# TRABALHO PARA A DISCIPLINA DE TÉCNICAS DE PROGRAMAÇÃO DO CURSO DE SISTEMAS DE INFORMAÇÃO DA UTFPR:

## *Espada Lendária – MODELO & ESPECIFICAÇÃO DO TRABALHO*

Gabriel Rodrigues Estefanes, Washington Ying Ye Wu
gabrielestefanes@alunos.utfpr.edu.br, washingtonwu@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: Sistemas de Informação
Universidade Tecnológica Federal do Paraná - UTFPR
Avenida Sete de Setembro, 3165 - Curitiba/PR, Brasil - CEP 80230-901

**Resumo** - O jogo de plataforma feito, foi desenvolvido para fins de aprendizado na disciplina de Técnicas de Programação, visando o estudo de programação orientada a objetos em C++. Nesse sentido, o jogo Espada Lendária, tem como objetivo controlar um ou dois jogadores que enfrentarão diferentes inimigos com o propósito de eliminar o inimigo final do jogo, o chefão. O jogo tem duas fases que se diferem pelos obstáculos e inimigos encontrados ao decorrer das fases. O desenvolvimento do jogo ocorreu seguindo-se previamente os requisitos textualmente propostos e sequencialmente realizada a modelagem via Diagrama de Classes em Linguagem de Modelagem Unificada (*Unified Modeling Language* - *UML*) baseando-se em um diagrama genérico e prévio proposto. Sem demora, o desenvolvimento do *software* seguiu-se realizado em linguagem de programação C++ visando os conceitos de Orientação de Objetos vistos durante o período letivo da disciplina. Dessa forma, o jogo de plataforma Espada Lendária, possibilitou que se cumprisse o objetivo de aprendizado da disciplina de Técnicas de Programação.

**Palavras-chave ou Expressões-chave** :Programação Orientada a Objetos, Jogo de Plataforma.

## INTRODUÇÃO

O trabalho tem como objetivo colocar em prática o conhecimento adquirido na disciplina de Técnicas de Programação por meio do desenvolvimento de um software.

O objeto de estudo para a realização do trabalho foi na temática de jogos de plataforma, onde o *software* desenvolvido deveria apresentar elementos e aspectos condizentes ao tema imposto como variedade de inimigos e plataformas como também diversos menus e opções de jogo a serem impostas ao usuário.

Para o desenvolvimento do jogo foram-se utilizados em suma, métodos de engenharia de Software, a compreensão dos requisitos textualmente propostos, realizada a modelagem via Diagrama de Classes em Linguagem de Modelagem Unificada (*Unified Modeling Language* - *UML*) baseando-se em um modelo previamente dado, implementação do software em linguagem de programação C++ orientada a objetos e posteriormente testes realizados pelos desenvolvedores do projeto.

Dessa forma, após serem abordados os principais temas do trabalho, nas próximas seções subsequentes será discutida a explicação do jogo em si e do desenvolvimento do jogo na versão orientada a objetos, como também a discussão e conclusão do trabalho obtido.

## EXPLICAÇÃO DO JOGO EM SI

Em Espada Lendária o objetivo do jogo é imobilizar os inimigos que existem em cada fase, onde na primeira fase os inimigos variam entre os esqueletos e os magos (Imagem 1), e na segunda fase os inimigos podem variar entre os esqueletos e o inimigo principal que seria o Boss (Imagem 2).

Os inimigos mencionados possuem interações diferentes com o jogador; o esqueleto causa dano apenas em curto alcance, o mago pode causar dano de qualquer distância desde que o jogador esteja em seu campo de visão já que ele dispara um projétil de magia, e o Boss devido a sua raiva causa mais dano ao oponente e possui uma maior quantidade de vida.

A transição das fases acontecem por meio de um menu para a seleção dessas; outra diferença entre as fases seriam os obstáculos presentes, onde na primeira fase teríamos plataformas normais, plataformas móveis e espinhos que podem causar dano ao jogador; já na segunda fase, de diferença com a primeira, seria a ausência dos espinhos.

