[hang]0.0 0pt

0.1 0pt

1.5 *

0.1.1 0pt

1.5 *

0.1.1.1 0pt

10.5 *

0.1.1.1.1 0pt 10.5

–

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO E ENGENHARIA DE COMPUTAÇÃO

AUGUSTO BORANGA
RODRIGO FRANZOI SCROFERNEKER

Implementação de um Tower Defense com Swift

Relatório apresentado como requisito parcial para a obtenção de conceito na Disciplina de Modelos de Linguagens de Programação

Prof. Dr. Lucas Mello Schnorr
Orientador

Porto Alegre
2017

SUMÁRIO

1 PROBLEMA

Escolhemos para implementar o problema do jogo Tower Defense.

Tower Defense é um subgênero de jogos de estratégia, onde o jogador deve defender seu território (que varia de acordo com a temática do jogo) contra o ataque de inimigos, podendo utilizar um montante inicial para adquirir elementos do jogo que o ajudem na defesa.

1.1 Dinâmica do jogo

Basicamente, há dois elementos essenciais em um jogo do estilo Tower Defense:

Territórios ou propriedades com certa quantidade de vida (o esgotamento desta implica em fim de jogo), que o jogador deve defender;

Inimigos (com outra quantidade de vida) atacando os territórios do jogador;

A dinâmica do jogo com estes elementos é de que há "ondas"de ataques dos inimigos ao jogador. Isto é, o ataque ocorre em partes, com pequenas pausas entre eles.

O jogador pode então, entre ou durante as ondas de ataques, utilizar-se de subterfúgios que atrapalhem a missão dos inimigos (como por exemplo, posicionar obstáculos ou outras estruturas que os ataquem). A aquisição destes equipamentos de defesa custa um certo valor que é decrementado do montante disponível para o jogador.

O objetivo do jogador é sobreviver ao final das N ondas de ataques inimigos.

1.2 Historicamente

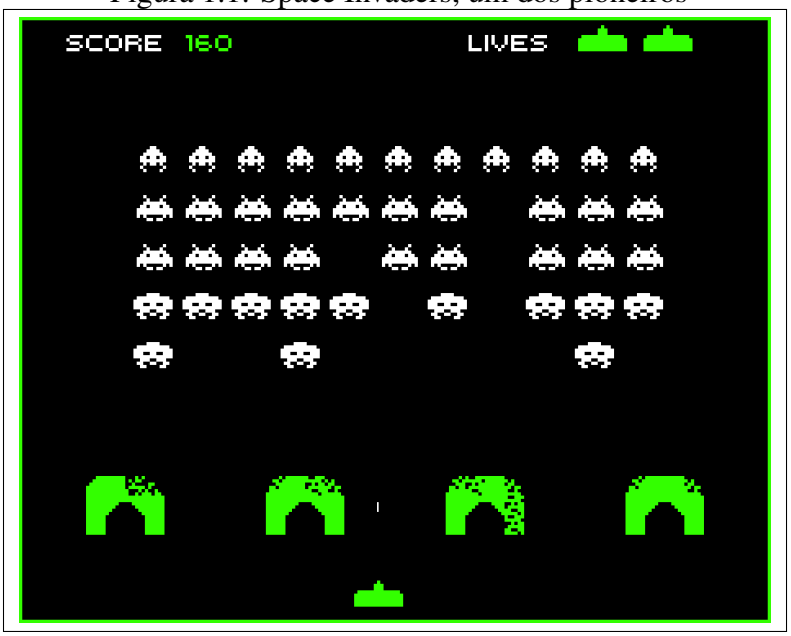
Os primeiros jogos de Tower Defense datam da década de 1980, considerada a Era de Ouro dos video-games.

No jogo *Space Invaders* - um clássico dos video-games lançado em 1978 -, o jogador deve defender seu território atirando em invasores alienígenas. É tido por uns como um precurssor dos jogos de Tower Defense, mas isto é contestado por outros pelo fato de não possuir um elemento fundamental do gênero: a possibilidade de aquisição de elementos extras que auxiliem o jogador na defesa.

Já o jogo *Rampart* - lançado em 1990 -, é amplamente considerado como o jogo que definiu o gênero, por possuir todos os elementos fundamentais deste. Em *Rampart*, as fases de preparação (posicionamento dos itens de defesa), ação (momento em que a

