The Complete Love2D Tutorial Book

Table of Contents

Introduction to Love2D

Love2D (LÖVE) is a free, open-source 2D game engine written in C++ that uses the Lua programming language for game logic. It's designed to be simple, lightweight, and cross-platform, making it perfect for indie game development.

Why Choose Love2D?

- **Easy to Learn**: Simple API and Lua scripting
- **Cross-Platform**: Windows, macOS, Linux, Android, iOS
- **Lightweight**: Small file size and minimal dependencies
- **Active Community**: Great documentation and community support
- **No License Fees**: Completely free and open source

Installation and Setup

Installing Love2D

Windows: 1. Download from love2d.org 2. Extract the ZIP file 3. Add the folder to your PATH (optional)

macOS:

```
brew install love
```

Ubuntu/Debian:

```
sudo apt install love
```

Setting Up Your Development Environment

Create a project folder structure:

```
my-game/
├── main.lua
├── conf.lua
├── assets/
│   ├── images/
│   ├── sounds/
│   └── fonts/
└── src/
    └── (additional lua files)
```

Running Your Game

```
love .
# or
love my-game/
```

Your First Love2D Game

Let's create a simple "Hello World" program:

main.lua

```lua
function love.draw()
    love.graphics.print("Hello, Love2D!", 400, 300)
end
```

Output: A window displaying "Hello, Love2D!" text at position (400, 300).

Adding Configuration

Create `conf.lua` to configure your game window:

```lua
function love.conf(t)
    t.title = "My First Love2D Game"
    t.window.width = 800
    t.window.height = 600
    t.window.resizable = false
    t.window.vsync = 1
end
```

Output: An 800x600 window with the title "My First Love2D Game".

Basic Game Structure

Love2D games are built around callback functions that handle different aspects of the game loop:

```lua
-- Game variables
local player = {
    x = 400,
    y = 300,
    speed = 200
}

-- Called once at the start
function love.load()
    love.window.setTitle("Game Structure Example")
    love.graphics.setBackgroundColor(0.1, 0.1, 0.2)
end

-- Called every frame to update game logic
function love.update(dt)
    -- dt is delta time (time since last frame)
    if love.keyboard.isDown("left") then
        player.x = player.x - player.speed * dt
    end
    if love.keyboard.isDown("right") then
        player.x = player.x + player.speed * dt
    end
end

-- Called every frame to draw graphics
function love.draw()
    love.graphics.setColor(1, 1, 1) -- white
    love.graphics.circle("fill", player.x, player.y, 20)

    -- Reset color
    love.graphics.setColor(1, 1, 1)
    love.graphics.print("Use arrow keys to move", 10, 10)
end

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

Output: A white circle that moves left and right with arrow keys, with instructions displayed at the top.

Drawing and Graphics

Basic Shapes

```lua
function love.draw()
    -- Set colors (red, green, blue, alpha) - values from 0 to 1
    love.graphics.setColor(1, 0, 0) -- Red
    love.graphics.rectangle("fill", 50, 50, 100, 80)

    love.graphics.setColor(0, 1, 0) -- Green
    love.graphics.circle("line", 200, 90, 40)

    love.graphics.setColor(0, 0, 1) -- Blue
    love.graphics.polygon("fill", 300, 50, 350, 50, 325, 130)

    love.graphics.setColor(1, 1, 0) -- Yellow
    love.graphics.line(400, 50, 500, 130, 450, 200)

    -- Reset to white
    love.graphics.setColor(1, 1, 1)
end
```

Output: A red filled rectangle, green circle outline, blue triangle, and yellow zigzag line.

Working with Images

```lua
local playerImg, backgroundImg

function love.load()
    -- Load images (place in assets/images/ folder)
    playerImg = love.graphics.newImage("assets/images/player.png")
    backgroundImg = love.graphics.newImage("assets/images/background.png")
end

function love.draw()
    -- Draw background
    love.graphics.draw(backgroundImg, 0, 0)

    -- Draw player with scaling and rotation
    local x, y = 400, 300
    local rotation = 0
    local scaleX, scaleY = 2, 2
    local offsetX = playerImg:getWidth() / 2
    local offsetY = playerImg:getHeight() / 2

    love.graphics.draw(playerImg, x, y, rotation, scaleX, scaleY, offsetX,
end
```

Text and Fonts

```lua
local font, largeFont

function love.load()
    font = love.graphics.newFont("assets/fonts/arial.ttf", 18)
    largeFont = love.graphics.newFont(32)
end

function love.draw()
    -- Default font
    love.graphics.print("Default font text", 10, 10)

    -- Custom font
    love.graphics.setFont(font)
    love.graphics.print("Custom font text", 10, 40)

    -- Large font with color
    love.graphics.setFont(largeFont)
    love.graphics.setColor(1, 0.5, 0) -- Orange
    love.graphics.print("Large orange text", 10, 80)

    love.graphics.setColor(1, 1, 1) -- Reset to white
end
```

Output: Text displayed in different fonts and sizes, with the last line in orange.

Input Handling

Keyboard Input

```lua
local player = {x = 400, y = 300, speed = 300}
local keys = {}

function love.update(dt)
    -- Continuous key checking
    if love.keyboard.isDown("w", "up") then
        player.y = player.y - player.speed * dt
    end
    if love.keyboard.isDown("s", "down") then
        player.y = player.y + player.speed * dt
    end
    if love.keyboard.isDown("a", "left") then
        player.x = player.x - player.speed * dt
    end
    if love.keyboard.isDown("d", "right") then
        player.x = player.x + player.speed * dt
    end
end

function love.keypressed(key)
    -- Single key press events
    keys[key] = true

    if key == "space" then
        -- Reset player position
        player.x, player.y = 400, 300
    elseif key == "escape" then
        love.event.quit()
    end
end

function love.keyreleased(key)
    keys[key] = false
end

function love.draw()
    love.graphics.circle("fill", player.x, player.y, 25)
    love.graphics.print("WASD or Arrow keys to move, Space to reset", 10,

