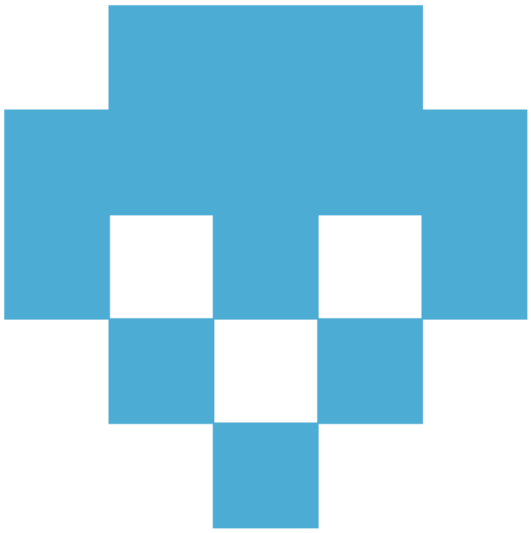


Pengenalan Bahasa Lua dan Framework LÖVE

RENDERTREE

2024

https://github.com/rendertree

Full Source Code:

https://github.com/rendertree/simtut/blob/main/demo/main.lua

ABSTRAKSI

Buku "Pengenalan Bahasa Lua dan Framework LÖVE" merupakan buku yang dibuat untuk memperkenalkan pembaca dengan bahasa pemrograman Lua dan framework LÖVE. Buku ini menyajikan konsep dasar Lua, termasuk sintaksis, variabel, kontrol alur, dan fungsi, dengan pendekatan yang mudah dipahami dan dipraktekan.

Selain itu, buku ini juga mengeksplorasi penggunaan framework LÖVE, sebuah toolkit yang populer untuk pengembangan game 2D. Pembaca akan dipandu langkah demi langkah dalam membuat game sederhana, mulai dari instalasi hingga contoh-contoh pengkodean menggunakan framework LÖVE.

Dengan dilengkapi contoh kode yang jelas dan latihan praktis, buku ini cocok bagi pemula yang ingin memulai mencoba bahasa Lua. Di akhir buku, pembaca diharapkan memiliki pemahaman tentang dasar-dasar Lua dan bagaimana mengaplikasikannya dalam membuat game sederhana menggunakan LÖVE.

Daftar Isi

Bab 1: Apa Itu Bahasa Pemrograman dan Framework?

Bahasa Pemrograman

Bahasa pemrograman adalah sistematisasi aturan sintaks dan semantik yang digunakan untuk menulis kode komputer. Setiap bahasa pemrograman memiliki karakteristik dan kegunaannya sendiri dalam mengekspresikan instruksi kepada komputer. Berikut ini beberapa contoh bahasa pemrograman yang populer:

- **Python:** Python adalah bahasa pemrograman tingkat tinggi yang terkenal dengan sintaks yang sederhana dan mudah dipahami. Python digunakan dalam berbagai bidang seperti pengembangan web, ilmu data, kecerdasan buatan, dan pengembangan perangkat lunak umum lainnya.
- **C#**: C# (C-Sharp) adalah bahasa pemrograman yang serbaguna dan dikembangkan oleh Microsoft. Bahasa ini banyak digunakan untuk pengembangan aplikasi berbasis Windows, aplikasi web, dan game menggunakan Unity. C# dikenal karena sintaksisnya yang jelas, dukungan kuat untuk pemrograman berorientasi objek, dan integrasi yang erat dengan platform .NET, yang memungkinkan pengembangan perangkat lunak yang efisien dan terstruktur.
- **JavaScript:** JavaScript adalah bahasa pemrograman yang digunakan untuk mengembangkan aplikasi web interaktif di sisi klien (front-end). Selain itu, JavaScript juga digunakan dengan Node.js untuk pengembangan server-side.
- **Lua:** Lua adalah bahasa skrip ringan yang sering digunakan sebagai bahasa skrip dalam aplikasi, game, dan sistem tertanam. Lua terkenal dengan kecepatan eksekusi yang tinggi dan ukurannya yang kecil.

Framework (Kerangka Kerja)

Framework atau kerangka kerja adalah seperangkat alat, aturan, dan komponen yang disusun untuk memfasilitasi pengembangan perangkat lunak. Framework menyediakan struktur kerja yang sudah jadi untuk membangun aplikasi dengan cepat dan efisien. Berikut adalah beberapa contoh framework yang umum digunakan:

- **Django (Python):** Django adalah framework Python yang kuat untuk pengembangan aplikasi web. Django menyediakan kerangka kerja MVC (Model-View-Controller) yang memudahkan pengembangan aplikasi web dengan fitur-fitur seperti manajemen basis data, keamanan, dan admin panel bawaan.

- **LÖVE (Lua):** LÖVE adalah framework permainan (game framework) yang menggunakan bahasa Lua untuk pengembangan game. LÖVE menyediakan fungsi-fungsi dan alat-alat untuk memudahkan pembuatan game dengan fokus pada kreativitas pengembang.

Setiap framework memiliki pendekatan, fitur, dan kelebihan yang berbeda-beda, tergantung pada jenis aplikasi yang sedang dikembangkan dan preferensi pengembangnya. Penggunaan framework yang tepat dapat mempercepat pengembangan, meningkatkan kualitas, dan memfasilitasi pengelolaan kode yang lebih baik dalam proyek pengembangan perangkat lunak atau game.

Pengenalan Bahasa Lua

Lua adalah bahasa pemrograman ringan, kuat, dan fleksibel yang dirancang untuk integrasi yang mudah ke dalam aplikasi lain. Lua dikembangkan pada awal 1990-an di Brasil sebagai bahasa skrip untuk memperkaya perangkat lunak simulasi ekstensi. Sejak itu, Lua telah berkembang menjadi salah satu bahasa skrip yang paling populer dan serbaguna di dunia.

Berikut adalah beberapa karakteristik utama dari Lua:

1. **Ringan dan Cepat:** Lua dikembangkan dengan fokus pada kinerja dan efisiensi. Interpreter Lua memiliki ukuran yang kecil (kurang dari 300 KB dalam beberapa implementasi), membuatnya ideal untuk aplikasi dengan keterbatasan sumber daya.
2. **Dinamis dan Terinterpretasi:** Lua adalah bahasa pemrograman dinamis yang diinterpretasi. Ini berarti kode Lua dieksekusi baris per baris oleh interpreter, tanpa perlu kompilasi sebelumnya.
3. **Mudah Dipelajari dan Dipahami:** Sintaks Lua sederhana dan bersih, membuatnya mudah dipelajari oleh pemula dan dipahami oleh pengembang yang berpengalaman. Lua memiliki gaya sintaks yang mirip dengan bahasa pemrograman lain seperti Python.
4. **Fleksibel dan Kuat:** Lua dirancang untuk menjadi fleksibel dan mudah diintegrasikan ke dalam berbagai aplikasi. Bahasa ini mendukung pemrograman prosedural, fungsional, dan berorientasi objek secara terbatas.
5. **Portabel dan Platform-Independent:** Kode Lua dapat dijalankan di berbagai platform dan sistem operasi tanpa perubahan besar. Ini membuatnya sangat berguna untuk pengembangan perangkat lunak lintas-platform.
6. **Dukungan Komunitas yang Luas:** Meskipun ukurannya yang relatif kecil, Lua memiliki komunitas pengguna yang aktif dan dukungan yang solid dari berbagai perangkat lunak dan aplikasi terkenal.

Penggunaan Lua

Lua digunakan di berbagai bidang, termasuk:

- **Game Development:** Lua sangat populer dalam industri game sebagai bahasa skrip untuk mengatur logika permainan, AI, animasi, dan interaksi dalam game. Banyak game besar menggunakan Lua untuk scripting, seperti World of Warcraft, Angry Birds, dan Civilization.

- **Embedded Systems:** Karena ukurannya yang kecil dan kinerjanya yang tinggi, Lua sering digunakan dalam sistem tertanam (embedded systems) dan perangkat lunak yang memerlukan penanganan sumber daya yang efisien.
- **Aplikasi Skrip dan Konfigurasi:** Lua digunakan sebagai bahasa skrip dalam berbagai aplikasi untuk memungkinkan pengguna untuk mengotomatisasi tugas atau menyesuaikan perilaku aplikasi.
- **Aplikasi Web dan Server-side:** Lua digunakan dalam pengembangan aplikasi web dengan bantuan framework seperti OpenResty, yang menyediakan integrasi Lua dengan server web Nginx untuk pengembangan web dinamis.

Pengenalan Framework LÖVE

LÖVE (atau disebut juga LOVE) adalah sebuah framework permainan (game framework) yang dirancang untuk memudahkan pengembangan game dengan menggunakan bahasa pemrograman Lua. Framework ini dikembangkan untuk menyediakan alat dan fitur yang diperlukan untuk membuat game secara efisien, dengan fokus pada kreativitas dan fleksibilitas pengembang.

Fitur Utama LÖVE:

1. **Sederhana dan Mudah Dipelajari:** LÖVE didesain agar mudah dipelajari oleh pemula sekalipun dalam pengembangan game. Strukturnya yang sederhana memungkinkan pengembang untuk fokus pada pengembangan gameplay dan visual tanpa harus memikirkan aspek teknis yang kompleks.
2. **Dukungan Multiplatform:** LÖVE mendukung beberapa platform utama seperti Windows, macOS, Linux, dan juga Android melalui pengembangan eksperimental. Ini memungkinkan pengembang untuk membuat game yang dapat dijalankan di berbagai sistem operasi tanpa banyak modifikasi.
3. **Fungsi-fungsi yang tersedia:**
    - **Grafika:** LÖVE menyediakan API untuk menggambar berbagai objek grafis seperti gambar, teksur, dan geometri.
    - **Suara dan Musik:** Mendukung pemutaran suara dan musik dalam game dengan dukungan untuk berbagai format audio.
    - **Input Pengguna:** Memungkinkan pengembang untuk menangani input dari keyboard, mouse, dan joystick.

