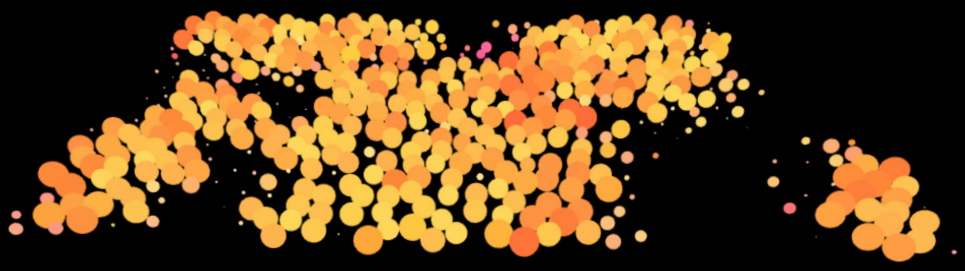


PROGRAMMING BASICS

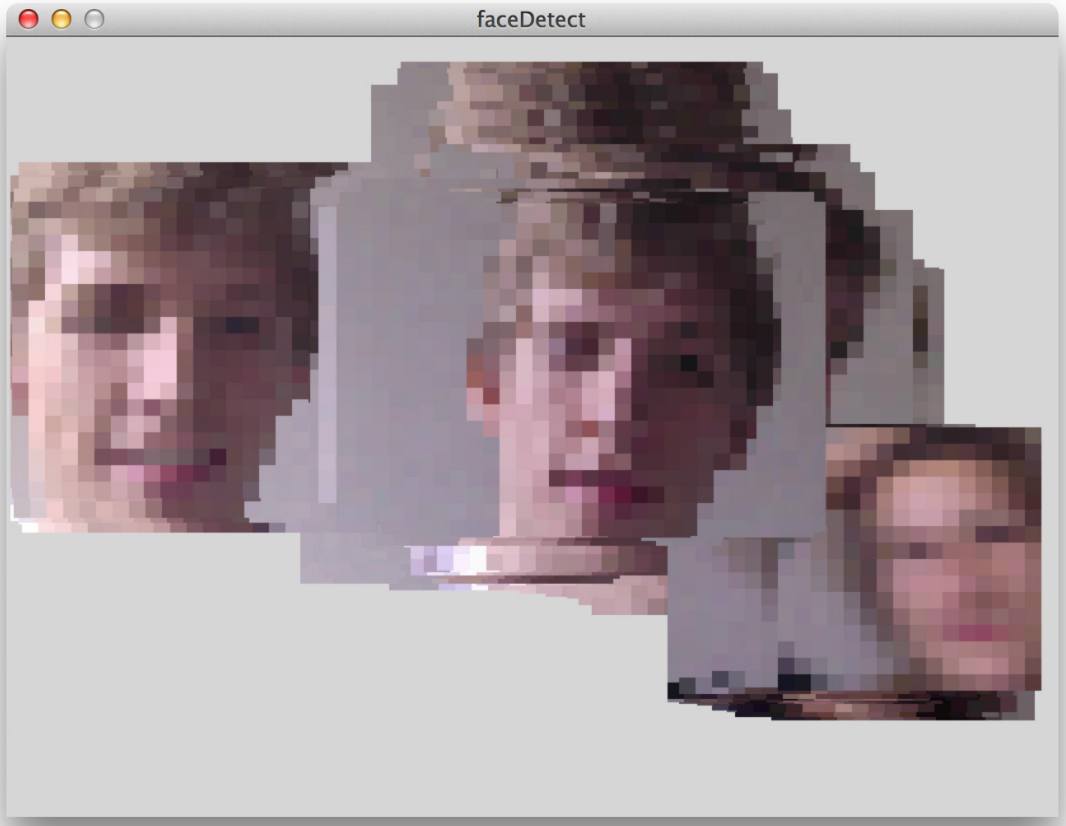
DAVID SIMON

01 FACE DETECTION

On the first day of our Programming Introduction with Processing I used OpenCV[1] to explore the basics of face recognition.
Combining a small algorithm for pixelation with the face recognition data, I created a proof-of-concept application which demonstrates obscuring faces in real-time.

