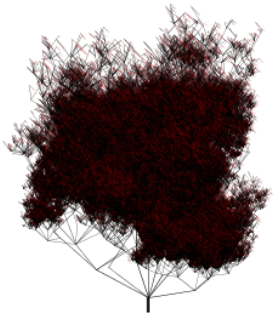


Processing

Programming Basic 17 - 27.09.2013

Dokumentation von Florian Herzog

Inhaltsverzeichnis

vorwort(0);

Diese Dokumentation ist in Form eines Tagebuchs geführt. Es wiederspiegelt alle Tage des Kurses Programming Basic.

Die Blogeinträge, den jeweiligen Source Code, sowie die uns gegebenen Übungen sind über die aufgelisteten Links einsehbar.

firstDay(1);

EINFÜHRUNG IN DIE COMPUTERWELT – PROCESSING PROGRAMMIERSPRACHE – ZEICHENAUSGABE – KOMPLEXE FORMEN – FARBE UND TRANSPARENZ – VARIABLEN – FUNKTIONEN

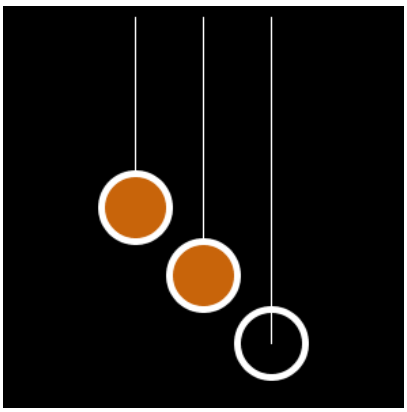
Blogeintrag
http://blogs.iad.zhdk.ch/codingspace/2013/09/19/herzogflorianlesson1/

CodingSpace-Lektionen
http://blogs.iad.zhdk.ch/codingspace/lektionen/lektion-1/
http://blogs.iad.zhdk.ch/codingspace/lektionen/lektion-1-5/
http://blogs.iad.zhdk.ch/codingspace/lektionen/funktionen/

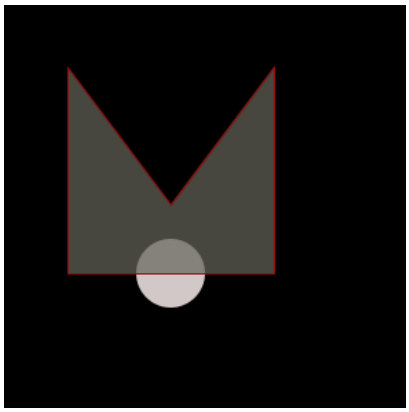
DIENSTAG, 17.09.2013
Wir begannen traditionell mit einer kurzen <icebreaker round> gefolgt von einem Einblick in die Computerwelt/Geschichte/Generative Art/ Programming Pioneers (zB: Sketch Pad – Ivan Sutherland / Karl Sims), sowie in die Programmiersprache <Processing>, welche auf <Java> basiert.
Die ersten Übungen bestanden darin eine Zeichenausgabe mit einfachen sowie komplexen Formen inkl. Farbe/Transparenz zu erreichen. Dadurch verwendeten wir <Variablen>, sogenannte Platzhalter wie (zB: int, float, boolean, char) sowie bestehende <Funktionen> von <Processing>.

EIGENERFAHRUNG
Obwohl ich mit <Java> bereits zu tun hatte und so einiges kenne, ist für mich die Art und Weise wie mit <Processing> gearbeitet wird sehr ungewohnt. Es ist das erste Mal, dass ich Programmcode benutze um direkt einen grafischen Output zu erzielen. Ich finde es sehr interessant die Ähnlichkeiten zu <Java> zu sehen und mein vorhandenes Wissen mit den neuen Erkenntnissen zu erweitern.

Erste Aufgabe Teil 1
Hier erzeugte ich meinen ersten Grafischen Output mit einfachen <ellipses> und <lines>. Über die <fill> sowie <stroke> Funktion konnte ich die Füllfarbe sowie Kontur festlegen.



