TRABALHO PARA A DISCIPLINA DE TÉCNICAS DE PROGRAMAÇÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DA UTFPR: SPACE HUNT – JOGO DE PLATAFORMA EM C++

CRISTIANO FROTA MENEZES

cristiano.2019@alunosutfpr.edu.br

Disciplina: **Técnicas de Programação – CSE20** / S73 – Prof. Dr. Jean M. Simão
Departamento Acadêmico de Informática – DAINF - Campus de Curitiba
Curso Bacharelado em: Engenharia da Computação / Sistemas de Informação
Universidade Tecnológica Federal do Paraná - UTFPR
Avenida Sete de Setembro, 3165 - Curitiba/PR, Brasil - CEP 80230-901

Resumo – Tendo em vista a segunda avaliação da matéria técnica de programação precisamos implementar um jogo de plataforma 2D por meio da linguagem C++. O jogo escolhido foi Space Hant, jogado apenas individualmente, foram implementadas 2 fazes, onde a necessidade é vencer em um ambiente espacial inimigos de outros planetas. Dentro dos requisitos predefinidos foram aplicados nas fases uma variedade de obstáculos e inimigos com um sistema simples de combate carecendo de um objetivo final.
Utilizando de um Diagrama de Classes como base, dentro dos requisitos propostos foram aplicados diversos recursos da programação orientada a objetos, assim como específicos da linguagem C++, entre eles o conceito de polimorfismo, Standard Template Library – STL, operadores entre outros.
Ao final do processo apesar de não estar em uma versão completa, foi aplicado uma quantidade de requisitos dentro do previsto, de forma a ampliar os conhecimentos dessa área de maneira efetiva.

Expressões-chave – Trabalho de Técnicas de Programação, Desenvolvimento de jogos em C++, Desenvolvimento de jogos SFML, Trabalho de programação orientada a objetos.

Abstract - In view of the second evaluation of the technical matter of programming, we need to implement a 2D platform game using the C++ language. The game chosen was Space Hant, played only individually, 2 phases were implemented, where the need is to win in a space environment enemies from other planets. Within the predefined requirements, a variety of obstacles and enemies were applied in the stages with a simple combat system lacking a final objective.
Using a Class Diagram as a basis, within the proposed requirements, several features of object-oriented programming were applied, as well as specific features of the C++ language, including the concept of polymorphism, Standard Template Library - STL, operators, among others. At the end of the process, despite not being in a complete version, a number of requirements were applied as expected, in order to effectively expand knowledge in this area.

Key Expressions – Programming Techniques Work, Game Development in C++, SFML Game Development, Object Oriented Programming Work.

Introdução – Este trabalho decorre no semestre 2023/2 da matéria de Técnicas de Programação, que serve para avaliar a segunda nota presente no semestre, para passagem da matéria e progressão dentro do curso, adquirindo conhecimento a cerca de programação orientada objetos utilizando da linguagem requisitada, tendo foco durante a aplicação no entendimento em como funciona no mercado de trabalho, onde tivemos reuniões com foco em demonstrar a progressão.

Tendo como objeto de estudo o desenvolvimento de um jogo 2D de plataforma, utilizando-se dos recursos da linguagem em questão assim como de métodos providos por ela, utilizando também de uma biblioteca gráfica para ilustrar a funcionalidade do jogo.

O jogo foi implementado como previsto na linguagem C++, com uso da biblioteca gráfica SFML, o código foi em sua grande maioria escrito e compilado em Visual Studio Code, também foi feito um diagrama UML por meio do programa Star UML.

