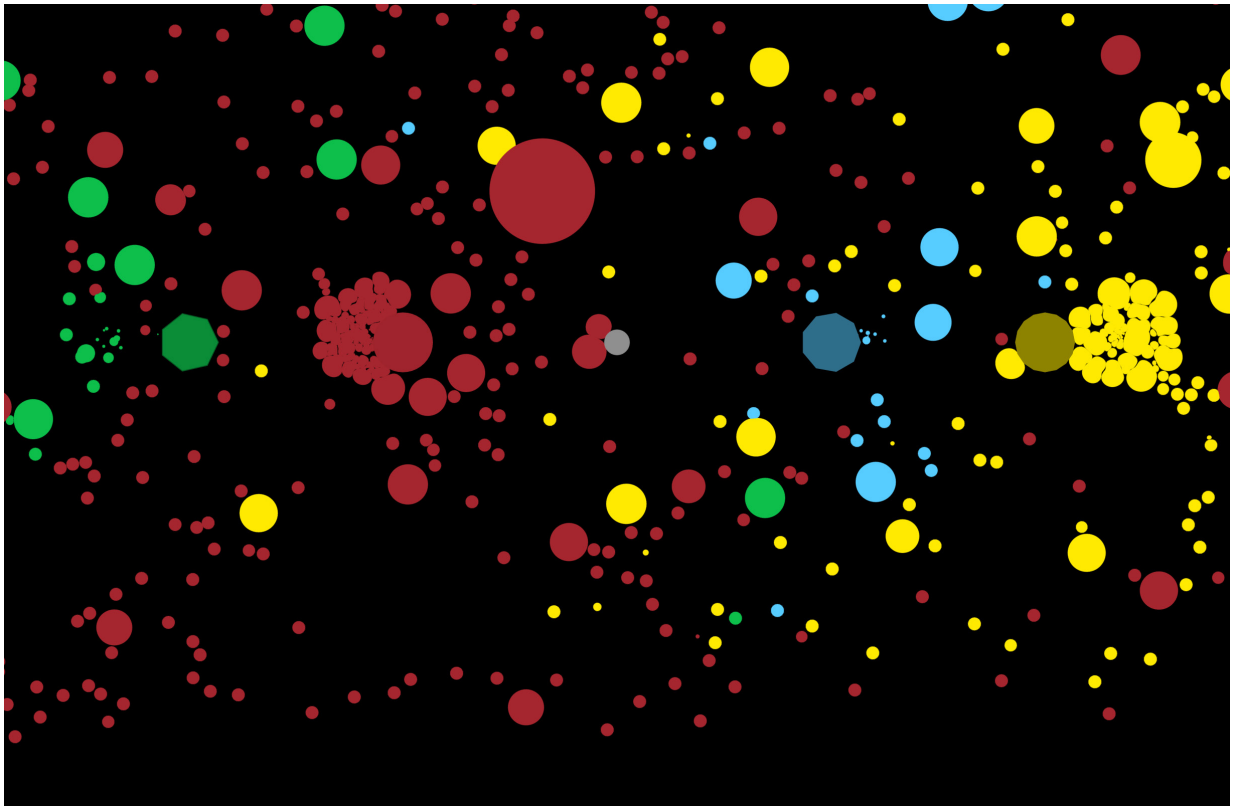




Der darauf folgende Entwurf war bereits die vollkommen funktionsfähige Finalisierung der oberen Informationsebene: Ich fügte zuerst die Ländersymbole, den Zeitindikator und die damit verbunden Interaktionen hinzu, dann den Zoom-Out und die Intiierungssequenz.



Ich beschloss dem System noch eine weitere Informationsebene hinzuzufügen und entwickelte im letzen Schritt den Zoom-In und die Darstellung und Generierung der inneren Ansicht von Personengruppen.



Das finale System an seiner Leistungsgrenze

Technologie

Sprache & Code

Die Anwendung ist als Web-App in HTML, CSS und JavaScript geschrieben und verlässt sich dabei auf moderne Standarts. Mit Hilfe der Open-Source-Libary Paper.js, die für das arbeiten mit SVG-Vektorgrafiken mit JavaScript entwickelt ist wird die Grafik gezeichnet – alle grafischen Elemente sind mit dieser Libary erzeugte Pfade. Die Darstellung erfolgt über das HTML5-Tag „Canvas“, das mit Hilfe der HTML5-Fullscreen-API als Vollbild angezeigt werden kann.

