Universidade Federal de Viçosa
Campus Rio Paranaíba
Sistemas de Informação
Prof. Guilherme Pena

SIN141
Computação Orientada a Objetos
Trabalho Prático
07/03/2018

PacMan UFV-CRP

1. Mensagem inicial

Sugiro que comecem o quanto antes possível e não deixem para ultima hora. Este trabalho servirá para "allegrar" vocês um pouco e muitas das ferramentas que vocês usarão para implementá-lo, vocês provavelmente terão de retirar as dúvidas na Internet.

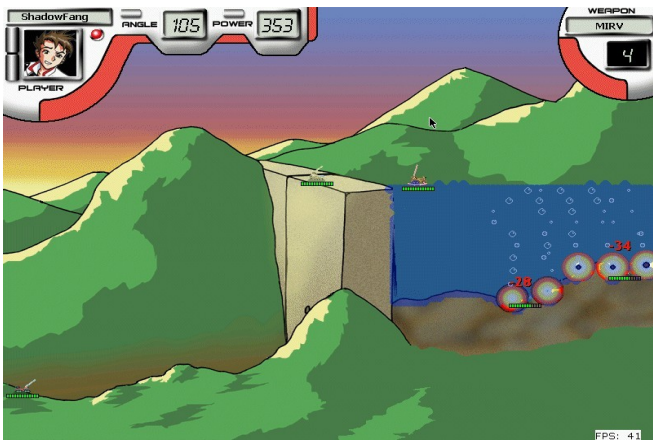
Neste caso, eu também estarei alerta em relação a qualquer tipo de fraude então CUIDADO, porque uma ação errada poderá prejudicar seriamente o seu grupo ou vários grupos.

A ideia princial do trabalho consiste no uso de **classes e seus principais conceitos** (construtores, destrutores, métodos, herança, polimorfismo ...) para a implementação de um jogo similar ao *PacMan* com auxílio das funções prontas da biblioteca gráfica ***Allegro*** através da linguagem C++ no Linux.

2. Allegro

Allegro é uma biblioteca livre de código fonte aberto para o desenvolvimento de Video games. O seu principal uso é no escopo da programação de jogos. Atualmente ela possui uma grande comunidade pois além de possuir diversos recursos nativamente (gráficos 2D, 3D com OpenGL, entrada de dados pelo teclado e mouse, RLE-Sprites, exibição de vídeos e controle de som) a API é bastante extensível fazendo que com existam diversos *addons* disponíveis. *[Fonte: Wikipedia].*

Exemplos de telas de jogos feitos em Allegro: (fonte: www.allegro.cc *e* http://liballeg.org/)



Atualmente a biblioteca Allegro se encontra na versão 5.2.2, de dezembro de 2016 e no PVANet encontra-se um roteiro de instalação do Allegro para o **Linux Mint 17.2** além de um link para um manual contendo as funções do Allegro.

3. Descrição do Trabalho

O trabalho da disciplina consiste na implementação de um jogo similar ao PacMan (Figura 1). No PacMan um personagem em forma de disco se desloca por um labirinto com o objetivo de coletar todas as pílulas espalhadas pelo caminho, ao mesmo tempo em que evita contato com fantasmas que se movimentam no mesmo labirinto.

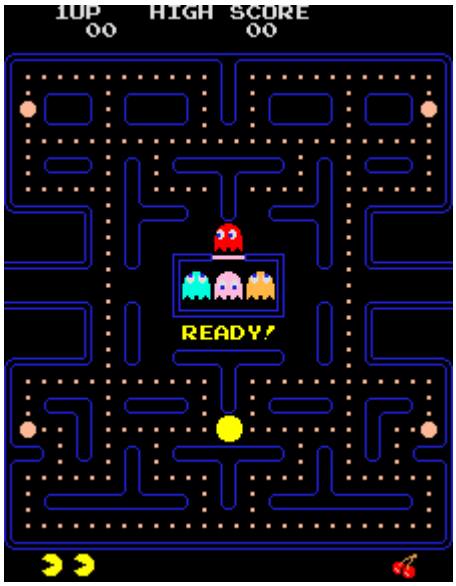


Fig. 1: Tela inicial da versão original do PacMan

O trabalho poderá ser implementado por grupos de até 3 alunos, e deverá utilizar a biblioteca ***Allegro*** utilizando a linguagem C++.