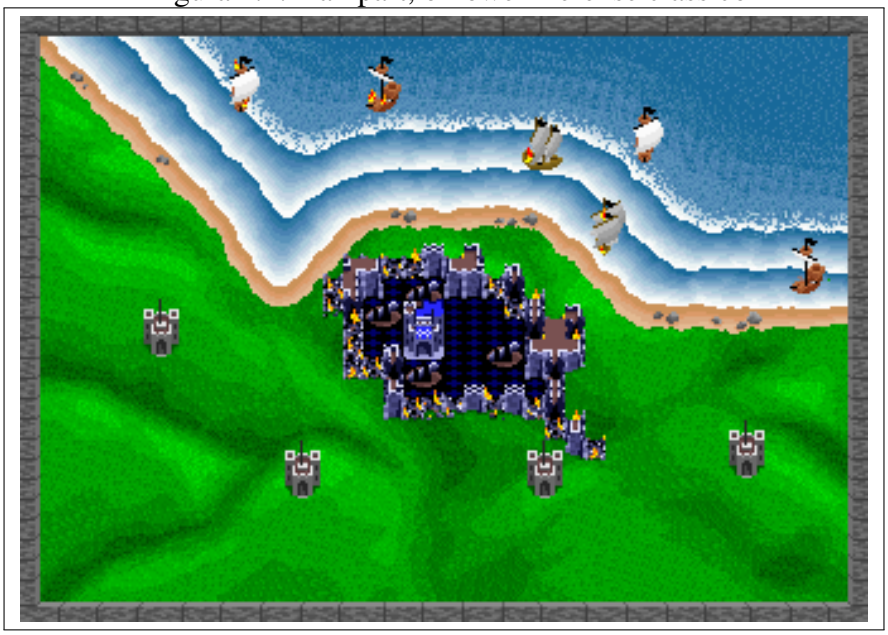
base é atacada e o jogador deve defendê-la) e reparação (consertar elementos danificados pelos ataques) são bem distintas.

Figura 1.1: Space Invaders, um dos pioneiros



Fonte: http://fantendo.wikia.com/wiki/File:Space-Invaders.png

Figura 1.2: Rampart, o Tower Defense clássico



Fonte: http://fantendo.wikia.com/wiki/File:Space-Invaders.png

2 LINGUAGEM

Swift é uma linguagem de programação multiparadigma que se apresenta como uma linguagem moderna e focada em três aspectos: segurança, performance e suporte à aplicação de design patterns.

Mesmo sendo uma linguagem consideravelmente nova, Swift demonstra uma comunidade de tamanho razoável, ficando em 11º nas linguagens mais populares e a 4ª mais amada no site stackoverflow em 2017.

Neste capítulo daremos um panorama a respeito da linguagem adotada, explanando acerca de suas origens e discutindo seus pontos positivos e negativos.

2.1 História

Swift é uma das linguagens de programação mais recentes desenvolvidas no mercado. Foi apresentada em 2014 na WWDC (Worldwide Developers Conference, evento organizado pela Apple para divulgação de novos produtos e features).

Baseada e influenciada por Objective-C, Ruby e Python, se mostrou uma linguagem poderosa e de fácil compreensão (devido principalmente, à sua sintaxe simples).

Inicialmente, Swift era de uso exclusivo por usuários de Mac OS, visto que este era o único sistema operacional habilitado a compilar a linguagem. Em dezembro de 2015, porém, um grande anúncio mudou o jogo: Swift viraria open source, abrindo um leque ainda maior de possibilidades de uso para a linguagem.

2.2 Aspectos técnicos

Swift é uma linguagem multiparadigma, suportando programação funcional e também oferecendo recursos de Orientação a Objetos, como classes e protocolos.

A linguagem possui tipagem estática, o que proporciona maior segurança (garantia dos tipos de dados esperados no código, evitando erros de tipos) e performance (não há gasto de máquina com checagem dos tipos) às aplicações que a utilizam.

Por ter construída pela Apple, possui alta integração com Obj-C, outra linguagem consagrada da Apple. Assim sendo, o programador que utiliza Swift tem à sua disposição uma série de bibliotecas em Obj-C, como por exemplo SpriteKit, uma biblioteca de construção de jogos que utilizaremos no desenvolvimento do trabalho.

Sua sintaxe foi pensada para ser o mais simples e expressiva possível, tornando o código mais fácil de ler e escrever.

2.3 Utilização

Mesmo sendo uma linguagem relativamente recente, Swift agradou a comunidade de desenvolvimento. Na pesquisa do site stackoverflow realizada em 2017, Swift ficou em 4º lugar nas linguagens preferidas pelos usuários.

Suas principais aplicações são:

[leftmargin=3em]**Mobile**: Swift é a linguagem oficial para desenvolvimento de aplicativos para a plataforma iOS. Obj-C também é suportada em iOS, mas o programador mobile é encorajado pela própria Apple a utilizar Swift por esta ser uma linguagem mais recente e simples de usar; **Desktop**: Similar à plataforma mobile (citada acima), aplicações para o sistema operacional dos computadores da Apple (macOS, antigamente chamado OS X) também podem ser escritas em Swift; **Servidor**: Além disso, Swift também pode ser utilizado para o desenvolvimento de aplicações no lado do servidor. Existem alguns frameworks e toolkits (como o Perfect) que auxiliam o desenvolvedor nesta tarefa.

2.4 Motivo da escolha

Optamos por Swift por dois motivos:

• • • Ambos os integrantes do grupo tem familiaridade com a linguagem, devido a experiência prévia desenvolvendo aplicativos mobile.

Swift é uma linguagem muito intuitiva e possui uma variedade de bibliotecas auxiliares disponíveis.

2.5 Análise

A análise crítica será feita ao término do trabalho.

3 IMPLEMENTAÇÃO

Esta seção será construída no futuro.

4 CONCLUSÃO

Esta seção será construída no futuro.

5 REFERÊNCIAS

Esta seção será construída no futuro.