Trabalho de Técnicas de Programação – CSE20/S71

Planejamento e implementação do jogo Green Threat

Prof. Dr. Jean M. Simão
Alunos: Pedro Henrique Secchi e Pedro Scarpin Ribeiro

Requisitos

N	Requisito	Situação
1	Apresentar graficamente menu de opções aos usuários do Jogo, no qual pode se escolher fases, ver colocação (ranking) de jogadores e demais opções pertinentes.	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.	50%
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.	50%
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 projetil contra o(s) jogador(es) e um dos inimigos dever ser um ‘Chefão’.	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.	100%

Requisitos

N	Requisito	Situação
6	Ter três tipos de obstáculos, cada qual com sua representação gráfica, sendo que ao menos causa dano em jogador se colidirem.	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.	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.	100%
9	Gerenciar colisões entre jogador para com inimigos e seus projeteis, bem como entre jogador para com obstáculos.	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.	50%

Diagrama de Classes

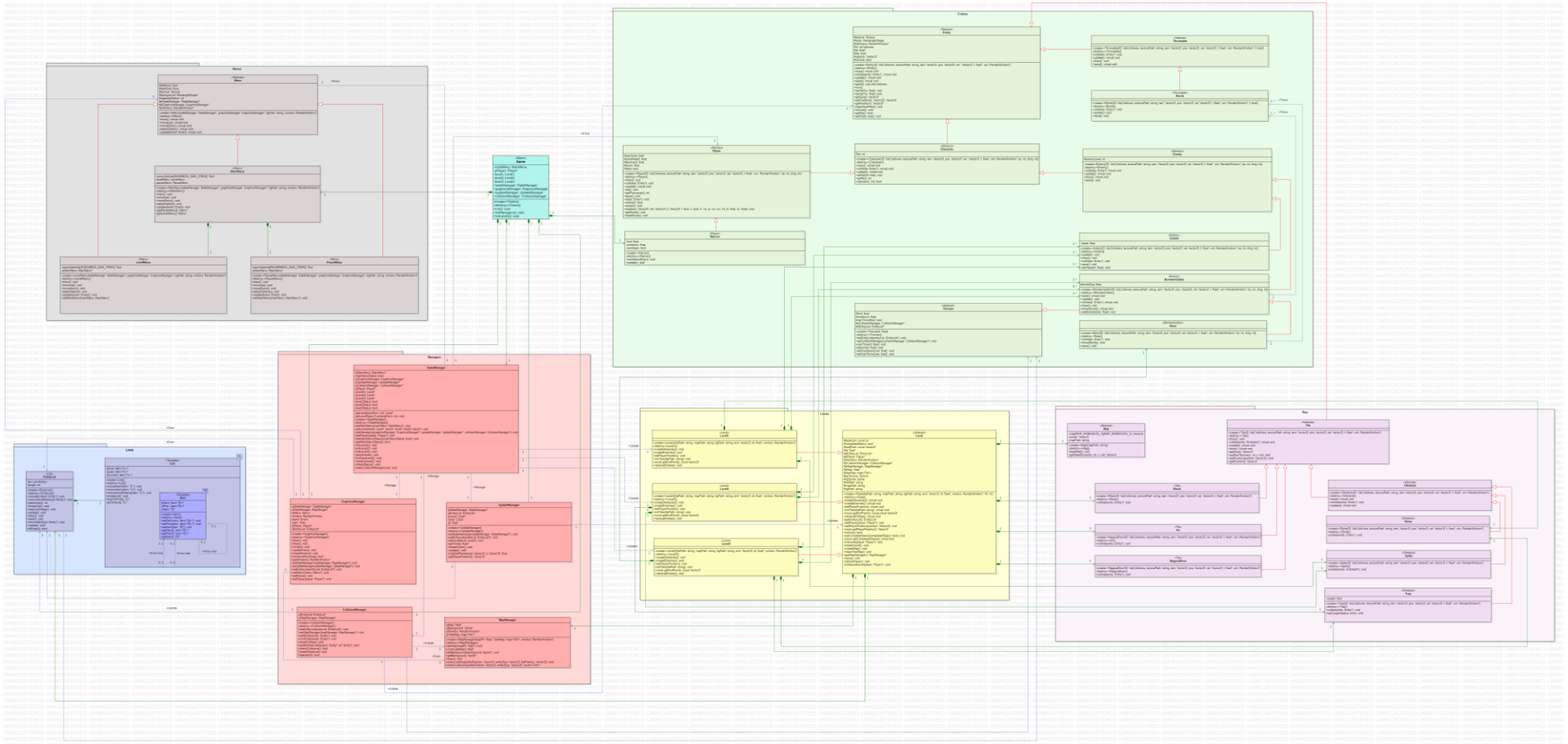


Diagrama de Classes

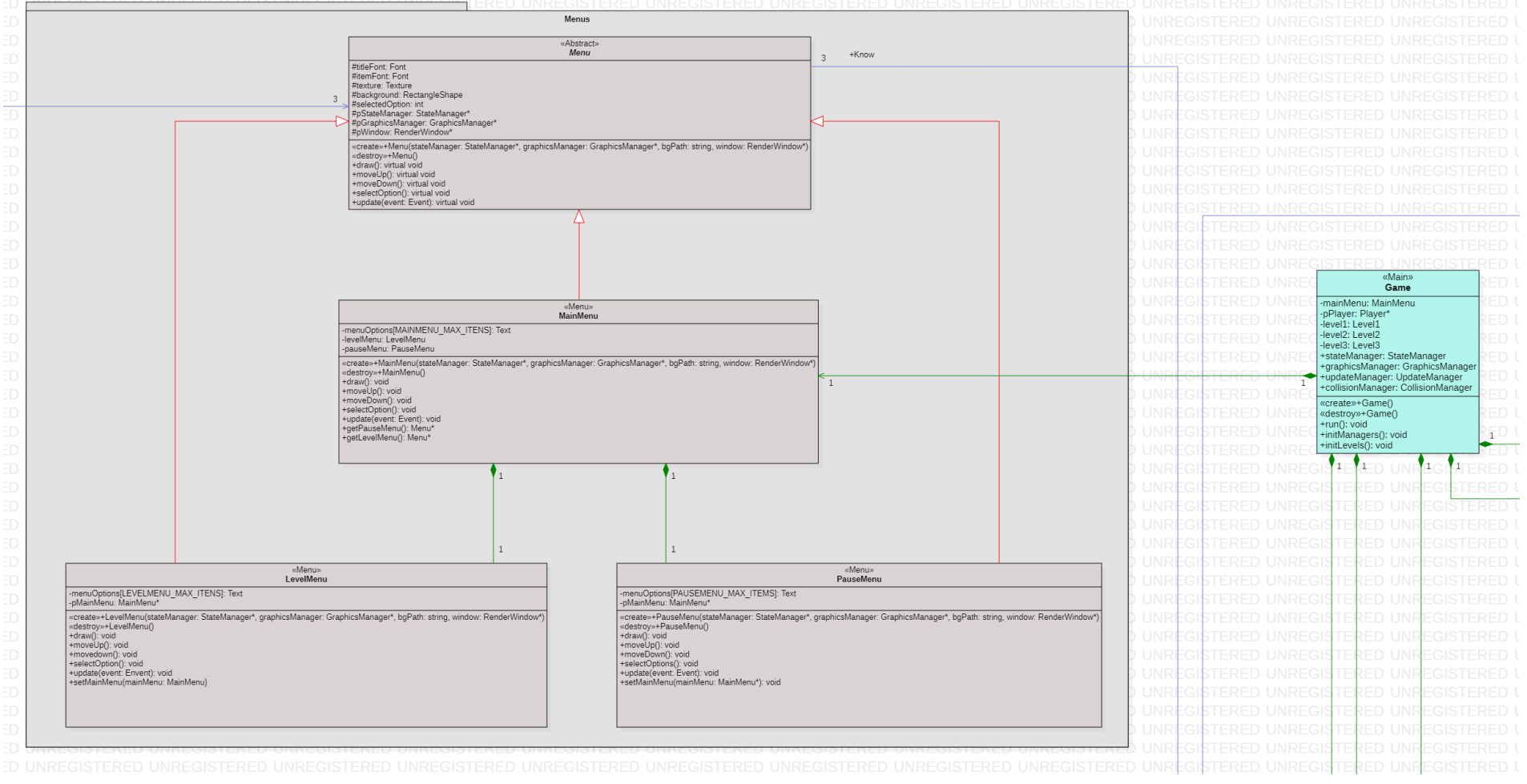


Diagrama de Classes

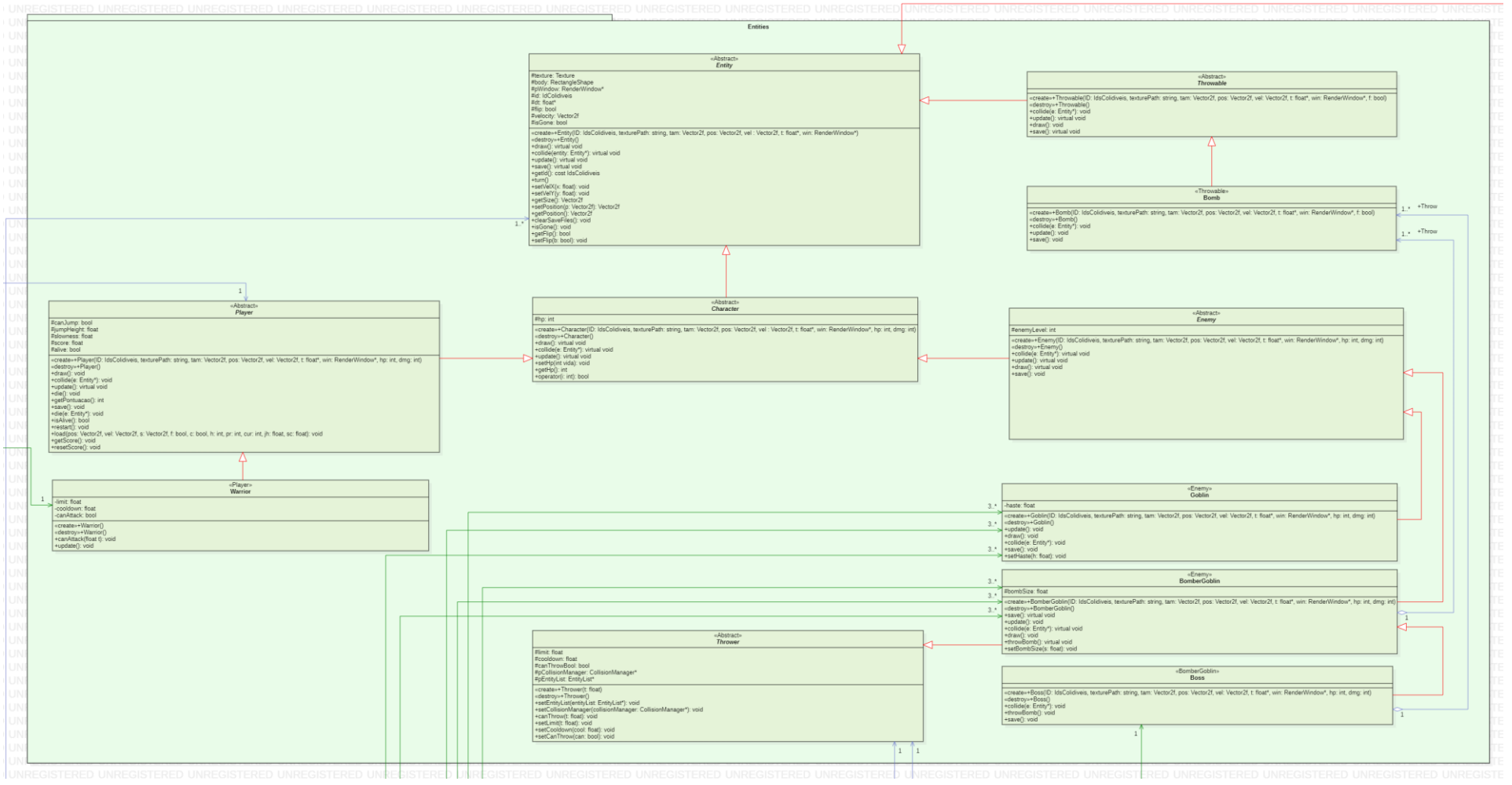
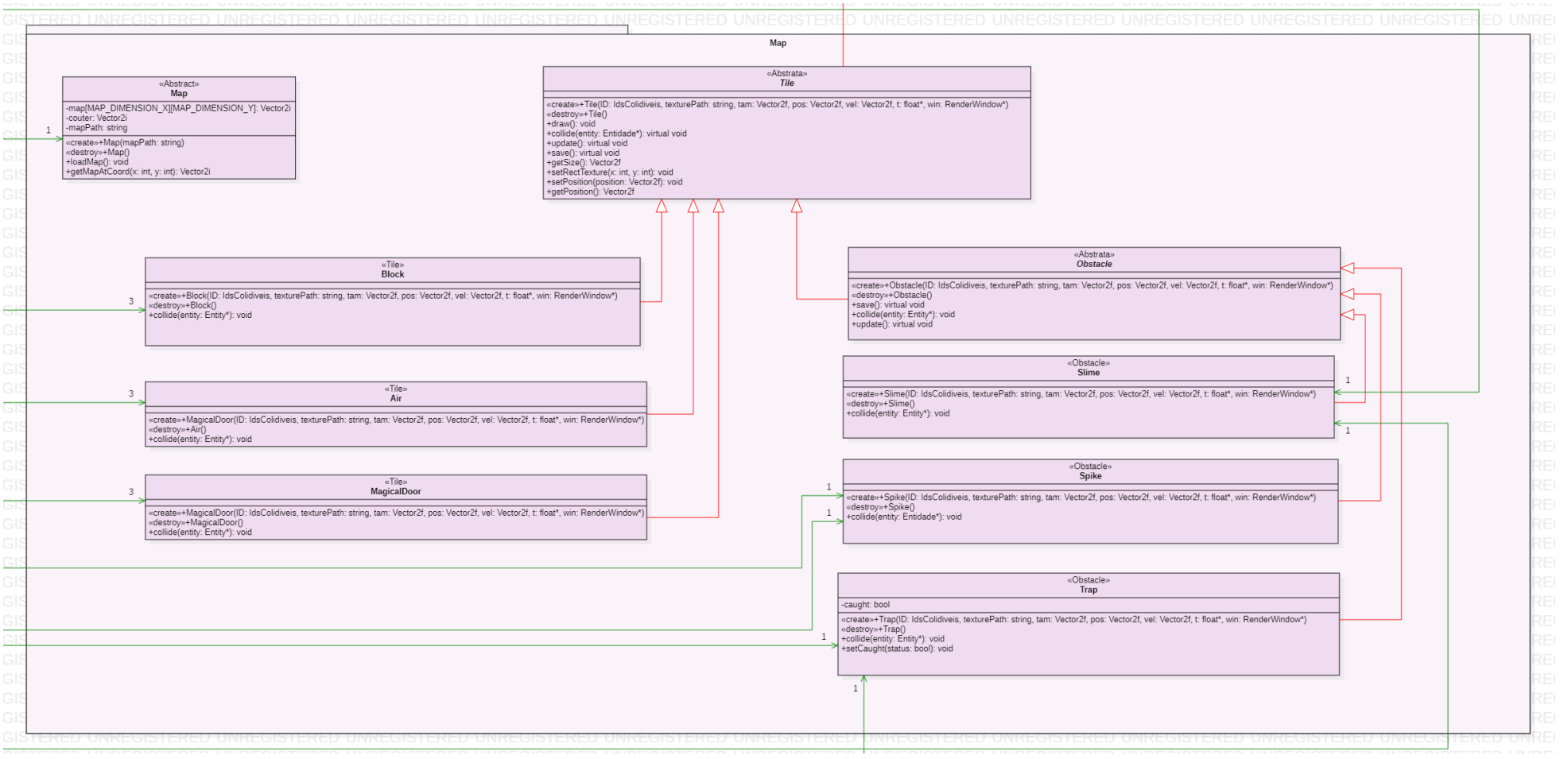