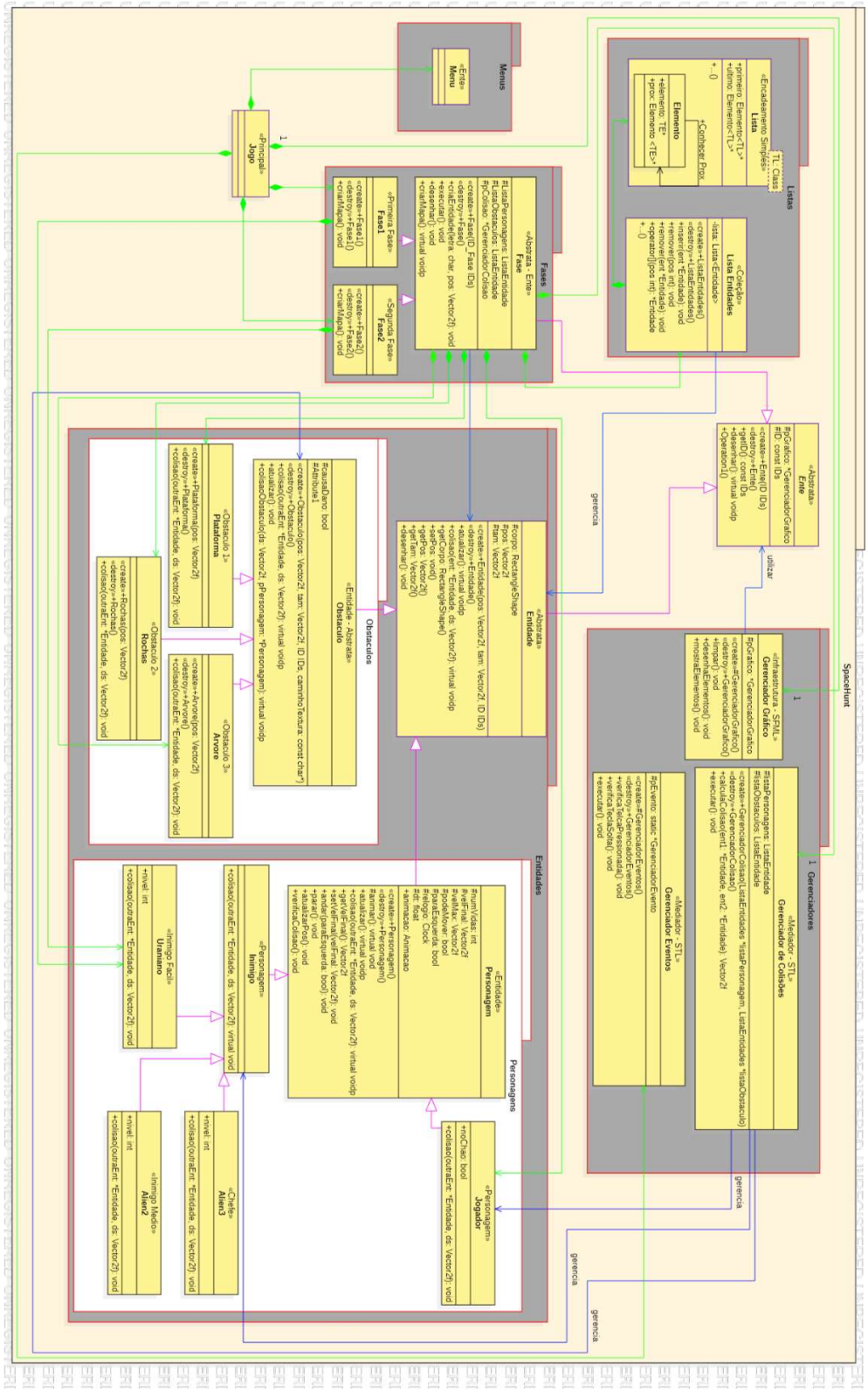
A frente teremos alguns tópicos explicando o processo de maneira mais abrangente e expecificando o que foi feito e os problemas enfrentados no percurso.