An extension to this sample could be made with a differentiation of different faces, i.e. tracking the face of a single person only. Or a extrapolation of a face's position in case of interrupted tracking.

[1] Open Source Computer Vision - http://opencv.org/


faceDetect


faceDetect

02 DRAWING

We were given the task to write a simple drawing application, which should grant the ability to draw in the color of choice. I created a simple Implementation, using a "Line" class. For the GUI I made use of the library ControlP5[1]. With a rightclick+drag one can delete lines.

[1] ControlP5 - http://www.sojamo.de/libraries/controlP5/


drawingProgram
255
80
151
87
23.22
STROKEWIDTH
RED
ORANGE
YELLOW
LGREEN
DGREEN
BLACK
VIOLET
LBLUE
DBLUE

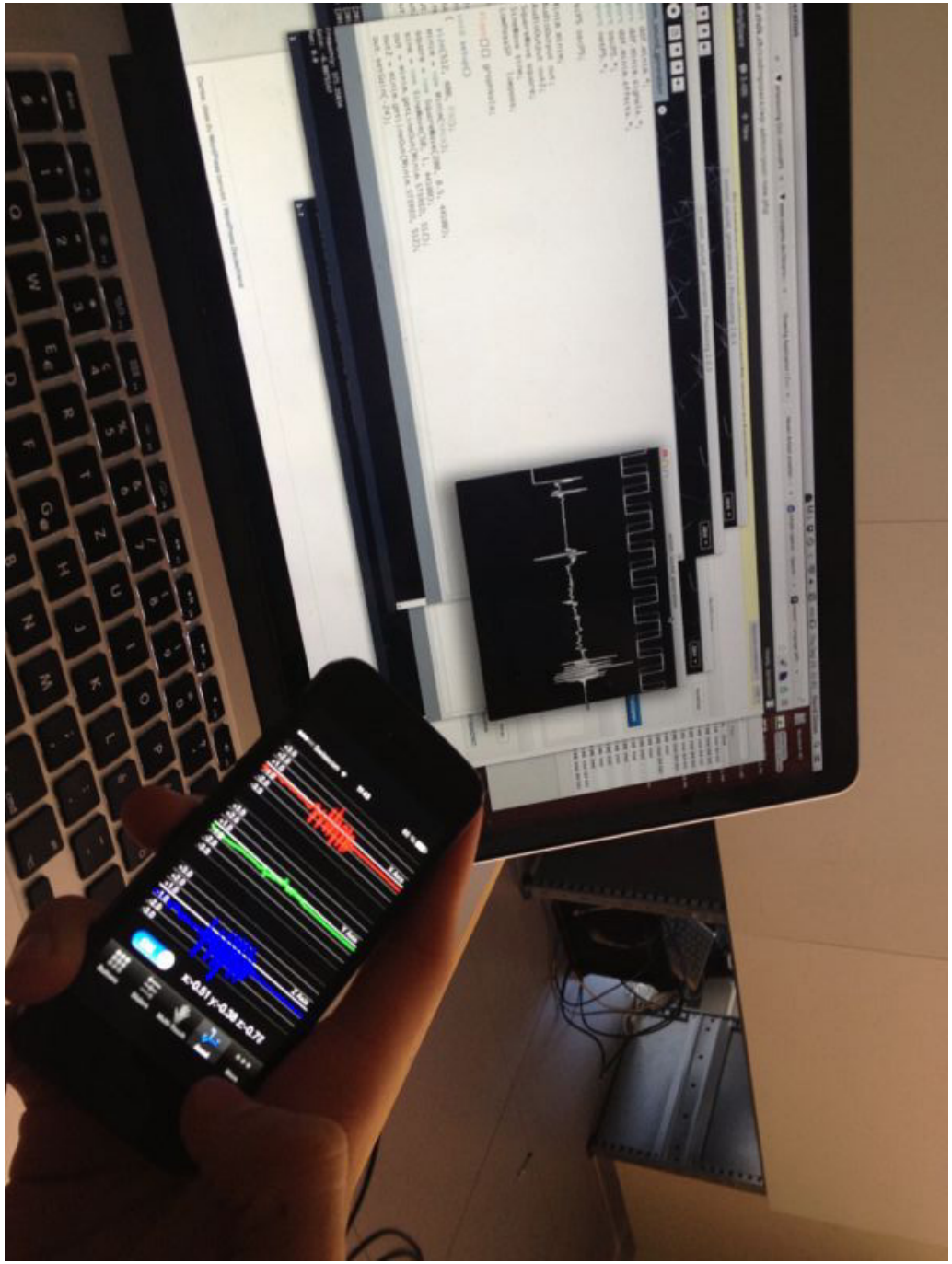
03 SOUND FROM MOVEMENT

I experimented with sound generation from movement using the accelerometer of my IPhone. To send the signal I used the app OSCmote[1]. Alternatively you could use GyrOSC[2] To receive to OSC protocol messages the library oscP5[3] was used.

The input was used to modulate a synthesizer's frequency and gain via the x- and z-axis respectively. The tone was generated with Minim[4].

---

[1] OSCMOTE - http://pixelverse.org/iphone/oscemote/
[2] GyrOSC - http://www.bitshapesoftware.com/instruments/gyrosc/
[3] oscP5 - http://www.sojamo.de/libraries/oscP5/
[4] Minim - http://code.compartmental.net/tools/minim/



03 SOUND FROM MOVEMENT

04 TREE SIMULATION

