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

ANDRÉ DEXHEIMER
GABRIEL PISCOYA
RODRIGO WIEBBELLING

Trabalho Final para a Disciplina de MLP do Instituto de Informática da UFRGS

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 INTRODUÇÃO

Este capítulo tem o objetivo de descrever de forma sucinta a historia das linguagens de programação e os principais tópicos envolvidos na realização deste trabalho. Logo após, serão abordados os temas diretamente relacionados ao trabalho.

Historia das Linguagens de Programação

As primeiras linguagens de programação eram simples códigos utilizados para automatizar processos nem sempre relacionadas à computação. Na década de 1940, com a criação do primeiro computador moderno, eram utilizados cartões perfurados para facilitar o processo de programação e diminuir a quantidade de erros introduzidos pelo programador. Não foi até meados de 1950 que surgiu a primeira linguagem de programação moderna: FORTRAN, criada por John Backus. Os seguintes anos foram frutíferos, vieram acompanhados de duas novas linguagens de programação: LISP - John McCarthy e COBOL - Grace Hopper.

No começo, todas as linguagens de programação somente permitiam a criação de programas monolíticos e careciam de recursos que facilitassem sua utilização. Somente no fim da década de 1970 que foram estabelecidos os principais paradigmas de programação conhecidos hoje em dia: imperativo, funcional e lógico. Durante estes anos, surgiu o termo "Programação Estruturada", que visava restringir o uso de desvios incondicionais (GoTo) (**??**).

Em 1980, foi criada C++, que combinava orientação a objetos e programação de sistemas, também foi introduzida uma mudança de pensamento na concepção de linguagens de programação, junto com o movimento RISC em arquitetura de computadores, despertou-se maior interesse no uso de compiladores para linguagens de alto nível.

Com a chegada da internet, sugiram as linguagens de scripting, que não são evolução direta de nenhuma linguagem já estabelecida anteriormente, e que foram concebidas com novas sintaxes e novas funções (**??**).

Ambiente e Linguagem de Programação

Como o objetivo do trabalho é aproximar os alunos das linguagens de programação modernas, optamos por escolher uma linguagem que seja amplamente usada na atualidade, também sabemos que ela deve ser multi paradigma, já que devemos implementar soluções utilizando dois paradigmas diferentes. Pelos motivos citados previamente, escolhemos **C++**.

Problema Abordado

A intenção inicial foi a de resolver um problema que já fosse conhecido pelos integrantes do grupo e que despertasse o interesse de todos, portanto escolhemos **Tower Defense**.

2 A LINGUAGEM C++

A linguagem de programação C++ foi criada por Bjarne Stroustrup nos anos de 1980, vindo a ter sua padronização ISO apenas 18 anos depois em 1998. Ela é uma linguagem compilada multi-paradigma, com suporte ao modelo imperativo, ao orientado a objetos, ao genérico, entre outros. Por causa disso, é de uso amplo entre as linguagens comerciais e acadêmicas.

Algumas características do C++

Operadores: O C++ possui todo o conjunto de operadores do C, além de alguns implementados apenas no C++, que dizem respeito à conversão entre tipos, os quais que podem ser const_cast, static_cast, dynamic_cast e reinterpret_cast. Além disso, a linguagem possui sobrecarga de operadores, permitindo que um mesmo operador tenha mais do que 1 significado dependendo do contexto em que é utilizado.

Pré-Processador: antes da compilação propriamente dita, o C++ passa pelo seu pré-processador, gerando modificações léxicas que servem como entrada para a compilação.

Objetos: O C++ tem suporte aos conceitos de orientação à objetos, permitindo a criação de classes que apresentam quatro características desses conceitos: abstração, encapsulamento, herança e polimorfismo. O encapsulamento permite proteger atributos e métodos do objeto, dessa forma é possível que outros trechos do programa tenham acesso apenas aos métodos de interface com a classe. A herança de classes permite que uma classe herde atributos e métodos de outra, podendo ser relacionado com a ideia de classes mãe e filha. O polimorfismo trata da capacidade de se utilizar um operador ou método de diferentes maneiras, facilitando a estendibilidade da classe.

Tratamento de Exceções: Erros podem ser tratados pelo sistema, permitindo que a aplicação se recupere de algum erro sem travar ou ter de ser fechada.

Espaço de Nomes: Permite uma melhor organização das bibliotecas, de forma que cada uma pode criar o seu próprio espaço de nomes para que não existam conflitos.

3 TOWER DEFENSE

É um estilo de jogo de estratégia que consiste em defender uma determinada entidade de inimigos. No nosso jogo, a entidade em questão é uma torre que se encontra no centro da tela. Esta torre possui uma certa quantidade de vida, velocidade de ataque, penetração de armadura, dano e alcance de ataque. Tais características podem ser melhoradas e outras habilidades podem ser adquiridas por meio de compras com a unidade monetária do jogo, obtida matando os inimigos.

Objetivo do jogo

Defender a sua torre de ondas progressivamente maiores de inimigos progressivamente mais fortes.

Os inimigos

Eles têm como objetivo atacar a torre até que sua vida chegue a 0 pontos, surgem de pontos aleatórios nas bordas da tela e vão em direção a torre. Eles possuem atributos definidos pelo nível do jogo, como: velocidade de ataque e de movimento, poder de ataque, quantidade de vida e de defesa. Existem 3 classes de inimigos: a classe "Soldier"se trata de um soldado que anda a pé e possui apenas armas de curto alcance. Ele vai em direção ao centro da tela e somente danifica a torre ao chegar nela. A classe "Horseman"se comporta de maneira semelhante ao soldier, porém possui mais defesa, dano de ataque e velocidade de movimento. A classe "Archer"é a que mais se diferencia das outras pois consegue atacar a torre de longas distâncias, tendo em suas características algo que as outras classes não têm, a distância de ataque, que indica a distância da qual o inimigo deve estar da torre para poder atacá-la.

4 CONCLUSÃO

Apresentar conclusão do trabalho...