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

José Geraldo Mirachi, Thiago Henrique Pincinato

josemirachi@alunos.utfpr.edu.br, thiago.h1993@hotmail.com

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** – A disciplina de Fundamentos de Programação 2 exige o desenvolvimento de um *software*, no formato de um jogo, para fins de aprendizado de técnicas de engenharia de *software*, particularmente de programação orientada a objetos em C++. Para tal, neste trabalho, escolheu-se o jogo Sonic Boom, onde o jogador terá que chegar até o final da fase passando por alguns inimigos e obstáculos. O jogo possui quatro fases que se diferenciam por dificuldades para o jogador, sendo que a última fase é onde o jogador desafia o chefão do jogo. Para o desenvolvimento do jogo foram levantados textualmente os requisitos, estabelecidos os casos de uso e elaborado uma modelagem (análise e projeto) usando recursos como Diagrama de Classes e Diagrama de Atividades em *Unified Modeling Language* (*UML*). Subsequentemente, em linguagem de programação C++, realizou-se o desenvolvimento que contemplou os conceitos usuais de Orientação a Objetos, bem como alguns conceitos avançados como Polimorfismo, Classe Abstrata, Gabaritos, Persistências de Objetos por Arquivos Binários, Sobrecarga de Operadores e *Standard Template Library* (*STL*). Após a implementação, os testes e uso do jogo demonstraram sua funcionalidade conforme os requisitos e o projeto. Por fim, salienta-se que o projeto permitiu cumprir o objetivo de aprendizado visado.

**Palavras-chave ou Expressões-chave:** Unified Modeling Language (UML), Programação Orientado a Objetos, Engenharia de Software.

**Abstract** - This document shows a model for the manuscript to the academic work of *Fundamentos de Programação 2* as well as it presents general instructions about this academic work. With respect to the abstract contents, it must give a general explanation about the work. Precisely, the abstract must shortly present the work motivation and context, its study object (ordinarily a game), its development process, and the obtained results. An instance of abstract would be:

**. . .**

**Key-words or Key-expressions** (maximum four, not exceeding three lines)**:** Paper Model to the Academic Work of Programming Course, Academic Work Related to C++ Implementation, Internal Rules for Work Elaboration, Examples of Elements for the Work of a Programming Course.

# INTRODUÇÃO

Este trabalhado é proposto visando ampliar a aplicação dos conceitos aprendidos em classe ou mesmo a aplicação de novos conceitos aprendidos extraclasse. Assim sendo, o *software*/jogo escolhido para ser implementado deve ter complexidade tal que permita utilizar diversos recursos da linguagem, sobretudo os ensinados em classe, portanto, deve-se conversar com o Professor da disciplina para verificar se o jogo escolhido se faz ou continua apropriado (caso alguém ainda não o tenha feito...). Não obstante, no decorrer deste documento são apresentados os requisitos mínimos solicitados para cada jogo.

Uma vez escolhido e implementado o jogo, este será expresso em um documento escrito. O documento será entregue no dia da apresentação do trabalho, conforme combinado em classe e especificado no documento de planejamento da disciplina. Tanto a apresentação do desenvolvimento (levantamento de requisitos, modelagem e implementação) do jogo, quanto o documento escrito serão avaliados, permitindo compor uma nota para o trabalho. Bem entendido que o desenvolvimento será avaliado inclusive por meio do acompanhamento do desenvolvimento que se dá pelas duas Atividades Práticas Supervisionadas (APS) e igualmente por demais interações para com o Professor que devem ser solicitadas pelos discentes (i.e. alunos).

Quanto ao trabalho escrito, este deve conter um conjunto de elementos segundo um modelo dado, o qual é detalhado nas seções subsequente deste presente documento. Justamente esse modelo tem por finalidade padronizar o trabalho escrito a ser apresentado na disciplina de Fundamentos de Programação 2 do DAELN/UTFPR. Os trabalhos apresentados que não sigam o padrão aqui apresentado poderão, a critério do Professor, ser penalizados e (no limite) até rejeitados. Idem para trabalhos não escritos corretamente.

Os trabalhos **não** poderão ser entregues ao Professor de maneira impressa (nem mesmo se usar frente e verso). Assim sendo, apenas a versão digital será aceita. Na verdade, é necessário enviar o trabalho escrito em formato digital ***.doc* e *.pdf*** para o *e-mail* do Professor (*jeansimao* ‘arroba’ *utfpr.edu.br*). Também é necessário enviar as implementações respectivas, diagramas de projeto e demais materiais de suporte. Para tal, podem-se utilizar sítios seguros de compartilhamento de arquivos, preferencialmente o Dropbox. Outrossim, os diagramas sim deverão ser entregues de maneira impressa, além de maneira digital, particularmente o Diagrama de Classes em *UML*.