Erste Aufgabe Teil 2
Über die Funktion <vertex> konnte ich eigene Formen erzeugen. Viele Funktionen von Processing kannte/kenne ich nicht. Doch fasziniert mich diese Art zu programmieren.



drawingTool(2);

FRAMEWORK EVENTS – BEDINGUNGEN – ZUFALLSZAHL – SCHLEIFEN – EIGENE FUNKTIONEN

Blogeintrag
http://blogs.iad.zhdk.ch/codingspace/2013/09/19/herzogflorianlesson2/

CodingSpace-Lektionen
http://blogs.iad.zhdk.ch/codingspace/lektionen/lektion-2/
http://blogs.iad.zhdk.ch/codingspace/lektionen/bedingungen/
http://blogs.iad.zhdk.ch/codingspace/zufallszahl/
http://blogs.iad.zhdk.ch/codingspace/lektionen/schleifen/

MITTWOCH, 18.09.2013
Wir lernten wie <Processing> mit der <setup> und <draw> Funktion arbeitet. Ebenfalls lernten wir die <Bedingungen, Zufallszahlen und Schleifen> kennen. Mit dem Neugelernten bekamen wir die Aufgabe ein Drawing-Tool zu programmieren. Eine einfache Form von dem bekannten <Paint-Programm>.

EIGENERFAHRUNG
<Processing> sagt mir immer mehr zu. Ich finde es toll, dass man vorzu ein grafisches Resultat sieht. Dadurch ist die ganze Programmiererei nicht ganz so trocken. Die <Bedingungen, Zufallszahlen, Schleifen und eigene Funktionen> studierte ich bereits im Privaten mittels <Java>. Also nichts Neues für mich. Dennoch ist es spannend es mit anderen Sachen zu kombinieren. Leider muss ich bereits am zweiten Tag feststellen, dass ich von meinem, im Vorfeld angeeigneten, Wissen bereits enorm viel vergessen habe. Ich sehe momentan kein grossen Vorteil gegenüber den Anderen, welche noch nie programmiert haben.

Bei diesem Beispiel stiess ich auf ein Problem als ich einen Button mit der Funktion einer Farbübergabe machen wollte. Der Lösungsweg fiel mir schwer. Die Denkensweise, die man sich zum Programmieren aneignen muss, macht mir stets Schwierigkeiten. Ich denke, ich habe alles verstanden und am Tag darauf merke ich, dass ich es am Vortag nicht verstanden habe.

Weitere Beispiele sind unter dem Blog-Link zu finden.

growingForest(3);

