

Einstieg Spiele-Programmierung in LÖVE

© 2017 Iwan Gabovitch (espws.de)

Lizenziert unter einer Attribution-ShareAlike 4.0 International Lizenz



1 Vorbereitung

1. Extrahiere **StartGamedev** und öffne den Texteditor mithilfe der **open-editor** Datei.
2. Lese Aufgaben aufmerksam, tippe Code (*Quelltext*) ab und teste Ergebnisse.

2 Miau Spiele-App

2.1 Interaktiver Sound

Tippe den folgenden Code **ab**, speichere und teste diesen:

```
1 function love.load()  
2     sound = love.audio.newSource( "meow1.ogg" )  
3 end  
4  
5 function love.mousepressed()  
6     sound:play()  
7 end
```

Der Code in `love.load()` lädt eine Sounddatei und `love.mousepressed()` spielt diese ab, wenn ein Mausknopf gedrückt oder der Touchscreen berührt wird.

2.2 Interaktive Bilder

Ergänze `love.load()` mit dem Laden zweier Bilder:

```
1 bild_auf = love.graphics.newImage( "open.png" )  
2 bild_zu = love.graphics.newImage( "closed.png" )
```

Füge folgende zwei Funktionen zu Deinem Code **hinzu**:

```
1 function love.update()  
2     bild_aktuell = bild_zu  
3     if sound.isPlaying() then bild_aktuell = bild_auf end  
4 end  
5  
6 function love.draw()  
7     love.graphics.draw( bild_aktuell, 0, 0 )  
8 end
```

`love.update()` berechnet, welches der Bilder das aktuelle ist. `love.draw()` zeichnet dieses. Beide Funktionen arbeiten 60 mal je Sekunde. Das Bild passt nicht ganz aber darum kümmern wir uns später.

2.3 Zufällige Miau Sounds

Ergänze `love.load()` mit folgender Liste bzw. Tabelle an Sounds:

```
1  soundliste = {
2    love.audio.newSource( "meow1.ogg" ),
3    love.audio.newSource( "meow2.ogg" ),
4    love.audio.newSource( "meow3.ogg" ),
5    love.audio.newSource( "meow4.ogg" ),
6    love.audio.newSource( "meow5.ogg" ),
7  }
```

Ersetze den Inhalt von `love.update()` mit Code, welcher die Soundliste benutzt:

```
1  bild_aktuell = bild_zu
2  for i,u in pairs(soundliste) do
3    if u.isPlaying() then bild_aktuell = bild_auf end
4  end
```

Ersetze den Inhalt von `love.mousepressed()` mit Code, welcher zufällige Sounds abspielt:

```
1  wahl = love.math.random(1,5)
2  soundliste[wahl]:stop()
3  soundliste[wahl]:play()
```

2.4 Anpassung an verschiedene Bildschirme

Ergänze den Inhalt von `love.load()` mit Berechnungen der Verhältnisse der Fenster- zu Bildgrößen:

```
1  fx = love.graphics.getWidth() / 1024
2  fy = love.graphics.getHeight() / 600
```

Ergänze den `love.graphics.draw()` Funktionsaufruf in `love.draw()` mit Skalier-Parametern:

```
1  love.graphics.draw(bild_aktuell, 0, 0, 0, fx, fy)
```

Das Bild wird dadurch an die Bildschirmgröße angepasst skaliert dargestellt, da Handys/Tablets nur eine Auflösung unterstützen. Dies ist zwar nicht optimal aber eine einfache Lösung für den Anfang.

2.5 Android-Port

Du kannst eigene Grafiken (am Computer oder auf Papier gemalt) oder eigene Sounds in Deine Miau Spiele-App einzubauen und das App-Icon ändern.

Wir empfehlen zu programmieren, dass der „Zurück“-Knopf die Android-App schließt:

```
1  function love.keypressed( key )
2    if key == "escape" then love.event.quit() end
3  end
```

Um die App auf Android spielbar zu machen, muss ein Zip-Archiv von dem Spiel erstellt werden, in `game.love` umbenannt werden und ins `StartGamedev`-Verzeichnis gelegt werden. Dann muss das `make-apk` Skript benutzt werden. Die resultierende `game.apk` muss dann aufs Handy/Tablet kopiert und dort installiert werden.

