TRABALHO PARA A DISCIPLINA DE TÉCNICAS DE PROGRAMAÇÃO

Julio Werner Scholz ; Yan Pietrzak Pinheiro
juliosscholz@alunos.utfpr.edu.br ; yanp@alunos.utfpr.edu.br

Disciplina: **Técnicas de Programação – CSE20** / S173 – 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 – A disciplina de Técnicas de Programação, ministrada pelo Prof. Dr. Jean M. Simão, exige o desenvolvimento de um software de plataforma em formato de um jogo em duas dimensões, para fins de aprendizado de técnicas relacionadas a programação orientada a objetos em C++. Para tal, foi desenvolvido um jogo de plataforma chamado de "Save the forest". Nele, o personagem deve enfrentar diversos inimigos e obstáculos em dado cenário até chegar na criatura que domina a floresta para então derrotá-lo. O jogo possui duas fases que se diferenciam por dificuldade e personagens. Para seu desenvolvimento, foi feito uma análise e levantamento de requisitos utilizando o Diagrama de Classes e Linguagem de Modelagem Unificada, para então desenvolver em uma linguagem de programação. Foi utilizado o C++ junto com a biblioteca gráfica do Allegro, contemplando diversos conceitos utilizados na orientação a objetos, como: Classe, Objeto, Relacionamento, Herança, Persistência de Objetos, Polimorfismo e Biblioteca Padrão de Gabaritos (Standard Template Library – STL). Após sua implementação, diversos testes foram realizados para demonstrar sua funcionalidade conforme os requisitos propostos. Por fim, o desenvolvimento do projeto permitiu colocar em prática diversos conceitos vistos em sala de aula, havendo um grande enriquecimento.
Palavras-chave: Trabalho acadêmico utilizando C++, Jogo 2D de plataformas em C++, Orientação a objetos, Artigo-relatório para a disciplina técnicas de programação.

Abstract - The discipline of Programming Techniques, taught by Prof. Dr. Jean M. Simão, requires the development of platform software in a two-dimensional game format, for the purpose of learning techniques related to object-oriented programming in C ++. For this, a platform game called "Save the forest" was developed. In it, the character must face several enemies and obstacles in a given scenario until he reaches the creature that dominates the forest and then defeats him. The game has two phases that differ by difficulty and characters. For its development, a requirements analysis and survey was made using the Class Diagram and Unified Modeling Language, to be developed in a programming language. C ++ was used in conjunction with Allegro's graphical library, which contemplates several concepts used in object orientation, such as: Class, Object, Relationship, Inheritance, Object Persistence, Polymorphism and Standard Template Library (STL). After its implementation, several tests were performed to demonstrate its functionality according to the proposed requirements. Finally, the development of the project allowed putting into practice several concepts seen in the classroom, with a great enrichment.
Key-words: Academic work using C ++, 2D Game Platforms in C ++, Object Oriented, Article-Reporting for discipline programming techniques.

INTRODUÇÃO

O trabalho requisitado pelo professor da disciplina de Técnicas de Programação ministrado na Universidade Tecnológica Federal do Paraná. Teve o objetivo de aplicar os conceitos apresentados em sala de aula a fim de desenvolver as habilidades dos participantes. Para seu desenvolvimento, foram utilizados diversos conceitos em relação a Orientação a Objetos presentes na linguagem de programação C++. Além disso, reuniões periódicas foram

requisitadas e realizadas.

Para tal objetivo, foi realizado o desenvolvimento de um software em forma de jogo em duas dimensões. Software que possui sua estrutura em C++, Orientado a Objetos junto a biblioteca gráfica do Allegro. Desenvolvimento que foi realizado em equipe, e orientado pelo professor por meio de reuniões para simular um cenário corporativo.

Para chegar no resultado final, foi feito inicialmente uma análise e levantamento de requisitos junto a uma pesquisa e leitura de materiais a respeito da biblioteca gráfica. Em seguida o projeto foi desenhado utilizando o Diagrama de Classes e Linguagem de Modelagem Unificada (Unified Modeling Language - UML). Em sequência, foi realizado então a codificação e implementação das classes previstas no diagrama de classes. Após a codificação foi realizado os testes necessários com o intuito de verificar qualquer tipo de pendência para então aplicar as devidas correções voltando sempre aos passos iniciais para empregá-las.