Quanto à introdução em si, mais precisamente, ela deve apresentar quatro parágrafos contento:

(1) em que contexto (*i.e.* disciplina de Fundamentos de Programação 2) este trabalho se dá e qual é o objetivo de tal realização;

(2) qual é o objeto de estudo e da implementação do trabalho (*i.e.* jogo ou outro sistema escolhido e acordado previamente com o Professor);

(3) o método utilizado que em suma é ciclo clássico de Engenharia de *Software* de forma simplificada (i.e. definição dos requisitos, modelagem-projeto via diagramas em *UML*, implementação em C++ orientado a objetos e testes pelo uso do *software*); e

(4) introdução às seções subseqüentes.

Uma vez explicado o necessário à introdução em si, este presente documento-modelo de artigo-relatório apresenta demais seções necessárias (mas não limitantes) que o trabalho deve conter, bem como seus conteúdos. Estas indicações de conteúdos mostram o que se faz necessário contemplar em cada seção, além de explicar alguns itens de formatação de elementos contemplados.

Este trabalho foi elaborado com o objetivo de relatar o desenvolvimento de um jogo utilizando os princípios de Engenharia de Software. Para tal implementou-se o jogo utilizando a linguagem de programação C++. Este projeto está diretamente envolvido com a disciplina de Fundamentos de Programação 2 que visa a programação utilizando os conceitos de Orientação a Objetos.

O objetivo de estudo deste projeto é a implementação de um jogo no estilo plataforma utilizando, da melhor forma possível, os conceitos de orientação à objetos, ou seja, o jogo deve ser implementado de forma que sua programação fique desacoplada e encapsulada.

Para realização deste trabalho, baseando-se no ciclo clássico de Engenharia de Software, primeiramente foi levantado os requisitos do jogo, conforme mostra a tabela 1. Após o levantamento dos requisitos foi efetuado a modelagem do jogo utilizando os diagramas em UML, tais como Diagrama de Atividades, Diagrama de Casos e Usos e Diagrama de Classes.

Introdução às seções subsequentes.

# EXPLICAÇÃO DO JOGO EM SI

Nesta seção se deve discorrer a explicação do jogo em si. Portanto, salienta-se que esta seção **não** é (em absoluto) para explicar o projeto ou a implementação do jogo.

Isto dito, esta seção assim como as demais, devem seguir as regras de formatação dadas. Justamente, quanto à formatação, o texto do trabalho deve seguir as seguintes regras (as quais foram em geral seguidas para compor este presente modelo):

* O trabalho deve ser totalmente digitado em fonte Times New Roman. Esta diretriz inclui, portanto, o título do trabalho, autores, filiação e endereços, títulos de seções e legendas de figuras e tabelas, além do texto normal do trabalho. O texto deve ser digitado com alinhamento ‘justificado’ ou ajustado.
* O trabalho completo, incluindo figuras e tabelas, deve ser limitado a doze (12) páginas (no máximo) em papel de tamanho padrão A4 (21 cm x 29,7 cm). Não reduzir figuras e tabelas a tamanhos que sacrifiquem o entendimento dos símbolos e legendas nelas contidos.
* Cada página, no tamanho A4, deve ser formatada de modo a apresentar 2,5 cm de margem em todos os lados do documento. Dentro desta área o texto deve ser formatado em uma única coluna, sem incluir moldura no texto**.**
* O título deve ser digitado em negrito, em letras maiúsculas, centralizado e em tamanho 14 pt, não excedendo três linhas, seguido de uma linha em branco (12 pt) e pelas linhas que conterão o(s) nome(s) do(s) autor(es), em tamanho 12 pt. Em seguida, deverá vir a filiação e o(s) endereço(s) para correspondência do(s) autor(es) (tamanho 10 pt) separada por uma linha em branco. Deve-se deixar 3 linhas de espaço antes do resumo, e uma linha entre os itens subsequentes (palavras-chave, *abstract* e *Key-words*).
* Digitar o título **Resumo** em negrito, alinhado à esquerda, tamanho 10 pt, seguido de um traço. Sem trocar de linha, digitar o resumo, em tamanho 10 pt com alinhamento justificado. Pular uma linha e digite o título **Palavras-chave:** em negrito, alinhado à esquerda, tamanho 10 pt. Digitar então no máximo quatro (4) palavras-chave, separadas por vírgulas, com somente a primeira letra de cada palavra chave em maiúscula. Na sequência devem vir o ***Abstract*** e ***Key-words*:** em inglês, seguindo o mesmo padrão de formatação do resumo e das palavras-chave. Tanto o resumo quanto o *abstract* devem conter no máximo 200 palavras.
* A seguir, separado por 2 linhas (12 pt), o texto deve ser iniciado pela Introdução. Os títulos das seções (Introdução, etc.) devem ser escritos em negrito, sem numeração, em maiúsculo e alinhados à esquerda, sendo que o conteúdo, propriamente dito, deve ser iniciado após espaçamento de uma linha e tabulação (1 cm).
* Ao final de cada seção deve-se deixar uma linha em branco. Todo o texto deverá ser escrito em espaço simples. Para as subseções, somente a primeira letra do subtítulo deve ser maiúscula, sendo todas em negrito, sem numeração, com o título alinhado à esquerda. Uma vez escrito o subtítulo, pular uma linha. Após esta linha (em branco), iniciar o texto da subseção.
* As ilustrações e gráficos podem ser em preto-e-branco ou em escala cinza ou mesmo coloridos, mas sempre centralizados. As notas de rodapé[[1]](#footnote-1) devem ser colocadas na parte inferior da página correspondente separadas por um traço conforme modelo. Usar o tamanho de 8 pt.
* As referências bibliográficas devem ser listadas no fim do artigo, na ordem de citação, conforme formato da Associação Brasileira de Normas Técnicas (ABNT). No texto, as citações devem ser referenciadas por seu número colocado entre colchetes, por exemplo, [1] e [2].

