UNIVERSIDADE FEDERAL DO ABC
BACHARELADO EM CIÊNCIAS E TECNOLOGIA
MCTA016 - PARADIGMAS DA PROGRAMAÇÃO

Edson Gomes Martinelli R.A: 11097416
Rafael Akio Shishito Matos R.A: 11079516

Projeto de Programação - Pacman simplificado desenvolvido em Haskell

Projeto solicitado pelo
Profº Emilio Francesquini
apresentado à disciplina de
Paradigmas da Programação

Santo André - SA
25 de agosto de 2019

1 Proposta de projeto

1.1 Problema

Dado nossos conhecimentos obtidos através das aulas e de pesquisas individuais, desenvolver, com qualidade e complexidade, um programa a fim de aplicar esse conhecimento em um projeto real, aprendendo mais tanto de Haskell especificamente, quanto de programação em geral através de técnicas usadas em uma linguagem funcional.

O projeto escolhido foi a implementação de um Pacman em sua versão simplificada, não contando algumas funcionalidades do jogo original.

1.2 Proposta de Implementação

Nesse projeto visamos implementar uma versão em Haskell do jogo Pacman do ano de 1980 utilizando a API gráfica Gloss que possui diversas funcionalidades que escondem a implementação pura do OpenGL, assim, evitando problemas que poderiam acontecer por falta de conhecimento com o binding do OpenGl para Haskell.

O projeto será uma versão simplificada, não possuindo contador de vidas, sendo assim, o jogador não terá uma segunda chance após a primeira morte, e o jogo deverá ser reiniciado.

O objetivo final do jogo é a captura de todos os pontos (cada pontos tem o valor 1) disponíveis no mapa, esse que poderá ser criado e modificado antes de cada partida através de um arquivo de texto presente no projeto e que funcionará perfeitamente caso algumas regras para sua criação sejam seguida: a criação de um Pacman através da indicação "P"e de todos os fantasmas com a indicação "C",para Clyde, "R", para Pinky, "I", para Inky e "B"para Blinky. As paredes podem ser feitas utilizando o marcador "X"e os caminhos como " ", assim como pontos com ".". O indicador V é dado para caminhos que o Pacman não poderá passar, como a gaiola dos fantasmas e o "G"indica o portão dessa gaiola que deverá apontar para o norte. É necessário também que o mapa seja retangular.

O jogador, Pacman, se movimentará a partir das setas do teclado não possuindo nenhum outro comando ou função de pausa para alcançar seu objetivo evitando encostar em qualquer fantasma.

Diferentemente da versão original do jogo, nesse projeto, o jogador não poderá obter as frutas, que aumentaram sua pontuação, ou pílulas energéticas que permitem que o Pacman consuma os fantasma por um curto período de tempo e aumente seus pontos.

Também com relação aos fantasmas e suas diferenças ao jogo original, nosso projeto não dará uma inteligencia artificial diferente para cada fantasma, assim, seus movimentos serão aleatórios mas com fluidez, não travando em paredes ou outros problemas que prejudiquem sua movimentação.

2 Compilando e rodando

Para compilar e executar apenas entre na pasta do projeto e execute os comandos:

```
stack build
stack exec projeto
```

Para executar os testes apenas entre na pasta do projeto e execute o comando:

```
stack test
```

3 Aspectos gerais

No decorrer do projeto tivemos diversos problemas sendo maioria deles decorrentes do fato de nunca termos utilizado a API Gloss e a falta de conhecimento com o Stack. Outro fator que complicou o processo de criação foi que tivermos que mudar o projeto muitas vezes conforme aprendíamos nooas conceitos do Haskell, assim, deixando sua primeira versão irreconhecível se comparado com a final.

Nosso primeiro grande problema com o a API Gloss foi a utilização de um modelo cartesiano não convencional por parte dela, o que dificultou a utilização da função Pictures, que une toda uma Lista de Picture passadas para ela. Para resolver isso, foi preciso a criação de uma fórmula para a conversão do sistema cartesiano do Gloss para o convencional.

Depois da criação de uma imagem estática e com evolução para a criação de movimentos do jogo, foi preciso criar um sistema que detecção de paredes. Para fazer isso, foi algo relativamente simples, pois apenas pegamos os blocos de cima, baixo, esquerda e direita e criamos uma lista que depois será conferida. A grande jogada aqui, foi a não remoção de blocos paredes dessa lista, mas sim a criação de um bloco nulo que substituirá essas paredes, o que permitiu que as posições fossem pegas ainda pelo ID da lista, mesmo após o retorno dos blocos válidos, sem a necessidade de conferir a posição desse bloco com relação ao Pacman.

Após isso, foi preciso entender o sistema de update utilizado pela função play, que, após alguns testes, se provou ineficaz para a detecção de colisão do jogo. Isso ocorre pelo fato da imagem só ser atualizada no próximo update após a detecção, dessa forma, precisamos "prever"o movimento do Pacman ou Fantasma, que é feito sabendo a velocidade atual, tupla que possui velocidade em x e em y. Ao prever uma posição a frente podemos parar o Pacman ou Fantasma exatamente no meio do bloco em caso de colisão.

Outro problema, agora relacionado com a jogabilidade do jogo, foi na movimentação do Pacman. De forma simplificada o Pacman se move nas direcionais assim que uma tecla é apertada e ele possa ir para essa direção (caso essa não seja uma parede, por exemplo), mas, devido a uma decisão do grupo, o Pacman só poderá se movimentar para outra direção quando estiver no meio de um bloco, o que evita que ele entre dentro de paredes em quinas, mas faz com que o jogo adquira uma característica de "frame perfect"ao tentar alterar a direção do Pacman, pois é preciso que a tecla seja apertada no exato momento que esse está no meio do bloco. Para solucionar esse problema foi preciso guardar a direção desejada pelo jogador e, quando possível, ir para tal direção. Esse sistema também existe no Pacman original e para cria-lo apenas criamos mas um

campo na data Pacman e agora ao clicar em uma tecla, apenas a velocidade desejada será atualizada e não sua velocidade atual.

Outro desafio adotado foi a criação de sprites para o fantasmas e Pacman, e tileset para o mapa. No caso do mapa, verificamos os 8 blocos adjacentes ao bloco que queremos decidir a imagem e com base nisso, cortamos o tileset na posição certa. As sprites são cortadas e escolhidas com base na velocidade do Pacman e dos fantasmas (as sprites dos fantasmas e do Pacman estão separadas) e acrescentadas ao campo de data respectivo.

Com relação ao Stack, o maior problema foi a configuração de algumas API's que foram utilizadas em algum momento do projeto, como o Juicy-Pixel, pois queríamos utilizar GIFs ao invés de sprites inicialmente, e o gloss-juicy que faz a interligação entre o Juicy-Pixel e o Gloss, onde, diferente do próprio Gloss, precisávamos configurar o stack.yaml, mas como não sabíamos disso, demoramos muito para conseguir utiliza-los. Essas API's não estão mais no projeto final.

Concluindo, desenvolvemos o jogo usando os conceitos de Haskell e paradigmas funcionais aprendidos em aula. Analisando o código, percebemos que ele seria muito maior em outras linguagens de programação não funcionais como o java, sem sacrificar performance.