LEMURYA

Trabalho acadêmico da disciplina de técnicas de programação.

Gabriel Henrique Linke
Pedro Sodré dos Santos

TABELA DE REQUISITOS

- 100% cumpridos

N.	Requisitos Funcionais	Situação	Implementação
1	Apresentar menu de opções aos usuários do Jogo.	Requisito previsto e inicialmente realizado.	Requisito cumprido via classe Menu (abstrata) e os respectivos objetos de suas subclasses (Menu Principal, Menu Pause e Menu Morte).
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 e inicialmente realizado.	Requisito cumprido inclusive via classe Player cujos objetos são agregados em jogo, podendo ser um ou dois efetivamente.
3	Disponibilizar ao menos duas fases que podem ser jogadas sequencialmente ou selecionadas.	Requisito previsto e inicialmente realizado.	Requisito cumprido, inclusive com 3 fases, sendo através de objetos das classes (FaseAquatica1, FaseAquatica2 e FaseNoturna3).
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).	Requisito previsto e inicialmente realizado.	Requisito cumprido, criando objetos das classes, Tritão, Esqueleto e Mago, sendo este último, o chefão capaz de lançar bolas de fogo (projétil).

TABELA DE REQUISITOS

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.	Requisito cumprido inclusive via método gerarInimigos, definido na classe abstrata Fase e redefinido nas suas classes derivadas.
6	Ter inimigo "Chefão" na última fase	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via objeto da classe Mago que é instanciado na última fase (FaseNoturna3).
7	Ter três tipos de obstáculos.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via classes Tronco, Pedra e Caixa cujos objetos são agregados na fase através da classe LemuryaPrototype.
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.	Requisito cumprido inclusive via método gerarObstaculos definido na classe Fase e redefinido nas suas classes derivadas.
9	Ter representação gráfica de cada instância.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive com a utilização da biblioteca gráfica SFML e via classe GerenciadorGrafico.
10	Ter em cada fase um cenário de jogo com os obstáculos.	Requisito previsto inicialmente e realizado.	Requisito cumprido via classe abstrata Fase e suas derivadas, com seus obstáculos sendo agregados através dos métodos gerarObstaculos, novoJogo e recuperarJogo.

TABELA DE REQUISITOS

11	Gerenciar colisões entre jogador e inimigos, bem como seus projeteis (em havendo).	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via classes Collider, que verifica a colisão, e o Gerenciador de Colisões, que verifica as colisões percorrendo todas as listas de entidades que são colidíveis.
12	Gerenciar colisões entre jogador e obstáculos.	Requisito previsto inicialmente e realizado.	Idem para o requisito acima.
13	Permitir cadastrar/salvar dados do usuário, manter pontuação durante jogo, salvar pontuação e gerar lista de pontuação (ranking).	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via classes GerenciadorDePontuação, atributo ranking do jogador, método do menu principal que gera a lista de pontuação.
14	Permitir Pausar o Jogo	Requisito previsto inicialmente e realizado	Requisito cumprido inclusive via classe MenuPause, que é instanciada caso o usuário pressione a tecla ESC.
15	Permitir Salvar Jogada.	Requisito previsto e realizado	Requisito realizado via classe MenuPause, e método checkSalvarJogo das classes derivadas de Fase, que permite salvar a jogada.