Em tempo, para melhor explicar o Jogo em Si, aconselha-se utilizar de recursos como gráficos, telas e figuras do próprio jogo.

A propósito, as figuras, tabelas, etc., devidamente referenciadas no texto, podem ser colocadas da maneira mais conveniente para o autor em uma ou duas colunas, desde que o texto permaneça em apenas uma coluna. Antes e após os elementos não textuais e suas respectivas legendas, deve-se deixar uma linha de espaçamento.

Os autores não devem se esquecer da colocação de legendas nas figuras, tabelas e outros elementos gráficos. As figuras devem ser numeradas seqüencialmente com algarismos arábicos conforme o exemplo da figura 1.



Figura 1. Centralizada na coluna e com legenda abaixo da figura.

Aproveitando o ensejo, talvez os autores façam uso de equações em alguma parte do texto. Neste âmbito, todas as equações deverão ser tabuladas a 1 cm da margem esquerda e numeradas seqüencialmente, com os números entre parênteses, conforme o exemplo abaixo:

(1) 

As equações devem ser referenciadas no texto da seguinte forma: "Substituindo a equação (1) na equação (3), obtém-se ..."

Sonic Boom é um jogo de plataforma em deslocação lateral (2D), cuja jogabilidade centra-se na capacidade de Sonic correr em alta velocidade através de níveis que incorporam molas, quedas altas e loops verticais. Os níveis têm perigos na forma de robots (inimigos), dentro dos quais o Dr. Robotnik aprisionou animais. Destruir um robot liberta o animal dentro dele, mas tal não é necessário para completar o jogo. O jogador deve evitar filas de espinhos afiados, cair em poços sem fundo, ser esmagado por paredes e plataformas movediças e a morte por afogamento. O principal ataque de Sonic é a sua própria rotação que, quando se enrola, fica como uma bola e gira rapidamente causando danos aos inimigos. O ataque pode ser feito quando o jogador salta ou rola no chão.

Espalhados pelos níveis estão alguns anéis de ouro, e coletando 100 destes o jogador é recompensado com uma vida extra.

O jogo está divido em três zonas (Green Hill, Marble, Spring Yard, Labyrinth, Star Ligh e Scrap Brain), cada uma com seus próprios estilos visuais e inimigos. Após completar os três níveis o jogador confronta o Dr. Robotnik numa luta de chefe.

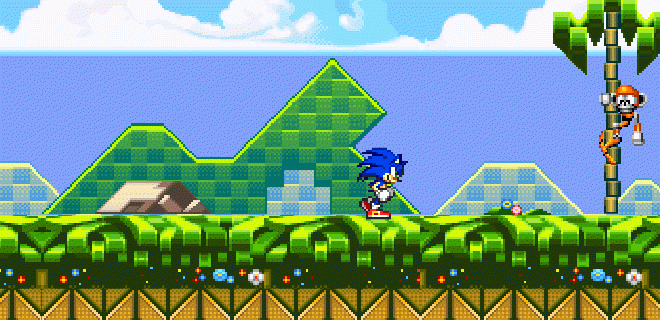


Figura 1. Fase 01 do Jogo



Figura 2. Sonic combatendo Dr. Robotnik

# DESENVOLVIMENTO DO/DE JOGO NA VERSÃO PROCEDIMENTAL

Idealmente, o jogo ou *software* em questão teria sido implementado em disciplina anterior sob o viés de abordagem procedimental (i.e. ‘orientação’ a procedimentos ou funções), o que permitiria subseqüentemente tecer reflexão comparativa entre a programação (e mesmo projeto) procedimental e a orientada a objetos.