- **Fisika:** Meskipun tidak termasuk dalam inti LÖVE, terdapat library eksternal seperti love.physics yang menyediakan simulasi fisika untuk game-game yang membutuhkannya.
- **Manajemen Jendela:** LÖVE menyediakan dukungan untuk manajemen jendela aplikasi, termasuk pengaturan resolusi dan mode layar penuh.
-

4. **Komunitas dan Ekosistem yang Aktif:** LÖVE memiliki komunitas pengembang yang aktif yang memberikan dukungan, sumber daya, dan contoh-contoh kode. Banyaknya tutorial dan dokumentasi juga memudahkan pemula untuk memulai belajar dan mengembangkan game dengan LÖVE.

Bab 2: Dasar-Dasar Bahasa Lua

Variabel

Di bahasa pemrograman Lua, variabel adalah nama yang digunakan untuk menyimpan nilai atau referensi ke nilai. Variabel digunakan untuk menyimpan data yang dapat diubah atau diakses selama eksekusi program. Variabel di Lua dapat dikategorikan menjadi dua jenis utama: variabel lokal dan variabel global.

Variabel dalam Lua bisa berupa nama yang dimulai dengan huruf atau garis bawah (_) dan dapat terdiri dari huruf, angka, dan garis bawah. Penulisan nama variabel bersifat case-sensitive, yang berarti nama, Nama, dan NAMA dianggap sebagai variabel yang berbeda.

Variabel Lokal

Variabel lokal adalah variabel yang dideklarasikan dalam suatu blok kode, seperti di dalam fungsi, dan hanya dapat diakses dalam blok tersebut. Variabel ini tidak dapat diakses dari luar blok di mana mereka dideklarasikan. Penggunaan variabel lokal membantu dalam mengelola ruang lingkup variabel dan menghindari konflik nama variabel.

Variabel Global

Variabel global adalah variabel yang dapat diakses dari mana saja dalam program Lua. Variabel ini dideklarasikan tanpa menggunakan kata kunci local. Karena dapat diakses dari mana saja, variabel global memiliki potensi untuk menyebabkan konflik nama dan kesalahan yang sulit dilacak jika tidak digunakan dengan hati-hati.

Perbedaan Utama

1. **Lingkup**:
    - **Lokal**: Hanya dapat diakses di dalam blok di mana variabel tersebut dideklarasikan.
    - **Global**: Dapat diakses dari mana saja dalam program.
2. **Deklarasi**:
    - **Lokal**: Menggunakan kata kunci local.
    - **Global**: Tidak menggunakan kata kunci local.
3. **Penggunaan**:
    - **Lokal**: Digunakan untuk menghindari konflik nama dan menjaga kode tetap terorganisir.

- **Global**: Digunakan ketika nilai perlu diakses di berbagai bagian program, tetapi harus digunakan dengan hati-hati.

Contoh penulisan variabel lokal dan global:

Variabel lokal: Untuk mendeklarasikan variabel lokal, Anda tidak perlu menentukan tipe datanya secara eksplisit. Cukup beri nama variabel dan berikan nilai kepadanya. Contoh:

```
local ukuran = 25
local warna = "Merah"
local panjang = 25.5
```

Variabel global: Sama seperti variabel lokal, untuk mendeklarasikan variabel global, Anda tidak perlu menentukan tipe datanya secara eksplisit. Cukup beri nama variabel dan berikan nilai kepadanya. Contoh:

```
ukuran = 25
warna = "Merah"
panjang = 25.5
```

Dari kedua contoh di atas:

- ukuran adalah variabel bertipe bilangan bulat (integer).
- warna adalah variabel bertipe string.
- panjang adalah variabel bertipe bilangan pecahan (floating point).

Operasi Aritmatika dalam Lua

Lua mendukung berbagai operasi aritmatika dasar yang umum digunakan dalam pemrograman:

- **Penjumlahan (+)**: Digunakan untuk menambahkan dua nilai.

  Contoh:

  ```
  local a = 10
  local b = 5
  local hasil = a + b  -- hasil akan menjadi 15
  ```

- **Pengurangan (-)**: Digunakan untuk mengurangi satu nilai dari nilai lain.

  Contoh:

  ```
  local a = 10
  local b = 5
  local hasil = a - b  -- hasil akan menjadi 5
  ```

- **Perkalian (*)**: Digunakan untuk mengalikan dua nilai.

  Contoh:

```
local a = 10
local b = 5
local hasil = a * b  -- hasil akan menjadi 50
```

- **Pembagian (/)**: Digunakan untuk membagi nilai pertama dengan nilai kedua.

  Contoh:

```
local a = 10
local b = 5
local hasil = a / b  -- hasil akan menjadi 2
```

- **Modulus (%)**: Digunakan untuk mendapatkan sisa hasil bagi dari pembagian dua nilai.

  Contoh:

```
local a = 10
local b = 3
local hasil = a % b  -- hasil akan menjadi 1 (10 dibagi 3 sisa 1)
```

- **Pangkat (^)**: Digunakan untuk menghitung nilai pangkat dari suatu bilangan.

  Contoh:

```
local a = 2
local b = 3
local hasil = a ^ b  -- hasil akan menjadi 8 (2 pangkat 3)
```

Penggunaan Variabel dan Operasi Aritmatika

Variabel dan operasi aritmatika dapat digunakan bersama-sama dalam ekspresi untuk melakukan perhitungan yang lebih kompleks:

```
local radius = 5
local luas = math.pi * radius * radius
print("Luas lingkaran dengan radius", radius, "adalah", luas)

local suhu_celcius = 30
local suhu_fahrenheit = (suhu_celcius * 9/5) + 32
print("Suhu dalam Celcius:", suhu_celcius, "dalam Fahrenheit:", suhu_fahrenheit)
```

Di sini, kita menggunakan variabel radius untuk menghitung luas lingkaran dan variabel suhu_celcius untuk menghitung konversi suhu ke Fahrenheit menggunakan operasi aritmatika.

Lua menyediakan dukungan yang kuat untuk manipulasi variabel dan operasi aritmatika, memungkinkan pengembang untuk melakukan berbagai jenis perhitungan dan manipulasi data dengan mudah dan efisien.

Statement dalam Lua

Statement dalam Lua adalah unit dasar dari eksekusi program. Lua mendukung berbagai jenis statement, termasuk:

1. **Pernyataan Atribusi (Assignment Statement)**: Digunakan untuk memberikan nilai ke variabel.

```
jumlah = 0
```

2. **Pernyataan Percabangan (Branching Statement)**: Digunakan untuk melakukan pengujian kondisi dan menjalankan blok kode tertentu berdasarkan hasil pengujian.

```
local jumlah = 0
while jumlah < 5 do
   print("Perulangan ke-" .. jumlah)
   jumlah = jumlah + 1
end
```

3. **Pernyataan Perulangan (Loop Statement)**: Digunakan untuk menjalankan blok kode berulang kali selama kondisi tertentu terpenuhi.

```
local jumlah = 0
while jumlah < 5 do
   print("Perulangan ke-" .. jumlah)
   jumlah = jumlah + 1
end
```

Scope Variabel dalam Lua

Variabel dalam Lua memiliki cakupan (scope) yang menentukan di mana variabel tersebut dapat diakses. Lua memiliki cakupan blok (block scope) yang artinya variabel yang dideklarasikan di dalam blok tertentu hanya dapat diakses di dalam blok itu. Contoh:

```
do
   local pesan = "Halo dunia!"
   print(pesan)  -- Output: Halo dunia!
end

print(pesan)  -- Akan menghasilkan error, karena variabel 'pesan' hanya dapat diakses
di dalam blok 'do'
```

Dengan memahami konsep variabel dan statement di Lua, Anda dapat membangun program yang lebih kompleks dan efisien. Penggunaan yang tepat dari variabel dan statement akan membantu Anda dalam mengorganisasi dan menjalankan logika program dengan baik.

Fungsi

Di Lua, fungsi adalah blok kode yang dapat dipanggil untuk mengeksekusi serangkaian instruksi tertentu. Fungsi digunakan untuk mengorganisasi kode, menghindari duplikasi kode, dan memungkinkan penggunaan kembali kode. Berikut adalah panduan lengkap untuk membuat dan menggunakan fungsi dalam bahasa Lua:

Mendefinisikan Fungsi

Anda dapat mendefinisikan fungsi di Lua menggunakan kata kunci function, diikuti dengan nama fungsi dan parameter yang diterima oleh fungsi (jika ada), di dalam sepasang tanda kurung. Berikut adalah contoh sederhana mendefinisikan fungsi:

```
-- Mendefinisikan fungsi tanpa parameter
function sapa()
    print("Halo, selamat datang!")
end

-- Memanggil fungsi
sapa()  -- Output: Halo, selamat datang!
```

Pada contoh tersebut, sapa adalah nama fungsi yang tidak memiliki parameter. Fungsi tersebut hanya mencetak pesan "Halo, selamat datang!" ke konsol saat dipanggil.

Fungsi dengan Parameter

Fungsi juga dapat menerima parameter, yang merupakan nilai yang diteruskan ke dalam fungsi saat dipanggil. Parameter didefinisikan di dalam tanda kurung setelah nama fungsi.

Contoh:

```
-- Fungsi dengan satu parameter
function sapa(nama)
    print("Halo, " .. nama .. "!")
end

-- Memanggil fungsi dengan memberikan nilai parameter
sapa("John")  -- Output: Halo, John!
sapa("Jane")  -- Output: Halo, Jane!
```

Pada contoh tersebut, nama adalah parameter dari fungsi sapa. Ketika fungsi dipanggil dengan nilai "John" atau "Jane" sebagai parameter, fungsi tersebut akan mencetak pesan sapaan sesuai dengan nilai parameter yang diterima.

Fungsi dengan Return Value

Fungsi dalam Lua dapat mengembalikan nilai menggunakan kata kunci return. Nilai yang dikembalikan dapat digunakan di tempat di mana fungsi dipanggil. Contoh:

```
-- Fungsi untuk menghitung luas persegi panjang
function luas_persegi_panjang(panjang, lebar)
    local luas = panjang * lebar
    return luas
end

-- Memanggil fungsi dan menyimpan nilai yang dikembalikan
local hasil = luas_persegi_panjang(5, 3)
print("Luas persegi panjang:", hasil)  -- Output: Luas persegi panjang: 15
```

Pada contoh tersebut, luas_persegi_panjang adalah fungsi yang menghitung luas persegi panjang berdasarkan panjang dan lebar yang diteruskan sebagai parameter. Fungsi tersebut mengembalikan nilai luas, yang kemudian dicetak sebagai output.