O jogo criado consiste em percorrer o cenário de uma floresta. O objetivo é de chegar na segunda fase para matar o chefão que está infectando a floresta. Para tal é necessário ultrapassar todos os desafios encontrados sem que acabe a vida do personagem.

EXPLICAÇÃO DO JOGO EM SI

O jogo criado no trabalho, recebeu o nome de “Save the forest”, isto baseado em seus cenários, personagens e obstáculos que acompanham o tema de floresta (Figura 1).

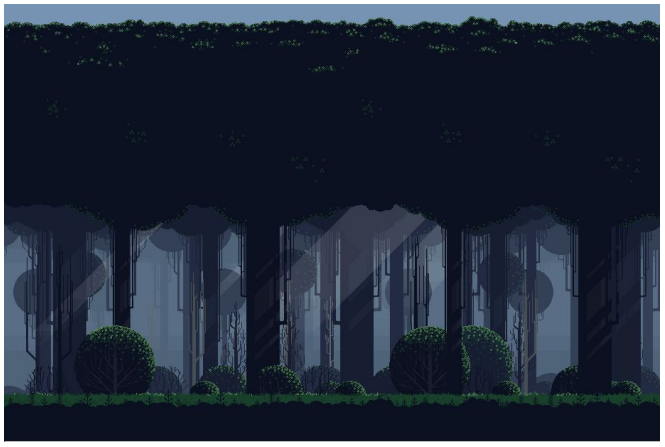

Figura 1. Cenário do jogo.

O jogo possui duas fases. A primeira ao chegar ao final e a segunda após derrotar o chefão e chegar ao término. Fases que podem ser jogadas sequencialmente ou selecionadas no menu principal (Figura 2).

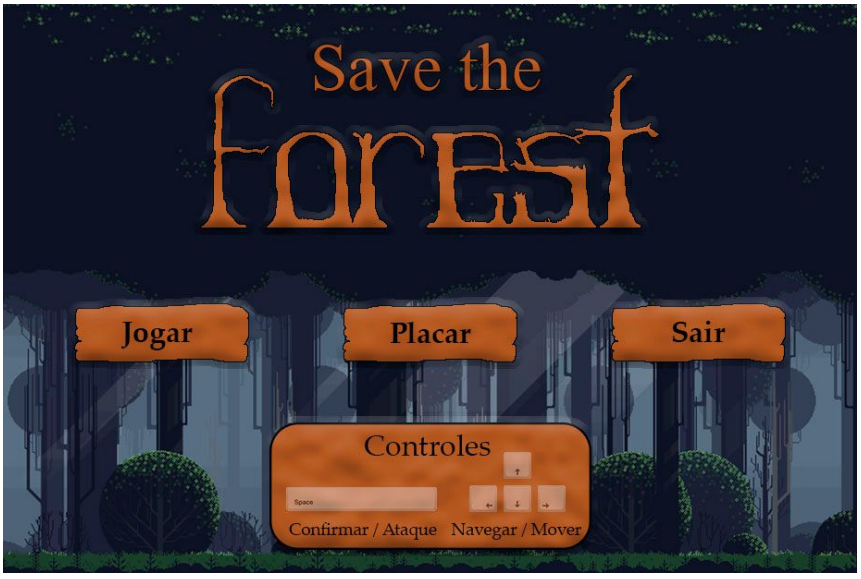

Figura 2. Menu inicial do jogo.

Foram implementados dois tipos de inimigos, um chamado de bocudo (Figura 3), em que seu ataque consiste numa mordida que ocorre ao encostar no inimigo, sua movimentação é terrestre e a vida é limitada.

Figura 3. Bocudo, inimigo presente nas fases 1 e 2.

E o abelhudo (Figura 4) que causa seu dano ao encostar e picar o personagem. Sua movimentação é aérea e ele pode aparecer em diversas posições da fases sendo que possui vida limitada.