    -- Show pressed keys
    local y = 40
    for key, pressed in pairs(keys) do
        if pressed then
            love.graphics.print("Key pressed: " .. key, 10, y)
            y = y + 20
        end
    end
end
```

Mouse Input

```lua
local circles = {}

function love.mousepressed(x, y, button)
    if button == 1 then -- Left mouse button
        table.insert(circles, {x = x, y = y, radius = 0, color = {math.ran
    end
end

function love.update(dt)
    -- Grow circles
    for i, circle in ipairs(circles) do
        circle.radius = circle.radius + 50 * dt

        -- Remove large circles
        if circle.radius > 50 then
            table.remove(circles, i)
        end
    end
end

function love.draw()
    -- Draw circles
    for _, circle in ipairs(circles) do
        love.graphics.setColor(circle.color[1], circle.color[2], circle.co
        love.graphics.circle("line", circle.x, circle.y, circle.radius)
    end

    love.graphics.setColor(1, 1, 1)
    love.graphics.print("Click to create expanding circles", 10, 10)

    -- Show mouse position
    local mx, my = love.mouse.getPosition()
    love.graphics.print("Mouse: " .. mx .. ", " .. my, 10, 30)
end
```

Output: Clicking creates colorful expanding circles that disappear after reaching maximum size.

Animation and Movement

Sprite Animation

```lua
local animation = {
    spriteSheet = nil,
    quads = {},
    currentFrame = 1,
    timer = 0,
    frameTime = 0.1
}

function love.load()
    -- Load sprite sheet (assuming 4 frames of 32x32 pixels)
    animation.spriteSheet = love.graphics.newImage("assets/images/characte

    -- Create quads for each frame
    local frameWidth, frameHeight = 32, 32
    for i = 0, 3 do
        animation.quads[i + 1] = love.graphics.newQuad(
            i * frameWidth, 0, frameWidth, frameHeight,
            animation.spriteSheet:getDimensions()
        )
    end
end

function love.update(dt)
    animation.timer = animation.timer + dt

    if animation.timer >= animation.frameTime then
        animation.timer = 0
        animation.currentFrame = animation.currentFrame + 1

        if animation.currentFrame > #animation.quads then
            animation.currentFrame = 1
        end
    end
end

function love.draw()
    love.graphics.draw(
        animation.spriteSheet,
        animation.quads[animation.currentFrame],
        400, 300, 0, 4, 4 -- Scale by 4x
    )
end
```

Smooth Movement with Interpolation

```lua
local player = {
    x = 100, y = 300,
    targetX = 100, targetY = 300,
    speed = 5
}

function love.update(dt)
    -- Smooth movement towards target
    local dx = player.targetX - player.x
    local dy = player.targetY - player.y

    player.x = player.x + dx * player.speed * dt
    player.y = player.y + dy * player.speed * dt
end

function love.mousepressed(x, y, button)
    if button == 1 then
        player.targetX = x
        player.targetY = y
    end
end

function love.draw()
    -- Draw target
    love.graphics.setColor(0.5, 0.5, 0.5)
    love.graphics.circle("line", player.targetX, player.targetY, 30)

    -- Draw player
    love.graphics.setColor(1, 0, 0)
    love.graphics.circle("fill", player.x, player.y, 20)

    love.graphics.setColor(1, 1, 1)
    love.graphics.print("Click to set target position", 10, 10)
end
```

Output: A red circle smoothly moves toward wherever you click, with a gray target circle showing the destination.

Sound and Music

Basic Audio

```lua
local sounds = {}
local music

function love.load()
    -- Load sound effects
    sounds.jump = love.audio.newSource("assets/sounds/jump.wav", "static")
    sounds.collect = love.audio.newSource("assets/sounds/collect.wav", "st

    -- Load background music
    music = love.audio.newSource("assets/sounds/background.ogg", "stream")
    music:setLooping(true)
    music:play()
end

function love.keypressed(key)
    if key == "space" then
        sounds.jump:play()
    elseif key == "c" then
        sounds.collect:play()
    elseif key == "m" then
        -- Toggle music
        if music:isPlaying() then
            music:pause()
        else
            music:play()
        end
    end
end

function love.draw()
    love.graphics.print("Space: Jump sound", 10, 10)
    love.graphics.print("C: Collect sound", 10, 30)
    love.graphics.print("M: Toggle music", 10, 50)

    local volume = love.audio.getVolume()
    love.graphics.print("Master Volume: " .. math.floor(volume * 100) .. "
end
```

Dynamic Audio Control

```lua
local audioManager = {
    masterVolume = 1.0,
    musicVolume = 0.7,
    sfxVolume = 0.8
}

function love.load()
    love.audio.setVolume(audioManager.masterVolume)
end

function love.update(dt)
    -- Volume controls
    if love.keyboard.isDown("up") then
        audioManager.masterVolume = math.min(1.0, audioManager.masterVolum
        love.audio.setVolume(audioManager.masterVolume)
    elseif love.keyboard.isDown("down") then
        audioManager.masterVolume = math.max(0.0, audioManager.masterVolum
        love.audio.setVolume(audioManager.masterVolume)
    end
end

function love.draw()
    love.graphics.print("Up/Down arrows: Adjust master volume", 10, 10)
    love.graphics.print("Master Volume: " .. math.floor(audioManager.maste
end
```

Game Physics

Basic Physics System

```lua
local world
local objects = {}

function love.load()
    -- Create physics world with gravity
    love.physics.setMeter(64) -- 64 pixels per meter
    world = love.physics.newWorld(0, 9.81 * 64, true) -- gravity pointing

    -- Create ground
    local ground = {}
    ground.body = love.physics.newBody(world, 400, 550, "static")
    ground.shape = love.physics.newRectangleShape(800, 100)
    ground.fixture = love.physics.newFixture(ground.body, ground.shape)
    ground.fixture:setUserData("ground")
    table.insert(objects, ground)

