Einfach ein Spiel Machen

15. Oktober 2014

Teil I
Einleitung

In diesem Workshop wirst Du Schritt für Schritt ein kleines Computerspiel erschaffen. Dazu wirst Du Lua-Programmicode abtippen und ergänzen. Dieser wird dann von der LÖVE Spiele-Engine als Programm ausgeführt.

In LÖVE gibt es drei *Funktionsblöcke*, welche die Grundstruktur jedes LÖVE-Spiels bilden:

- In `love.load()` werden Befehle einmalig zur Vorbereitung ausgeführt. Zum Beispiel sagen wir hier, welche Bilddateien für den Hund und für die Katze benutzt werden und wo sie am Anfang des Spiels auf dem Gras stehen.
- In `love.update()` werden Änderungen im Spiel verarbeitet. Beispiel: „Die linke Pfeiltaste ist im gedrückten zustand. An welcher stelle befindet sich der Hund aktuell?“ Im idealfall wird der Funktionsblock `love.update()` 60 mal je Sekunde berechnet, also mit einem Zeitabstand von $0,01\overline{6}$ Sekunden.
- In `love.draw()` werden Malbefehle verarbeitet. Wir werden Gras, einen Hund und eine Katze malen und etwas Text und Zahlen, um zu zeigen, wie oft der Hund die Katze schon angebellt hat.

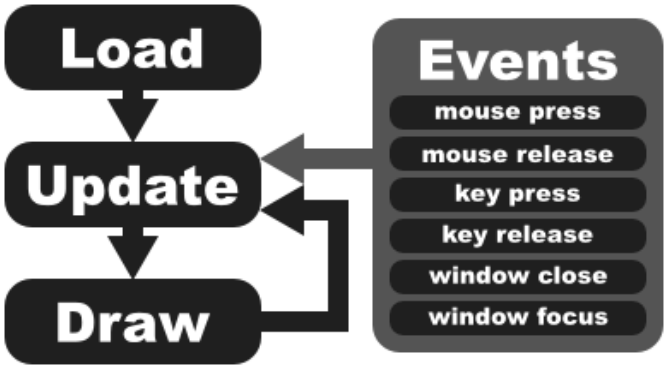

Abbildung 0.1: Grundaufbau eines LÖVE-Spiels

So sieht das ganze als Programmcode aus:

```
function love.load()
end
function love.update()
end
function love.draw()
end
```

Teil II
Das Spiel

In Deinem Spiel bewegst Du einen Hund auf einer Wiese und verjagst herumlaufende Katzen mit lautem Bellen.

1 Die richtige Datei

Starte den Texteditor NOTEPAD++ und öffne die Datei „main.lua“ im Verzeichnis „Spiel“.

Wir werden **Doppelpunkte**, **Anführungszeichen**, **Mathematische Symbole** und **Klammern** tippen und die Tasten **Alt**, **Strg**, **F4** und **F6** drücken. Drücke mal **F6**. Was passiert?

2 Malen

Als erstes soll das Spiel einen Hund malen.

2.1 Hund

Tippe folgenden Programmcode ab:

```
function love.load()
  hundBild = love.graphics.newImage("hund.png")
end

function love.update()
end

function love.draw()
  -- Male den Hund am punkt x=300, y=100
  love.graphics.draw(hundBild, 300, 100)
end
```

Achte auf Groß- und Kleinschreibung, sonst funktioniert das Spiel nicht.

Speichere nun die Datei (z.B. durch Druck der Tastenkombination **Strg**+**S**) und Drücke **F6**. Wir haben NOTEPAD++ so konfiguriert, dass **F6** das Spiel mithilfe von LÖVE ausführt. Diesen Schritt (Speichern und Ausführen) musst Du jedes Mal wiederholen, wenn Du Dein Spiel testen möchtest.

Es sollte ein Programmfenster mit Schwarzer Hintergrundfarbe und einem kleinen Hundegesicht erscheinen.

Um den Test zu beenden, kannst Du **Alt+F4** drücken oder das Fenster mit der Maus schließen.

2.2 Gras

Damit der Hund nicht im Dunkeln sitzt, soll das Spiel Gras malen. Füge folgenden Code als neue Zeile in den `love.load()` Funktionsblock ein:

```
grasBild = love.graphics.newImage("gras.png")
```

Der folgende Code gehört als neue Zeile in den `love.graphics()`:

```
-- Zeichnet größeres Bild von Gras
love.graphics.draw(grasBild, 0, 0)
```

Nach dem Speichern und Ausführen sollte nun eine Wiese zu sehen sein. Je nachdem, ob Du den Hund oder das Gras an erster Stelle im `love.draw()` Funktionsblock stehen hast, ist der Hund zu sehen oder nicht. In letzterem Fall tausche die zwei `love.graphics.draw()` Zeilen miteinander aus.