Figura 4. Abelhudo, inimigo presente nas fases 1 e 2.

Além disso foi implementado o chefão (Figura 5) na segunda fase, personagem que ataca através de seu tronco de árvore que segura em sua mão.

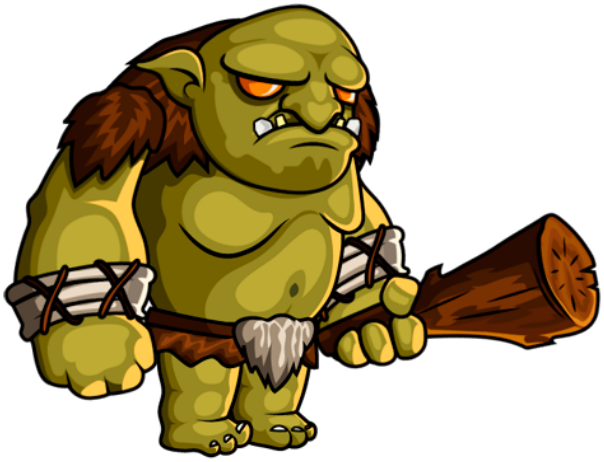

Figura 5. Chefão encontrado no final da segunda fase.

Além dos inimigos foram inseridos obstáculos (Figura 6) nas duas fases, que servem para dificultar a passagem e causam dano se o inimigo encostar neles. Possuem vida ilimitada.

Figura 6. Da esquerda para a direita, Carnívora, Cobreta e espinhos.

DESENVOLVIMENTO DO JOGO NA VERSÃO ORIENTADA A OBJETOS

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

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, sendo que no último caso seria para que os dois joguem de maneira concomitante.	Requisito previsto inicialmente e realizado
3	Disponibilizar ao menos duas fases que podem ser jogadas sequencialmente ou selecionadas.	Requisito previsto inicialmente e realizado parcialmente

4	Ter três tipos distintos de inimigos (o que pode incluir 'Chefão', vide abaixo).	Requisito previsto inicialmente e realizado
5	Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias, podendo ser várias instâncias e sendo pelo menos 5 instâncias por tipo.	Requisito previsto inicialmente e realizado
6	Ter inimigo "Chefão" na última fase	Requisito previsto e realizado
7	Ter três tipos de obstáculos.	Requisito previsto inicialmente e realizado
8	Ter em cada fase ao menos dois tipos de obstáculos com número aleatório de instâncias (i.e., objetos) sendo pelo menos 5 instâncias por tipo.	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 parcialmente realizado
14	Permitir Pausar o Jogo	Requisito previsto inicialmente e realizado
15	Permitir Salvar Jogada.	Requisito previsto e parcialmente realizado

O jogo, em cada fase é feito um loop que fica se repetindo, e a cada repetição ele desenha cada elemento da tela em sua respectiva posição, e quando é necessário movimentação é incrementado valores na posição dos elementos da tela.

O jogo, possui classes denominadas Obstáculos e Inimigos, elas que são derivadas de entidades e que derivam seus respectivos personagens citados anteriormente (Obstáculos: Carnívora, espinho e cobreta)(Inimigos: Chefao, Bocudo e Abelhudo), personagens que possuem seus bitmaps gerenciados na classe Gerenciador_de_bitmaps, mas que são criados na classe ListaEntidades que é instanciada na classe Fase.

Além da lista de entidades, o software possui também uma lista de plataformas, que também é instanciada na Fase.

Os objetos, como Obstáculos e Inimigos necessitam estar em posições definidas para atenderem os requisitos propostos. Para isso, as listas possuem a função inicializar, que cria a lista e define valores para todas as entidades.

É também utilizado a função de salvar jogada, ela é responsável por gerar um arquivo de texto que contém os dados necessários para o software carregar a jogada.

Ao iniciar o jogo um cronômetro é inicializado, e é ele que controla a repetição do loop principal por meio da contagem de frames por segundo. Este cronômetro é pausado ao utilizar o menu de opções dentro do jogo e reiniciado toda vez que se termina uma fase.

