

Namespace TaleOfTiles

Classes

[MainGameManager](#)

The MainGameManager class is a singleton that manages various aspects of the game.

[PlayerLevel](#)

The PlayerLevel class handles the logic for player leveling.

Class MainGameManager

Namespace: [TaleOfTiles](#)

Assembly: Assembly-CSharp.dll

The MainGameManager class is a singleton that manages various aspects of the game.

```
public class MainGameManager : Singleton<MainGameManager>
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton](#)<[MainGameManager](#)> ← MainGameManager

Inherited Members

[Singleton](#)<[MainGameManager](#)>.m_instance , [Singleton](#)<[MainGameManager](#)>.Instance ,
[Singleton](#)<[MainGameManager](#)>.Awake() , [Singleton](#)<[MainGameManager](#)>.Prepare() ,
[SingletonBase](#).AllSingletons

Remarks

This class handles the initialization and coordination of different managers like AudioManager, UIManager, etc. It is responsible for setting up the game environment, managing the game state, and transitioning between different scenes and states.

Fields

additional_screens_prefab

```
public GameObject additional_screens_prefab
```

Field Value

GameObject

Prefab for additional screens in the game.

audio_manager

```
public AudioManager audio_manager
```

Field Value

[AudioManager](#)

Manager for handling audio-related functionalities.

battle_arena_manager

```
public BattleArenaManager battle_arena_manager
```

Field Value

[BattleArenaManager](#)

Manager for the battle arena mechanics.

cards_holder_manager

```
public CardsHolderManager cards_holder_manager
```

Field Value

[CardsHolderManager](#)

Manager for the player's card collection.

combbat_manager

```
public CombatManager combbat_manager
```

Field Value

[CombatManager](#)

Manager for handling combat and game logic.

data_manager

```
public DataManager data_manager
```

Field Value

[DataManager](#)

Manager for handling data storage and retrieval.

general_audio_group_prefab

```
public GameObject general_audio_group_prefab
```

Field Value

GameObject

Prefab for general audio in the game.

journey_manager

```
public JourneyManager journey_manager
```

Field Value

[JourneyManager](#)

Manager for handling the game's journey or campaign mode.

loading_screen_manager

```
public LoadingScreenManager loading_screen_manager
```

Field Value

[LoadingScreenManager](#)

Manager for handling loading screens.

player

```
private PlayerCharacter player
```

Field Value

[PlayerCharacter](#)

Represents the player character in the game.

player_level

```
public PlayerLevel player_level
```

Field Value

[PlayerLevel](#)

Represents the player's level in the game.

scene_switcher_manager

```
public SceneSwitcherManager scene_switcher_manager
```

Field Value

[SceneSwitcherManager](#)

Manager for scene transitions.

tosave

```
public JourneyObject tosave
```

Field Value

[JourneyObject](#)

Object to store the current state of the journey.

ui_manager

```
public UIManager ui_manager
```

Field Value

[UIManager](#)

Manager for user interface elements and interactions.

Methods

CreatePlayerCharacter()

Creates a new player character and initializes its properties.

```
public void CreatePlayerCharacter()
```

GetPlayer()

Returns the current player character.

```
public PlayerCharacter GetPlayer()
```

Returns

[PlayerCharacter](#)

The current player character.

HomeButtonClick(GameObject)

Handles the home button click event.

```
public void HomeButtonClick(GameObject go = null)
```

Parameters

go GameObject

The GameObject that calls this method.

LoadBattleArena()

Loads the battle arena.

```
public void LoadBattleArena()
```

LoadBattleArenaCoroutine()

A coroutine to load the battle arena.

```
private IEnumerator LoadBattleArenaCoroutine()
```

Returns

[IEnumerator](#)

An IEnumerator for coroutine sequencing.

LoadJourneyMap(string, GameObject)

Loads the journey map.

```
public void LoadJourneyMap(string journey_id = "", GameObject caller = null)
```

Parameters

journey_id [string](#)

ID of the journey to be loaded.

caller [GameObject](#)

The GameObject that calls this method.

LoadJourneyMapCoroutine(GameObject)

A coroutine to load the journey map.

```
private IEnumerator LoadJourneyMapCoroutine(GameObject caller = null)
```

Parameters

caller [GameObject](#)

The GameObject that calls this method.

Returns

[IEnumerator](#)

An IEnumerator for coroutine sequencing.

LoadJourneySelectionScene(GameObject)

Loads the journey selection scene.

```
public void LoadJourneySelectionScene(GameObject caller = null)
```

Parameters

caller GameObject

The GameObject that calls this method.

LoadJourneySelectionSceneCoroutine(GameObject)

A coroutine to load the journey selection scene.

```
private IEnumerator LoadJourneySelectionSceneCoroutine(GameObject caller = null)
```

Parameters

caller GameObject

The GameObject that calls this method.

Returns

[IEnumerator](#)

An IEnumerator for coroutine sequencing.

LoadLevelSelectionScreen(GameObject)

Loads the level selection screen.

```
public void LoadLevelSelectionScreen(GameObject caller = null)
```

Parameters

caller GameObject

The GameObject that calls this method.

LoadLevelSelectionScreenCoroutine(GameObject)

A coroutine to load the level selection screen.

```
private IEnumerator LoadLevelSelectionScreenCoroutine(GameObject caller = null)
```

Parameters

caller GameObject

The GameObject that calls this method.

Returns

[IEnumerator](#)

An IEnumerator for coroutine sequencing.

PrepareManagers()

A coroutine to prepare all the managers sequentially.

```
private IEnumerator PrepareManagers()
```

Returns

[IEnumerator](#)

An IEnumerator for coroutine sequencing.

SelectLevel(string)

Selects a level by its ID.

```
public void SelectLevel(string level_id)
```

Parameters

level_id [string](#)

ID of the level to be selected.

SelectLevel(LevelObject)

Selects a level by its LevelObject instance.

```
public void SelectLevel(LevelObject level)
```

Parameters

level [LevelObject](#)

LevelObject instance of the level to be selected.

SetMainCameraToAllCanvases()

Sets the main camera to all canvases.

```
private void SetMainCameraToAllCanvases()
```

Start()

Starts the game manager by initializing all the managers and preparing the game environment.

```
private void Start()
```

Class PlayerLevel

Namespace: [TaleOfTiles](#)

Assembly: Assembly-CSharp.dll

The PlayerLevel class handles the logic for player leveling.

```
public class PlayerLevel
```

Inheritance

[object](#) ↗ ← PlayerLevel

Properties

CurrentLevel

```
public int CurrentLevel { get; }
```

Property Value

[int](#) ↗

Gets the current level of the player.

CurrentLevelExpCost

```
public int CurrentLevelExpCost { get; }
```

Property Value

[int](#) ↗

Gets the experience cost of the current level.

CurrentTotalExp

```
public int CurrentTotalExp { get; }
```

Property Value

[int ↗](#)

Gets the current total experience of the player.

ExtraExpFromCurrentLevel

```
public int ExtraExpFromCurrentLevel { get; }
```

Property Value

[int ↗](#)

Gets the extra experience from the current level.

NextLevelExpCost

```
public int NextLevelExpCost { get; }
```

Property Value

[int ↗](#)

Gets the experience cost of the next level.

Methods

ExpCostPerLevel(float)

Calculates the experience cost for a given target level.

```
private int ExpCostPerLevel(float target_level)
```

Parameters

target_level [float](#)

The target level for which the cost is calculated.

Returns

[int](#)

The experience cost for the given target level.

Namespace TaleOfTiles.AI

Classes

[AIBehaviour](#)

A class representing the AI behavior in a tile-matching game. It includes functionalities for the AI to analyze the board, make decisions, and perform moves.

[BoardPos](#)

Represents a position on the board.

[Combination](#)

Represents a combination of game pieces.

[Move](#)

Represents a potential move on the board.

[VirtualBoard](#)

Represents a virtual game board used for AI decision-making.

[VirtualGamePiece](#)

Represents a virtual representation of a game piece.

Structs

[Move.GemActiveEffectScore](#)

Defines the score for active gem effects in a move.

[Move.GemStatusEffectScore](#)

Defines the score for status effects caused by gems in a move.

Class AIBehaviour

Namespace: [TaleOfTiles.AI](#)

Assembly: Assembly-CSharp.dll

A class representing the AI behavior in a tile-matching game. It includes functionalities for the AI to analyze the board, make decisions, and perform moves.

```
public class AIBehaviour : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← AIBehaviour

Fields

board

Reference to the BoardManager of the game.

```
public BoardManager board
```

Field Value

[BoardManager](#)

inputBlocked

Indicates if AI input is currently blocked.

```
public bool inputBlocked
```

Field Value

[bool](#)

intelligence

Represents the intelligence level of the AI.

```
public float intelligence
```

Field Value

[float](#)

makingMoves

Indicates if AI is in the process of making moves.

```
public bool makingMoves
```

Field Value

[bool](#)

target_effect

Target active effect that AI aims to achieve with its move.

```
public GemActiveEffectType target_effect
```

Field Value

[GemActiveEffectType](#)

target_status_effect

Target status effect that AI aims to apply with its move.

```
public GemStatusEffectType target_status_effect
```

Field Value

[GemStatusEffectType](#)

wait_time_after_move

Time to wait after making a move.

```
public float wait_time_after_move
```

Field Value

[float](#)

wait_time_before_move

Time to wait before making a move.

```
public float wait_time_before_move
```

Field Value

[float](#)

Properties

player

Gets the PlayerCharacter associated with the board.

```
public PlayerCharacter player { get; }
```

Property Value

[PlayerCharacter](#)

Methods

MakeMoveSmart()

Coroutine for executing a smart move in the game. Waits for a specified time, evaluates the board, and performs the best possible move.

```
private IEnumerator MakeMoveSmart()
```

Returns

[IEnumerator](#)

Returns an IEnumerator for coroutine management.

MakeOneMove()

Initiates the AI move process.

```
public void MakeOneMove()
```

MakeOneMoveCoroutine()

Coroutine to manage the process of making a single move. Blocks input during the move and then enables it upon completion.

```
private IEnumerator MakeOneMoveCoroutine()
```

Returns

[IEnumerator](#)

Returns an IEnumerator for coroutine management.

Start()

Start is called before the first frame update. Initializes the game board and related settings.

```
private void Start()
```

Class BoardPos

Namespace: [TaleOfTiles.AI](#)

Assembly: Assembly-CSharp.dll

Represents a position on the board.

```
public class BoardPos
```

Inheritance

[object](#) ← BoardPos

Constructors

BoardPos()

Constructor to create a board position with specified coordinates.

```
public BoardPos()
```

BoardPos(int, int)

```
public BoardPos(int x, int y)
```

Parameters

x [int](#)

y [int](#)

Fields

X

```
public int x
```

Field Value

[int](#)

y

```
public int y
```

Field Value

[int](#)

Methods

Equals(object)

Determines if the specified object is equal to the current object.

```
public override bool Equals(object obj)
```

Parameters

[obj](#) [object](#)

The object to compare with the current object.

Returns

[bool](#)

True if the specified object is equal to the current object; otherwise, false.

GetHashCode()

Serves as the default hash function.

```
public override int GetHashCode()
```

Returns

[int](#)

A hash code for the current object.

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

[string](#)

A string that represents the current object.

Class Combination

Namespace: [TaleOfTiles.AI](#)

Assembly: Assembly-CSharp.dll

Represents a combination of game pieces.

```
public class Combination
```

Inheritance

[object](#) ↗ ← Combination

Constructors

Combination()

Default constructor for a combination.

```
public Combination()
```

Combination(List<VirtualGamePiece>)

Constructor for creating a combination with a list of game pieces.

```
public Combination(List<VirtualGamePiece> gamePieces)
```

Parameters

gamePieces [List](#) ↗ <[VirtualGamePiece](#)>

List of VirtualGamePiece objects to form the combination.

Fields

gamePieces

```
public List<VirtualGamePiece> gamePieces
```

Field Value

[List](#) <[VirtualGamePiece](#)>

Properties

gamePiecesCount

```
public int gamePiecesCount { get; }
```

Property Value

[int](#)

Methods

FindFirstCombo(List<VirtualGamePiece>)

Finds the first combination of matching game pieces in the given list.

```
private List<VirtualGamePiece> FindFirstCombo(List<VirtualGamePiece> input_list)
```

Parameters

input_list [List](#) <[VirtualGamePiece](#)>

List of game pieces to search for combinations.

Returns

[List](#) <[VirtualGamePiece](#)>

List of matching game pieces forming a combination.

GamePieceListString(List<VirtualGamePiece>)

Converts a list of game pieces to a string representation.

```
public static string GamePieceListString(List<VirtualGamePiece> list)
```

Parameters

list [List](#)<[VirtualGamePiece](#)>

List of VirtualGamePiece objects.

Returns

[string](#)

A string representing the list of game pieces.

SearchConnectedPiece(VirtualGamePiece, List<VirtualGamePiece>, List<VirtualGamePiece>)

Searches for connected pieces matching the caller piece and adds them to the matches list.

```
private void SearchConnectedPiece(VirtualGamePiece caller_vp, List<VirtualGamePiece> rest,
List<VirtualGamePiece> matches)
```

Parameters

caller_vp [VirtualGamePiece](#)

The calling virtual game piece.

rest [List](#)<[VirtualGamePiece](#)>

List of remaining game pieces to search.

matches [List](#)<[VirtualGamePiece](#)>

List to add matching game pieces to.

SplitIntoCombos()

Splits the current combination into multiple combinations based on matching game pieces.

```
public List<Combination> SplitIntoCombos()
```

Returns

[List](#) <[Combination](#)>

A list of combinations, each containing a group of matching game pieces.

Exceptions

[Exception](#)

Thrown when the number of iterations exceeds a set limit, indicating a possible infinite loop.

Class Move

Namespace: [TaleOfTiles.AI](#)

Assembly: Assembly-CSharp.dll

Represents a potential move on the board.

```
public class Move
```

Inheritance

[object](#) ← Move

Constructors

Move()

Default constructor. Initializes a new instance of the Move class with default start and end positions.

```
public Move()
```

Move(BoardPos, BoardPos, VirtualGamePiece, VirtualGamePiece)

Constructor that initializes a new instance of the Move class with specified start and end positions and game pieces.

```
public Move(BoardPos start_pos, BoardPos end_pos, VirtualGamePiece startGamePiece,  
VirtualGamePiece endGamePiece)
```

Parameters

`start_pos` [BoardPos](#)

Starting position of the move.

