

ParticleSystem

+ std::vector< Particle > m_particles

+ void update(sf::Time t_deltaTime)

+ void render(sf::RenderWindow &m_window)

+ void addParticle(const Particle &m_particle)

Building

sf::Texture m_smokeTexture

sf::Vector2f m_position

std::string m_name

sf::Texture m_buildingTexture

sf::Sprite m_buildingSprite

sf::CircleShape m_placementRadius

sf::RectangleShape m_healthBar

sf::RectangleShape m_healthBarBackground

int m_health

int m_maxHealth

int m_cost

BuildingType m_type

bool m_placementRadiusVisible

+ Building(BuildingType m_type)

+ virtual ~Building()

+ virtual void update(sf::Time t_deltaTime)

+ virtual void render(sf::RenderWindow &m_window)

+ void takeDamage(float m_damageAmount)

+ void setPosition(const sf::Vector2f &m_position)

+ sf::Vector2f getPosition()

+ int getCost() const

+ float getHealth() const

+ BuildingType getType() const

+ bool checkAffordability()

+ void setPlacementRadiusSize(float m_radius)

+ void setHealth(float newHealth)

+ const sf::Sprite & getBuildingSprite() const

+ const sf::Texture & getBuildingTexture() const

+ const sf::CircleShape & getPlacementRadius() const

+ void updateHealthBar()

void initSmokeEffect()

void spawnSmokeEffect()

Unit

+ std::vector< Bullet > m_bullets

+ std::vector< Missile > m_missiles

+ int m_unitIndex

+ bool isSelected

+ bool m_active

+ bool m_isEnemy

UnitTypeClass m_unitTypeClass

std::vector< Unit * > m_enemyUnits

std::vector< Building * > m_enemyBuildings

const std::vector< std::vector< Tile > > m_tiles

std::vector< sf::Vector2f > m_debugRays

sf::Texture m_unitTexture

sf::Sprite m_unitSprite

sf::RectangleShape m_healthBarBackground

sf::RectangleShape m_healthBarForeground

sf::Texture m_weaponTexture

sf::Sprite m_weaponSprite

sf::CircleShape m_viewCircleShape

sf::Shader m_glowShader

int m_cost

float m_health

const float m_maxHealth

float m_viewRadius

float m_damage

float m_speed

float m_stoppingDistance

float m_slowingRadius

float m_maxForce

float m_rotationSpeed

float m_bulletSpeed

float m_closestDistance

float m_closestBuildingDistance

float m_arrivalTolerance

const float PI

bool isOrbiting

bool isReloading

sf::Vector2f m_position

sf::Vector2f m_targetPosition

sf::Vector2f m_velocity

sf::Vector2f m_directionToEnemy

sf::Vector2f m_acceleration

sf::Clock m_slowEffectClock

bool m_isSlowed

bool m_isGraduallySlowed

bool m_inPostSlowWait

float m_slowDownStartTime

float m_minimumSpeedFactor

float m_slowEffectDuration

float m_originalSpeed

float m_postSlowWaitDuration

+ Unit()

+ virtual ~Unit()

+ virtual void update(sf::Time t_deltaTime, std::vector< Unit * > &allyUnits)

+ virtual void render(sf::RenderWindow &m_window)

+ virtual UnitType getUnitType() const =0

+ void setPosition(const sf::Vector2f &m_position)

+ void setHealth(float m_setHealth)

+ void moveTo(const sf::Vector2f &m_targetPos)

+ void setSelected(bool m_selected)

+ void setTargetPosition(const sf::Vector2f &m_targetPos)

+ void takeDamage(float m_damageAmount)

+ void addHealth(float m_healthAmount)

+ void applySlowEffect(float m_speedFactor, float m_duration, float m_postSlowWait)

+ void setEnemyUnits(std::vector< Unit * > &m_enemyUnits)

+ void setEnemyBuildings(std::vector< Building * > &m_enemyBuildings)

+ void setTiles(const std::vector< std::vector< Tile > > &m_tiles)

+ const sf::Sprite & getSprite() const

+ sf::Vector2f getPosition() const

+ sf::Vector2f getTargetPosition() const

+ sf::Vector2f normalize(const sf::Vector2f m_source)

+ sf::Vector2f steerTowards(sf::Vector2f m_target)

+ sf::Vector2f rotateVector(sf::Vector2f m_vector, float m_angleDegrees)

+ sf::Vector2f lerp(const sf::Vector2f &m_start, const sf::Vector2f &m_end, float m_time)

+ sf::Vector2f findAvoidanceDirection(const sf::Vector2f &m_currentPosition, float m_checkAheadDistance)

+ float angleFromVector(const sf::Vector2f &m_vector)

+ float getViewRadius() const

+ float distance(const sf::Vector2f &a, const sf::Vector2f &b)

+ float magnitude(const sf::Vector2f &v) const

+ float getHealth() const

+ float toDegrees(float radians)

+ float angleBetweenVectors(sf::Vector2f vec1, sf::Vector2f vec2)

+ float getDamage() const

+ bool checkAffordability()

+ bool isActive() const

void initView()

void initHealthBar()

void initShader()

void avoidCollisionsWithUnits(std::vector< Unit * > &m_allyUnits)

void avoidCollisionsWithWalls()

void orientSpriteToMovement(sf::Time t_deltaTime)

virtual void squadEntityRemoval()

virtual void squadEntityRegain()

VehicleUnit

+ VehicleUnit()

+ virtual ~VehicleUnit()

+ void update(sf::Time t_deltaTime, std::vector< Unit * > &m_allyUnits) override

+ void render(sf::RenderWindow &m_window) override