Além disto o cronômetro é utilizado para fornecer uma pontuação do jogador no término das fases e então permitir o seu cadastro.

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

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).	Sim	Todos .h e .cpp
		- Métodos (com retorno *const* e parâmetro *const*). - Construtores (sem/com parâmetros) e destrutores	Sim	Todos .h e .cpp
		- Classe Principal.	Sim	Main.cpp & Principal.h/.cpp
		- Divisão em .h e .cpp.	Sim	No desenvolvimento como um todo.
	2	**Relações de:**		

		- Associação direcional. - Associação bidirecional.	Sim Sim	Nas listas de entidades, lista de plataformas
		- Agregação via associação - Agregação propriamente dita.	Sim Sim	Fases, Inimigos e Obstaculos
		- Herança elementar. - Herança em diversos níveis.	Sim Sim	Entidade.cpp, Inimigo.cpp, Obstáculo.cpp, Personagem.cpp
		- Herança múltipla.	Não	
	3	**Ponteiros, generalizações e exceções**		
		- Operador *this*.	Não	
		- Alocação de memória (*new* & *delete*).	Sim	Principal.cpp Fase1.cpp Fase2.cpp ListasEntidades.cpp & ListaPlataformas.cpp
		- Gabaritos/*Templates* criada/adaptados pelos autores (e.g. Listas Encadeadas via *Templates*).	Não	
		- Uso de Tratamento de Exceções (*try catch*).	Não	
	4	**Sobrecarga de:**		
		- Construtoras e Métodos.	Não	
		- Operadores (2 tipos de operadores pelo menos).	Não	
		Persistência de Objetos (via arquivo de texto ou binário)		
		- Persistência de Objetos.	Não	
		- Persistência de Relacionamento de Objetos.	Não	
	5	**Virtualidade:**		

		- Métodos Virtuais.	Sim	Entidade.cpp & Inimigo.cpp
		- Polimorfismo	Sim	Em todos os inimigos e entidades
		- Métodos Virtuais Puros / Classes Abstratas	Sim	Entidade.cpp, Inimigo.cpp, Obstaculo.cpp
		- Coesão e Desacoplamento	Não	
	6	**Organizadores e Estáticos**		
		- Espaço de Nomes (*Namespace*) criada pelos autores.	Não	
		- Classes aninhadas (*Nested*) criada pelos autores.	Não	
		- Atributos estáticos e métodos estáticos.	Sim	Gerenciador_de_Bitmaps.h/.cpp
		- Uso extensivo de constante (*const*) parâmetro, retorno, método...	Sim	Em todos os gets.
	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)	Não Sim	List : ListaEntidades.cpp & ListaPlataformas.cpp
		- Pilha, Fila, Bifila, Fila de Prioridade, Conjunto, Multi-Conjunto, Mapa **ou** Multi-Mapa.	Sim	Multimapa de par de inteiros e Bitmap, no Gerenciador_de_bitmaps.cpp
		Programação concorrente		
		- *Threads* (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando Posix, C-Run-Time **ou** Win32API ou afins.	Não	

		- *Threads* (Linhas de Execução) no âmbito da Orientação a Objetos com uso de Mutex, Semáforos, **ou** Troca de mensagens.	Não	
	8	**Biblioteca Gráfica / Visual**		
		- Funcionalidades Elementares. - Funcionalidades Avançadas como: ● tratamento de colisões ● duplo *buffer*	Sim Sim	Criação de display, desenhar imagens no display, entrada do teclado, timers, fila de eventos.
		- Programação orientada e evento em algum ambiente gráfico. OU - *RAD – Rapid Application Development* (Objetos gráficos como formulários, botões etc).	Sim	Ambiente gráfico: ALLEGRO 5
		Interdisciplinaridades por meio da utilização de Conceitos de Matemática, Física etc		
		- Ensino Médio.	Não	
		- Ensino Superior.	Não	
	9	**Engenharia de Software**		
		- Levantamento e rastreabilidade de cumprimento de requisitos.	Sim	Realizado antes da codificação.
		- Diagrama de Classes em *UML*.	Sim	Realizado antes e durante a codificação
		- Uso intensivo de padrões de projeto (GOF).	Não	
		- Testes a luz da Tabela de Requisitos	Sim	Todos os testes realizados, tiveram como objetivo cumprir os requisitos.
	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).	Sim	Via cópias manuais, e sempre salvando novas versões na nuvem