`end_pos` [BoardPos](#)

Ending position of the move.

startGamePiece [VirtualGamePiece](#)

Game piece at the starting position.

endGamePiece [VirtualGamePiece](#)

Game piece at the ending position.

Move(BoardPos, BoardPos, VirtualGamePiece, VirtualGamePiece, List<Combination>)

Constructor that initializes a new instance of the Move class with specified start and end positions, game pieces, and combinations.

```
public Move(BoardPos start_pos, BoardPos end_pos, VirtualGamePiece startGamePiece,  
VirtualGamePiece endGamePiece, List<Combination> combinations)
```

Parameters

start_pos [BoardPos](#)

Starting position of the move.

end_pos [BoardPos](#)

Ending position of the move.

startGamePiece [VirtualGamePiece](#)

Game piece at the starting position.

endGamePiece [VirtualGamePiece](#)

Game piece at the ending position.

combinations [List](#)<[Combination](#)>

List of combinations resulting from the move.

Fields

board

Represents the current state of the board.

```
public VirtualGamePiece[,] board
```

Field Value

[VirtualGamePiece\[,\]](#)

combinations

List of combinations created as a result of the move.

```
public List<Combination> combinations
```

Field Value

[List ↗ <Combination>](#)

endGamePiece

The game piece at the ending position of the move.

```
public VirtualGamePiece endGamePiece
```

Field Value

[VirtualGamePiece](#)

end_pos

Ending position of the move.

```
public BoardPos end_pos
```

Field Value

[BoardPos](#)

gem_active_effects

List of active gem effects generated by the move.

```
public List<Move.GemActiveEffectScore> gem_active_effects
```

Field Value

[List](#) <[Move.GemActiveEffectScore](#)>

gem_status_effects

List of status effects caused by gems as a result of the move.

```
public List<Move.GemStatusEffectScore> gem_status_effects
```

Field Value

[List](#) <[Move.GemStatusEffectScore](#)>

gem_super_combos

List of super combos triggered by the move.

```
public List<GemSuperCombo> gem_super_combos
```

Field Value

[List](#) <[GemSuperCombo](#)>

startGamePiece

The game piece at the starting position of the move.

```
public VirtualGamePiece startGamePiece
```

Field Value

[VirtualGamePiece](#)

start_pos

Starting position of the move.

```
public BoardPos start_pos
```

Field Value

[BoardPos](#)

Methods

CalculateEffects(PlayerCharacter)

Calculates the effects of the move based on the current combinations. Updates the lists of active effects, status effects, and super combos.

```
public void CalculateEffects(PlayerCharacter player)
```

Parameters

player [PlayerCharacter](#)

The player character for whom the effects are being calculated.

GetEffect(GemActiveEffectType)

Retrieves the score for a specific type of active gem effect.

```
public Move.GemActiveEffectScore GetEffect(GemActiveEffectType type)
```

Parameters

type [GemActiveEffectType](#)

The type of gem active effect to retrieve.

Returns

[Move.GemActiveEffectScore](#)

The score for the specified type of gem active effect.

GetStatusEffect(GemStatusEffectType, bool)

Retrieves the score for a specific type of status effect. Can optionally combine scores of the same type.

```
public Move.GemStatusEffectScore GetStatusEffect(GemStatusEffectType type, bool  
combine_same_effects_by_type = true)
```

Parameters

type [GemStatusEffectType](#)

The type of status effect to retrieve.

combine_same_effects_by_type [bool](#)

Whether to combine scores of the same type of status effect.

Returns

[Move.GemStatusEffectScore](#)

The score for the specified type of status effect.

PrintMove()

Prints details of the move, including combinations and effects, to the debug log.

```
public void PrintMove()
```

RecalculateCombinations()

Recalculates the combinations in the move by splitting existing combinations into smaller ones if necessary.

```
private void RecalculateCombinations()
```

Struct Move.GemActiveEffectScore

Namespace: [TaleOfTiles.AI](#)

Assembly: Assembly-CSharp.dll

Defines the score for active gem effects in a move.

```
[Serializable]
public struct Move.GemActiveEffectScore
```

Fields

effect_type

```
public GemActiveEffectType effect_type
```

Field Value

[GemActiveEffectType](#)

effect_value

```
public float effect_value
```

Field Value

[float](#)

shield_type

```
public GemEffectShieldType shield_type
```

Field Value

GemEffectShieldType

Struct Move.GemStatusEffectScore

Namespace: [TaleOfTiles.AI](#)

Assembly: Assembly-CSharp.dll

Defines the score for status effects caused by gems in a move.

```
[Serializable]
public struct Move.GemStatusEffectScore
```

Fields

affected_gem_count

```
public float affected_gem_count
```

Field Value

[float](#)

duration

```
public float duration
```

Field Value

[float](#)

effect_power

```
public float effect_power
```

Field Value

[float](#) ↗

effect_type

`public GemStatusEffectType effect_type`

Field Value

[GemStatusEffectType](#)

touch_damage

`public float touch_damage`

Field Value

[float](#) ↗

Class VirtualBoard

Namespace: [TaleOfTiles.AI](#)

Assembly: Assembly-CSharp.dll

Represents a virtual game board used for AI decision-making.

```
public class VirtualBoard
```

Inheritance

[object](#) ← VirtualBoard

Methods

ChooseMove(List<Move>, PlayerCharacter, GemActiveEffectType, GemStatusEffectType, float)

Chooses the best move for a player based on the current game situation, possible move effects, and AI intelligence.

```
public Move ChooseMove(List<Move> moves, PlayerCharacter player, GemActiveEffectType target_effect = GemActiveEffectType.None, GemStatusEffectType target_status_effect = GemStatusEffectType.None, float intelligence = 1)
```

Parameters

moves [List](#)<[Move](#)>

List of potential moves.

player [PlayerCharacter](#)

The player character for whom the move is being calculated.

target_effect [GemActiveEffectType](#)

The targeted active gem effect.

target_status_effect [GemStatusEffectType](#)

The targeted status effect.

intelligence [float](#)

The intelligence level of the AI making the decision.

Returns

[Move](#)

The chosen move based on the criteria.

ChooseMoveInSortedlistByGemEffectAndIntelligence(List<Move>, GemActiveEffectType, float, int)

Chooses a move from a sorted list of moves based on a specific gem effect type and AI intelligence level.

```
private Move ChooseMoveInSortedlistByGemEffectAndIntelligence(List<Move> available_moves,  
GemActiveEffectType effect_type, float intelligence, int start_index = 1)
```

Parameters

available_moves [List](#)<Move>

List of available moves.

effect_type [GemActiveEffectType](#)

The gem effect type to consider.

intelligence [float](#)

The AI intelligence level.

start_index [int](#)

The starting index for choosing a move.

Returns

[Move](#)

The chosen move based on the specified criteria.

ChooseMoveInSortedlistByGemStatusEffectAndIntelligence(List<Move>, GemStatusEffectType, float, int)

Chooses a move from a sorted list of moves based on a specific gem status effect type and AI intelligence level.

```
private Move ChooseMoveInSortedlistByGemStatusEffectAndIntelligence(List<Move>
available_moves, GemStatusEffectType effect_type, float intelligence, int start_index = 1)
```

Parameters

`available_moves` [List](#)<[Move](#)>

List of available moves.

`effect_type` [GemStatusEffectType](#)

The gem status effect type to consider.

`intelligence` [float](#)

The AI intelligence level.

`start_index` [int](#)

The starting index for choosing a move.

Returns

[Move](#)

The chosen move based on the specified criteria.

CollapseBoard(VirtualGamePiece[,])

Collapses the game board, moving game pieces down to fill empty spaces.

```
public bool CollapseBoard(VirtualGamePiece[,] board_field)
```

Parameters

board_field [VirtualGamePiece\[,\]](#)

The game board to collapse.

Returns

[bool](#) ↗

True if any pieces were moved; otherwise, false.

FillBoard(VirtualGamePiece[,], VirtualGamePiece[])

Fills the game board with virtual game pieces, ensuring no immediate matches are formed upon filling.

```
public VirtualGamePiece[,] FillBoard(VirtualGamePiece[,] board_field,  
VirtualGamePiece[] gems)
```

Parameters

board_field [VirtualGamePiece\[,\]](#)

The game board to fill.

gems [VirtualGamePiece\[\]](#)

Array of virtual game pieces to use for filling.

Returns

[VirtualGamePiece\[\]](#)

The filled game board.

FilterMovesListGemEffect(List<Move>, GemActiveEffectType)

Filters a list of moves based on a specific gem effect type.

```
private List<Move> FilterMovesListGemEffect(List<Move> moves, GemActiveEffectType type)
```

Parameters

moves [List<Move>](#)

List of potential moves.

type [GemActiveEffectType](#)

The type of gem effect to filter by.

Returns

[List<Move>](#)

A list of moves filtered by the specified gem effect type.

FilterMovesListGemStatusEffect(List<Move>, GemStatusEffectType)

Filters a list of moves based on a specific gem status effect type.

```
private List<Move> FilterMovesListGemStatusEffect(List<Move> moves,  
GemStatusEffectType type)
```

Parameters

moves [List<Move>](#)

List of potential moves.

type [GemStatusEffectType](#)

The type of gem status effect to filter by.

Returns

[List<Move>](#)

A list of moves filtered by the specified gem status effect type.

FindAllMatches(VirtualGamePiece[,])

Finds all matches on the game board.

```
public List<BoardPos> FindAllMatches(VirtualGamePiece[,] board_field)
```

Parameters

board_field [VirtualGamePiece\[,\]](#)

The game board to search for matches.

Returns

[List](#)<[BoardPos](#)>

A list of positions where matches are found.

FindAllMoves(VirtualGamePiece[,])

Finds all possible moves on the game board.

```
private List<Move> FindAllMoves(VirtualGamePiece[,] board_field)
```

Parameters

board_field [VirtualGamePiece\[,\]](#)

The game board to search for moves.

Returns

[List](#)<[Move](#)>

A list of all possible moves.

FindBestMove(VirtualGamePiece[,], PlayerCharacter, GemActiveEffectType, GemStatusEffectType, float)

Identifies the best move for a player based on the current state of the game board and desired effects.

```
public Move FindBestMove(VirtualGamePiece[,] board_field, PlayerCharacter player,  
GemActiveEffectType target_effect = GemActiveEffectType.None, GemStatusEffectType
```

```
target_status_effect = GemStatusEffectType.None, float intelligence = 1)
```

Parameters

board_field [VirtualGamePiece](#)[,]

The current game board.

player [PlayerCharacter](#)

The player character for whom the move is being calculated.

target_effect [GemActiveEffectType](#)

The targeted active gem effect.

target_status_effect [GemStatusEffectType](#)

The targeted status effect.

intelligence [float](#)

The intelligence level of the AI making the decision.

Returns

[Move](#)

The best move identified, or null if no moves are available.

FindMatches(int, int, Vector2, VirtualGamePiece[,], int)

```
private List<VirtualGamePiece> FindMatches(int startX, int startY, Vector2 searchDirection,  
VirtualGamePiece[,] board, int minLength = 3)
```

Parameters

startX [int](#)

startY [int](#)

searchDirection [Vector2](#)

`board` [VirtualGamePiece\[,\]](#)

`minLength` [int](#)

Returns

[List](#) <[VirtualGamePiece](#)>

FindMatchesAt(int, int, VirtualGamePiece[,])

Finds all matches on the board that include a virtual game piece at a specified position.

```
public List<BoardPos> FindMatchesAt(int x, int y, VirtualGamePiece[,] board_field)
```

Parameters

`x` [int](#)

X-coordinate of the game piece.

`y` [int](#)

Y-coordinate of the game piece.

`board_field` [VirtualGamePiece\[,\]](#)

The game board to search for matches.

Returns

[List](#) <[BoardPos](#)>

A list of positions where matches are found.

GetRandomVirtualGamePiece(VirtualGamePiece[])

Selects a random virtual game piece from an array, considering the rarity of each piece.

```
public VirtualGamePiece GetRandomVirtualGamePiece(VirtualGamePiece[] gems)
```

Parameters

`gems` [VirtualGamePiece\[\]](#)

Array of virtual game pieces to choose from.

Returns

[VirtualGamePiece](#)

A randomly selected virtual game piece.

HasMatchOnFill(int, int, VirtualGamePiece[,], int)

```
private bool HasMatchOnFill(int x, int y, VirtualGamePiece[,] board_field, int minLength = 3)
```

Parameters

`x` [int](#)

`y` [int](#)

`board_field` [VirtualGamePiece\[,\]](#)

`minLength` [int](#)

Returns

[bool](#)

IsWithinBounds(int, int, VirtualGamePiece[,])

Checks if a given position is within the bounds of the game board.

```
private bool IsWithinBounds(int x, int y, VirtualGamePiece[,] board)
```

Parameters

x [int](#)

X-coordinate of the position to check.

y [int](#)

Y-coordinate of the position to check.

board [VirtualGamePiece](#)[,]

The game board.

Returns

[bool](#)

True if the position is within bounds; otherwise, false.

MakeCopyBoard(VirtualGamePiece[,])

Creates a copy of the given game board.

```
public VirtualGamePiece[,] MakeCopyBoard(VirtualGamePiece[,] board_field)
```

Parameters

board_field [VirtualGamePiece](#)[,]

The game board to be copied.

Returns

[VirtualGamePiece](#)[,]

A new copy of the provided game board.

SelectMoveByBiggestGemEffect(List<Move>, GemActiveEffectType)

Selects the best move based on the highest gem effect value of a specified type.

```
private Move SelectMoveByBiggestGemEffect(List<Move> moves, GemActiveEffectType type)
```

Parameters

moves [List](#)<[Move](#)>

List of potential moves.

type [GemActiveEffectType](#)

The type of gem effect to consider for selection.

Returns

[Move](#)

The move with the highest gem effect value of the specified type.

SelectMoveByBiggestGemStatusEffect(List<Move>, GemStatusEffectType)

Selects the best move based on the highest gem status effect value of a specified type.

```
private Move SelectMoveByBiggestGemStatusEffect(List<Move> moves, GemStatusEffectType type)
```

Parameters

moves [List](#)<[Move](#)>

List of potential moves.

type [GemStatusEffectType](#)

The type of gem status effect to consider for selection.

