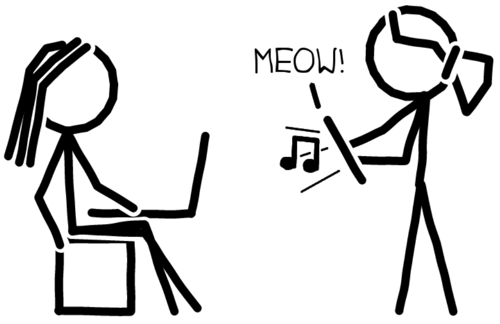
Start Gamedev - LÖVE Game Programming



1 Prepare

1. Extract **StartGamedev** and open the text editor using the `open-editor` file.
2. Read the tasks, type the code (*source code*) and test the results.

2 Meow game app

2.1 Interactive sound

Type in the following code, save it and test it:

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

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

The code in `love.load()` loads a sound file and `love.mousepressed()` plays it when a mouse button is pressed or the touchscreen is touched.

2.2 Interactive image

Insert the loading of two images into `love.load()`:

```lua
  img_open = love.graphics.newImage( "open.png" )
  img_closed = love.graphics.newImage( "closed.png" )
```

Add the following two functions to your code:

```lua
function love.update()
  img_current = img_closed
  if sound:isPlaying() then img_current = img_open end
end

function love.draw()
  love.graphics.draw( img_current, 0, 0 )
end
```

`love.update()` calculates, which of the images is the current one. `love.draw()` draws it. Both functions work 60 times per second. The image doesn't quite fit but we will take care of that later.

2.3 Random meow sounds

Add the following list (or table) of sounds to `love.load()`:

```
  soundlist = {
    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" ),
  }
```

Replace the content of `love.update()` with code, which uses the sound list:

```
  img_current = img_closed
  for i,u in pairs(soundlist) do
    if u:isPlaying() then img_current = img_open end
  end
```

Replace the content of `love.mousepressed()` with code which plays random sounds:

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

2.4 Adapt to different screens

Add calculations of the relations between image and window size to `love.load()`:

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

Add scaling parameters to the `love.graphics.draw()` function call in `love.draw()`:

```
  love.graphics.draw(img_current, 0, 0, 0, fx, fy)
```

The image fits to the screen size this way, since mobile phones/tablets only have one resolution. This is not optimal but a simple solution for the start.

2.5 Android port

You can put own graphics (drawn on the computer or on paper) and sounds into your meow game app and change the app icon.
We recommend to code the "back" button to close the Android app:

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

To make the app playable on Android, a zip archive of the game must be made, renamed to `game.love` and put into the `StartGamedev` directory. Then use the `make-apk` script. The resulting `game.apk` must then be put on the mobile phone/tablet and installed there.

3 Cat and mouse game app

3.1 Image and sound

Type in the following code (without `-- comments`), save it and test it:

```lua
function love.load()
  love.window.setMode( 1280, 720) -- Changes screen size
  grassImg  = love.graphics.newImage( "grass.png" )
  catImg    = love.graphics.newImage( "cat.png" )
  mouseImg  = love.graphics.newImage( "mouse.png" )
  catX = 400 -- Position of the cat
  catY = 300
  mouseX = 300  -- Position of the mouse
  mouseY = 150
  musik = love.audio.newSource( "music.ogg" )
  musik:setLooping( true )
  musik:play()
end

function love.draw()
  love.graphics.draw( grassImg, 0, 0 )
  love.graphics.draw( catImg, catX, catY )
  love.graphics.draw( mouseImg, mouseX, mouseY )
end
```

The code in `love.load()` changes the screen resolution, loads the images and music, sets position variables and plays the msuic. `love.draw()` draws the images, 60 times per second. They don't quite fit but we will take care of that later.

3.2 Automatic and interactive movement

Add mouse click position variables and sounds to `love.load()`:

```lua
  clickX = 400
  clickY = 300
  squeak = love.audio.newSource( "squeak.ogg" )
  meow   = love.audio.newSource( "meow.ogg" )
```

Add the following three functions to your code:

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

function love.update()
  mouseX = mouseX + 7
  if mouseX > 800 then
    mouseX = -48
    mouseY = love.math.random( 20, 400 )
  end
  if distance( catX, catY, mouseX, mouseY ) < 40 then
    squeak:play()
    mouseX = 999
  end
  if distance( catX, catY, clickX, clickY ) > 8 then
    diffX = clickX - catX
    diffY = clickY - catY
    norm = math.sqrt( diffX^2 + diffY^2 )
    unitX = diffX / norm
    unitY = diffY / norm
    catX = catX + unitX * 5
    catY = catY + unitY * 5
  end
end

function love.mousepressed( x, y )
  clickX = x
  clickY = y
  meow:play()
end
```

The `distance()` function calculates the distance between two dots thanks to the Pythagoras' theorem or the formula $c = \sqrt{a^2 + b^2}$.
`love.update()` 1. Moves the mouse, 2. Puts the mouse back, after it crosses the right border or 3. when cat and mouse touch, 4. moves the cat
The code in `love.mousepressed()` changes the `clickX` and `clickY` variables each time a mouse button is pressed or the touchscreen is touched.

3.3 Screen size

Add calculations of the relations between image and window size to `love.load()`:

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

Add scaling parameters to the `love.graphics.draw()` function call in `love.draw()`:

```lua
  love.graphics.draw( grassImg, 0, 0, 0, fx, fy )
  love.graphics.draw( catImg, catX * fx, catY * fy, 0, fx, fy )
  love.graphics.draw( mouseImg, mouseX * fx, mouseY * fy, 0, fx, fy )
```

Replace the variable assignments in `love.mousepressed()`, to project from the screen:

```lua
  clickX = x/fx
  clickY = y/fy
```

3.4 Score and time

Add image sizes, font configuration, time and score to `love.load()`:

```lua
  width  = love.graphics.getWidth()
  height = love.graphics.getHeight()
  love.graphics.setNewFont(height/15)
  timeStart = love.timer.getTime()
  time  = 30
  score = 0
```

Add time calculation to `love.update()`:

```lua
  time = 30 - math.floor(love.timer.getTime() - timeStart)
```

Add a score counter to the `if` block in `love.update()` which reacts to cat and mouse touching:

```lua
    if time > 0 then
      score = score + 1
    end
```

Add displaying time and score to `love.draw()`:

```lua
  text = "Time: " .. time .. ", Score: " .. score
  love.graphics.printf(text, 0, 0, width, "center")
```

You should put the content of `love.update()` into a `if time > 0 then ...  end` block to stop the game after the time runs out. You can use a similar block in `love.draw()` to display a “Game Over!” message.

4 Matrix music DJ app

Type in the following code (without `-- comments`), save it and test it:

```lua
function love.load()
  la, lg = love.audio, love.graphics
  names = { "lead", "drums", "drumsb", "clap" }
  instr = {{},{}}        -- Table of instruments with...
  for i = 1, 2 do        -- two rows and...
    for j = 1, #names do -- four columns
      instr[i][j] = {}
      instr[i][j].snd = la.newSource( names[j] .. i .. ".ogg" )
      instr[i][j].snd:setLooping( true ) -- Endless looping on
      instr[i][j].snd:setVolume( 0 )     -- Loudness to 0
      instr[i][j].snd:play()             -- Track playback starts
      instr[i][j].color = { 60*j, love.math.random(200), 200 }
    end
  end
  columns = #instr[1]       -- 4 columns
  rows    = #instr          -- 2 rows
  width   = lg.getWidth()   -- Screen size
  height  = lg.getHeight()
  fieldW  = width / columns -- Touch field size
  fieldH  = height / rows
end

function love.draw()
  for i, row in ipairs(instr) do    -- i is the index, row is the value
    for j, instrument in ipairs(row) do
      lg.setColor(instrument.color) -- Instruments have own colors
      lg.rectangle( "fill", (j-1)*fieldW, (i-1)*fieldH, fieldW, fieldH )
      if instrument.snd:getVolume() == 1 then
        lg.setColor( 255, 255, 255, 95 ) -- on/off state is displayed
        lg.circle( "fill", (j-0.5)*fieldW, (i-0.5)*fieldH, fieldW*0.4 )
      end
    end
  end
end

function love.mousepressed(x, y)   -- Gets started by mouse/touch
  whereW = math.ceil( x / fieldW ) -- Calculating column
  whereH = math.ceil( y / fieldH ) -- Calculating row
  if instr[whereH][whereW].snd:getVolume() == 1 then
    instr[whereH][whereW].snd:setVolume(0) -- Loudness 0%
  else
    instr[whereH][whereW].snd:setVolume(1) -- Loudness 100%
  end
end
```

The code makes intense use of tables/lists and `for` loops as well as calculations, which might need a bit more time to be understood.