2.3 Erklärung

In der Aufgabe 2.1 sorgen wir mithilfe der *Funktion* `love.graphics.newImage()` dafür, dass LÖVE die Bilddatei `hund.png` findet und das wir auf dieses Bild mit dem Wort `hundBild` später zugreifen können. Wörter, denen wir mithilfe des Gleichheitszeichens eine Bedeutung zuweisen, heißen *Variablen*.

Die Funktion `love.graphics.draw()` zeichnet `hundBild`, welches auf die Bilddatei `hund.png` verweist. Der Hund erscheint 300 *Pixel* (Bildpunkte) vom linken Rand und 100 Pixel vom oberen Rand entfernt.

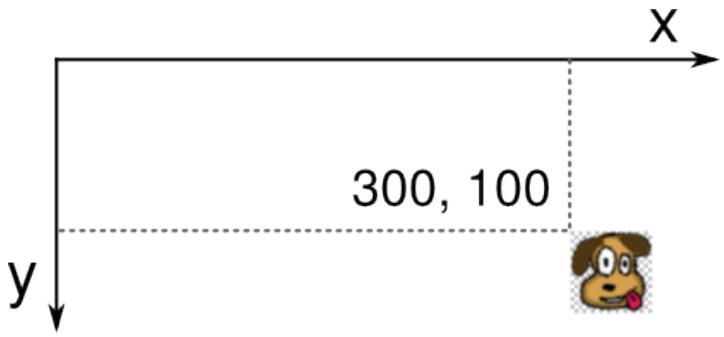

Abbildung 2.1: Hund im LÖVE-Koordinatensystem

In 2.2 wiederholen wir das ganze mit einer anderen Bilddatei und der linken oberen Ecke – 0 Pixel vom linken und 0 Pixel vom oberen Rand entfernt.

Zwei Minuszeichen bedeuten, dass der Text danach ein Kommentar und nicht Programmcode ist. Das ist nützlich für Notizen und Erinnerungen. Ihr braucht die Kommentare nicht abtippen, könnt aber gerne welche für euch selbst schreiben.

3 Bewegen

Jetzt soll der Hund bewegt werden.

3.1 Position

LÖVE muss die Position des Hundes wissen. Tippe folgendes in `love.load()`:

```
hundX = 300
hundY = 100
```

Wir haben Variablen für die X- und Y-Wert des Punktes erstellt, an dem sich der Hund befindet.

3.2 Tastenerkennung und Bewegung

Diese Position Jetzt prüfen wir, ob Pfeiltasten gedrückt werden und wenn ja, wird die Position des Hundes geändert. Füge folgenden `if`-Abfrageblock in `love.update()` ein:

```
if love.keyboard.isDown("left") then
  hundX = hundX - 5
end
```

`love.keyboard.isDown()` ist eine Funktion, welche den Namen einer Taste als *Parameter* (Inhalt der Klammern) erhält und dann `true` oder `false` ausgibt, abhängig davon, ob die Taste gedrückt wird oder nicht. Der Code im `if`-Abfrageblock, wird ausgeführt, wenn die `if`-Abfrage `true` ergibt.

Das Spiel reagiert aber noch nicht auf die Taste, `love.graphics.draw(hundBild, 300, 100)` verwendet ja immer die Parameter `300` und `100` für die Position des Hundes. Ersetze die Zahlen mit den Variablen, die wir erstellt haben.

Kopiere den `if`-Abfrageblock drei mal und passe die Kopien an, sodass die Tasten „up“, „right“ und „down“ die Position des Hundes – beziehungsweise die Variablen `hundX` und `hundY` – ebenfalls ändern.

4 Katze

Eine Katze will unseren Hund nerven, indem sie über den Bildschirm läuft und verschwindet, wenn sie den Rand erreicht, um an einer neuen Stelle aufzutauchen.

4.1 Position, Richtung und Malen

Genau wie der Hund, muss LÖVE das Katzenbild kennen und wissen, wo die Katze sitzt.

```
katzeBild = love.graphics.newImage("katze.png")
katzeX = 0
katzeY = 400
```

In `love.draw()` schreibe folgendes, damit die Katze auch zu sehen ist:

```
love.graphics.draw(katzeBild, katzeX, katzeY)
```

4.2 Bewegung

Damit sich die Position der Katze andauernd ändert, schreibe folgendes in `love.update()`:

```
katzeX = katzeX + 7
```

Nun läuft sie immer nach rechts.