Returns

[Move](#)

The move with the highest gem status effect value of the specified type.

ShuffleBoard(VirtualGamePiece[,])

Shuffles the positions of virtual game pieces on the board.

```
public VirtualGamePiece[,] ShuffleBoard(VirtualGamePiece[,] board)
```

Parameters

board [VirtualGamePiece\[\]](#)

The virtual game board to be shuffled.

Returns

[VirtualGamePiece\[\]](#)

The shuffled virtual game board.

Class VirtualGamePiece

Namespace: [TaleOfTiles.AI](#)

Assembly: Assembly-CSharp.dll

Represents a virtual representation of a game piece.

```
public class VirtualGamePiece
```

Inheritance

[object](#) ← VirtualGamePiece

Constructors

VirtualGamePiece()

Initializes a new instance of the VirtualGamePiece class.

```
public VirtualGamePiece()
```

VirtualGamePiece(VirtualGamePiece)

Initializes a new instance of the VirtualGamePiece class as a copy of another virtual game piece.

```
public VirtualGamePiece(VirtualGamePiece other)
```

Parameters

other [VirtualGamePiece](#)

The other VirtualGamePiece to copy data from.

VirtualGamePiece(GamePiece)

Initializes a new instance of the VirtualGamePiece class using a real game piece.

```
public VirtualGamePiece(GamePiece game_piece)
```

Parameters

game_piece [GamePiece](#)

The real game piece to copy data from.

Fields

_effect_status

Status effect type of this game piece.

```
public GemStatusEffectType _effect_status
```

Field Value

[GemStatusEffectType](#)

assosiated_game_piece

The actual game piece associated with this virtual representation.

```
public GamePiece assosiated_game_piece
```

Field Value

[GamePiece](#)

effect_duration

Duration of the effect applied by this game piece.

```
public int effect_duration
```

Field Value

[int](#)

gem_effects

List of active gem effects associated with this game piece.

```
public List<GemActiveEffect> gem_effects
```

Field Value

[List](#) <[GemActiveEffect](#)>

gem_status_effects

List of status effects associated with this game piece.

```
public List<GemStatusEffect> gem_status_effects
```

Field Value

[List](#) <[GemStatusEffect](#)>

matchValue

Match value of the game piece, determining its type or category.

```
public string matchValue
```

Field Value

[string](#)

super_combos

List of super combos associated with this game piece.

```
public List<GemSuperCombo> super_combos
```

Field Value

[List](#) <[GemSuperCombo](#)>

tier

Tier of the gem item associated with this game piece.

```
public GemItemTier tier
```

Field Value

[GemItemTier](#)

touch_damage

Damage caused by touching this game piece.

```
public float touch_damage
```

Field Value

[float](#)

xIndex

X index of the game piece on the board.

```
public int xIndex
```

Field Value

[int](#)

yIndex

X index of the game piece on the board.

`public int yIndex`

Field Value

[int](#)

Methods

Equals(object)

Determines whether the specified object is equal to the current object.

`public override bool Equals(object obj)`

Parameters

`obj` [object](#)

The object to compare with the current object.

Returns

[bool](#)

true if the specified object is equal to the current object; otherwise, false.

GetHashCode()

Serves as the default hash function.

`public override int GetHashCode()`

Returns

[int↗](#)

A hash code for the current object.

SetCoord(int, int)

Sets the coordinates of the game piece.

```
public void SetCoord(int x, int y)
```

Parameters

x [int↗](#)

The x-coordinate to set.

y [int↗](#)

The y-coordinate to set.

Namespace TaleOfTiles.Arena

Classes

[BattleArenaBackgroundImage](#)

A class that represents the background image of a battle arena.

[BattleArenaManager](#)

This class handles the functionality of the battle arena in the game. It inherits from the Singleton class to ensure only one instance of this class exists.

[BattleArenaTopButtons](#)

The BattleArenaTopButtons class manages the top buttons in the battle arena.

[BattleButtonsController](#)

A class that controls the buttons in a battle.

[BoardDeadlock](#)

A class that checks for board deadlock in a match-3 game.

[BoardManager](#)

Class that manages the game board and related functions.

[BoardShuffler](#)

Provides methods to shuffle game pieces on the board.

[Bomb](#)

Represents a bomb in the game, which is just a GamePiece with a BombType.

[BubbleText](#)

Handles the display and animation of bubble text in the game.

[Collectible](#)

Represents a collectible in the game. It is a type of GamePiece with no match value. Collectibles can be cleared by a bomb or by reaching the bottom of the screen.

[CombatManager](#)

This class handles the functionality of the combat in the game. It inherits from the Singleton class to ensure only one instance of this class exists.

[CombinationScoreManager](#)

A static class that handles the calculation of scores and effects for combinations in the game.

[GamePiece](#)

Script to handle the functionality and display of a game piece.

[HealthBar](#)

Manages the health bar UI for a player character.

[PlayerBattleCardUI](#)

Manages the UI for a player's battle card, including health, available moves, and status effects.

[Probability](#)

Class for probability-related operations.

[StatusBarController](#)

Represents an individual status bar element in the UI.

[SwitchBoardButton](#)

A class for a switch board button in a game.

[Tile](#)

Represents a tile in a game board.

[Utils](#)

Utility class providing various helper methods.

Enums

[AnimationType](#)

Enum for the types of animations that can be performed.

[BombType](#)

Enum for the various bombs available in the game.

[CombatManager.GameState](#)

The possible states of the game.

[InterpType](#)

Enum for the possible interpolation types.

[MatchValue](#)

Enum for the possible match values.

[StatusBarController](#)

Defines types of status bar elements.

[TileType](#)

Defines the types of tiles.

Delegates

[Utils.GetColorDelegate](#)

[UtilsSetColorDelegate](#)

[Utils.todo](#)

Enum AnimationType

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Enum for the types of animations that can be performed.

```
public enum AnimationType
```

Fields

Bubble = 0

Resize = 1

Class BattleArenaBackgroundImage

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

A class that represents the background image of a battle arena.

```
public class BattleArenaBackgroundImage : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BattleArenaBackgroundImage

Remarks

This class is a MonoBehaviour, and thus can be attached to a GameObject in the Unity3D game engine.

Methods

Start()

This method is called before the first frame update.

```
private void Start()
```

Remarks

Use this for initialization. This method is called once per instance (i.e., clone) of a script is enabled just before any of the Update methods are called the first time.

Update()

This method is called once per frame.

```
private void Update()
```

Remarks

Use this for regular updates that are happening frame by frame. Most game code will go here.

Class BattleArenaManager

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

This class handles the functionality of the battle arena in the game. It inherits from the Singleton class to ensure only one instance of this class exists.

```
public class BattleArenaManager : Singleton<BattleArenaManager>
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ←
[Singleton](#)<[BattleArenaManager](#)> ← BattleArenaManager

Inherited Members

[Singleton](#)<[BattleArenaManager](#)>.m_instance , [Singleton](#)<[BattleArenaManager](#)>.Instance ,
[Singleton](#)<[BattleArenaManager](#)>.Awake() , [SingletonBase](#).AllSingletons

Fields

defeat_mult

The multiplier for the defeat reward.

```
private float defeat_mult
```

Field Value

[float](#)

level_config

The configuration of the level.

```
public LevelObject level_config
```

Field Value

[LevelObject](#)

level_enemy

The enemy of the level.

```
public EnemyObject level_enemy
```

Field Value

[EnemyObject](#)

player

The player character.

```
public PlayerCharacter player
```

Field Value

[PlayerCharacter](#)

Methods

DefeatRewardExp()

Returns the experience reward for a defeat.

```
public int DefeatRewardExp()
```

Returns

[int↗](#)

DefeatRewardGold()

Returns the gold reward for a defeat.

```
public int DefeatRewardGold()
```

Returns

[int](#)

GivePlayerReward(bool)

Gives the player a reward based on the outcome of the battle.

```
public void GivePlayerReward(bool victory)
```

Parameters

victory [bool](#)

LoadBattle(GameObject, bool)

Loads the battle.

```
public void LoadBattle(GameObject caller = null, bool reloading = true)
```

Parameters

caller GameObject

reloading [bool](#)

LoadBattleCoroutine(GameObject, bool)

Coroutine to load the battle.

```
internal IEnumerator LoadBattleCoroutine(GameObject caller = null, bool reloading = true)
```

Parameters

caller GameObject

reloading [bool](#)

Returns

[IEnumerator](#)

LoadEnemy()

Loads the enemy for the battle.

```
private void LoadEnemy()
```

Prepare()

Prepares the battle arena manager.

```
public override IEnumerator Prepare()
```

Returns

[IEnumerator](#)

SetupLevel(LevelObject)

Sets up the level for the battle.

```
public void SetupLevel(LevelObject level)
```

Parameters

`level` [LevelObject](#)

VictoryRewardExp()

Returns the experience reward for a victory.

```
public int VictoryRewardExp()
```

Returns

[int↗](#)

VictoryRewardGold()

Returns the gold reward for a victory.

```
public int VictoryRewardGold()
```

Returns

[int↗](#)

Class BattleArenaTopButtons

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

The BattleArenaTopButtons class manages the top buttons in the battle arena.

```
public class BattleArenaTopButtons : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BattleArenaTopButtons

Fields

surrender_button

```
public Button surrender_button
```

Field Value

Button

Gets or sets the surrender button.

Methods

Start()

Start is called before the first frame update. It removes all existing listeners from the surrender button and adds a new one which triggers defeat in the CombatManager when clicked.

```
private void Start()
```

Class BattleButtonsController

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

A class that controls the buttons in a battle.

```
public class BattleButtonsController : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BattleButtonsController

Fields

cards_button

The button that opens the cards panel.

```
public Button cards_button
```

Field Value

Button

end_turn_button

The button that ends the current turn.

```
public Button end_turn_button
```

Field Value

Button

human_board

The board manager for the human player.

```
public BoardManager human_board
```

Field Value

[BoardManager](#)

show_enemy_board

The button that shows the enemy board.

```
public Button show_enemy_board
```

Field Value

Button

Methods

Start()

This method is called before the first frame update.

```
private void Start()
```

Remarks

Use this for initialization. This method is called once per instance (i.e., clone) of a script is enabled just before any of the Update methods are called the first time.

Update()

This method is called once per frame.

```
private void Update()
```

Remarks

Use this for regular updates that are happening frame by frame. Most game code will go here.

Class BoardDeadlock

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

A class that checks for board deadlock in a match-3 game.

```
public class BoardDeadlock : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BoardDeadlock

Methods

GetMinimumMatches(List<GamePiece>, int)

Returns a sub-list of game pieces that match the minimum required matches.

```
private List<GamePiece> GetMinimumMatches(List<GamePiece> gamePieces, int minForMatch = 2)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The list of game pieces to check.

minForMatch [int](#)

The minimum number of matches required.

Returns

[List](#)<[GamePiece](#)>

A sub-list of game pieces that match the minimum required matches.

GetNeighbors(GamePiece[,], int, int)

Returns a list of neighboring game pieces at the specified coordinate.

```
private List<GamePiece> GetNeighbors(GamePiece[,] allPieces, int x, int y)
```

Parameters

allPieces [GamePiece\[,\]](#)

The 2D array of all game pieces.

x [int](#)

The x-coordinate.

y [int](#)

The y-coordinate.

Returns

[List](#) <[GamePiece](#)>

A list of neighboring game pieces.

GetRowOrColumnList(GamePiece[,], int, int, int, bool)

Returns a list of game pieces in a row or column at the given coordinate.

```
private List<GamePiece> GetRowOrColumnList(GamePiece[,] allPieces, int x, int y, int  
listLength = 3, bool checkRow = true)
```

Parameters

allPieces [GamePiece\[,\]](#)

The 2D array of all game pieces.

x [int](#)

The x-coordinate.

y [int](#)

The y-coordinate.

listLength [int](#)

The length of the list to check.

checkRow [bool](#)

Flag indicating whether to check a row or column.

Returns

[List](#)<[GamePiece](#)>

A list of game pieces in the specified row or column.

HasMoveAt(GamePiece[,], int, int, int, bool)

Checks if there is a possible move at the specified position on the board.

```
private bool HasMoveAt(GamePiece[,] allPieces, int x, int y, int listLength = 3, bool checkRow = true)
```

Parameters

allPieces [GamePiece](#)[]

The 2D array of all game pieces.

x [int](#)

The x-coordinate.

y [int](#)

The y-coordinate.

listLength [int](#)

The length of the list to check.

checkRow [bool](#)

Flag indicating whether to check a row or column.

Returns

[bool](#)

True if there is a possible move, false otherwise.

IsDeadlocked(GamePiece[,], int)

Checks if the board is deadlocked, i.e. if there are no more moves available.

```
public bool IsDeadlocked(GamePiece[,] allPieces, int listLength = 3)
```

Parameters

[allPieces](#) [GamePiece](#)[,]

The 2D array of all game pieces.

[listLength](#) [int](#)

The length of the list to check.

Returns

[bool](#)

True if the board is deadlocked, false otherwise.

Class BoardManager

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Class that manages the game board and related functions.

```
public class BoardManager : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BoardManager

Fields

_score

The current score of the game.

```
public float _score
```

Field Value

[float](#)

adjacentBombPrefabs

Array of GameObjects representing the prefabs for adjacent bombs.

```
public GameObject[] adjacentBombPrefabs
```

Field Value

GameObject[]

allGamePieces

A 2D array of all the game pieces.

```
public GamePiece[,] allGamePieces
```

Field Value

[GamePiece\[,\]](#)

allTilesArray

A 2D array of all the tiles present in the game.

```
public Tile[,] allTilesArray
```

Field Value

[Tile\[,\]](#)

baseGemPrefab

Base gem prefab to be used for creating gems.

```
public GameObject baseGemPrefab
```

Field Value

GameObject

chameleon_activated

Flag to check if the chameleon is currently activated.

```
public bool chameleon_activated
```

Field Value

[bool](#)

chanceForCollectible

Chance for a top-row tile to get a collectible.

```
[Range(0, 1)]  
public float chanceForCollectible
```

Field Value

[float](#)

clickedTile

The currently clicked tile by the user.

```
public Tile clickedTile
```

Field Value

[Tile](#)

collapseTime

The amount of time required for the board to collapse.