O jogo possui alguns menus disponíveis para interações com o usuário, como o menu de salvar o jogo, que salva o progresso do personagem nas fases; o menu colocação, que mostra a pontuação dos usuários que já jogaram o jogo; e o menu de pausa, que pausa o jogo.

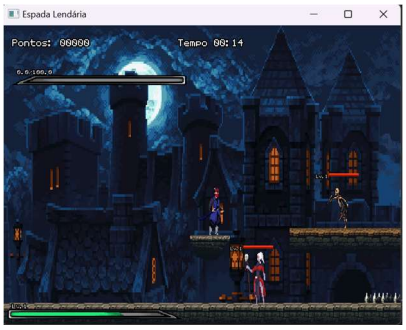


Imagem 1. Primeira fase.

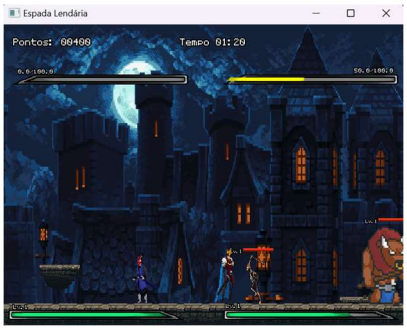


Imagem 2. Segunda fase.

## DESENVOLVIMENTO DO JOGO NA VERSÃO ORIENTADA A OBJETOS

Para o desenvolvimento do *software* foram definidos requisitos que atendessem a necessidade do usuário. Esses requisitos apresentam as principais funcionalidades que o jogo deve ter para cumprir seus objetivos. Na tabela abaixo, é possível verificar de forma detalhada os requisitos que serviram como base para o desenvolvimento do jogo.

Tabela 1. Lista de Requisitos do Jogo e exemplos de 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 (previstas nos demais requisitos). | Requisito previsto inicialmente e realizado. | Requisito cumprido via classe Menu e seu respectivo objeto, com suporte da SFML. |
| 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. | Requisito previsto inicialmente e realizado. | Requisito cumprido inclusive via classes Jogador1 e Jogador2. |
| 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. | Requisito previsto inicialmente e realizado. | Requisito cumprido via seleção de fases no menu de fases. |

| | | | |
|---|---|---|---|
| 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'. | Requisito previsto inicialmente e realizado. | Requisito cumprido como se observa no pacote Personagens, onde na hierarquia de personagens existem três tipos de inimigos. |
| 5 | Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias, podendo ser várias instâncias (definindo um máximo) e sendo pelo menos 3 instâncias por tipo. | Requisito previsto inicialmente e realizado. | Requisito cumprido na criação das fases, podendo ser observado tanto na classe FasePrimeira como também na FaseSegunda. |
| 6 | Ter três tipos de obstáculos, cada qual com sua representação gráfica, sendo que ao menos um causa dano em jogador se colidirem. | Requisito previsto inicialmente e realizado. | Requisito cumprido como se observa no pacote Obstaculos, onde existem dois tipos de plataforma e o espinho que concede dano ao jogar. |
| 7 | Ter em cada fase ao menos dois tipos de obstáculos com número aleatório (definindo um máximo) de instâncias (*i.e.*, objetos), sendo pelo menos 3 instâncias por tipo. | Requisito previsto inicialmente e realizado. | Requisito cumprido na criação das fases, podendo ser observado tanto na classe FasePrimeira como também na FaseSegunda. |
| 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. | Requisito previsto inicialmente e realizado. | Requisito cumprido na criação das fases, podendo ser observado tanto na classe FasePrimeira como também na FaseSegunda. |
| 9 | Gerenciar colisões entre jogador para com inimigos e seus projeteis, bem como entre jogador para com obstáculos. Ainda, todos eles devem sofrer o efeito de alguma ´gravidade´ no âmbito deste jogo de plataforma vertical e 2D. | Requisito previsto inicialmente e realizado. | Requisito cumprido, podendo se observar no Gerenciador de Colisões. |
| 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/Recuperar** Jogada. | Requisito previsto inicialmente e realizado parcialmente. | Requisito realizado parcialmente via classe MenuColocacao, MenuSalvarJogada e GerenciadorArquivo. |
| **Total de requisitos funcionais apropriadamente realizados.** | | | 90 - 95% |

O desenvolvimento do jogo foi realizado utilizando a Programação Orientada a Objetos (POO). Inicialmente, foram definidos os requisitos funcionais descritos na Tabela 1 que serviram como base para a modelagem do sistema e permitiram que grande parte delas fossem implementadas no *software.*

A modelagem do jogo de plataforma foi feito utilizando diagramas de classes em UML (Unified Modeling Language). O diagrama apresenta as principais entidades do jogo, como jogadores, inimigos, obstáculos, fases, menus e outros; e também suas respectivas interações, como herança, encapsulamento e polimorfismo.

