Processing

Felix Stricker

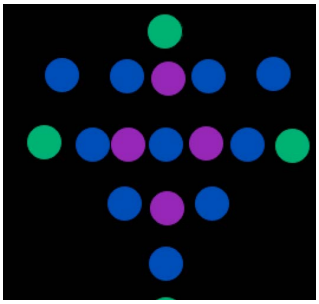
Dozent: Max Reihner
Modul: BDE-VIAD-V-1040 Programming Basics
18.09.13 - 27.09.13

Dokumentation

Dies ist die Dokumentation des Moduls Programming Basics, welches nach zwei Wochen abgeschlossen wurde.

Das Modul bestand aus täglichen Programmieraufgaben mit Processing, sowie eine grössere Endaufgabe. Die wichtigsten Aufgaben sind in dieser Dokumentation beschrieben.

19.09.13 – Zeichnungsprogramm



Für den Einstieg in das Programmieren verwenden wir Processing. Processing basiert auf Java. Da ich vor vielen Jahren in der Lehre das programmieren mit Java erlernt habe, war der Wiedereinstieg relativ einfach.

Zuerst mussten wir dem Mauscursor einen Punkt folgen lassen. Danach sollten wir ein Zeichnungsprogramm erstellen. Ich versuchte mein erstes Programm zu erweitern, merkte aber, dass das durch den draw()-loop nicht sehr einfach ist. Ich habe mich dann darauf beschränkt bei einem Klick einen Punkt zu zeichnen und diesen in einer ArrayList zu speichern. Bei jedem draw() wird diese ArrayList durchlaufen und die Punkte werden wiederhergestellt.

In einem neuen Programm erstellte ich ein zweites Zeichnungsprogramm mit folgenden Funktionen:

- Farbwahl: 1-6
- Vergrösserung des Pinsels (Kreis): Q
- Verkleinerung des Pinsels: (A)
- Darstellung einer Vorschau des aktuellen Pinsels
- Radieren mit der rechten Maustaste

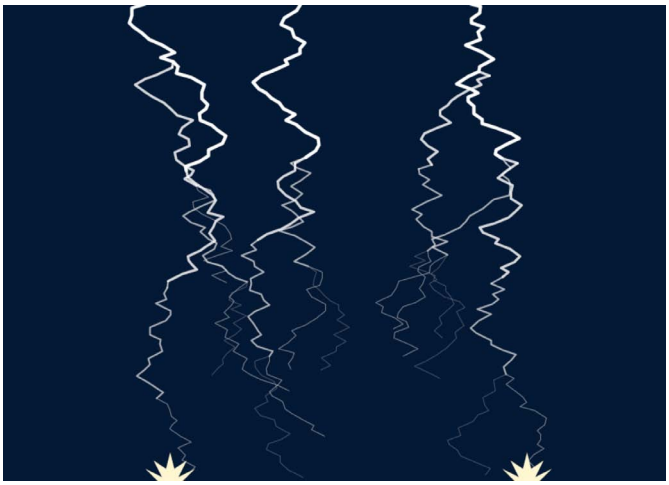
20.09.13 – Wald

Unsere Aufgabe war es einen zufälligen Wald zu zeichnen. Zuerst habe ich eine Funktion erstellt, mit welcher ein Baum gezeichnet wird. Damit der Wald etwas abwechslungsreich erscheint, habe ich zwei Baumvarianten definiert. Einen Apfelbaum und eine Tanne. Die grüne Farbe der Blätter wird per Zufall generiert.

Die Bäume werden mit den Funktionen pushMatrix() und popMatrix() ebenfalls zufällig verteilt. Dies hat den Vorteil, dass man einen Baum mit allen Elementen als Ganzes verschieben kann. Das Resultat sah nicht sehr realistisch aus, da die Baumstämme teilweise über den Baumkronen abgebildet wurden. Um dieses Problem zu beheben habe ich die Y-Position in Abhängigkeit von der Variablen i gestellt. Was dazu führt, dass der Wald von hinten nach vorne gezeichnet wird.

Um das ganze Programm zu erweitern, versuchte ich auf dem Horizont Windräder zu zeichnen und diese zu animieren. Dazu musste ich die noLoop() Funktion entfernen und den Wald im setup() generieren lassen. Der Horizont wird über draw() laufend neu gezeichnet. Das Rotieren der Windräder geschieht mit einer Matrix, welche rotiert wird.

23.09.13 – Gewitter

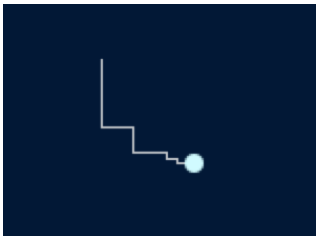


Der Mechanismus hinter den rekursiven Funktionen sollte uns anhand eines Programmierbeispiels verständlich gemacht werden. Bei diesem Beispiel wird ein Baum generiert. Wir erhielten den Auftrag, dieses Beispiel zu erweitern oder ein neues Programm mit rekursiven Funktionen zu erstellen.