Diagrama de Classes



Conceitos Utilizados

N	Conceito	Utilizado
1	**Elementares: 100% concluído**	
	- Classes, objetos. & - Atributos (privados), variáveis e constantes. & - Métodos (com e sem retorno).	Sim
	- Métodos (com retorno const e parâmetro const). & - Construtores (sem/com parâmetros) e destrutores	Sim
	- Classe Principal.	Sim
	- Divisão em .h e .cpp.	Sim
2	**Relações: 100% concluído**	
	- Associação direcional. & - Associação bidirecional.	Sim
	- Agregação via associação. & - Agregação propriamente dita.	Sim
	- Herança elementar. & - Herança em diversos níveis.	Sim
	- Herança múltipla.	Sim

Conceitos Utilizados

N	Conceito	Utilizado
3	**Ponteiros, generalizações e exceções : 100% concluído**	
	- Operador this para fins de relacionamento bidirecional.	Sim
	- Alocação de memória (new & delete).	Sim
	- Gabaritos/Templates criada/adaptados pelos autores (e.g. Listas Encadeadas via Templates).	Sim
	- Uso de Tratamento de Exceções (try catch).	Sim
4	**Sobrecarga: 100% concluído**	
	- Construtoras e Métodos.	Sim
	- Operadores (2 tipos de operadores pelo menos).	Sim
	Persistência de Objetos (via arquivo de texto ou binário)	
	- Persistência de Objetos.	Sim
	- Persistência de Relacionamento de Objetos.	Sim
5	**Virtualidade: 100% concluído**	
	- Métodos Virtuais.	Sim
	- Polimorfismo	Sim

Conceitos Utilizados

N	Conceito	Utilizado
5	**Virtualidade: 100% concluído**	
	- Métodos Virtuais Puros / Classes Abstratas	Sim
	- Coesão e Desacoplamento	Sim
6	**Organizadores e Estáticos: 75% concluído**	
	- Espaço de Nomes (Namespace) criada pelos autores.	Sim
	- Classes aninhadas (Nested) criada pelos autores.	Sim
	- Atributos estáticos e métodos estáticos.	Não
	- Uso extensivo de constante (const) parâmetro, retorno, método…	Sim
7	**Standard Template Library (STL) e String OO: 50% concluído**	
	- 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
	- Pilha, Fila, Bifila, Fila de Prioridade, Conjunto, Multi-Conjunto, Mapa OU Multi-Mapa.	Sim
	Programação concorrente	
	-Threads (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando Posix, C-Run-Time OU Win32API ou afins	Não