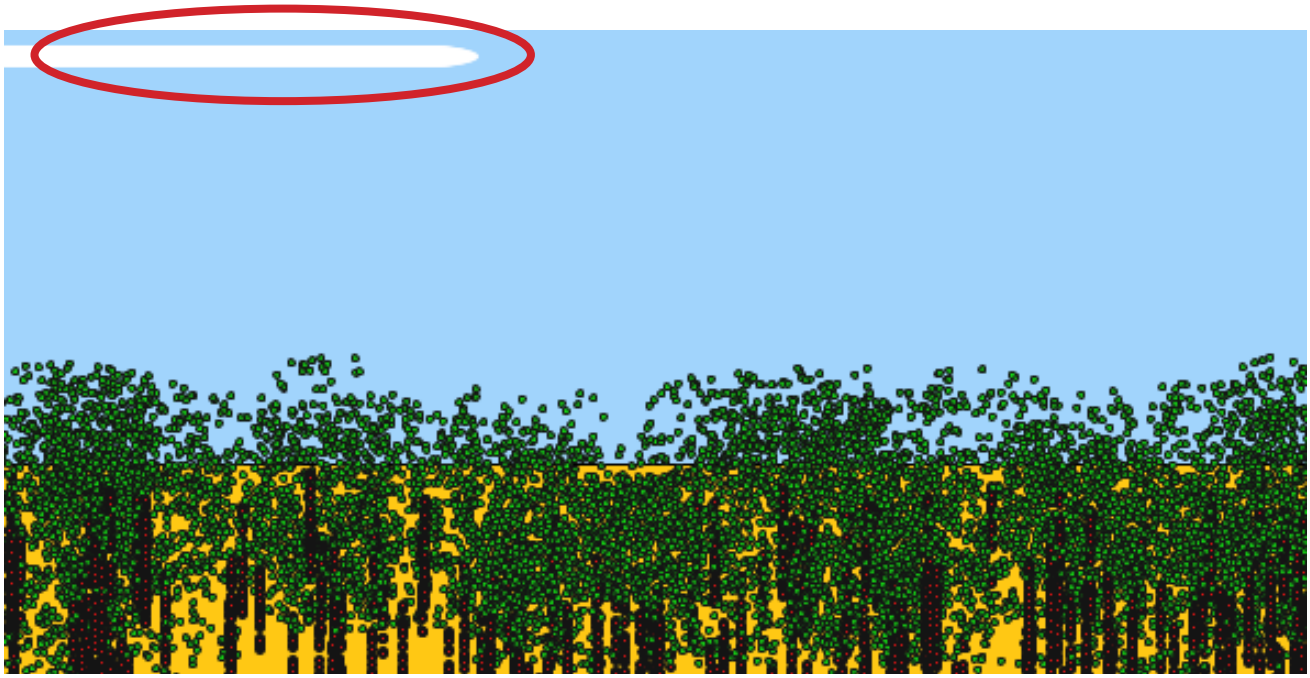
ZUSATZSCHLEIFEN – KOORDINATENSYSTEM

Blogeintrag
http://blogs.iad.zhdk.ch/codingspace/2013/09/19/herzogflorianlesson3/

CodingSpace-Lektionen
http://blogs.iad.zhdk.ch/codingspace/lektionen/lektion-4/
http://blogs.iad.zhdk.ch/codingspace/koordinatensystem/

Donnerstag, 19.09.2013
Wir bearbeiteten hauptsächlich das Koordinatensystem mit welchem man ganze Formen verschieben, skalieren und rotieren kann. Die Aufgabe bestand darin, einen eigenen Wald zu erstellen, der seine Form/Grösse etc. verändern kann.

EIGENERFAHRUNG
Als ich die <random> Funktion für die Blätter und den Stamm benutzte und diese mit <push/popMatrix> kombinierte, stiess ich auf Hindernisse. Das Problem war, dass der Stamm immer verschoben zu den Blättern stand. Ich löste das Problem mit einere eigenen Variabel und Verschachtelung der <push/popMatrix>. Zwei weitere Probleme konnte ich noch nicht lösen. Die Wolken, welche mit jedem <pressedMouse> nach rechts fährt, wird nicht mehr gelöscht. Jedoch kann ich den <background> nicht in die <draw> Funktion nehmen. Es würde das Wolkenproblem zwar lösen, jedoch würde nur immer ein Baum angezeigt werden. Das nächste Hinderniss, welches ich noch nicht meistern konnte ist, dass die Bäume im Hintergrund nicht kleiner werden und nicht hintereinander stehen. Doch warte ich ab, bis wir die <arrays> anschauen, um das Problem zu lösen.



recursiveFunctions(4);

KOORDINATENSYSTEM – REKURSIVE FUNKTIONEN – FRAKTALE

Blogeintrag
http://blogs.iad.zhdk.ch/codingspace/2013/09/23/herzogflorianlesson4/

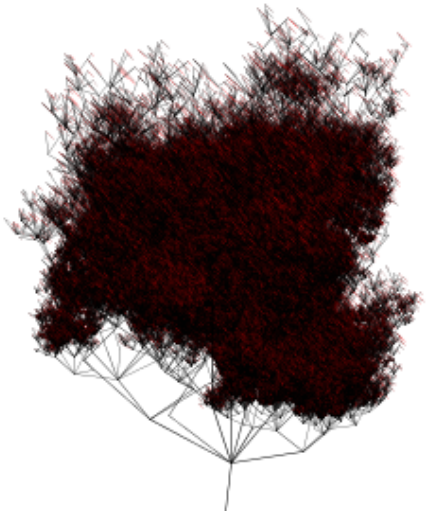
CodingSpace-Lektionen
http://blogs.iad.zhdk.ch/codingspace/koordinatensystem/
http://blogs.iad.zhdk.ch/codingspace/lektionen/lektion-5/

FREITAG, 20.09.2013
Wir bearbeiteten die rekursive Funktion sowie ein weiteres Beispiel wo die Position der Maus direkten Einfluss auf die <push/popMatrix> hat. Bei der Aufgabe mit der rekursiven Funktion, sollten wir mit den Werten herumspielen und die Veränderungen des grafischen Outputs studieren, damit wir den ganze Code besser verstehen. Ebenfalls gab es eine weitere Knacknuss, bei welcher wir einen kleinen Smiley um einen Grösseren kreisen lassen mussten (siehe CodingSpace-Lektionen).