Os elementos do jogo PacMan serão baseados na carga de imagens que poderão ser de qualquer tema a gosto do grupo. Para posterior edição das imagens eu sugiro um programa editor como **GIMP**, **Inkscape** por exemplo.

Para que o trabalho possa ficar pronto até o final do semestre, consideramos algumas regras para o trabalho e para o jogo:

1. Todos os elementos que aparecerem no jogo deverão ser implementados através de **Classes** (as regras das classes serão descritas nas etapas). Além disso, **qualquer alteração** feita nos respectivos objetos deverá acontecer através de **funções-membro**.

2. Os elementos **pílulas** e **personagens** deverão modelar ***Sprites***, conjuntos de imagens que se sobrepõem dando uma sensação de movimento.

3. **o jogo termina quando não há mais pílulas no labirinto** (não é necessário criar várias fases) **ou quando o Pacman é atacado por um fantasma** (não é necessário dar novas chances, ter várias vidas, etc); (*O que não é pedido fica opcional*)

4. não existem as “frutas”, que aparecem de tempos em tempos no centro labirinto e valem ponto extra quando comidas pelo Pacman; (*Opcional*)

5. não existem as pílulas de força que quando comidas pelo Pacman tornam os fantasmas vulneráveis; (*Opcional*)

6. não existem os “túneis” que permitem sair de um lado da tela e aparecer no outro. (*Opcional*)

O trabalho será dividido em quatro etapas que deverão ser entregues nas datas estipuladas e contendo as regras de implementação solicitadas.

ATENÇÃO: A implementação do trabalho será evolutiva entre as etapas, isto é, caso algum grupo não tenha entregado a **etapa 1** por exemplo**,** na data estipulada, ele perderá os pontos daquela etapa. Além disso, na data de entrega da **etapa 2**, o trabalho deverá conter os requisitos das **etapas 1 e 2** para valer os pontos apenas da etapa 2. Todos os pontos serão julgados na entrega da última etapa (trabalho completo) de acordo com as entregas anteriores.

As etapas são descritas a seguir:

2.1. Etapa 1: Definição do Mapa e Classes Simples
Entrega: Até 20 de Abril.

Esta fase consiste na definição e construção de um mapa representando o labirinto e seus elementos básicos:

- **Tijolos**, que formam as paredes;
- **Pílulas** (*sprite*) distribuídas no labirinto;
- Células vazias, sem pílula.

Classe: Todos os elementos que estarão nessa etapa deverão ser modelados por **classes** com seus ***atributos*** e ***métodos específicos***. Além disso, deverão conter os ***construtores, destrutores***;

Ao ser executado, o programa deverá exibir uma representação gráfica para o mapa, que deverá formar um desenho como o apresentado na Figura 2 (sem os personagens). Todas as células do mapa que não conterem tijolos, deverão conter uma pílula.