DIAGRAMA DE CLASSES UML

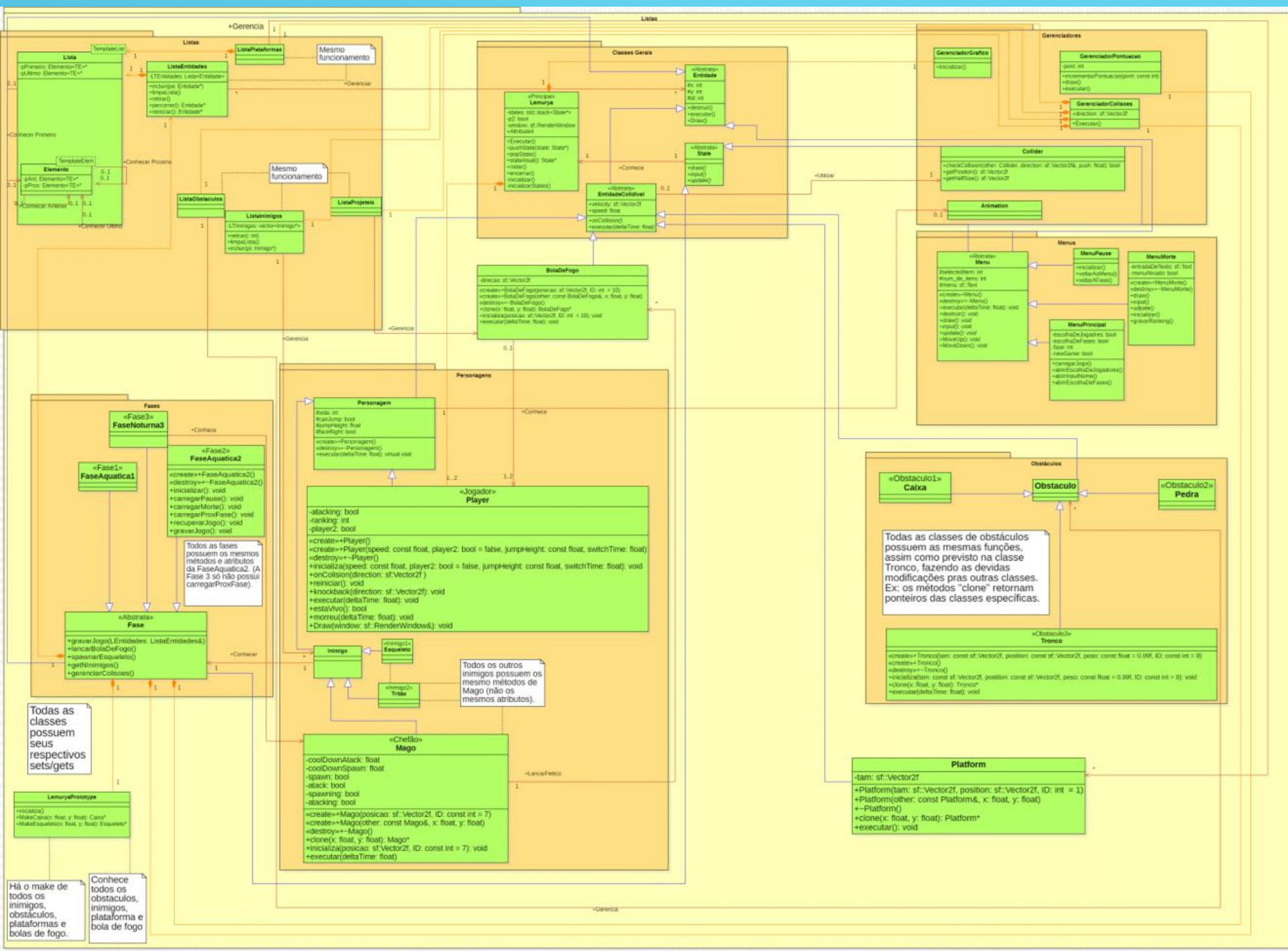

Listas
+Gerencia
Lista
ListaEntidades
ListaPlataformas
Mesmo funcionamento
ListaObstaculos
ListaInimigos
ListaProjeteis
Classes Gerais
Lemurya
Entidade
State
EntidadeCalidivel
Gerenciadores
GerenciadorGrafico
GerenciadorPontuacao
GerenciadorColisoes
Collider
Animation
Menus
Menu
MenuPause
MenuMorte
MenuPrincipal
BolaDeFogo
Personagens
Personagem
Player
Fases
FaseNoturna3
FaseAquatica2
FaseAquatica1
Fase
Inimigo
Esqueleto
Tritão
Todos os outros inimigos possuem os mesmo métodos de Mago (não os mesmos atributos).
Todos as fases possuem os mesmos métodos e atributos da FaseAquatica2. (A Fase 3 só não possui carregarProxFase).
Obstáculos
Obstaculo
Caixa
Pedra
Tronco
Todas as classes de obstáculos possuem as mesmas funções, assim como previsto na classe Tronco, fazendo as devidas modificações pras outras classes. Ex: os métodos "clone" retornam ponteiros das classes específicas.
Mago
Platform
Todas as classes possuem seus respectivos sets/gets
LemuryaPrototype
Há o make de todos os inimigos, obstáculos, plataformas e bolas de fogo.
Conhece todos os obstaculos, inimigos, plataforma e bola de fogo

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

- 89% dos conceitos foram utilizados

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 & Lemurya.h/.cpp
	- Divisão em .h e .cpp.	Sim	No desenvolvimento como um todo.

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

2	**Relações de:**		
	- Associação direcional. & - Associação bidirecional.	Sim	Bidirecional entre FaseNoturna3 e Mago Direcional entre BolaDeFogo e Player.
	- Agregação via associação. & - Agregação propriamente dita.	Sim	Agregação via associação entre Lista de Entidades e Lista. Agregação propriamente dita entre GerenciadorDeColisoes e ListaPlataformas, por exemplo.
	- Herança elementar. & - Herança em diversos níveis.	Sim	Elementar em EntidadeColidivel e Entidade. Em diversos níveis entre Mago e Entidade, por exemplo.