    -- Create a box
    createBox(400, 100)
end

function createBox(x, y)
    local box = {}
    box.body = love.physics.newBody(world, x, y, "dynamic")
    box.shape = love.physics.newRectangleShape(40, 40)
    box.fixture = love.physics.newFixture(box.body, box.shape)
    box.fixture:setRestitution(0.6) -- Bounciness
    box.fixture:setUserData("box")
    table.insert(objects, box)
end

function love.update(dt)
    world:update(dt)
end

function love.mousepressed(x, y, button)
    if button == 1 then
        createBox(x, y)
    end
end

function love.draw()
    -- Draw all physics objects
    for _, obj in ipairs(objects) do
        local body = obj.body
        local shape = obj.shape

        if obj.fixture:getUserData() == "ground" then
            love.graphics.setColor(0, 1, 0) -- Green for ground
        else
            love.graphics.setColor(1, 0, 0) -- Red for boxes
        end

        if shape:getType() == "rectangle" then
            love.graphics.polygon("fill", body:getWorldPoints(shape:getPoi
        end
    end

    love.graphics.setColor(1, 1, 1)
    love.graphics.print("Click to drop boxes", 10, 10)
end
```

Output: Boxes fall due to gravity and bounce on a green ground platform when you click.

Collision Detection

AABB Collision Detection

```lua
-- Axis-Aligned Bounding Box collision detection
function checkCollision(a, b)
    return a.x < b.x + b.width and
           b.x < a.x + a.width and
           a.y < b.y + b.height and
           b.y < a.y + a.height
end

local player = {x = 50, y = 50, width = 40, height = 40, speed = 200}
local obstacles = {
    {x = 200, y = 100, width = 60, height = 60},
    {x = 400, y = 200, width = 80, height = 40},
    {x = 300, y = 350, width = 100, height = 20}
}
local collectibles = {
    {x = 150, y = 150, width = 20, height = 20, collected = false},
    {x = 500, y = 300, width = 20, height = 20, collected = false}
}

function love.update(dt)
    local newX, newY = player.x, player.y

    -- Movement
    if love.keyboard.isDown("left") then
        newX = newX - player.speed * dt
    end
    if love.keyboard.isDown("right") then
        newX = newX + player.speed * dt
    end
    if love.keyboard.isDown("up") then
        newY = newY - player.speed * dt
    end
    if love.keyboard.isDown("down") then
        newY = newY + player.speed * dt
    end

    -- Collision check with obstacles
    local testPlayer = {x = newX, y = newY, width = player.width, height =
    local collision = false

    for _, obstacle in ipairs(obstacles) do
        if checkCollision(testPlayer, obstacle) then
            collision = true
            break
        end
    end

    -- Only move if no collision
    if not collision then
        player.x, player.y = newX, newY
    end

    -- Check collectibles
    for _, collectible in ipairs(collectibles) do
        if not collectible.collected and checkCollision(player, collectibl
            collectible.collected = true
        end
    end
end

function love.draw()
    -- Draw player
    love.graphics.setColor(0, 0, 1) -- Blue
    love.graphics.rectangle("fill", player.x, player.y, player.width, play
```

```
    -- Draw obstacles
    love.graphics.setColor(1, 0, 0) -- Red
    for _, obstacle in ipairs(obstacles) do
        love.graphics.rectangle("fill", obstacle.x, obstacle.y, obstacle.w
    end

    -- Draw collectibles
    love.graphics.setColor(1, 1, 0) -- Yellow
    for _, collectible in ipairs(collectibles) do
        if not collectible.collected then
            love.graphics.rectangle("fill", collectible.x, collectible.y,
        end
    end

    love.graphics.setColor(1, 1, 1)
    love.graphics.print("Use arrow keys to move. Avoid red obstacles, coll
end
```

Circle Collision Detection

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

function circleCollision(circle1, circle2)
    local dist = distance(circle1.x, circle1.y, circle2.x, circle2.y)
    return dist < (circle1.radius + circle2.radius)
end

local player = {x = 400, y = 300, radius = 25, speed = 200}
local enemies = {
    {x = 200, y = 200, radius = 30, vx = 50, vy = 30},
    {x = 600, y = 400, radius = 25, vx = -40, vy = -50}
}

function love.update(dt)
    -- Player movement
    if love.keyboard.isDown("left") then
        player.x = player.x - player.speed * dt
    end
    if love.keyboard.isDown("right") then
        player.x = player.x + player.speed * dt
    end
    if love.keyboard.isDown("up") then
        player.y = player.y - player.speed * dt
    end
    if love.keyboard.isDown("down") then
        player.y = player.y + player.speed * dt
    end

    -- Enemy movement and boundary bouncing
    for _, enemy in ipairs(enemies) do
        enemy.x = enemy.x + enemy.vx * dt
        enemy.y = enemy.y + enemy.vy * dt

        -- Bounce off walls
        if enemy.x - enemy.radius < 0 or enemy.x + enemy.radius > 800 then
            enemy.vx = -enemy.vx
        end
        if enemy.y - enemy.radius < 0 or enemy.y + enemy.radius > 600 then
            enemy.vy = -enemy.vy
        end

        -- Check collision with player
        if circleCollision(player, enemy) then
            -- Reset player position
            player.x, player.y = 400, 300
        end
    end
end

function love.draw()
    -- Draw player
    love.graphics.setColor(0, 1, 0) -- Green
    love.graphics.circle("fill", player.x, player.y, player.radius)

    -- Draw enemies
    love.graphics.setColor(1, 0, 0) -- Red
    for _, enemy in ipairs(enemies) do
        love.graphics.circle("fill", enemy.x, enemy.y, enemy.radius)
    end

    love.graphics.setColor(1, 1, 1)
    love.graphics.print("Avoid the red circles!", 10, 10)
end
```

Output: A green circle (player) that resets position when touching red bouncing enemy circles.