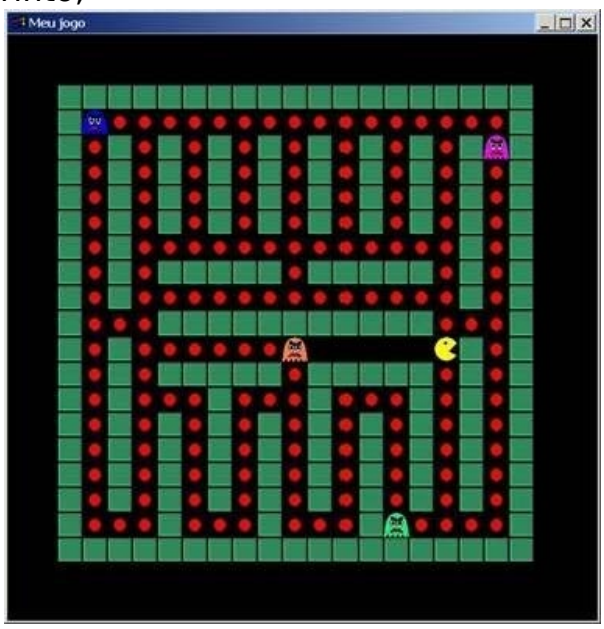


Fig 2. Tela Inicial do jogo

2.2. Etapa 2: Movimentação do PacMan e Coleta das Pílulas. *Entrega: Até 11 de Maio.*

Esta fase consiste na definição do comportamento do Pacman e na contagem de pontos. Ao ser executado, o programa deverá exibir o mapa construído na **etapa 1** e um elemento representando o personagem Pacman, que inicialmente deverá estar posicionado em algum ponto próximo ao centro do mapa.

Deve exibir também um placar (menu) indicando o número de pontos obtidos, que será equivalente ao número de células visitadas pelo personagem que continham uma pílula .

A movimentação do personagem deve seguir as regras descritas abaixo:

- O personagem não poderá atravessar uma parede.
- Quando o personagem passar sobre uma pílula, ela deve ser apagada do labirinto e o placar deverá registrar um aumento de uma unidade.
- Se não houver mais nenhuma pílula no labirinto, o jogo pode ser encerrado.
- A movimentação deve ser controlada pelas teclas de movimentação (“setas” do teclado).
- A direção de movimento poderá ser: PARA CIMA, PARA BAIXO, PARA ESQUERDA ou PARA DIREITA.
- O personagem estará sempre em um dos estados de movimento (“PARA CIMA”, “PARA BAIXO”, “PARA ESQUERDA” ou “PARA DIREITA”). Esse estado indica o próximo movimento do personagem, exceto quando o jogador altera a direção usando o teclado ou quando um obstáculo é encontrado.
- Cada estado será representado por um desenho diferente. Na Figura 3, por exemplo, o Pacman está no estado “PARA ESQUERDA”.
- Quando uma das setas for pressionada pelo jogador, o sistema deverá interpretar esse evento como uma “intenção” de movimentação do personagem.
- Se as condições do ambiente permitirem que o personagem altere sua direção de movimento de acordo com a “intenção” definida pelo jogador, ou seja, não há tijolo adjacente na direção escolhida, o sistema deverá executar essa alteração na direção.
- Se as condições do ambiente não permitirem que o personagem altere sua direção de movimento de acordo com a “intenção” definida pelo jogador, o personagem deverá continuar se deslocando na direção em que estava. Mas o sistema deverá armazenar a última “intenção” do jogador até que ela possa ser concretizada, ou até que o usuário altere essa “intenção”.

Para exemplificar a forma como o sistema deverá tratar a movimentação do personagem, observe o exemplo a seguir.

Na Figura 3, o Pacman está no estado “PARA ESQUERDA”. Se a última tecla pressionada pelo jogador tiver sido a tecla de movimentação “PARA ESQUERDA”, o Pacman vai se deslocar para a esquerda até encontrar um obstáculo.

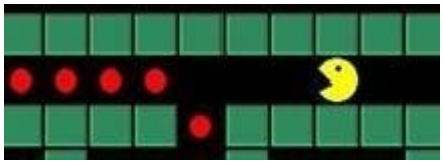


Fig 3: PacMan se deslocando para a esquerda

Suponha que, neste momento (situação descrita ainda pela Figura 3), o jogador pressione a tecla de movimentação “PARA BAIXO”. Com esse procedimento, o jogador o registra a intenção de mover o personagem para baixo. Como não é possível mover o Pacman para baixo, no contexto em que se encontra, então ele continua no estado “PARA ESQUERDA”. Mas o sistema registra a última intenção do jogador, que foi a de movimento para baixo.

Suponha que o jogador não pressione nenhuma tecla até que o Pacman atinja a posição exibida na Figura 4. O sistema continua com a intenção de movimento “PARA BAIXO” registrada, e nesse momento, é possível realizar o movimento desejado.

