

Start Gamedev - LÖVE Game Programming

© 2016 Iwan Gabovitch (qubodup.itch.io/startgamedev)
Licensed under an [Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/)



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:

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

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

Add the following two functions to your code:

```
1 function love.update()  
2     img_current = img_closed  
3     if sound:isPlaying() then img_current = img_open end  
4 end  
5  
6 function love.draw()  
7     love.graphics.draw( img_current, 0, 0 )  
8 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()`:

```
1  soundlist = {
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  }
```

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

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

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

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

2.4 Adapt to different screens

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

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

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

```
1  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:

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

```
1 function love.load()
2     love.window.setMode( 1280, 720) -- Changes screen size
3     grassImg = love.graphics.newImage( "grass.png" )
4     catImg   = love.graphics.newImage( "cat.png" )
5     mouseImg = love.graphics.newImage( "mouse.png" )
6     catX = 400 -- Position of the cat
7     catY = 300
8     mouseX = 300 -- Position of the mouse
9     mouseY = 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( grassImg, 0, 0 )
17     love.graphics.draw( catImg, catX, catY )
18     love.graphics.draw( mouseImg, mouseX, mouseY )
19 end
```

The code in `love.load()` changes the screen resolution, loads the images and music, sets position variables and plays the music. `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()`:

```
1 clickX = 400
2 clickY = 300
3 squeak = love.audio.newSource( "squeak.ogg" )
4 meow    = love.audio.newSource( "meow.ogg" )
```

Add the following three functions to your code:

```
1 function distance( 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     mouseX = mouseX + 7
9     if mouseX > 800 then
10         mouseX = -48
11         mouseY = love.math.random( 20, 400 )
12     end
13     if distance( catX, catY, mouseX, mouseY ) < 40 then
14         squeak:play()
15         mouseX = 999
16     end
17     if distance( catX, catY, clickX, clickY ) > 8 then
18         diffX = clickX - catX
19         diffY = clickY - catY
20         norm = math.sqrt( diffX^2 + diffY^2 )
21         unitX = diffX / norm
22         unitY = diffY / norm
23         catX = catX + unitX * 5
24         catY = catY + unitY * 5
25     end
26 end
27
28 function love.mousepressed( x, y )
29     clickX = x
30     clickY = y
31     meow:play()
32 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()`:

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

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

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

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

```
1  clickX = x/fx
2  clickY = y/fy
```

3.4 Score and time

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

```
1  width  = love.graphics.getWidth()
2  height = love.graphics.getHeight()
3  love.graphics.setNewFont(height/15)
4  timeStart = love.timer.getTime()
5  time     = 30
6  score    = 0
```

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

```
1  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:

```
1  if time > 0 then
2      score = score + 1
3  end
```

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

```
1  text = "Time: " .. time .. ", Score: " .. score
2  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:

```
1 function love.load()
2   la, lg = love.audio, love.graphics
3   names = { "lead", "drums", "drumsb", "clap" }
4   instr = {{},{}} -- Table of instruments with...
5   for i = 1, 2 do -- two rows and...
6     for j = 1, #names do -- four columns
7       instr[i][j] = {}
8       instr[i][j].snd = la.newSource( names[j] .. i .. ".ogg" )
9       instr[i][j].snd:setLooping( true ) -- Endless looping on
10      instr[i][j].snd:setVolume( 0 ) -- Loudness to 0
11      instr[i][j].snd:play() -- Track playback starts
12      instr[i][j].color = { 60*j, love.math.random(200), 200 }
13    end
14  end
15  columns = #instr[1] -- 4 columns
16  rows = #instr -- 2 rows
17  width = lg.getWidth() -- Screen size
18  height = lg.getHeight()
19  fieldW = width / columns -- Touch field size
20  fieldH = height / rows
21 end
22
23 function love.draw()
24   for i, row in ipairs(instr) do -- i is the index, row is the value
25     for j, instrument in ipairs(row) do
26       lg.setColor(instrument.color) -- Instruments have own colors
27       lg.rectangle( "fill", (j-1)*fieldW, (i-1)*fieldH, fieldW, fieldH )
28       if instrument.snd:getVolume() == 1 then
29         lg.setColor( 255, 255, 255, 95 ) -- on/off state is displayed
30         lg.circle( "fill", (j-0.5)*fieldW, (i-0.5)*fieldH, fieldW*0.4 )
31       end
32     end
33   end
34 end
35
36 function love.mousepressed(x, y) -- Gets started by mouse/touch
37   whereW = math.ceil( x / fieldW ) -- Calculating column
38   whereH = math.ceil( y / fieldH ) -- Calculating row
39   if instr[whereH][whereW].snd:getVolume() == 1 then
40     instr[whereH][whereW].snd:setVolume(0) -- Loudness 0%
41   else
42     instr[whereH][whereW].snd:setVolume(1) -- Loudness 100%
43   end
44 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.