Tower Defence A* Arena

NODE
int x,y
bool walkable
float gCost, hCost, baseCost, slowMultiplier
Node* parent

Node(int x , **int** y, **bool** walkable)
float getFCost() const
bool operator==(const **Node&** other) const
void resetCosts()
float getMovementCost()

GRID
int width, height
vector<vector<Node>> nodes
sf::Vector2i startCell, endCell

Grid(int w, **int** y)
void initialize(**int** w, **int** h);
Node* getNode(**int** x, **int** y);
vector<Node*> getNeighbors(**Node*** node);
bool isWalkable(**int** x, **int** y);
void setObstacle(**int** x, **int** y, **bool** blocked);
void setStartEnd(**sf::Vector2i** start, **sf::Vector2i** end);
void draw(**sf::RenderWindow&** window);
void resetCosts()
void applyFrostEffects(**int** centerX, **int** centerY, **int** radius,
float slowMultiplier)
void resetFrostEffects()
void resetCosts()

ENEMY_MANAGER
vector<unique_ptr<Enemy>> enemies;
Grid* grid;
AStarPathfinder* pathfinder;
sf::Vector2i startCell, endCell;

---

void spawnEnemy();
void update(**float** deltaTime);
void draw(**sf::RenderWindow&** window);
void recalculatePaths();

TOWER_MANGER
vector<unique_ptr<Tower>> towers
Grid* grid
Projectile_Manger projectileManager

---

void update(**float** deltaTime, **vector<unique_ptr<Enemy>>&** enemies)
void draw(**sf::RenderWindow&** window)
void isOccupied (**sf::Vector2i** gridPos)
void placeTower(**TowerType** type, **sf::Vector2i** gridPos)

PROJECTILE_MANAGER
vector<unique_ptr<Projectile>> projectiles

---

void spawnProjectile(**sf::Vector2f** start, **sf::Vector2f** dir, **float** speed, **int** dmg)
void update(**float** deltaTime, **vector<unique_ptr<Enemy>>&** enemies)
void draw(**sf::RenderWindow&** window)

PROJECTILE
sf::Vector2f position
sf::Vector2f direction
float speed
int damage
bool active

---

void update(**float** deltaTime)
void draw(**sf::RenderWindow&** window)
void checkCollision(**Enemy&** enemy)

A_STAR_PATHFINDER
Grid* grid;
priority_queue<Node*, vector<Node*>, CompareFCost> openSet;
unordered_set<Node*> closedSet;

A_Star_PathFinder(Grid grid)
vector<Node*> findPath(**Node*** start, **Node*** end);
float calculateHCost(**Node*** a, **Node*** b);
void reconstructPath(**Node*** start, **Node*** end, **vector<Node*>&** path);

GAME_MANAGER
Grid grid
AStarPathfinder pathfinder
EnemyManager enemyManager
TowerManager towerManager
sf::RenderWindow window
float deltaTime

---

void initialize()
void handleInput()
void update()
void render()
void run()

TOWER_TYPE<<ENUM>>
Barrier
Gatling
Frost
Artillery

TOWER <<ABSTRACT>>
sf::Vector2f position
float range
float fireRate
float cooldown
int cost
bool isBlocking
TOWER_TYPE towerType

---

virtual void update(**float** deltaTime, **vector<unique_ptr<Enemy>>&** enemies)
virtual void draw(**sf::RenderWindow&** window)
bool canAttack(**Enemy*** enemy)

UI_MANAGER
int money, lives
TowerType selectedTower

---

void update(**float** deltaTime)
void draw(**sf::RenderWindow&** window)
void updateStats(**int** deltaMoney, **int** deltaLives)

ENEMY
sf::Vector2f position;
float speed;
int health;
vector<Node*> path;
int currentNodeIndex;
bool reachedGoal;

---

void setPath(const **std::vector<Node*>&** path);
void update(**float** deltaTime);
void draw(**sf::RenderWindow&** window);
bool isDead() const;
void takeDamage(int dmg);

MIN_HEAP
vector<Node*> heap

---

void push(**Node*** node)
Node* pop()
void headpifyUp(**int** index)
void heapifyDown(**int** index)
bool empty()

NORMAL_ENEMY

NormalEnemy(const std::vector<Node*>& path)

TANK_ENEMY

TankEnemy(const std::vector<Node*>& path)

GATLING TOWER
float bulletSpeed

BarrierTower(sf::Vector2f pos)
void update(**float** deltaTime, **vector<unique_ptr<Enemy>>&** enemies)
void draw(**sf::RenderWindow&** window)

FROST TOWER
float slowMultiplier
int aoeRangeCells
float pathCostIncrease
sf::RectanglerShape aoeVisual

FrostTower(sf::Vector2f pos)
void update(**float** deltaTime, **vector<unique_ptr<Enemy>>&** enemies)
void draw(**sf::RenderWindow&** window)
void applyForstEffect(**Grid&** grid)

SHEILD_ENEMY
int shield

SheildEnemy(const std::vector<Node*>& path)
void takeDamage(**int** dmg)
int getShield()

FAST_ENEMY

FastEnemy(const std::vector<Node*>& path)

BARRIER TOWER

BarrierTower(sf::Vector2f pos)
void update(**float** deltaTime, **vector<unique_ptr<Enemy>>&** enemies)
void draw(**sf::RenderWindow&** window)

ARTILLERY TOWER
float aoeRadius
int damage

ArtilleryTower(sf::Vector2f pos)
void update(**float** deltaTime, **vector<unique_ptr<Enemy>>&** enemies)
void draw(**sf::RenderWindow&** window)