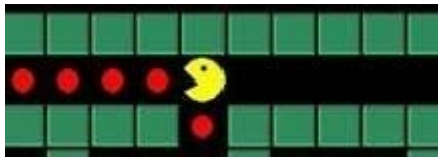


Fig 4: PacMan continuava se deslocando para a esquerda

A Figura 5 mostra o próximo movimento que o Pacman deve realizar, nas condições estabelecidas. Observe que o personagem moveu-se para baixo, apesar do jogador não estar pressionando a tecla correspondente, no mesmo momento.

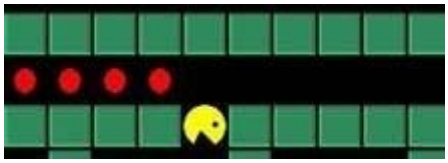


Fig. 5: Mudança de direção para baixo

A partir de então, o sistema continua registrando que a última intenção do jogador foi um movimento para baixo, até que uma tecla de movimentação seja pressionada novamente.

Classe: A **classe** do personagem nessa etapa também deverá possuir seus **atributos** e **métodos específicos para qualquer alteração ao objeto** e deverá conter os **construtores, destrutores e métodos get/set**;

Observação: *Para que o jogador possa controlar o Pacman de maneira satisfatória, o sistema pode ter que estabelecer um atraso no cálculo dos movimentos. Sem inserir esse atraso, os movimentos podem ficar rápidos demais para serem controlados.*

3.3. Etapa 3: Movimentação dos Fantasmas e Herança

Entrega: Até 30 de Maio.

O jogo inicia com 4 fantasmas posicionados nas extremidades do labirinto (ou no centro se você tiver feito um local apropriado). Cada fantasma deve ser pintado com uma cor diferente.

Se um fantasma move-se para uma célula no mesmo momento ocupada pelo Pacman, o jogo deve ser interrompido. Nesta fase, é suficiente terminar o programa, sem dar novas chances ao usuário.

O movimento dos fantasmas poderá ser de 2 tipos: **aleatório** e de **perseguição**.

O jogo deverá ter pelo menos um fantasma com cada tipo de movimento. Ou seja, é permitido ter 3 fantasmas com movimento aleatório e um fantasma com movimento de perseguição, 3 fantasmas com movimento de perseguição e um fantasma com movimento aleatório, ou 2 fantasmas com cada tipo de movimento.

No entando o movimento de perseguição de pelo menos um fantasma deverá ser implementado apenas na etapa 4. Para a etapa 3 todos os fantasmas poderão ter o movimento aleatório.

O movimento aleatório é caracterizado por:

- A direção de movimento poderá ser: PARA CIMA, PARA BAIXO, PARA ESQUERDA, PARA DIREITA.

- O fantasma estará sempre em um dos estados de movimento (“PARA CIMA”, “PARA BAIXO”, “PARA ESQUERDA” ou “PARA DIREITA”). Esse estado indica o próximo movimento do fantasma, exceto quando um obstáculo ou uma encruzilhada são encontrados.

- Como obstáculos, devem ser considerados os tijolos das paredes e outros fantasmas. Ou seja, um fantasma não pode “atravessar” outro.

- Uma encruzilhada é um ponto onde um fantasma pode ter mais de duas possibilidades de movimentos diferentes.

- Ao encontrar um obstáculo ou encruzilhada, o fantasma de movimento aleatório deve sortear uma nova direção de deslocamento, considerando apenas as direções válidas no contexto em que se encontra.

O exemplo a seguir mostra como funciona o esquema de encruzilhadas.

Suponha que o fantasma da Figura 6 realize um movimento do tipo aleatório e esteja no estado “PARA BAIXO”. Quando ele se movimentar três células para baixo, encontrará uma encruzilhada.

A Figura 7 marca a encruzilhada com um círculo amarelo. Observe que, na célula indicada, é possível realizar um movimento em três direções diferentes.

Quando chega à encruzilhada, um fantasma de movimento aleatório sorteia uma nova direção de deslocamento. No caso da Figura 7, a nova direção pode ser “PARA CIMA”, “PARA BAIXO” ou “PARA ESQUERDA”.

Sorteios semelhantes devem ser realizados quando um fantasma de movimento aleatório encontra um obstáculo ou outro fantasma.