Fungsi Sebagai Nilai

Di Lua, fungsi juga dapat dianggap sebagai nilai yang dapat disimpan dalam variabel atau digunakan sebagai argumen untuk fungsi lain. Contoh:

```
-- Fungsi yang mengembalikan fungsi lain
function pembuat_sapaan(sapaan)
    return function(nama)
        print(sapaan .. ", " .. nama .. "!")
    end
end

-- Menggunakan fungsi sebagai nilai
local sapaHalo = pembuat_sapaan("Halo")
local sapaSelamatPagi = pembuatSapaan("Selamat pagi")

sapaHalo("John")  -- Output: Halo, John!
sapaSelamatPagi("Jane")  -- Output: Selamat pagi, Jane!
```

Pada contoh tersebut, pembuat_sapaan() adalah fungsi yang mengembalikan sebuah fungsi anonim (tanpa nama) yang menggunakan parameter nama untuk mencetak pesan sapaan yang diteruskan saat fungsi pembuat_sapaan() dipanggil.

Fungsi Rekursif

Lua mendukung rekursi, yaitu kemampuan sebuah fungsi memanggil dirinya sendiri. Contoh sederhana dari fungsi rekursif:

```
-- Fungsi rekursif untuk menghitung faktorial
function faktorial(n)
    if n == 0 then
        return 1
    else
        return n * faktorial(n - 1)
    end
end

-- Memanggil fungsi rekursif
local hasil = faktorial(5)
print("Hasil faktorial:", hasil)  -- Output: Hasil faktorial: 120
```

Pada contoh di atas, faktorial adalah fungsi rekursif yang menghitung faktorial dari bilangan bulat n.

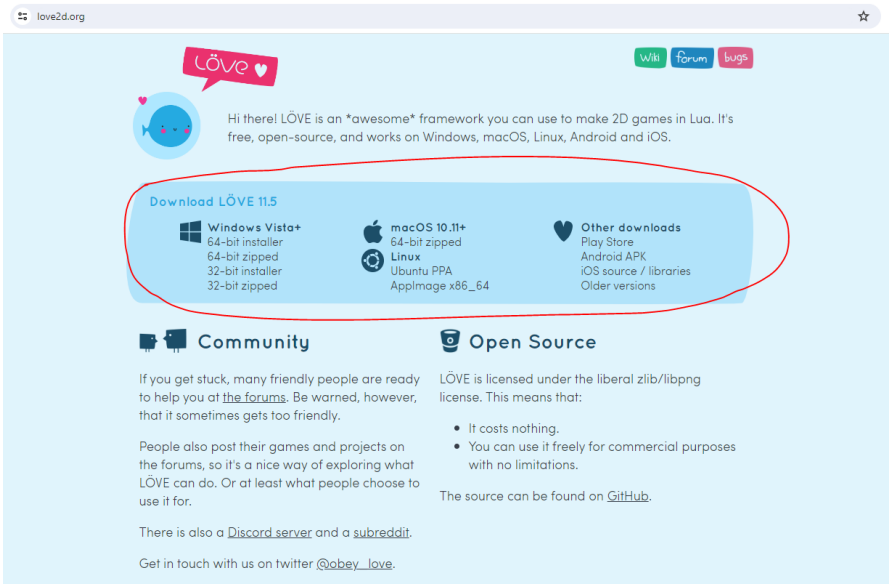
Kesimpulan: Penggunaan Global dan Local dalam Fungsi

Variabel yang dideklarasikan di dalam sebuah fungsi dengan menggunakan kata kunci local hanya dapat diakses di dalam fungsi tersebut (variabel lokal). Sedangkan jika variabel dideklarasikan tanpa menggunakan kata kunci local, maka variabel tersebut akan menjadi variabel global yang dapat diakses dari mana saja di dalam program.

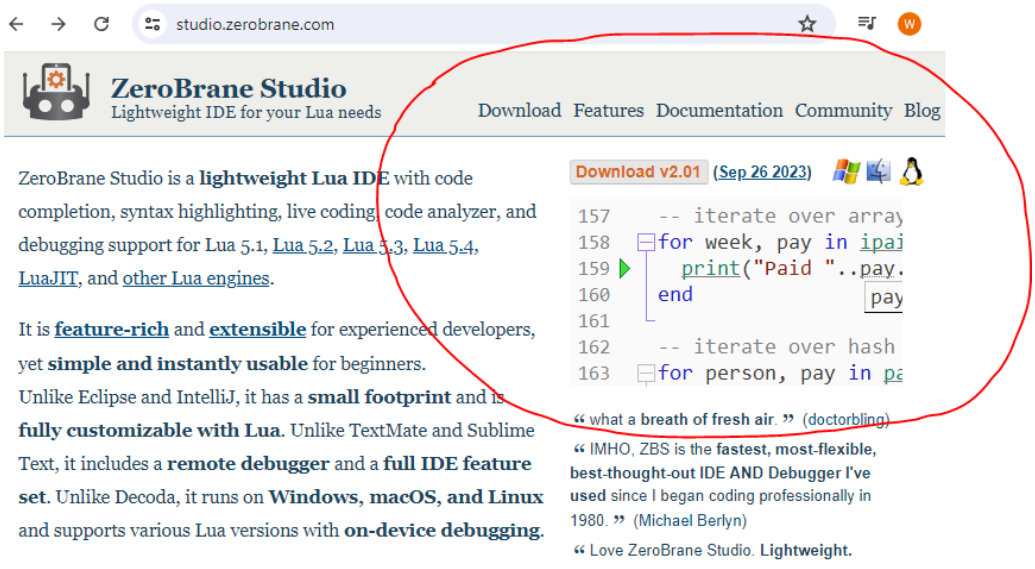
Bab 3: Dasar-Dasar Framework LÖVE

Instalasi: disini kita akan menggunakan peralatan yang paling simpel dan ramah untuk pemula.

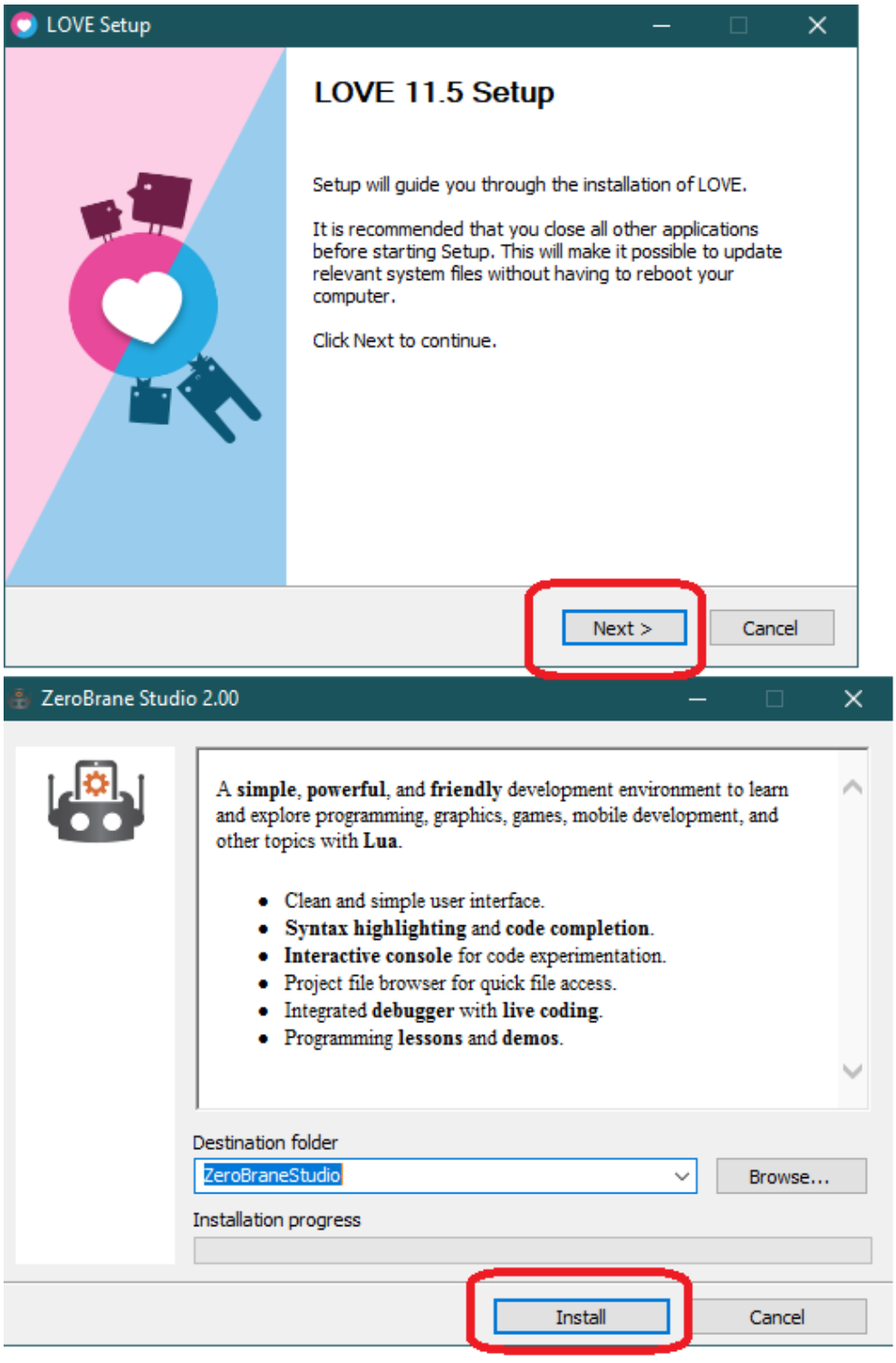
1. Anda bisa langsung mengunduh file instalasi LÖVE di https://LÖVE.org/



2. Untuk mempermudah meng-koding Lua, Anda dapat menggunakan tool/alat ZeroBrane Studio yang dapat diunduh di https://studio.zerobrane.com/



3. Menginstal LÖVE dan ZeroBrane Studio sama seperti menginstall aplikasi pada umumnya.



Fungsi-Fungsi pada Framework Lua

Namespace

Pada konteks ini, namespace adalah suatu cara untuk mengelompokkan dan mengatur fungsi, variabel, dan modul dalam sebuah ruang lingkup tertentu untuk mencegah konflik nama dan untuk menjaga kode tetap terstruktur dan mudah diatur. Dalam LÖVE (Love2D), love berfungsi sebagai namespace utama yang mengelompokkan semua fungsi dan modul yang disediakan oleh framework tersebut.