To create a forest, was the given assignment. The result is a real-time simulation application of a modeled tree population with it's own "DNA".

Trees are represented by circles, small at first, increasing in size as they grow older and finally stagnating. The color of a tree is an expression of their attribute "Resilience". The more resilient a tree becomes the yellower it is.

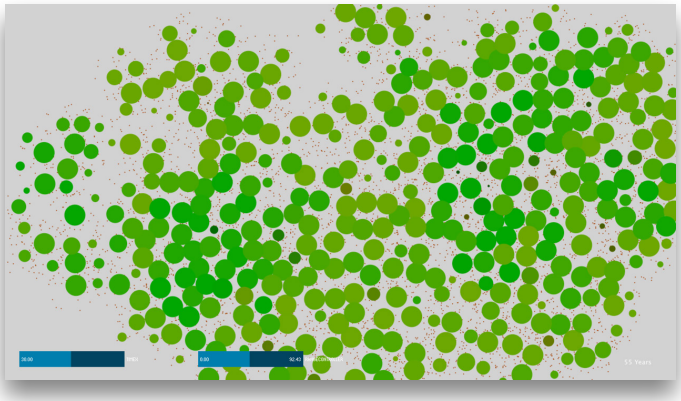
The reproduction cycle inside the virutal environment is a reduced modeled process in which trees send out pollen, which in turn can fertilize trees they hit. If such a fertilization succeeds a tree will then proceed to spread it's seeds which contains a newly created "genome". This new genome is a result of the mixture of the tree's own genome and the one from the pollen that fertilized the tree. There is no concept of genders represented.
Seeds have a chance to germinate and will proceed to do so only if they are not within the range of another tree (read: "underneath it"). All sprouted trees have a variable chance to die, which increases almost linearly with age and then dramatically increases as a tree's age nears 200 years.

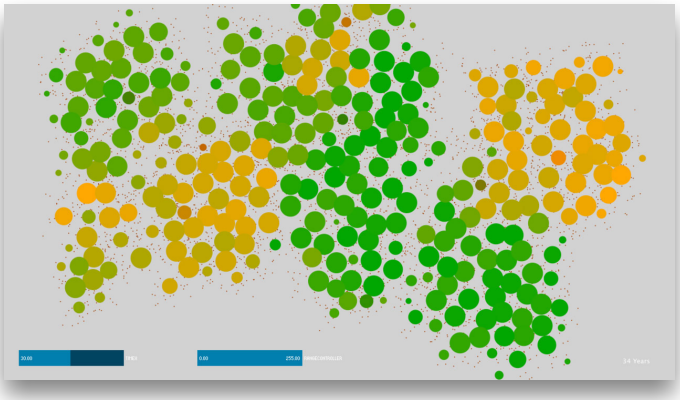
The program allows the user to take influence on the environment. The range of tolerance for resilience can be regulated. This allows to simulate evolutionary pressure by slowly increasing the minimum requirement for "Resilience". One can spread seeds with the mouse by pressing "S" and thereby start an inital population.

