
LAGGANDINO
Mário Cordeiro Junior
Técnicas de Programação - S73

Requisitos Funcionais

Número	Requisito	Situação
1	Apresentar graficamente menu de opções aos usuários do Jogo.	Requisito previsto inicialmente e realizado. (100%)
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 e realizado. (100%)
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 e realizado. (100%)
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 projéteis contra o(s) jogador(es) e um dos inimigos deve ser um 'Chefão'.	Requisito previsto inicialmente e realizado. (100%)
5	Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias, podendo ser várias instâncias e sendo pelo menos 3 instâncias por tipo.	Requisito previsto inicialmente e realizado. (100%)

Requisitos Funcionais

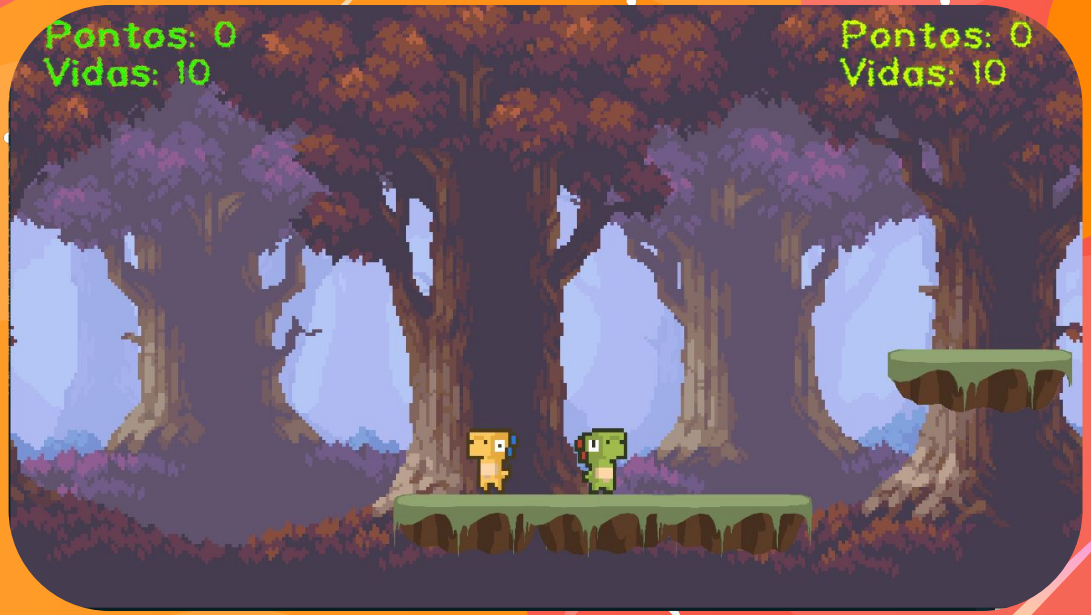
Número	Requisito	Situação
6	Ter três tipos de obstáculos, cada qual com sua representação gráfica, sendo que ao menos um causa dano em jogador se colidirem.	Requisito previsto inicialmente e realizado. (100%)
7	Ter em cada fase ao menos dois tipos de obstáculos com número aleatório de instâncias (i.e., objetos), sendo pelo menos 3 instâncias por tipo.	Requisito previsto inicialmente e realizado. (100%)
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. (100%)
9	Gerenciar colisões entre jogador para com inimigos e seus projéteis, bem como entre jogador para com obstáculos. Ainda, todos eles devem sofrer o efeito da gravidade no âmbito deste jogo de plataforma vertical e 2D.	Requisito previsto inicialmente e realizado. (100%)
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 realizado. (100%)