Dengan namespace love, Anda bisa mengakses berbagai modul dan fungsi spesifik yang berkaitan dengan pengembangan game. Misal: love.(di ikuti nama fungsi), contohnya love.load(). Berikut adalah 3 contoh fungsi utama yang digunakan pada aplikasi yang dibuat menggunakan LÖVE:

Penjelasan Fungsi love.load()

Dalam konteks pemrograman menggunakan LÖVE, fungsi love.load() adalah fungsi callback yang dieksekusi satu kali saat game dimulai. Fungsinya adalah untuk menginisialisasi variabel dan memuat sumber daya yang dibutuhkan game sebelum mulai berjalan.

Contoh sederhana dari penggunaan love.load():

```
function love.load()
    -- Inisialisasi variabel posisi pemain
    player = {
        x = 100,
        y = 100,
        speed = 200
    }

    -- Memuat gambar pemain
    player.image = love.graphics.newImage("player.png")

    -- Memuat font untuk teks
    mainFont = love.graphics.newFont("arial.ttf", 24)
    love.graphics.setFont(mainFont)

    -- Memuat suara
    shootSound = love.audio.newSource("shoot.wav", "static")
end
```

Penjelasan:

- player diinisialisasi sebagai tabel dengan posisi dan kecepatan.
- Gambar pemain dimuat dan disimpan dalam player.image.
- Font utama dimuat dan diatur sebagai font aktif.
- Suara tembakan dimuat dan disimpan dalam shootSound.

Semua inisialisasi ini dilakukan dalam love.load(), sehingga saat game mulai berjalan, semua sumber daya sudah siap digunakan.

Penjelasan Fungsi love.draw()

Dalam konteks pemrograman menggunakan LÖVE, fungsi love.draw() memiliki peran yang sangat penting. Fungsi ini digunakan untuk menggambar atau merender semua elemen grafis dari game ke layar. Setiap frame, LÖVE akan memanggil fungsi ini secara otomatis untuk memperbarui tampilan visual dari game. Di dalam love.draw(), Anda akan meletakkan semua kode yang berfungsi untuk menggambar seperti karakter, latar belakang, teks, dan elemen-elemen grafis lainnya.

Contoh sederhana dari penggunaan love.draw():

```
function love.draw()
   -- Mengatur warna gambar menjadi merah
   love.graphics.setColor(1, 0, 0)
   -- Menggambar persegi panjang di posisi x=100, y=100 dengan lebar 200 dan tinggi 150
   love.graphics.rectangle("fill", 100, 100, 200, 150)
end
```

Dalam contoh di atas, setiap kali frame baru dirender, LÖVE akan menggambar sebuah persegi panjang merah di layar pada posisi yang ditentukan. Anda bisa menggunakan berbagai fungsi dari love.graphics di dalam love.draw() untuk menggambar berbagai macam elemen visual sesuai kebutuhan game Anda.

Penjelasan Fungsi love.update()

Dalam konteks pemrograman menggunakan LÖVE, fungsi love.update(dt) digunakan untuk mengupdate logika permainan, seperti pergerakan objek, deteksi tabrakan, dan pemrosesan input. Fungsi ini dipanggil secara terus-menerus setiap frame oleh LÖVE, dan dt (delta time) merupakan argumen yang berisi waktu dalam detik yang telah berlalu sejak frame terakhir. Menggunakan dt memungkinkan animasi dan pergerakan tetap mulus dan konsisten, tidak tergantung pada kecepatan frame rate.

Contoh penggunaan love.update(dt):

```
-- Posisi x dari objek
local x = 100
-- Kecepatan pergerakan objek dalam piksel per detik
local speed = 200

function love.update(dt)
   -- Menggerakkan objek ke kanan berdasarkan kecepatan dan waktu yang telah berlalu
   x = x + speed * dt
end

function love.draw()
   -- Menggambar objek di posisi x yang diperbarui
   love.graphics.rectangle("fill", x, 100, 50, 50)
end
```

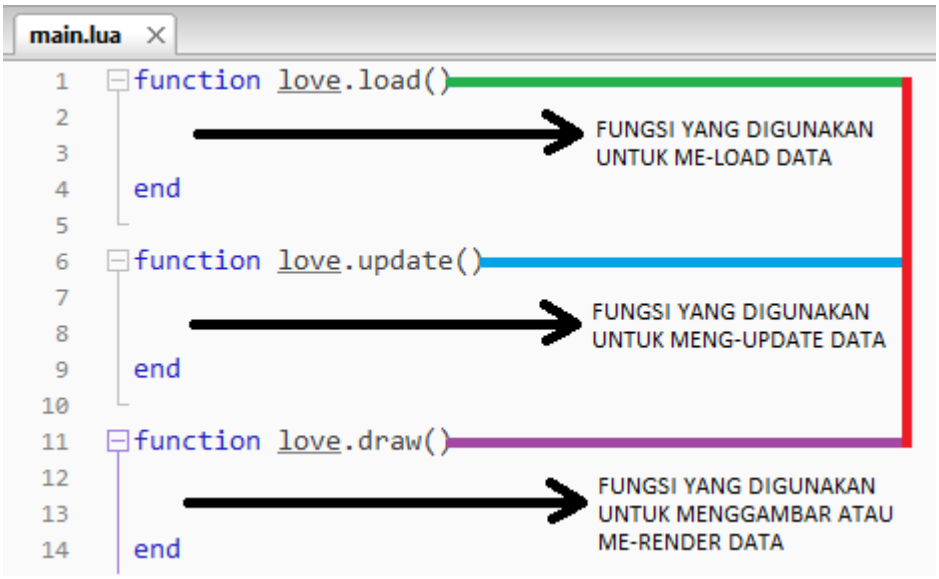
Penjelasan:

- x adalah posisi horizontal dari objek.
- speed adalah kecepatan pergerakan objek dalam piksel per detik.
- Di dalam love.update(dt), posisi x diperbarui dengan menambahkan speed * dt setiap frame, membuat objek bergerak ke kanan seiring waktu.

Dengan cara ini, pergerakan objek akan tetap konsisten terlepas dari variasi dalam frame rate, karena perubahan posisi dihitung berdasarkan waktu yang telah berlalu (delta time)

Kesimpulan Sementara

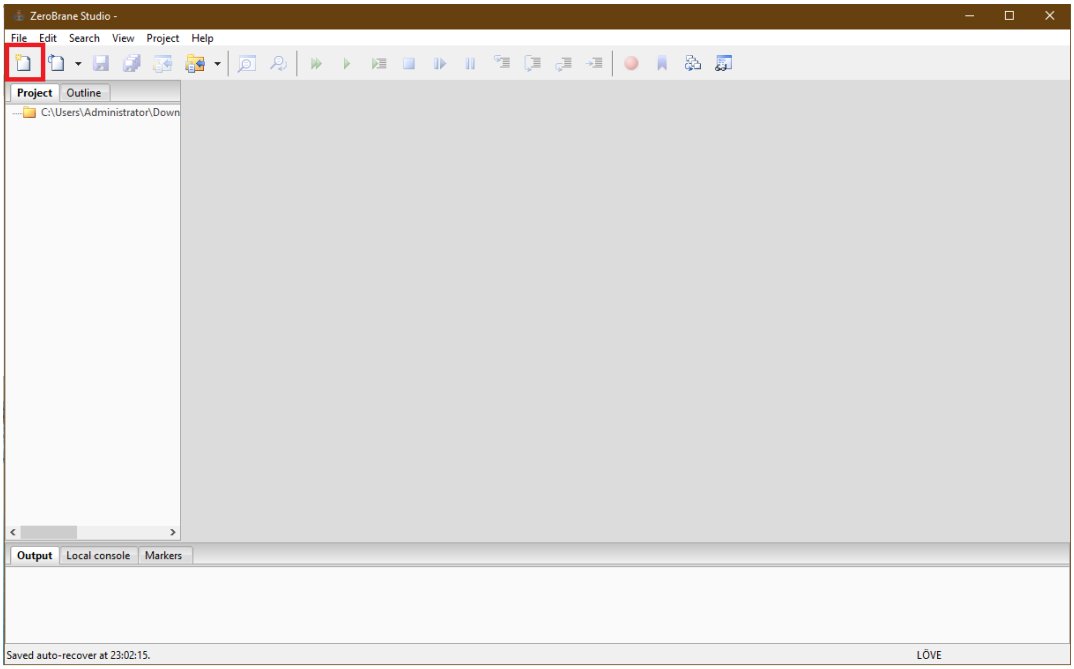
Dari ketiga contoh penggunaan fungsi tersebut, dapat disimpulkan struktur dari penggunaan fungsi-fungsi tersebut pada pembuatan sebuah game seperti ini:

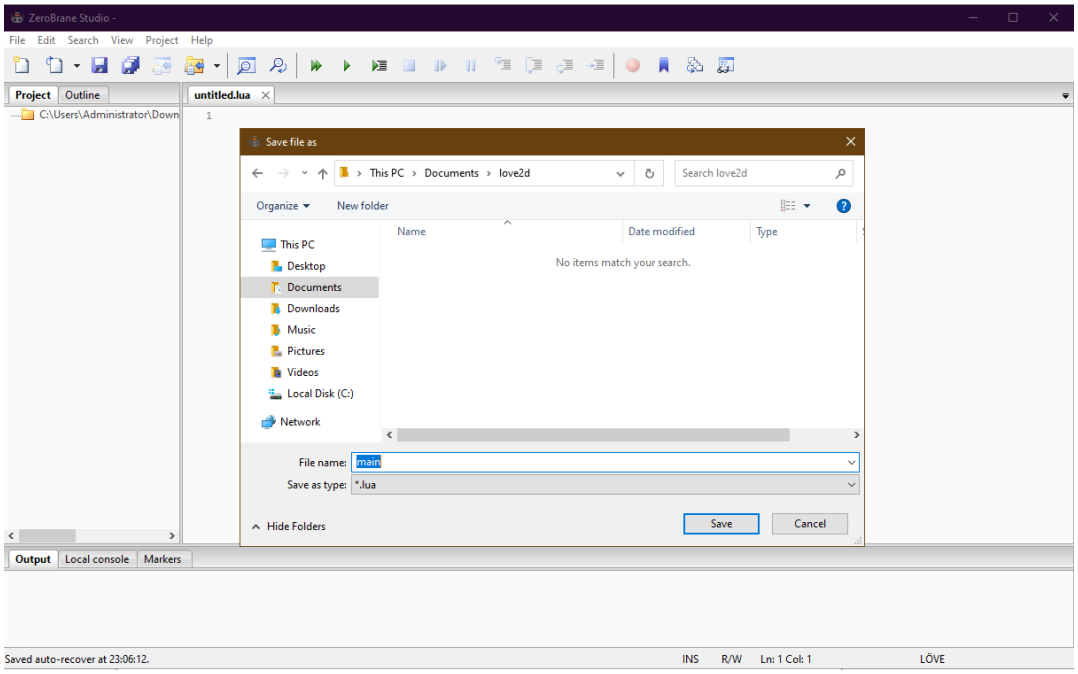


Setelah mengetahaui susunan penggunan dari fungsi tersebut, di bab selanjutnya kita akan membuat game sederhana, dengan menulisan kode-kode pada fungsi-fungsi tersebut. Dan Semua fungsi dan modul yang diperlukan untuk pembuatan game menggunakan LÖVE diakses melalui namespace love.

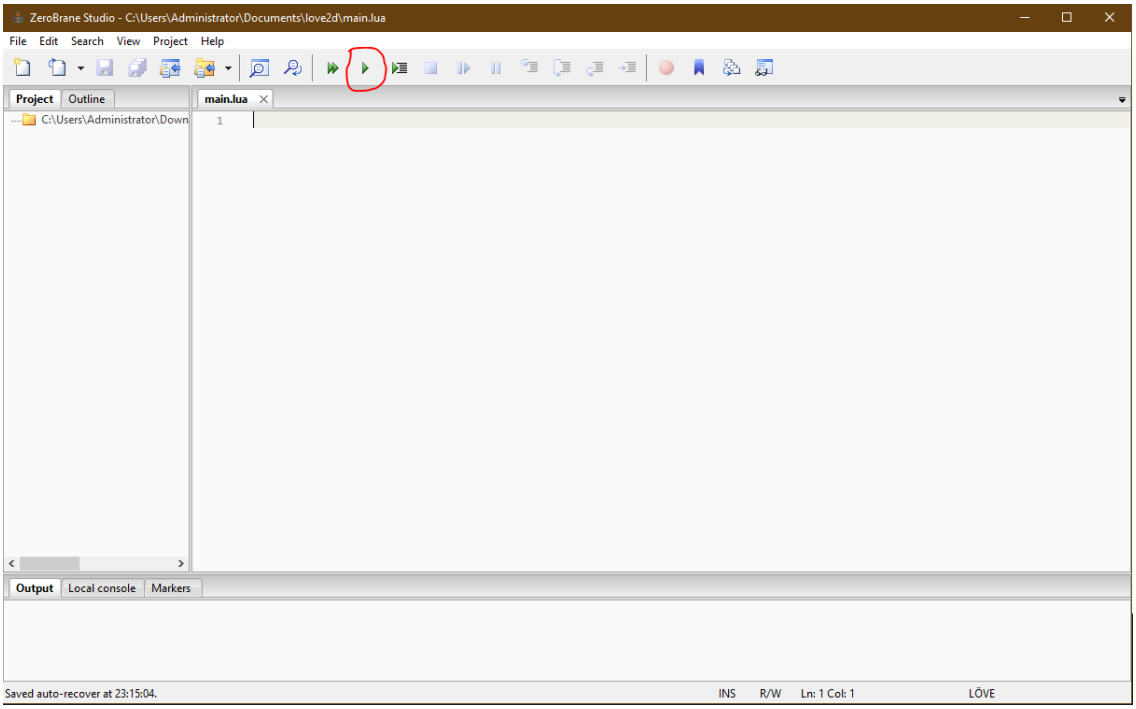
Mencoba Menjalankan Program

- Buat file dengan nama main.lua kemudian simpan file tersebut.

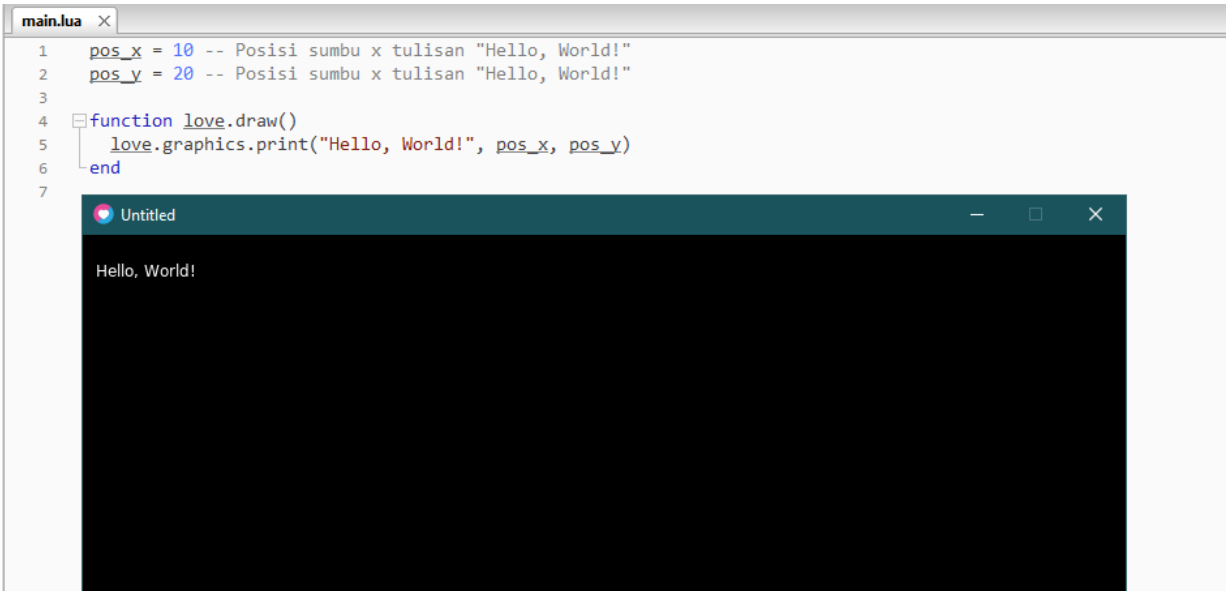




- Anda dapat menjalankan program dengan menakan tombol f5 atau tombol play pada menu bar seperti dibawah:



- Mencoba menjalan program dangan menampilkan tulisan "Hello, World!"menggunakan fungsi love.graphics.print() dan love.draw():



Kesimpulan Dari Bab 3

Sampai disini kita sudah membahas dasar-dasar konsep pemrograman bahasa Lua dan penggunaan dasar LÖVE. Untuk melanjutkan ke bab berikutnya, pastikan Anda benar-benar sudah memahami setiap poin dari bab 1 sampai bab 3. Anda dapat menguji pemahaman Anda dengan mengerjakan soal-soal latihan dibawah:

Latihan

1. Buatlah dua variabel a dan b, kemudian buatlah fungsi untuk menghitung luas persegi dengan parameter variabel a dan b tersebut.

2. Buatlah sebuah variabel bernama: angka, kemudian buatlah sebuah fungsi untuk mengecek apakah angka tersebut genap atau ganjil.

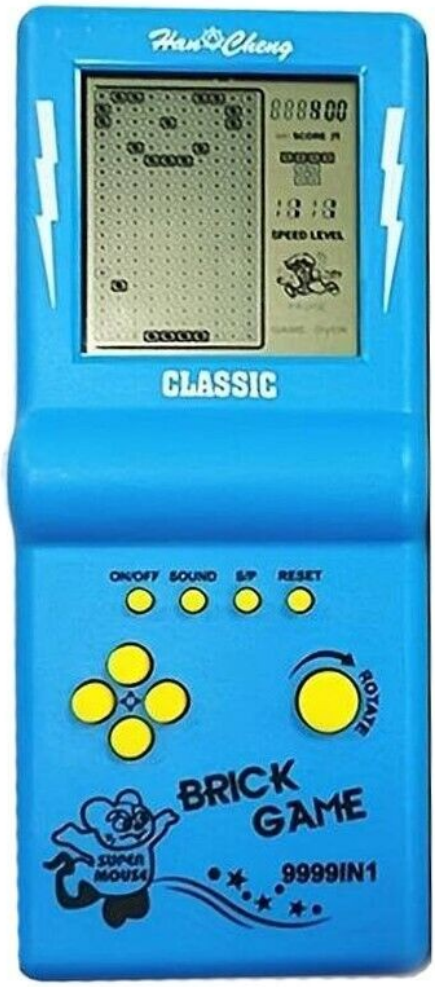
3. Dengan menggunakan fungsi love.graphics.print() dan love.draw(), tampilkan lah tulisan “It's My Life” pada layar.

4. Dengan menggunakan dua variabel pos_x dan pos_y, buatlah tulisan “It's My Life” pada soal sebelumnya bergerak, dengan menggunakan kode yang ditulis pada fungsi love.update() untuk menggeser tulisan tersebut.

