Dino++

Guilherme Aguilar de Oliveira
Tiago Gonçalves da Silva

Requisitos e porcentagem cumprida

N.	Requisitos Funcionais	Porcentagem
1	Apresentar menu de opções aos usuários do Jogo.	100%
2	Permitir um ou dois jogadores aos usuários do Jogo, sendo que no último caso seria para que os dois joguem de maneira concomitante.	100%
3	Disponibilizar ao menos duas fases que podem ser jogadas sequencialmente ou selecionadas.	100%
4	Ter três tipos distintos de inimigos (o que pode incluir 'Chefão', vide abaixo), sendo que pelo menos um dos inimigos deve ser capaz de lançar projetil contra o(s) jogador(es).	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 5 instâncias por tipo.	100%
6	Ter inimigo "Chefão" na última fase	100%
7	Ter três tipos de obstáculos.	100%
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.	100%
9	Ter representação gráfica de cada instância.	100%
10	Ter em cada fase um cenário de jogo com os obstáculos.	100%
11	Gerenciar colisões entre jogador e inimigos, bem como seus projeteis (em havendo).	100%
12	Gerenciar colisões entre jogador e obstáculos.	100%
13	Permitir cadastrar/salvar dados do usuário, manter pontuação durante jogo, salvar pontuação e gerar lista de pontuação (*ranking*).	100%
14	Permitir Pausar o Jogo	100%
15	Permitir Salvar Jogada.	100%

Porcentagem geral: 100%

1. Apresentar menu de opções

Classes:

- Menu
- MenuPrincipal
- MenuPause

2. Permitir um ou dois jogadores

Classes:

- Jogador
- Guigo
- Titi

3. Duas Fases

Classes:

- Fase
- Montanha
- Floresta

4. Três inimigos

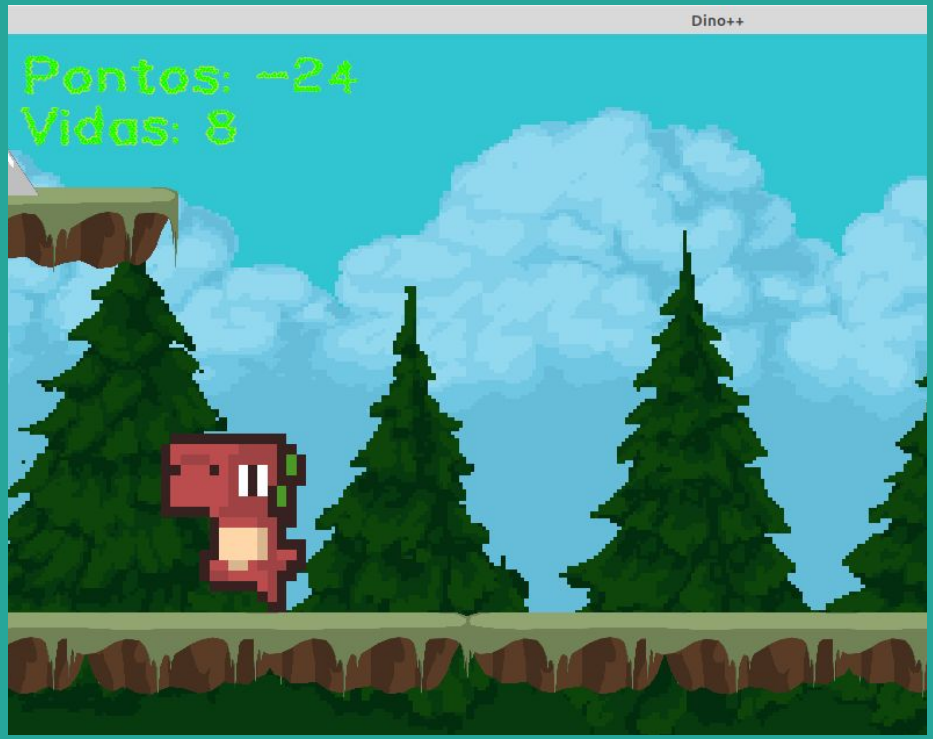
Classes:

- Inimigo
- Andino
- Atiradino
- AtiradinoThread
- ChefeDino

6. Chefão

Classes:

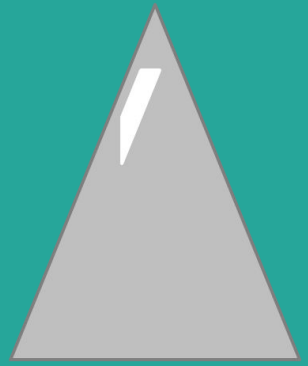
- Inimigo
- ChefeDino



7. Três Obstáculos

Classes:

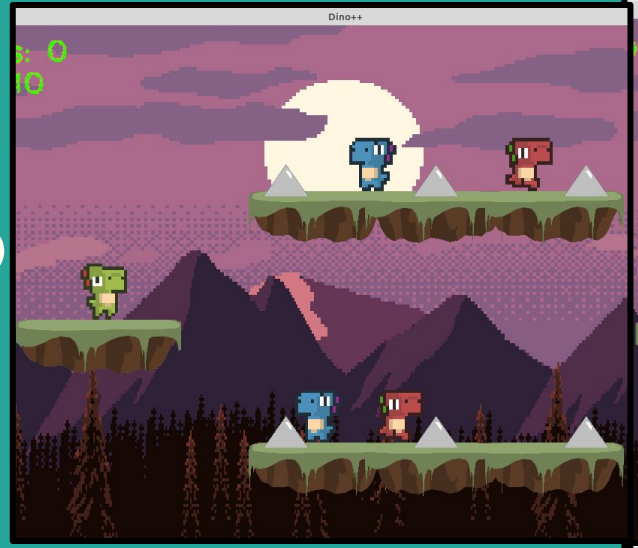
- Obstaculo
- Espinho
- Galho
- Pedra

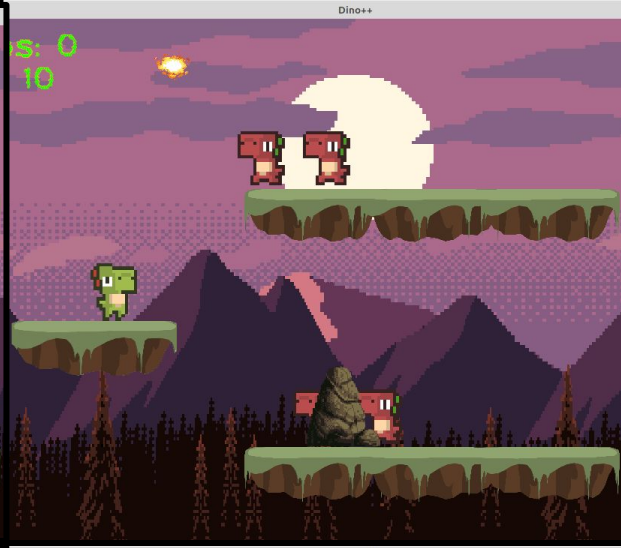


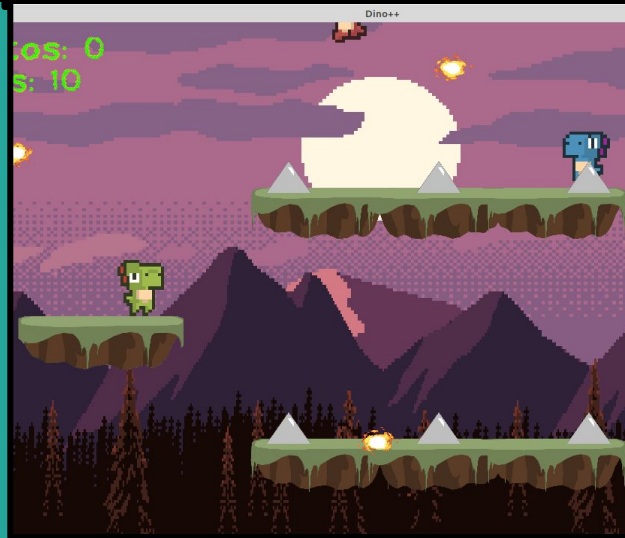
5, 8. Instâncias aleatórias e pelo menos 5

Classes:

- FabricaFase
- FabricaMontanha
- FabricaFloresta







Todos outros requisitos

N.	Conceitos	%	Onde / O quê
1	**Elementares:**		
	- Classes, objetos. & - Atributos (privados), variáveis e constantes. & - Métodos (com e sem retorno).	100	Todos .h e .cpp
	- Métodos (com retorno *const* e parâmetro *const*). & - Construtores (sem/com parâmetros) e destrutores	100	Todos .h e .cpp
	- Classe Principal.	100	Main.cpp & Jogo.h/.cpp
	- Divisão em .h e .cpp.	100	No desenvolvimento como um todo.
2	**Relações de:**		
	- Associação direcional. & - Associação bidirecional.	100	No desenvolvimento como um todo.
	- Agregação via associação. & - Agregação propriamente dita.	100	No desenvolvimento como um todo.
	- Herança elementar. & - Herança em diversos níveis.	100	No desenvolvimento como um todo.
	- Herança múltipla.	100	AtiradinoThread.h
3	**Ponteiros, generalizações e exceções**		
	- Operador *this*.	100	No desenvolvimento como um todo. Ex Corpo_Grafico
	- Alocação de memória (*new* & *delete*).	100	No desenvolvimento como um todo. Ex FabricaFase.cpp
	- Gabaritos/*Templates* criada/adaptados pelos autores (e.g. Listas Encadeadas via *Templates*).	100	Lista.h.
	- Uso de Tratamento de Exceções (*try catch*).	100	Persistidora.cpp.