A classe "Jogador", por exemplo, é responsável por apresentar os personagens controlados pelos usuários, enquanto a classe "Inimigo", com suas subclasses para cada inimigo específico, permite uma variedade no comportamento e nas características gráficas.

Na fase de programação, foi utilizado a linguagem de programação em C++ para colocar em prática o projeto modelado na Linguagem de Modelagem Unificada. As classes foram

implementadas com a integração da biblioteca gráfica SFML, para fins de gerenciar a interface visual e as interações com o usuário.

## TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

Para o desenvolvimento do *software* foram sugeridos conceitos que poderiam ser implementados. Esses conceitos apresentam as principais características de uma Programação Orientada a Objetos (POO). Na tabela abaixo, é possível verificar de forma detalhada os conceitos que foram aplicados no jogo desenvolvido.

| **N.** | Conceitos | Uso | Onde / O quê / Justificativa em uma linha |
|---|---|---|---|
| **1** | **Elementares:** | | |
| 1.1 | - Classes, objetos. & - Atributos (privados), variáveis e constantes. & - Métodos (com e sem retorno). | Sim | Todos .h e .cpp, como nas classes no *namespace* EspadaLendaria. |
| 1.2 | - Métodos (com retorno *const* e parâmetro *const*). & - Construtores (sem/com parâmetros) e destrutores | Sim | Na maioria dos .h e .cpp, como nas classes nos *namespaces* EspadaLendaria.. |
| 1.3 | - Classe Principal. | Sim | Main.cpp & Principal.h/.cpp |
| 1.4 | - Divisão em .h e .cpp. | Sim | No desenvolvimento como um todo, como nas classes nos *namespaces* EspadaLendaria. |
| **2** | **Relações de:** | | |
| 2.1 | - Associação direcional. & - Associação bidirecional. | Sim | Em vários dos .h e .cpp, como nas classes nos *namespaces* Observador,Jogador. |
| 2.2 | - Agregação via associação. & - Agregação propriamente dita. | Sim | Em vários dos .h e .cpp, como nas classes nos *namespaces* Lista e Entidade. |
| 2.3 | - Herança elementar. & - Herança em diversos níveis. | Sim | Em alguns dos .h e .cpp, como nas classes nos *namespace* EspadaLendaria.. |
| 2.4 | - Herança múltipla. | Não | Não implementado.. |
| **3** | **Ponteiros, generalizações e exceções** | | |
| 3.1 | - Operador *this* para fins de relacionamento bidirecional. | Sim | Precisamente nos .h e .cpp, das classes ObservadorJogador e Jogador.. |
| 3.2 | - Alocação de memória (*new* & *delete*). | Sim | Em vários dos .h e .cpp, como nas classes nos *namespaces* Menu e Jogador. |
| 3.3 | - Gabaritos/*Templates* criada/adaptados pelos autores (*e.g.*, Listas Encadeadas via *Templates*). | Sim | Precisamente nos .h, das classes Elemento e Lista. |
| 3.4 | - Uso de Tratamento de Exceções (*try catch*). | Sim | Em vários dos .h e .cpp, como nas classes nos *namespaces* Entidade e Jogador. |