Isto dito, nesta seção se deveria discorrer a explicação resumida do desenvolvimento do jogo ou *software* em questão caso ele já tivesse sido implementado anteriormente sob o viés de abordagem procedimental em linguagem C (ou mesmo em outra linguagem ‘procedimental’).

Entretanto, caso o jogo ou *software* em questão não tenha sido previamente implementado sob a abordagem procedimental, pode-se então discutir e mesmo apresentar (sucintamente) algum outro jogo ou *software* feito anteriormente, utilizando abordagem procedimental.

Isto também permitiria mostrar que se teria experiência prévia para tecer reflexão comparativa entre a abordagem procedimental e a orientada a objetos. Tal comparação se daria na seção chamada COMPARAÇÃO ENTRE DESENVOLVIMENTOS explicada mais ao final deste modelo de artigo-relatório.

Nesta explicação em questão, sobre desenvolvimento procedimental, aconselhar-se-ia:

* Listar sucintamente os requisitos funcionais do jogo/*software* visado.
* Utilizar um fluxograma para explicar a dinâmica do jogo/*software* visado e mesmo utilizar parte do código. Este último, caso realmente se faça necessário e enquadrado em uma moldura com se fosse uma figura.
* Utilizar, até mesmo, um ‘diagrama de blocos’ e/ou ‘diagrama de fluxo de dados’ para explicar os módulos existentes (e.g. funções ou conjunto de funções) e suas conexões.
* Valorizar as ‘sofisticações’ que tenham sido realizadas, como uma eventual função que permita ao jogador humano enfrentar um dado ‘jogador artificial’.
* **Evitar frases que não reflitam a realidade** como “implementou-se a inteligência artificial”. Este exemplo dado não reflete a realidade porque certamente os autores não implementaram todo conteúdo da disciplina/domínio de Inteligência Artificial. Neste caso o correto seria: “implementou-se uma técnica ou solução que pode ser classificado como uma solução (elementar) de Inteligência Artificial”.

O projeto apresentado, em sua versão procedimental, ou seja, orientado a procedimentos ou funções, é apresentado abaixo através de um fluxograma.

Diagrama_de_Atividades.png

Figura 3. Fluxograma Versão Procedimental

# DESENVOLVIMENTO DO JOGO NA VERSÃO ORIENTADA A OBJETOS

Nesta seção se deve discorrer a explicação do desenvolvimento do jogo/*software* utilizando orientação a objeto, culminando na programação em C++. A explicação deve-se ser feita de maneira tal a **não** ser um relatório técnico repleto de detalhes, mas que seja capaz de sintetizar e valorizar os recursos técnicos utilizados (i.e. sucinto e suficiente).

Nesta explicação, deve-se primeiramente listar (normalmente textualmente) os **requisitos funcionais** levantados para o jogo/*software* em questão (certamente aproveitando os esforços da APS1). Os requisitos devem estar enquadrados em uma tabela de duas colunas onde a primeira coluna traz os requisitos e a segunda a sua situação (*status*) que pode ser ‘realizado’, ‘semi realizado’, ‘abandonado’ etc. **A Tabela 1 exemplifica o exposto definindo, ademais, os requisitos mínimos que cada jogo deve ter**. Ainda, os requisitos também devem ser convertidos em Casos de Uso e expressos por meio de Diagramas de Casos de Uso em *UML*.

Para o desenvolvimento do jogo, em sua versão orientada a Objetos, foram levantados alguns requisitos funcionais conforme é mostrado na Tabela 1.