	- Herança múltipla.	Sim	Fase herdando de Entidade e State, por exemplo.

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

3	**Ponteiros, generalizações e exceções**		
	- Operador *this*/	Sim	Em diversos momentos. Ao inicializar o prototype dentro de fase, por exemplo.
	- Alocação de memória (*new* & *delete*).	Sim	Em diversos momentos. Delete em destrutoras e new nos métodos de carregar um novo estado do jogo, como no carregarPause de fase.
	- Gabaritos/*Templates* criada/adaptados pelos autores (e.g. Listas Encadeadas via *Templates*).	Sim	Classe List.
	- Uso de Tratamento de Exceções (*try catch*).	Sim	Classe GerenciadorGrafico.

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

4	**Sobrecarga de:**		
	- Construtoras e Métodos.	Sim	Construtoras nos objetos que podem ser clonados. Métodos em FaseAquatica1, por exemplo.
	- Operadores (2 tipos de operadores pelo menos).	Não	-
	Persistência de Objetos (via arquivo de texto ou binário)		
	- Persistência de Objetos.	Sim	Via métodos de gravarJogo e recuperarJogo das fases.
	- Persistência de Relacionamento de Objetos.	Sim	Persistência do relacionamento dos inimigos com os players, por exemplo

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

5	**Virtualidade:**		
	- Métodos Virtuais.	Sim	Na classe Fase, por exemplo
	- Polimorfismo	Sim	Chamadas de update e Draw da classe Entidades, por exemplo.
	- Métodos Virtuais Puros / Classes Abstratas	Sim	Classes menu e fase, por exemplo.
	- Coesão e Desacoplamento	Sim	No projeto como um todo

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

6	**Organizadores e Estáticos**		
	- Espaço de Nomes (*Namespace*) criada pelos autores.	Não	-
	- Classes aninhadas (*Nested*) criada pelos autores.	Sim	Classes List e Entidade
	- Atributos estáticos e métodos estáticos.	Parcialmente	Atributo estático VIEW_HEIGHT em Lemurya
	- Uso extensivo de constante (*const*) parâmetro, retorno, método...	Sim	No projeto como um todo

