Start Gamedev Introduction to LÖVE

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.
3. Inside one task (e.g. 1.1), continuously expand the code. When you begin a new task (e.g. 1.2), clear the code in your text editor first.
4. Functions (e.g. `function love.draw() ... end`) should only appear once.
5. Use the Tab key to indent (left of Q). Keep your code readable.
6. Functions, loops and conditions end with `end`. The lines above `end` are the body.
7. Your line numbers can differ from the numbers in the tasks.

2 Better than paper: draw in LÖVE

2.1 Your favourite rectangle

A rectangle at position x=100, y=200. 300 pixels in width and 150 in height.

```
1 function love.draw()
2   love.graphics.rectangle("fill",100,200,300,150)
3 end
```

1. Move the rectangle.
2. The screen is of size 800,600. Align the rectangle with the upper right corner.
3. Replace `"fill"` with `"line"`, what happens?
4. Draw a second rectangle somewhere else. Copy only line 2.
5. Make the screen white.

2.2 Two rectangles

```
1 function love.draw()
2   love.graphics.setColor(0,255,0)
3   love.graphics.rectangle("fill",100,200,300,150)
4   love.graphics.setColor(255,255,255)
5   love.graphics.rectangle("fill",300,400,100,50)
6 end
```

1. Change numbers in line 2. What happens?
2. This representation of colors using three numbers (0-255) is called RGB (Red-Green-Blue). Make the smaller rectangle blue.
3. Move the rectangles so that they overlap. Which rectangle is in front?
4. Swap lines 3 and 5. What changed?

2.3 Some lines

```
function love.draw()
  love.graphics.line(100,0,100,200)
  love.graphics.line(0,200,100,200)
  love.graphics.rectangle("fill",100,200,300,150)
end
```

1. Move the rectangle. Adjust the lines accordingly.
2. This was tedious. Variable can do this automatically for us! Read on.

2.4 Variables

```
x = 100
y = 200

function love.draw()
  love.graphics.line(100,0,x,y)
  love.graphics.line(0,200,x,y)
  love.graphics.rectangle("fill",x,y,300,150)
end
```

1. What would happen if you changed `x` and `y`?
2. Change line 2 to: `y = x`. What does this mean?
3. Change line 2 back to: `y = 200`. Change line 1 to: `x = y`. You will get an error. Can you correct the code?
4. Introduce a variable for the width of the rectangle.

3 Interaction

3.1 A moving picture

```
x = 100
y = 200

function love.draw()
  love.graphics.line(100,0,x,y)
  love.graphics.line(0,200,x,y)
  love.graphics.rectangle("fill",x,y,300,150)
end

function love.mousepressed()
  x = x + 10
end
```

1. Try clicking the game. Something should happen.
2. Make the box go backwards.
3. Make the box go upwards.
4. Make the box bigger on mousepress.

3.2 Asking for the right click/touch

```
1  a = 100
2  b = 200
3
4  function love.draw()
5    love.graphics.rectangle("fill",a,b,300,150)
6  end
7
8  function love.mousepressed(mx, my)
9    local dir = "right"
10   if mx < 400 then dir = "left" end
11   if dir == "right" then a = a + 10 end
12 end
```

1. Where do you have to click to move the rectangle?
2. Copy line 11 but change it to reduce `a` when `dir == "left"`.
3. Let the box touch the border but let it go no further (e.g. add `and a < 500` before `then`).

3.3 It needs to do things on its own

```
1  x = 100
2  y = 200
3
4  function love.draw()
5    love.graphics.rectangle("fill",x,y,300,150)
6  end
7
8  function love.update()
9    y = y - 1
10 end
```

Everything inside the `love.update` block is executed 60 times per second.

1. Stop the box at the top. `if y > 200 then` ... `end` or similar might help.
2. At the top of the code insert `velocity = 1`. Let the box move with `y = y - velocity` instead of `y = y - 1`.
3. Reduce `veolcity` continuously by 0.01. This simulates gravity.
4. Increase `velocity` when you click it (use `love.mousepressed()`).
5. Stop the box at the bottom of the screen.
6. Print the velocity to screen with `love.graphics.print(velocity,10,10)`
7. Set `velocity` to 0 when the rectangle touches the top border.
8. Give the player a goal. Notify the player when they reached that goal. Example: A cheap parking game. Draw a line at height 100. Change the color of the box, `if 0.5 > velocity and velocity > -0.5 and 105 > y and velocity > 95 then`.

3.4 Loops

The while-loop executes the program written into it (its body) as long as its condition `y < 500` is true.

```
1  x = 0
2
3  function love.draw()
4    y = 0
5
6    while y < 500 do
7      love.graphics.rectangle("fill",x,y,300,150)
8      y = y + 200
9    end
10 end
11
12 function love.mousepressed(mx, my)
13   x = x + 50
14 end
```

1. Switch lines 7 and 8. Do you understand what difference this makes?
2. Move line 4 in line 2. The screen should go black. Why is that?
3. Revert your changes, then draw more, but smaller rectangles vertically using the while loop.
4. Introduce a new variable `z = 0`. Add a new while loop. Make it increase `z` and let it draw some rectangles horizontally.
5. Move `z = 0` and the new while loop inside the old while loop. Now every time `y` is increased your program goes through all your z values! Use this to make a checkerboard.
6. Let the whole checkerboard be moved by clicking the mouse.

3.5 Lists

```
1  a = {100,200,500}
2
3  function love.draw()
4    i = 1
5    while i <= 3 do
6      love.graphics.rectangle("fill",a[i],a[i],10,10)
7      i = i + 1
8    end
9  end
```

`a` ist a list (table). `a[1]` equals `100`. Here, `1` is the *index* of `100` in `a`.

1. Add a number to the list `a`. `#a` is the length of a. Draw all 4 elements.
2. `a[5] = 5 * 10` sets a new element. Use a while loop to let `a` be `{10,20,....,200}`.
3. `a[#a+1] = v` is the same as adding `v` to the list. Whenever the mouse is clicked, add the x-coordinate of the click to the list.