		- Reuniões com o professor para acompanhamento do andamento do projeto.	Não	2/4 reuniões realizadas nas datas 14/10 e 20/10
		- Reuniões com monitor da disciplina para acompanhamento do andamento do projeto.	Não	5/10 encontros realizados 14/10, 12/10, 19/10, 20/10,21/10
		- Revisão do trabalho escrito de outra equipe e vice-versa.	Não	

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

No.	Conceitos	*Listar apenas os utilizados* Situação
1	**Elementares**	Classe e Objetos foram utilizado porque possibilitam uma melhor organização do código e possibilita a reutilização do código Atributos foram utilizados porque são parte fundamental das classe. Construtores e destrutores foram utilizados pois são essenciais no programação orientada a objetos .Divisão em .h e .cpp, foi utilizado para uma manipulação mais fácil dos códigos e melhorar a organização.
2	**Relações**	Heranças elementares e em diversos níveis foram utilizadas para reaproveitamento do código;
3	**Ponteiros, generalizações e exceções**	Alocação dinâmica de memória, foi utilizada para instanciamento em tempo de execução de objetos
4	**Sobrecarga e persistência de objetos**	
5	**Virtualidade**	Métodos virtuais utilizados para que as funções corretas fossem chamadas independentemente do tipo de referência

6	**Organizadores Estáticos**	Uso extensivo de const em retorno de “gets” assim o ponteiro constante, não pode ter seu endereço alterado, garantindo uma maior segurança.
7	**Standard Template Library *(STL)* e String** **OO** **Programação Concorrente**	Utilização de list, para gerenciar e armazenar inimigos, obstáculos e plataformas. Utilização de map, pois a classe GerenciadorDeBitmaps é baseada em um mapa.
8	**Biblioteca Gŕafica / Visual**	Como pedido, foi utilizada uma biblioteca gŕafica, neste caso ALLEGRO 5.
9	**Engenharia de Software**	Fazer o levantamento de requisitos foi fundamental para o planejamento do Jogo, assim entender o que seria utilizado ou não no jogo, se tornou mais fácil. Em seguida elaborar o diagrama do jogo em UML, proporcionou uma visão mais ampla de todos os aspectos do jogo.
10	**Execução do projeto**	Controle de versões foi fundamental, pois aconteceram eventuais erros e voltar numa versão anterior para poder encontrar o problema foi fundamental.

DISCUSSÃO E CONCLUSÕES

O jogo desenvolvido durante o trabalho, nos permitiu ter uma visão mais ampla a respeito da orientação a objetos e suas aplicações. O seu desenvolvimento facilitou, e muito para entender conceitos e operações relacionadas a orientação a objetos.

Além disso nos permitiu atuar em equipe em cima de seu desenvolvimento. E também permitiu uma visão mais ampla do cenário corporativo em meio ao desenvolvimento de um software requisitado por um cliente.

CONSIDERAÇÕES PESSOAIS

Foi um trabalho extenso e complexo, principalmente ao utilizar uma biblioteca jamais vista, mas que nos permitiu ter uma visualização melhor do que ocorre durante a execução.

Em relação às consequências dele, foi extremamente benéfico, pois foram utilizados diversos conceitos que tinham sido vistos somente em sala de aula, e que após este trabalho foi visto melhor suas aplicações e consequentemente serão utilizados com mais frequência.

DIVISÃO DO TRABALHO

Desde o início do processo de implementação foi feita uma divisão ótima das tarefas, para que ninguém ficasse sobrecarregado, dividindo igualmente o número de tarefas a ser realizado durante o desenvolvimento do jogo.

AGRADECIMENTOS

Agradecemos ao professor por elaborar um projeto voltado a aplicação dos conceitos vistos e apresentados em sala de aula e ao aprimoramento de nossas habilidades.