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

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)	Sim	Em GerenciadorDePontuacao e ListaObstaculos, por exemplo.
	- Pilha, Fila, Bifila, Fila de Prioridade, Conjunto, Multi-Conjunto, Mapa **OU** Multi-Mapa.	Sim	Em Lemurya (Pilha) e setRanking em MenuPrincipal (Multimap).
	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	-

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

8	**Biblioteca Gráfica / Visual**		
	- Funcionalidades Elementares. & - Funcionalidades Avançadas como: • tratamento de colisões • duplo *buffer*	Sim	Representações gráficas e escrever texto através da biblioteca SFML.
	- Programação orientada e evento em algum ambiente gráfico. **OU** - *RAD – Rapid Application Development* (Objetos gráficos como formulários, botões etc).	Sim	Utilização dos Eventos da biblioteca SFML na implementação dos menus.
	Interdisciplinaridades por meio da utilização de Conceitos de Matemática e/ou Física.		
	- Ensino Médio.	Sim	Conceitos de matemática e física básica, como velocidade, por exemplo.
	- Ensino Superior.	Sim	Norma de vetores, por exemplo.

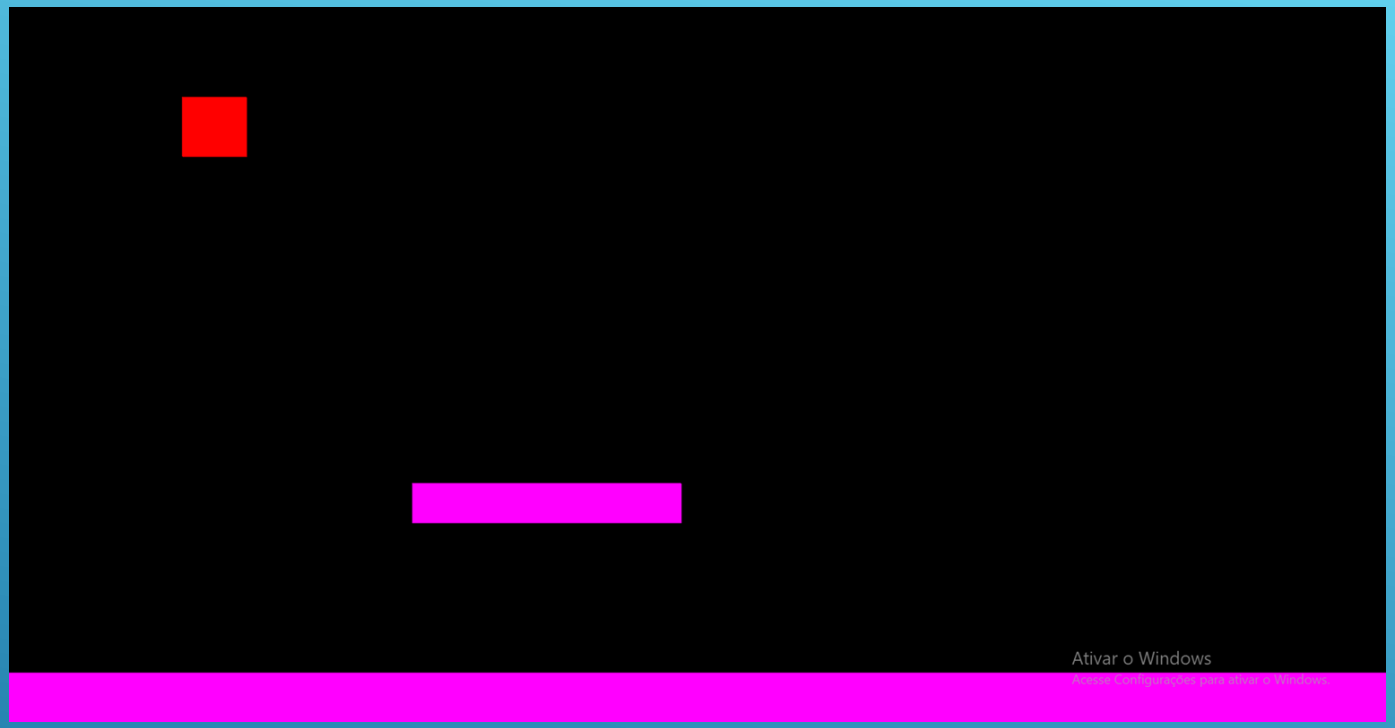
TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

