

Freie Universität Berlin, Institut für Informatik, Arbeitsgruppe Software Engineering

Christopher Oezbek, Mojgan Mohajer WS 2005

2005-10-13

Aufgabe 1-1: (Verständnis)

Geht die folgende Liste von Begriffen durch und überlegt, ob ihr bei jedem Begriff versteht, was er bedeutet. Wenn nicht, dann greift auf die Vorlesungsfolien und das Glossar zurück.

- Quellcode
- Maschinencode
- Compiler
- Programmiersprache
- Programmieren

Dieses Übungsblatt ist v.a. für alle diejenigen sehr schwer, die noch nie programmiert haben. Nicht verzagen und Fragen stellen!! Achtung: Begriffsverwirrungen!

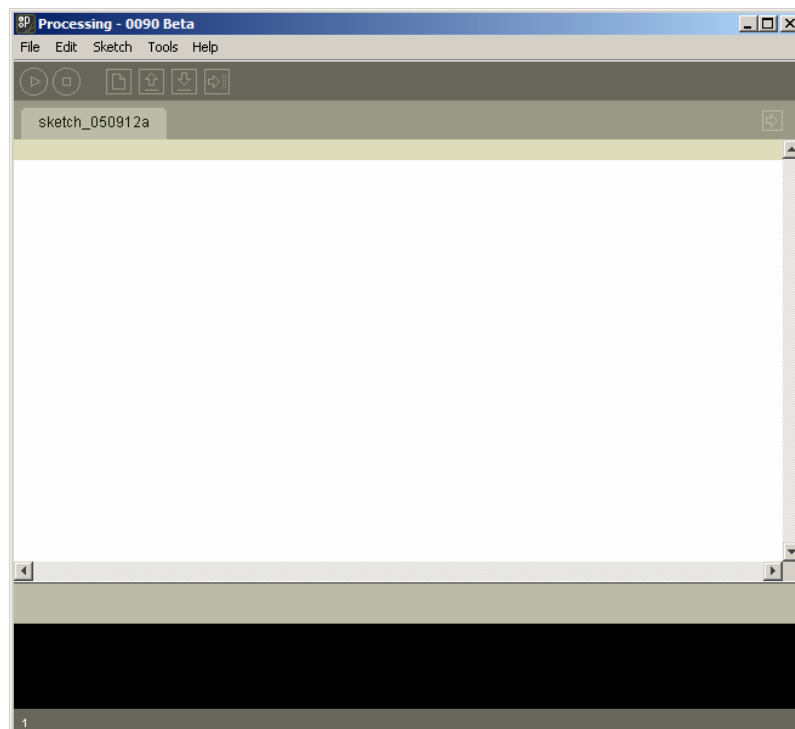
Aufgabe 2-1: (Processing installieren)

Um einen kurzen Einstieg in die Programmierung zu vollziehen, wollen wir die graphische Programmierumgebung "Processing" verwenden. Ihr findet hierzu eine Zip-Datei unter

<http://www.inf.fu-berlin.de/inst/ag-se/teaching/K-BKI-2005/install/processing.zip>

Tipp: Die Dateiendung zip weist euch bereits darauf hin, dass diese Datei ein ganzes Archiv von anderen Dateien ist. Entpackt die Datei in euer Heimatverzeichnis Z:\

In dem Verzeichnis findet ihr dann eine Datei Processing.exe. Wenn ihr diese ausführt und folgende Ansicht seht, dann war die Installation erfolgreich:

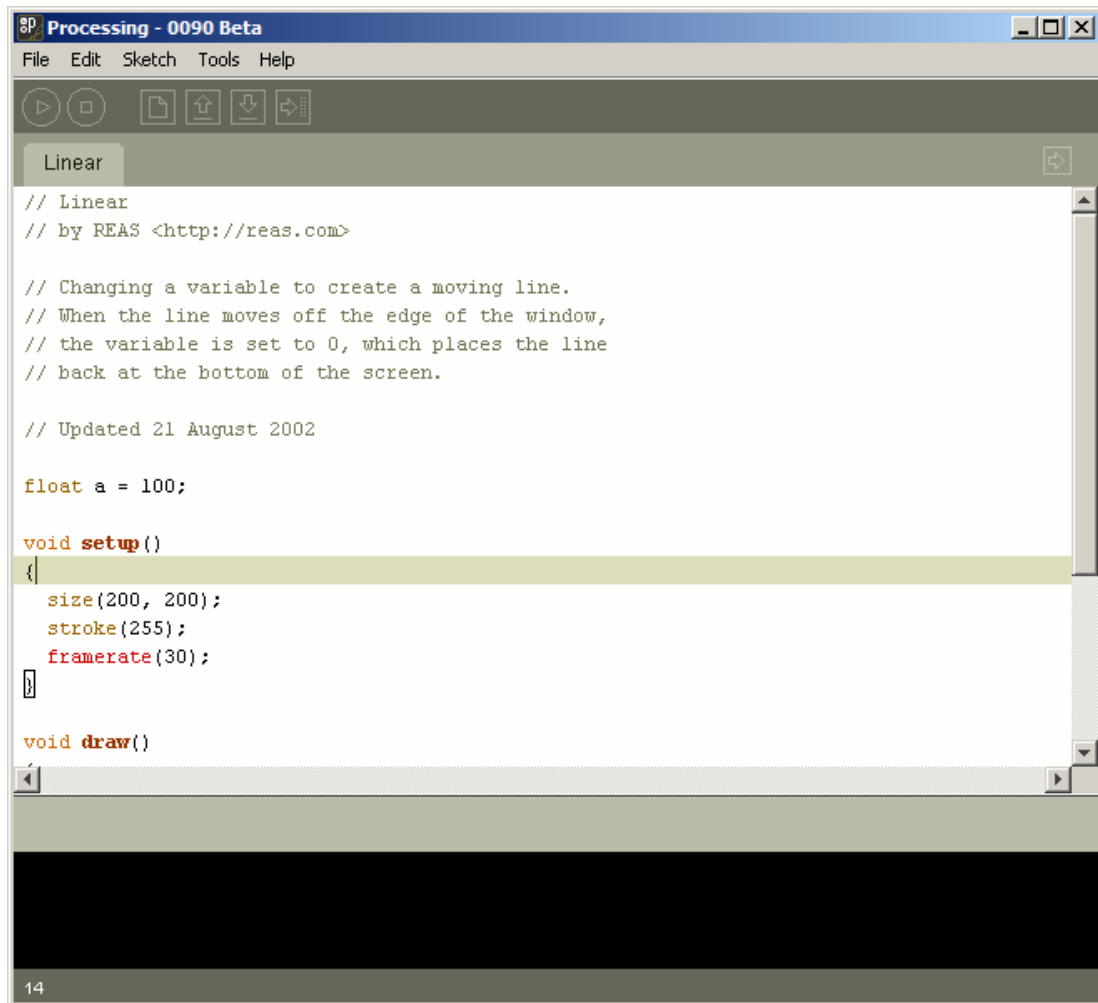


Aufgabe 2-2: (Processing entdecken)

Processing ist eine Programmierumgebung für Künstler. Dies bedeutet, dass ihr in diesem Programm selber Programme schreiben und ausführen könnt, bei denen es sehr einfach ist, graphische Effekte zu erzeugen.

Das Programm hat 3 Bereiche: Die graue Befehlsleiste oben, mit einem Start- und Stop-Knopf; die weiße Fläche zur Eingabe von Programmen durch euch und den schwarzen Bereich unten. In diesem werden euch Nachrichten von Processing ausgegeben.

Zuerst wollen wir jetzt ein Beispiel laden und ausführen. Klickt dazu auf File -> Sketchbook -> Examples -> Motion -> Linear. Processing lädt jetzt ein Beispiel in den weißen Bereich:

The image shows a screenshot of the Processing IDE window titled "Processing - 0090 Beta". The window has a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for play, stop, save, and other functions. The main area is a text editor showing the code for a sketch named "Linear". The code includes comments, variable declarations, and function definitions for setup and draw. The draw function contains a line drawing command. The IDE has a dark grey background and a white text area.

```
Processing - 0090 Beta
File Edit Sketch Tools Help

Linear

// Linear
// by REAS <http://reas.com>

// Changing a variable to create a moving line.
// When the line moves off the edge of the window,
// the variable is set to 0, which places the line
// back at the bottom of the screen.

// Updated 21 August 2002

float a = 100;

void setup()
{
  size(200, 200);
  stroke(255);
  framerate(30);
}

void draw()
```

Wenn ihr jetzt den Play-Knopf drückt, wird der Text in dem weißen Kasten in ein Programm übersetzt und dann ausgeführt. Es sollte sich ein kleines Fenster öffnen und ein weißer Balken über den Bildschirm fliegen.

