TRABALHO PARA A DISCIPLINA DE TÉCNICAS DE PROGRAMAÇÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DA UTFPR: Ninja++

Henrique Castro Santos
henricastrosantos@gmail.com

Disciplina: **Técnicas de Programação – CSE20** – 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 – O presente documento relata o processo de desenvolvimento do trabalho final da disciplina de Técnicas de Programação da Universidade Tecnológica Federal do Paraná realizado por Henrique Castro Santos. O trabalho em si consiste no desenvolvimento e implementação de um jogo de plataforma que visa apurar o aprendizado das técnicas ensinadas em sala de aula. O jogo de escolha pelo discente foi o: "Ninja++", este possui duas fases e tem como protagonista um ninja. Foram utilizados diversos conceitos do C++ que foram passados durante as aulas como o Polimorfismo, Classe Abstrata, Persistências de Objetos por Arquivos e etc. Pode-se afirmar que o jogo conseguiu alcançar o seu objetivo com sucesso, esse sendo o aprendizado das técnicas de programação fundamentais do C++.

Palavras-chave: Artigo-Relatório para a disciplina de Técnicas de Programação, Implementação de conceitos básicos e avançados em C++.

Abstract – This document is relating the development of a plataform game wich is the final Project of the subject Técnicas de Programação of the Computer Engineering course of the Univerisdade Tecnológica Federal do Paraná. The programing language used was the C++.

Key-words: Final paper for the subject Técnicas de Programação for the Computer Engineering course. Use of basic and advanced concepts of C++ to develop a platform game.

INTRODUÇÃO

Este trabalho foi realizado para matéria de Técnicas de Programação e o mesmo busca testar o conhecimento dos alunos sobre o conteúdo passado durante o semestre. Consequentemente, o aprendizado também ocorre durante a implementação dos conceitos utilizados.

Para isto, com a ajuda de uma biblioteca gráfica, foi requisitado o desenvolvimento de um jogo de plataforma em C++ onde todos os conteúdos demonstrados durante o semestre podem ser utilizados. Além de utilizar conceitos de programação, diversos conceitos de outras matérias são utilizados como a Física e a Geometria Analítica.

O desenvolvimento do jogo seguiu o ciclo padrão da Engenharia de Software com o levantamento de requisitos, desenvolvimento do diagrama em UML, a implementação do código e do conhecimento obtido em aula e testes com uso de software.

Seguindo daqui, será apresentada primeiramente a ideia do jogo e seus conceitos mais básicos para em seguida entrar em termos mais técnicos.

EXPLICAÇÃO DO JOGO EM SI

Para fazer um jogo, é necessária uma interface gráfica para que o usuário interaja. Para isso, foi utilizado o SFML que é uma biblioteca gráfica desenvolvida para o C++. A partir daí sprites do jogo Ninja Gaiden foram os escolhidos para apresentar o jogo. Para ilustrar, segue abaixo uma imagem do jogo com seus sprites já implementados :