Requisito 1

Apresentar graficamente menu de opções aos usuários do Jogo

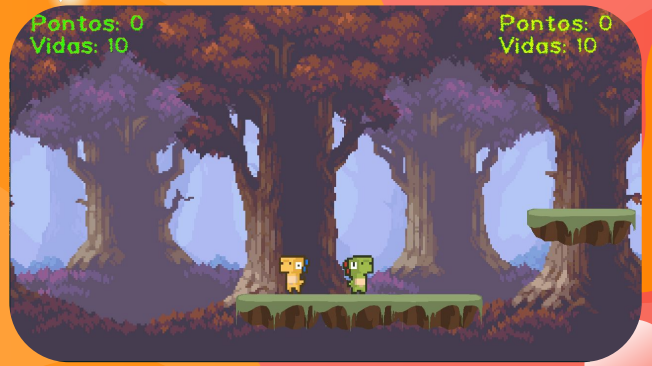
Requisito 2

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.



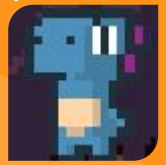
Requisito 3

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 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 projéteis contra o(s) jogador(es) e um dos inimigos deve ser um 'Chefão'.

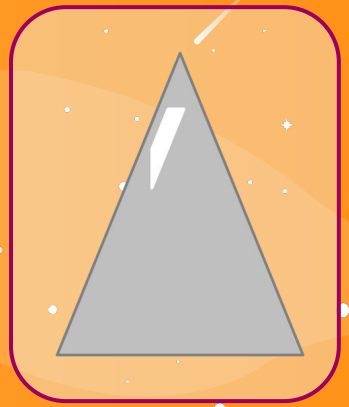


Requisito 5

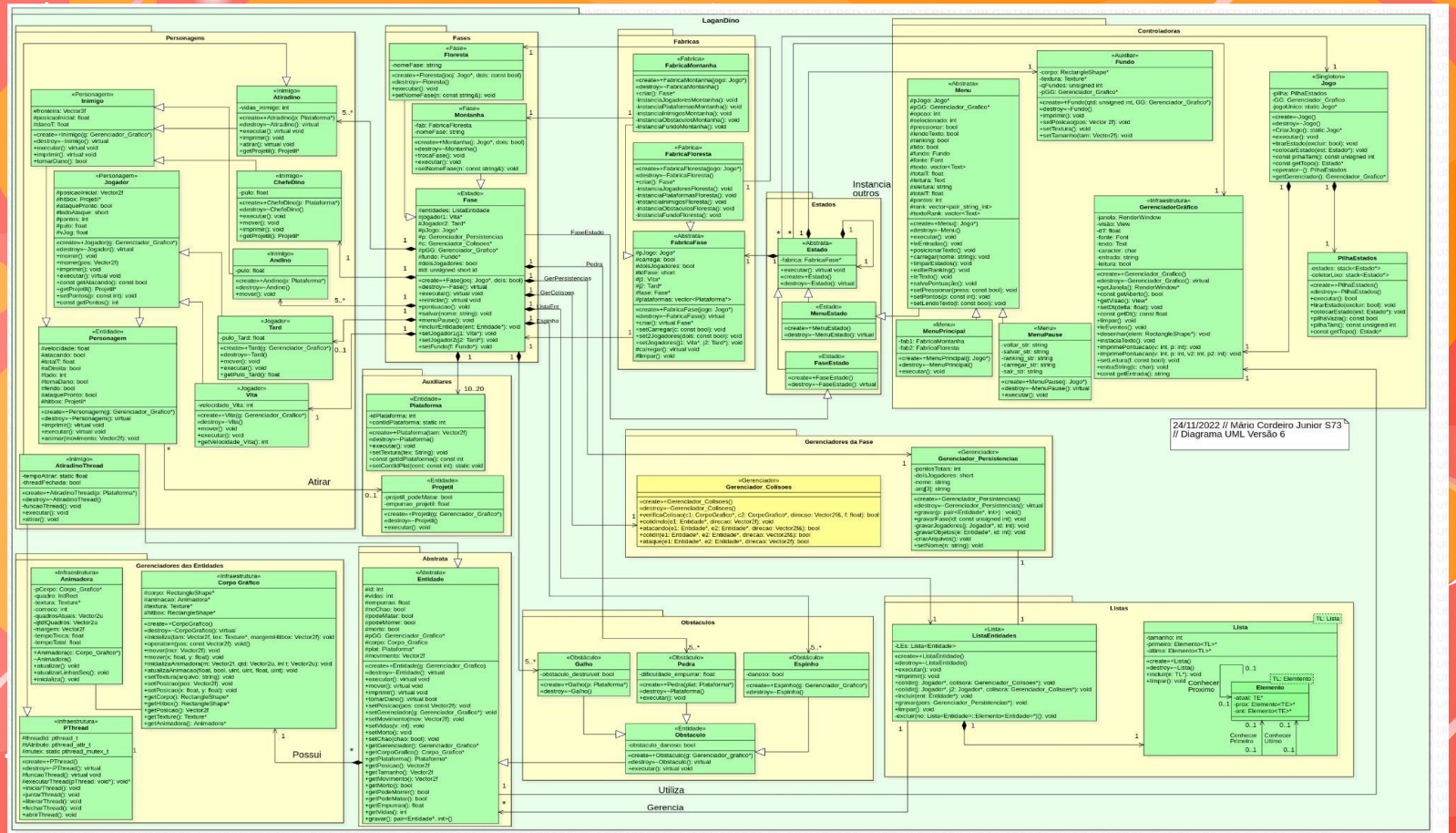
Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias, podendo ser várias instâncias e sendo pelo menos 3 instâncias por tipo.

