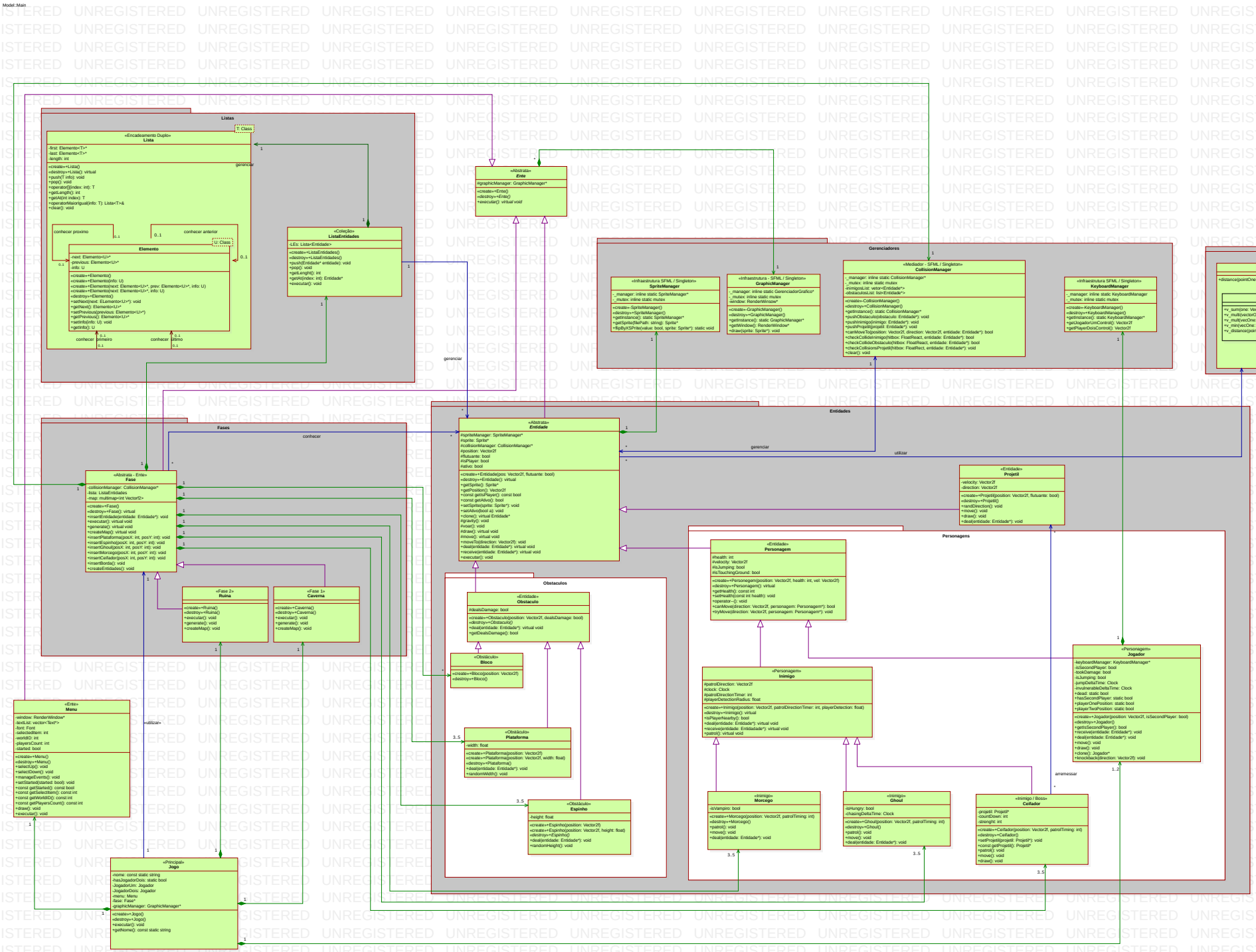
Model::Main
Listas
«Encadeamento Duplo»
Lista
T: Class
-first: Elemento<T>*
-last: Elemento<T>*
-length: int
+create=+Lista()
+destroy=+Lista(): virtual
+push(T info): void
+pop(): void
+operator[](index: int): T
+getLength(): int
+getAt(int index): T
+operatorMaiorIgual(info: T): Lista<T>&
+clear(): void
gerenciar
conhecer proximo
conhecer anterior
Elemento
U: Class
-next: Elemento<U>*
-previous: Elemento<U>*
-info: U
+create=+Elemento()
+create=+Elemento(info: U)
+create=+Elemento(next: Elemento<U>*, prev: Elemento<U>*, info: U)
+create=+Elemento(next: Elemento<U>*, info: U)
+destroy=+Elemento()
+setNext(next: ELemento<U>*): void
+getNext(): Elemento<U>*
+setPrevious(previous: Elemento<U>*)
+getPrevious(): Elemento<U>*
+setInfo(info: U): void
+getInfo(): U
conhecer primeiro
conhecer ultimo
«Abstrata»
Ente
#graphicManager: GraphicManager*
+create=+Ente()
+destroy=+Ente()
+executar(): virtual void
«Coleção»
ListaEntidades
-LEs: Lista<Entidade>
+create=+ListaEntidades()
+destroy=+ListaEntidades()
+push(Entidade* entidade): void
+pop(): void
+getLength(): int
+getAt(index: int): Entidade*
+executar(): void
Gerenciadores
«Infraestrutura SFML / Singleton»
SpriteManager
_manager: inline static SpriteManager*
_mutex: inline static mutex
+create=-SpriteManager()
+destroy=+SpriteManager()
+getInstance(): static SpriteManager*
+getSpriteBePath: string): Sprite*
+BpByXSPrite(value: bool, sprite: Sprite*): static void
«Infraestrutura - SFML / Singleton»
GraphicManager
_manager: inline static GerenciadorGrafico*
_mutex: inline static mutex
-window: RenderWinsow*
+create=-GraphicManager()
+destroy=+GraphicManager()
+getinstance(): static GraphicManager*
+getWindow(): RenderWindow*
+draw(sprite: Sprite*): void
«Mediador - SFML / Singleton»
CollisionManager
_manager: inline static CollisionManager*
_mutex: inline static mutex
-inimigosList: vetor<Entidade*>
-obstaculosList: list<Entidade*>
+create=-CollisionManager()
+destroy=+CollisionManager()
+getInstance(): static CollisionManager*
+pushObstaculo(obstaculo: Entidade*): void
+pushInimigo(inimigo: Entidade*): void
+pushProjetil(projetil: Entidade*): void
+canMoveTo(position: Vector2f, direction: Vector2f, entidade: Entidade*): bool
+checkCollideInimigo(hitbox: FloatRect, entidade: Entidade*): bool
+checkCollideObstaculo(hitbox: FloatRect, entidade: Entidade*): bool
+checkCollisionsProjetil(hitbox: FloatRect, entidade: Entidade*): void
+clear(): void
«Infraestrutura SFML / Singleton»
KeyboardManager
_manager: inline static KeyboardManager
_mutex: inline static mutex
+create=-KeyboardManager()
+destroy=+KeyboardManager()
+getinstance(): static KeyboardManager*
+getJogadorLimControl(): Vector2f