3 Katz und Maus Spiele-App

3.1 Bild und Sound

Tippe den folgenden Code ab (ohne -- Kommentare), speichere und teste diesen:

```
1 function love.load()
2     love.window.setMode( 1280, 720) -- Ändert Bildschirmgröße
3     grasBild = love.graphics.newImage( "grass.png" )
4     katzeBild = love.graphics.newImage( "cat.png" )
5     mausBild = love.graphics.newImage( "mouse.png" )
6     katzeX = 400 -- Position der Katze
7     katzeY = 300
8     mausX = 300 -- Position der Maus
9     mausY = 150
10    musik = love.audio.newSource( "music.ogg" )
11    musik:setLooping( true )
12    musik:play()
13 end
14
15 function love.draw()
16     love.graphics.draw( grasBild, 0, 0 )
17     love.graphics.draw( katzeBild, katzeX, katzeY )
18     love.graphics.draw( mausBild, mausX, mausY )
19 end
```

Der Code in `love.load()` verändert die Fensterauflösung, lädt Bilder und Musik, setzt Positions-Variablen und spielt die Musik. `love.draw()` malt die Bilder, 60 mal je Sekunde. Sie passen nicht ganz aber darum kümmern wir uns später.

3.2 Automatische und Interaktive Bewegung

Ergänze `love.load()` mit Mausclick-Positions-Variablen und Sounds:

```
1  klickX = 400
2  klickY = 300
3  quietsch = love.audio.newSource( "squeak.ogg" )
4  miau      = love.audio.newSource( "meow.ogg" )
```

Füge folgende drei Funktionen zu Deinem Code hinzu:

```
1  function distanz( x1, y1, x2, y2 )
2    a = x1 - x2
3    b = y1 - y2
4    return( math.sqrt( a^2 + b^2 ) )
5  end
6
7  function love.update()
8    mausX = mausX + 7
9    if mausX > 800 then
10     mausX = -48
11     mausY = love.math.random( 20, 400 )
12   end
13   if distanz( katzeX, katzeY, mausX, mausY ) < 40 then
14     quietsch:play()
15     mausX = 999
16   end
17   if distanz( katzeX, katzeY, klickX, klickY ) > 8 then
18     diffX = klickX - katzeX
19     diffY = klickY - katzeY
20     norm = math.sqrt( diffX^2 + diffY^2 )
21     einhX = diffX / norm
22     einhY = diffY / norm
23     katzeX = katzeX + einhX * 5
24     katzeY = katzeY + einhY * 5
25   end
26 end
27
28 function love.mousepressed( x, y )
29   klickX = x
30   klickY = y
31   miau:play()
32 end
```

Die Funktion `distanz()` berechnet den Abstand zwischen zwei Punkten mithilfe des Satzes des Pythagoras bzw. der Formel $c = \sqrt{a^2 + b^2}$.

`love.update()` 1. Bewegt die Maus, 2. Setzt die Maus zurück, wenn sie über den rechten Rand läuft oder 3. wenn Katze und Maus sich berühren, 4. Bewegt die Katze.

Der Code in `love.mousepressed()` verändert die `klickX` und `klickY` Variablen jedes Mal, wenn ein Mausknopf oder der Touchscreen berührt wird.

3.3 Bildschirmgröße

Ergänze den Inhalt von `love.load()` mit Berechnungen der Verhältnisse der Fenster- zu Bildgrößen:

```
1  fx = love.graphics.getWidth() / 800  
2  fy = love.graphics.getHeight() / 450
```

Ergänze die `love.graphics.draw()` Funktionsaufrufe in `love.draw()` mit Skalier-Parametern:

```
1  love.graphics.draw( grasBild, 0, 0, 0, fx, fy )  
2  love.graphics.draw( katzeBild, katzeX * fx, katzeY * fy, 0, fx, fy )  
3  love.graphics.draw( mausBild, mausX * fx, mausY * fy, 0, fx, fy )
```

Ersetze die Variablenzuweisungen in `love.mousepressed()`, um vom Bildschirm aufs Spielfeld zu projizieren:

```
1  klickX = x/fx  
2  klickY = y/fy
```