```
public float collapseTime
```

Field Value

[float](#)

collectibleCount

Current number of collectibles on the board.

```
public int collectibleCount
```

Field Value

[int](#)

collectiblePrefabs

Array of GameObjects representing the collectible prefabs.

```
public GameObject[] collectiblePrefabs
```

Field Value

GameObject[]

colorBombPrefab

GameObject representing the color bomb prefab.

```
public GameObject colorBombPrefab
```

Field Value

GameObject

columnBombPrefabs

Array of GameObjects representing the prefabs for column bombs.

```
public GameObject[] columnBombPrefabs
```

Field Value

GameObject[]

combo_counter

Text component for displaying combo count.

```
public Text combo_counter
```

Field Value

Text

debug_board_fill

Flag to debug board fill.

```
public bool debug_board_fill
```

Field Value

[bool](#)

fillMoveTime

Time used to fill the board.

```
public float fillMoveTime
```

Field Value

[float](#)

fillYOffset

Y Offset used to make the pieces "fall" into place to fill the Board.

```
public int fillYOffset
```

Field Value

[int](#)

gemSize

Size of a gem on the board.

```
private Vector2 gemSize
```

Field Value

Vector2

gemsGrid

Game object representing the grid of gems.

```
public GameObject gemsGrid
```

Field Value

GameObject

gemsPrefabs

List of GameObjects representing the gem prefabs.

```
public List<GameObject> gemsPrefabs
```

Field Value

[List](#)<GameObject>

height

The height of the game board.

```
public int height
```

Field Value

[int](#)

isRefilling

Flag indicating whether the board is refilling.

```
public bool isRefilling
```

Field Value

[bool](#)

m_boardDeadlock

Instance of the BoardDeadlock class.

```
private BoardDeadlock m_boardDeadlock
```

Field Value

[BoardDeadlock](#)

m_boardShuffler

Instance of the BoardShuffler class.

```
private BoardShuffler m_boardShuffler
```

Field Value

m_clickedTileBomb

Reference to a bomb created on the clicked tile.

```
private GameObject m_clickedTileBomb
```

Field Value

GameObject

m_particleManager

Instance of the ParticleManager class for managing particle effects.

```
private ParticleManager m_particleManager
```

Field Value

[ParticleManager](#)

m_playerInputEnabled

Flag indicating whether the player's input is enabled.

```
public bool m_playerInputEnabled
```

Field Value

[bool](#) ↗

m_scoreMultiplier

The current score multiplier, based on the number of chain reactions caused.

```
private int m_scoreMultiplier
```

Field Value

[int](#)

m_targetTileBomb

Reference to a bomb created on the target tile.

```
private GameObject m_targetTileBomb
```

Field Value

[GameObject](#)

maxCollectibles

Maximum number of collectibles allowed per board.

```
public int maxCollectibles
```

Field Value

[int](#)

player

PlayerCharacter object representing the player.

```
public PlayerCharacter player
```

Field Value

[PlayerCharacter](#)

rowBombPrefabs

Array of GameObjects representing the prefabs for row bombs.

```
public GameObject[] rowBombPrefabs
```

Field Value

GameObject[]

score_counter

Text component for displaying score.

```
public Text score_counter
```

Field Value

Text

static_random

System.Random instance for generating random numbers.

```
private Random static_random
```

Field Value

[Random](#) ↗

swapTime

The amount of time required to swap game pieces between the target and clicked tile.

```
public float swapTime
```

Field Value

[float](#)

targetTile

The tile that is being targeted by the player.

`public Tile targetTile`

Field Value

[Tile](#)

tilesGrid

GameObject representing the grid of tiles.

`public GameObject tilesGrid`

Field Value

GameObject

width

The width of the game board.

`public int width`

Field Value

[int](#)

Properties

score

Property to get and set the score. Updates the score_counter text on set.

```
public float score { get; set; }
```

Property Value

[float](#)

Methods

ActivateBomb(GameObject)

Activates a bomb in the game board and treats it as a normal GamePiece.

```
private void ActivateBomb(GameObject bomb)
```

Parameters

bomb GameObject

The bomb to be activated.

BreakTileAt(List<GamePiece>)

Damages breakable tiles corresponding to a list of game pieces.

```
private void BreakTileAt(List<GamePiece> gamePieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The list of game pieces whose positions should have their tiles damaged.

BreakTileAt(int, int)

Damages a breakable tile at the specified position.

```
private void BreakTileAt(int x, int y)
```

Parameters

x [int](#)

The x-coordinate of the tile's position.

y [int](#)

The y-coordinate of the tile's position.

CanAddCollectible()

Checks if a collectible can be added to the game board. Currently always returns false.

```
private bool CanAddCollectible()
```

Returns

[bool](#)

CheckDeadLock()

Checks if the game board is in a deadlock situation.

```
[ContextMenu("Check Deadlock")]
public bool CheckDeadLock()
```

Returns

[bool](#)

True if the board is in deadlock, false otherwise.

ClearAndCollapseRoutine(List<GamePiece>)

Coroutine to clear GamePieces from the Board and collapse any empty spaces.

```
private IEnumerator ClearAndCollapseRoutine(List<GamePiece> gamePieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The game pieces to clear from the board.

Returns

[IEnumerator](#)

ClearAndRefillBoard(List<GamePiece>)

Clears and refills the Board.

```
private void ClearAndRefillBoard(List<GamePiece> gamePieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The game pieces to clear from the board.

ClearAndRefillBoardRoutine(List<GamePiece>)

Coroutine to clear GamePieces and collapse empty spaces, then refill the Board.

```
private IEnumerator ClearAndRefillBoardRoutine(List<GamePiece> gamePieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The game pieces to clear from the board.

Returns

[IEnumerator](#)

ClearBoard()

Clears the board by iterating over all game pieces and clearing them.

```
public void ClearBoard()
```

ClearPieceAt(List<GamePiece>, List<Combination>)

Clears a list of game pieces and scores combinations of pieces.

```
public IEnumerator ClearPieceAt(List<GamePiece> gamePieces, List<Combination> combinations)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The list of game pieces to clear.

combinations [List](#)<[Combination](#)>

The list of combinations to score.

Returns

[IEnumerator](#)

An IEnumerator to be used in a coroutine.

ClearPieceAt(List<GamePiece>, List<GamePiece>)

Clears a list of game pieces.

```
private IEnumerator ClearPieceAt(List<GamePiece> gamePieces, List<GamePiece> bombedPieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The list of game pieces to clear.

bombedPieces [List](#)<[GamePiece](#)>

The list of game pieces destroyed by bombs.

Returns

[IEnumerator](#)

An IEnumerator to be used in a coroutine.

ClearPieceAt(int, int)

Clears the game piece at the specified position.

```
private void ClearPieceAt(int x, int y)
```

Parameters

x [int](#)

The x-coordinate of the position to clear.

y [int](#)

The y-coordinate of the position to clear.

ClickTile(Tile)

Sets the clicked tile to the specified tile.

```
public void ClickTile(Tile gem)
```

Parameters

gem [Tile](#)

The tile that was clicked.

CollapseColumn(List<int>)

Compresses all columns given a list of column indices.

```
private List<GamePiece> CollapseColumn(List<int> columnsToCollapse)
```

Parameters

columnsToCollapse [List](#)<[int](#)>

The list of column indices to compress.

Returns

[List](#)<[GamePiece](#)>

A list of GamePiece objects that were moved during the compression.

CollapseColumn(List<GamePiece>)

Compresses all columns given a list of GamePieces.

```
private List<GamePiece> CollapseColumn(List<GamePiece> gamePieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The list of game pieces whose columns should be compressed.

Returns

[List](#)<[GamePiece](#)>

A list of GamePiece objects that were moved during the compression.

CollapseColumn(int, float)

Compresses a given column to remove any empty Tile spaces.

```
private List<GamePiece> CollapseColumn(int column, float collapseTime = 0.1)
```

Parameters

column [int](#)

The column to compress.

collapseTime [float](#)

The time it takes for the compression to occur.

Returns

[List](#)<[GamePiece](#)>

A list of GamePiece objects that were moved during the compression.

CreateGemsPrefabs()

Creates gem prefabs based on the player's active skills.

```
public void CreateGemsPrefabs()
```

DropBomb(int, int, Vector2, List<GamePiece>)

Drops a bomb at the specified position on the game board, given a list of matching game pieces.

```
private GameObject DropBomb(int x, int y, Vector2 swapDirection, List<GamePiece> gamePieces)
```

Parameters

x [int](#)

The x-coordinate of the position.

y [int](#)

The y-coordinate of the position.

swapDirection Vector2

The direction of the swap that created the match.

gamePieces [List](#)<[GamePiece](#)>

The list of matching game pieces.

Returns

GameObject

The Bomb object created at the specified position.

EndTurn()

Ends the turn by calling EndTurn on all game pieces.

```
public void EndTurn()
```

FillBoard(int, float)

Fills the game board with game pieces, ensuring no matches are created during the fill.

```
private void FillBoard(int falseYOffset = 0, float moveTime = 0.1)
```

Parameters

falseYOffset [int](#)

The Y offset for filling the board.

moveTime [float](#)

The time taken to move a piece.

FillBoardFromList(List<GamePiece>)

Fills the game board using a known list of game pieces, ensuring no matches are created during the fill.

```
private void FillBoardFromList(List<GamePiece> gamePieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The list of game pieces to use.

FillRandomCollectibleAt(int, int, int, float)

Creates a random collectible at the specified position on the game board.

```
private GamePiece FillRandomCollectibleAt(int x, int y, int falseYOffset = 0, float moveTime = 0.1)
```

Parameters

x [int](#)

The x-coordinate of the position.

y [int](#)

The y-coordinate of the position.

falseYOffset [int](#)

The Y offset for creating the collectible.

moveTime [float](#)

The time taken to move a piece.

Returns

[GamePiece](#)

The GamePiece object created at the specified position.

FillRandomGamePieceAt(int, int, int, float)

Creates a random game piece at the specified position on the game board.

```
private GamePiece FillRandomGamePieceAt(int x, int y, int falseYOffset = 0, float moveTime  
= 0.1)
```

Parameters

x [int](#)

The x-coordinate of the position.

y [int](#)

The y-coordinate of the position.

falseYOffset [int](#)

The Y offset for creating the game piece.

moveTime [float](#)

The time taken to move a piece.

Returns

[GamePiece](#)

The GamePiece object created at the specified position.

FindAllCollectibles()

Finds all Collectibles in the Board.

```
private List<GamePiece> FindAllCollectibles()
```

Returns

[List](#)<[GamePiece](#)>

A list of GamePiece objects that are collectibles.

FindAllMatchValue(string)

Finds all GamePieces on the Board with a certain MatchValue.

```
private List<GamePiece> FindAllMatchValue(string mValue)
```

Parameters

mValue [string](#)

The match value to search for.

Returns

[List](#)<[GamePiece](#)>

A list of GamePiece objects with the specified match value.

FindAllMatches()

Finds all matches in the game Board.

```
private List<GamePiece> FindAllMatches()
```

Returns

[List](#)<[GamePiece](#)>

A list of all GamePiece objects that are part of a match.

FindAllMoves()

Finds all possible moves and matches afterwards.

```
public List<Move> FindAllMoves()
```

Returns

[List](#)<[Move](#)>

A list of all possible Move objects.

FindCollectiblesAt(int, bool)

Finds all Collectibles at a certain row.

```
private List<GamePiece> FindCollectiblesAt(int row, bool clearedAtBottomOnly = false)
```

Parameters

row [int](#)

The row to search for collectibles.

clearedAtBottomOnly [bool](#)

If true, only returns collectibles that can be cleared at the bottom.

Returns

[List](#)<[GamePiece](#)>

A list of GamePiece objects that are collectibles in the specified row.

FindGamePieceByMatchValue(GameObject[], string)

Finds a GamePiece prefab with a given match value from an array of prefabs.

```
private GameObject FindGamePieceByMatchValue(GameObject[] gamePiecePrefabs,  
string matchValue)
```

Parameters

gamePiecePrefabs `GameObject[]`

The array of prefabs to search from.

matchValue `string`

The match value to search for.

Returns

`GameObject`

A `GameObject` from the prefabs that has the specified match value.

FindHorizontalMatches(int, int, int)

Finds all horizontal matches given a position (x,y) in the Board.

```
private List<GamePiece> FindHorizontalMatches(int startX, int startY, int minLength = 3)
```

Parameters

startX `int`

The x-coordinate of the starting position.

startY `int`

The y-coordinate of the starting position.

minLength `int`

The minimum length of a match.

Returns

`List<GamePiece>`

A list of matching GamePiece objects if a match is found, null otherwise.

FindMatchValue(List<GamePiece>)

Given a list of GamePieces, returns the first valid MatchValue found.

```
private string FindMatchValue(List<GamePiece> gamePieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The list of game pieces to search from.

Returns

[string](#)

The match value of the first valid game piece in the list.

FindMatches(int, int, Vector2, int)

Finds matches of game pieces starting from a position and searching in a specific direction.

```
private List<GamePiece> FindMatches(int startX, int startY, Vector2 searchDirection, int minLength = 3)
```

Parameters

startX [int](#)

The x-coordinate of the starting position.

startY [int](#)

The y-coordinate of the starting position.

searchDirection Vector2

The direction to search in.

minLength [int](#)

The minimum length of a match.

Returns

[List](#)<[GamePiece](#)>

A list of matching GamePiece objects if a match is found, null otherwise.

FindMatchesAt(List<GamePiece>, int)

Finds all matches given a list of GamePieces.

```
private List<GamePiece> FindMatchesAt(List<GamePiece> gamePieces, int minLength = 3)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The list of game pieces to search from.

minLength [int](#)

The minimum length of a match.

Returns

[List](#)<[GamePiece](#)>

A list of matching GamePiece objects if a match is found.

FindMatchesAt(int, int, int)

Finds horizontal and vertical matches at a position (x,y) in the Board.

```
private List<GamePiece> FindMatchesAt(int x, int y, int minLength = 3)
```

Parameters

x [int](#)

The x-coordinate of the position.

y [int](#)

The y-coordinate of the position.

minLength [int](#)

The minimum length of a match.

Returns

[List](#)<[GamePiece](#)>

A list of matching GamePiece objects if a match is found.

FindVerticalMatches(int, int, int)

Finds all vertical matches given a position (x,y) in the Board.

