Universidade Federal do Rio Grande

Henrique Bertochi - 162647

Vicenzo Copetti – 164433

Projeto final Linguagens de Programação

RELATÓRIO: Jogo PacMan em Lua

Rio Grande, RS

2024

Por que escolhemos a linguagem Lua?

A escolha da linguagem Lua para o desenvolvimento do nosso jogo foi motivada por três principais razões:

1. **Desafio e Aprendizado:** Optamos por uma linguagem que não tínhamos experiência prévia, buscando uma nova aventura e a oportunidade de adquirir conhecimento em uma tecnologia desconhecida.
2. **Biblioteca LOVE2D:** Durante nossa pesquisa sobre bibliotecas recomendadas para desenvolvimento de jogos, descobrimos o LOVE2D. Essa biblioteca se destacou por sua didática e facilidade de uso, além de oferecer uma documentação de qualidade e uma vasta quantidade de recursos educacionais que facilitam a criação de jogos.
3. **Semelhança com Python:** Lua possui uma sintaxe que se assemelha bastante à do Python, o que facilitou o processo de adaptação e aprendizado para nossa equipe, que já possuía experiência com Python.

O que implementamos no jogo?

O PacMan é um clássico mundialmente reconhecido, e nos esforçamos para recriar o jogo com a maioria das funcionalidades. As funções implementadas incluem:

- Movimento do PacMan
- Movimento dos Fantasmas
    - No jogo original, cada um dos 4 fantasmas possuem uma lógica de movimentação, por exemplo o Blinky (Vermelho) tende a perseguir o Pacman, porém no nosso jogo a movimentação deles é aleatória a partir do momento que batem em uma parede
- Comer as pílulas
    - Pílulas normais e pílulas especiais que permitem matar os Fantasmas
- Comer os Fantasmas
- Sistema de Vidas
- Sistema de Pontos
- Sistema de Levels
    - A cada vitória (comer todas as pílulas) aumenta a velocidade para o próximo level
- Sistema de Menu
- Sistema de GameOver
    - Ao perder as 3 vidas

Principais Funcionalidades do Software:

1. **Lógica de Movimentação do Pacman**

- **Descrição**: Parte do código responsável pela movimentação do Pacman, demonstrando como ele responde aos comandos do jogador (cima, baixo, esquerda, direita).

```
function updatePacman()
    if isUpPressed then
        pacman.nextDirection = 1
        pacman.nextDirectionText = "up"
    end
    if isDownPressed then
        pacman.nextDirection = 3
        pacman.nextDirectionText = "down"
    end
    if isLeftPressed then
        pacman.nextDirection = 4
        pacman.nextDirectionText = "left"
    end
    if isRightPressed then
        pacman.nextDirection = 2
        pacman.nextDirectionText = "right"
    end
```

1 - Recebimento de Input do Jogador

```
pacman.upFree = false
pacman.downFree = false
pacman.leftFree = false
pacman.rightFree = false
if map[pacman.mapX][pacman.mapY - 1] == false then
    pacman.upFree = true
end
if map[pacman.mapX][pacman.mapY + 1] == false then
    pacman.downFree = true
end
if map[pacman.mapX - 1][pacman.mapY] == false then
    pacman.leftFree = true
end
if map[pacman.mapX + 1][pacman.mapY] == false then
    pacman.rightFree = true
end

if pacman.mapY == 10 then
    if (pacman.mapX == 13) or (pacman.mapX == 14) then
        pacman.downFree = false
    end
end
```

2 - Verificação das Paredes e Limites de Movimento

```
pacman.movement = delta * pacman.speed
if pacman.direction == 1 then
    if pacman.upFree then
        pacman.y = pacman.y - pacman.movement
    end
    if pacman.y > (pacman.mapY * gridSize + 0.5 * gridSize) then
        pacman.y = pacman.y - pacman.movement
    end
end

if pacman.direction == 3 then
    if pacman.downFree then
        pacman.y = pacman.y + pacman.movement
    end
    if pacman.y < (pacman.mapY * gridSize + 0.5 * gridSize) then
        pacman.y = pacman.y + pacman.movement
    end
end

if pacman.direction == 2 then
    if pacman.rightFree then
        pacman.x = pacman.x + pacman.movement
    end
    if pacman.x < (pacman.mapX * gridSize + 0.5 * gridSize) then
        pacman.x = pacman.x + pacman.movement
    end
end

if pacman.direction == 4 then
    if pacman.leftFree then
        pacman.x = pacman.x - pacman.movement
    end
    if pacman.x > (pacman.mapX * gridSize + 0.5 * gridSize) then
        pacman.x = pacman.x - pacman.movement
    end
end
```

```
if pacman.direction ~= pacman.nextDirection then
    if (math.abs(pacman.x - (pacman.mapX * gridSize + 0.5 * gridSize))
    < 2.5) and (math.abs(pacman.y - (pacman.mapY * gridSize + 0.5 * gridSize))
    < 2.5) then
        if (pacman.nextDirection == 1) and pacman.upFree then
            pacmanChangeDirection()
        end
        if (pacman.nextDirection == 3) and pacman.downFree then
            pacmanChangeDirection()
        end
        if (pacman.nextDirection == 2) and pacman.rightFree then
            pacmanChangeDirection()
        end
        if (pacman.nextDirection == 4) and pacman.leftFree then
            pacmanChangeDirection()
        end
    end
end
```

