TRABALHO PARA A DISCIPLINA DE FUNDAMENTOS DE PROGRAMAÇÃO 2 DO CURSO DE ENGENHARIA ELETRÔNICA DA UTFPR: *Mario World*

Gabriel Oliveira Schultz, Walter Hugo
Gabrielschultz57@gmail.com, walterhugo_souza@hotmail.com.br

Disciplina: **Fundamentos de Programação 2** / S11 – Prof. Dr. Jean M. Simão
Departamento Acadêmico de Eletrônica – DAELN - Campus de Curitiba
Universidade Tecnológica Federal do Paraná - UTFPR
Avenida Sete de Setembro, 3165 - Curitiba/PR, Brasil - CEP 80230-901

Resumo –Este relatório apresenta o jogo Mario, realizado na disciplina de Fundamentos de Programação 2, para fins didáticos, em ambiente de linguagem C++, por meio da biblioteca gráfica allegro, bem como o uso de técnicas de engenharia de software. O jogo consiste basicamente em ajudar o nosso herói, Mário, a enfrentar inimigos ao longo das fases, com o objetivo final de salvar a princesa. O projeto foi desenvolvido levantando os requisitos básicos e utilizando-se da construção de um diagrama de classes em UML (*Unified Modeling Language*). O jogo apresenta conceitos elementares em programação C++, bem como conceitos avançados como Polimorfismo, Classe Abstrata e STL (*Standard Template Library*). Pode-se concluir que o projeto permitiu adquirir maior conhecimento e domínio sobre a linguagem C++, assim como nos introduziu aos demais processos no desenvolvimento de softwares de alta complexidade.

Palavras-chave ou Expressões-chave : Fundamentos da Programação, Engenharia de software, Linguagem de programação em C++.

INTRODUÇÃO

O presente relatório apresenta a produção de um software de complexidade relativamente alta, escolhido pelo professor um jogo, em linguagem orientada a objetos C++, com o objetivo de aplicas conceitos aprendidos na disciplina de Fundamentos de Programação 2.

O objeto de estudo escolhido foi o *video-game* dos anos 80 Super Mario Bros, que foi um dos pioneiros e melhor conceituado da plataforma de rolagem lateral. Mario é um encanador que atravessa diversos mundos e fases para recuperar a princesa Peach que foi "sequestrada" pelo vilão Bowser.

O jogo foi desenvolvido usando o ciclo elementar de Engenharia de Software, ou seja, inicialmente foi feita a definição dos requisitos, a sua modelagem via diagrama de classes e após, foi realizada a implementação em código C++, bem como testes de uso do software.

Este relatório apresentará o jogo em si, alguns dos passos de desenvolvimento do mesmo, como levantamento de requisitos até o que foi realmente implementado no jogo. Por fim, o papel de cada aluno no desenvolvimento.

EXPLICAÇÃO DO JOGO EM SI

No jogo, o jogador deve controlar o personagem Mario, passando pelos obstáculos e inimigos para resgatas a princesa.

Logo de início é apresentado um Menu, em que o usuário pode escolher dentre as seguintes opções: Novo Jogo, Ranking, Instruções e Sair. Caso o usuário escolha Novo Jogo, ele terá duas opções: 1 player e 2 Players e se quer jogar em sequência ou uma fase em especifico, após tais "cheklists" o jogo será iniciado. O ranking se obtém com as 5 melhores pontuações, ou seja, quem conseguir adquirir o maior número de estrelas.

Este Arcade apresenta um total de 3 fases que podem ser jogadas sequencialmente ou escolhidas individualmente. Em cada fase há diferentes tipos de inimigos, podendo um mesmo inimigo ser encontrado em todas as fases, e também diferentes tipos de obstáculos no mapa, tais como, tuneis de esgoto, tijolos e escadarias.