Bab 4: Membuat Game Sederhana

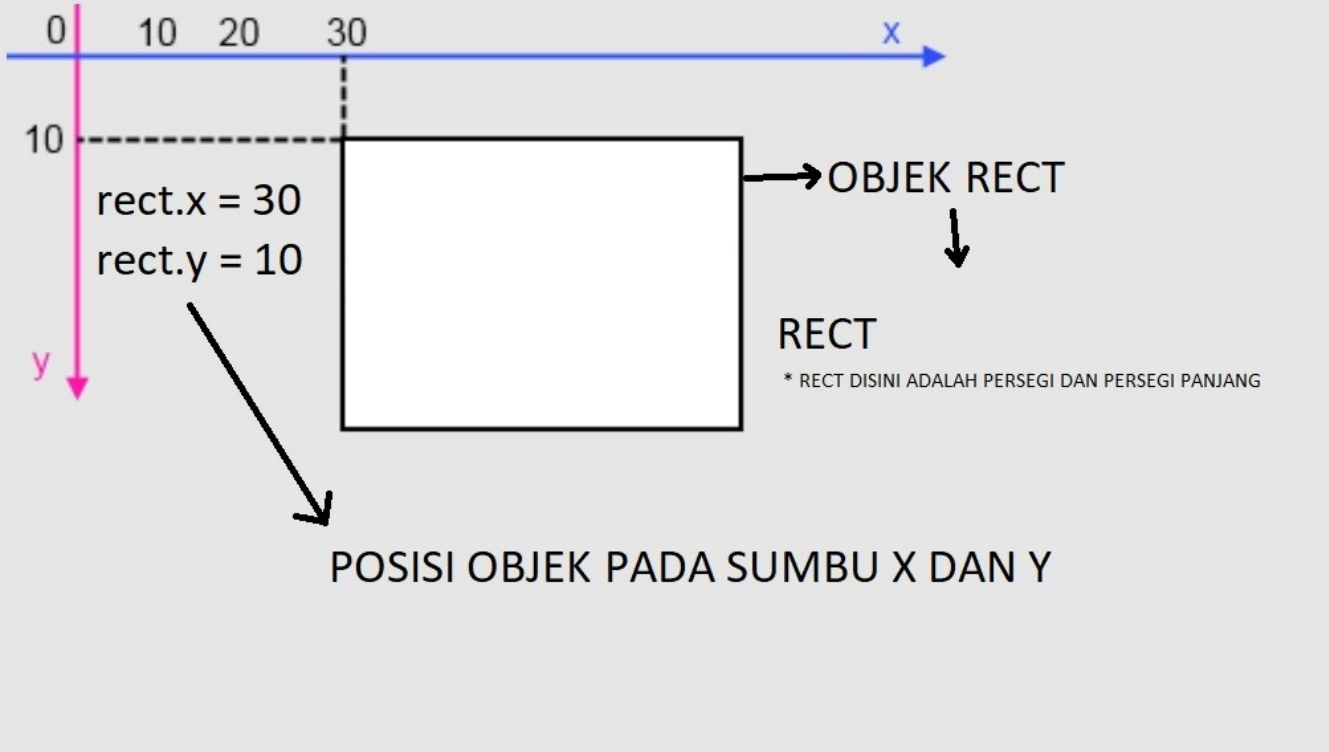
Salam Pembuka

Setelah memahami dasar-dasar bahasa Lua dan framework LÖVE, kini saatnya membuat sebuah game sederhana menggunakan LÖVE dan Lua. Game sederhana (tapi versi yang lebih “berwarna”) yang akan dibuat seperti ini:



Penggunaan Istilah

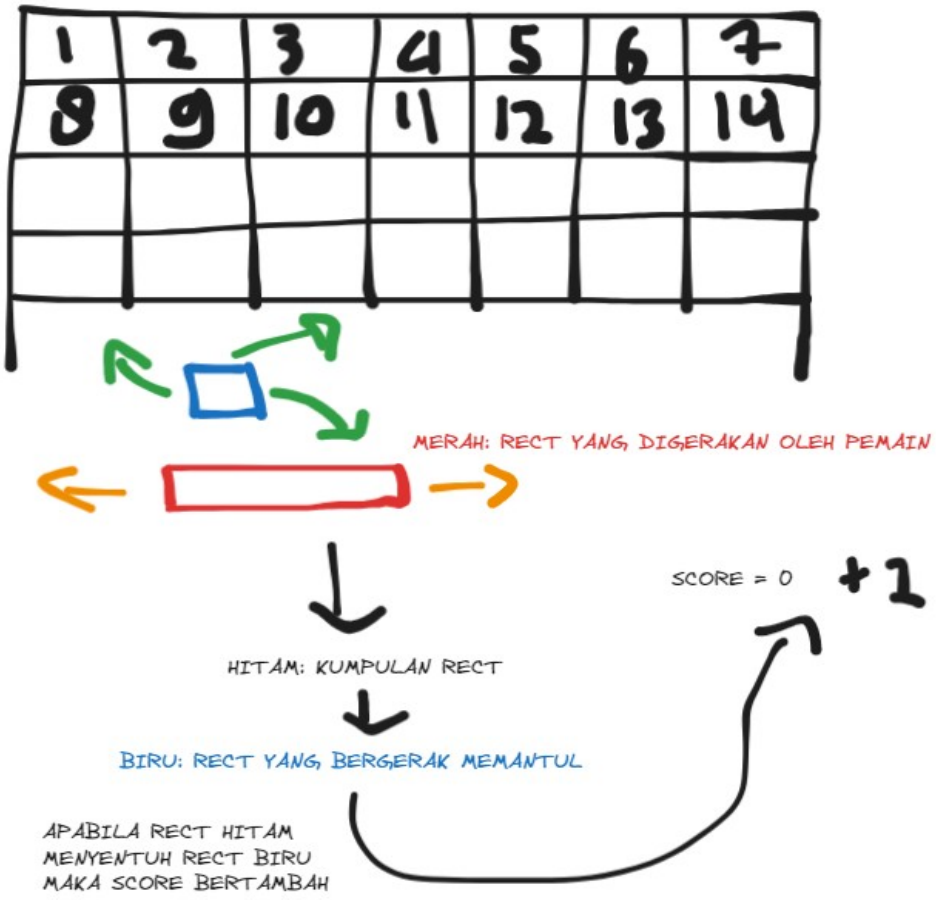
Penggunakan kata “objek”, kata “objek” disini bukanlah objek yang secara spesifik merujuk pada konsep pemrograman berorientasi objek (OOP). Melainkan simpel saja, objek disini adalah objek selayaknya objek menurut pemahaman manusia pada umumnya. Selain objek ada juga sedikit istilah matematika, seperti rect untuk persegi dan persegi panjang, dan sumbu x dan y untuk posisi. Berikut sedikit istilah matematika yang akan di gunakan pada pembuatan game ini:



Penjelasan:

- Contoh penggunaan istilah objek untuk menunjuk sebuah persegi atau persegi panjang.
- Penggunaan sumbu x dan y untuk merepresentasikan posisi objek persegi atau persegi panjang.

Contoh Penggunaan Rect pada Data Persegi dan Persegi Panjang



Dalam pemrograman grafik, biasanya hanya menyediakan satu penamaan untuk persegi panjang (rectangle) dan tidak secara eksplisit untuk persegi (square). Ini karena persegi adalah kasus khusus dari persegi panjang. Dengan kata lain, persegi adalah persegi panjang di mana semua sisi sama panjang. Berikut beberapa alasan mengapa ini umum terjadi:

- **Kesederhanaan dan Efisiensi**: Menyediakan satu variabel untuk persegi panjang mengurangi redundansi dan mempermudah pemeliharaan kode. Kita hanya perlu membuat satu set fungsi untuk persegi panjang, yang juga dapat menangani kasus khusus persegi.
- **Generalitas**: Dengan hanya memiliki variabel persegi panjang, kita dapat lebih fleksibel. Kode yang sama dapat digunakan untuk berbagai macam persegi panjang tanpa perlu membuat kelas tambahan hanya untuk persegi.

Array dan Metatables pada Lua

Dalam bahasa pemrograman Lua, array adalah jenis tabel di mana indeks yang digunakan untuk mengakses elemen-elemen array dimulai dari 1. Berikut adalah beberapa contoh untuk membuat dan menggunakan array dalam Lua:

Membuat Array

Berikut adalah contoh dasar pembuatan array dalam Lua:

```
-- Membuat array dengan 5 elemen
local array = {1, 2, 3, 4, 5}
```

Mengakses Elemen Array

Untuk mengakses elemen array, gunakan indeks yang dimulai dari 1:

```
print(array[1])  -- Output: 1
print(array[5])  -- Output: 5
```

Menambah Elemen ke Array

Anda bisa menambah elemen baru ke array dengan menentukan indeks baru:

```
array[6] = 6
print(array[6])  -- Output: 6
```

Mengubah Nilai Elemen Array

Anda bisa mengubah nilai dari elemen tertentu dengan menentukan indeksnya:

```
array[3] = 10
print(array[3])  -- Output: 10
```

Mengiterasi Elemen Array

Menggunakan for loop untuk mengiterasi elemen dalam array:

```
for i = 1, #array do
   print(array[i])
end
```

Fungsi-Fungsi Array

Lua memiliki beberapa fungsi bawaan yang berguna untuk mengoperasikan array:

- table.insert(array, value): Menambahkan nilai ke akhir array.
- table.remove(array, index): Menghapus nilai dari array pada indeks tertentu.
- table.sort(array): Mengurutkan elemen-elemen dalam array.

Contoh penggunaan fungsi-fungsi tersebut:

```
-- Menambah elemen ke array
table.insert(array, 7)
print(array[7])  -- Output: 7

-- Menghapus elemen dari array
table.remove(array, 3)
print(array[3])  -- Output: 4 (karena elemen ke-3 sebelumnya dihapus)

-- Mengurutkan array
table.sort(array)
for i = 1, #array do
   print(array[i])
end
```

Metatables di Lua

Metatables: Metatables adalah fitur khusus dalam Lua yang memungkinkan Anda untuk mengubah perilaku tabel. Dengan menggunakan metatables, Anda bisa mendefinisikan bagaimana tabel bereaksi terhadap operasi tertentu, seperti penambahan dua tabel, pengaksesan elemen yang tidak ada, atau pemanggilan fungsi. Contoh:

```
-- Membuat tabel biasa
t = {a = 1, b = 2}

-- Membuat metatable
mt = {
   __index = function(table, key)
      return key .. " tidak ditemukan"
   end
}

-- Mengatur metatable untuk tabel t
setmetatable(t, mt)

-- Mengakses elemen yang tidak ada
print(t.c)  -- Output: "c tidak ditemukan"
```

Perbedaan Utama Array dan Metatables

- **Fungsi**: Array digunakan untuk menyimpan data dalam urutan tertentu, sementara metatables digunakan untuk mengubah perilaku dari tabel.
- **Penggunaan**: Array digunakan langsung sebagai tabel biasa, sedangkan metatables perlu dikaitkan dengan tabel menggunakan fungsi setmetatable.