Game States

State Management System

```lua
local gameState = "menu"
local states = {}

-- Menu State
states.menu = {}
function states.menu.draw()
    love.graphics.setColor(1, 1, 1)
    love.graphics.print("MAIN MENU", 350, 200, 0, 2, 2)
    love.graphics.print("Press SPACE to start", 320, 300)
    love.graphics.print("Press Q to quit", 340, 350)
end

function states.menu.keypressed(key)
    if key == "space" then
        gameState = "game"
    elseif key == "q" then
        love.event.quit()
    end
end

-- Game State
states.game = {}
local player = {x = 400, y = 300, score = 0}

function states.game.update(dt)
    if love.keyboard.isDown("left") then
        player.x = player.x - 200 * dt
    end
    if love.keyboard.isDown("right") then
        player.x = player.x + 200 * dt
    end

    player.score = player.score + dt * 10
end

function states.game.draw()
    love.graphics.setColor(0, 1, 0)
    love.graphics.circle("fill", player.x, player.y, 25)

    love.graphics.setColor(1, 1, 1)
    love.graphics.print("Score: " .. math.floor(player.score), 10, 10)
    love.graphics.print("Left/Right to move, ESC for pause", 10, 30)
end

function states.game.keypressed(key)
    if key == "escape" then
        gameState = "pause"
    end
end

-- Pause State
states.pause = {}
function states.pause.draw()
    -- Draw game in background (dimmed)
    love.graphics.setColor(0.5, 0.5, 0.5)
    states.game.draw()

    -- Draw pause overlay
    love.graphics.setColor(0, 0, 0, 0.7)
    love.graphics.rectangle("fill", 0, 0, 800, 600)

    love.graphics.setColor(1, 1, 1)
    love.graphics.print("PAUSED", 350, 250, 0, 2, 2)
    love.graphics.print("Press ESC to resume", 320, 320)
    love.graphics.print("Press M to return to menu", 300, 350)
```

```lua
    love.graphics.print("Press M to return to menu", 300, 330)
end

function states.pause.keypressed(key)
    if key == "escape" then
        gameState = "game"
    elseif key == "m" then
        gameState = "menu"
        -- Reset game
        player.x, player.y, player.score = 400, 300, 0
    end
end

-- Main Love2D callbacks
function love.update(dt)
    if states[gameState].update then
        states[gameState].update(dt)
    end
end

function love.draw()
    if states[gameState].draw then
        states[gameState].draw()
    end
end

function love.keypressed(key)
    if states[gameState].keypressed then
        states[gameState].keypressed(key)
    end
end
```

Output: A complete state system with menu, game, and pause states, each with their own logic and display.

File I/O and Save Systems

Basic Save System

```lua
local saveData = {
    highScore = 0,
    playerName = "Player",
    level = 1,
    unlockedLevels = {1}
}

function saveGame()
    local serialized = serialize(saveData)
    love.filesystem.write("savegame.lua", "return " .. serialized)
end

function loadGame()
    if love.filesystem.getInfo("savegame.lua") then
        local chunk = love.filesystem.load("savegame.lua")
        if chunk then
            saveData = chunk()
        end
    end
end

function serialize(t)
    local result = "{"
    for k, v in pairs(t) do
        if type(k) == "string" then
            result = result .. k .. "="
        else
            result = result .. "[" .. k .. "]="
```

```lua
        end

        if type(v) == "table" then
            result = result .. serialize(v)
        elseif type(v) == "string" then
            result = result .. '"' .. v .. '"'
        else
            result = result .. tostring(v)
        end
        result = result .. ","
    end
    result = result .. "}"
    return result
end

local currentScore = 0

function love.load()
    loadGame()
end

function love.update(dt)
    currentScore = currentScore + dt * 5

    if currentScore > saveData.highScore then
        saveData.highScore = currentScore
    end
end

function love.keypressed(key)
    if key == "s" then
        saveGame()
    elseif key == "l" then
        loadGame()
    elseif key == "r" then
        -- Reset save
        saveData = {highScore = 0, playerName = "Player", level = 1, unloc
        currentScore = 0
    end
end

function love.draw()
    love.graphics.print("Current Score: " .. math.floor(currentScore), 10,
    love.graphics.print("High Score: " .. math.floor(saveData.highScore),
    love.graphics.print("Player: " .. saveData.playerName, 10, 50)
    love.graphics.print("Level: " .. saveData.level, 10, 70)

    love.graphics.print("S: Save game", 10, 120)
    love.graphics.print("L: Load game", 10, 140)
    love.graphics.print("R: Reset save", 10, 160)
end
```

Configuration Files

```lua
local config = {
    masterVolume = 1.0,
    musicVolume = 0.8,
    sfxVolume = 1.0,
    fullscreen = false,
    vsync = true,
    controls = {
        up = "w",
        down = "s",
        left = "a",
        right = "d",
        jump = "space"
    }
}
```

```lua
function loadConfig()
    if love.filesystem.getInfo("config.lua") then
        local chunk = love.filesystem.load("config.lua")
        if chunk then
            local loadedConfig = chunk()
            -- Merge with defaults
            for k, v in pairs(loadedConfig) do
                config[k] = v
            end
        end
    end
end

function saveConfig()
    local configString = "return {\n"
    for k, v in pairs(config) do
        if type(v) == "table" then
            configString = configString .. "  " .. k .. " = {\n"
            for k2, v2 in pairs(v) do
                configString = configString .. "    " .. k2 .. " = \"" ..
            end
            configString = configString .. "  },\n"
        elseif type(v) == "string" then
            configString = configString .. "  " .. k .. " = \"" .. v .. "\
        else
            configString = configString .. "  " .. k .. " = " .. tostring(
        end
    end
    configString = configString .. "}"

    love.filesystem.write("config.lua", configString)
end

function love.load()
    loadConfig()
    love.audio.setVolume(config.masterVolume)
    love.window.setFullscreen(config.fullscreen)
end

function love.keypressed(key)
    if key == "f" then
        config.fullscreen = not config.fullscreen
        love.window.setFullscreen(config.fullscreen)
    elseif key == "=" then
        config.masterVolume = math.min(1.0, config.masterVolume + 0.1)
        love.audio.setVolume(config.masterVolume)
    elseif key == "-" then
        config.masterVolume = math.max(0.0, config.masterVolume - 0.1)
        love.audio.setVolume(config.masterVolume)
    elseif key == "s" and love.keyboard.isDown("lctrl") then
        saveConfig()
    end
end

function love.draw()
    love.graphics.print("Master Volume: " .. math.floor(config.masterVolum
    love.graphics.print("Fullscreen: " .. tostring(config.fullscreen), 10,

    love.graphics.print("F: Toggle fullscreen", 10, 80)
    love.graphics.print("+/-: Adjust volume", 10, 100)
    love.graphics.print("Ctrl+S: Save config", 10, 120)
end
```