| | | | |
|---|---|---|---|
| **4** | **Sobrecarga de:** | | |
| 4.1 | - Construtoras e Métodos. | Sim | Em vários dos .h e .cpp, como nas classes nos *namespaces* Inimigo e Jogador. |
| 4.2 | - Operadores (2 tipos de operadores pelo menos – Quais? ). | Não | |
| --- | **Persistência de Objetos (via arquivo de texto ou binário)** | | |
| 4.3 | - Persistência de Objetos. | Sim | Em vários dos .h e .cpp, como nas classes nos *namespaces* Fase e Jogador. |
| 4.4 | - Persistência de Relacionamento de Objetos. | Sim | Em vários dos .h e .cpp, como nas classes nos *namespaces* Inimigo e Jogador. |
| **5** | **Virtualidade:** | | . . . |
| 5.1 | - Métodos Virtuais Usuais. | Sim | Em vários dos .h e .cpp, como nas classes no *namespace Personagem*. |
| 5.2 | - Polimorfismo. | Sim | Em vários dos .h e .cpp, como nas classes no *namespace* Inimigo. |
| 5.3 | - Métodos Virtuais Puros / Classes Abstratas. | Sim | Em vários dos .h e .cpp, como nas classes no *namespace* Inimigo. |
| 5.4 | - Coesão/Desacoplamento efetiva e intensa com o apoio de padrões de projeto. | Sim | Em alguns dos .h e .cpp, como nas classes nos *namespace* EspadaLendaria. |
| **6** | **Organizadores e Estáticos** | | |
| 6.1 | - Espaço de Nomes (*Namespace*) criado pelos autores. | Sim | Como EspadaLendaria,Entidade. |
| 6.2 | - Classes aninhadas (*Nested*) criada pelos autores. | Não | . |
| 6.3 | - Atributos estáticos e métodos estáticos. | Sim | Em alguns dos .h e .cpp, como nas classes nos *namespace* EspadaLendaria. |
| 6.4 | - Uso extensivo de constante (*const*) parâmetro, retorno, método... | Sim | Em alguns dos .h e .cpp, como nas classes nos *namespace* EspadaLendaria. |
| **7** | **Standard Template Library *(STL )* e String OO** | | |
| 7.1 | - A classe Pré-definida *String* ou equivalente. **&** - *Vector* e/ou *List* da *STL* (p/ objetos ou ponteiros de objetos de classes definidos pelos autores) | Sim | Em alguns dos .h e .cpp, como na classe Inimigo nos *namespace* EspadaLendaria. |
| 7.2 | - Pilha, Fila, Bifila, Fila de Prioridade, Conjunto, MultiConjunto, Mapa **OU** MultiMapa. | Sim | Na Classe EstadoJogar.hpp e cpp |
| --- | **Programação concorrente** | | |
| 7.3 | - *Threads* (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando Posix, C-RunTime **OU** Win32API ou afins. | *Não* | . . . |

| | | | |
|---|---|---|---|
| 7.4 | - *Threads* (Linhas de Execução) no âmbito da Orientação a Objetos com uso de Mutex, Semáforos, **OU** Troca de mensagens. | *Não* | . . . |
| **8** | **Biblioteca Gráfica / Visual** | | |
| 8.1 | - Funcionalidades Elementares. **&** -Funcionalidades Avançadas como: • tratamento de colisões • duplo *buffer* | Sim | *Está presente tanto em plataforma, Inimigos e os Jogadores, o sistema de colisão não permite que o jogador/inimigo passe por cima do outro e tem colisão com a plataforma.* |
| 8.2 | - Programação orientada e evento efetivo (com gerenciador apropriado de eventos inclusive) em algum ambiente gráfico. **OU** - *RAD – Rapid Application Development* (Objetos gráficos como formulários, botões etc). | Sim | Presente no Sistema de Menu |
| --- | **Interdisciplinaridades via utilização de Conceitos de <u>Matemática Contínua e/ou Física</u>.** | | |
| 8.3 | - Ensino Médio Efetivamente. | *Sim* | Distância Entre 2 Centros e MRU/MRUV |
| 8.4 | - Ensino Superior Efetivamente. | | Especificar quais conceitos aqui. |
| **9** | **Engenharia de Software** | | |
| 9.1 | - Compreensão, melhoria e rastreabilidade de cumprimento de requisitos. & | Sim | Maioria dos requisitos feitos. |
| 9.2 | - Diagrama de Classes em *UML*. | Sim | Feito. |
| 9.3 | - Uso efetivo e intensivo de padrões de projeto *GOF*, *i.e.*, mais de 5 padrões. | Sim | Singleton,Chain of Responsibility ,Template Method, Builder, Observer e State são os mais presentes |
| 9.4 | - Testes à luz da Tabela de Requisitos e do Diagrama de Classes. | Sim | |
| **10** | **Execução de Projeto** | | |