```
private List<GamePiece> FindVerticalMatches(int startX, int startY, int minLength = 3)
```

Parameters

startX [int](#)

The x-coordinate of the starting position.

startY [int](#)

The y-coordinate of the starting position.

minLength [int](#)

The minimum length of a match.

Returns

[List](#)<[GamePiece](#)>

A list of matching GamePiece objects if a match is found, null otherwise.

GetAdjacentPieces(int, int, int)

Gets all GamePieces adjacent to a position (x,y).

```
private List<GamePiece> GetAdjacentPieces(int x, int y, int offset = 1)
```

Parameters

x [int](#)

The x-coordinate of the position.

y [int](#)

The y-coordinate of the position.

offset [int](#)

The distance from the position to consider as adjacent.

Returns

[List](#)<[GamePiece](#)>

A list of GamePiece objects adjacent to the specified position.

GetBombedPieces(List<GamePiece>)

Given a list of GamePieces, returns a new List of GamePieces that would be destroyed by bombs from the original list.

```
private List<GamePiece> GetBombedPieces(List<GamePiece> gamePieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The original list of game pieces.

Returns

[List](#)<[GamePiece](#)>

A list of GamePiece objects that would be destroyed by bombs from the original list.

GetColumnPieces(int)

Gets a list of GamePieces in a specified column.

```
private List<GamePiece> GetColumnPieces(int column)
```

Parameters

column [int](#)

The column to retrieve game pieces from.

Returns

[List](#)<[GamePiece](#)>

A list of GamePiece objects in the specified column.

GetColumns(List<GamePiece>)

Given a List of GamePieces, return a list of columns by index number.

```
private List<int> GetColumns(List<GamePiece> gamePieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The list of game pieces to find columns from.

Returns

[List](#)<[int](#)>

A list of column indices that the game pieces are in.

GetGamePiece(Tile)

Retrieves the GamePiece associated with a Tile.

```
public GamePiece GetGamePiece(Tile tile)
```

Parameters

tile [Tile](#)

The tile to get the game piece for.

Returns

[GamePiece](#)

The GamePiece associated with the provided tile.

GetRandomCollectible()

Retrieves a random collectible from the set of collectible prefabs.

```
private GameObject GetRandomCollectible()
```

Returns

[GameObject](#)

A GameObject representing a random collectible.

GetRandomExistingGamePiece()

Retrieves a random existing game piece from the game board.

```
public GamePiece GetRandomExistingGamePiece()
```

Returns

[GamePiece](#)

A GamePiece object randomly chosen from the existing game pieces on the board.

GetRandomExistingGamePieceWithStatus(GemStatusEffectType)

Retrieves a random existing game piece with a specific status from the game board.

```
public GamePiece GetRandomExistingGamePieceWithStatus(GemStatusEffectType type)
```

Parameters

type [GemStatusEffectType](#)

The status type to filter game pieces by.

Returns

[GamePiece](#)

A GamePiece object randomly chosen from the existing game pieces on the board with the specified status.

GetRandomExistingGamePieces(float)

Retrieves a list of random existing game pieces from the game board.

```
public List<GamePiece> GetRandomExistingGamePieces(float part_of_board)
```

Parameters

part_of_board [float](#)

The fraction of the board to retrieve game pieces from.

Returns

[List](#) <[GamePiece](#)>

A list of GamePiece objects randomly chosen from the existing game pieces on the board.

GetRandomGamePiece()

Returns a random game piece from the list of gem prefabs, with consideration for gem rarity.

```
private GameObject GetRandomGamePiece()
```

Returns

GameObject

A GameObject representing a random game piece.

GetRandomObject(GameObject[])

Returns a random object from an array of GameObjects.

```
private GameObject GetRandomObject(GameObject[] objectArray)
```

Parameters

objectArray GameObject[]

Array of GameObjects to choose from.

Returns

GameObject

A random GameObject from the provided array.

GetRowPieces(int)

Gets a list of GamePieces in a specified row.

```
private List<GamePiece> GetRowPieces(int row)
```

Parameters

row [int](#)

The row to retrieve game pieces from.

Returns

[List](#)<[GamePiece](#)>

A list of GamePiece objects in the specified row.

GetTile(GamePiece)

Retrieves the Tile associated with a GamePiece.

```
public Tile GetTile(GamePiece piece)
```

Parameters

piece [GamePiece](#)

The game piece to get the tile for.

Returns

[Tile](#)

The Tile associated with the provided game piece.

HasMatchOnFill(int, int, int)

Checks if a match exists in the given direction starting from the position specified by x and y.

```
private bool HasMatchOnFill(int x, int y, int minLength = 3)
```

Parameters

x [int](#)

The x-coordinate of the position to check from.

y [int](#)

The y-coordinate of the position to check from.

minLength [int](#)

The minimum length of a match.

Returns

[bool](#)

True if a match exists, false otherwise.

InitTiles()

Initializes the tiles on the game board.

```
private void InitTiles()
```

IsCollapsed(List<GamePiece>)

Checks if the GamePieces have reached their destination positions on collapse.

```
private bool IsCollapsed(List<GamePiece> gamePieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The list of game pieces to check.

Returns

[bool](#)

True if all game pieces have reached their destination, false otherwise.

IsColorBomb(GamePiece)

Checks if a GamePiece is a color bomb.

```
private bool IsColorBomb(GamePiece gamePiece)
```

Parameters

gamePiece [GamePiece](#)

The game piece to check.

Returns

[bool](#)

True if the GamePiece is a color bomb, false otherwise.

IsCornerMatch(List<GamePiece>)

Checks if a list of matching GamePieces forms an L shaped match.

```
private bool IsCornerMatch(List<GamePiece> gamePieces)
```

Parameters

gamePieces [List](#)<[GamePiece](#)>

The list of game pieces to check.

Returns

[bool](#)

True if the game pieces form an L shaped match, false otherwise.

IsNextTo(Tile, Tile)

Checks if two tiles are adjacent to each other.

```
private bool IsNextTo(Tile start, Tile end)
```

Parameters

start [Tile](#)

The first tile.

end [Tile](#)

The second tile.

Returns

[bool](#) ↗

True if the tiles are adjacent, false otherwise.

IsWithinBounds(int, int)

Checks if a position is within the game board's bounds.

```
private bool IsWithinBounds(int x, int y)
```

Parameters

x [int](#) ↗

The x-coordinate of the position.

y [int](#) ↗

The y-coordinate of the position.

Returns

[bool](#) ↗

True if the position is within the game board's bounds, false otherwise.

MakeBomb(GameObject, int, int)

Creates a bomb at the specified position on the game board.

```
private GameObject MakeBomb(GameObject prefab, int x, int y)
```

Parameters

prefab GameObject

The prefab to create the bomb from.

x [int](#)

The x-coordinate of the position.

y [int](#)

The y-coordinate of the position.

Returns

GameObject

The Bomb object created at the specified position.

MakeGamePiece(GameObject, int, int, int, float)

Creates a game piece at a specific location on the game board.

```
private void MakeGamePiece(GameObject prefab, int x, int y, int falseYOffset = 0, float moveTime = 0.1)
```

Parameters

prefab GameObject

The GameObject to instantiate as a game piece.

x [int](#)

The x-coordinate of the location.

y [int](#)

The y-coordinate of the location.

falseYOffset [int](#)

The Y offset for creating the game piece.

moveTime [float](#)

The time taken to move a piece.

PlaceGamePiece(GamePiece, int, int)

Places a game piece at a specific location on the game board.

```
public void PlaceGamePiece(GamePiece gamePiece, int x, int y)
```

Parameters

gamePiece [GamePiece](#)

The game piece to place.

x [int](#)

The x-coordinate of the location.

y [int](#)

The y-coordinate of the location.

RefillRoutine()

Refills the game board.

```
private IEnumerator RefillRoutine()
```

Returns

[IEnumerator](#)

ReleaseTile()

Releases the clicked and target tiles and switches their positions if both are set.

```
public void ReleaseTile()
```

ReloadBoard()

Reloads the game board by creating gems, clearing the board, initializing tiles, and filling the board.

```
public void ReloadBoard()
```

RemoveCollectibles(List<GamePiece>)

Removes any Collectibles from the list if they can cleared by Bombs.

```
private List<GamePiece> RemoveCollectibles(List<GamePiece> bombedPieces)
```

Parameters

bombedPieces [List](#)<[GamePiece](#)>

The list of game pieces affected by bombs.

Returns

[List](#)<[GamePiece](#)>

A list of GamePiece objects after removing collectibles.

RestartPartcilesOnGamePiece()

Restarts particles on all game pieces.

```
public void RestartPartcilesOnGamePiece()
```

RestartParticlesWithEffect(GemStatusEffectType)

Restarts particle effects with a specific effect status.

```
public void RestartParticlesWithEffect(GemStatusEffectType effect_status)
```

Parameters

effect_status [GemStatusEffectType](#)

The effect status to restart particles for.

ScoreMove()

Handles the scoring of a move and swaps turns if necessary.

```
public void ScoreMove()
```

SelectNextTile(Tile)

Sets the target tile to the specified tile if it is adjacent to the clicked tile.

```
public void SelectNextTile(Tile gem)
```

Parameters

gem [Tile](#)

The tile to set as the target.

ShuffleBoard()

Starts the coroutine to shuffle the game board.

```
public void ShuffleBoard()
```

ShuffleBoardRoutine()

Coroutine to shuffle non-bomb and non-collectible GamePieces.

```
private IEnumerator ShuffleBoardRoutine()
```

Returns

[IEnumerator](#)

Start()

Method called at the start of the game. Initializes components.

```
private void Start()
```

SwitchTiles(Tile, Tile)

Swaps the positions of two tiles.

```
public void SwitchTiles(Tile clickedTile, Tile targetTile)
```

Parameters

clickedTile [Tile](#)

The tile that was clicked on.

targetTile [Tile](#)

The tile to switch with the clicked tile.

SwitchTilesRoutine(Tile, Tile)

Coroutine to swap the positions of two tiles.

```
private IEnumerator SwitchTilesRoutine(Tile clickedTile, Tile targetTile)
```

Parameters

clickedTile [Tile](#)

The tile that was clicked on.

targetTile [Tile](#)

The tile to switch with the clicked tile.

Returns

[IEnumerator](#)

Class BoardShuffler

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Provides methods to shuffle game pieces on the board.

```
public class BoardShuffler : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BoardShuffler

Methods

MovePieces(GamePiece[,], float)

Moves GamePieces into onscreen positions after being shuffled in the array.

```
public void MovePieces(GamePiece[,] allPieces, float swapTime = 0.5)
```

Parameters

allPieces [GamePiece](#)[,]

The array of all game pieces.

swapTime [float](#)

The time it takes for a game piece to swap positions.

RemoveNormalPieces(GamePiece[,])

Removes non-bomb and collectible pieces from your GamePiece array and returns them as a List.

```
public List<GamePiece> RemoveNormalPieces(GamePiece[,] allPieces)
```

Parameters

`allPieces` [GamePiece](#)[]

The array of all game pieces.

Returns

[List](#) <[GamePiece](#)>

A list of game pieces that are not bombs or collectibles.

ShuffleList(List<GamePiece>)

Shuffles a list of GamePieces in place using Fisher-Yates shuffle.

```
public void ShuffleList(List<GamePiece> piecesToShuffle)
```

Parameters

`piecesToShuffle` [List](#) <[GamePiece](#)>

The list of game pieces to shuffle.

Class Bomb

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Represents a bomb in the game, which is just a GamePiece with a BombType.

```
public class Bomb : GamePiece
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [GamePiece](#) ← Bomb

Inherited Members

[GamePiece.board](#) , [GamePiece.xIndex](#) , [GamePiece.yIndex](#) , [GamePiece.matchValue](#) ,
[GamePiece.brightnessBoost](#) , [GamePiece.saturationIncrease](#) , [GamePiece.frozen_effect](#) ,
[GamePiece.weakened_effect](#) , [GamePiece.burn_effect](#) , [GamePiece.hex_effect](#) ,
[GamePiece.touch_damage](#) , [GamePiece.effect_duration](#) , [GamePiece.tier](#) , [GamePiece.effect_status](#) ,
[GamePiece.effect_status](#) , [GamePiece.gem_effects](#) , [GamePiece.gem_status_effects](#) ,
[GamePiece.super_combos](#) , [GamePiece.gem_frame_image](#) , [GamePiece.gem_frame_image_mask](#) ,
[GamePiece.shadow](#) , [GamePiece.gem_ui_background_image](#) , [GamePiece.gem_sprite](#) ,
[GamePiece.gem_sprite](#) , [GamePiece.gem_frame_sprite](#) , [GamePiece.gem_frame_sprite](#) ,
[GamePiece.frame_color](#) , [GamePiece.frame_color](#) , [GamePiece.gem_ui_image](#) ,
[GamePiece.gem_ui_frame](#) , [GamePiece.m_isMoving](#) , [GamePiece.interpolation](#) , [GamePiece.Start\(\)](#) ,
[GamePiece.Init\(BoardManager\)](#) , [GamePiece.SetCoord\(int, int\)](#) , [GamePiece.Move\(int, int, float\)](#) ,
[GamePiece.MoveRoutine\(Vector3, float, int, int\)](#) , [GamePiece.ChangeColor\(GamePiece\)](#) ,
[GamePiece.GetAverageColor\(Sprite\)](#) , [GamePiece.LightenColor\(Color, float, float\)](#) ,
[GamePiece.GetComplementaryColor\(Color\)](#) , [GamePiece.EndTurn\(\)](#).

Fields

bombType

```
public BombType bombType
```

Field Value

[BombType](#)

Gets or sets the type of the bomb.

Enum BombType

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Enum for the various bombs available in the game.

```
public enum BombType
```

Fields

Adjacent = 3

Color = 4

Column = 1

None = 0

Row = 2

Class BubbleText

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Handles the display and animation of bubble text in the game.

```
public class BubbleText : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BubbleText

Fields

effect_text_front

The front of the text effect.

```
public Text effect_text_front
```

Field Value

Text

effect_text_shadow

The shadow of the text effect.

```
public Text effect_text_shadow
```

Field Value

Text

Methods

EnlargeAndDestroy(float, Vector2)

Coroutine to enlarge and then destroy the bubble text.