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 |  |
| 3 | Disponibilizar ao menos três fases que podem ser jogadas sequencialmente ou selecionadas. |  |
| 4 | Ter seis tipos distintos de inimigos |  |
| 5 | Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias, podendo ser várias instâncias. |  |
| 6 | Ter inimigo “Chefão” na última fase |  |
| 7 | Ter quatro tipos de obstáculos. |  |
| 8 | Ter em cada fase entre um e quatro tipos de obstáculos com número aleatório de obstáculos. |  |
| 9 | Ter representação gráfica de instâncias. |  |
| 10 | Ter em cada fase um cenário de jogo com obstáculos. |  |
| 11 | Gerenciar colisões entre jogador e inimigos. |  |
| 12 | Gerenciar colisões entre jogador e inimigos. |  |
| 13 | Permitir cadastrar/salvar dados do usuário, manter pontuação durante jogo, salvar pontuação e gerar lista de pontuação (*ranking*). |  |
| 14 | Permitir Pausar o Jogo |  |
| 15 | Permitir Salvar Jogada. |  |
| 16 |  |  |

Isto feito a explicação do desenvolvimento segue devendo-se:

* “Converter” requisitos em Casos de Uso e expressá-los por meio de Diagramas de Casos de Uso em *UML*.
* Utilizar um Diagrama de Classes em *UML* para explicar as classes e suas relações (que normalmente atendem aos requisitos).
* Utilizar um Diagrama de Atividades (“Fluxograma”) em *UML* para explicar a dinâmica do programa e utilizar até mesmo parte do código (caso isto realmente se faça necessário).
* Utilizar demais diagramas da *UML* **caso** o(s) aluno(s) os saiba(m) ou esteja(m) fazendo esforços para apreendê-los.
* A luz dos diagramas, explicar o desenvolvimento de maneira sucinta e suficiente.
* Valorizar as ‘sofisticações’ que tenham sido realizadas, como uma eventual função que permita ao jogador humano enfrentar um dado ‘jogador artificial’.
* Valorizar a interdisciplinaridade como a aplicação de conceitos de física e matemática aprendidos em disciplinas do ensino médio e (preferencialmente) em disciplinas da graduação.

Para a implementação em C++ deve-se usar uma biblioteca gráfica (*e.g*. *Allegro*, *TCL*/*TK*, *OpenGL* ou outra), pois isto valoriza o trabalho esteticamente além de demonstrar a capacidade de ‘pesquisa’ e aprendizado. Não deixe de valorizar esta capacidade de pesquisa e aprendizado nesta seção.

Esta seção em questão é muito importante no trabalho e será corrigida com muita atenção pelo Professor. Pede-se, por fim, que todos os autores revisem cuidadosamente a versão final do trabalho para evitar erros de português, digitação e/ou formatação. Na verdade, além disto, uma equipe poderia revisar o trabalho escrito da outra e vice-versa para fins de aprimoramento.

Após o levantamento de forma textual dos requisitos, foi elaborado os diagramas de atividades e de casos e usos em UML, conforme mostrado nas figuras 4 e 5.