Figura 1 . Menu

Figura 2. Instruções

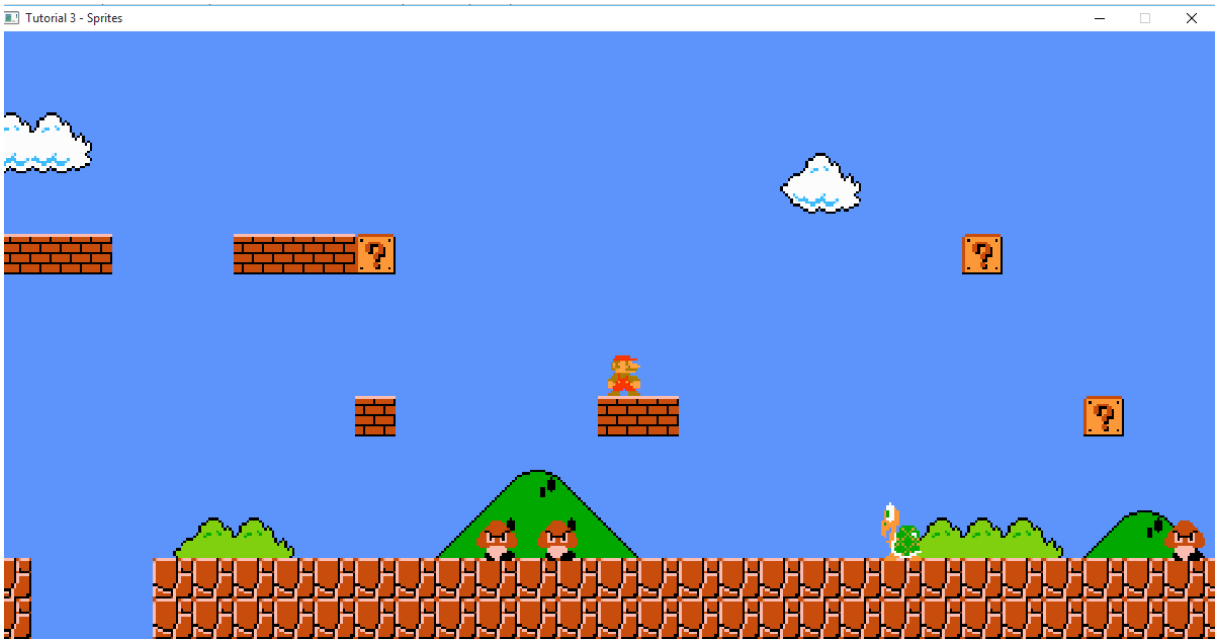

Figura 3 – Fase 1

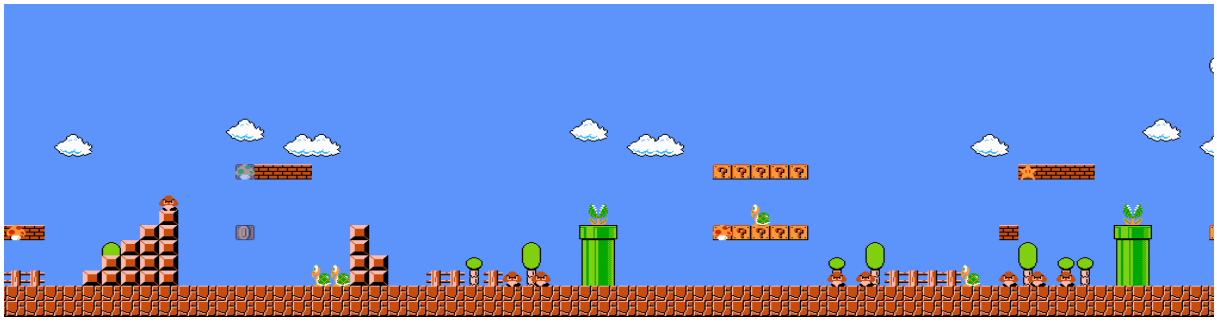

Figura 4 – Fase 2

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	Requisito previsto inicialmente, mas não realizado
3	Disponibilizar ao menos três fases que podem ser jogadas sequencialmente ou selecionadas.	Requisito previsto inicialmente e realizado parcialmente – faltou ainda a última fase.
4	Ter seis tipos distintos de inimigos	Requisito previsto inicialmente e parcialmente realizado
5	Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias, podendo ser várias instâncias.	Requisito previsto inicialmente e realizado
6	Ter inimigo “Chefão” na última fase	Requisito previsto e não realizado
7	Ter quatro tipos de obstáculos.	Requisito previsto inicialmente e realizado
8	Ter em cada fase entre um e quatro tipos de obstáculos com número aleatório de obstáculos.	Requisito previsto inicialmente e realizado
9	Ter representação gráfica de instâncias.	Requisito previsto inicialmente e realizado
10	Ter em cada fase um cenário de jogo com obstáculos.	Requisito previsto inicialmente e realizado
12	Gerenciar colisões entre jogador e inimigos.	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, mas não realizado
14	Permitir Pausar o Jogo	Requisito previsto inicialmente, mas não realizado
15	Permitir Salvar Jogada.	Requisito previsto, mas não realizado

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

Nesta seção, em relação aos conceitos aprendidos, deve-se apresentar uma tabela de conceitos utilizados e não utilizados tal qual a Tabela 2. Deve-se também apresentar outra tabela justificando o uso ou não uso, tal qual a Tabela 3.

Oportunamente, todas as tabelas que venham a ser utilizadas deverão ser numeradas seqüencialmente com algarismos arábicos, conforme o exemplo abaixo:

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

N.	Conceitos	Uso	Onde
1	**Elementares:**		
	- Classes, objetos,	Sim	Todos .h e .cpp
	- Atributos (privados), variáveis e constantes	Sim	Todos .h e .cpp
	- Métodos (com e sem retorno).	Sim	Todos .h e .cpp
	- Métodos (com retorno *const* e parâmetro *const*).	Sim	Todos .h e .cpp
	- 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 projeto.
2	**Relações de:**		
	- Associação		
	- Agregação via associação		
	- Agregação propriamente dita.		
	- Herança elementar.		Personagem.h e Jogador.h / Obstaculo.h e Escada.h
	- Herança em diversos níveis.		
	- Herança múltipla.		
3	**Ponteiros, generalizações e exceções**		
	- Operador *this*		Método "Set" de todas as classes;
	- Alocação de memória (*new* & *delete*)	Não	
	- Gabaritos/*Templates* criada/adaptados pelos autores (e.g. Listas Encadeadas via Templates)	Não	
	- Uso de Tratamento de Exceções	Não	
4	**Sobrecarga de:**		
	- Construtoras e Métodos.	Não	
	- Operadores (2 tipos de operadores pelo menos)		
	Persistência de Objetos		
	- Texto via Arquivos de Fluxo	Não	
	- Binário	Não	
5	**Virtualidade:**		
	- Métodos Virtuais.	Sim	Obstaculo.h
	- Polimorfismo	Sim	Escada.h/Caixa.h
	- Métodos Virtuais Puros / Classes Abstratas	Sim	
	- Coesão e Desacoplamento		
6	**Engenharia de Software**		
	- Levantamento de Requisitos Textualmente e Tabelado (Ou por meio equivalente como Diagrama de Requisitos da SysML)	Sim	
	- Levantamento de Casos de Uso e sua expressão por meio de Diagrama de Casos de Uso em *UML*	Não	
	- Diagrama de Classes em *UML*	Sim	
	- Diagrama de Atividades em *UML*		
	- Outros diagramas em *UML*, Diag. de Estados, Diag. de Seqüência, Diag. de Pacotes etc. - E/ou outros diagramas estabelecidos, como Diag. de Fluxo de Dados (DFD) ou Diag. em SysML (Diag. de Requisitos,. de Blocos etc).	Não	
7	**Biblioteca Gráfica**		
	- Funcionalidades Elementares.	Sim	