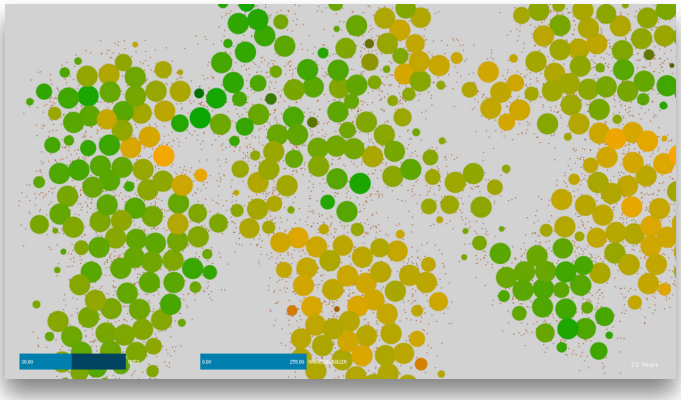
The program could be extended in many directions. The implemtentation of DNA is not limited. Therefore further genes or even sets of genes could be implemented with minimal effort. Easily one could introduce new attributes.

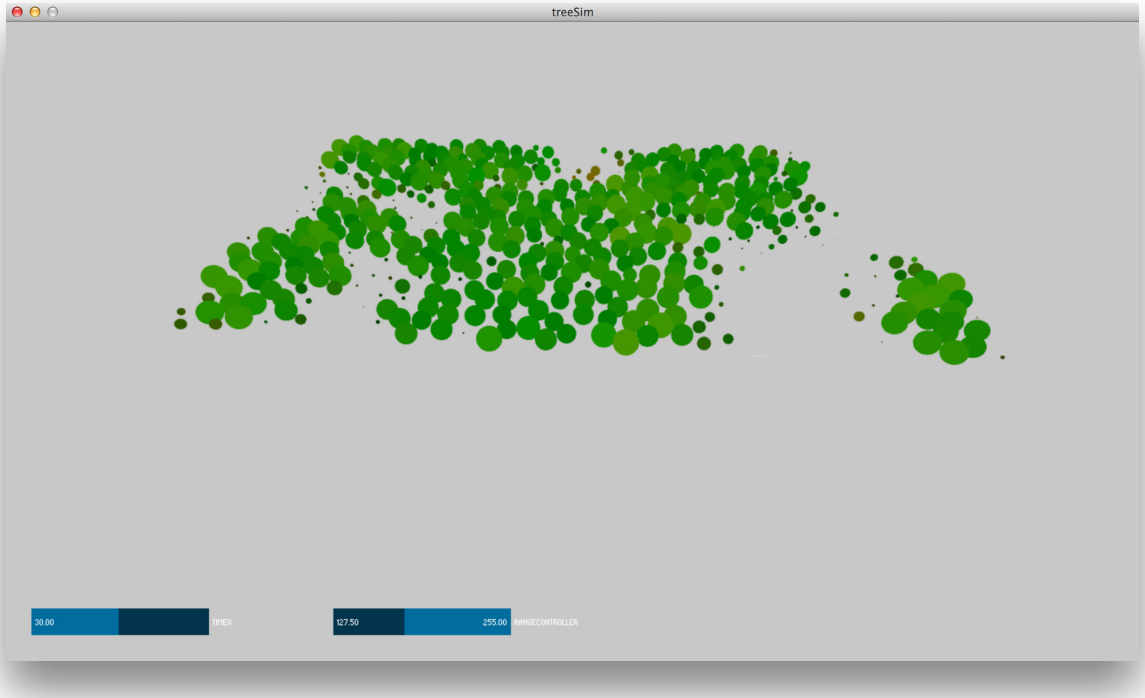
Trees, seeds and pollen are all extensions to the DNA class. The application does not distinguish between different "types", e.g. race or species in the DNA. This is intented since differentiation and compatibility should be expressed through the DNA itselff.