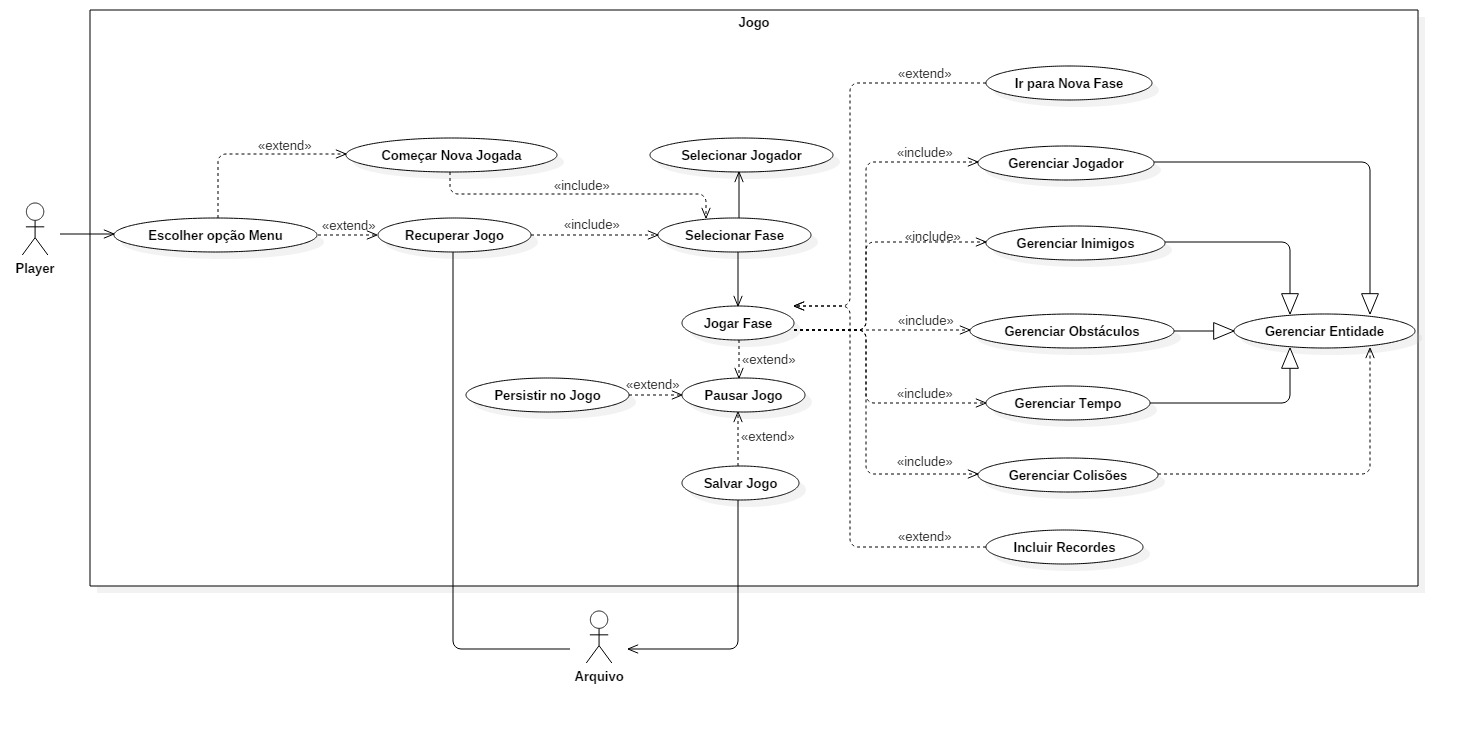


Figura 4. Diagrama de Casos e Usos

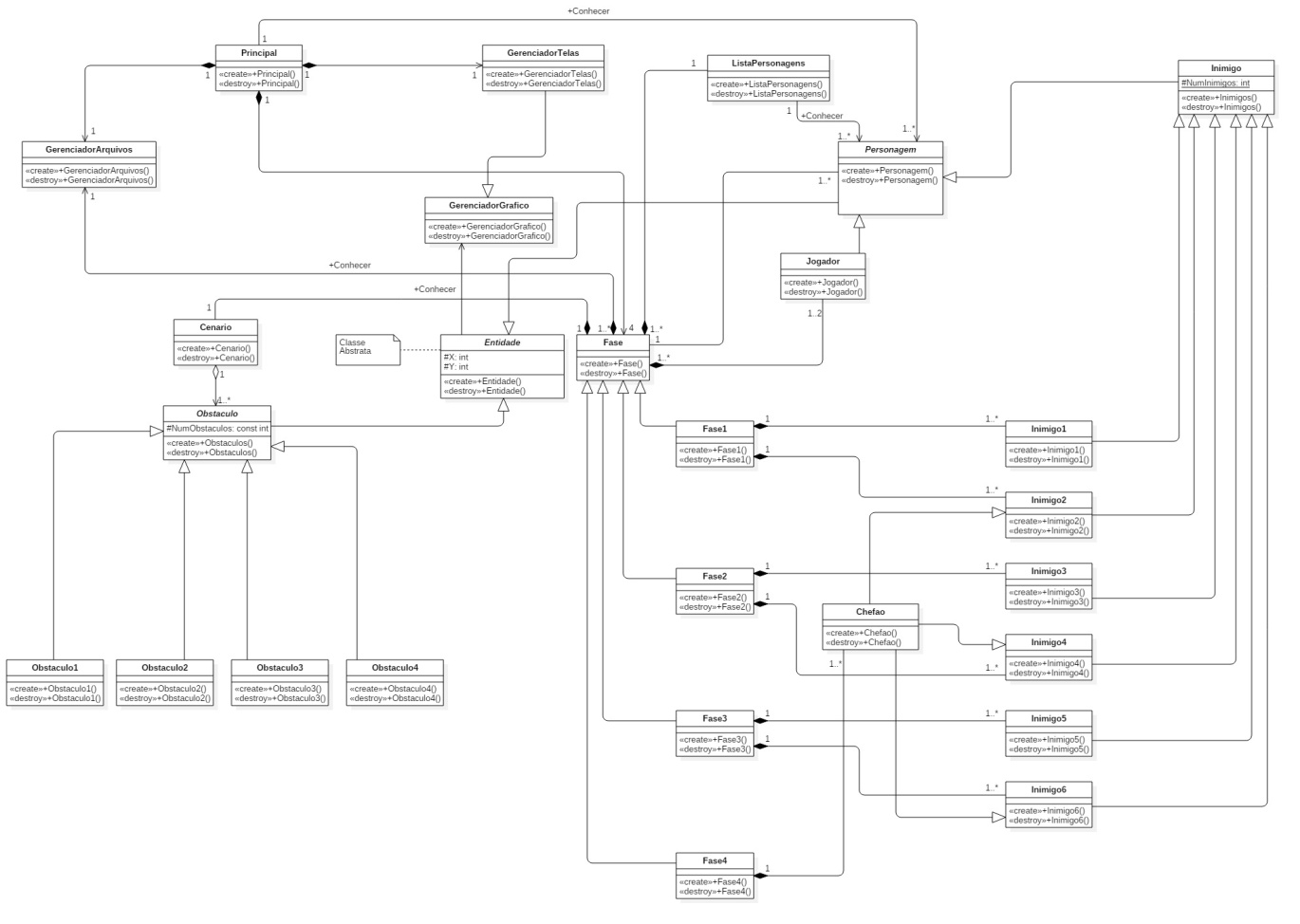


Figura 5. Diagrama de Classes

# 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:

Para realização do jogo foram utilizados alguns conceitos aprendidos durante o semestre na disciplina de Fundamentos de Programação 2. Tais conceitos são apresentados na tabela 2.

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. | |  | |  | |
| - Herança em diversos níveis. | |  | |  | |
| - Herança múltipla. | |  | |  | |
| **3** | | **Ponteiros, generalizações e exceções** | | | | | |
| - Operador *this* | |  | |  | |
| - Alocação de memória (*new* & *delete*) | |  | |  | |
| - Gabaritos/*Templates* criada/adaptados pelos autores (e.g. Listas Encadeadas via Templates) | |  | |  | |
| - Uso de Tratamento de Exceções | |  | |  | |
| **4** | | **Sobrecarga de:** | | | | | |
| - Construtoras e Métodos. | |  | |  | |
| - Operadores (2 tipos de operadores pelo menos) | |  | |  | |
| **Persistência de Objetos** | | | | | |
| - Texto via Arquivos de Fluxo | |  | |  | |
| - Binário | |  | |  | |
| **5** | | **Virtualidade:** | |  | |  | |
| - Métodos Virtuais. | |  | |  | |
| - Polimorfismo | |  | |  | |
| - Métodos Virtuais Puros / Classes Abstratas | |  | |  | |
| - Coesão e Desacoplamento | |  | |  | |
| **6** | | **Engenharia de Software** | | | | | |
| **- Levantamento de Requisitos Textualmente e Tabelado**  (Ou por meio equivalente como Diagrama de Requisitos da SysML) | |  | |  | |
| **- Levantamento de Casos de Uso e sua expressão por meio de Diagrama de Casos de Uso em *UML*** | |  | |  | |
| **- Diagrama de Classes em *UML*** | |  | |  | |
| **- 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). | |  | |  | |
| **7** | | Biblioteca Gráfica | | | | | |
