Einstieg Spiele-Programmierung in LÖVE

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:

```
function love.load()
  sound = love.audio.newSource( "meow1.ogg" )
end

function love.mousepressed()
  sound:play()
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:

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

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

```
function love.update()
  bild_aktuell = bild_zu
  if sound:isPlaying() then bild_aktuell = bild_auf end
end

function love.draw()
  love.graphics.draw( bild_aktuell, 0, 0 )
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:

```
  soundliste = {
    love.audio.newSource( "meow1.ogg" ),
    love.audio.newSource( "meow2.ogg" ),
    love.audio.newSource( "meow3.ogg" ),
    love.audio.newSource( "meow4.ogg" ),
    love.audio.newSource( "meow5.ogg" ),
  }
```

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

```
  bild_aktuell = bild_zu
  for i,u in pairs(soundliste) do
    if u:isPlaying() then bild_aktuell = bild_auf end
  end
```

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

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

2.4 Anpassung an verschiedene Bildschirme

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

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

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

```
  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:

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

```
function love.load()
  love.window.setMode( 1280, 720) -- Ändert Bildschirmgröße
  grasBild  = love.graphics.newImage( "grass.png" )
  katzeBild = love.graphics.newImage( "cat.png" )
  mausBild  = love.graphics.newImage( "mouse.png" )
  katzeX = 400 -- Position der Katze
  katzeY = 300
  mausX = 300  -- Position der Maus
  mausY = 150
  musik = love.audio.newSource( "music.ogg" )
  musik:setLooping( true )
  musik:play()
end

function love.draw()
  love.graphics.draw( grasBild, 0, 0 )
  love.graphics.draw( katzeBild, katzeX, katzeY )
  love.graphics.draw( mausBild, mausX, mausY )
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 Mausklick-Positions-Variablen und Sounds:

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

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

```
function distanz( x1, y1, x2, y2 )
  a = x1 - x2
  b = y1 - y2
  return( math.sqrt( a^2 + b^2 ) )
end

function love.update()
  mausX = mausX + 7
  if mausX > 800 then
    mausX = -48
    mausY = love.math.random( 20, 400 )
  end
  if distanz( katzeX, katzeY, mausX, mausY ) < 40 then
    quietsch:play()
    mausX = 999
  end
  if distanz( katzeX, katzeY, klickX, klickY ) > 8 then
    diffX = klickX - katzeX
    diffY = klickY - katzeY
    norm = math.sqrt( diffX^2 + diffY^2 )
    einhX = diffX / norm
    einhY = diffY / norm
    katzeX = katzeX + einhX * 5
    katzeY = katzeY + einhY * 5
  end
end

function love.mousepressed( x, y )
  klickX = x
  klickY = y
  miau:play()
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:

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

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

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

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

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

3.4 Punkte und Zeit

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

```
breite = love.graphics.getWidth()
hoehe  = love.graphics.getHeight()
love.graphics.setNewFont(hoehe/15)
zeitStart = love.timer.getTime()
zeit   = 30
punkte = 0
```

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

```
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:

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

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

```
text = "Zeit: " .. zeit .. ", Punkte: " .. punkte
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:

```
function love.load()
  la, lg = love.audio, love.graphics
  namen = { "lead", "drums", "drumsb", "clap" }
  instr = {{},{}}         -- Instrumente-Tabelle mit...
  for i = 1, 2 do         -- Zwei Zeilen und...
    for j = 1, #namen do -- Vier Spalten
      instr[i][j] = {}
      instr[i][j].snd = la.newSource( namen[j] .. i .. ".ogg" )
      instr[i][j].snd:setLooping( true ) -- Endlosschleife an
      instr[i][j].snd:setVolume( 0 )     -- Lautstärke auf 0
      instr[i][j].snd:play()             -- Tracks werden abgespielt
      instr[i][j].farbe = { 60*j, love.math.random(200), 200 }
    end
  end
  spalten = #instr[1]       -- 4 Spalten
  zeilen  = #instr          -- 2 Zeilen
  breit   = lg.getWidth()   -- Bildschirm-Größe
  hoch    = lg.getHeight()
  feldB   = breit / spalten -- Schaltfelder-Größe
  feldH   = hoch / zeilen
end

function love.draw()
  for i, zeile in ipairs(instr) do  -- i ist Index, zeile ist Wert
    for j, instrument in ipairs(zeile) do
      lg.setColor(instrument.farbe) -- Instrumente haben eigene Farben
      lg.rectangle( "fill", (j-1)*feldB, (i-1)*feldH, feldB, feldH )
      if instrument.snd:getVolume() == 1 then
        lg.setColor( 255, 255, 255, 95 ) -- An/Aus-Zustand wird gezeigt
        lg.circle( "fill", (j-0.5)*feldB, (i-0.5)*feldH, feldB*0.4 )
      end
    end
  end
end

function love.mousepressed(x, y)   -- Wird von Maus/Touch gestartet
  woBreit = math.ceil( x / feldB ) -- Spaltenberechnung
  woHoch  = math.ceil( y / feldH ) -- Zeilenberechnung
  if instr[woHoch][woBreit].snd:getVolume() == 1 then
    instr[woHoch][woBreit].snd:setVolume(0) -- Lautstärke 0%
  else
    instr[woHoch][woBreit].snd:setVolume(1) -- Lautstärke 100%
  end
end
```

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