Graphical representation is currently seperated from the DNA. Mutating "form" and perhaps even mutating functionality could be further extensions. A first attemp on this was made trying to involve L-Systems growing in depth with age. Graphical performance was the initial limitation in this approach.








treeSim
30.00
127.50
255.00

05 L-SYSTEMS

When we moved to recursion, I was hinted at the very interesting topic of Lindenmayer systems.

> *"An L-system consists of an alphabet of symbols that can be used to make strings, a collection of production rules that expand each symbol into some larger string of symbols, an initial "axiom" string from which to begin construction, and a mechanism for translating the generated strings into geometric structures. L-systems were introduced and developed in 1968 by Aristid Lindenmayer, a Hungarian theoretical biologist and botanist at the University of Utrecht. Lindenmayer used L-systems to describe the behaviour of plant cells and to model the growth processes of plant development. L-systems have also been used to model the morphology of a variety of organisms and can be used to generate self-similar fractals such as iterated function systems."*[1]

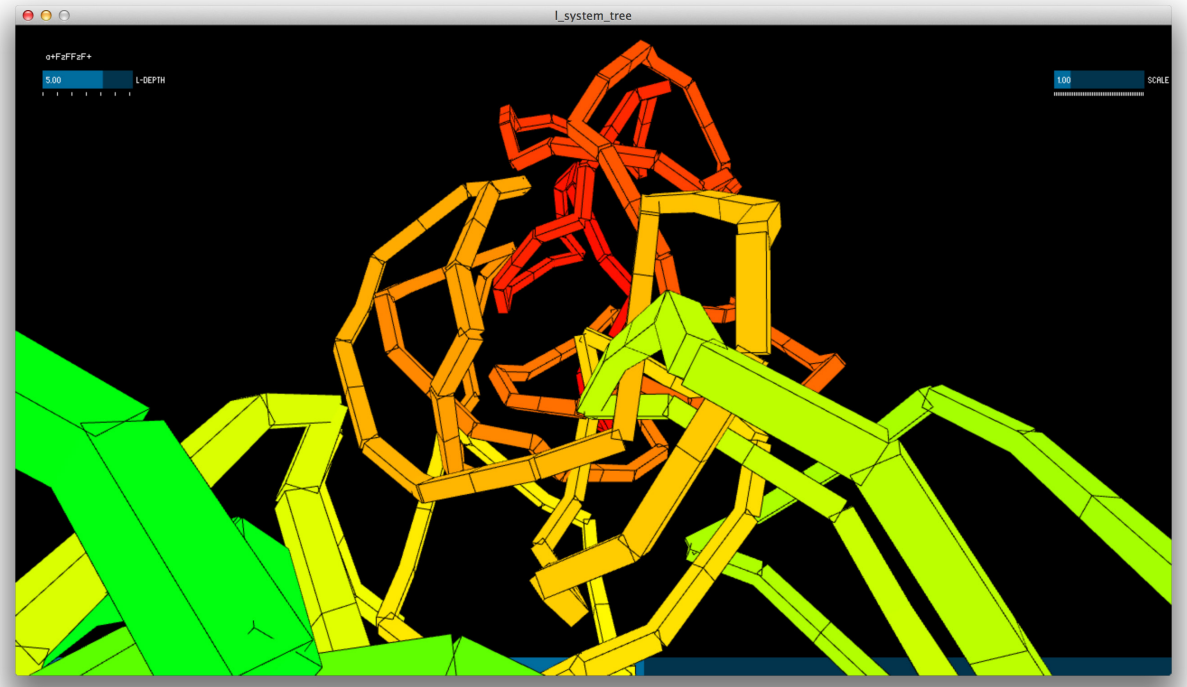
I was looking at several implementations in both pure Java and Processing as well, but none quite seemed to meet my requirements, so I went ahead rewrote the generation algorithm inspired by a very generic implementation[2] and then used parts of another[3] - though only selectively, since it's source code is written very poorly - to implement the graphical representation.
What I found especially intriguing about this second implementation[3] was the use of the HSB color mode to visualize to depth of a generated system.

