

Universidade Federal do Rio Grande

Henrique Bertochi - 162647

Vicenzo Copetti – 162..

## 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?**

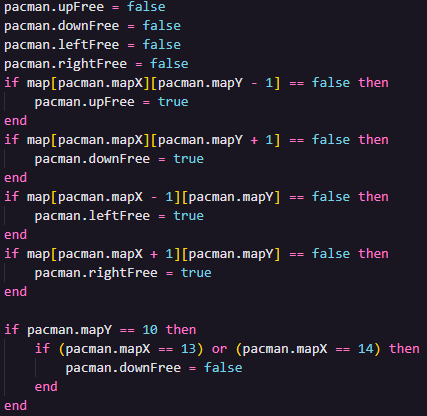
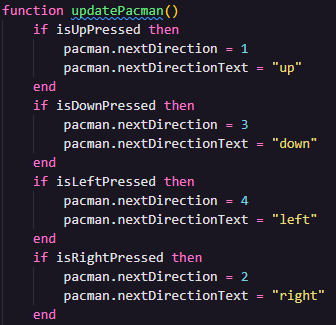
O PacMan é um clássico mundialmente reconhecido, e nos esforçamos para recriar o jogo com a maioria das funcionalidades. As funções implementadas incluem:

* 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

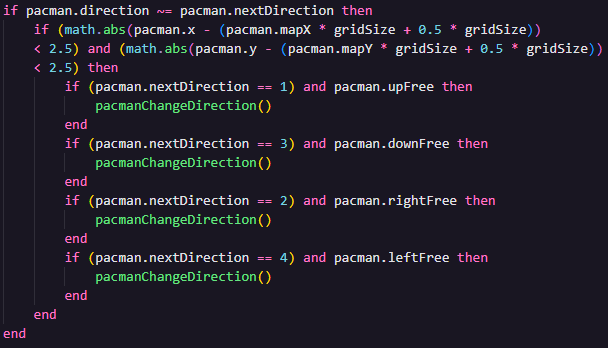
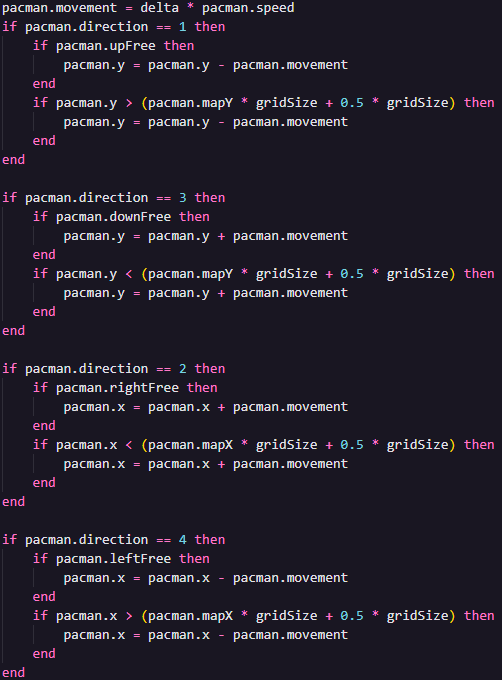
**Principais Funcionalidades do Software:**

### **Lógica de Movimentação do Pacman**

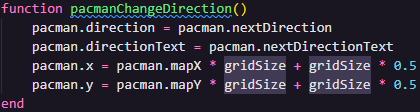
* **Descrição**: Parte do código responsável pela movimentação do Pacman, demonstrando como ele responde aos comandos do jogador (cima, baixo, esquerda, direita).



1 - Recebimento de Input do Jogador 2 - Verificação das Paredes e Limites de Movimento



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

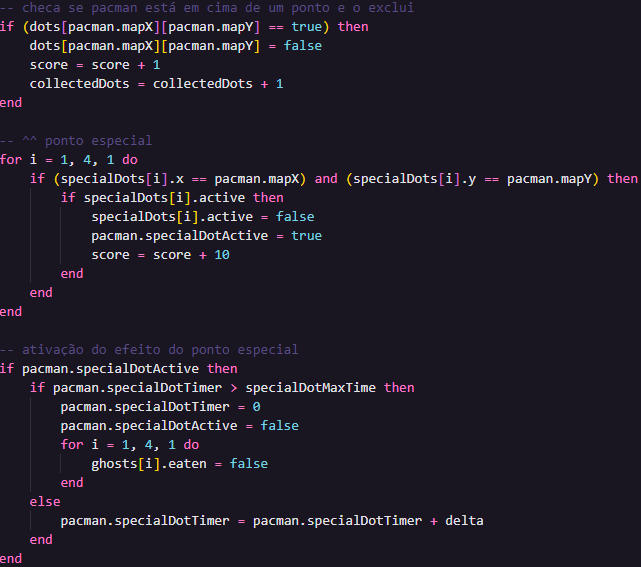


5 – Por fim, muda a direção do PacMan

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

1. **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).

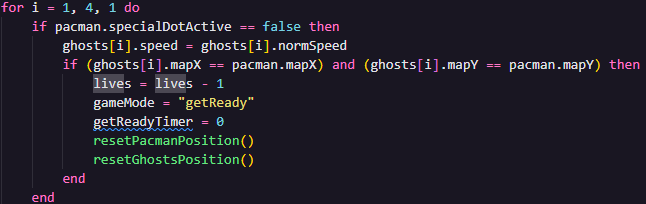


6 - Pontuação por comer Pílulas Normais, Pílulas Especiais e efeito temporário das Pílulas Especiais

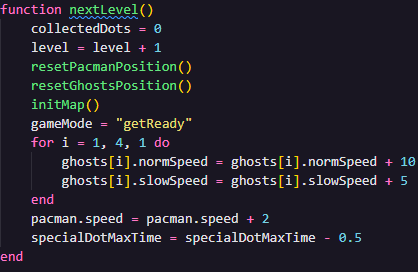
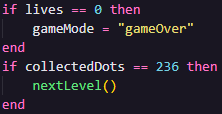
* **Abordagem técnica**: Incremento de pontos e exibição do placar em tempo real.