| - Funcionalidades Elementares. | |  | |  | |
| - Funcionalidades Avançadas como:   * tratamento de colisões * duplo *buffer* * *especificar aqui outras* | |  | |  | |
| *Obs.: especificar quais funcionalidades.* | | | | | |
| **Interdisciplinaridades por meio da utilização de Conceitos de Matemática, Física etc** | | | | | |
| - Ensino Médio(especificar quais Conceitos aqui) | |  | |  | |
| - Ensino Superior(especificar quais Conceitos aqui) | |  | |  | |
| **8** | | **Organizadores:** | | | | | |
| Espaço de Nomes (*Namespace*) criada pelos autores. | |  | |  | |
| Classes aninhadas. | |  | |  | |
| **Estáticos e String:** | |  | |  | |
| Atributos estáticos e chamadas estáticas de métodos. | |  | |  | |
| A classe Pré-definida *String* ou equivalente. | |  | |  | |
| **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). | |  | |  | |
| Pilhas, Filas, Bifilas, Filas de Prioridade, Conjuntos, Multi-Conjuntos, Mapas **ou** Multi-Mapas\*. | |  | |  | |
| \*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.* | |  | |  | |

Os conceitos utilizados (ou não) para modelagem do jogo, conforme listados na tabela 2, são justificados na tabela 3.

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 ...  Atributos, Variáveis e Constantes foram utilizados porque ...  . . . |
| 2 | **Relações** | Associação foi utilizado porque.... |
| ... | . . . | ... |
| 10 | **Uso de conceitos avançados.** | Padrões de Projeto não foi utilizado porque.... |

# COMPARAÇÃO ENTRE DESENVOLVIMENTOS

Nesta seção se deve apresentar uma comparação geral entre o desenvolvimento orientado a objetos e o desenvolvimento procedimental segundo a percepção dos desenvolvedores nos trabalhos realizados no âmbito da disciplina de Fundamentos de Programação 1 e Fundamentos de Programação 2 (ou até no âmbito de outras disciplinas se for pertinente).