Ich wollte etwas Neues machen und hatte zuerst die Idee, ein sich zufällig ausbreitendes Feuerwerk darzustellen. Diese Idee habe ich aber wieder verworfen, weil ich mich lieber mit dem Entstehen eines Gewitters mit Blitzen auseinander setzen wollte.

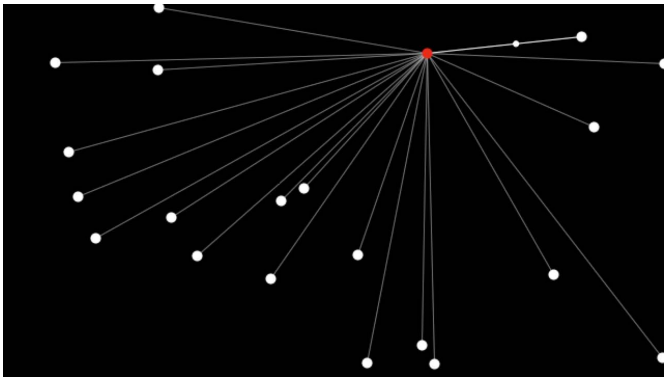
Mittels einer Klasse Thunderbolt kann ich einen Blitz generieren. Die Stärke eines Blitzes wird dabei zufällig gewählt. Damit der Blitz realistisch erscheint, habe ich eine weitere Klasse erstellt, welche eine Zufallslinie anhand eines Startpunktes und einer bestimmten Länge erstellt. Je nach Stärke des Blitzes, endet dieser an einer unterschiedlichen Stelle. Trifft der Blitz auf den Erdboden, erscheint dort eine simple Explosions-Zeichnung.

24.09.13 – Snake



Um zu lernen wie in Processing ein Zustand gespeichert und verändert werden kann, experimentierte ich selbständig anhand einer kleinen Übung. Mein Ziel war es, eine Linie, welche immer in Bewegung ist, mit der Tastatur manipulieren zu können, was der Grundfunktion des Snake-Spiels entspricht.

24.09.13 – PVector, PGraphics und Animation



Wir wurden beauftragt, in einer Animationsaufgabe einen Punkt einer Linie folgen zu lassen. Ich habe diese Aufgabe erweitert, um die Klassen PVector und PGraphics besser kennenzulernen und zu verstehen.

Auf einer Fläche werden weisse Punkte zufällig verteilt. Mit einem Klick wird ein roter Punkt bei der aktuellen Position des Mauscursors gezeichnet. Dieser Punkt kann mittels der Tastatur oder der Maus verschoben werden. Von allen weissen Punkten aus wird eine Linie zum roten Punkt gezogen. Beim weissen Punkt, welcher sich am nächsten beim roten Punkt befindet, wird die Linie stärker dargestellt sowie die Animation platziert.

25.09.13 - 27.09.13 – BreakOut

Für die Endaufgabe bekamen wir den Code mit den Grundfunktionalitäten des legendären Spiels BreakOut. Die Aufgabe bestand darin, dieses Spiel mit eigenen Ideen zu erweitern und zu verändern. Am Ende mussten wir der Klasse ein funktionierendes Programm präsentieren.

Ideen für Funktionalitäten

Ich begann die Aufgabe damit, mögliche Funktionalitäten für das Spiel zu sammeln:

- Scoreboard mit Punktestand, Zeit und Level
- Pause Modus
- mehrere Levels
- Ballgeschwindigkeit laufend erhöhen
- Bricks mit Aktionen belegen:
  Schläger vergrössern
  Ball verlangsamen
  mehrere Treffer notwendig

Diese Funktionalitäten habe ich in das Spiel integriert. Dazu habe ich einen End-Screen erstellt, welcher bei erfolgreichem Beenden des Spiels angezeigt wird.

Optische Darstellung

Nach dem Konzeptgespräch mit Max, suchte ich nach Ideen, wie ich dem Spiel optisch und inhaltlich einen Sinn geben kann. Ich kam auf das Thema Fussball. Mit dem Ball (=Fussball) und dem Racket (=Fuss) waren bereits zwei Elemente vorhanden waren, die ich übernehmen konnte. Dazu brauchte ich noch ein Tor. Die Bricks, welche durch die einzelnen Fussballer dargestellt werden, verschwinden, wenn sie mindestens dreimal mit dem Ball getroffen werden. Der Torwart gibt jedoch immer erst nach zehn Treffern auf.

Der Spieler kann nun Tore schiessen und die Spielfiguren durch mehrmaliges Treffen zum Aufgeben zwingen. Um das Spiel spannender zu machen, erhält die gegnerische Mannschaft einen Punkt, falls der Spieler den Ball mit dem Fuss verpasst. Damit das Spiel zu einem Ende kommt, wollte ich wie beim realen Fussball die Spielzeit begrenzen. Jedoch gefiel mir die Idee, anhand einer bestimmten Anzahl Tore den Gewinner zu bestimmen, besser.

Es stehen drei Levels zur Verfügung. Sie unterscheiden sich in der Anzahl Spielfiguren und deren Position. So wird die Spielschwierigkeit von Level zu Level erhöht.

Der Code ist zu finden unter:
http://blogs.iad.zhdk.ch/codingspace/2013/10/01/breakout-fussball