Relatório Trabalho 02 de L.F.A

Samuel Terra (0011946) Matheus Calixto (0011233)

[1]Instituto Federal de Ciência e Tecnologia de Minas Gerais
São Luiz Gonzaga, s/n - Formiga / MG - Brasil

calixtinn@gmail.com, samuelterra22@gmail.com

Resumo. *Este é um trabalho da disciplina de Linguagens Formais e Autômatos Finitos, que aborda a implementação de um jogo baseado no clássico Pac-Man, utilizando AFD's e Threads.*

1. Introdução

O problema proposto para este trabalho prático, foi criar um jogo baseado no Pac-Man, onde os elementos móveis dos mesmos (4 fantasmas e 1 pac-man), fossem controlados por um AFD. Além disso, era necessário cumprir alguns outros requisitos solicitados na especificação, como por exemplo, para cada fantasma, era necessário utilizar um AFD e uma Thread. Solicita também a criação de um elemento imóvel denominado Árbitro, que fica encarregado de gerenciar o jogo.

O programa foi construído na linguagem Python 2.7, por proporcionar facilidade de implementação, diversas funções de manipulação de objetos, interface gráfica e estruturas de dados, além de ser uma linguagem nativa do Sistema Operacional Linux, onde este trabalho foi implementado.

A estrutura do programa foi construída em cima do padrão MVC (*Model, View and Controller*) de orientação a Objetos. O MVC é um padrão de arquitetura de software onde realiza a separação da aplicação em três camadas. Com a camada *Model* é possível elaborar a modelagem dos objetos mais simples no sitema (ex. Automato, Estado, Transição). Já na camada *Controller*, é onde fica todas as regras de negócio, os métodos que realmente realizam todo o esforço com a implementação de todas as funcionalidades. E camada que é chamada de *View* é possível realizar a interação com o usuário, nela apenas é solicitado as informações de entrada e passadas para o *Controller* que é também instanciado.

A interação com o usuário é feita através de uma interface gráfica (labirinto) construída pelos integrantes do grupo, onde os jogadores controlam o Pac-Man através das setas do teclado. Não há nada em modo texto para o usuário, apenas no modo gráfico.

2. Implementação

A implementação do trabalho foi realizada pelos dois alunos de maneira online, que utilizaram recursos como: IDE *PyCharm* e controle de versão. A divisão das tarefas foi realizada de maneira igual e justa entre os integrantes, o que contribuiu de maneira excelente para o bom andamento do trabalho. As dificuldades foram solucionadas rapidamente através da troca de ideias, e as decisões de implementação foram discutidas de maneira saudável.

2.1. A Classe AFD

Do que diz respeito ao código, o objeto AFD foi construído a partir de outros objetos: Estados (*States*) e Transições (*Transitions*). O objeto *State*, que representa um estado de um autômato, possui os seguintes atributos:

- **ID:** Um número inteiro salvo como carácter, que é a identificação do estado.
- **Name:** O nome do estado.
- **PosX:** Um número real, representando a coordenada do eixo X referente à posição do estado no plano cartesiano do software JFLAP.
- **PosY:** Um número real, representando a coordenada do eixo Y referente à posição do estado no plano cartesiano do software JFLAP.
- **Initial:** Uma flag booleana, indicando se o estado é um estado inicial (*True*) ou não (*False*).
- **Final:** Uma flag booleana, indicando se o estado é um estado final (*True*) ou não (False).

Já o objeto Transition que também possui uma classe própria, assim como o objeto *State*, representa as transições entre os estados desse AFD. Cada transição possui os seguintes atributos:

- **ID:** Um número inteiro salvo como carácter, que é a identificação da transição.
- **From:** Um número inteiro salvo como caractere, que indica o estado de partida da transição.
- **To:** Um número inteiro salvo como caractere, que indica o estado de destino da transição
- **Read:** Um caractere que é consumido ao se realizar uma transição de um estado a outro.

Por fim, através desses objetos, o objeto AFD, que representa o autômato, é construído. A classe AFD possui os seguintes atributos:

- **States:** Uma lista de objetos do tipo *State*, que comporta todos os estados do AFD.
- **Trasitions:** Uma lista de objetos do tipo *Transition*, que comporta todas as transições do AFD.
- **Initial:** Um número inteiro, salvo como caractere, que representa o estado inicial do AFD.
- **Finals:** Uma lista de caracteres, contendo o ID de todos os estados que são finais.
- **Alphabet:** Uma lista contendo todos os caracteres que fazem parte do alfabeto do referido AFD.

Com esses objetos, conclui-se a constituição da interface *Model* do modelo MVC, e com isso a primeira parte do trabalho que era criar uma classe que representasse um Autômato, foi concluída.

2.2. Manipulação do AFD

Assim como no primeiro trabalho prático, foram utilizadas algumas funções para que o autômato de cada fantasma funcionasse. Na classe AFDController, foram utilizadas basicamente as funções de carregamento do autômato a partir do software JFLAP, e a função de movimentação dentro do AFD a partir de um dado estado.