9	**Engenharia de *Software***		
	- Compreensão, melhoria e rastreabilidade de cumprimento de requisitos. &	Sim	Durante reuniões com o professor e monitor.
	- Diagrama de Classes em *UML*.	Sim	No desenvolvimento como um todo.
	- Uso efetivo (quiçá) intensivo de padrões de projeto (particularmente GOF).	Sim	Classes LemuryaPrototype(Prototype)State(State) MenuPrincipal(Iterator)
	- Testes a luz da Tabela de Requisitos e do Diagrama de Classes.	Sim	Durante reuniões com o professor e monitor.

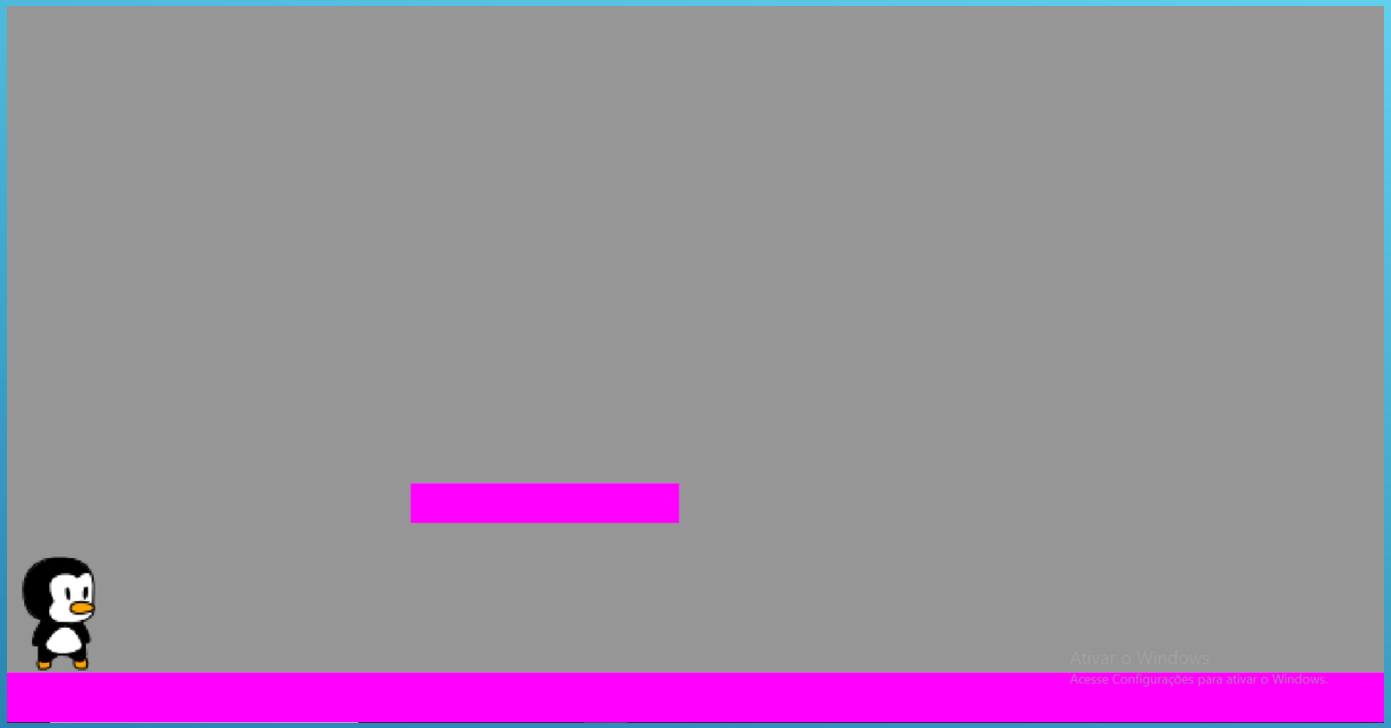
TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