```
private IEnumerator EnlargeAndDestroy(float delay, Vector2 init_size)
```

Parameters

delay [float](#)

The delay before the resizing starts.

init_size Vector2

The initial size of the bubble text.

Returns

[IEnumerator](#)

Init(Color, string, Vector2, float, float, AnimationType)

Initializes and starts the animation for the bubble text.

```
public void Init(Color text_color, string text_value, Vector2 destination, float delay,
float size_multiplicator = 1, AnimationType anim = AnimationType.Bubble)
```

Parameters

text_color Color

The color of the text.

text_value [string](#)

The string value of the text.

destination Vector2

The destination position for the text.

delay [float](#)

The delay before the animation starts.

size_multipliator [float](#)

The size multiplier for the text.

anim [AnimationType](#)

The type of animation to perform.

MoveAnim(Vector2, float)

Coroutine to move the bubble text and then destroy it.

```
private IEnumerator MoveAnim(Vector2 destination, float delay)
```

Parameters

destination Vector2

The destination position for the text.

delay [float](#)

The delay before the movement starts.

Returns

[IEnumerator](#)

Class Collectible

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Represents a collectible in the game. It is a type of GamePiece with no match value. Collectibles can be cleared by a bomb or by reaching the bottom of the screen.

```
public class Collectible : GamePiece
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [GamePiece](#) ← Collectible

Inherited Members

[GamePiece.board](#) , [GamePiece.xIndex](#) , [GamePiece.yIndex](#) , [GamePiece.matchValue](#) ,
[GamePiece.brightnessBoost](#) , [GamePiece.saturationIncrease](#) , [GamePiece.frozen_effect](#) ,
[GamePiece.weakened_effect](#) , [GamePiece.burn_effect](#) , [GamePiece.hex_effect](#) ,
[GamePiece.touch_damage](#) , [GamePiece.effect_duration](#) , [GamePiece.tier](#) , [GamePiece.effect_status](#) ,
[GamePiece.effect_status](#) , [GamePiece.gem_effects](#) , [GamePiece.gem_status_effects](#) ,
[GamePiece.super_combos](#) , [GamePiece.gem_frame_image](#) , [GamePiece.gem_frame_image_mask](#) ,
[GamePiece.shadow](#) , [GamePiece.gem_ui_background_image](#) , [GamePiece.gem_sprite](#) ,
[GamePiece.gem_sprite](#) , [GamePiece.gem_frame_sprite](#) , [GamePiece.gem_frame_sprite](#) ,
[GamePiece.frame_color](#) , [GamePiece.frame_color](#) , [GamePiece.gem_ui_image](#) ,
[GamePiece.gem_ui_frame](#) , [GamePiece.m_isMoving](#) , [GamePiece.interpolation](#) ,
[GamePiece.Init\(BoardManager\)](#) , [GamePiece.SetCoord\(int, int\)](#) , [GamePiece.Move\(int, int, float\)](#) ,
[GamePiece.MoveRoutine\(Vector3, float, int, int\)](#) , [GamePiece.ChangeColor\(GamePiece\)](#) ,
[GamePiece.GetAverageColor\(Sprite\)](#) , [GamePiece.LightenColor\(Color, float, float\)](#) ,
[GamePiece.GetComplementaryColor\(Color\)](#) , [GamePiece.EndTurn\(\)](#).

Fields

clearedAtBottom

Flag to indicate whether the collectible can be cleared by reaching the bottom of the screen.

```
public bool clearedAtBottom
```

Field Value

[bool](#) ↗

clearedByBomb

Flag to indicate whether the collectible can be cleared by a bomb.

```
public bool clearedByBomb
```

Field Value

[bool](#) ↗

Methods

Start()

Initializes the collectible.

```
private void Start()
```

Class CombatManager

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

This class handles the functionality of the combat in the game. It inherits from the Singleton class to ensure only one instance of this class exists.

```
public class CombatManager : Singleton<CombatManager>
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton<CombatManager>](#) ← CombatManager

Inherited Members

[Singleton<CombatManager>.m_instance](#) , [Singleton<CombatManager>.Instance](#) ,
[Singleton<CombatManager>.Awake\(\)](#) , [SingletonBase.AllSingletons](#)

Fields

ai_agent

The AI player's behavior.

```
public AIBehaviour ai_agent
```

Field Value

[AIBehaviour](#)

ai_battle_card

The UI component for the AI's battle card.

```
public RectTransform ai_battle_card
```

Field Value

RectTransform

ai_board

The AI player's game board.

```
public BoardManager ai_board
```

Field Value

[BoardManager](#)

game_state

The current state of the game.

```
public CombatManager.GameState game_state
```

Field Value

[CombatManager.GameState](#)

human_board

The human player's game board.

```
public BoardManager human_board
```

Field Value

[BoardManager](#)

human_board_on_screen

Whether the human player's board is currently on screen.

```
public bool human_board_on_screen
```

Field Value

[bool](#)

move_delay

The delay between moves.

```
public float move_delay
```

Field Value

[float](#)

player_battle_card

The UI component for the player's battle card.

```
public RectTransform player_battle_card
```

Field Value

RectTransform

Methods

ApplyStatusToGems(BoardManager, float, GemStatusEffectScore)

Applies a status effect to a certain number of gems on a game board.

```
private void ApplyStatusToGems(BoardManager target_board, float gems_count,  
Move.GemStatusEffectScore status_effect)
```

Parameters

target_board [BoardManager](#)

The game board to apply the status effect to.

gems_count [float](#)

The number of gems to apply the status effect to.

status_effect [Move.GemStatusEffectScore](#)

The status effect to apply.

DealDamage(PlayerCharacter, PlayerCharacter, float, GemEffectShieldType, bool)

Deals damage to a target player from an attacking player.

```
private void DealDamage(PlayerCharacter attacker, PlayerCharacter target, float base_damage,  
GemEffectShieldType damage_type, bool poison = false)
```

Parameters

attacker [PlayerCharacter](#)

The attacking player.

target [PlayerCharacter](#)

The target player.

base_damage [float](#)

The base damage to deal.

damage_type [GemEffectShieldType](#)

The type of damage to deal.

poison bool

Whether the damage is poison damage.

Defeat()

Ends the game and declares the AI player as the winner.

```
public void Defeat()
```

EnableCombatManager()

Enables the combat manager, unblocking the player's input and blocking the AI's input.

```
public void EnableCombatManager()
```

EndCoroutine(bool)

Coroutine to end the game.

```
private IEnumerator EndCoroutine(bool victory)
```

Parameters

victory bool

Whether the game was won.

Returns

IEnumerator

EnemyTurn()

Starts the enemy's turn.

```
[ContextMenu("Enemy Turn")]
public void EnemyTurn()
```

EnemyTurnCoroutine()

Coroutine for the enemy's turn.

```
private IEnumerator EnemyTurnCoroutine()
```

Returns

[IEnumerator](#)

HumanTurn()

Starts the human player's turn.

```
[ContextMenu("Human Turn")]
public void HumanTurn()
```

HumanTurnCoroutine()

Coroutine for the human player's turn.

```
private IEnumerator HumanTurnCoroutine()
```

Returns

[IEnumerator](#)

PlayMonsterGetDamageSound()

Plays the sound effect for the AI player receiving damage.

```
public void PlayMonsterGetDamageSound()
```

PlayMonsterMakeHitSound()

Plays the sound effect for the AI player making a hit.

```
public void PlayMonsterMakeHitSound()
```

PlayPlayerGetDamageSound()

Plays the sound effect for the human player receiving damage.

```
public void PlayPlayerGetDamageSound()
```

PlayPlayerMakeHitSound()

Plays the sound effect for the human player making a hit.

```
public void PlayPlayerMakeHitSound()
```

PlayerEndTurn(PlayerCharacter)

Ends the player's turn.

```
public IEnumerator PlayerEndTurn(PlayerCharacter player)
```

Parameters

player [PlayerCharacter](#)

Returns

[IEnumerator](#)

Prepare()

Prepares the combat manager.

```
public override IEnumerator Prepare()
```

Returns

[IEnumerator](#)

ResizeBattleCards(bool)

Coroutine to resize the battle cards.

```
public IEnumerator ResizeBattleCards(bool make_player_card_bigger = true)
```

Parameters

`make_player_card_bigger` [bool](#)

Whether to make the player's card bigger.

Returns

[IEnumerator](#)

ScoreGamePieces(Combination, BoardManager)

Scores game pieces as part of a combination on a game board.

```
public IEnumerator ScoreGamePieces(Combination combo, BoardManager board)
```

Parameters

`combo` [Combination](#)

The combination of game pieces.

[board](#) [BoardManager](#)

The game board.

Returns

[IEnumerator](#)

SetupCombatManagerCoroutine()

Coroutine to set up the combat manager.

```
public IEnumerator SetupCombatManagerCoroutine()
```

Returns

[IEnumerator](#)

ShowMessage(string, Color, float)

Displays a message on the UI.

```
public void ShowMessage(string message, Color color, float size = 1.5)
```

Parameters

message [string](#)

The message to display.

color Color

The color of the message.

size [float](#)

The size of the message.

SwapBoard()

Starts the process of swapping the game board.

```
public void SwapBoard()
```

SwapBoardCoroutine()

Coroutine to swap the game board.

```
public IEnumerator SwapBoardCoroutine()
```

Returns

[IEnumerator](#)

Victory()

Ends the game and declares the human player as the winner.

```
public void Victory()
```

Enum CombatManager.GameState

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

The possible states of the game.

```
public enum CombatManager.GameState
```

Fields

ai_turn = 4

end = 6

human_turn = 3

look_at_enemy_board = 5

pause = 1

preparing = 0

swiping_boards = 2

Class CombinationScoreManager

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

A static class that handles the calculation of scores and effects for combinations in the game.

```
public static class CombinationScoreManager
```

Inheritance

[object](#) ← CombinationScoreManager

Methods

CalculateActiveEffectValue(float, float, float)

Calculates the value of an active effect based on the provided parameters.

```
public static float CalculateActiveEffectValue(float basic_effect_value, float
combo_multiplier, float gems_in_match)
```

Parameters

basic_effect_value [float](#)

The basic value of the effect.

combo_multiplier [float](#)

The multiplier for the combo.

gems_in_match [float](#)

The number of gems in the match.

Returns

[float](#)

The calculated value of the active effect.

CalculateActiveEffects(List<Combination>, PlayerCharacter)

Calculates the active effects for a list of combinations for a specific player.

```
public static List<Move.GemActiveEffectScore> CalculateActiveEffects(List<Combination>  
combinations, PlayerCharacter player)
```

Parameters

combinations [List<Combination>](#)

The list of combinations.

player [PlayerCharacter](#)

The player for whom the effects are being calculated.

Returns

[List<Move.GemActiveEffectScore>](#)

A list of active effect scores for the combinations.

CalculateStatusEffects(List<Combination>, PlayerCharacter)

Calculates the status effects for a list of combinations for a specific player.

```
public static List<Move.GemStatusEffectScore> CalculateStatusEffects(List<Combination>  
combinations, PlayerCharacter player)
```

Parameters

combinations [List<Combination>](#)

The list of combinations.

player [PlayerCharacter](#)

The player for whom the effects are being calculated.

Returns

[List](#) <[Move.GemStatusEffectScore](#)>

A list of status effect scores for the combinations.

CalculateSuperCombos(List<Combination>, PlayerCharacter)

Calculates the super combos for a list of combinations for a specific player.

```
public static List<GemSuperCombo> CalculateSuperCombos(List<Combination> combinations,  
PlayerCharacter player)
```

Parameters

combinations [List](#) <[Combination](#)>

The list of combinations.

player [PlayerCharacter](#)

The player for whom the super combos are being calculated.

Returns

[List](#) <[GemSuperCombo](#)>

A list of super combos for the combinations.

Class GamePiece

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Script to handle the functionality and display of a game piece.

```
public class GamePiece : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← GamePiece

Derived

[Bomb](#), [Collectible](#)

Fields

_effect_status

The status effect of the game piece.

```
public GemStatusEffectType _effect_status
```

Field Value

[GemStatusEffectType](#)

_frame_color

The color of the game piece's frame.

```
private Color _frame_color
```

Field Value

Color

_gem_frame_sprite

The sprite of the game piece's frame.

```
private Sprite _gem_frame_sprite
```

Field Value

Sprite

_gem_sprite

The sprite of the game piece.

```
private Sprite _gem_sprite
```

Field Value

Sprite

board

The board manager.

```
public BoardManager board
```

Field Value

[BoardManager](#)

brightnessBoost

The brightness boost of the game piece.

```
public float brightnessBoost
```

Field Value

[float](#)

burn_effect

The burn effect of the game piece.

```
public GameObject burn_effect
```

Field Value

GameObject

effect_duration

The effect duration of the game piece.

```
public int effect_duration
```

Field Value

[int](#)

frozen_effect

The frozen effect of the game piece.

```
public GameObject frozen_effect
```

Field Value

GameObject

gem_effects

The list of active effects of the game piece.

```
public List<GemActiveEffect> gem_effects
```

Field Value

[List](#)<[GemActiveEffect](#)>

gem_frame_image

The UI image component for the game piece's frame.

```
[SerializeField]  
private Image gem_frame_image
```

Field Value

Image

gem_frame_image_mask

The UI image component for the game piece's frame mask.

```
[SerializeField]  
private Image gem_frame_image_mask
```

Field Value

Image

gem_status_effects

The list of status effects of the game piece.

```
public List<GemStatusEffect> gem_status_effects
```

Field Value

[List ↗](#) <[GemStatusEffect](#)>

gem_ui_background_image

The UI image component for the game piece's background.

```
[SerializeField]  
private Image gem_ui_background_image
```

Field Value

Image

gem_ui_frame

The UI frame component for the game piece.

```
[SerializeField]  
private Image gem_ui_frame
```

Field Value

Image

gem_ui_image

The UI image component for the game piece.

```
[SerializeField]  
private Image gem_ui_image
```

Field Value

Image

hex_effect

The hex effect of the game piece.

```
public GameObject hex_effect
```

Field Value

GameObject

interpolation

The type of interpolation when the game piece moves from one position to another.

```
public InterpType interpolation
```

Field Value

[InterpType](#)

m_isMoving

Flag to check if the game piece is moving.

```
public bool m_isMoving
```

Field Value

[bool](#)

matchValue

The match value of the game piece.

```
public string matchValue
```

Field Value

[string](#)

saturationIncrease

The saturation increase of the game piece.

```
public float saturationIncrease
```

Field Value

[float](#)

shadow

The UI image component for the game piece's shadow.

```
[SerializeField]  
private Image shadow
```

Field Value

Image

super_combos

The list of super combos of the game piece.

```
public List<GemSuperCombo> super_combos
```

Field Value

[List](#)<[GemSuperCombo](#)>

tier

The tier of the game piece.

```
public GemItemTier tier
```

Field Value

[GemItemTier](#)

touch_damage

The touch damage of the game piece.

```
public float touch_damage
```

Field Value

[float](#) ↗

weakened_effect

The weakened effect of the game piece.

```
public GameObject weakened_effect
```

Field Value

[GameObject](#)

xIndex

The x index of the game piece in the board.

```
public int xIndex
```

Field Value

[int↗](#)

yIndex

The y index of the game piece in the board.

```
public int yIndex
```

Field Value

[int↗](#)

Properties

effect_status

```
public GemStatusEffectType effect_status { get; set; }
```

Property Value

[GemStatusEffectType](#)

Gets or sets the status effect of the gem.

frame_color

```
public Color frame_color { get; set; }
```

Property Value

Color

Gets or sets the color of the frame. If set, this also updates the color of the UI frame.

gem_frame_sprite

```
public Sprite gem_frame_sprite { get; set; }
```

Property Value

Sprite

Gets or sets the sprite of the gem. If set, this also updates the UI image of the gem, sets the color of the UI frame, and updates the UI background image.

gem_sprite

```
public Sprite gem_sprite { get; set; }
```

Property Value

Sprite

Gets or sets the sprite of the gem.