Seht euch jetzt mal den Quellcode für dieses Programm an (ich habe ein paar Hinweise dran geschrieben):

```
// Linear...

float a = 100; // Eine Variable um uns zu merken, wo die Linie ist

void setup() // Mach das einmal am Anfang
{
  size(200, 200); // kleines Fenster mit 200 mal 200 Punkten bitte
  stroke(255); // Bitte den Stift auf weiße Farbe setzen
  framerate(30); // Zeichne 30 Bilder die Sekunde
}

void draw() // Wird 30 mal pro Sekunde gemacht
{
  background(51); // Hintergrund auf eine dunkle Farbe setzen
  a = a - 1; // Die Linie einen Punkt nach oben verschieben
  if (a < 0) { // sind wir ganz oben
    a = height; // dann die Linie ganz nach unten setzen
  }
  line(0, a, width, a); // Linie zeichenn
}
```

Ihr werdet auf diesem Übungsblatt nicht genug Zeit haben, um Programmieren systematisch zu lernen. Vielleicht könnt ihr aber durch die Beispiele schon ein bisschen was lernen.

Hier ein paar Hinweise:

- Alles, was nach zwei Schrägstrichen steht, wird vom Computer ignoriert. Man nennt dies einen **Kommentar**. So kann man also einzelne Zeilen ausschalten. Ausprobieren: Stellt // vor die Zeile "a = a - 1;" und der Strich bewegt sich nicht mehr.
- Die drei Zeilen, die in den Klammern zu setup() stehen, werden einmal am Anfang ausgeführt und dann wird 30mal die Sekunde draw ausgeführt, d.h. alle Befehle, die dort in der Klammer stehen.
- Nach jedem Befehl für den Computer muss ein Semikolon stehen, sonst kapiert der Computer das nicht.
- Farben könnt ihr von 0 bis 255 setzen. Null ist schwarz und 255 ist Weiß.
- Wenn ihr beim Starten Fehler bekommt, dann ladet einfach das Beispiel neu oder macht eure Änderungen rückgängig.

Hier ein paar Ideen, die ihr versuchen könnt:

Ändert ein paar von den Zahlen z.B. a = a - 10;

Macht das Fenster größer mit size.

Wenn ihr nur noch Bahnhof versteht, dann lest euch mal die folgende Internet-Seite zusammen durch:

http://processing.org/learning/tutorials/structure_1/index.html

!!!!!!!!!!!!!!!!!!!!!!!!!!WECHSELN!!!!!!!!!!!!!!!!!!!!!!!!!!!!

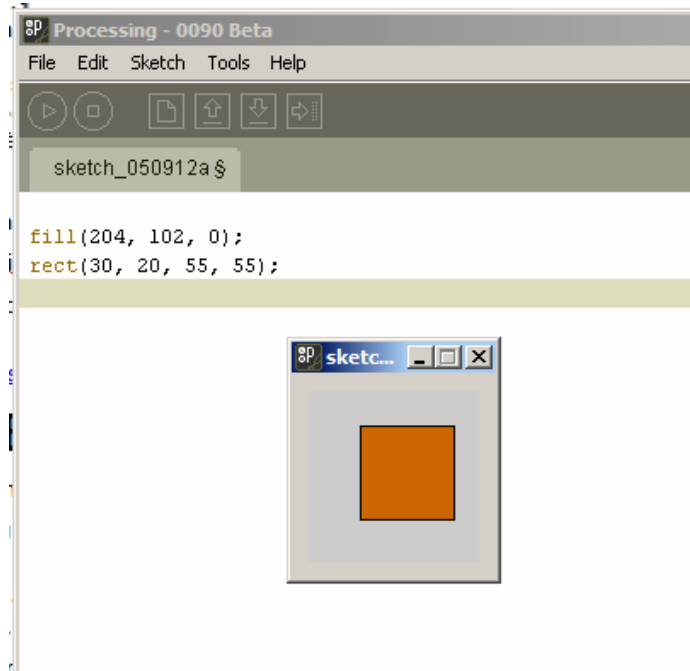
Aufgabe 2-3: (Processing-Befehle ansehen)

Klickt jetzt auf File -> New und besorgt euch damit eine neue leere Seite. Öffnet dann die Hilfe mittels Help -> Reference.