3 e 4 - Movimentação do PacMan com base na direção atual e se é possível mudar a direção

```
function pacmanChangeDirection()
    pacman.direction = pacman.nextDirection
    pacman.directionText = pacman.nextDirectionText
    pacman.x = pacman.mapX * gridSize + gridSize * 0.5
    pacman.y = pacman.mapY * gridSize + gridSize * 0.5
end
```

5 – Por fim, muda a direção do PacMan

- **Abordagem Técnica**: Uso de input (teclas do teclado), colisões com paredes e limites do mapa.

2. Sistema de Pontuação

- **Descrição**: Lógica de como a pontuação é calculada quando o Pacman come pílulas normais, pílulas de poder, e fantasmas (durante o modo assustado).

```
-- checa se pacman está em cima de um ponto e o exclui
if (dots[pacman.mapX][pacman.mapY] == true) then
    dots[pacman.mapX][pacman.mapY] = false
    score = score + 1
    collectedDots = collectedDots + 1
end

-- ^^ ponto especial
for i = 1, 4, 1 do
    if (specialDots[i].x == pacman.mapX) and (specialDots[i].y == pacman.mapY) then
        if specialDots[i].active then
            specialDots[i].active = false
            pacman.specialDotActive = true
            score = score + 10
        end
    end
end

-- ativação do efeito do ponto especial
if pacman.specialDotActive then
    if pacman.specialDotTimer > specialDotMaxTime then
        pacman.specialDotTimer = 0
        pacman.specialDotActive = false
        for i = 1, 4, 1 do
            ghosts[i].eaten = false
        end
    else
        pacman.specialDotTimer = pacman.specialDotTimer + delta
    end
end
```

6 - Pontuação por comer Pílulas Normais, Pílulas Especiais e efeito temporário das Pílulas Especiais

- **Abordagem técnica**: Incremento de pontos e exibição do placar em tempo real.

3. Vida e Levels

- **Descrição**: Como o jogo gerencia a vida do Pacman e como ele reinicia uma nova rodada quando o Pacman é capturado ou todas as pílulas são comidas.

```
for i = 1, 4, 1 do
    if pacman.specialDotActive == false then
        ghosts[i].speed = ghosts[i].normSpeed
        if (ghosts[i].mapX == pacman.mapX) and (ghosts[i].mapY == pacman.mapY) then
            lives = lives - 1
            gameMode = "getReady"
            getReadyTimer = 0
            resetPacmanPosition()
            resetGhostsPosition()
        end
    end
```

7 – Verifica se o PacMan é pego pelo Fantasma e perde uma vida

```
function nextLevel()
    collectedDots = 0
    level = level + 1
    resetPacmanPosition()
    resetGhostsPosition()
    initMap()
    gameMode = "getReady"
    for i = 1, 4, 1 do
        ghosts[i].normSpeed = ghosts[i].normSpeed + 10
        ghosts[i].slowSpeed = ghosts[i].slowSpeed + 5
    end
    pacman.speed = pacman.speed + 2
    specialDotMaxTime = specialDotMaxTime - 0.5
end
```

8 - Avançar de Nível e Aumentar a Dificuldade

```
if lives == 0 then
    gameMode = "gameOver"
end
if collectedDots == 236 then
    nextLevel()
end
```

9 – Define se o fim da rodada foi Game Over ou Next Level

4. Lógica das Pílulas de Poder

- **Descrição**: Função que controla o comportamento das pílulas de poder, como mudar o estado dos fantasmas e o tempo de duração da invencibilidade.

```
for i = 1, 4, 1 do
    if (specialDots[i].x == pacman.mapX) and (specialDots[i].y == pacman.mapY) then
        if specialDots[i].active then
            specialDots[i].active = false
            pacman.specialDotActive = true
            score = score + 10
        end
    end
end

-- ativação do efeito do ponto especial
if pacman.specialDotActive then
    if pacman.specialDotTimer > specialDotMaxTime then
        pacman.specialDotTimer = 0
        pacman.specialDotActive = false
        for i = 1, 4, 1 do
            ghosts[i].eaten = false
        end
    else
        pacman.specialDotTimer = pacman.specialDotTimer + delta
    end
end
```