Methods

ChangeColor(GamePiece)

Change the color of this game piece to match another game piece.

```
public void ChangeColor(GamePiece pieceToMatch)
```

Parameters

pieceToMatch [GamePiece](#)

EndTurn()

Ends the turn of the game piece.

```
public void EndTurn()
```

GetAverageColor(Sprite)

Get the average color of a sprite.

```
private Color GetAverageColor(Sprite sprite)
```

Parameters

sprite Sprite

Returns

Color

GetComplementaryColor(Color)

Get the complementary color of a color.

```
private Color GetComplementaryColor(Color color)
```

Parameters

color Color

Returns

Color

Init(BoardManager)

Initializes this instance with the given BoardManager.

```
public void Init(BoardManager board)
```

Parameters

board [BoardManager](#)

The BoardManager to use for initialization.

LightenColor(Color, float, float)

Lighten the color of a sprite.

```
private Color LightenColor(Color color, float brightnessBoost, float saturationDecrease)
```

Parameters

color Color

brightnessBoost [float](#)

saturationDecrease [float](#)

Returns

Color

Move(int, int, float)

Move the game piece to a specific location.

```
public void Move(int destX, int destY, float timeToMove)
```

Parameters

destX [int](#)

destY [int](#)

timeToMove [float](#)

MoveRoutine(Vector3, float, int, int)

Coroutine to handle movement of the game piece.

```
private IEnumerator MoveRoutine(Vector3 destination, float timeToMove, int xIndex,  
int yIndex)
```

Parameters

destination Vector3

timeToMove [float](#)

xIndex [int](#)

yIndex [int](#)

Returns

[IEnumerator](#)

SetCoord(int, int)

Sets the coordinates of the game piece in the board.

```
public void SetCoord(int x, int y)
```

Parameters

x [int](#)

The x index of the game piece in the board.

y [int](#)

The y index of the game piece in the board.

Start()

Initializes the game piece.

```
private void Start()
```

Class HealthBar

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Manages the health bar UI for a player character.

```
public class HealthBar : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← HealthBar

Fields

bubble_text_delay

```
public float bubble_text_delay
```

Field Value

[float](#)

bubble_text_prefab

```
public GameObject bubble_text_prefab
```

Field Value

GameObject

change_color_coroutine

```
private Coroutine change_color_coroutine
```

Field Value

Coroutine

critical_damage_color

```
public Color critical_damage_color
```

Field Value

Color

current_hp_text

```
public Text current_hp_text
```

Field Value

Text

damage_color

```
public Color damage_color
```

Field Value

Color

filled_hp_bar

```
public RectTransform filled_hp_bar
```

Field Value

RectTransform

heal_color

```
public Color heal_color
```

Field Value

Color

max_hp_text

```
public Text max_hp_text
```

Field Value

Text

player_image

```
public Image player_image
```

Field Value

Image

Properties

current_hp

Gets or sets the current health value, updates the health bar, and shows death animation if health reaches zero.

```
public float current_hp { get; set; }
```

Property Value

[float ↗](#)

max_hp

Gets or sets the maximum health value and updates the health bar.

```
public float max_hp { get; set; }
```

Property Value

[float ↗](#)

Methods

MakeAlive()

Resets the visual representation to indicate the character is alive.

```
public void MakeAlive()
```

ShowDeath()

Initiates the death animation process.

```
private void ShowDeath()
```

ShowDeathCoroutine()

Coroutine for displaying the death animation.

```
private IEnumerator ShowDeathCoroutine()
```

Returns

[IEnumerator](#)

An enumerator for coroutine execution.

ShowDeathProtectionMessage()

Displays a message for death protection.

```
public void ShowDeathProtectionMessage()
```

ShowHealthChange(float, bool)

Displays health change on the health bar.

```
public void ShowHealthChange(float change, bool critical_damage)
```

Parameters

[change float](#)

The amount of health change.

[critical_damage bool](#)

Indicates if the damage is critical.

ShowMissDamage()

Displays a message when damage is missed.

```
public void ShowMissDamage()
```

Start()

Initialize the health bar state.

```
private void Start()
```

UpdateBar()

Updates the health bar based on the current health ratio.

```
private void UpdateBar()
```

Enum InterpType

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Enum for the possible interpolation types.

```
public enum InterpType
```

Fields

EaseIn = 2

EaseOut = 1

Linear = 0

SmoothStep = 3

SmootherStep = 4

Enum MatchValue

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Enum for the possible match values.

```
public enum MatchValue
```

Fields

Blue = 1

Cyan = 7

Green = 4

Indigo = 3

Magenta = 2

None = 11

Orange = 9

Purple = 8

Red = 6

Teal = 5

Wild = 10

Yellow = 0

Class PlayerBattleCardUI

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Manages the UI for a player's battle card, including health, available moves, and status effects.

```
public class PlayerBattleCardUI : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← PlayerBattleCardUI

Fields

HealthBar

```
public HealthBar HealthBar
```

Field Value

[HealthBar](#)

available_moves_text

```
public Text available_moves_text
```

Field Value

Text

charater_image

```
public Image charater_image
```

Field Value

Image

critical_damage_icon

```
public Sprite critical_damage_icon
```

Field Value

Sprite

damage_evasion_icon

```
public Sprite damage_evasion_icon
```

Field Value

Sprite

death_protection_icon

```
public Sprite death_protection_icon
```

Field Value

Sprite

frame_mask

```
public GameObject frame_mask
```

Field Value

GameObject

poison_icon

```
public Sprite poison_icon
```

Field Value

Sprite

regeneration_icon

```
public Sprite regeneration_icon
```

Field Value

Sprite

single_status_prefab

```
[SerializeField]  
private GameObject single_status_prefab
```

Field Value

GameObject

status_bar_container

```
[SerializeField]  
private GameObject status_bar_container
```

Field Value

statuses_list

```
private List<StatusBarElement> statuses_list
```

Field Value

[List](#)<[StatusBarElement](#)>

Methods

GetStatusBarElement(StatusBarElementType)

Retrieves a specific status bar element by its type.

```
public StatusBarElement GetStatusBarElement(StatusBarElementType type)
```

Parameters

type [StatusBarElementType](#)

The type of status bar element to retrieve.

Returns

[StatusBarElement](#)

The first status bar element of the specified type.

ResetStatuses()

Resets and clears all statuses from the status bar.

```
public void ResetStatuses()
```

ShowInventory()

Shows the inventory if the current player battle card is this object.

```
private void ShowInventory()
```

Start()

Initializes the manager, setting up the frame mask button to show inventory.

```
private void Start()
```

UpdateStatus(StatusBarElementType, float, bool)

Manages the UI for a player's battle card, including health, available moves, and status effects.

```
public void UpdateStatus(StatusBarElementType type, float value, bool set_value = false)
```

Parameters

type [StatusBarElementType](#)

value [float](#)

set_value [bool](#)

Class Probability

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Class for probability-related operations.

```
public class Probability
```

Inheritance

[object](#) ← Probability

Fields

random

```
private static Random random
```

Field Value

[Random](#)

Methods

Success(float)

Returns whether a probabilistic event was successful.

```
public static bool Success(float probability)
```

Parameters

probability [float](#)

The probability of success, expressed as a decimal or percentage.

Returns

bool ↗

Class StatusBarElement

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Represents an individual status bar element in the UI.

```
public class StatusBarElement : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StatusBarElement

Fields

ScaleRoot

```
[SerializeField]  
private RectTransform ScaleRoot
```

Field Value

RectTransform

_element_icon_sprite

```
private Sprite _element_icon_sprite
```

Field Value

Sprite

_status_text

```
private float _status_text
```

Field Value

[float](#)

appereance_effect_coroutine

```
private Coroutine appereance_effect_coroutine
```

Field Value

Croutine

icon_image_component

```
[SerializeField]  
private Image icon_image_component
```

Field Value

Image

status_element_type

```
public StatusBarElementType status_element_type
```

Field Value

[StatusBarElementType](#)

status_text_component

```
[SerializeField]  
private Text status_text_component
```

Field Value

Text

Properties

element_icon_sprite

Gets or sets the icon sprite of the status bar element.

```
public Sprite element_icon_sprite { get; set; }
```

Property Value

Sprite

status_text

Gets or sets the text displayed on the status bar element.

```
public float status_text { get; set; }
```

Property Value

[float](#)

Methods

DestroyCoroutine()

```
private IEnumerator DestroyCoroutine()
```

Returns

[IEnumerator](#)

DestroyStatusBarElement()

Destroys the status bar element with an effect.

```
public void DestroyStatusBarElement()
```

StatusAppearEffect()

Initiates the appearance effect of the status bar element.

```
public void StatusAppearEffect()
```

Enum StatusBarElementType

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Defines types of status bar elements.

```
public enum StatusBarElementType
```

Fields

CriticalDamage = 1

DamageEvasion = 2

DeathProtection = 3

None = 0

Poison = 5

Regeneration = 4

Class SwitchBoardButton

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

A class for a switch board button in a game.

```
public class SwitchBoardButton : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← SwitchBoardButton

Methods

Start()

Called before the first frame update.

```
private void Start()
```

Update()

Called once per frame.

```
private void Update()
```

Class Tile

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Represents a tile in a game board.

```
public class Tile : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← Tile

Fields

board

```
public BoardManager board
```

Field Value

[BoardManager](#)

breakableSprites

```
public Sprite[] breakableSprites
```

Field Value

Sprite[]

breakableValue

```
public int breakableValue
```

Field Value

[int](#) ↗

m_spriteRenderer

```
private SpriteRenderer m_spriteRenderer
```

Field Value

SpriteRenderer

normalColor

```
public Color normalColor
```

Field Value

Color

position

```
public Vector3 position
```

Field Value

Vector3

tileType

```
public TileType tileType
```

Field Value

[TileType](#)

xIndex

```
public int xIndex
```

Field Value

[int↗](#)

yIndex

```
public int yIndex
```

Field Value

[int↗](#)

Methods

Awake()

Initializes the tile.

```
private void Awake()
```

BreakTile()

Initiates the process of breaking the tile if it is breakable.

```
public void BreakTile()
```

BreakTileRoutine()

Coroutine to handle the breaking of a breakable tile.

```
private IEnumerator BreakTileRoutine()
```

Returns

[IEnumerator](#)

OnMouseDown()

Handles the mouse down event on the tile.

```
public void OnMouseDown()
```

OnMouseEnter()

Handles the mouse enter event on the tile.

```
public void OnMouseEnter()
```

OnMouseUp()

Handles the mouse up event on the tile.

```
public void OnMouseUp()
```

Start()

Sets up the tile at the start of the game.

```
private void Start()
```

Enum TileType

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Defines the types of tiles.

```
public enum TileType
```

Fields

Breakable = 2

Normal = 0

Obstacle = 1

Class Utils

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

Utility class providing various helper methods.

```
public static class Utils
```

Inheritance

[object](#) ← Utils

Methods

ChangeColorSmooth(GetColorDelegate, SetColorDelegate, Color, float)

Coroutine to change the color on a UI element smoothly over time.

```
public static IEnumerator ChangeColorSmooth(Utils.GetColorDelegate getColor,
                                         Utils.SetColorDelegate setColor, Color target_color, float delay)
```

Parameters

getColor [Utils.GetColorDelegate](#)

setColor [Utils.SetColorDelegate](#)

target_color Color

delay [float](#)

Returns

[IEnumerator](#)

FlashColor(GetColorDelegate, SetColorDelegate, Color, float)

Coroutine to flash a color on a UI element over time.

```
public static IEnumerator FlashColor(Utils.GetColorDelegate getColor, UtilsSetColorDelegate  
setColor, Color target_color, float delay)
```

Parameters

getColor [Utils.GetColorDelegate](#)

setColor [Utils.SetColorDelegate](#)

target_color Color

delay [float](#)

Returns

[IEnumerator](#)

InterpolateCanvasGroupAlpha(CanvasGroup, float, float, float)

Coroutine to interpolate the alpha value of a CanvasGroup over time.

```
public static IEnumerator InterpolateCanvasGroupAlpha(CanvasGroup canvas_group, float  
start_value, float target_value, float delay)
```

Parameters

canvas_group CanvasGroup

start_value [float](#)

target_value [float](#)

delay [float](#)

Returns

[IEnumerator](#)

MoveObjects(GameObject, Vector3, float)

Coroutine to move a GameObject by a certain offset over time.

```
public static IEnumerator MoveObjects(GameObject obj, Vector3 offset, float delay)
```

Parameters

obj GameObject

offset Vector3

delay [float](#)

Returns

[IEnumerator](#)

MoveObjects(GameObject[], Vector3, float)

Coroutine to move multiple GameObjects by a certain offset over time.