Explicação – O jogo consiste em derrotar os alienígenas inimigos em diversos planetas, no entanto comecei primeiro configurando a biblioteca gráfica da SFML que viria a ser utilizada, após isso partimos para o desenvolvimento em si, primeiro criei uma classe jogador onde inicialmente tinhamos apenas uma representação básica da SFML de um cubo verde, em seguida implementei uma maneira básica de movimentação para uma base.
Utilizando de herança e o desenvolvimento de classes, desenvolvi tanto a personagem de maneira mais profunda, como inimigos, cujo a frente foram derivados de maneira única nos alienigenas Uraniano, Venusiano e Verme, durante a execução podemos ver apenas o Uraniano, mas é um fato que pode ser alterado, da mesma forma com os obstáculos, onde temos a plataforma, arvore e rochas, as rochas são um obstaculo que causa dano ao jogador. A ideia do cerne do jogo seria um caçador espacial(jogador), que exploram os mais diversos planetas caçando aliens, devido a isso implementamos uma parte gráfica toda temática, assim como um efeito de parallax que da uma profundidade ao cenário do jogo. Foi implementado um sistema de colisão onde foi utilizado equações simples.

Desenvolvimento orientado Objetos -

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.	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, mas realizado apenas PARCIALMENTE. - falta segundo jogador.	Requisito cumprido inclusive via classe Jogador cujos objetos são agregados em jogo, podendo ser apenas um jogador, entretanto.
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, mas realizado apenas PARCIALMENTE – falta uma maneira de concluir e dar sequencia a fase.	Requisito cumprido, via classe abstrata Fase e podendo ser selecionadas no meu, porém não podem ser jogadas sequencialmente e não temos uma maneira de vencer a fase.
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 projetil contra o(s) jogador(es) e um dos inimigos dever ser um ‘Chefão’.	Requisito previsto inicialmente, mas realizado apenas PARCIALMENTE – falta projetil e classe abstrata chefe.	Tal requisito foi realizado PARCIALMENTE como se observa no pacote Personagens, faltando no entanto desenvolver o projetil e uma divisão nos inimigos para Chefão.
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 não realizado.	Não Implementado
6	Ter três tipos de obstáculos,	Requisito previsto inicialmente	Requisito cumprido

	cada qual com sua representação gráfica, sendo que ao menos um causa dano em jogador se colidirem.	e realizado.	totalmente como é possível observar dentro da pasta obstaculos, temos uma classe base e as derivadas com plataforma, rochas e arvore.
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 não realizado.	Não Implementado
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 totalmente como é possível observar dentro da pasta obstaculos, temos uma classe base e as derivadas com plataforma, rochas e arvore.
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 via gerenciador de colisões e métodos colisao dentro de cada entidade.
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.	Requisito previsto inicialmente e não realizado.	Não Implementado
Total de requisitos funcionais apropriadamente realizados. (Cada tópico vale 10%, sendo que para ser contabilizado deve estar realizado efetivamente e não parcialmente)			40% (40 por cento)



Reflexão – É evidente as vantagens de utilizar a programação orientada a objetos, principalmente em ambitos onde se aproximamos de uma ideia de representar o mundo real, a divisão por classes que consideramos objetos e a fácil manipulação dos mesmos, traz uma flexibilidade ao código tornando ela muito mais prática e eficiente.

Conclusão – O trabalho tem uma quantia boa de conceitos aplicados, mas peca nos requisitos fundamentais e desenvolvimento, apesar do conhecimento e haver uma ideia boa por trás, falta continuidade.

Considerações Pessoais – A cerca do desempenho geral acredito que o trabalho possa ter deixado a desejar, porém existiu a dificuldade da dupla cuja a qual negligenciou e iludiu um falso progresso no trabalho, forçando a carga toda a ser feita apenas por mim, de maneira extremamente acelerada nos prazos, resultando em algo não completo conforme o requisitado, acredito que ainda assim aprendi muito com a matéria, por mais que não consiga a nota desejada, apesar das dificuldades e possivelmente fracasso, a matéria me despertou um profundo interesse na área.

Agradecimentos – Ainda presto aqui meus agradecimentos ao Professor pela compreensão que teve em relação aos prazos das reuniões e por mais que eu não consiga a nota, por ter me deixado entregar o trabalho apesar de tudo.