+getPlayerDoisControl(): Vector2f
Utils
Math
+distance(pointOne: Vector2f, pointTwo: Vector2f): static float
Vector
+v_sum(one: Vector2f, two: Vector2f): static Vector2f
+v_mult(vectorOne: Vector2f, constant: float): static Vector2f
+v_mult(vecOne: Vector2f, vecTwo: Vector2f): static Vector2f
+v_min(vecOne: Vector2f, vecTwo: Vector2f): static Vector2f
+v_distance(pointOne: Vector2f, pointTwo: Vector2f): static Vector2f
gerenciar
Fases
conhecer
«Abstrata - Ente»
Fase
-collisionManager: CollisionManager*
-lista: ListaEntidades
-map: multimap<int Vector2>
+create=+Fase()
+destroy=+Fase(): virtual
+insertEntidade(entidade: Entidade*): void
+executar(): virtual void
+generate(): virtual void
+createMap(): virtual void
+insertPlataforma(posX: int, posY: int): void
+insertEspinho(posX: int, posY: int): void
+insertGhoul(posX: int, posY: int): void
+insertMorcego(posX: int, posY: int): void
+insertCeifador(posX: int, posY: int): void
+insertBorda(): void
+createEntidades(): void
«Fase 2»
Ruina
+create=+Ruina()
+destroy=+Ruina()
+executar(): void
+generate(): void
+createMap(): void
«Fase 1»
Caverna
+create=+Caverna()
+destroy=+Caverna()
+executar(): void
+generate(): void
+createMap(): void
Entidades
«Abstrata»
Entidade
#spriteManager: SpriteManager*
#sprite: Sprite*
#collisionManager: CollisionManager*
#position: Vector2f
#flutuante: bool
#isPlayer: bool
#ativo: bool
+create=+Entidade(pos: Vector2f, flutuante: bool)
+destroy=+Entidade(): virtual
+getSprite(): Sprite*
+getPosition(): Vector2f
+const getIsPlayer(): const bool
+const getAtivo(): bool
+setSprite(sprite: Sprite*): void
+setAtivo(bool a): void
+clone(): virtual Entidade*
#gravity(): void
#vivar(): void
#draw(): virtual void
#move(): virtual void
+moveTo(position: Vector2f): void
+deal(entidade: Entidade*): virtual void
+receive(entidade: Entidade*): virtual void
+executar(): void
gerenciar
utilizar
«Entidade»
Projetil
-velocity: Vector2f
-direction: Vector2f
+create=+Projetil(position: Vector2f, flutuante: bool)
+destroy=+Projetil()
+randDirection(): void
+move(): void
+draw(): void
+deal(entidade: Entidade*): void
Obstaculos
«Entidade»
Obstaculo
#dealsDamage: bool
+create=+Obstaculo(position: Vector2f, dealsDamage: bool)
+destroy=+Obstaculo()
+deal(entidade: Entidade*): virtual void
+getDealsDamage(): bool
«Obstáculo»
Bloco
+create=+Bloco(position: Vector2f)
+destroy=+Bloco()
«Obstáculo»
Plataforma
-width: float
+create=+Plataforma(position: Vector2f)
+create=+Plataforma(position: Vector2f, width: float)
+destroy=+Plataforma()
+deal(entidade: Entidade*): void
+randomWidth(): void
«Obstáculo»
Espinho
-height: float
+create=+Espinho(position: Vector2f)
+create=+Espinho(position: Vector2f, height: float)
+destroy=+Espinho()