10 - Interação com Pílulas de Poder e Temporização da Invencibilidade

```
if pacman.specialDotActive then
    ghosts[i].speed = ghosts[i].slowSpeed
    if (ghosts[i].mapX == pacman.mapX) and (ghosts[i].mapY == pacman.mapY) then
        if ghosts[i].eaten == false then
            ghosts[i].eaten = true
        end
    end
end
```

11 - Interação com Pílulas de Poder e Temporização da Invencibilidade

```
if pacman.specialDotActive then
    ghosts[i].speed = ghosts[i].slowSpeed
    if (ghosts[i].mapX == pacman.mapX) and (ghosts[i].mapY == pacman.mapY) then
        if ghosts[i].eaten == false then
            ghosts[i].eaten = true
        end
    end
end
```

12 - Comportamento dos Fantasmas durante a invencibilidade do Pacman, que se tocar um fantasma ele é "comido"

Demonstrando a Programação Orientada a Objeto no código:

- **Polimorfismo**

O polimorfismo é um princípio da programação orientada a objetos que permite que diferentes objetos respondam de maneiras distintas à mesma ação ou método. Isso significa que funções ou métodos podem ser aplicados a diferentes tipos de objetos, com comportamentos ajustados de acordo com as características específicas de cada um.

Dessa forma, o polimorfismo promove flexibilidade e reutilização de código, permitindo que um mesmo método funcione de maneira genérica para diferentes objetos, mas com implementações personalizadas para cada um.

1. **Movimentação do PacMan:**

- No código de PacMan, o movimento ocorre com base na direção atual e na velocidade, conforme mostrado no trecho abaixo:

```
pacman.movement = delta * pacman.speed
if pacman.direction == 1 then
    if pacman.upFree then
        pacman.y = pacman.y - pacman.movement
    end
    if pacman.y > (pacman.mapY * gridSize + 0.5 * gridSize) then
        pacman.y = pacman.y - pacman.movement
    end
end

if pacman.direction == 3 then
    if pacman.downFree then
        pacman.y = pacman.y + pacman.movement
    end
    if pacman.y < (pacman.mapY * gridSize + 0.5 * gridSize) then
        pacman.y = pacman.y + pacman.movement
    end
end

if pacman.direction == 2 then
    if pacman.rightFree then
        pacman.x = pacman.x + pacman.movement
    end
    if pacman.x < (pacman.mapX * gridSize + 0.5 * gridSize) then
        pacman.x = pacman.x + pacman.movement
    end
end

if pacman.direction == 4 then
    if pacman.leftFree then
        pacman.x = pacman.x - pacman.movement
    end
    if pacman.x > (pacman.mapX * gridSize + 0.5 * gridSize) then
        pacman.x = pacman.x - pacman.movement
    end
end
```

2. Movimentação do Fantasma:

- Os fantasmas usam uma lógica muito semelhante para se mover com base na direção atual e velocidade, mas com suas próprias variáveis e condições:

```lua
function moveGhosts(ghost)
    local movement = ghosts[ghost].speed * delta
    if ghosts[ghost].direction == 1 then
        if ghosts[ghost].upFree then
            ghosts[ghost].y = ghosts[ghost].y - movement
        end
        if ghosts[ghost].y > (ghosts[ghost].mapY * gridSize + 0.5 * gridSize) then
            ghosts[ghost].y = ghosts[ghost].y - movement
        end
    end
    if ghosts[ghost].direction == 3 then
        if ghosts[ghost].downFree then
            ghosts[ghost].y = ghosts[ghost].y + movement
        end
        if ghosts[ghost].y < (ghosts[ghost].mapY * gridSize + 0.5 * gridSize) then
            ghosts[ghost].y = ghosts[ghost].y + movement
        end
    end
    if ghosts[ghost].direction == 2 then
        if ghosts[ghost].rightFree then
            ghosts[ghost].x = ghosts[ghost].x + movement
        end
        if ghosts[ghost].x < (ghosts[ghost].mapX * gridSize + 0.5 * gridSize) then
            ghosts[ghost].x = ghosts[ghost].x + movement
        end
    end
    if ghosts[ghost].direction == 4 then
        if ghosts[ghost].leftFree then
            ghosts[ghost].x = ghosts[ghost].x - movement
        end
        if ghosts[ghost].x > (ghosts[ghost].mapX * gridSize + 0.5 * gridSize) then
            ghosts[ghost].x = ghosts[ghost].x - movement
        end
    end
end
```

3. Comparação:

Ambos os trechos de código compartilham a mesma estrutura de movimentação, mas com entidades diferentes (PacMan e fantasmas). Esse é um exemplo de **polimorfismo**, onde a mesma lógica de movimentação é reutilizada com variações no comportamento para cada entidade (usando diferentes variáveis, como pacman.speed e ghosts[ghost].speed):

- **PacMan:** Muda de posição (x, y) com base na velocidade (pacman.speed), direção (pacman.direction), e checagem de liberdade de movimento (e.g., pacman.upFree).