| | | | |
|---|---|---|---|
| 10.1 | - Controle de versão de modelos e códigos automatizados (via github e/ou afins). &<br>- Uso de alguma forma de cópia de segurança (*i.e.*, *backup*). | *Via github* | https://github.com/EstefanesGabriel/EspadaLendariaC- |
| 10.2 | - Reuniões com o professor para acompanhamento do andamento do projeto.<br>**[ITEM OBRIGATÓRIO PARA A ENTREGA DO TRABALHO]** | *Sim* | 2 Reuniões marcadas com o Professor, 2 Oficinas Peteco,4 Reuniões com Monitores. |
| 10.3 | - Reuniões com monitor da disciplina para acompanhamento do andamento do projeto.<br>**[ITEM OBRIGATÓRIO PARA A ENTREGA DO TRABALHO]** | *Sim* | 4 Reuniões com Monitores Feito. |
| 10.4 | - Revisão do trabalho escrito de outra equipe e vice-versa. | *Não* | Especificar qual equipe |
| **Total de conceitos apropriadamente utilizados.** | | | **85-90%** |

## DISCUSSÃO E CONCLUSÕES

Dessa forma, o desenvolvimento do *software* permitiu fazer com que fosse utilizado todo o conhecimento adquirido ao decorrer do semestre e aplicá-lo de forma prática na construção de um jogo de plataforma; durante o desenvolvimento houveram algumas dificuldades, como a implementação do segundo jogador e o gerenciamento dos arquivos de salvamento, mas que em sua maioria foram resolvidos. Os resultados obtidos foram os esperados, foram realizados aproximadamente 90% dos requisitos e a conclusão foi um jogo de qualidade seguindo os modelos sugeridos.

## DIVISÃO DO TRABALHO

A tabela a seguir apresenta a distribuição das atividades relacionadas ao desenvolvimento do jogo entre a dupla Gabriel e Washington.

Tabela 4. Lista de Atividades e Responsáveis.

| **Atividades.** | **Responsáveis** |
|---|---|
| Implementação da Classe Animacao | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |

| | |
|---|---|
| Implementação da Classe Arma | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Boss | Mais Washington Ying Ye Wu que Gabriel Rodrigues Estefanes |
| Implementação da Classe Botao | Mais Washington Ying Ye Wu que Gabriel Rodrigues Estefanes |
| Implementação da Classe BotaoTexto | Mais Washington Ying Ye Wu que Gabriel Rodrigues Estefanes |
| Implementação da Classe Camada | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Camera | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Elemento | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Ente | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Entidade | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Espinho | Mais Washington Ying Ye Wu que Gabriel Rodrigues Estefanes |
| Implementação da Classe Esqueleto | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Estado | Gabriel Rodrigues Estefanes |
| Implementação da Classe EstadoJogar | Mais Gabriel Rodrigues Estefanes que Washington Ying Ye Wu |
| Implementação da Classe EstadoMenu | Mais Gabriel Rodrigues Estefanes que Washington Ying Ye Wu |

| | |
|---|---|
| Implementação da Classe Fase | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe FasePrimeira | Mais Gabriel Rodrigues Estefanes que Washington Ying Ye Wu |
| Implementação da Classe FaseSegunda | Mais Gabriel Rodrigues Estefanes que Washington Ying Ye Wu |
| Implementação da Classe FotoSalvar | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Fundo | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe GerenciadorArquivo | Gabriel Rodrigues Estefanes |
| Implementação da Classe GerenciadorColisao | Washington Ying Ye Wu |
| Implementação da Classe GerenciadorEstado | Gabriel Rodrigues Estefanes |
| Implementação da Classe GerenciadorEvento | Washington Ying Ye Wu |
| Implementação da Classe GerenciadorGrafico | Gabriel Rodrigues Estefanes |
| Implementação da Classe IDs | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Imagem | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Inimigo | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Jogador | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Jogador1 | Mais Washington Ying Ye Wu que Gabriel Rodrigues Estefanes |