N.	Conceitos	%	Onde / O quê
4	**Sobrecarga de:**		
	- Construtoras e Métodos.	100	No desenvolvimento do projeto como um todo.
	- Operadores (2 tipos de operadores pelo menos).	100	Corpo_Grafico.h & PilhaEstados.h.
	Persistência de Objetos (via arquivo de texto ou binário)		
	- Persistência de Objetos.	100	Persistidora.cpp, FabricaFase.cpp.
	- Persistência de Relacionamento de Objetos.	100	Persistidora.cpp.
5	**Virtualidade:**		
	- Métodos Virtuais.	100	No desenvolvimento do projeto como um todo.
	- Polimorfismo	100	No desenvolvimento do projeto como um todo.
	- Métodos Virtuais Puros / Classes Abstratas	100	No desenvolvimento do projeto como um todo.
	- Coesão e Desacoplamento	100	No desenvolvimento como um todo.

N.	Conceitos	%	Onde / O quê
6	**Organizadores e Estáticos**		
	- Espaço de Nomes (*Namespace*) criada pelos autores.	100	Todos os .h e .cpp
	- Classes aninhadas (*Nested*) criada pelos autores.	100	Lista.h.
	- Atributos estáticos e métodos estáticos.	100	PThread.h, Jogo.h, AtiradinoThread.h e Plataforma.h.
	- Uso extensivo de constante (*const*) parâmetro, retorno, método...	100	No desenvolvimento do projeto como um todo.
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)	100	No desenvolvimento do projeto como um todo. Menu.h, Menu.cpp.
	- Pilha, Fila, Bifila, Fila de Prioridade, Conjunto, Multi-Conjunto, Mapa **OU** Multi-Mapa.	100	PilhaEstados.h e PilhaEstados.cpp.
	Programação concorrente		
	- *Threads* (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando Posix, C-Run-Time **OU** Win32API ou afins.	100	PThread.h, PThread.cpp, AtiradinoThread.h e AtiradinoThread.cpp.
	- *Threads* (Linhas de Execução) no âmbito da Orientação a Objetos com uso de Mutex, Semáforos, **OU** Troca de mensagens.	100	AtiradinoThread.h, AtiradinoThread.cpp, & PThread.h.