Advanced Graphics

Shaders

```lua
local shader
local time = 0

function love.load()
    -- Create a simple wave shader
    local shaderCode = [[
        uniform float time;
        uniform vec2 screenSize;

        vec4 effect(vec4 color, Image texture, vec2 texture_coords, vec2 s
            vec2 uv = screen_coords / screenSize;

            // Create wave effect
            float wave = sin(uv.x * 10.0 + time * 2.0) * 0.1;
            uv.y += wave;

            // Color cycling
            vec3 col = vec3(
                0.5 + 0.5 * sin(time + uv.x * 3.0),
                0.5 + 0.5 * sin(time + uv.y * 3.0 + 2.0),
                0.5 + 0.5 * sin(time + uv.x * 3.0 + 4.0)
            );

            return vec4(col, 1.0) * color;
        }
    ]]

    shader = love.graphics.newShader(shaderCode)
end

function love.update(dt)
    time = time + dt
end

function love.draw()
    -- Use shader
    love.graphics.setShader(shader)
    shader:send("time", time)
    shader:send("screenSize", {love.graphics.getWidth(), love.graphics.get

    -- Draw something with the shader
    love.graphics.rectangle("fill", 0, 0, 800, 600)

    -- Reset shader
    love.graphics.setShader()

    love.graphics.setColor(1, 1, 1)
    love.graphics.print("Animated shader background", 10, 10)
end
```

Canvas and Render Targets

```lua
local canvas
local particles = {}

function love.load()
    canvas = love.graphics.newCanvas(800, 600)

    -- Create some particles
    for i = 1, 50 do
        table.insert(particles, {
            x = math.random(800),
            y = math.random(600),
            vx = math.random(-100, 100),
            vy = math.random(-100, 100),
            life = 1.0,
            decay = math.random(0.5, 2.0)
        })
    end
end

function love.update(dt)
    for i = #particles, 1, -1 do
        local p = particles[i]
        p.x = p.x + p.vx * dt
        p.y = p.y + p.vy * dt
        p.life = p.life - p.decay * dt

        -- Wrap around screen
        if p.x < 0 then p.x = 800 end
        if p.x > 800 then p.x = 0 end
        if p.y < 0 then p.y = 600 end
        if p.y > 600 then p.y = 0 end

        if p.life <= 0 then
            table.remove(particles, i)
            -- Add new particle
            table.insert(particles, {
                x = math.random(800),
                y = math.random(600),
                vx = math.random(-100, 100),
                vy = math.random(-100, 100),
                life = 1.0,
                decay = math.random(0.5, 2.0)
            })
        end
    end
end

function love.draw()
    -- Draw to canvas with additive blending
    love.graphics.setCanvas(canvas)
    love.graphics.setBlendMode("add")

    for _, p in ipairs(particles) do
        love.graphics.setColor(1, p.life, p.life * 0.5, p.life * 0.1)
        love.graphics.circle("fill", p.x, p.y, 10)
    end

    -- Reset canvas and blend mode
    love.graphics.setCanvas()
    love.graphics.setBlendMode("alpha")

    -- Draw canvas to screen
    love.graphics.setColor(1, 1, 1, 1)
    love.graphics.draw(canvas)

    love.graphics.print("Particle trail effect using canvas", 10, 10)
end
```

Lighting System

```lua
local lightCanvas, shadowCanvas
local lights = {}
local walls = {}

function love.load()
    lightCanvas = love.graphics.newCanvas(800, 600)
    shadowCanvas = love.graphics.newCanvas(800, 600)

    -- Add some lights
    table.insert(lights, {x = 200, y = 200, radius = 150, r = 1, g = 1, b
    table.insert(lights, {x = 600, y = 400, radius = 200, r = 0.5, g = 0.5

    -- Add some walls
    table.insert(walls, {x = 300, y = 150, width = 20, height = 200})
    table.insert(walls, {x = 500, y = 300, width = 150, height = 20})
end

function love.update(dt)
    -- Move first light with mouse
    lights[1].x, lights[1].y = love.mouse.getPosition()
end

function love.draw()
    -- Clear light canvas
    love.graphics.setCanvas(lightCanvas)
    love.graphics.clear(0.1, 0.1, 0.2, 1) -- Dark ambient

    -- Draw lights
    love.graphics.setBlendMode("add")
    for _, light in ipairs(lights) do
        local gradient = love.graphics.newMesh({
            {light.x, light.y, 0, 0, light.r, light.g, light.b, 1},
            {light.x - light.radius, light.y - light.radius, 0, 0, light.r
            {light.x + light.radius, light.y - light.radius, 0, 0, light.r
            {light.x + light.radius, light.y + light.radius, 0, 0, light.r
            {light.x - light.radius, light.y + light.radius, 0, 0, light.r
        }, "fan")
        love.graphics.draw(gradient)
    end

    love.graphics.setBlendMode("alpha")
    love.graphics.setCanvas()

    -- Draw scene
    love.graphics.setColor(1, 1, 1)
    love.graphics.draw(lightCanvas)

    -- Draw walls
    love.graphics.setColor(0.3, 0.3, 0.3)
    for _, wall in ipairs(walls) do
        love.graphics.rectangle("fill", wall.x, wall.y, wall.width, wall.h
    end

    love.graphics.setColor(1, 1, 1)
    love.graphics.print("Move mouse to control yellow light", 10, 10)
end
```

Output: A lighting system with multiple colored lights casting illumination over a dark scene with walls.