2.3. Elementos Utilizados

Para o desenvolvimento do trabalho, foram utilizadas algumas bibliotecas disponíveis na linguagem Python, tais como: threading, math, time, pygame, copy, random. Pode-se dar um destaque maior para a biblioteca PyGame, que foi utilizada para se construir a interface gráfica do jogo.

Através do PyGame, foi possível construir o labirinto e todos os objetos presentes nele: cápsulas, barreiras, bordas, placares, fantasmas, o próprio Pac-Man, entre outros elementos. Todos esses objetos foram construídos manualmente, definindo a sua posição no plano cartesiano, o que gastou um tempo considerável para ser finalizado. Além disso, o PyGame possibilita a reprodução de arquivos .wav dentro do jogo, bem como carregar imagens para serem utilizadas no projeto. O PyGame oferece uma imensa quantidade de recursos para a criação de jogos.

Além disso, foi requisitada a utilização de Threads para controlar os elementos do jogo. Cada fantasma possui um AFD e uma Thread, no entanto, cada um possui uma inteligência diferente:

- **Fantasma Vermelho:** O fantasma vermelho, se movimenta no labirinto de forma randômica. Porém em um intervalo de 4 em 4 segundos, o Árbitro envia a localização do Pac-Man ao fantasma, que calcula, através da fórmula da distância entre dois pontos, a melhor direção para que, seu objetivo de pegar o Pac-Man seja alcançado.
- **Fantasma Azul:** Já o fantasma azul, movimenta-se por todo o tempo randomicamente.
- **Fantasma Laranja:** O fantasma laranja funciona de uma maneira mais simples, onde, sempre que o Pac-Man muda de direção, o mesmo também muda, porém para a direção contrária!!
- **Fantasma Roxo:** O fantasma Roxo, possui exatamente a mesma lógica do fantasma vermelho, porem ele recebe sinais do árbitro de 2 em 2 segundos, o que o torna mais propenso a capturar o Pac-Man.

2.4. O Árbitro

O árbitro é o elemento imóvel do jogo, que não é controlado por nenhum autômato. Ele é responsável pela gerência do jogo. Dentre as funcionalidades do Árbitro estão:

- Envio de sinais de posição absoluta do Pac-Man de tempos em tempos para os fantasmas vermelho e roxo.
- É ele quem dispara todas as threads
- É ele quem controla a quantidade de vidas do Pac-Man
- É ele quem reproduz os sons nos momentos corretos

O árbitro também gerencia a condição de vitória ou derrota do usuário no jogo.

2.5. Mecânica do Jogo

O labirinto é composto por barreiras, cápsulas e frutas, além dos fantasmas. Estes são os elementos nos quais o Pac-Man pode interagir. São um total de 155 cápsulas espalhadas por todo o mapa, e o usuário só vence o jogo caso consiga fazer com que o Pac-Man

coma todas as cápsulas. Para tal tarefa, ele possui um total de 3 tentativas, que são as vidas. O Pac-Man pode perder essas vidas encostando nos fantasmas. A cada vez que ele encosta em um fantasma, ele perde uma vida e tanto ele quanto os fantasmas, voltam para as posições iniciais, recomeçando assim o jogo.

No labirinto, há 3 elementos que dão ao Pac-Man, o poder de comer os fantasmas, as frutas! Assim que o Pac-Man come uma fruta no labirinto, ele fica invencível por um curto período de tempo (cerca de 10 segundos) e consegue destruir os fantasmas. Durante esse tempo, caso o Pac-Man, encoste em um dos fantasmas, é reproduzido um som que demonstra uma congratulação e o fantasma é eliminado do jogo.

O usuário não vence o jogo se comer todos os fantasmas, mas sim se comer todas as 155 cápsulas. Caso o Pac-Man perca todas as suas vidas antes de consumir essas cápsulas, o jogo mostra uma mensagem de Game Over, e então fecha. Caso contrário, o jogo mostra uma mensagem de Vitória e o jogo é fechado.

3. Conclusão

Com a realização deste trabalho, foi possível obter mais conhecimentos sobre a utilidade dos AFD's no mundo real e também sobre novas bibliotecas para a construção de jogos em Python.

Tendo, todos os requisitos obrigatórios do trabalho, satisfeitos, foi possível implementar alguns adicionais para melhorar a diversão no jogo, como por exemplo os sons engraçados em determinados eventos.

Este trabalho foi, de grande dificuldade pelo fato do não conhecimento prévio da biblioteca PyGame, e o pouco conhecimento de Threads em Python inicialmente. Porém com muito esforço, tudo começou a caminhar bem, e o trabalho foi finalizado com sucesso.

No mais, os conhecimentos adquiridos durante a realização deste trabalho, serão de grande valia para o decorrer da disciplina.