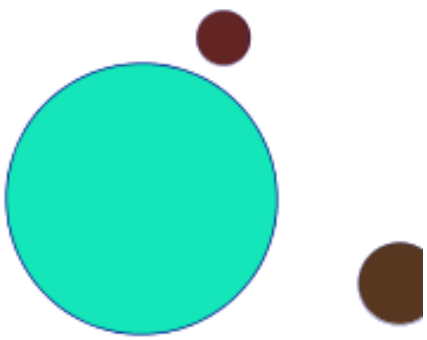
EIGENERFAHRUNG
Mit der Anwendung der rekursiven Funktion konnte ich den Code gut nachvollziehen.
Jedoch bereitet es mir Schwierigkeiten das Prinzip auf etwas völlig anderes anzuwenden. Ich werde noch Tutorials anschauen, um dieses Problem zu beheben.

Beim zweiten Beispiel, wo sich der eine Smiley um den Anderen dreht, tüftelte ich einwenig an der Verschachtelungsmöglichkeiten von <push/popMatrix> herum.

Hier versuchte ich die bereits vorhandene rekursive Funktion so ab zu ändern, dass sich ein sehr ralistischen und detail-reichen Busch ergibt.



Ich ersetzte die Smileys mit verschieden farbigen Ellipsen und programmierte die zwei Kleinen so, dass sie um die grosse Ellipse kreisen.



arrays[5];

ARRAYS – LISTEN – SVG + BILDER – VEKTOREN – ANIMATION

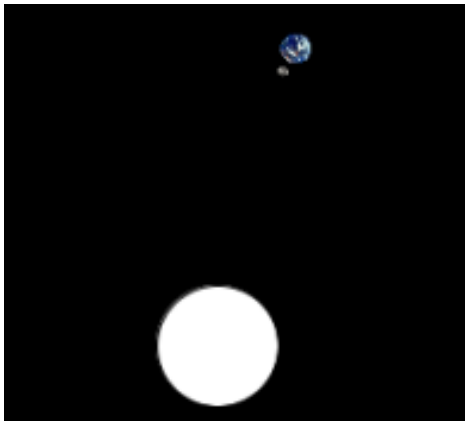
Blogeintrag
http://blogs.iad.zhdk.ch/codingspace/2013/09/24/herzogflorianlesson6/

CodingSpace-Lektionen
http://blogs.iad.zhdk.ch/codingspace/lektionen/arrays-listen/
http://blogs.iad.zhdk.ch/codingspace/svg-bilder/
http://blogs.iad.zhdk.ch/codingspace/lektionen/vektoren/
http://blogs.iad.zhdk.ch/codingspace/lektionen/animation/
http://blogs.iad.zhdk.ch/codingspace/animation-loesung/

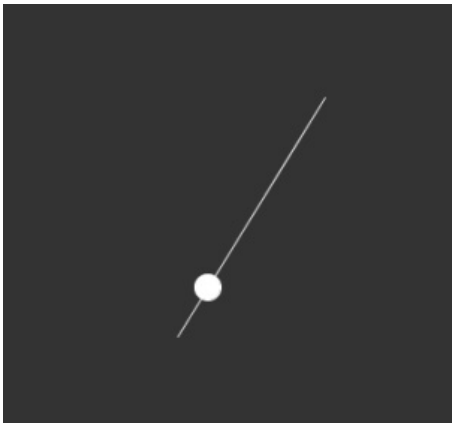
DIENSTAG, 24.09.2013
Nach den Kurzpräsentationen der Hausaufgaben, schnitten wir die Themen <Arrays, ArrayList, SVG + Bilder, Animationen> an. Wir mussten zuerst mit <Arrays, ArrayList, SVG + Bilder> herumtüfteln, um die ganze Sache besser zu verstehen.

EIGENERFAHRUNG
Die Logik hinter der ganze Sache zu verstehen und korrekt anzuwenden macht Spass, jedoch muss man zuerst darauf kommen wie's funktioniert.
Über einfache <if> Abfragen kann man sehr schnell und effizient der Lösung näher kommen. Einfache Datentypen wie <int> oder <boolean> eignen sich super, um weitere Bedingungen einzubauen. Die <Vektoren> muss ich nochmals genauer anschauen. Ich habe zwar das Prinzip verstanden, jedoch bin ich noch sehr unsicher.