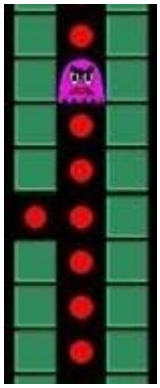


Fig. 6: Fantasma movendo-se para baixo

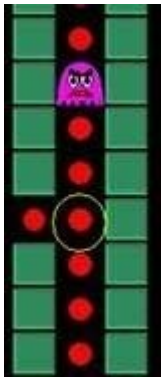


Fig. 7: Indicação de uma encruzilhada

Classe: Nesta etapa, as **classes** do personagem e dos fantasmas **compartilharão alguns atributos e métodos em comum**. Logo nesta etapa, essas duas classes deverão possuir o **relacionamento de herança** com uma classe ancestral que conterá os **membros** comuns.

Dessa forma, as implementações específicas do personagem e fantasmas deverão estar em suas respectivas classes. A classe dos fantasmas também deverá conter os **construtores, destrutores e métodos get/set**;

3.4. Etapa 4: Estratégia de Perseguição e Polimorfismo

Entrega: Até 15 de Junho.

O movimento de perseguição é caracterizado por deslocamentos no sentido de minimizar a distância entre o fantasma e o Pacman. Diversas estratégias podem ser seguidas, e cada grupo poderá implementar a estratégia que desejar.

Nesta etapa, um documento (TXT) explicando a estratégia de perseguição deverá ser redigido e a avaliação desta etapa levará em conta a qualidade da estratégia e sua documentação (texto explicando os mecanismos utilizados).

Classe: Nesta etapa, a **classe** do fantasma novamente deverá possuir o **relacionamento de herança** para fantasmas aleatórios e fantasmas inteligentes (com alguma perseguição) e a **função de movimentação** deverá assumir um comportamento **polimórfico**.

3.5 Informações Adicionais

Para que o jogo fique interessante, o sistema pode ter que estabelecer um atraso no cálculo dos movimentos dos fantasmas. É desejável que esse atraso seja maior que o atraso estabelecido para o PacMan, fazendo com que o PacMan seja mais veloz que os fantasmas.

Embora não seja obrigatório, você pode acrescentar outras características no jogo pra torná-lo mais interessante e diferente (ou mais parecido com o jogo original).

Se necessário, o grupo pode dividir as tarefas entre os integrantes. Entretanto, no momento da apresentação do trabalho ao professor, todos os integrantes devem estar presentes, e conhecendo o funcionamento do jogo.

4. Entrega

O desenvolvimento do trabalho é dividido em 4 etapas. Para cada uma das etapas, o grupo receberá a pontuação quando a entrega for feita no tempo e da forma correta em dois **ambientes (RunCodes e PVANet)**:

4.1 Instruções de Entrega

- Ambiente de entrega 1 (RunCodes): Compacte em um arquivo do tipo **(.zip)** todos os códigos do projeto naquela etapa, digo, apenas os arquivos **(.h)** e **(.cpp)** que foram implementados pelo grupo. Após a compactação, este zip deverá ser entregue por **apenas um integrante do grupo** no **RunCodes** no exercício disponível.

- Ambiente de entrega 2 (PVANet): Compacte em um arquivo todos os documentos pertinentes do trabalho naquela etapa (**arquivos necessários para compilação e execução, códigos e imagens**). Após a compactação, este arquivo deverá ser entregue por **apenas um integrante do grupo** no **PVANet** no exercício disponível.

Na última fase, além de entregar o projeto completo e o documento que descreve a estratégia de implementação. Cada grupo deverá apresentar o trabalho para o professor que fará uma entrevista individual e em grupo em alguma **data e local a serem estipulados**.

5. Considerações Finais

Desejo a todos um bom trabalho, muitas dúvidas serão geradas no processo, empenham-se para resolvê-las em equipe usando o livro, as aulas, a internet.

Por fim, deixo os meus Agradecimentos ao Prof. André Gustavo dos Santos. (DPI - UFV / Viçosa).

Bom Trabalho!
Divirtam-se!