Die Anwendung funktioniert in verschiedenen Browsern, die bester Performance hat sie jedoch in Google Chrome. Wird die Web-App von einem Server geladen und nicht lokal gestartet, so ist sogar noch performanter.

Insgesamt besteht die Anwendung aus ca. 1050 Zeilen Code, der sehr ausführlich kommentiert ist. Die finale Version des Codes liegt bei.

Responsiveness

Da es sich um Vektorgrafiken handelt ist die Darstellung auflösungsunabhängig. Zusätzlich wurde die Anwendung so entwickelt, dass sie komplett responsive ist, also in verschiedenen Auflösungen und Seitenverhältnissen einwandfrei funktioniert. Eine Mindestbreite der Ansicht ist zwar nötig damit die „Länder“ nebeneinander passen, ansonsten werden jedoch alle Werte ständig an die aktuelle Fenstergröße angepasst.

Eine Ansicht bei der das besonders zum Tragen kommt ist die Detailansicht der Gruppierungen: Hier sind die einzelnen Elemente in einem gleichmäßigen Raster angeordnet. Damit das bei wirklich allen Seitenverhältnissen funktioniert wird zu erst ermittelt, ob die Darstellung horizontal oder vertikal mehr Raum hat und die Abstände zwischen den Formen werden in beide Dimensionen aus dem kleineren Wert berechnet. So wird einerseits ein einheitliches Raster erzeugt, andererseits garantiert, dass das Raster zwar bildschrimfüllend angezeigt wird, aber nicht abgeschnitten werden muss.

Soft Bounce

Die Pfade bestehen aus Bézier-Kurven, es bietet sich also an sie nicht als statische Objekte zu sehen. Sie reagiere deshalb beim Aufeinanderprallen aufeinander und verhalten sich wie weiche, elastische Objekte. Das gibt des Visualisierung Dynamik und Spannung. Durch die realistische Impulsweitergabe wirkt das System außerdem natürlicher und verhält sich erwartungsgemäßer.

Der Effekt wird dadurch erzeugt, dass es sich bei den Objekten nicht tatsächlich um Kreise handelt, sondern um Kurven deren Ankerpunkte sternförmig um ein Zentrum angeordnet sind. Nur durch ein Smoothing erscheinen die Polygone kreisförmig. Die einzelnen Ankerpunkte können vom Zentrum weg, oder in Richtung Zentrum verschoben werden, versuchen aber immer wieder ihre Ausgangsposition einzunehmen, dabei richten sie ihre Geschwindigkeit nach der Entfernung zu den jeweils links und rechts liegenden Ankerpunkten.

```
// Shape
this.numSegment = Math.floor((2*this.maxRadius*Math.PI)/(this.maxRadius/2));
this.boundOffset = [];
this.boundOffsetBuff = [];
this.sidePoints = [];

// For each segment craeate a new point, and a corresponding direction and offset from the cen-
ter
for (var i = 0; i < this.numSegment; i ++) {
        this.path.add(new Point());

        // Initial segment-direction
        this.sidePoints.push(new Point({
                angle: 360 / this.numSegment * i,
                length: 1
        }));

        // Initial offset is the radius
        this.boundOffset.push(this.radius);
        this.boundOffsetBuff.push(this.radius);
}

this.path.closed = true;
this.path.smooth();
```

Wegfindung

Bei der Initialisierung wird den sich bewegenden Elemente eine zufällige Richtung und eine zufällige Geschwindigkeit zugewiesen. Diese können sich dann durch Abprallen von entweder anderen Elementen oder dem Rand der Ansicht ändern. Dabei verhalten sie sich basierend auf eine physikalischen Simulation (entweder Impulsweitergabe oder Einfallswinkel gleich Ausfallswinkel).

Sollen sich zehn Elemente gruppieren, so beginnen sie sich aufeinander zu zu bewegen um sich bei Berührung zu verbinden. Damit dies geschieht sucht sich in dem Moment in dem die Gruppierung ausgelöst wird jedes Element aus den zu gruppierenden Elementen jenes aus, das sich am nächsten an dem Element befindet und größer als es selbst ist und bewegt sich darauf zu.