# DISCUSSÃO E CONCLUSÕES

Esta seção deverá apresentar uma reflexão sobre o desenvolvimento e os resultados obtidos. Certamente uma conclusão bem elaborada ajuda na avaliação do Professor. Por sua vez, a avaliação do trabalho como um todo (pelo Professor) será baseada em:

* Quantidade e qualidade de conceitos utilizados na elaboração do *software*, envolvendo particularmente os bons princípios de Orientação Objetos, como organização, encapsulamento e reutilização, todos baseados no princípio de coesão e desacoplamento doutrinados em classe.
* Complexidade do problema e respectiva quantidade e qualidade de projeto e código o que, novamente, envolve a correção na aplicação dos princípios da Orientação a Objetos (i.e. coesão, desacoplamento, encapsulamento, organização e reutilização), o número de classes, o número e a forma de relacionamento, a complexidade algorítmica etc.
* Qualidade do trabalho escrito e da apresentação.

Em termos gerais, pode-se considerar o primeiro item com um peso de 35%, o segundo com um peso de 35% e o terceiro com um peso de 30%, lembrando que eles estão inter-relacionados (sobretudo o primeiro e o segundo). Não obstante, esta porcentagem é relativa, pois (por exemplo) um item muito bem desenvolvido pode até compensar (um pouco) outro não tão bem desenvolvido em proporções diferentes deste referencial dado. Lembrar ainda que, além das *APS*, fazer seu projeto ser acompanhado pelo Professor (a medida que avança) é fundamental.

# CONSIDERAÇÕES PESSOAIS

Caso o aluno ou os alunos desejam expressar algum sentimento relativo, por exemplo, aos aprendizados e dificuldades encontrados, isto deve ser feita nesta seção opcional. Nesta seção pode-se até utilizar primeira pessoa, entretanto seria melhor a forma impessoal.

Neste sentido, **todas as demais seções devem ser escritas de forma impessoal** (o que significa não usar primeira e segunda pessoa singular ou plural – em suma não usar “eu” ou “nós” no texto). Ademais, salienta-se que o trabalho deve ser redigido linguagem correta, na forma culta e sem exageros poéticos, com textos não prolixos e bem encadeados.

# DIVISÃO DO TRABALHO

Esta seção deverá ter uma tabela salientando quem desenvolveu cada classe/módulo do *software* e realizou demais atividades como as de ‘engenharia de *software’*, a redação do trabalho escrito, a revisão da redação do trabalho e a preparação da apresentação do trabalho.

Tabela 4. Lista de Atividades e Responsáveis.

|  |  |
| --- | --- |
| Atividades. | Responsáveis |
| Levantamento de Requisitos | Fulano e Ciclano |
| Diagramas de Classes | Fulano e Ciclano |
| Programação em C++ | Fulano e Ciclano em geral |
| Implementação de *Template* | Fulano |
| Implementação da Persistência dos Objetos... | Ciclano |
| ... |  |
| Escrita do Trabalho | Mais Fulano que Ciclano |
| Revisão do Trabalho | Mais Ciclano que Fulano |

# AGRADECIMENTOS

Havendo agradecimentos de ordem profissional, estes deverão vir antes das referências.

Aqui se pode salientar e agradecer caso outra equipe tenha revisado o trabalho.

# REFERÊNCIAS CITADAS NO TEXTO

[1] DEITEL, H. M.; DEITEL, P. J. C++ Como Programar. 5ª Edição. Bookman. 2006.

[2] STADZISZ, P. C. Projeto de Software usando UML. Apostila CEFET-PR 2002.

<http://www.dainf.ct.utfpr.edu.br/~jeansimao/Fundamentos2/EngSoftware/Apostila%20UML%20-%20Stadzisz%202002.pdf>

[3] SIMÃO, J. M. Site das Disciplina de Fundamentos de Programação 2, Curitiba – PR, Brasil, Acessado em 09/09/2015, às 15:15:

<http://www.dainf.ct.utfpr.edu.br/~jeansimao/Fundamentos2/Fundamentos2.htm>.

# REFERÊNCIAS UTILIZADAS NO DESENVOLVIMENTO

[A] BEZERRA, E. Princípios de Análise e Projeto de Sistemas com UML. Editora Campus. 2003. ISBN 85-352-1032-6..

[B] HORSTMANN, C. Conceitos de Computação com o Essencial de C++, 3ª edição, Bookman, 2003, ISBN 0-471-16437-2

1. Exemplo de nota de rodapé – Apenas os diagramas em UML que devem ser impressos, preferencialmente em papel reciclado. [↑](#footnote-ref-1)