Aufgabe 7

Ralph Müller und Dennis Blöte, 18.12.2006

Interaktive Programmierung: Das Programm MouseFunctions

Das Programm „MouseFunction“ stellt ein mittlerweile typisches und bekanntes Verfahren dar - das sogenannte „drag and drop“. Dies ermöglicht es Objekte mit Hilfe des Mauszeigers von ihrer Ursprungsposition wegzubewegen und an einer Neuen abzulegen.

Die Funktion setup(), die in Processing die Grundeinstellungen für das Programm vornimmt (Festlegen der Farben, Größe der Bühne, …), legt zuerst die Arbeitsfläche fest (size(200, 200);). Die Variablen bx und by, welche im Quellcode vorher als Fließkkommazahl deklariert wurden, werden nun initialisiert. Sie bekommen jeweils die Hälfte der Arbeitsflächenbreite und -höhe zugewiesen. Dabei sind "width" und "height" reservierte Wörter in Processing, die die Arbeitsfläche numerisch angeben.
Anschließend wird der Zeichenmode der Funktion rectangle() gesetzt. Die Übergabe der Konstante "CENTER_RADUIS" als Parameter, lässt die Funktion rectangle() später ein Rechteck um einen Mittelpunkt zeichnen.

Die Funktion draw() ist für den Ablauf des Programms zuständig. Sie füllt in jedem neuen Frame (Zustand) die Bühne zunächst komplett mit schwarz und zeichnet anschließend die neu positionierte Box wieder auf die Bühne.
Dies geschieht in mehreren Schritten: Zuerst wird gecheckt, ob sich der Cursor über der Box befindet. Ist dies der Fall und die Maustaste ist nicht gedrückt, so wir die Box mit einem weissen Rahmen versehen. Dies geschieht, um dem Nutzer zu signalisieren, dass er mit der Box interagieren kann.
Wenn sich der Mauszeiger über der Box befindet, wird zusätzlich die Zustandsvariable bover auf true gesetzt. Dadurch weiß die Funktion mousePressed(), ob der Kasten „gegriffen“ werden kann - dazu später mehr.
Sollte sich der Mauszeiger nicht über der Box befinden, wird die Zustandsvariable bover auf false gesetzt und der weisse Rahmen um die Box entfernt.
Am Ende von draw() wird wie anfangs erwähnt die Box neu auf die Bühne gezeichnet.

Die Funktion mousePressed() wird aufgerufen, sobald der Benutzer mit der Maus auf irgendeinen Punkt der Bühne klickt. Da im vorherigen Schritt mittels der Funktion draw() die Zustandsvariable bover gesetzt wurde, weiss die Funktion mousePressed(), ob der Cursor über der Box ist. Wenn ja, wird die Box „gegriffen“ und weiss gefüllt. Des Weiteren wird die Zustandsvariable locked auf true gesetzt, was bedeutet, dass die Box zur Zeit vom Benutzer gezogen wird.
Am Ende der Funktion wird die Differenz aus der Position des Mauszeigers und der eigentlichen Boxposition berechnet. Dies ist notwendig, damit in der folgenden Funktion, mouseDragged(), die neue Boxposition gesetzt werden kann.

In der Funktion mouseDragged() wird wie eben angesprochen die Position der Box neu berechnet, falls der Zustand locked auf true gesetzt ist. Die Variablen bx und by geben die neue Position der Box an (welche erst im nächsten draw() entsprechend auf die Bühne gerendert wird) und berechnen sich aus den in der Funktion mousePressed() berechneten Dirrerenzwerten des Mauszeigers zur Box.

Die Funktion mouseReleased() wird aufgerufen, sobald der Benutzer die Maus loslässt. Sie setzt die Zustandsvariable locked auf false, so dass die anderen Funktionen wissen, dass die Box nicht mehr gezogen wird.

Interaktive Programmierung

Da wir das Programm schon im Quelltext dokumentiert haben (siehe Anhang), beziehen wir uns hier lediglich auf die Schritte der Bearbeitung der Aufgabe.

Mir hat die Bearbeitung der Aufgabe viel Spaß gemacht und ich denke, dass ich einiges neues über Processing gelernt habe: Arbeiten mit Farben, reagieren auf Benutzerinteraktion und auch den Einsatz von Klassen.
Mir erschien es sinnvoll, die einzelnen Felder als Objekte zu repräsentieren, da ma in ihnen so ihre Zustände speichern kann: Ist das Feld momentan angeklickt oder nicht? Welchen Farbton, welche Größe und Position hat es.
Zunächst haben wir das Programm mit Arrays zum Speichern der Zustände geschrieben. Das war aber eher unschön und weniger flexibel als die jetzige objektorientierte Lösung.
In unserer jetzigen Lösung lässt sich das Programm durch ein paar Parameter beeinflußen, so dass man durch wenige Änderungen sehr vielfältige Ergebnisse erzielen kann. Beispielsweise ließe sich die Größe der Felder oder der Bühne anpassen oder man könnte die Abfolge der Farben beeinflußen.

Mit der Programmierung hatten wir eigentlich keine Schwierigkeiten. Da ich vorher schon viele Erfahrung mit dem Programm Flash gemacht habe, kannte ich einige der Umsetzungsmuster schon aus dem Bereich der Webprogrammierung: Buttons highlighten, sobald die Maus über ihnen ist oder sie auf aktiv setzen, nachdem sie geklickt wurden. So konnte man sich darauf konzentrieren, zu lernen, wie man diese Dinge in Processing umsetzt.

Zunächst hatten wir mit Grauwerten gearbeitet. Da wir aber noch Zeit hatten und das auf Dauer ja auch irgendwann langweilig wird, haben wir uns dazu entschieden, das Programm so umzuschreiben, das unsere Felder farbig sind. Dabei dachten wir zuerst daran, es mit RGB umzusetzen, allerdings liegt bei der Aufgabenstellung ja nahe, es mit HSB zu machen, weil man so die Sättigung mit einem Parameter beeinflußen kann.

Zusammenfassend könnte man sagen, dass das eine Aufgabe war, die einen etwas gefordert und dadurch gefördert hat. Wir haben viel neues zur Arbeit mit Processing gelernt :)