	- Funcionalidades Avançadas como: • tratamento de colisões • duplo *buffer* • *especificar aqui outras*	Sim	Função de colisão em Personagem.h/Personagem.cpp Buffer: Método "Draw()" Camera.cpp
	Obs.: especificar quais funcionalidades.		
	Interdisciplinaridades por meio da utilização de Conceitos de Matemática, Física etc		
	- Ensino Médio (especificar quais Conceitos aqui)	Sim	Conceito de velocidade e aceleração Método Update de Jogador.cpp, linha 104.
	- Ensino Superior (especificar quais Conceitos aqui)		
8	**Organizadores:**		
	Espaço de Nomes (*Namespace*) criada pelos autores.	Não	
	Classes aninhadas.	Sim	
	Estáticos e String:		
	Atributos estáticos e chamadas estáticas de métodos.	Não	
	A classe Pré-definida *String* ou equivalente.	Não	
9	**Standard Template Library *(STL)***		
	Vector da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores).		
	List da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores).	Sim	Manipulação presente em Jogador.cpp, método "Update()"
	Pilhas, Filas, Bifilas, Filas de Prioridade, Conjuntos, Multi-Conjuntos, Mapas ***ou*** *Multi-Mapas*.*	Sim	
	*Obs.: Listar apenas os utilizados		
10	**Uso de Conceito Avançado no tocante a Orientação a Objetos.**		
	Ou Padrões de Projeto: *GOF* ***Ou Programação orientada a eventos e visual:*** *Objetos gráficos como formulários, botões etc (*Listar apenas os utilizados) ***Ou Programação concorrente:*** *Threads (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando* Posix, C-Run-Time ***ou*** Win32API ou afins (com ou sem uso de *Mutex, Semáforos,* ***ou*** *Troca de mensagens*). ***Ou API de Comunicação em Rede:*** *Cliente Servidor.*	Não	

COMPARAÇÃO ENTRE DESENVOLVIMENTOS

Observa-se uma melhor organização do projeto, em função do uso do ciclo clássico de engenharia de software, principalmente pela elaboração do projeto utilizando-se de um diagrama classes em UML. A divisão de arquivos em. h e .cpp colaborou para a detecção de erros, além de ser um bom modo de organizar o código fonte. A utilização dos conceitos de herança e permitiu que o código fosse mais compacto, e simples para o programador. Outro ponto importante foi o uso da (STL), e do polimorfismo. Estes métodos, que caracterizam um programa orientado a objetos, não existem sob a análise procedimental. O uso de tais recursos resultou num acoplamento maior entre os distintos módulos do programa, além de possibilitarem a implementação de diferentes soluções.

DISCUSSÃO E CONCLUSÕES

Após a amostragem de todas as tabelas e discussões temos por encerrar a apresentação deste relatório com alguns comentários.

Mesmo não tendo cumprido todos os requisitos da tabela, o jogo apresenta uma boa parte de conhecimentos em linguagem de programação e desenvolvimento de softwares.

Mas é de valia ressaltar seus pontos positivos, o jogo além de possuir uma bela temática tentou seguir o jogo Mario original, o que não é fácil, pois é um jogo extremamente rico em detalhes e sofisticações gráficas. Por fim, o usuário e aquele que poderá um dia ter contato com o jogo poderá disfrutar de um jogo divertido, que poderá oferecer-lhe um bom tempo de apreciação

DIVISÃO DO TRABALHO

.

Tabela 4. Lista de Atividades e Responsáveis.

Atividades.	Responsáveis
Levantamento de Requisitos	Gabriel e Walter
Diagramas de Classes	Gabriel e Walter
Programação em C++	Mais Walter que Gabriel
Criação de arte	Gabriel e Walter
Criação de fases	Gabriel e Walter
Escrita do Trabalho	Gabriel
Revisão do Trabalho	Gabriel