Hier baute ich bei einer bereits vorhandenen Übung eine drag&drop Funktion ein und lies zwei Bilder um das Hauptobjekt kreisen. Die Bilder speicherte ich in ein <array>. Des Weiteren veränderten wir einen uns gegebenen Code (Animation Beispiel 2), um das vorgegebene Ziel zu erreichen.



Bei dieser Aufgabe änderte ich einen bereits vorhanden Code so ab, dass die man mit zwei Klicks einen Vektor generieren sowie ausrichten kann und die Ellipse hin und her fährt.



finalChallenge(6);

KLASSE – ENDAUFGABE

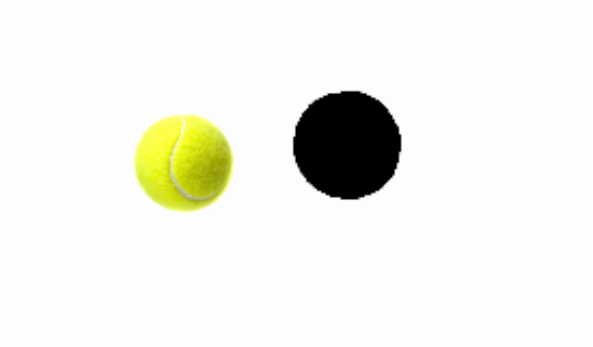
Blogeintrag
http://blogs.iad.zhdk.ch/codingspace/2013/10/01/herzogflorianlesson7/

CodingSpace-Lektionen
http://blogs.iad.zhdk.ch/codingspace/lektionen/klassen/
http://blogs.iad.zhdk.ch/codingspace/endaufgabe/

MITTWOCH, 25.09.2013
Die Klassen, die alles vereinfachen, war das Vormittagsthema.
Am Nachmittag wurde uns die Endaufgabe vorgestellt und wir durften damit starten.

EIGENERFAHRUNG
Die Klassen vereinfachen auf jeden Fall die Sache, sofern man sie anzuwenden weiss.
Die Aufsplitung der Struktur kenne ich zwar, doch ist es eine Weile her seit dem letsten Mal als ich die Klassen in <Java> benutzte. Was mir fehlt ist die Vermittlung der Art zu Denken und Handhabung des Codes. Die Endaufgabe ist gewagt. Einen bestehenden Code zu änderen und verstehen zu müssen ist mühsam. Einerseits finde ich es gut ein grösseres Projekt zu zeigen, damit man nicht immer mit Scheuklappen herumirrt, jedoch sind die meisten total überfordert und schrauben nur an Werten herum, statt Funktionen neu zu programmieren. Ich verbrachte den Nachmittag mich mit dem vorhandenen Code zurecht zu finden.

Bei diesem Beispiel erstellte ich eine Klasse, um die zwei Bälle im Anzeigefeld bouncen zu lassen. Ebenfalls baute ich eine Kollisionsabfrage ein, damit die Bälle voneinander prallen.



breakOut(7);