Requisito 6

Ter três tipos de obstáculos, cada qual com sua representação gráfica, sendo que ao menos um causa dano em jogador se colidirem.



UML


Personagens
«Personagem»
Inimigo
«Inimigo»
Atiradino
«Personagem»
Jogador
«Inimigo»
ChefeDino
«Inimigo»
Andino
«Jogador»
Tard
«Jogador»
Vita
«Entidade»
Personagem
«Inimigo»
AtiradinoThread
Fases
«Fase»
Floresta
«Fase»
Montanha
«Estado»
Fase
Auxiliares
«Entidade»
Plataforma
«Entidade»
Projetil
Atirar
Gerenciadores das Entidades
«Infraestrutura»
Animadora
«Infraestrutura»
Corpo Grafico
«Infraestrutura»
PThread
Possui
Abstrata
«Abstrata»
Entidade
LaganDino
Fabricas
«Fabrica»
FabricaMontanha
«Fabrica»
FabricaFloresta
«Abstrata»
FabricaFase
Instancia outros
Estados
«Abstrata»
Estado
«Estado»
MenuEstado
«Estado»
FaseEstado
Controladoras
«Auxiliar»
Fundo
«Singleton»
Jogo
«Abstrata»
Menu
«Infraestrutura»
GerenciadorGráfico
PilhaEstados
«Menu»
MenuPrincipal
«Menu»
MenuPause
Gerenciadores da Fase
«Gerenciador»
Gerenciador_Persistencias
«Gerenciador»
Gerenciador_Colisoes
Obstaculos
«Obstáculo»
Galho
«Obstáculo»
Pedra
«Obstáculo»
Espinho
«Entidade»
Obstaculo
Utiliza
Gerencia
Listas
«Lista»
ListaEntidades
Lista
Elemento
Conhecer Primeiro
Conhecer Ultimo
Conhecer Proximo
24/11/2022 // Mário Cordeiro Junior S73
// Diagrama UML Versão 6