Jadi, meskipun array dan metatables sama-sama merupakan tabel di Lua, mereka digunakan untuk tujuan yang sangat berbeda dan tidak dapat dipertukarkan.

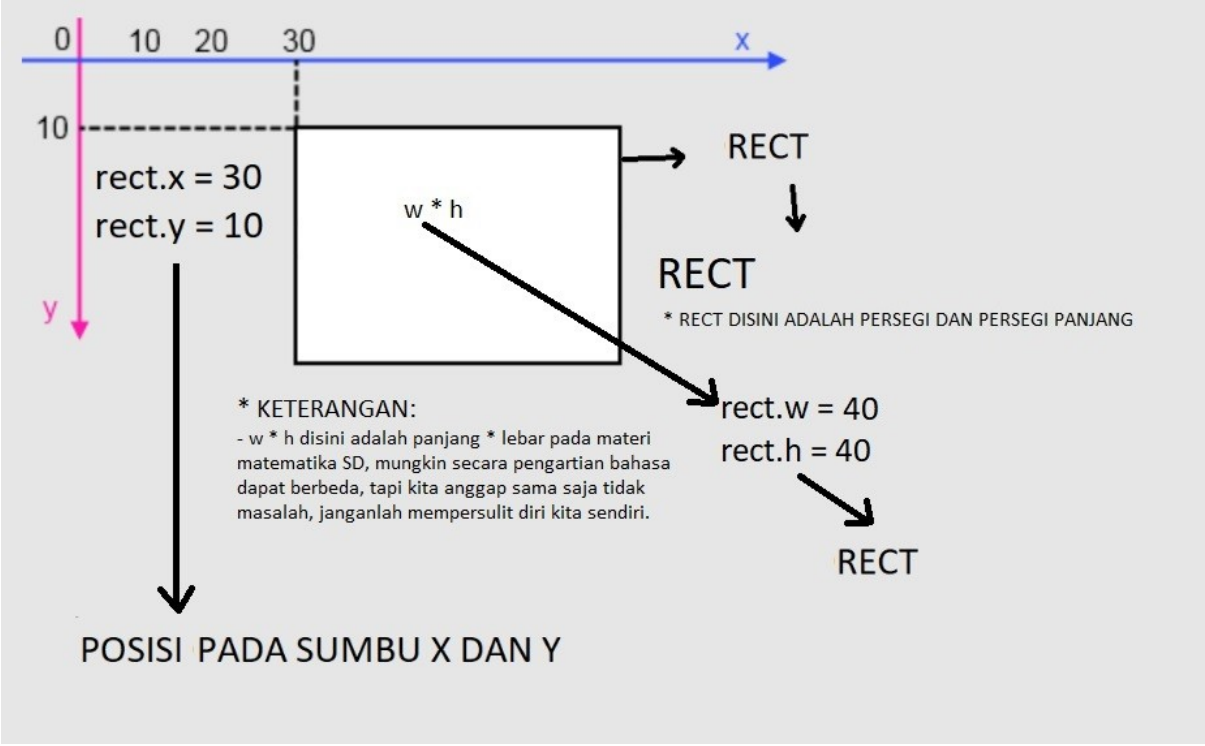
Penggunaan Array pada Pembuatan Game

Pada pembuatan game sederhana ini, array tersebut digunakan untuk menyimpan sekumpulan persegi yang ada pada game. Cara penggunaan array untuk menyimpan persegi pun sama seperti yang dicontohkan untuk menyimpan bilangan bulat. Seperti ini contohnya:

Membuat Persegi

```
-- Membuat persegi yang diberi nama rect
rect = { x = 30, y = 10, w = 40, h = 40 }
```

Penjelasan:



Menggunakan Array Untuk Menyimpan Persegi

Setelah membuat persegi tersebut, selanjutnya kita dapat menyimpan persegi tersebut ke sebuah array seperti ini:

```
-- Membuat persegi yang diberi nama rect
rect = { x = 30, y = 10, w = 50, h = 40 }

-- Membuat array yang nantinya akan di isi persegi
rects = {}

-- Memasukkan rect ke dalam array rects menggunakan fungsi table.insert
table.insert(rects, rect)
```

Implementasi Tabel Read-Only untuk Konstanta dengan Metatables di Lua

Metatables untuk membuat sebuah tabel yang hanya bisa dibaca (read-only) yang berfungsi untuk menyimpan konstanta.

```
-- Using metatables to create a read-only table for constants
local function read_only(t)
   local proxy = {}
   local mt = {
      __index = t,
      __new_index = function(table, key, value)
         error("Attempt to modify read-only table", 2)
      end,
      __metatable = false
   }
   setmetatable(proxy, mt)
   return proxy
end
```

Penjelasan:

- **Fungsi read_only**
  - Fungsi ini menerima satu argumen, yaitu t, yang merupakan tabel asli yang ingin dijadikan hanya bisa dibaca.
- **Membuat Proxy Table**
  - local proxy = {}: Membuat tabel kosong yang akan menjadi proxy (perantara) untuk tabel asli.

- **Mendefinisikan Metatable**
  - local mt = {}: Membuat metatable yang akan menentukan perilaku dari proxy table.
  - __index = t: Menentukan bahwa setiap kali ada akses ke elemen tabel proxy yang tidak ada, Lua akan mencari elemen tersebut di tabel asli t.
  - __new_index = function(table, key, value): Mendefinisikan fungsi yang akan dipanggil setiap kali ada usaha untuk menambahkan atau mengubah elemen dalam proxy table. Fungsi ini akan menghasilkan error dengan pesan "Attempt to modify read-only table", yang berarti tabel ini tidak bisa diubah.
  - __metatable = false: Menyembunyikan metatable sehingga tidak bisa diakses atau diubah dari luar, membuatnya lebih aman.
- **Mengatur Metatable**
  - setmetatable(proxy, mt): Mengaitkan metatable mt dengan proxy table proxy.
- **Mengembalikan Proxy Table**
  - return proxy: Mengembalikan proxy table yang sudah dikonfigurasi sebagai hasil dari fungsi read_only.

Menggunakan Metatables untuk Tabel Hanya Baca (Read-Only)

- **Mendefinisikan Konstanta**:
  - Kode menggunakan fungsi read_only untuk membuat tabel CONST yang hanya bisa dibaca. Ini memastikan bahwa nilai-nilai konstanta seperti RECT_W dan RECT_H tidak dapat diubah secara tidak sengaja selama runtime.
  - Ini membantu menjaga integritas data penting yang seharusnya tetap konstan sepanjang eksekusi program.

```
local CONST = read_only({
   RECT_W = 50,
   RECT_H = 50
})
```

Menggunakan Array untuk Menyimpan dan Mengelola Objek Persegi

- **Menyimpan Koleksi Objek**:
  - Tabel rects digunakan sebagai array untuk menyimpan koleksi objek persegi. Setiap persegi didefinisikan dengan atribut seperti x, y, w, h, dan id.
  - Array ini memungkinkan penanganan dan manipulasi banyak objek persegi secara efisien, misalnya saat perlu menggambar mereka di layar atau memeriksa tabrakan.

```
local rects = {}
```

Inisialisasi Objek Persegi:

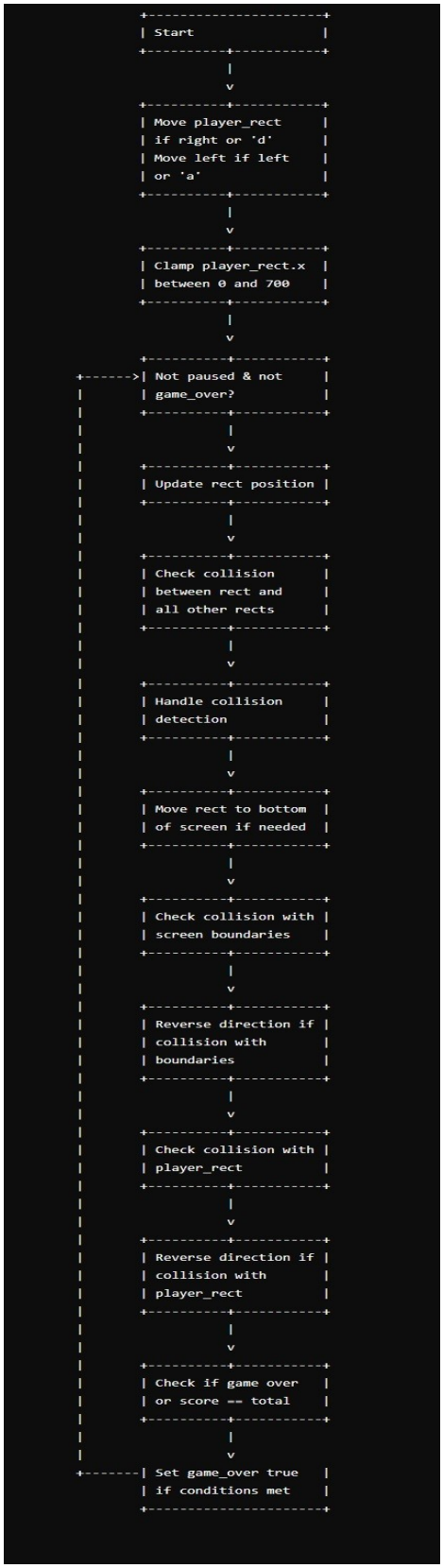
- Fungsi init_rects digunakan untuk mengisi array rects dengan objek persegi. Fungsi ini memastikan bahwa semua persegi diatur dengan benar pada posisi dan ukuran yang diinginkan.

```
function init_rects()
   for i in ipairs(rects) do
      rects[i] = nil
   end

   for i = 1, 16 do
      table.insert(rects, {
         x = (i - 1) * CONST.RECT_W,
         y = 50,
         w = CONST.RECT_W,
         h = CONST.RECT_H,
         id = i + 16 * 0
      })
   end
end
```

Kesimpulan

- **Metatables**: Digunakan untuk membuat tabel hanya baca (read-only) yang menyimpan konstanta. Ini memastikan bahwa nilai-nilai konstanta tidak bisa diubah, menjaga integritas data penting.
- **Array**: Digunakan untuk menyimpan dan mengelola koleksi objek, dalam hal ini objek persegi. Array memudahkan pengelolaan sejumlah besar objek dengan atribut yang serupa, memungkinkan operasi seperti iterasi, penggambaran, dan deteksi tabrakan dilakukan secara efisien.