Hier seht ihr jetzt fast alle Befehle, die Processing kennt. Wenn ihr auf einen Eintrag klickt, bekommt ihr den Befehl erklärt. Jeweils mit einem Beispiel.

Aufgabe: Seht euch 10 Befehle an, deren Namen euch interessieren und kopiert den Beispielquellcode nach Processing und führt das Programm dann aus. Spiel mit jedem dieser Befehle etwas herum, indem ihr Zeilen kopiert und verändert.

Als Beispiel der Befehl Fill:



Aufgabe 2-3: (Ein bisschen Inspiration einholen)

Besucht die Webseite der Berliner Universität der Künste und seht euch an, was die Kunststudenten programmiert haben und natürlich auch immer den Sourcecode:

<http://workshop.evolutionzone.com/exercises.html>

Alternativ:

http://processing.org/exhibition/network_page_2.html

Aufgabe 3-1: (Selber aktiv werden)

Nun ist es an euch. Versucht, selbst etwas mit Processing zu bauen. Wechselt euch immer wieder ab und baut eure eigenes Processing-Programm. Schickt den Source-Code eures besten selbst geschriebenen Programms mit folgendem Titel an oezbek@inf.fu-berlin.de

Übung 4 - Name 1, Name 2 - Titel eures Programms.

Wenn ihr das Gefühl habt, nicht genug über Programmierung zu wissen, um kompliziertere Sachen zu machen, dann könnt ihr zuerst Aufgabe 4 machen.

Aufgabe 4-1: (Programmierung Essentials)

Noch einmal die wichtigsten Fakten, die ihr schon kennt:

- Programme bestehen aus Befehlen oder Anweisungen für den Computer.
- Jeder Befehl muss mit einem Semikolon beendet werden.

Variablen:

Eine Variable bei der Programmierung ist ähnlich einer Variable in der Mathematik. Sie hat einen Wert. Anders als in der Mathematik kann man diesen Wert aber im Laufe des Programms ändern.

Zuerst muss man eine Variable allerdings deklarieren, sonst weiß das Programm nicht, dass sie da ist:

```
int x; // Jetzt kennt der Computer eine Variable  
      // mit Namen X, die ganze Zahlen aufnehmen kann.
```

Jetzt kann man dieser Variable einen Wert geben:

```
x = 2;
println(x);
```

Und man erhält 2 im schwarzen Kasten (ausprobieren).

Nachdem eine Variable einen Wert hat, kann man sie auch für Berechnungen verwenden:

```
int y;
y = 4 * x + 3;
println(y); // y = 11
```

Entscheidungen:

Man kommt schnell an den Punkt, wo man entweder eine oder eine andere Sache machen will. Hierzu benötigt man die IF-Anweisung.

```
int a = 23;
int b = 23;
if(a == b) {
    println("A und B sind gleich");
} else {
    println("A und B sind nicht gleich");
}
```

Ihr könnt auch mehrere Entscheidungen verknüpfen, z.B.

```
if (a == b && a * b = 49)
```

&& ist UND, || ist ODER, != ist UNGLEICH

Sachen mehrmals tun:

Hierfür gibt es Schleifen, die so funktionieren:

```
for (int i = 0; i < 10; i++){
    println(i); // Schreibt die Zahlen 0 bis 10
}
```

Funktionen:

Manchmal will man mehrere Befehle zu einer Gruppe zusammenfassen. Dies geht mit Funktionen:

```
void hallo(){ // hier wird die Funktion nur definiert
    println("Hallo");
    println("Wie geht es Dir?");
}
hallo(); // führt die beiden println-Befehle aus
```

Man ruft eine Funktion auf, indem man ihren Namen mit Klammern schreibt und definiert sie, indem man void davor setzt (es gibt noch mehr Möglichkeiten).