**TRABALHO PARA A DISCIPLINA DE**

**TÉCNICAS DE PROGRAMAÇÃO DO CURSO DE**

**ENGENHARIA DE COMPUTAÇÃO DA UTFPR:**

CORRE DO GAÚCHO

Diogo Da Silva Gouveia, Gustavo Esmanhotto Bareta

diogogouveia@alunos.utfpr.edu.br, bareta@alunos.utfpr.edu.br

Disciplina: **Técnicas de Programação – CSE20 / S71** – Prof. Dr. Jean M. Simão

**Departamento Acadêmico de Informática – DAINF** - Campus de Curitiba

Curso Bacharelado em: Engenharia da Computação

**Universidade Tecnológica Federal do Paraná - UTFPR**

Avenida Sete de Setembro, 3165 - Curitiba/PR, Brasil - CEP 80230-901

**Resumo –** Com o intuito de aplicar na prática o que foi aprendido na Disciplina de Técnicas de Programação, tal como satisfazer a exigência de desenvolvimento de um software orientado a objetos (POO) em C++, seguindo as boas práticas de desenvolvimento (coesão e desacoplamento), e documentação, os autores deste trabalho desenvolveram o jogo “Corre do Gaúcho”. Com duas fases e até dois jogadores simultâneos, o objetivo é eliminar os inimigos para avançar de fase, e na última fase derrotar o dragão que aterroriza a vida dos moradores locais. Para a documentação do jogo foi utilizado um Diagrama de Classes em Unified Modeling Language (UML) e também controle de versões através da ferramenta Git. Após implementações, os testes do jogo feitos pelos próprios desenvolvedores demonstraram satisfatibilidade conforme os requisitos e a modelagem elaborada.

**Palavras-chave** Trabalho Acadêmico com foco em Orientação a Objetos em C++; Jogo com aplicação de biblioteca gráfica SFML; Exemplo de aplicação de Engenharia de software; Exemplo de aplicação prática de conteúdos da disciplina de Técnicas de Programação.

**Abstract -** In order to apply in practice what was learned in the Discipline of Programming Techniques, as well as meeting the requirement of developing object-oriented software (OOP) in C ++, following good development practices (cohesion and decoupling) and documentation, the authors of this work developed the game “Corre do Gaúcho”. With two stages and up to two simultaneous players, the objective is to eliminate enemies to advance in the game, and in the last stage defeat the dragon that terrorizes the lives of local residents. For the game documentation was used a Class Diagram in Unified Modeling Language (UML) and also version control through Git tool. After implementations, the game tests performed by the developers themselves showed satisfiability according to the requirements and the elaborated modeling.

**Key-words** Academic work with focus on Object Orientated Programming in C++; Game with application of SFML graphic library; Example of application of Software engineering; Example of practical application of contents of Programming Techniques’s Discipline.

INTRODUÇÃO

EXPLICAÇÃO DO JOGO EM SI

DESENVOLVIMENTO DO JOGO NA VERSÃO ORIENTADA A OBJETO

Tabela 1. Lista de Requisitos do Jogo e suas Situações.

| N. | Requisitos Funcionais | Situação | Implementação |
| --- | --- | --- | --- |
| 1 | Apresentar graficamente menu de opções aos usuários do Jogo, no qual pode se escolher fases, ver colocação (ranking) de jogadores e demais opções pertinentes. |  |  |
| 2 | Permitir um ou dois jogadores com representação gráfica aos usuários do Jogo, sendo que no último caso seria para que os dois joguem de maneira concomitante. |  |  |
| 3 | Disponibilizar ao menos duas fases que podem ser jogadas sequencialmente ou selecionadas, via menu, nas quais jogadores tentam neutralizar inimigos por meio de algum artifício e vice-versa. | . |  |
| 4 | Ter pelo menos três tipos distintos de inimigos, cada qual com sua representação gráfica, sendo que ao menos um dos inimigos deve ser capaz de lançar projéteis contra o(s) jogador(es) e um dos inimigos deve ser um ‘Chefão’. |  |  |
| 5 | Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias, podendo ser várias instâncias e sendo pelo menos 3 instâncias por tipo. |  |  |
| 6 | Ter três tipos de obstáculos, cada qual com sua representação gráfica, sendo que ao menos causa dano em jogador se colidirem. |  |  |
| 7 | Ter em cada fase ao menos dois tipos de obstáculos com número aleatório de instâncias (i.e., objetos), sendo pelo menos 3 instâncias por tipo. |  |  |
| 8 | Ter em cada fase um cenário de jogo constituído por obstáculos, sendo que parte deles seriam plataformas ou similares, sobre as quais pode haver inimigos e podem subir jogadores. |  |  |
| 9 | Gerenciar colisões entre jogador para com inimigos e seus projéteis, bem como entre jogador para com obstáculos. |  |  |
| 10 | Permitir: (1) salvar nome do usuário, manter/salvar pontuação do jogador (incrementada via neutralização de inimigos) controlado pelo usuário e gerar lista de pontuação (ranking). E (2) Pausar e Salvar Jogada |  |  |
| **Total de requisitos funcionais apropriadamente realizados.** | | |  |