4.3 Rand

Was wenn die Katze nicht mehr im 800 Pixel breiten Bild zu sehen ist? Dann soll sie vom linken Rand vom neuen anfangen zu laufen.

```
-- "a > b" ergibt true, wenn a größer als b ist
if katzeX > 800 then
  -- 0 auf der X-Achse ist ganz links
  katzeX = 0
  -- Position auf Y-Achse soll zufällig sein
  katzeY = math.random(0, 550)
end
```

In `love.update()` prüfen wir stur die Horizontale Position der Katze auf der X-Achse mithilfe einer `if`-*Abfrage* („if“ bedeutet „wenn“) und dem Schlüsselwort `or` („oder“) sowie `then` („dann“), welches die Abfrage beendet. Der Code im `if`-Block wird nur ausgeführt, wenn die *Bedingung* in der Abfrage `true` („wahr“) lautet.

4.4 Bellen und Verscheuchen

Wenn die Katze dem Hund zu nahe kommt, bellt dieser und die Katze verzieht sich um so schneller... Schreibe folgendes in `love.load()`:

```
bellen = love.audio.newSource("bellen.ogg")
```

Und in `love.update()`:

```
xFern = hundX - katzeX
yFern = hundY - katzeY
abstand = math.sqrt(xFern^2 + yFern^2)
if abstand < 40 then
  bellen:play()
  katzeX = 999
end
```

Nach dem Bellen tun wir so, als wäre die Katze plötzlich ganz weit nach rechts gesprungen (999 auf der X-Achse). Beim nächsten Durchlauf von `love.update()` (meist 0,0167 Sekunden später) wirkt das genau so als wäre die Katze wie in **4.3** von selbst aus dem Bild gelaufen.

4.5 Erklärung

Die Berechnung von `abstand` erfolgt mithilfe des Euklidischen Abstands ($c = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$), welcher auf dem Satz des Pythagoras basiert ($a^2 + b^2 = c^2$).

5 Gewinnen und Zeit

Nach einer Weile hat die Katze genug vom Spiel und es wird der Punktestand angezeigt.

5.1 Timer, Game Over und Textanzeige

Schreibe diesen Code in `love.load()`:

```
zeitStart = love.timer.getTime()
zeitJetzt = zeitStart
spielVorbei = false
-- wie oft hat der Hund die Katze angebellt
punkte = 0
```

Der folgende Code kommt in `love.update()`:

```
zeitJetzt = math.floor(love.timer.getTime())
if 30 + zeitStart - zeitJetzt < 0 then
  spielVorbei = true
end
```

Dieser Code soll ausgeführt werden, nachdem der Hund bellt:

```
  if not spielVorbei then
    punkte = punkte + 1
  end
```

Und der folgende gehört in `love.draw()`:

```
if spielVorbei then
  nachricht = "Spielende, Punkte: " .. punkte
else
  nachricht = "Punkte: " .. punkte
end
love.graphics.print(nachricht, 300, 20)
```

5.2 Erklärung

Zu beginn des Spiels merken wir uns eine Zeit und vergleichen sie mit der aktuellen Zeit während dem Spiel.

Sobald 30 Sekunden vorbei sind, verhindert die Variable `spielVorbei`, dass Punkte weiter gezählt werden. Auch wird abhängig von `spielVorbei` ein anderer Text angezeigt.

6 Bonusaufgaben

1. Macht das Spiel mehr Spaß wenn der Hund und die Katze schneller sind?
2. Kannst Du einen zweiten Hund ins Spiel bringen, der von den Tasten „w“, „a“, „s“ und „d“ gesteuert wird?
3. Es gibt die Datei `miauen.ogg`. Klingt es besser als das Bellen?
4. Kannst Du dafür sorgen, dass der Hund nicht mehr aus dem Bild laufen kann?
5. Wie würdest Du die verbliebene Zeit während dem Spiel anzeigen?
6. Kannst Du mehr als eine Katze herumlaufen lassen?

Teil III
Mehr erfahren

Du kannst LÖVE von http://love2d.org herunterladen und auf http://love2d.org/wiki/ detaillierte Information zur LÖVE-Programmierung finden. Diese ist größtenteils Englischsprachig, Am Unteren Rand der Webseite kann man auf Deutsch umschalten.

Du kannst das Material für diesen Workshop von http://github.com/qubodup/spiel-machen herunterladen