Conceitos utilizados e não utilizados

N.	Conceitos	Uso	Onde / O quê
1	**Elementares:**		
1.1	- Classes, objetos. & - Atributos (privados), variáveis e constantes. & - Métodos (com e sem retorno).	Sim	Todos .h e .cpp.
1.2	- Métodos (com retorno *const* e parâmetro *const*). & - Construtores (sem/com parâmetros) e destrutores	Sim	Todos .h e .cpp.
1.3	- Classe Principal.	Sim	Main.cpp & Principal.h/.cpp.
1.4	- Divisão em .h e .cpp.	Sim	No desenvolvimento como um todo.
2	**Relações de:**		
2.1	- Associação direcional. & - Associação bidirecional.	Sim	No desenvolvimento como um todo.
2.2	- Agregação via associação. & - Agregação propriamente dita.	Sim	No desenvolvimento como um todo.
2.3	- Herança elementar. & - Herança em diversos níveis.	Sim	No desenvolvimento como um todo.
2.4	- Herança múltipla.	Não	AtiradinoThread.h.
3	**Ponteiros, generalizações e exceções**		
3.1	- Operador *this* para fins de relacionamento bidirecional.	Não	No desenvolvimento como um todo. Ex Corpo_Grafico.cpp.
3.2	- Alocação de memória (*new & delete*).		No desenvolvimento como um todo. Ex FabricaFase.cpp.
3.3	- Gabaritos/*Templates* criada/adaptados pelos autores (*e.g.*, Listas Encadeadas via *Templates*).		Lista.h.
3.4	- Uso de Tratamento de Exceções (*try catch*).		Gerenciador_Persistencias.cpp.
4	**Sobrecarga de:**		
4.1	- Construtoras e Métodos.		No desenvolvimento como um todo.
4.2	- Operadores (2 tipos de operadores pelo menos – Quais?).		Corpo_Grafico.h & PilhaEstados.h
---	**Persistência de Objetos (via arquivo de texto ou binário)**		
4.3	- Persistência de Objetos.		Gerenciador_Persistencias.cpp, FabricaFase.cpp.
4.4	- Persistência de Relacionamento de Objetos.		Gerenciador_Persistencias.cpp.
5	**Virtualidade:**		
5.1	- Métodos Virtuais Usuais.		No desenvolvimento do projeto como um todo.
5.2	- Polimorfismo.		No desenvolvimento do projeto como um todo.
5.3	- Métodos Virtuais Puros / Classes Abstratas.		No desenvolvimento do projeto como um todo.
5.4	- Coesão/Desacoplamento efetiva e intensa com o apoio de padrões de projeto.		No desenvolvimento do projeto como um todo.
6	**Organizadores e Estáticos**		
6.1	- Espaço de Nomes (*Namespace*) criada pelos autores.		Todos os .h e .cpp.
6.2	- Classes aninhadas (*Nested*) criada pelos autores.		Lista.h.
6.3	- Atributos estáticos e métodos estáticos.		PThread.h, Jogo.h, AtiradinoThread.h e Plataforma.h.
6.4	- Uso extensivo de constante (*const*) parâmetro, retorno, método...		No desenvolvimento do projeto como um todo.