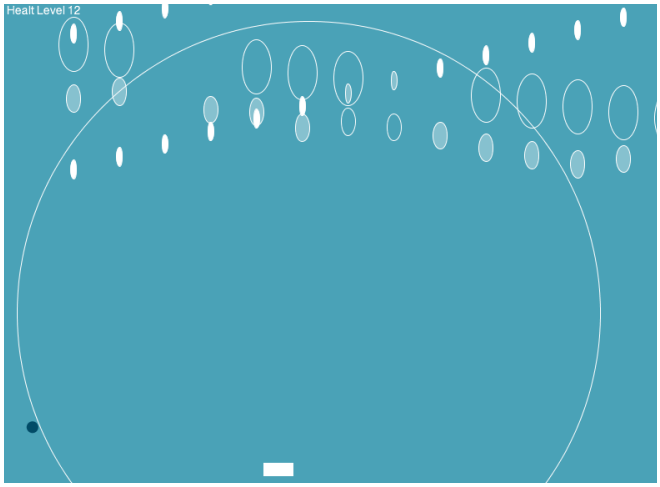
ENDAUFGABE

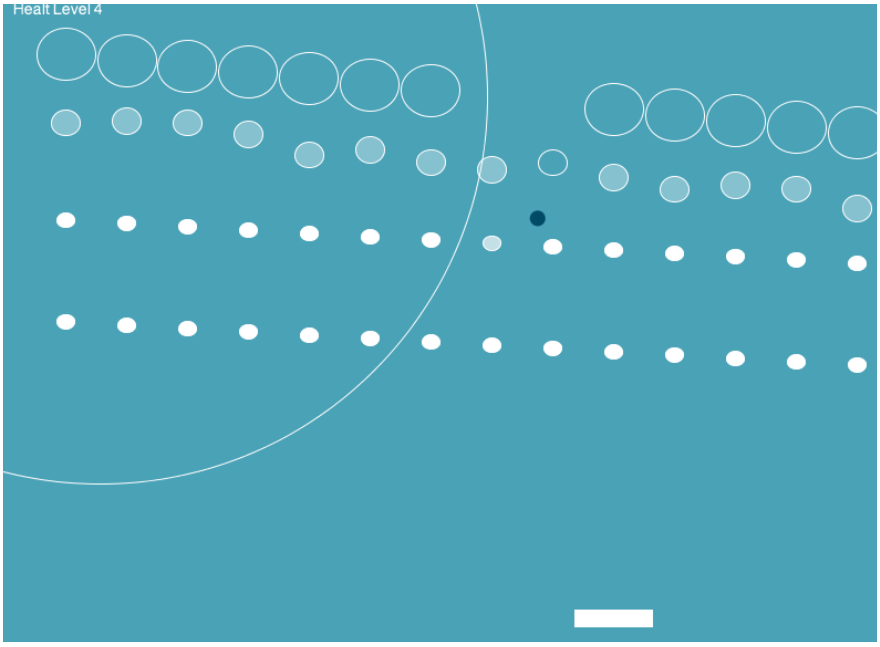
CodingSpace-Lektionen
http://blogs.iad.zhdk.ch/codingspace/endaufgabe/

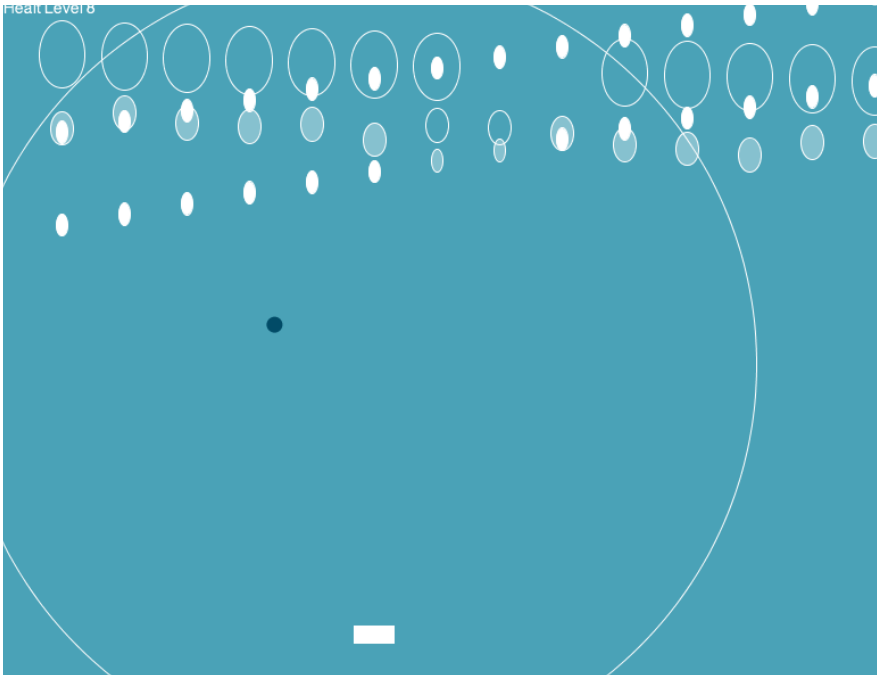
DONNERSTAG, 26.09.2013
Kurze Besprechung des Vorgehens mit Max, danach freies Arbeiten im Unterrichtszimmer.
Max stand für Fragen zur Verfügung.