Performance Optimization

Object Pooling

```lua
local bulletPool = {}
local activeBullets = {}
local poolSize = 100

function love.load()
    -- Pre-create bullet objects
    for i = 1, poolSize do
        table.insert(bulletPool, {
            x = 0, y = 0,
            vx = 0, vy = 0,
            active = false,
            life = 0
        })
    end
end

function getBullet()
    for _, bullet in ipairs(bulletPool) do
        if not bullet.active then
            return bullet
        end
    end
    return nil -- Pool exhausted
end

function fireBullet(x, y, vx, vy)
    local bullet = getBullet()
    if bullet then
        bullet.x = x
        bullet.y = y
        bullet.vx = vx
        bullet.vy = vy
        bullet.active = true
        bullet.life = 3.0 -- 3 seconds
        table.insert(activeBullets, bullet)
    end
end

function love.update(dt)
    -- Fire bullets on mouse click
    if love.mouse.isDown(1) then
        local mx, my = love.mouse.getPosition()
        fireBullet(400, 300, (mx - 400) * 2, (my - 300) * 2)
    end

    -- Update active bullets
    for i = #activeBullets, 1, -1 do
        local bullet = activeBullets[i]
        bullet.x = bullet.x + bullet.vx * dt
        bullet.y = bullet.y + bullet.vy * dt
        bullet.life = bullet.life - dt

        if bullet.life <= 0 or bullet.x < 0 or bullet.x > 800 or bullet.y
            bullet.active = false
            table.remove(activeBullets, i)
        end
    end
end

function love.draw()
    -- Draw bullets
    love.graphics.setColor(1, 1, 0)
    for _, bullet in ipairs(activeBullets) do
        love.graphics.circle("fill", bullet.x, bullet.y, 3)
    end

    love.graphics.setColor(1, 1, 1)
    love.graphics.print("Active bullets: " .. #activeBullets, 10, 10)
    love.graphics.print("Pool usage: " .. #activeBullets .. "/" .. poolSiz
    love.graphics.print("Hold mouse to fire bullets", 10, 50)
```

```
    love.graphics.print("Hold mouse to fire bullets", 10, 90)
end
```

Spatial Partitioning

```
local grid = {}
local gridSize = 64
local gridWidth = math.ceil(800 / gridSize)
local gridHeight = math.ceil(600 / gridSize)
local entities = {}

function love.load()
    -- Initialize grid
    for x = 1, gridWidth do
        grid[x] = {}
        for y = 1, gridHeight do
            grid[x][y] = {}
        end
    end

    -- Create entities
    for i = 1, 100 do
        table.insert(entities, {
            x = math.random(800),
            y = math.random(600),
            vx = math.random(-50, 50),
            vy = math.random(-50, 50),
            radius = 5,
            gridX = 0, gridY = 0
        })
    end
end

function updateGrid()
    -- Clear grid
    for x = 1, gridWidth do
        for y = 1, gridHeight do
            grid[x][y] = {}
        end
    end

    -- Add entities to grid
    for _, entity in ipairs(entities) do
        local gx = math.max(1, math.min(gridWidth, math.ceil(entity.x / gr
        local gy = math.max(1, math.min(gridHeight, math.ceil(entity.y / g
        entity.gridX, entity.gridY = gx, gy
        table.insert(grid[gx][gy], entity)
    end
end

function love.update(dt)
    -- Update entities
    for _, entity in ipairs(entities) do
        entity.x = entity.x + entity.vx * dt
        entity.y = entity.y + entity.vy * dt

        -- Bounce off walls
        if entity.x < 0 or entity.x > 800 then
            entity.vx = -entity.vx
        end
        if entity.y < 0 or entity.y > 600 then
            entity.vy = -entity.vy
        end
    end

    updateGrid()
end

function love.draw()
```

```
function love.draw()
    -- Draw grid
    love.graphics.setColor(0.2, 0.2, 0.2)
    for x = 1, gridWidth do
        love.graphics.line(x * gridSize, 0, x * gridSize, 600)
    end
    for y = 1, gridHeight do
        love.graphics.line(0, y * gridSize, 800, y * gridSize)
    end

    -- Draw entities
    love.graphics.setColor(1, 1, 1)
    for _, entity in ipairs(entities) do
        love.graphics.circle("fill", entity.x, entity.y, entity.radius)
    end

    love.graphics.print("Spatial partitioning with " .. #entities .. " ent
end
```

Building and Distribution

Creating a .love File

```lua
-- build.lua - Build script
local lfs = require("lfs")

function copyFile(src, dest)
    local srcFile = io.open(src, "rb")
    local destFile = io.open(dest, "wb")

    if srcFile and destFile then
        destFile:write(srcFile:read("*all"))
        srcFile:close()
        destFile:close()
        return true
    end
    return false
end

function buildLoveFile()
    print("Building .love file...")

    -- Create build directory
    lfs.mkdir("build")

    -- Copy game files
    local files = {
        "main.lua",
        "conf.lua",
        -- Add more files as needed
    }

    for _, file in ipairs(files) do
        copyFile(file, "build/" .. file)
    end

    -- Copy assets directory
    -- (This would require recursive directory copying)

    -- Create .love file (this is just a renamed .zip)
    os.execute("cd build && zip -r ../MyGame.love .")

    print("Build complete: MyGame.love")
end

buildLoveFile()
```

