UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Departamento de Informática Aplicada
INF 01121 - MODELOS DE LINGUAGENS DE PROGRAMAÇÃO
Turma B
Docente: Leandro Krug Wives
Discentes: Bruna Souza de Oliveira, Giulia Rocha Giozza e Thales Alcantara Rocha

Swift-War

Jogo de batalha por turnos desenvolvido em Swift

Porto Alegre, 20 de maio de 2018.

Sumário

Introdução

O presente trabalho teve como objetivo desenvolver os conceitos aprendidos na disciplina de Modelos de Linguagens de Programação, sendo apresentado como Trabalho Final na mesma. Para tanto, coube aos discentes escolher uma linguagem de programação com características multiparadigma, com a qual seria solucionado um problema. Devido a interesses pessoais e conhecimentos prévios dos integrantes do grupo, foi escolhida a linguagem Swift. Com ela, foi desenvolvido um jogo de batalha por turnos no estilo da versão brasileira do jogo americano Risk, War, o qual recebeu o nome de *"Swift-War"*. Primeiramente, o problema foi solucionado utilizando o paradigma orientado a objetos, para então ser resolvido a partir do paradigma funcional.

Apresentação da Linguagem

Swift foi desenvolvida pela própria Apple em 2 de junho de 2014, para ser a linguagem de programação da empresa, para as plataformas *iOS*, *macOS*, *watchOS*, *tvOS* e *Linux*. Atualmente encontra-se na versão 4.1 e está entre as 20 linguagens mais populares. A linguagem foi criada para manter compatibilidade com a *API Cocoa* e com código existente em *Objective-C*.

Swift é uma linguagem compilada multiparadigma e de propósito geral, inspirada na linguagens *C* e *Objective-C*, para programação de sistemas mobile ou desktop, que pode ser desenvolvida em dispositivos *Mac* ou *Linux*. Possui elementos oriundos de *C#, CLU*, *D*, *Haskell*, *Object Pascal*, *Objective-C*, *Python*, *Ruby*, *Rust*.

Tipos básicos fazem parte do core da linguagem e podem ser manipulados diretamente, oferecendo maior simplicidade sintática, o que pode ser observado no exemplo abaixo:

```
var string = "a"
string += "b”
```

Possui 5 tipos de controle de acesso, que ignoram hierarquia de herança:

1. ***open***: indica que a classe pode ser subclasse fora do módulo. Apenas para classes e seus métodos
2. ***public***: acessível de qualquer módulo
3. ***private***: acessível apenas no escopo imediato
4. ***internal***: acessível apenas dentro do módulo
5. ***fileprivate***: acessível apenas de dentro do arquivo

Tipo opcional permite referências ou valores operados de maneira similar à linguagem *C*, onde um ponteiro pode referir-se a um valor ou ser nulo. Usando o caractere *"!"*, assumindo que o valor não é nulo, a instância é exposta. Se for nulo, há erro de *null-pointer*. Já o caractere *"?"* faz o *unwraping* apenas se não for nulo, suprimindo o erro de *null-pointer*. Isso permite que métodos sejam chamados em sequência sem a necessidade de realizar testes antes. Segue outro exemplo:

```
let leaseStart = aBuilding.TenantList[5].leaseDetails?.startDate
```

A linguagem permite que objetos sejam passados por referência ou por valor (como em tipos básicos), usando declaração por *class* ou *struct*, respectivamente. Por esse motivo, todos os dados são genericamente referidos como instâncias, ao invés de objetos ou valores. No entanto, *struct* não aceita herança.

O gerenciamento de memória é feito por *Automatic Reference Counting*, que permite fácil alocação e desalocação de memória. Ele também pode gerar ciclo de referência, causando *memory leakage*. Para evitar esse ciclo, a linguagem oferece as palavras chave *weak* (pais e filhos podem não se relacionar) e *unowned* (um filho sempre tem um pai, mas um pai não precisa ter um filho).

O Problema

War é um jogo de estratégia, no qual há um mapa político da Terra, dividido em 42 territórios. O objetivo é ocupar todos os territórios, eliminando os outros jogadores. Ele pode ser jogado entre 2 a 6 pessoas sendo possível formar e dissolver alianças ao longo do jogo. Costuma ser um jogo de longa duração. Jogado em turnos, a peças representam exércitos e os resultados são determinados por dados.

Criado em 1957, é um dos jogos mais populares até hoje, possuindo versões de tabuleiro, computador, videogame e smartphone. A versão reproduzida neste trabalho deu suporte para a entrada de dois jogadores e será desenvolvida para plataforma mobile.

Desenvolvimento e Código

Como estratégia inicial de desenvolvimento do jogo, o grupo optou por tornar o jogo minimamente funcional. Não em termos do paradigma e sim em termos de jogabilidade; de poder apresentar um aplicativo, ainda que com falhas e sem todas as funcionalidades implementadas, que ofertasse alguma interação com o usuário. Devido ao maior conhecimento do paradigma O.O. do que Funcional por parte dos integrantes do grupo, optou-se por atender aos requisitos O.O. num primeiro momento para então adaptar o código a uma solução funcional. Dentre as funcionalidades desenvolvidas nessa primeira etapa, pôde-se destacar a seleção dos estados e a movimentação das tropas.

O primeiro objetivo foi delimitar o território e estipular jogadores e regras. Considerando que o jogo seria implementado para ser jogado no celular, o grupo optou por limitar o território ao Brasil e dividir as tropas por estados. Os bônus, que no jogo original são ganhados ao conquistar todos os países de um mesmo continente, seriam distribuídos ao conquistar todos os estado de uma mesma região brasileira. Quanto aos jogadores, foi definido que seriam o usuário e

Na primeira etapa de desenvolvimento do código, foram instanciados os estados, aos quais futuramente seriam atribuídas as tropas, de acordo com o jogador ao qual pertencesse, localizadas no mesmo, suas fronteiras e variáveis para identificar o estado selecionado e, em alguns casos o estado para o qual o jogador decide mover suas tropas ou atacar.

Na medida em que foram incluídos funções e procedimentos, notou-se a necessidade de estender a classe *UILabel* usada nos estados, por exemplo, para incluir propriedades aos mesmos tais como suas fronteiras, o que a classe do Swift aparentemente não oferece. A ideia era utilizar os métodos de manipulação *get* e *set* para acessar e alterar seu conteúdo. Devido ao conhecimento disponível da sintaxe da linguagem, alguns recursos para a solução orientada a objetos tiveram que, então, serem deixados para ser solucionados em uma segunda etapa.

Recursos O.O. da Linguagem

Recursos Funcionais da Linguagem

Análise Crítica

Conclusões

Referência Bibliográfica

- https://swift.org/ - acessado em 20 de abril de 2018
- https://developer.apple.com/swift/ - acessado em 20 de abril de 2018