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 GitHub e cópias manuais
	- Reuniões com o professor para acompanhamento do andamento do projeto.	Sim	3 Reuniões (25/10, 12/11, 22/11).
	- Reuniões com monitor da disciplina para acompanhamento do andamento do projeto.	Sim	8 Reuniões (14/10, 16/10, 18/10, 25/10, 6/11, 13/11, 18/11, 20/11).
	- Revisão do trabalho escrito de outra equipe e vice-versa.	Sim	Gustavo Brunholi Chierici e Jhonny Kristyan

DESENVOLVIMENTO DO JOGO



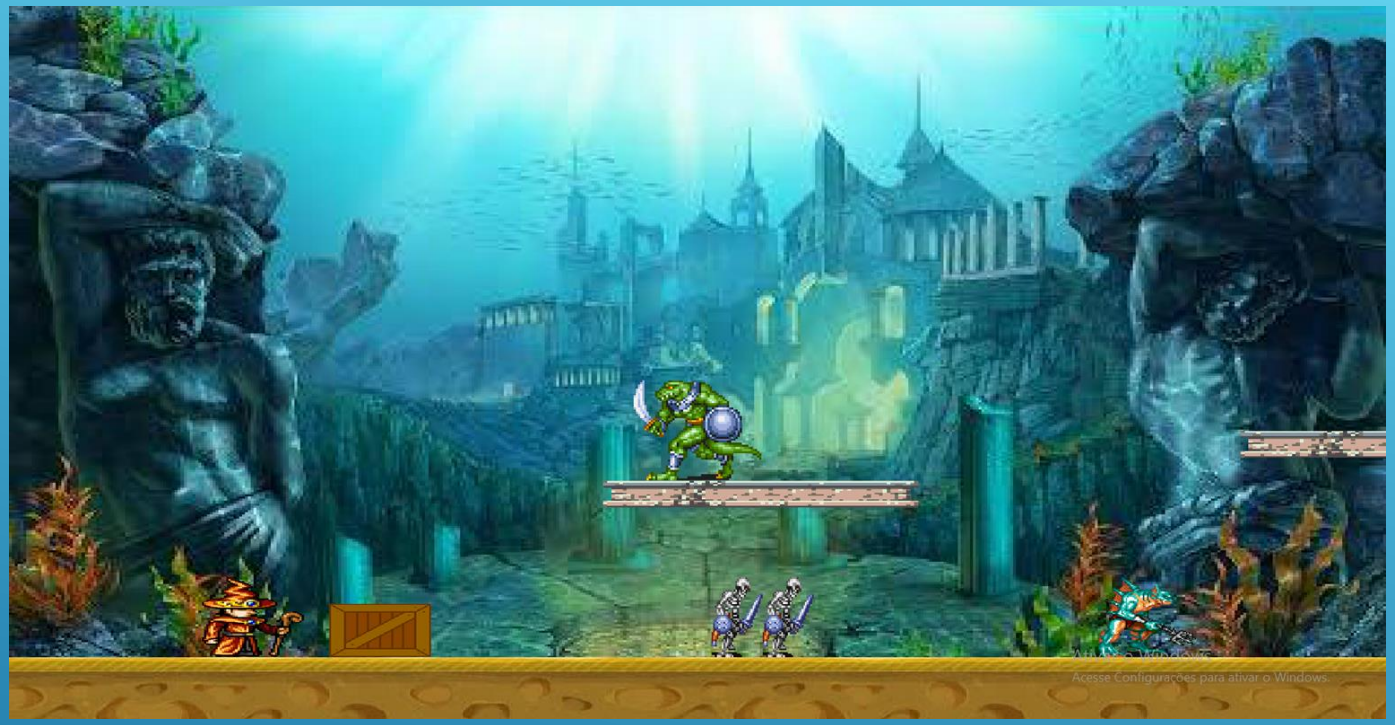
INÍCIO DO JOGO



COLOCADO PERSONAGEM NA TELA

ADICIONADO MAPA E BACKGROUND

ADICIONADOS OBSTÁCULOS



ADICIONADOS INIMIGOS

ADICIONADO O PRIMEIRO MENU

ADICIONADO PROJÉTIL

ADICIONADA PONTUAÇÃO

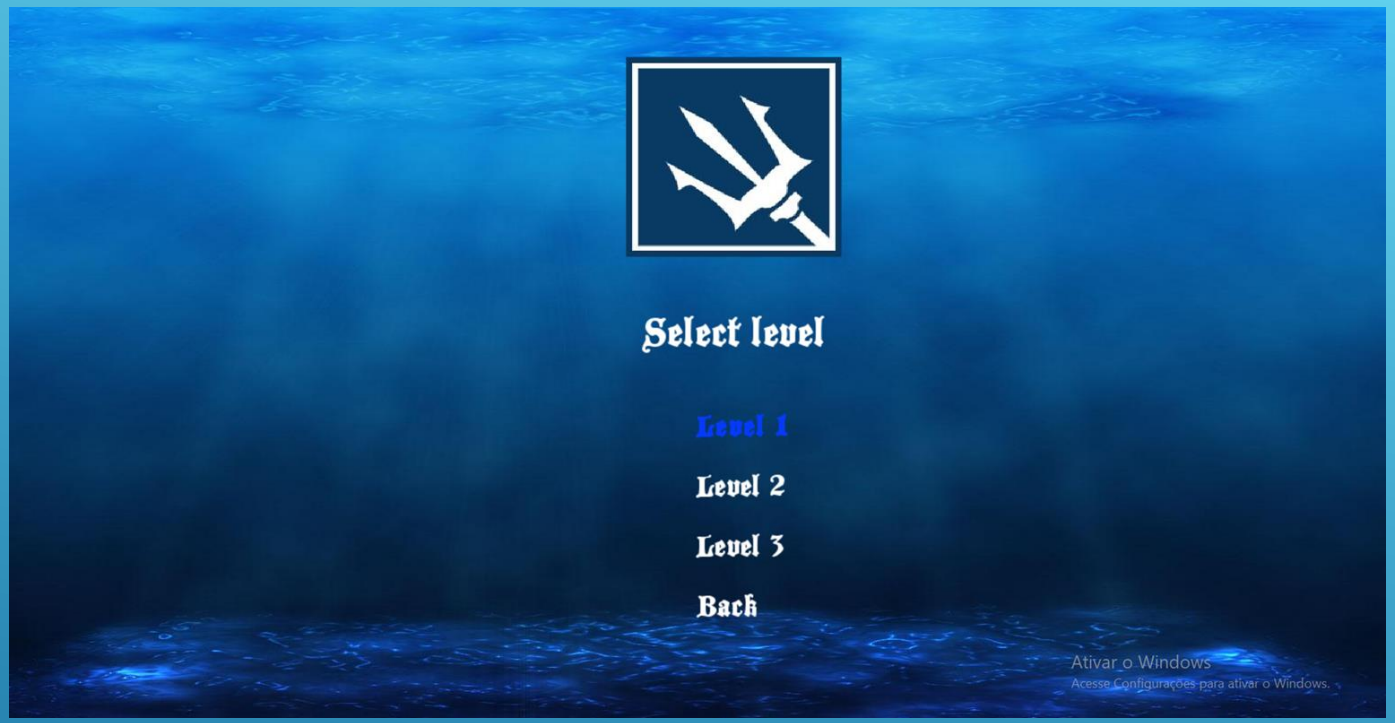
RESULTADO FINAL