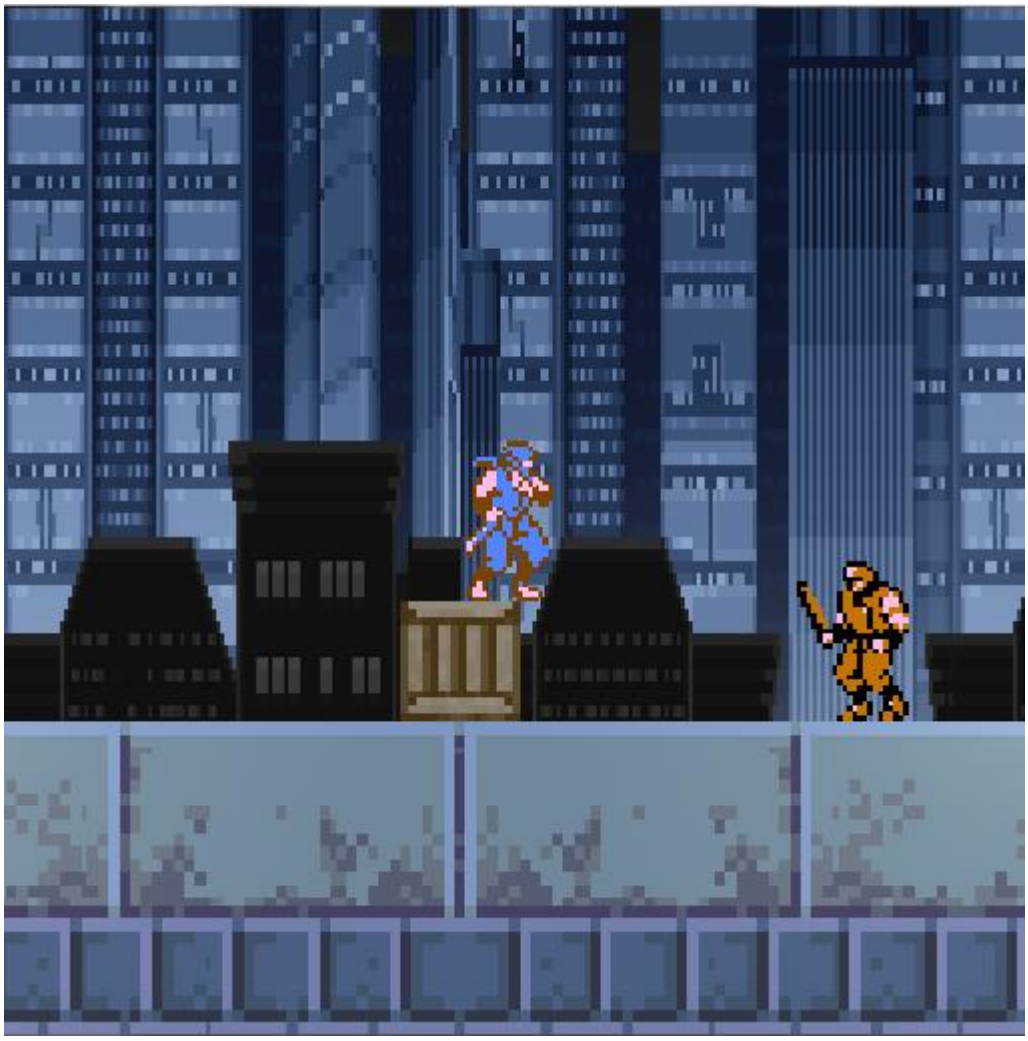

Figura 1

Foram utilizados conceitos clássicos de jogos de plataforma como o desenvolvimento de uma gravidade, sistema de pontuação, existência de inimigos, obstáculos e um sistema de colisão.

DESENVOLVIMENTO DO JOGO NA VERSÃO ORIENTADA A OBJETOS

Para ser possível a organização de todas essas ideias, os conceitos de programação orientada a objeto foram utilizados com a criação de diversas classes que gerenciam e criam as interações existentes no jogo.

Além de simplesmente fazer um jogo funcional, foi necessário seguir uma tabela de requisitos. A mesma se torna necessária para avaliar a utilização dos conceitos apresentados em sala de aula além dos requisitos exigidos são basicamente o que define um jogo de Plataforma. Segue abaixo a tabela de requisitos do jogo e a sua situação com relação a implementação dos mesmos no jogo:

Tabela 1.

N.	**Requisitos Funcionais**	**Situação**
1	Apresentar menu de opções aos usuários do Jogo	Requisito previsto inicialmente e realizado
2	Permitir um ou dois jogadores aos usuários do Jogo	Requisito previsto inicialmente e realizado
3	Disponibilizar ao menos duas fases que podem ser jogadas sequencialmente ou selecionadas.	Requisito previsto inicialmente e realizado
4	Ter 3 tipos distintos de inimigos.	Requisito previsto inicialmente e realizado
5	Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias.	Requisito previsto inicialmente e parcialmente realizado
6	Ter inimigo “Chefão” na última fase	Requisito previsto inicialmente e realizado
7	Ter 3 tipos de obstáculos	Requisito previsto inicialmente e realizado
8	Ter em cada fase entre um e 3 tipos de obstáculos com número aleatório de instâncias.	Requisito previsto inicialmente e realizado

9	Ter representação gráfica de cada instância.	Requisito previsto inicialmente e realizado
10	Ter em cada fase um cenário de jogo com os obstáculos.	Requisito previsto inicialmente e realizado
11	Gerenciar colisões entre jogador e inimigos.	Requisito previsto inicialmente e realizado
12	Gerenciar colisões entre jogador e obstáculos	Requisito previsto inicialmente e realizado
13	Permitir cadastrar/salvar dados do usuário, manter pontuação durante jogo, salvar pontuação e gerar lista de pontuação (*ranking*).	Requisito previsto inicialmente e realizado
14	Permitir Pausar o Jogo	Requisito previsto inicialmente e realizado
15	Permitir Salvar Jogada.	Requisito previsto inicialmente e realizado

Como foi possível observar, 15 dos 15 conceitos foram realizados sendo 2 deles parcialmente. A falta de instâncias aleatórias causou o requisito ter sido cumprido parcialmente.

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

Tabela 2.