Tabela 2. Lista de Conceitos Utilizados e Não Utilizados no Trabalho

| N. | Conceitos | Uso | Onde/O quê |  |
| --- | --- | --- | --- | --- |
| 1 | **Elementares**: |  |  |  |
|  | -Classes, objetos. &  Atributos (privados), variáveis e constantes. &  - Métodos (com e sem retorno). |  |  |  |
|  | - Métodos (com retorno const e parâmetro const). &  Construtores (sem/com parâmetros) e destrutores |  |  |  |
|  | - Classe Principal. |  |  |  |
|  | - Divisão em .h e .cpp. |  |  |  |
| 2 | **Relações de:** |  |  |  |
|  | - Associação direcional. & - Associação bidirecional. |  |  |  |
| - Agregação via associação. &  - Agregação propriamente dita. |  |  |  |
| - Herança elementar. &  - Herança em diversos níveis. |  |  |  |
| - Herança múltipla. |  |  |  |
| 3 | **Ponteiros, generalizações e exceções** |  |  |  |
| - Operador this. |  |  |  |
| - Alocação de memória (new & delete). |  |  |  |
| - Gabaritos/Templates criada/adaptados pelos autores (e.g. Listas Encadeadas via Templates). |  |  |  |
| - Uso de Tratamento de Exceções (try catch). |  |  |  |
| 4 | **Sobrecarga de:** |  |  |  |
| - Construtoras e Métodos. |  |  |  |
| - Operadores (2 tipos de operadores pelo menos). |  |  |  |
| Persistência de Objetos (via arquivo de texto ou binário) |  |  |  |
| - Persistência de Objetos. |  |  |  |
| - Persistência de Relacionamento de Objetos. |  |  |  |
| 5 | **Virtualidade:** |  |  |  |
| - Métodos Virtuais. |  |  |  |
| - Polimorfismo |  |  |  |
| - Métodos Virtuais Puros / Classes Abstratas |  |  |  |
| - Coesão e Desacoplamento |  |  |  |
| 6 | **Organizadores e Estáticos** |  |  |  |
| - Espaço de Nomes (Namespace) criada pelos autores. |  |  |  |
| - Classes aninhadas (Nested) criada pelos autores. |  |  |  |
| - Atributos estáticos e métodos estáticos. |  |  |  |
| - Uso extensivo de constante (const) parâmetro, retorno, método... |  |  |  |
| 7 | Standard Template Library (STL) e String OO |  |  |  |
| - A classe Pré-definida String ou equivalente. &  - Vector e/ou List da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores) |  |  |  |
| - Pilha, Fila, Bifila, Fila de Prioridade, Conjunto, MultiConjunto, Mapa OU Multi-Mapa. |  |  |  |
| Programação concorrente |  |  |  |
| - Threads (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando Posix, C-Run-Time OU Win32API ou afins. |  |  |  |
| - Threads (Linhas de Execução) no âmbito da Orientação a Objetos com uso de Mutex, Semáforos, OU Troca de mensagens. |  |  |  |
| 8 | Biblioteca Gráfica / Visual |  |  |  |
| - Funcionalidades Elementares. & - Funcionalidades Avançadas como:  • tratamento de colisões  • duplo buffer |  |  |  |
|  | - Programação orientada e evento em algum ambiente gráfico.  OU  - RAD – Rapid Application Development (Objetos gráficos como formulários, botões etc). |  |  | |
| **Interdisciplinaridades por meio da utilização de Conceitos de Matemática e/ou Física.** | | | |
| - Ensino Médio. |  |  | |
| - Ensino Superior. |  |  | |
| 9 | Engenharia de Software | | | |
| - Compreensão, melhoria e rastreabilidade de cumprimento de requisitos. & |  |  | |
| - Diagrama de Classes em UML. |  |  | |
| - Uso efetivo (quiçá) intensivo de padrões de projeto (particularmente GOF). |  |  | |
| - Testes à luz da Tabela de Requisitos e do Diagrama de Classes. |  |  | |
| 10 | Execução de Projeto | | | |
| - Controle de versão de modelos e códigos automatizado  (via SVN e/ou afins) OU manual (via cópias manuais). & - Uso de alguma forma de cópia de segurança (backup). |  |  | |
|  | - Reuniões com o professor para acompanhamento do andamento do projeto. |  |  | |
|  | - Reuniões com monitor da disciplina para acompanhamento do andamento do projeto. |  |  | |
|  | - Revisão do trabalho escrito de outra equipe e vice-versa. |  |  | |

Tabela 3. Lista de Justificativas para Conceitos Utilizados e Não Utilizados no Trabalho

| No. | Conceitos | Situação |
| --- | --- | --- |
| 1 | Elementares |  |
| 2 | Relações |  |
| 3 | Ponteiros, generalizações e exceções |  |
| 4 | Sobrecarga e Persistência |  |
| 5 | Virtualidade |  |
| 6 | Organizadores e Estáticos |  |
| 7 | STL, String OO e programação concorrente |  |
| 8 | Biblioteca Gráfica / Visual |  |
| 9 | Engenharia de Software | . |
| 10 | Execução de Projeto |  |

DISCUSSÃO E CONCLUSÕES

DIVISÃO DO TRABALHO

| Atividades. | Responsáveis |
| --- | --- |
| Levantamento de Requisitos |  |
| Diagramas de Classes |  |
| Programação em C++ |  |
| Implementação de Template |  |
| Implementação da Persistência dos Objetos |  |
| Mecânica de Jogo |  |
| Tratamento de Colisões |  |
| Gerenciadora Gráfica e de Eventos |  |
| Tile Map (Mapa de Blocos) |  |
| Arte (sprites, barra de vida, animações) |  |
| Entidades (jogadores, inimigos, obstáculos) |  |
| Menus |  |
| Programação Concorrente |  |
| Classificação de jogadores |  |
| Escrita do Trabalho |  |
| Revisão do Trabalho |  |

AGRADECIMENTOS

REFERÊNCIAS CITADAS NO TEXTO

REFERÊNCIAS UTILIZADAS NO DESENVOLVIMENTO

[A] SIMÃO, J. M. Site da Disciplina de Técnicas de Programação, Curitiba – PR, Brasil. Acessado em 06/05/2021, às 11:51:

<https://pessoal.dainf.ct.utfpr.edu.br/jeansimao/Fundamentos2/Fundamentos2.htm>