MENU PRINCIPAL

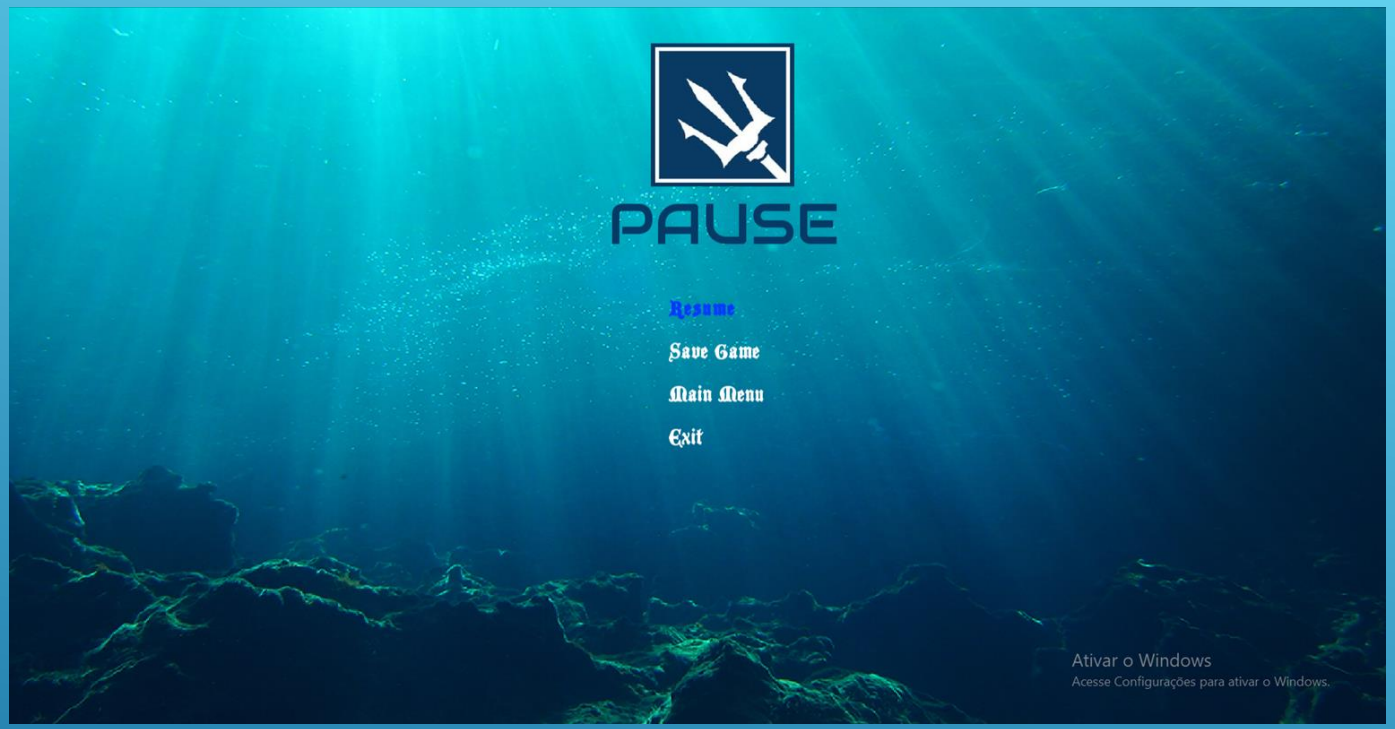

RANKING
Linke 610
Sodrico 510
Pedro 430
Linke 400
e 290
BACK
Ativar o Windows
Acesse Configurações para ativar o Windows.

RANKING

ESCOLHA DE PLAYERS



ESCOLHA DA FASE



MENU DE PAUSE

FASE 1


Score: 40
Ativar o Windows
Acesse Configurações para ativar o Windows.

FASE 2

FASE 3

MENU DE MORTE

MENU DE VITÓRIA

CONCLUSÃO