+deal(entidade: Entidade*): void
+randomHeight(): void
Personagens
«Entidade»
Personagem
#health: int
#velocity: Vector2f
#isJumping: bool
#isTouchingGround: bool
+create=+Personagem(position: Vector2f, health: int, vel: Vector2f)
+destroy=+Personagem(): virtual
+getHealth(): const int
+setHealth(const int health): void
+operator--(): void
+canMove(direction: Vector2f, personagem: Personagem*): bool
+tryMove(direction: Vector2f, personagem: Personagem*): void
«Personagem»
Inimigo
#patrolDirection: Vector2f
#clock: Clock
#patrolDirectionTimer: int
#playerDetectionRadius: float
+create=+Inimigo(position: Vector2f, patrolDirectionTimer: int, playerDetection: float)
+destroy=+Inimigo(): virtual
+isPlayerNearby(): bool
+deal(entidade: Entidade*): virtual void
+receive(entidade: Entidadade*): virtual void
+patrol(): virtual void
«Personagem»
Jogador
-keyboardManager: KeyboardManager*
-isSecondPlayer: bool
-tookDamage: bool
-isJumping: bool
-jumpDeltaTime: Clock
-invulnerableDeltaTime: Clock
+dead: static bool
+hasSecondPlayer: static bool
+playerOnePosition: static bool
+playerTwoPosition: static bool
+create=+Jogador(position: Vector2f, isSecondPlayer: bool)
+destroy=+Jogador()
+getIsSecondPlayer(): bool
+receive(entidade: Entidade*): void
+deal(entidade: Entidade*): void
+move(): void
+draw(): void
+clone(): Jogador*
+hnocback(direction: Vector2f): void
«inimigo»
Morcego
-isVampiro: bool
+create=+Morcego(position: Vector2f, patrolTiming: int)
+destroy=+Morcego()
+patrol(): void
+move(): void
+deal(entidade: Entidade*): void
«inimigo»
Ghoul
-isHungry: bool
-chasingDeltaTime: Clock
+create=+Ghoul(position: Vector2f, patrolTiming: int)
+destroy=+Ghoul()
+patrol(): void
+move(): void
+deal(entidade: Entidade*): void
«inimigo / Boss»
Ceifador
-projetil: Projetil*
-countDown: int
-strenght: int
+create=+Ceifador(position: Vector2f, patrolTiming: int)
+destroy=+Ceifador()
+setProjetil(projetil: Projetil*): void
+const getProjetil(): Projetil*
+patrol(): void
+move(): void
+draw(): void
arremessar
«Entie»
Menu
-window: RenderWindow*
-textList: vector<Text*>
-font: Font
-selectedItem: int
-worldID: int
-playersCount: int
-started: bool
+create=+Menu()
+destroy=+Menu()
+selectUp(): void
+selectDown(): void
+manageEvents(): void
+setStarted(started: bool): void
+const getStarted(): const bool
+const getSelectItem(): const int
+const getWorldID(): const int
+const getPlayersCount(): const int
+draw(): void
+executar(): void
«utilizar»
«Principal»
Jogo
-nome: const static string
-hasJogadorDois: static bool
-JogadorUm: Jogador
-JogadorDois: Jogador
-menu: Menu
-fase: Fase*
-graphicManager: GraphicManager*
+create=+Jogo()
+destroy=+Jogo()
+executar(): void
+getNome(): const static string