1. **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.



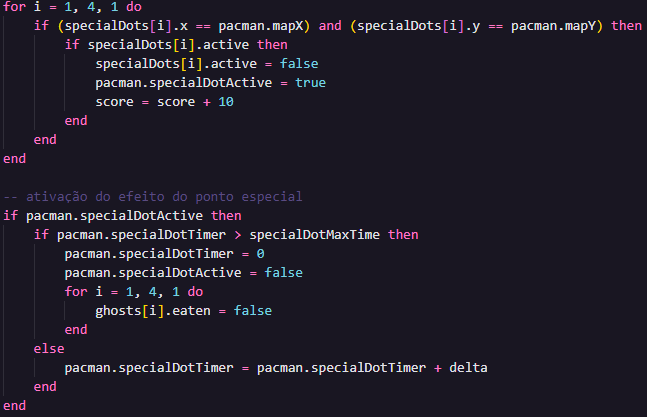
7 – Verifica se o PacMan é pego pelo Fantasma e perde uma vida

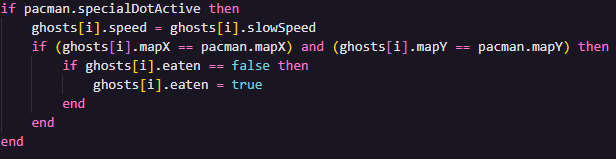
8 - Avançar de Nível e Aumentar a Dificuldade 9 – Define se o fim da rodada foi Game Over ou Next Level

1. **Lógica das Pílulas de Poder**

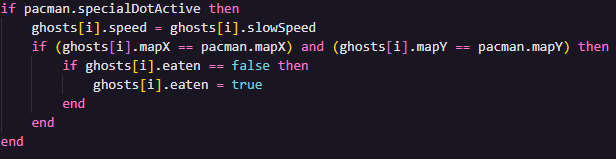
* **Descrição**: Função que controla o comportamento das pílulas de poder, como mudar o estado dos fantasmas e o tempo de duração da invencibilidade.

****

**10 -** Interação com Pílulas de Poder e Temporização da Invencibilidade



**11 -** Interação com Pílulas de Poder e Temporização da Invencibilidade

****

**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:**

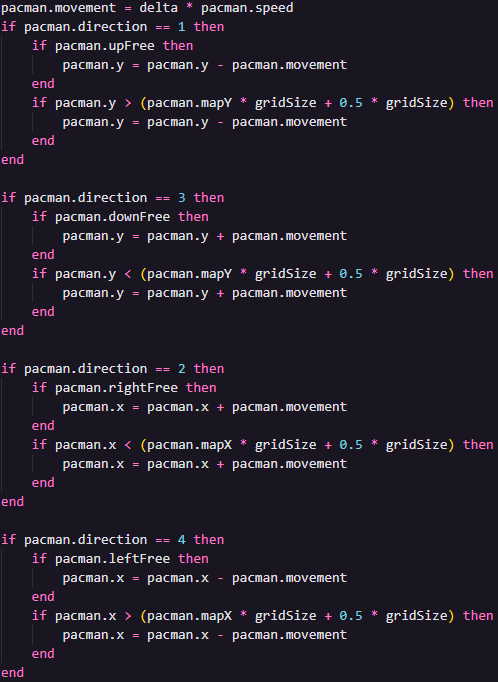
* **Polimorfismo**

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:



1. **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:



### **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.

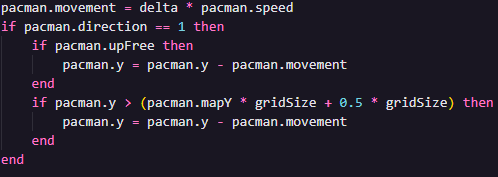
No código, a estrutura de herança pode ser vista onde a classe Ghost herda funções e atributos de PacMan, compartilhando comportamentos como movimentação e controle de direções.

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

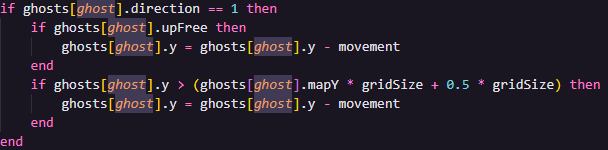
Os atributos comuns de Pac-Man e dos fantasmas incluem: x, y, mapX, mapY, direction, nextDirection, speed, upFree, downFree, leftFree, e rightFree. Estes atributos controlam as coordenadas no mapa, direção, e a velocidade do movimento.

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:**

****

1. **Movimentação do Fantasma:**

****

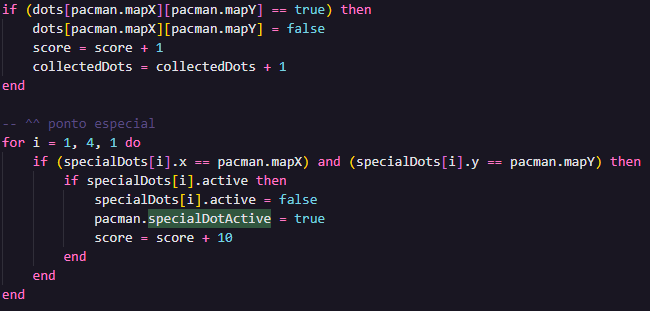
1. **Comparação:**

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.

****

### **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.