3.4 Punkte und Zeit

Ergänze den Inhalt von `love.load()` mit Bildgrößen, Schrift-Einstellung, Zeit und Punkten:

```
1  breite = love.graphics.getWidth()  
2  hoehe = love.graphics.getHeight()  
3  love.graphics.setNewFont(hoehe/15)  
4  zeitStart = love.timer.getTime()  
5  zeit = 30  
6  punkte = 0
```

Ergänze den Inhalt von `love.update()` mit einer Zeitberechnung:

```
1  zeit = 30 - math.floor(love.timer.getTime() - zeitStart)
```

Ergänze den `if`-Block in `love.update()`, der auf Katz-Maus Berührungen reagiert, mit einer Punkte-Hochrechnung:

```
1  if zeit > 0 then  
2      punkte = punkte + 1  
3  end
```

Ergänze den Inhalt von `love.draw()`, sodass Zeit und Punkte angezeigt werden:

```
1  text = "Zeit: " .. zeit .. ", Punkte: " .. punkte  
2  love.graphics.printf(text, 0, 0, breite, "center")
```

Du solltest den Inhalt von `love.update()` in einen `if zeit > 0 then ... end`-Block packen, um das Spiel nach Zeitablauf anzuhalten. Du kannst einen ähnlichen Block in `love.draw()` verwenden, um eine „Game Over!“ Nachricht anzuzeigen.

4 Matrix-Musik DJ-App

Tippe den folgenden Code ab (ohne -- Kommentare), speichere und teste diesen:

```
1 function love.load()
2   la, lg = love.audio, love.graphics
3   namen = { "lead", "drums", "drumsb", "clap" }
4   instr = {{},{}}          -- Instrumente-Tabelle mit...
5   for i = 1, 2 do          -- Zwei Zeilen und...
6     for j = 1, #namen do  -- Vier Spalten
7       instr[i][j] = {}
8       instr[i][j].snd = la.newSource( namen[j] .. i .. ".ogg" )
9       instr[i][j].snd:setLooping( true ) -- Endlosschleife an
10      instr[i][j].snd:setVolume( 0 )     -- Lautstärke auf 0
11      instr[i][j].snd:play()             -- Tracks werden abgespielt
12      instr[i][j].farbe = { 60*j, love.math.random(200), 200 }
13    end
14  end
15  spalten = #instr[1]      -- 4 Spalten
16  zeilen  = #instr        -- 2 Zeilen
17  breit   = lg.getWidth() -- Bildschirm-Größe
18  hoch    = lg.getHeight()
19  feldB   = breit / spalten -- Schaltfelder-Größe
20  feldH   = hoch / zeilen
21 end
22
23 function love.draw()
24   for i, zeile in ipairs(instr) do -- i ist Index, zeile ist Wert
25     for j, instrument in ipairs(zeile) do
26       lg.setColor(instrument.farbe) -- Instrumente haben eigene Farben
27       lg.rectangle( "fill", (j-1)*feldB, (i-1)*feldH, feldB, feldH )
28       if instrument.snd:getVolume() == 1 then
29         lg.setColor( 255, 255, 255, 95 ) -- An/Aus-Zustand wird gezeigt
30         lg.circle( "fill", (j-0.5)*feldB, (i-0.5)*feldH, feldB*0.4 )
31       end
32     end
33   end
34 end
35
36 function love.mousepressed(x, y) -- Wird von Maus/Touch gestartet
37   woBreit = math.ceil( x / feldB ) -- Spaltenberechnung
38   woHoch  = math.ceil( y / feldH ) -- Zeilenberechnung
39   if instr[woHoch][woBreit].snd:getVolume() == 1 then
40     instr[woHoch][woBreit].snd:setVolume(0) -- Lautstärke 0%
41   else
42     instr[woHoch][woBreit].snd:setVolume(1) -- Lautstärke 100%
43   end
44 end
```

Der Code macht intensiven Gebrauch von Tabellen/Listen und `for`-Schleifen, sowie von mathematischen Berechnungen, die etwas langsamer verdaut werden sollten.