- **Fantasmas:** Aplicam a mesma lógica de movimento com suas variáveis específicas (e.g., ghosts[ghost].upFree, ghosts[ghost].speed), mas também lidam com situações especiais, como se PacMan ativou um ponto especial ou não.

Herança

Herança é um dos pilares da POO, onde uma classe "filha" pode herdar atributos e comportamentos de uma classe "pai". Isso permite o reuso de código e a extensão de funcionalidades de uma maneira organizada.

A classe "filha" pode, além de herdar, sobrescrever ou adicionar novos comportamentos. No contexto de jogos, isso pode ser utilizado para definir comportamentos gerais de personagens, como movimentação, e permitir que outros personagens mais específicos, como fantasmas ou o PacMan, compartilhem essas funcionalidades.

No código, a estrutura de herança pode ser vista onde a classe Ghost herda funções e atributos de PacMan, compartilhando comportamentos como movimentação e controle de direções.

- **Atributos compartilhados (simulação de herança):**

Os atributos comuns de Pac-Man e dos fantasmas incluem: x, y, mapX, mapY, direction, nextDirection, speed, upFree, downFree, leftFree, e rightFree. Estes atributos controlam as coordenadas no mapa, direção, e a velocidade do movimento.

A função de movimentação é similar para ambos. O comportamento de mover-se com base na direção e na verificação de obstáculos funciona da mesma forma.

1. **Movimentação do PacMan:**

```
pacman.movement = delta * pacman.speed
if pacman.direction == 1 then
    if pacman.upFree then
        pacman.y = pacman.y - pacman.movement
    end
    if pacman.y > (pacman.mapY * gridSize + 0.5 * gridSize) then
        pacman.y = pacman.y - pacman.movement
    end
end
```

2. **Movimentação do Fantasma:**

```
if ghosts[ghost].direction == 1 then
    if ghosts[ghost].upFree then
        ghosts[ghost].y = ghosts[ghost].y - movement
    end
    if ghosts[ghost].y > (ghosts[ghost].mapY * gridSize + 0.5 * gridSize) then
        ghosts[ghost].y = ghosts[ghost].y - movement
    end
end
```

3. **Comparação:**
   O conceito de herança é simulado ao reutilizar atributos e funções comuns entre PacMan e os fantasmas. Ambos compartilham atributos para controlar sua posição e lógica de movimentação, ainda que sejam implementados de forma separada.

- ## Endereçamento:

O conceito de endereçamento em programação orientada a objetos, especialmente em Lua, se refere ao fato de que tabelas (objetos) são passadas e manipuladas por referência.

Isso significa que alterações feitas em uma tabela dentro de uma função ou trecho de código afetarão a tabela original, pois ambos os pontos de referência (o original e o manipulado) apontam para o mesmo endereço de memória.

```
if (dots[pacman.mapX][pacman.mapY] == true) then
    dots[pacman.mapX][pacman.mapY] = false
    score = score + 1
    collectedDots = collectedDots + 1
end

-- ^^ ponto especial
for i = 1, 4, 1 do
    if (specialDots[i].x == pacman.mapX) and (specialDots[i].y == pacman.mapY) then
        if specialDots[i].active then
            specialDots[i].active = false
            pacman.specialDotActive = true
            score = score + 10
        end
    end
end
```

1. Comparação:

- **Manipulação Direta de Tabelas:**

dots[pacman.mapX][pacman.mapY] e specialDots[i] são tabelas manipuladas diretamente no código. Como as tabelas são passadas por referência, qualquer modificação nesses objetos diretamente afeta as tabelas originais.

- **Alterações Refletidas em Tempo Real:**

Quando você altera o valor de dots[pacman.mapX][pacman.mapY] para false, essa mudança é refletida imediatamente em todos os locais que referenciam dots. Assim, a alteração de um ponto no mapa é refletida no estado geral do jogo.

Similarmente, quando specialDots[i].active é alterado para false, essa mudança afeta o estado do ponto especial na tabela specialDots. A tabela specialDots é manipulada diretamente, e qualquer alteração se reflete em todas as referências a essa tabela.

- **Atualização do Estado Global:**

As variáveis score e collectedDots são atualizadas com base nas alterações feitas nas tabelas. Como essas variáveis estão fora das tabelas e são modificadas diretamente, elas refletem o impacto das mudanças feitas em dots e specialDots.