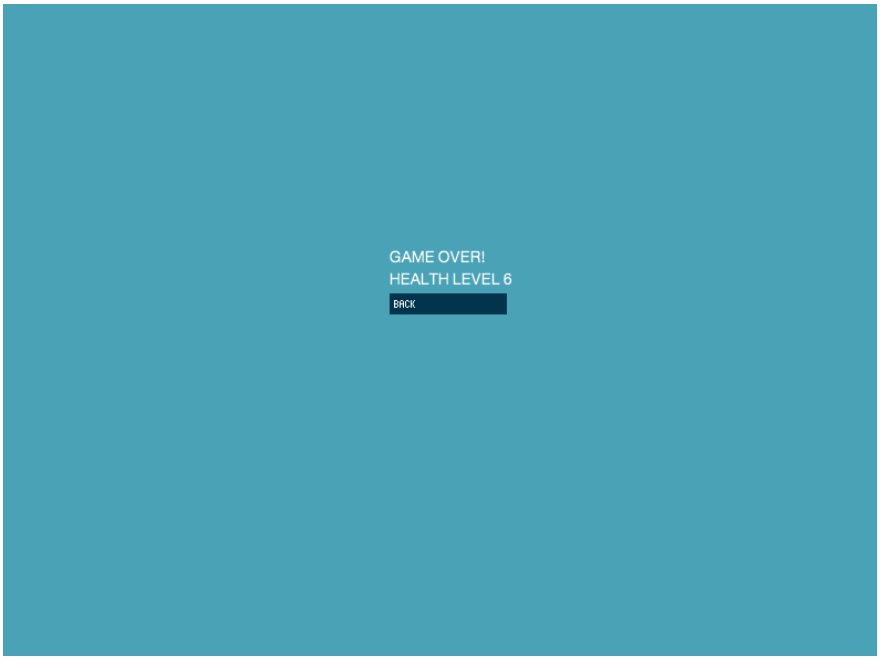
EIGENERFAHRUNG
Max stellte mein Vorgehen auf den Kopf. Ich wollte zu Beginn ein traditionelles GUI gestallten und nicht wirklich viel am eigentlichen Spiel ändern. Max fand das schlecht. Einen <Swing> sollte ich erarbeiten. Meine Unentschlossenheit verschluckte die Zeit des Vormittags. Am Nachmittag kam mir die Idee des <BreakOut> einer Krankheit. Meine Absicht war das GUI sehr einfach und reduziert zu halten, um eine gewisse Sterilität zu vermitteln. Ich wollte eine Flüssigkeite mit Bakterien in einer Petrischale nachahmen. Das eigentliche BreakOut Spiel wollte ich beibehalten, jedoch mit einer gewissen Besonderheit. Da es bei Flüssigkeiten verschiedene Schichten geben kann, wollte ich einen <Floateffekt> erstellen. Die <Bricks> sollen die Bakterien sein, die in der Flüssigkeit verschieden herumfloaten und der Spieler muss mit dem Ball (Antikörper) die Bakterien zerstören. Die Krankheit darf natürlich nicht ausbrechen.

Die meiste Arbeit erledigte ich zu Hause.




Healt Level 4


Healt Level 8


GAME OVER!
HEALTH LEVEL 6
BACK

projectPresentation(8);

ENDAUFGABE - PRÄSENTATION

CodingSpace-Lektionen
http://blogs.iad.zhdk.ch/codingspace/endaufgabe/

FREITAG, 27.09.2013
Wir hatten bis 16:00 Uhr Zeit, um alles fertig zu programmieren. Ab 16:00 Uhr starteten alle Projektpräsentation sowie die Einzelbesprechungen.

EIGENERFAHRUNG
Den Vormittag nutzte ich noch um das GUI zu verbessern. Danach machte ich mir keinen Stress mehr und setzte einen Schlusspunkt. Mit dem Verlauf meiner kleinen Präsentation bin ich, bis auf den zu leisen Sound, zufrieden. Es war sehr spannend alle Projekte zu sehen und sich davon inspirieren zu lassen. Die Einzelbesprechung war ebenfalls gut. Max gab mir ein Feedback und fragte ebenfalls nach, was ich als gut reps. als weniger gut empfunden hatte.

Note: C/D

Zurückblickend kann ich sagen, dass ich viel Neues über Processing und wie es anzuwenden ist gelernt habe. Von der Sprache her, kannte ich bereits sehr viel. Da ich jedoch während den letzten Monaten nicht zum programmieren kam, fiel mir alles sehr schwer, so als hätte ich noch nie programmiert. Es war eine sehr gute Repetition/Auffrischung. Zukünftig werde ich öfters programmieren, so dass ich eine solche Erfahrung nicht wieder machen muss.