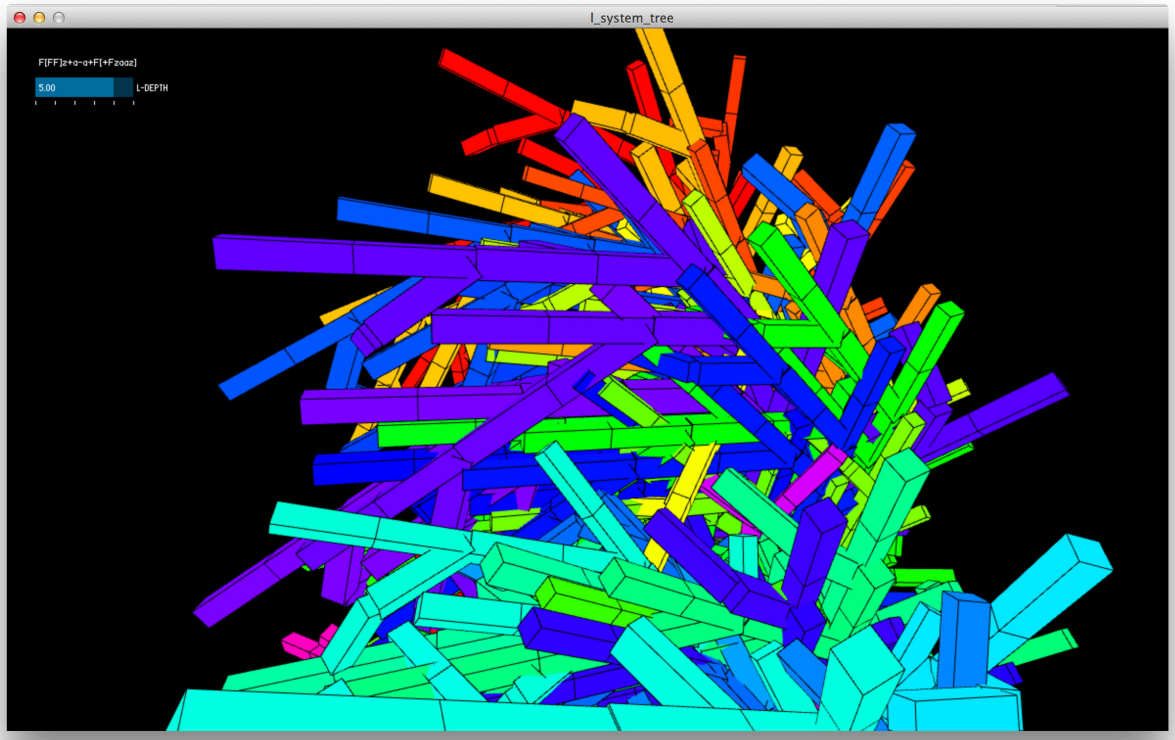
I then proceeded to add GUI elements which linked to certain parameters to give the user control over the representation of a rule or "tree". Configurable are depth and angle. Additionally the user can chose between a set of premade rules using the keys 1-9.

The result is generator for three dimensional L-Systems, which allows for exploration & play with this recursive concept.

---

[1] L-System - https://en.wikipedia.org/wiki/L-system
[1] http://www.javaview.de/vgp/tutor/lsystem/LSystem.java
[3] http://web.mit.edu/~eric_r/Public/lsystems/lsys3D12.pde


l_system_tree
a+FzFFzF+
5.00
L-DEPTH
1.00
SCALE


l_system_tree
F[FF]z+a-a+F[+Fzaaz]
5.00
L-DEPTH

06 QUADRA PONG

Our final task was to adjust, extend and modify a given example of a simple breakout game[1]. I quickly realized what limitations the provided code sample introduced and that my idea of a game was hardly doable with it. The decision was made to write "Quadra Pong" from scratch. For collision detection I reused the line intersection method from the original game.