Flowchart


Start
Move player_rect if right or 'd' Move left if left or 'a'
Clamp player_rect.x between 0 and 700
Not paused & not game_over?
Update rect position
Check collision between rect and all other rects
Handle collision detection
Move rect to bottom of screen if needed
Check collision with screen boundaries
Reverse direction if collision with boundaries
Check collision with player_rect
Reverse direction if collision with player_rect
Check if game over or score == total
Set game_over true if conditions met

Menuliskan Kode pada Fungsi love.load()

Seperti yang sudah disimpulkan di bab 3, untuk membuat sebuah game kita menuliskan kode pada 3 fungsi yaitu: love.load(), love.update(), dan love.draw(). Dengan demikian fungsi love.load() berfungsi untuk menginisialisasi data. Berikut data-data yang nantinya akan di update dan di gambar pada fungsi love.update() dan love.draw():

```
function love.load()
    -- Untuk menset judul dari window
    love.window.setTitle("Classic Game")

    -- Untuk menset warna background
    love.graphics.setBackgroundColor(0, 0, 0)

    -- Variabel untuk mendeteksi tumbukan
    is_colliding_rects = false
    is_colliding_player = false

    -- Inisialisasi sekumpulan persegi
    init_rects()

    -- Inisialisasi rect yang akan dipantul-pantulkan
    rect = {
        x = 100,
        y = 300,
        w = 40,
        h = 40,
        speed_x = 150, -- Speed in the X direction
        speed_y = 100, -- Speed in the Y direction
    }

    -- Lebar dan tinggi layar
    screen_width = love.graphics.getWidth()
    screen_height = love.graphics.getHeight()
end
```

Menuliskan Kode pada Fungsi love.update()

Pada fungsi ini, kita akan menulisakan kode-kode yang berfungsi untuk meng-update posisi dari semua persegi dan persegi panjang, dan kondisi yang mempengaruhi variabel score seperti ini:

```
function love.update(dt)
    -- Memperbaharui posisi pemain
    if love.keyboard.isDown("right") or love.keyboard.isDown("d") then
        player_rect.x = player_rect.x + SPEED * dt
    elseif love.keyboard.isDown("left") or love.keyboard.isDown("a") then
        player_rect.x = player_rect.x - SPEED * dt
    end

    -- Batas pergerakan pemain
    player_rect.x = math.max(0, math.min(700, player_rect.x))

    -- Memperbaharui rect yang bergerak
    if not paused and not game_over then
        -- Update the rectangle's position
        rect.x = rect.x + rect.speed_x * dt
        rect.y = rect.y + rect.speed_y * dt
    end

    -- Mengecek tumbukan dari rect yang bergerak dan sekumpulan rect
    local move_rect = { x = rect.x, y = rect.y, w = rect.w, h = rect.h }
    local is_colliding_rects = false
    for i = 1, #rects do
        if check_collision_rect(move_rect, rects[i]) and rects[i].id ~= 99 then
            is_colliding_rects = true
            rects[i].id = 99
            score = score + 1;
            break
        end
    end
```

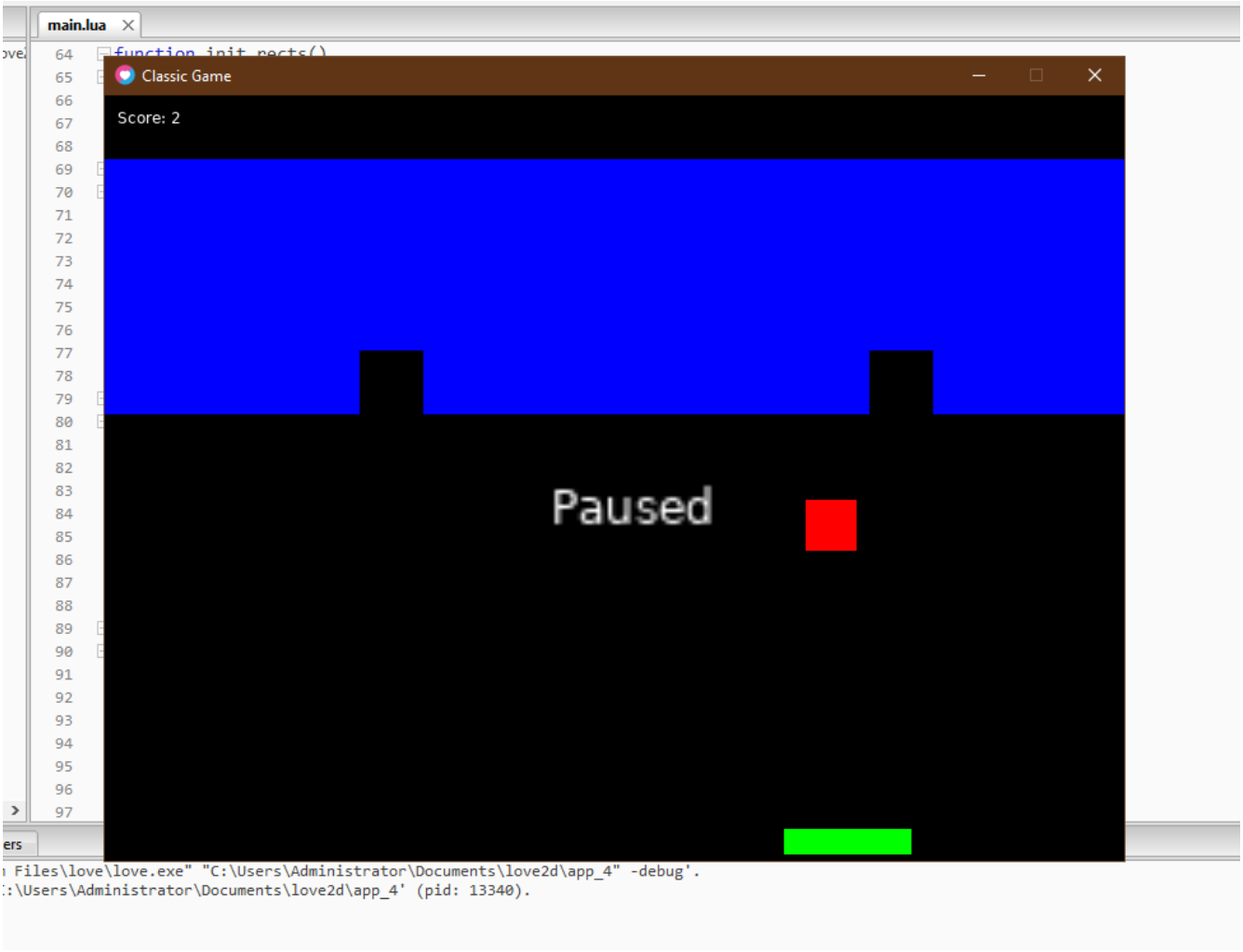
```
local is_colliding_player = false
   if check_collision_rect(player_rect, move_rect) then
     is_colliding_player = true
   end

   -- Mengecek tumbukan dari keseluruhan rect dan pinggiran layar
   -- Memperbaruhi kecepatan dari rect yang mempengaruhi arah rect
   if is_colliding_rects or rect.y < 0 then
      rect.speed_y = -rect.speed_y
   else
      if rect.x < 0 then
         rect.x = 0
         rect.speed_x = -rect.speed_x
      elseif rect.x + rect.w > screen_width then
         rect.x = screen_width - rect.w
         rect.speed_x = -rect.speed_x
      end

      if is_colliding_player then
         rect.speed_y = -rect.speed_y
      end
   end

   -- Mengecek apakah permainan berakhir
   if rect.y > screen_width and not game_over or score == rects_total then
     game_over = true
   end
end
```

Menuliskan Kode pada Fungsi love.draw()

Pada fungsi ini akan menulisakan kode-kode yang berfungsi untuk meng-gambar persegi dan persegi panjangnya dari data-data yang sudah di load dan di update pada fungsi-fungsi sebelumnya. Dan juga tulisan untuk status game:

```
function love.draw()
    -- Menggambar sekumpulan rect
    for i, rect in ipairs(rects) do
        if rect.id ~= 99 then
            love.graphics.setColor(0, 0, 1)
            love.graphics.rectangle("fill", rect.x, rect.y, rect.w, rect.h)
        end
    end

    -- Menggambar rect yang bergerak
    love.graphics.setColor(1, 0, 0)  -- Red for the moving rect
    love.graphics.rectangle("fill", rect.x, rect.y, rect.w, rect.h)

    -- Menggambar rect player
    love.graphics.setColor(0, 1, 0)  -- Green for the player rect
    love.graphics.rectangle("fill", player_rect.x, player_rect.y, player_rect.w,
player_rect.h)

    -- Menggambar tulisan skor
    love.graphics.setColor(1, 1, 1)  -- White text
    local score_text = "Score: " .. tostring(score)
    love.graphics.print(score_text, 10, 10)

   -- Menggambar tulisan jika permainan dijeda
    if paused then
      love.graphics.print("Paused", screen_width/2-50, screen_height/2, 0, 3, 3)
    end

    -- Menggambar tulisan jika permainan berakhir
    if game_over then
      love.graphics.print("Game Over", screen_width/2-100, screen_height/2-50, 0, 3,
3)
    end
end
```

Hasilnya Setelah Dijalankan



Full Source Code:

https://github.com/rendertree/simtut/blob/main/demo/main.lua

Latihan

- Untuk menguji pemahaman Anda coba buat susunan perseginya menjadi acak seperti yang ada pada game klasik.

Referensi

- https://www.lua.org/
- https://love2d.org/
- https://studio.zerobrane.com/