conceitos utilizados e não utilizados

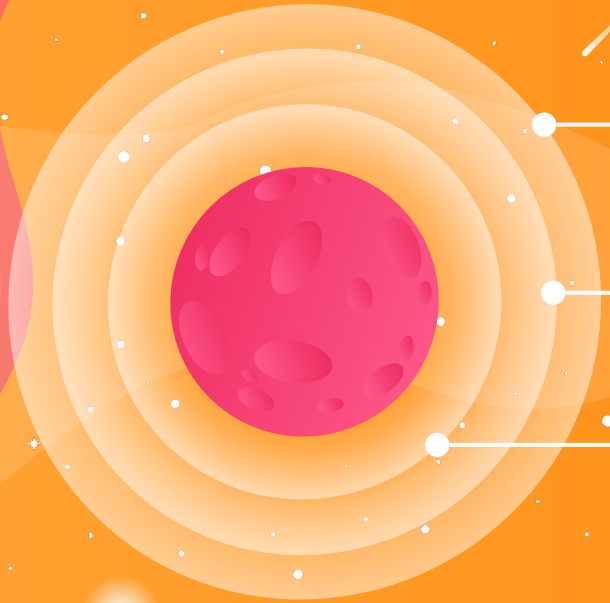
7	**Standard Template Library *(STL)* e String OO**			
7.1	- A classe Pré-definida *String* ou equivalente. **&** - *Vector* e/ou *List* da *STL* (p/ objetos ou ponteiros de objetos de classes definidos pelos autores)		No desenvolvimento do projeto como um todo. Ex Menu.h e Menu.cpp.	
7.2	- Pilha, Fila, Bifila, Fila de Prioridade, Conjunto, Multi-Conjunto, Mapa **OU** Multi-Mapa.		PilhaEstados.h e PilhaEstados.cpp	
---	**Programação concorrente**			
7.3	- *Threads* (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando Posix, C-Run-Time **OU** Win32API ou afins.		PThread.h, PThread.cpp, AtiradinoThread.h e AtiradinoThread.cpp.	
7.4	- *Threads* (Linhas de Execução) no âmbito da Orientação a Objetos com uso de Mutex, Semáforos, **OU** Troca de mensagens.		PThread.h, PThread.cpp, AtiradinoThread.h e AtiradinoThread.cpp.	
8	**Biblioteca Gráfica / Visual**			
8.1	- Funcionalidades Elementares. **&** - Funcionalidades Avançadas como: • tratamento de colisões • duplo *buffer*		Biblioteca SFML 2.5.1: Gerenciador_Grafico.h & Gerenciador_Grafico.cpp.	
8.2	- Programação orientada e evento efetiva (com gerenciador apropriado de eventos inclusive) em algum ambiente gráfico. **OU** - *RAD – Rapid Application Development* (Objetos gráficos como formulários, botões etc).		Gerenciador_Grafico.cpp, Menu.cpp, Vita.cpp, Tard.cpp, Jogo.cpp.	
---	**Interdisciplinaridades via utilização de Conceitos de Matemática Contínua e/ou Física.**			
8.3	- Ensino Médio Efetivamente.		Álgebra, Funções, Gravidade, Probabilidade, Geometria plana, Geometria Analítica, Movimento acelerado.	
8.4	- Ensino Superior Efetivamente.		Lógica, Estrutura de dados, Geometria Analítica, Cálculo 1, Velocidade Instantânea.	

9	**Engenharia de Software**		
9.1	- Compreensão, melhoria e rastreabilidade de cumprimento de requisitos. &		Diagrama de Classes como apoio, e impressão do modelo para trabalho com o fim de melhor atender os requisitos. Outrossim, foram marcadas reuniões com o monitor e o professor para sanar as eventuais dúvidas.
9.2	- Diagrama de Classes em *UML*.		Largamente utilizado na implementação do projeto, sendo constantemente atualizado tendo em sua totalidade 6 versões.
9.3	- Uso efetivo e intensivo de padrões de projeto *GOF*, *i.e.*, mais de 5 padrões.		Singleton, Composite, Factory Method, State.
9.4	- Testes à luz da Tabela de Requisitos e do Diagrama de Classes.		Foram utilizadas ferramentas para debugar e testar o código como gdb e valgrind, sempre visando atender os requisitos satisfatoriamente.
10	**Execução de Projeto**		
10.1	- Controle de versão de modelos e códigos automatizados (via github e/ou afins). & - Uso de alguma forma de cópia de segurança (*i.e., backup*).		Utilizando a ferramenta Git e uma de umas respectivas interfaces acompanhado de um repositório online chamado GitHub.
10.2	- Reuniões com o professor para acompanhamento do andamento do projeto.		Reuniões nos dias: 03/11, 10/11, 17/11, 24/11.
10.3	- Reuniões com monitor da disciplina para acompanhamento do andamento do projeto.		Reuniões online nos dias: 7/11, 14/11, 21/11
10.4	- Revisão do trabalho escrito de outra equipe e vice-versa.		Equipe formada pelos alunos Pedro Lucas e Ian Neves.

Conceitos apropriadamente utilizados: 100%

Implementação

conclusão



Grande
auto-aperfeiçoamento

Resultado de jogabilidade
satisfatório

Entendimento da
importância dos conceitos


OBRIGADO!
Perguntas?