Conceitos Utilizados

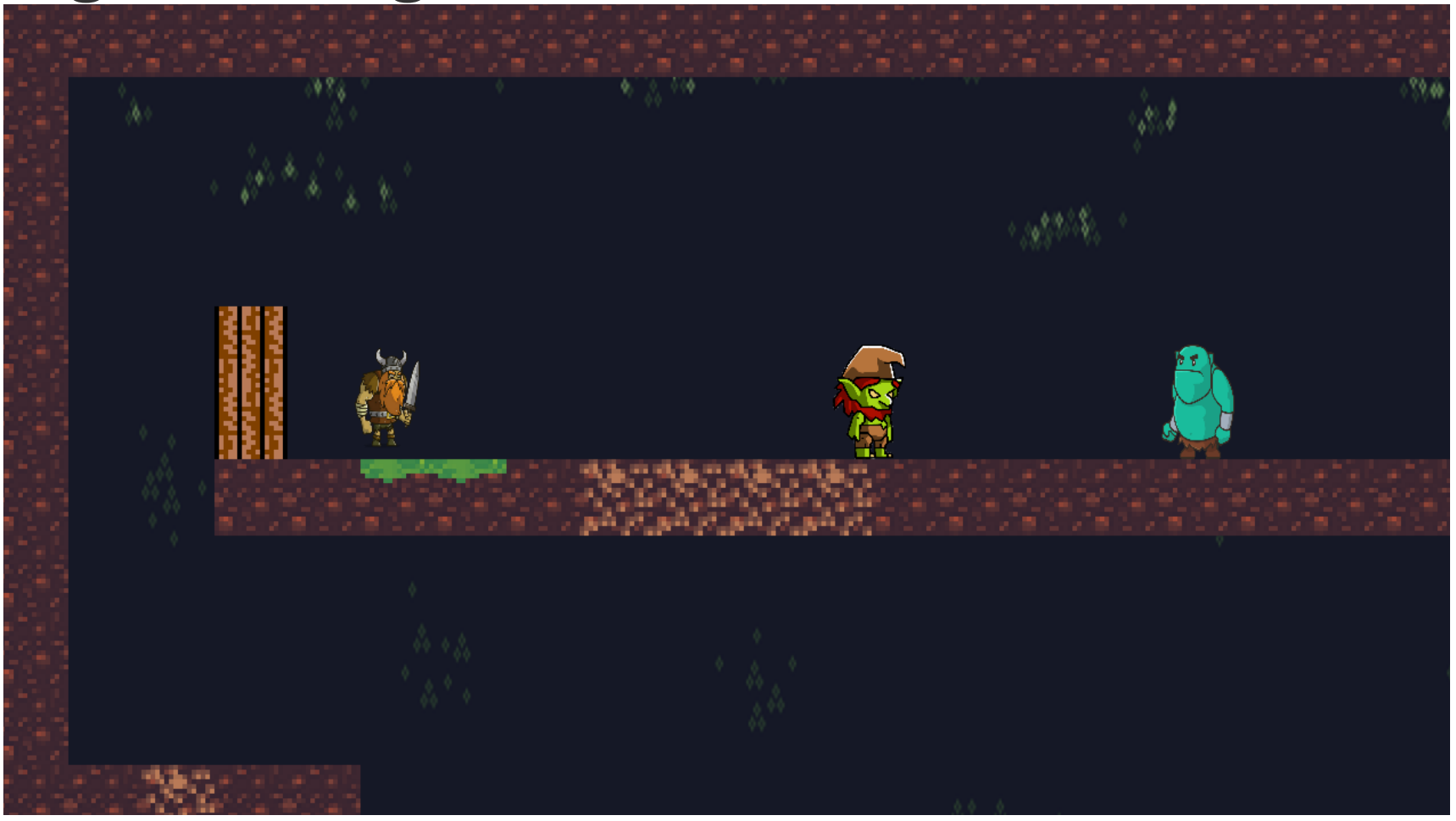
N	Conceito	Utilizado
7	**Standard Template Library (STL) e String OO : 50% concluído**	
	Programação concorrente	
	-Threads (Linhas de Execução) no âmbito da Orientação a Objetos com uso de Mutex, Semáforos, OU Troca de mensagens.	Não
8	**Biblioteca Gráfica / Visual: 100% concluído**	
	- Funcionalidades Elementares. & - Funcionalidades Avançadas como: -tratamento de colisões -duplo buffer	Sim
	- Programação orientada e evento em algum ambiente gráfico. OU - RAD – Rapid Application Development (Objetos gráficos como formulários, botões etc).	Sim
	Interdisciplinaridades via utilização de Conceitos de Matemática Contínua e/ou Física.	
	- Ensino Médio.	Sim
	- Ensino Superior.	Sim

Conceitos Utilizados

N	Conceito	Utilizado
9	**Engenharia de Software: 75% concluído**	
	- Compreensão, melhoria e rastreabilidade de cumprimento de requisitos. &	Sim
	- Diagrama de Classes em UML.	Sim
	- Uso efetivo e intensivo de padrões de projeto GOF.	Não
	- Testes à luz da Tabela de Requisitos e do Diagrama de Classes.	Sim
10	**Execução de Projeto: 50% concluído**	
	- Controle de versão de modelos e códigos automatizados (via SVN e/ou afins). & - Uso de alguma forma de cópia de segurança (backup).	Sim
	- Reuniões com o professor para acompanhamento do andamento do projeto.	Não
	Interdisciplinaridades via utilização de Conceitos de Matemática Contínua e/ou Física.	
	- Reuniões com monitor da disciplina para acompanhamento do andamento do projeto.	Não
	- Revisão do trabalho escrito de outra equipe e vice-versa.	Sim
Total dos conceitos utilizados		**85%**

Imagens do Jogo

Imagens do Jogo



Conclusão

A realização do projeto possibilitou experimentar um ambiente de produção que não só contou com o docente no papel do gerente de projeto, mas também na posição de um cliente instituindo os requisitos.

Além disso, ficou evidente grandes diferenças entre a programação orientada a objetos e a programação estruturada. O fato de se basear em uma composição de objetos instanciados de classes tornam os softwares que utilizam o paradigma POO mais fáceis de serem entendidos e implementados.

Apesar de o resultado não ser o almejado inicialmente, a demanda foi concluída de forma satisfatória, tendo em vista que a maior parte dos requisitos foram realizados.