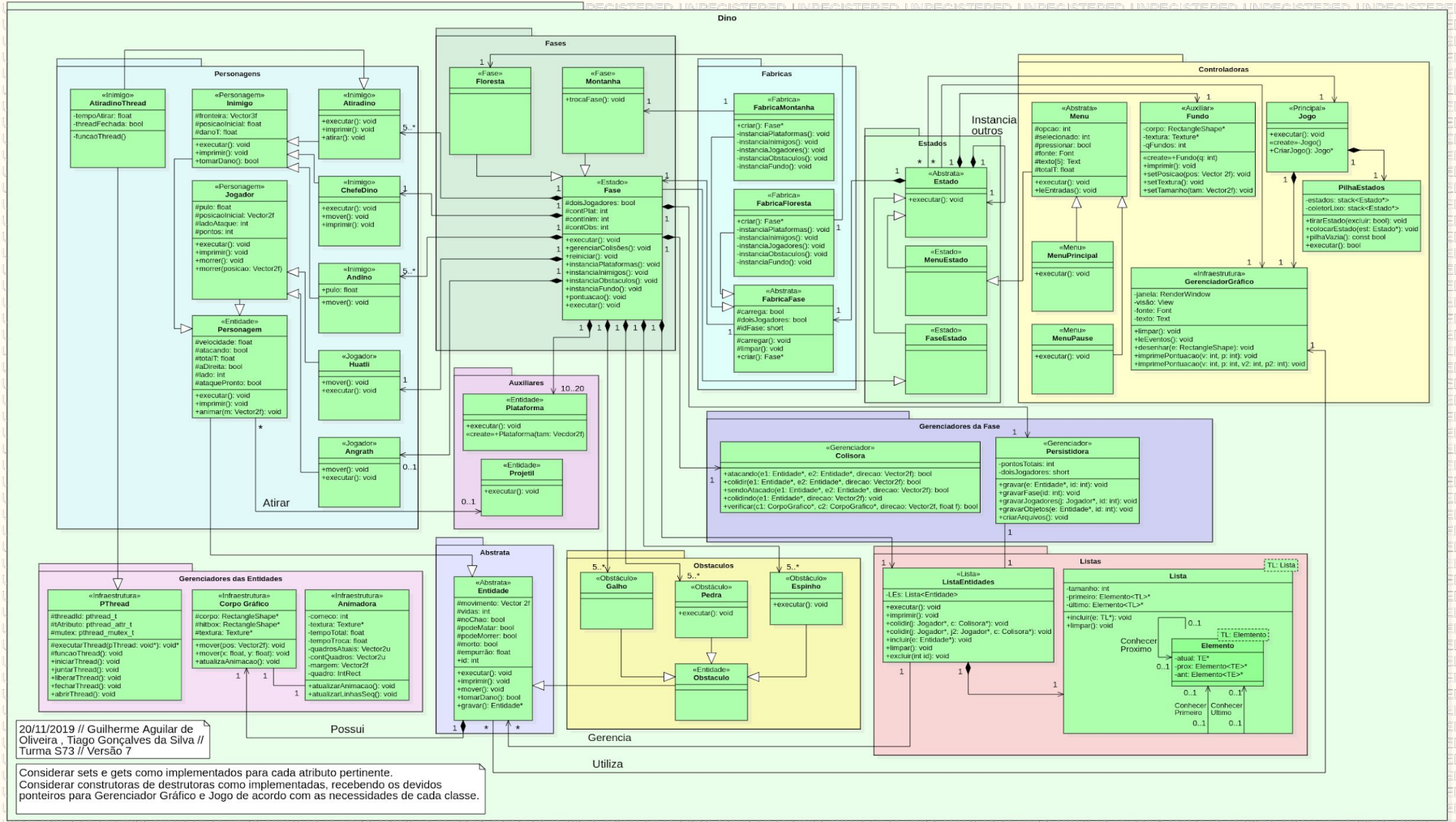
N.	Conceitos	%	Onde / O quê
8	**Biblioteca Gráfica / Visual**		
	- Funcionalidades Elementares. & - Funcionalidades Avançadas como: ● tratamento de colisões ● duplo *buffer*	100	Biblioteca SFML: Gerenciador_Grafico.h & Gerenciador_Grafico.cpp
	- Programação orientada e evento em algum ambiente gráfico. **OU** - *RAD – Rapid Application Development* (Objetos gráficos como formulários, botões etc).	100	Gerenciador_Grafico.cpp, Menu.cpp, Guigo.cpp, Titi.cpp & Jogo.cpp.
	Interdisciplinaridades por meio da utilização de Conceitos de Matemática e/ou Física.		
	- Ensino Médio.	100	Álgebra, Funções, Gravidade, Probabilidade, Geometria plana, Geometria Analítica, Movimento Acelerado.
	- Ensino Superior.	100	Lógica, Estrutura de dados, Geometria Analítica, Cálculo 1, Velocidade instantânea.

N.	Conceitos	%	Onde / O quê
9	**Engenharia de Software**		
	- Compreensão, melhoria e rastreabilidade de cumprimento de requisitos. &	100	Diagrama de Classes como apoio, e impressão do modelo para o trabalho com o fim de melhor atender os requisitos. Outrossim, foram marcadas reuniões com o monitor e o professor para sanar eventuais dúvidas.
	- Diagrama de Classes em *UML*.	100	Largamente utilizado na implementação do projeto, sendo constantemente atualizado tendo em sua totalidade sete versões.
	- Uso efetivo (quiçá) intensivo de padrões de projeto (particularmente GOF).	100	Singleton, Composite, Factory Method, State.
	- Testes a luz da Tabela de Requisitos e do Diagrama de Classes.	100	Foram utilizados ferramentas para debugar e testar o código como gdb, sempre visando atender os requisitos satisfatoriamente.

N.	Conceitos	%	Onde / O quê
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).	100	Utilizado a ferramenta Git e uma de suas respectivas interfaces acompanhado de um repositório online chamado GitHub.
	- Reuniões com o professor para acompanhamento do andamento do projeto.	100	Reuniões no LIC/CITEC dias 29/10, 5/11 e em sala de aula dias 12/11 e 13/11.
	- Reuniões com monitor da disciplina para acompanhamento do andamento do projeto.	100	Reuniões nos dias: 4/10, 8/10, 14/10, 16/10, 18/10, 21/10, 25/10, 13/11, 18/11 22/11.
	- Revisão do trabalho escrito de outra equipe e vice-versa.	100	Equipe formada pelos alunos Matheus dos Santos e Meika Farias.

Porcentagem geral: 100%

Implementação



20/11/2019 // Guilherme Aguilar de Oliveira , Tiago Gonçalves da Silva // Turma S73 // Versão 7

Considerar sets e gets como implementados para cada atributo pertinente.
Considerar construtoras de destrutoras como implementadas, recebendo os devidos ponteiros para Gerenciador Gráfico e Jogo de acordo com as necessidades de cada classe.

Implementação

- Polimorfismo
- Colisões
- Classes de controle
- Threads
- Persistência

Conclusão

- Resultado satisfatório
- Grande auto-aperfeiçoamento
- Entendimento da importância de conceitos