| | |
|---|---|
| Implementação da Classe Jogador2 | Mais Washington Ying Ye Wu que Gabriel Rodrigues Estefanes |
| Implementação da Classe Lista | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe ListaEntidade | Mais Gabriel Rodrigues Estefanes que Washington Ying Ye Wu |
| Implementação da Classe ListaObservador | Mais Washington Ying Ye Wu que Gabriel Rodrigues Estefanes |
| Implementação da Classe Mago | Mais Washington Ying Ye Wu que Gabriel Rodrigues Estefanes |
| Implementação da Classe Menu | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe MenuCarregarJogo | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe MenuColocacao | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe MenuFase | Gabriel Rodrigues Estefanes |
| Implementação da Classe MenuGameOver | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe MenuPausa | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe MenuPrincipal | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe MenuQntjogadores | Washington Ying Ye Wu |
| Implementação da Classe MenuSalvarJogada | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |

| | |
|---|---|
| Implementação da Classe Nivel | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Observador | Washington Ying Ye Wu |
| Implementação da Classe ObservadorFase | Mais Washington Ying Ye Wu que Gabriel Rodrigues Estefanes |
| Implementação da Classe ObservadorJogador | Mais Washington Ying Ye Wu que Gabriel Rodrigues Estefanes |
| Implementação da Classe ObservadorMenu | Mais Washington Ying Ye Wu que Gabriel Rodrigues Estefanes |
| Implementação da Classe Obstaculo | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Personagem | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Plataforma | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe PlataformaMovel | Mais Gabriel Rodrigues Estefanes que Washington Ying Ye Wu |
| Implementação da Classe Principal | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Projetil | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Classe Texto | Mais Gabriel Rodrigues Estefanes que Washington Ying Ye Wu |
| Implementação da Classe TextoAnimado | Mais Gabriel Rodrigues Estefanes que Washington Ying Ye Wu |
| Implementação da Classe Vida | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |

| | |
|---|---|
| Compreensão de Requisitos | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Diagramas de Classes | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Programação em C++ | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação de *Template* | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Implementação da Persistência dos Objetos... | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Escrita do Trabalho | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |
| Revisão do Trabalho | Gabriel Rodrigues Estefanes e Washington Ying Ye Wu |

- Gabriel Rodrigues Estefanes trabalhou em 100% das atividades ou as realizando ou colaborando nelas efetivamente.

- Washington Ying Ye Wu trabalhou em 100% das atividades ou as realizando ou colaborando nelas efetivamente.

## AGRADECIMENTOS PROFISSIONAIS

Agradeço a orientação do Prof. Dr. Jean M. Simão em suas reuniões, o auxílio dos monitores Edison Furusato, Ricardo Reyes e Giovane Salvi no desenvolvimento do jogo, como também da ajuda dos membros do PETECO no esclarecimento de algumas dúvidas.

## REFERÊNCIAS CITADAS NO TEXTO

[1] SIMÃO, J. M. TÉCNICA DE PROGRAMAÇÃO. Página de Internet do Prof. Simão, 2024. Disponível em: https://pessoal.dainf.ct.utfpr.edu.br/jeansimao/Fundamentos2/Fundamentos2.htm. Acesso em: 02/09/2024.

## REFERÊNCIAS UTILIZADAS NO DESENVOLVIMENTO

[1] SIMÃO, J. M. TÉCNICA DE PROGRAMAÇÃO. Página de Internet do Prof. Simão, 2024. Disponível em: https://pessoal.dainf.ct.utfpr.edu.br/jeansimao/Fundamentos2/Fundamentos2.htm. Acesso em: 02/09/2024.

[2] CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas. Introdução a estruturas de dados: com técnicas de programação em C. 2ª Edição. Rio de Janeiro: Elsevier, 2016.

[3] GEGE++. Criando um Jogo em C++ do ZERO. 2022. Disponível em: https://youtube.com/playlist?list=PLR17O9xbTbIBBoL3lli44N8LdZVvg-_uZ&si=mprpOKKm2gYq30z0. Acesso em: 02/09/2024.