Im Moment der Berührung wird das Zentrum des größeren Elements zum Ziel des kleineren Elements erklärt. Das kleinere Element verschwindet dann also im größeren und wird zu diesem addiert, so bald es sich komplett im Radius des größeren Elements befindet.

```
// Get move to
for (var i = 0; i < balls.length; i++) {
    // If this has no target yet
    // If the balls should be joining, come from the same country, have the same join Id and
the other ball is bigger set it as a moveTo
    if (this.isDying == false && this.shouldJoin && balls[i].shouldJoin && this.shouldJoinId ==
balls[i].shouldJoinId && this.countryId == balls[i].countryId) {

        var smallestDist = 50000000;
        var dist = this.position.getDistance(balls[i].position);

        if (dist > 0 && dist < smallestDist && dist > this.radius + balls[i].radius) {
                smallestDist = dist;
                this.hasMoveTo = true;
                this.moveTo = balls[i].position;
                this.moveToRadius = balls[i].radius;
                this.moveToId = balls[i].id;
        }
    }
}

// If it has a target align the velocity to get there
if (this.hasTarget)    {
    var buff = this.velocity.length;
    this.velocity = (this.target - this.position);
    this.velocity.length = buff+2;
}

// If it has a moveTo align the velocity to get there
if (this.hasMoveTo == true && this.isDying == false)  {
    var dist = this.position.getDistance(this.moveTo);

    if (dist > this.radius + this.moveToRadius && this.id != this.moveToId)  {
        var buff = this.velocity.length;
        this.velocity = (this.moveTo - this.position);
        this.velocity.length = buff;
    }
}

// If it has reached is destination give it a new random direction
if (this.position.getDistance(this.moveTo) <= this.radius)    {
    this.hasMoveTo = false;
    this.velocity = new Point({
            angle: 360 * Math.random(),
            length: Math.random() * 2 + 4
        });
}
```

Dynamische Generierung

Der Code der Anwendung ist komplett objektorientiert. Zu Beginn werden nur die Länder von Hand initialisiert, alle weiteren Elemente werden aus den entsprechenden Ländern heraus (teilweise zeitabhängig) generiert und beim Gruppieren und Zusammenfassen auch wieder entfernt.

Die Hauptelemente, also die Kreise die Kinder in der Standartansicht repräsentieren, verhalten sich alle gleich und haben gleiche Eigenschaften, sie unterscheiden sich nur durch ihre Größe. Die in der Detailansicht einer Gruppe angezeigten Symbole unterscheiden sich jedoch stärker: hier werden Geschlecht und Kindersterblichkeit zusätzlich angezeigt. Hier für wird beim Generieren der Elemente basierend aus den dem Land zugehörigen Wahrscheinlichkeiten für jedes Element neu per Zufall ermittelt, welche Eigenschaften zutreffen.

Dynamisches Pulsieren

Ist die Geschwindigkeit mit der sich die Helligkeit beim Pulsieren verändert fest angegeben, so erscheint sie bei langsamen Rhythmen viel zu schnell, bei sehr schnellen wird eventuell die höchste Helligkeit hier erreicht. Die Änderungsgeschwindigkeit wird deshalb immer in Abhängigkeit vom globalen Rhythmus errechnet.

```
// Create small shapes
for (var i = 0; i < this.size; i++)    {

    // Get random gender
    if (Math.random() < this.male)    {
                this.balls[i] = new Path.RegularPolygon(new Point (10,10), 4, 30);
    } else     {
        this.balls[i] = new Path.RegularPolygon(new Point (10,10), 3, 30);
    }

    // Get child mortality
        this.balls[i].fillColor = this.color;
        if (Math.random() < this.mortality)   {
        this.balls[i].fillColor.brightness -= 0.6;
        } else  {
        this.balls[i].fillColor.brightness -= 0.1;
        }
        this.balls[i].fillColor.alpha = 0;
}
```

```
// Get the lightUpSpeed according to the current global rhythm
var lightUpSpeed = (rhythm / this.rhythm) * 0.005;
if (lightUpSpeed > 0.2)        {
       lightUpSpeed = 0.2;
} else if (lightUpSpeed < 0.01)       {
       lightUpSpeed = 0.01;
}
```