The result is an easily extensible four player Pong game. Each player occupies a side of the playing area. In the normal mode two balls spawn and players have to defend their respective side and get the balls into the areas of the other players to score.

Since four players are not always at hand, I implemented AI Players which can be enabled or disabled within the controlConfig. For demonstration purposes all four AI players can be enabled and observed. A certain imperfection was neccessary for them to be fun to play against.

The aesthetics of the game were kept very reduced as a reference to the original Pong game.

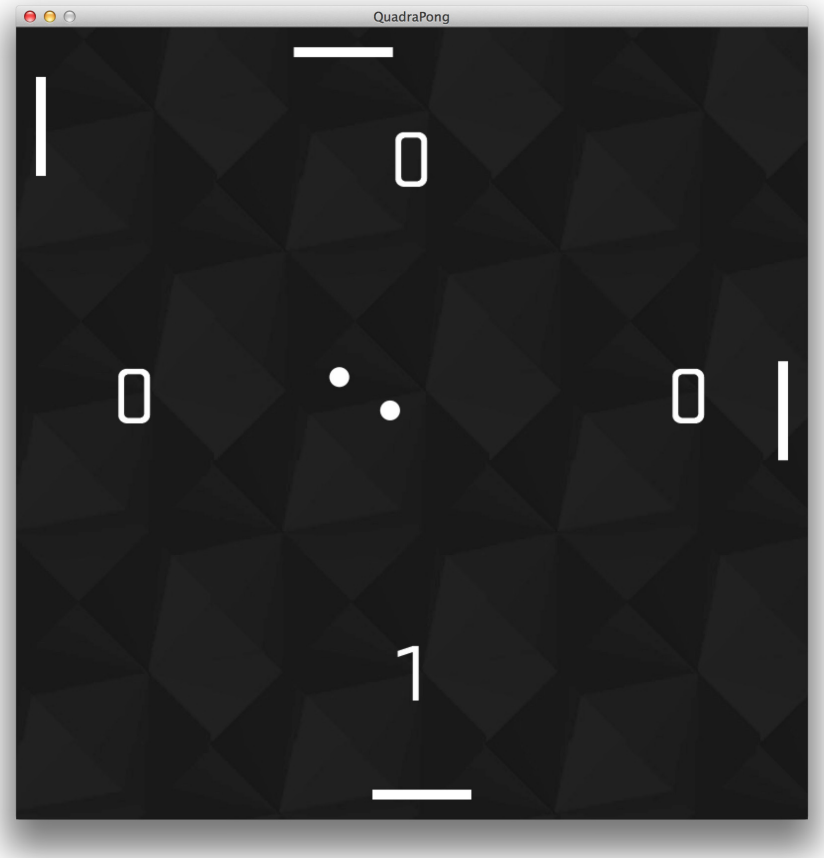
As an alternative option to play is using a Smartphone with an App for submitting OSC signals or similar device capable of TUIO. This allows for a player or multiple players to steer their plattform using touch controls.

The final goal would have been an integration of kinect controls. A top-down beamer and kinect camera could provide a playeable area on a table for example. Players could then control their plattform by simply hovering a hand over the projected image or closely next to it.

Interesting extenions could involve more than four players, or even teams and different approaches to controlling a players plattform.

---

[1] Breakout - http://blogs.iad.zhdk.ch/codingspace/endaufgabe/


QuadraPong
0
0
0
1


QuadraPong
QUADRA PONG
Press Space to Play

CONCLUSION

During the two weeks of the introductory course into programming I was able to explore new fields and experiment with interesting concepts, ideas and technologies.

Already knowing C# and Unity, I made my way into Java and Processing quite easily. I found it fascinating and motivating to dive into different aspects everyday and then bringing the expierence together in one final project.

All code samples mentioned can be found on
http://blogs.iad.zhdk.ch/codingspace/author/davidsimon/

David Simon