Cross-Platform Configuration

```lua
-- conf.lua - Production configuration
function love.conf(t)
    t.identity = "MyAwesomeGame"
    t.version = "11.4"
    t.console = false -- Disable console on Windows

    t.window.title = "My Awesome Game"
    t.window.icon = "assets/images/icon.png"
    t.window.width = 800
    t.window.height = 600
    t.window.resizable = false
    t.window.minwidth = 800
    t.window.minheight = 600
    t.window.fullscreen = false
    t.window.fullscreentype = "desktop"
    t.window.vsync = 1
    t.window.msaa = 0
    t.window.display = 1
    t.window.highdpi = false
    t.window.usedpiscale = true
    t.window.borderless = false
    t.window.centered = true

    t.modules.audio = true
    t.modules.event = true
    t.modules.graphics = true
    t.modules.image = true
    t.modules.joystick = true
    t.modules.keyboard = true
    t.modules.math = true
    t.modules.mouse = true
    t.modules.physics = true
    t.modules.sound = true
    t.modules.system = true
    t.modules.thread = true
    t.modules.timer = true
    t.modules.touch = true
    t.modules.video = false -- Disable if not used
    t.modules.window = true
end
```

Complete Game Example

Here's a complete Asteroids-style game that demonstrates many of the concepts covered:

```lua
-- Asteroids Game
local gameState = "menu"
local score = 0
local lives = 3
local level = 1

-- Player
local player = {
    x = 400, y = 300,
    vx = 0, vy = 0,
    angle = 0,
    thrust = false,
    size = 8
}

-- Game objects
local bullets = {}
local asteroids = {}
local particles = {}

-- Constants
local THRUST_POWER = 300
```

```lua
local ROTATION_SPEED = 5
local BULLET_SPEED = 400
local FRICTION = 0.98

function love.load()
    love.window.setTitle("Asteroids - Love2D")
    math.randomseed(os.time())
    resetGame()
end

function resetGame()
    player.x, player.y = 400, 300
    player.vx, player.vy = 0, 0
    player.angle = 0

    bullets = {}
    asteroids = {}
    particles = {}

    createAsteroids(4 + level)
end

function createAsteroids(count)
    for i = 1, count do
        local asteroid = {
            x = math.random(100, 700),
            y = math.random(100, 500),
            vx = math.random(-100, 100),
            vy = math.random(-100, 100),
            size = math.random(20, 40),
            angle = 0,
            spin = math.random(-3, 3)
        }

        -- Make sure asteroid doesn't spawn on player
        local dx = asteroid.x - player.x
        local dy = asteroid.y - player.y
        if math.sqrt(dx*dx + dy*dy) < 100 then
            asteroid.x = asteroid.x + 200
            asteroid.y = asteroid.y + 200
        end

        table.insert(asteroids, asteroid)
    end
end

function love.update(dt)
    if gameState == "game" then
        updateGame(dt)
    end
end

function updateGame(dt)
    -- Player input
    if love.keyboard.isDown("left") then
        player.angle = player.angle - ROTATION_SPEED * dt
    end
    if love.keyboard.isDown("right") then
        player.angle = player.angle + ROTATION_SPEED * dt
    end

    player.thrust = love.keyboard.isDown("up")

    -- Player movement
    if player.thrust then
        local thrustX = math.sin(player.angle) * THRUST_POWER * dt
        local thrustY = -math.cos(player.angle) * THRUST_POWER * dt
        player.vx = player.vx + thrustX
        player.vy = player.vy + thrustY
```

```lua
        -- Add thrust particles
        for i = 1, 3 do
            addParticle(
                player.x - math.sin(player.angle) * 15,
                player.y + math.cos(player.angle) * 15,
                -math.sin(player.angle) * 100 + math.random(-20, 20),
                math.cos(player.angle) * 100 + math.random(-20, 20),
                0.5
            )
        end
    end

    -- Apply friction
    player.vx = player.vx * FRICTION
    player.vy = player.vy * FRICTION

    -- Update player position
    player.x = player.x + player.vx * dt
    player.y = player.y + player.vy * dt

    -- Wrap around screen
    wrapPosition(player)

    -- Update bullets
    for i = #bullets, 1, -1 do
        local bullet = bullets[i]
        bullet.x = bullet.x + bullet.vx * dt
        bullet.y = bullet.y + bullet.vy * dt
        bullet.life = bullet.life - dt

        wrapPosition(bullet)

        if bullet.life <= 0 then
            table.remove(bullets, i)
        end
    end

    -- Update asteroids
    for _, asteroid in ipairs(asteroids) do
        asteroid.x = asteroid.x + asteroid.vx * dt
        asteroid.y = asteroid.y + asteroid.vy * dt
        asteroid.angle = asteroid.angle + asteroid.spin * dt
        wrapPosition(asteroid)
    end

    -- Update particles
    for i = #particles, 1, -1 do
        local p = particles[i]
        p.x = p.x + p.vx * dt
        p.y = p.y + p.vy * dt
        p.life = p.life - dt

        if p.life <= 0 then
            table.remove(particles, i)
        end
    end

    -- Collision detection
    checkCollisions()

    -- Check win condition
    if #asteroids == 0 then
        level = level + 1
        resetGame()
    end
end

function wrapPosition(obj)
    if obj.x < 0 then obj.x = 800 end
```

```lua
    if obj.x > 800 then obj.x = 0 end
    if obj.y < 0 then obj.y = 600 end
    if obj.y > 600 then obj.y = 0 end
end

function checkCollisions()
    -- Bullet-asteroid collisions
    for bi = #bullets, 1, -1 do
        local bullet = bullets[bi]
        for ai = #asteroids, 1, -1 do
            local asteroid = asteroids[ai]
            local dx = bullet.x - asteroid.x
            local dy = bullet.y - asteroid.y
            local distance = math.sqrt(dx*dx + dy*dy)

            if distance < asteroid.size then
                -- Remove bullet and asteroid
                table.remove(bullets, bi)
                table.remove(asteroids, ai)

                score = score + 100

                -- Create explosion particles
                for i = 1, 10 do
                    addParticle(
                        asteroid.x, asteroid.y,
                        math.random(-100, 100),
                        math.random(-100, 100),
                        1.0
                    )
                end

                -- Split large asteroids
                if asteroid.size > 15 then
                    for i = 1, 2 do