N.	Conceito	Uso	Onde
1	Elementares:		
1.1	Classes, objetos	Sim	Todos .h e .cpp
1.2	Atributos (privados), variáveis e constantes	Sim	Todos .h e .cpp
1.3	Métodos (com e sem retorno)	Sim	Todos .h e .cpp
1.4	Métodos (com retorno const e parâmetro const)	Sim	Todos .h e .cpp
1.5	Construtores (com/sem) parâmetros e destrutores	Sim	Todos .h e .cpp
1.6	Classe Principal	Sim	Main.cpp, Jogo.h e Jogo.cpp
1.7	Divisão em .h e .cpp	Sim	No projeto
2	Relações de:		
2.1	Associação	Sim	Classe Fase com Menu...
2.2	Agregação via Associação	Não	
2.3	Agregação propriamente dita	Sim	Classe Fase com Jogo...
2.4	Herança elementar	Sim	Classe Inimigo com Ladrão...
2.5	Herança em diversos níveis	Sim	Classe Entidade com Personagem com Inimigo com Chefão
2.6	Herança múltipla	Não	
3	Ponteiros, generalizações e exceções		
3.1	Operador *this*	Sim	Personagem.cpp

3.2	Alocação de memória (*new/delete*)	Sim	F1.cpp F2.cpp Jogo.cpp
3.3	Gabarito/Template criado/adaptado pelos autores	Não	
3.4	Tratamento de exceções (*try catch)*	Não	
4	Sobrecarga		
4.1	Sobrecarga de construtoras e métodos	Sim	Personagem.cpp
4.2	Sobrecarga de operadores (2 ao menos)	Não	
4.3	Persistência de objeto: texto via arquivo de fluxo	Sim	MenuPausa.cpp
4.4	Persistência de objeto: binário	Sim	MenuPausa.cpp
5	Virtualidade		
5.1	Métodos virtuais	Sim	Personagem.cpp
5.2	Polimorfismo	Sim	Entidade.cpp
5.3	Métodos virtuais puros/classes abstratas	Sim	Entidade.cpp, Obstáculo.cpp
5.4	Coesão e desacoplamento	Sim	Todo o projeto
6	Engenharia de Software		
6.1	Levantamento de requisitos (texto e tabela)	Sim	Todo o projeto
6.2	Diagrama de classes UML	Sim	Todo o projeto
7	Biblioteca Gráfica	Sim	Todo o projeto
7.1	Funcionalidades elementares	Sim	Todo o projeto
7.2	Funcionalidade avançadas: tratamento de colisões áudio duplo buffer	Sim	Tratamento de Colisões – Colisor.h e Colisor.cpp
8	Interdisciplinaridade		
8.1	Ensino médio	Sim	Personagem.cpp
8.2	Ensino Superior (Geometria Analítica)	Sim	Fase.cpp

8.3	Namespace e/ou classes aninhadas	Não	
8.4	Atributo estático e chamada estática de método	Não	
9	Standard Template Library (STL) e String OO		
9.1	Vector STL	Sim	Fase.h
9.2	Classe pré definida String	Sim	Menu.h
10	Uso de conceitos avançados em OO: Programação orientada a eventos e visual	Sim	Fase.cpp

Como demonstrado acima, foram utilizados 30 dos 37 conceitos doutrinados pelo professor em sala de aula, cerca de 81%.

DISCUSSÃO E CONCLUSÕES

Como foi mostrado acima, a maioria dos conceitos discutidos em aula foram utilizados e implementados para ser possível a conclusão do jogo. O trabalho serviu como uma ótima forma de aprendizado e aprimoramento de habilidades tanto no uso de biblioteca gráfica como na programação orientada a objetos.

CONSIDERAÇÕES PESSOAIS

A maior dificuldade encontrada foi a de desenvolver um trabalho que foi feito para mais de uma pessoa criar e por isso a qualidade do trabalho não atingiu as expectativas do seu criador.

DIVISÃO DO TRABALHO

Tabela 3.

Atividades	Responsáveis
Levantamento de requisitos	Henrique Castro
Diagrama de Classes	Henrique Castro
Programação em C++	Henrique Castro
Implementação de persistência de objetos	Henrique Castro
Edição de sprites	Henrique Castro
Implementação do menu	Henrique Castro
Implementação de colisões no geral	Henrique Castro
Implementação dos movimentos no geral	Henrique Castro
Desenvolvimento das fases	Henrique Castro
Implementação do pause	Henrique Castro
Implementação do ranking	Henrique Castro
Edição de cenários	Henrique Castro
Implementação do cadastro de dados do usuário	Henrique Castro
Implementação da opção “salvar”	Henrique Castro
Escrita do trabalho	Henrique Castro
Revisão do trabalho	Henrique Castro