                        local newAsteroid = {
                            x = asteroid.x,
                            y = asteroid.y,
                            vx = math.random(-150, 150),
                            vy = math.random(-150, 150),
                            size = asteroid.size * 0.6,
                            angle = 0,
                            spin = math.random(-5, 5)
                        }
                        table.insert(asteroids, newAsteroid)
                    end
                end
                break
            end
        end
    end

    -- Player-asteroid collisions
    for _, asteroid in ipairs(asteroids) do
        local dx = player.x - asteroid.x
        local dy = player.y - asteroid.y
        local distance = math.sqrt(dx*dx + dy*dy)

        if distance < asteroid.size + player.size then
            lives = lives - 1

            -- Create explosion
            for i = 1, 20 do
                addParticle(
                    player.x, player.y,
                    math.random(-200, 200),
                    math.random(-200, 200),
                    2.0
                )
            end
```

```lua
            end

            if lives <= 0 then
                gameState = "gameover"
            else
                -- Reset player position
                player.x, player.y = 400, 300
                player.vx, player.vy = 0, 0
            end
            break
        end
    end
end

function addParticle(x, y, vx, vy, life)
    table.insert(particles, {
        x = x, y = y,
        vx = vx, vy = vy,
        life = life
    })
end

function love.keypressed(key)
    if gameState == "menu" then
        if key == "space" then
            gameState = "game"
            score = 0
            lives = 3
            level = 1
            resetGame()
        elseif key == "escape" then
            love.event.quit()
        end
    elseif gameState == "game" then
        if key == "space" then
            -- Fire bullet
            local bullet = {
                x = player.x + math.sin(player.angle) * 15,
                y = player.y - math.cos(player.angle) * 15,
                vx = math.sin(player.angle) * BULLET_SPEED,
                vy = -math.cos(player.angle) * BULLET_SPEED,
                life = 2.0
            }
            table.insert(bullets, bullet)
        elseif key == "escape" then
            gameState = "menu"
        end
    elseif gameState == "gameover" then
        if key == "space" then
            gameState = "menu"
        end
    end
end

function love.draw()
    if gameState == "menu" then
        drawMenu()
    elseif gameState == "game" then
        drawGame()
    elseif gameState == "gameover" then
        drawGameOver()
    end
end

function drawMenu()
    love.graphics.setColor(1, 1, 1)
    love.graphics.print("ASTEROIDS", 300, 200, 0, 3, 3)
    love.graphics.print("Press SPACE to start", 290, 300)
    love.graphics.print("ESC to quit", 330, 350)
    love.graphics.print("Controls:", 50, 450)
```

```lua
    love.graphics.print("Left/Right arrows: Rotate", 50, 470)
    love.graphics.print("Up arrow: Thrust", 50, 490)
    love.graphics.print("Space: Fire", 50, 510)
end

function drawGame()
    love.graphics.setColor(1, 1, 1)

    -- Draw player
    if player.thrust then
        love.graphics.setColor(1, 0.5, 0) -- Orange when thrusting
    end

    local x1 = player.x + math.sin(player.angle) * player.size
    local y1 = player.y - math.cos(player.angle) * player.size
    local x2 = player.x - math.sin(player.angle - 2.5) * player.size
    local y2 = player.y + math.cos(player.angle - 2.5) * player.size
    local x3 = player.x - math.sin(player.angle + 2.5) * player.size
    local y3 = player.y + math.cos(player.angle + 2.5) * player.size

    love.graphics.polygon("line", x1, y1, x2, y2, x3, y3)

    love.graphics.setColor(1, 1, 1)

    -- Draw bullets
    for _, bullet in ipairs(bullets) do
        love.graphics.circle("fill", bullet.x, bullet.y, 2)
    end

    -- Draw asteroids
    for _, asteroid in ipairs(asteroids) do
        love.graphics.push()
        love.graphics.translate(asteroid.x, asteroid.y)
        love.graphics.rotate(asteroid.angle)
        love.graphics.circle("line", 0, 0, asteroid.size)
        love.graphics.pop()
    end

    -- Draw particles
    love.graphics.setColor(1, 0.5, 0)
    for _, p in ipairs(particles) do
        local alpha = p.life / 2.0
        love.graphics.setColor(1, 0.5, 0, alpha)
        love.graphics.circle("fill", p.x, p.y, 2)
    end

    love.graphics.setColor(1, 1, 1)

    -- Draw UI
    love.graphics.print("Score: " .. score, 10, 10)
    love.graphics.print("Lives: " .. lives, 10, 30)
    love.graphics.print("Level: " .. level, 10, 50)
end

function drawGameOver()
    love.graphics.setColor(1, 0, 0)
    love.graphics.print("GAME OVER", 280, 250, 0, 3, 3)
    love.graphics.setColor(1, 1, 1)
    love.graphics.print("Final Score: " .. score, 320, 320)
    love.graphics.print("Press SPACE to return to menu", 250, 370)
end
```

Output: A complete Asteroids game with player ship, asteroids, bullets, collision detection, particle effects, and multiple game states.

Conclusion

This tutorial has covered the essential aspects of Love2D game development, from basic setup to advanced techniques. Love2D's simplicity combined with Lua's flexibility makes it an excellent choice for 2D game development.

Key Takeaways:

1. **Start Simple**: Begin with basic shapes and movement before adding complexity
2. **Use Game States**: Organize your game with proper state management
3. **Optimize Early**: Use techniques like object pooling and spatial partitioning
4. **Plan Your Architecture** : Structure your code for maintainability
5. **Test Frequently**: Run your game often during development
6. **Learn from Examples**: Study existing Love2D games and code

Next Steps:

- Experiment with different genres (platformers, RPGs, puzzle games)
- Learn advanced graphics techniques (shaders, lighting)
- Explore multiplayer networking with libraries like sock.lua
- Study game design principles and player psychology
- Join the Love2D community forums and Discord

Resources:

- Official Love2D Documentation
- Love2D Community
- Awesome Love2D
- LÖVE Tutorials

Happy game development with Love2D!