```
public static IEnumerator MoveObjects(GameObject[] objects, Vector3 offset, float delay)
```

Parameters

objects GameObject[]

offset Vector3

delay [float](#)

Returns

[IEnumerator](#)

RebuildLayoutNextFrame(RectTransform, Action, float)

Coroutine to rebuild a RectTransform's layout at the end of the next frame.

```
public static IEnumerator RebuildLayoutNextFrame(RectTransform rect, Action callback = null,  
float callback_delay = 0.5)
```

Parameters

rect RectTransform

callback [Action](#)

callback_delay [float](#)

Returns

[IEnumerator](#)

RemoveChilds(Transform)

Removes all child GameObjects from a parent Transform.

```
public static void RemoveChilds(Transform parent)
```

Parameters

parent Transform

ResizeObjects(RectTransform, Vector2, float)

Coroutine to resize a RectTransform to a target size over time.

```
public static IEnumerator ResizeObjects(RectTransform obj, Vector2 targetSize, float delay)
```

Parameters

obj RectTransform

targetSize Vector2

delay [float](#)

Returns

[IEnumerator](#)

ResizeObjects(RectTransform[], Vector2, float)

Coroutine to resize multiple RectTransforms to a target size over time.

```
public static IEnumerator ResizeObjects(RectTransform[] objects, Vector2 targetSize,  
float delay)
```

Parameters

objects RectTransform[]

targetSize Vector2

delay [float](#)

Returns

[IEnumerator](#)

ResizeObjectsScale(RectTransform, Vector2, float)

Coroutine to resize a RectTransform's scale to a target size over time.

```
public static IEnumerator ResizeObjectsScale(RectTransform obj, Vector2 targetSize,  
float delay)
```

Parameters

obj RectTransform

targetSize Vector2

delay [float](#)

Returns

[IEnumerator](#)

Shuffle(int[])

Shuffles an array of integers in-place using the Fisher-Yates algorithm.

```
public static void Shuffle(int[] array)
```

Parameters

array [int](#)[]

The array to shuffle.

Delegate Utils.GetColorDelegate

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

```
public delegate Color Utils.GetColorDelegate()
```

Returns

Color

Delegate Utils.SetColorDelegate

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

```
public delegate void UtilsSetColorDelegate(Color color)
```

Parameters

color Color

Delegate Utils.todo

Namespace: [TaleOfTiles.Arena](#)

Assembly: Assembly-CSharp.dll

```
public delegate void Utils.todo()
```

Namespace TaleOfTiles.Audio

Classes

[AudioManager](#)

This class handles the functionality of the game's audio. It inherits from the Singleton class to ensure only one instance of this class exists.

[GeneralAudioGroup](#)

Manages general audio clips for the game, including background music and sound effects. Inherits from Singleton to ensure only one instance exists.

Class AudioManager

Namespace: [TaleOfTiles.Audio](#)

Assembly: Assembly-CSharp.dll

This class handles the functionality of the game's audio. It inherits from the Singleton class to ensure only one instance of this class exists.

```
[RequireComponent(typeof(AudioSource))]  
public class AudioManager : Singleton<AudioManager>
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ←
[Singleton<AudioManager>](#) ← AudioManager

Inherited Members

[Singleton<AudioManager>.m_instance](#) , [Singleton<AudioManager>.Instance](#) ,
[Singleton<AudioManager>.Awake\(\)](#) , [SingletonBase.AllSingletons](#)

Fields

background_music

The audio source for the game's background music.

```
public AudioSource background_music
```

Field Value

effect_sounds

The audio source for the game's effect sounds.

```
public AudioSource effect_sounds
```

Field Value

AudioSource

enemy_sounds

The audio source for the enemy's sounds.

```
public AudioSource enemy_sounds
```

Field Value

AudioSource

general_audio_group

The group of general audio clips.

```
private GeneralAudioGroup general_audio_group
```

Field Value

[GeneralAudioGroup](#)

own_audio_player

The game's own audio player.

```
public AudioSource own_audio_player
```

Field Value

AudioSource

player_sounds

The audio source for the player's sounds.

```
public AudioSource player_sounds
```

Field Value

Properties

defeat_audio_clip

The audio clip for the defeat screen.

```
public AudioClip defeat_audio_clip { get; }
```

Property Value

AudioClip

loading_screen_audio_clip

The audio clip for the loading screen.

```
public AudioClip loading_screen_audio_clip { get; }
```

Property Value

AudioClip

player_get_damage_sounds

The list of audio clips for when the player gets damaged.

```
public List<AudioClip> player_get_damage_sounds { get; }
```

Property Value

[List](#) <AudioClip>

player_hit_sounds

The list of audio clips for when the player hits an enemy.

```
public List<AudioClip> player_hit_sounds { get; }
```

Property Value

[List](#) <AudioClip>

victory_audio_clip

The audio clip for the victory screen.

```
public AudioClip victory_audio_clip { get; }
```

Property Value

AudioClip

Methods

ChangeBackgroundMusic(AudioClip, float)

Changes the background music.

```
public void ChangeBackgroundMusic(AudioClip new_clip, float fade_time)
```

Parameters

`new_clip` AudioClip

`fade_time` [float](#)

ChangeMusic(AudioSource, AudioClip, float)

Changes the music from the current audio source to a new audio clip.

```
private void ChangeMusic(AudioSource audioSource, AudioClip newClip, float fadeTime)
```

Parameters

audioSource AudioSource

newClip AudioClip

fadeTime [float](#)

ChangeMusicCoroutine(AudioSource, AudioClip, float)

Coroutine to handle the transition of music from the current audio clip to a new audio clip.

```
private IEnumerator ChangeMusicCoroutine(AudioSource audioSource, AudioClip newClip, float fadeTime)
```

Parameters

audioSource AudioSource

newClip AudioClip

fadeTime [float](#)

Returns

[IEnumerator](#)

FadeInCoroutine(AudioSource, float)

Coroutine to fade in an audio source.

```
private IEnumerator FadeInCoroutine(AudioSource audioSource, float fadeTime)
```

Parameters

audioSource AudioSource

fadeTime [float](#)

Returns

[IEnumerator](#)

FadeOutCoroutine(AudioSource, float)

Coroutine to fade out an audio source.

```
private IEnumerator FadeOutCoroutine(AudioSource audioSource, float fadeTime)
```

Parameters

audioSource AudioSource

fadeTime [float](#)

Returns

[IEnumerator](#)

MuteBackgroundMusic()

Mutes the background music.

```
public void MuteBackgroundMusic()
```

MuteEffects()

Mutes the effect sounds.

```
public void MuteEffects()
```

PlayBackgroundMusic(AudioClip)

Plays the background music.

```
public void PlayBackgroundMusic(AudioClip clip)
```

Parameters

`clip` AudioClip

PlayDefeatMusic()

Plays the defeat music.

```
public void PlayDefeatMusic()
```

PlayEnemySound(AudioClip)

Plays an enemy sound.

```
public void PlayEnemySound(AudioClip clip)
```

Parameters

`clip` AudioClip

PlayLoadingScreenMusic()

Plays the loading screen music.

```
public void PlayLoadingScreenMusic()
```

PlayPlayerSound(AudioClip)

Play a specific sound clip from the player's audio source.

```
public void PlayPlayerSound(AudioClip clip)
```

Parameters

`clip` AudioClip

PlaySoundSingleTime(AudioClip)

Plays a sound once.

```
public void PlaySoundSingleTime(AudioClip clip)
```

Parameters

`clip` AudioClip

PlayVictoryMusic()

Plays the victory music.

```
public void PlayVictoryMusic()
```

Prepare()

Prepares the audio manager.

```
public override IEnumerator Prepare()
```

Returns

[IEnumerator](#)

Class GeneralAudioGroup

Namespace: [TaleOfTiles.Audio](#)

Assembly: Assembly-CSharp.dll

Manages general audio clips for the game, including background music and sound effects. Inherits from Singleton to ensure only one instance exists.

```
public class GeneralAudioGroup : Singleton<GeneralAudioGroup>
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton<GeneralAudioGroup>](#) ← GeneralAudioGroup

Inherited Members

[Singleton<GeneralAudioGroup>.m_instance](#) , [Singleton<GeneralAudioGroup>.Instance](#) ,
[Singleton<GeneralAudioGroup>.Awake\(\)](#) , [Singleton<GeneralAudioGroup>.Prepare\(\)](#) ,
[SingletonBase.AllSingletons](#)

Fields

defeat_audio_clip

Audio clip to be played upon defeat.

```
public AudioClip defeat_audio_clip
```

Field Value

AudioClip

loading_screen_audio_clip

Audio clip to be played during the loading screen.

```
public AudioClip loading_screen_audio_clip
```

Field Value

AudioClip

player_get_damage_sounds

A list of audio clips to be played when the player receives damage.

```
public List<AudioClip> player_get_damage_sounds
```

Field Value

[List](#) <AudioClip>

player_hit_sounds

A list of audio clips to be played when the player is hit.

```
public List<AudioClip> player_hit_sounds
```

Field Value

[List](#) <AudioClip>

victory_audio_clip

Audio clip to be played upon achieving victory.

```
public AudioClip victory_audio_clip
```

Field Value

AudioClip

Namespace TaleOfTiles.Card

Classes

[BigCardUI](#)

A class that represents a big card in the UI.

[CardGemUI](#)

Handles the display and functionality of the gem card in the UI.

[CardUI](#)

Handles the display and functionality of the card in the UI.

[CardsHolderManager](#)

Manages the collection of card objects for a player. This class is responsible for storing, activating, deactivating, and retrieving player's cards. It extends the Singleton pattern to ensure only one instance of the manager exists.

[GemEffectAttributeUI](#)

Handles the display of gem effect attributes in the user interface.

[GemEffectUI](#)

Manages the UI elements for displaying gem effects in a game.

[SingleCardPageUI](#)

Manages the UI for displaying individual card pages.

Structs

[GemEffectAttribute](#)

Represents a single gem effect attribute, holding its name and value.

Delegates

[CardUI.CardBuildEvent](#)

Delegate for the event when the card build is complete.

Class BigCardUI

Namespace: [TaleOfTiles.Card](#)

Assembly: Assembly-CSharp.dll

A class that represents a big card in the UI.

```
public class BigCardUI : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BigCardUI

Fields

card_id

The ID of the card.

```
public string card_id
```

Field Value

[string](#)

card_image

The image of the card.

```
public Image card_image
```

Field Value

Image

card_label

The label of the card.

```
public TextMeshProUGUI card_label
```

Field Value

TextMeshProUGUI

equip_button

The button to equip or unequip the card.

```
[SerializeField]  
private Button equip_button
```

Field Value

Button

Methods

ActivateCard()

Activates the card.

```
private void ActivateCard()
```

DeactivateCard()

Deactivates the card.

```
private void DeactivateCard()
```

SetCard(CardObject)

Sets the card data for the UI element.

```
public void SetCard(CardObject card)
```

Parameters

card [CardObject](#)

The card object to set.

ShowCard()

Shows the card in the UI.

```
public void ShowCard()
```

Class CardGemUI

Namespace: [TaleOfTiles.Card](#)

Assembly: Assembly-CSharp.dll

Handles the display and functionality of the gem card in the UI.

```
public class CardGemUI : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← CardGemUI

Fields

collapsed

Flag to indicate whether the gem's card is collapsed or not.

```
private bool collapsed
```

Field Value

[bool](#)

full_height

The full height of the gem's card.

```
public float full_height
```

Field Value

[float](#)

gem

The gem to be displayed in the UI.

```
public GemItem gem
```

Field Value

[GemItem](#)

gem_description

The UI element for displaying the gem's description.

```
public Text gem_description
```

Field Value

Text

gem_effect_ui_prefab

The prefab for the UI of the gem's effect.

```
public GameObject gem_effect_ui_prefab
```

Field Value

GameObject

gem_effects_container

The container for the gem's effects in the UI.

```
public RectTransform gem_effects_container
```

Field Value

gem_icon

The UI element for displaying the gem's icon.

```
public Image gem_icon
```

Field Value

Image

gem_name

The UI element for displaying the gem's name.

```
public Text gem_name
```

Field Value

Text

parent

The parent card UI of this gem UI.

```
public CardUI parent
```

Field Value

[CardUI](#)

passive_skill

The passive skill to be displayed in the UI.

```
public PassiveSkillObject passive_skill
```

Field Value

[PassiveSkillObject](#)

short_info_container

The container for the gem's short information in the UI.

```
public RectTransform short_info_container
```

Field Value

RectTransform

Methods

BuildGem()

Builds the gem card UI elements.

```
public IEnumerator BuildGem()
```

Returns

[IEnumerator](#)

CardBuilt()

Handles the event when the card is built.

```
public void CardBuilt()
```

Init(PassiveSkillObject)

Initializes the gem card with a PassiveSkillObject.

```
public IEnumerator Init(PassiveSkillObject passive_skill)
```

Parameters

`passive_skill` [PassiveSkillObject](#)

The passive skill to display in the card.

Returns

[IEnumerator](#)

Init(GemItem)

Initializes the gem card with a GemItem.

```
public IEnumerator Init(GemItem gem)
```

Parameters

`gem` [GemItem](#)

The gem to display in the card.

Returns

[IEnumerator](#)

ToggleCollapse()

Toggles the collapse state of the gem card.

```
public void ToggleCollapse()
```

ToggleCollapseCoroutine()

Coroutine to handle the collapse and expand animation of the gem card.

```
private IEnumerator ToggleCollapseCoroutine()
```

Returns

[IEnumerator](#)

Class CardUI

Namespace: [TaleOfTiles.Card](#)

Assembly: Assembly-CSharp.dll

Handles the display and functionality of the card in the UI.

```
public class CardUI : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← CardUI

Fields

active_skills_container

The container for the card's active skills in the UI.

```
public RectTransform active_skills_container
```

Field Value

RectTransform

card

The card to be displayed in the UI.

```
public CardObject card
```

Field Value

[CardObject](#)

card_background

The UI element for displaying the card's background.

```
public Image card_background
```

Field Value

Image

card_description

The UI element for displaying the card's description.

```
public Text card_description
```

Field Value

Text

card_gem_ui_prefab

The prefab for the UI of the card's gem.

```
public GameObject card_gem_ui_prefab
```

Field Value

GameObject

card_icon

The UI element for displaying the card's icon.

```
public Image card_icon
```

Field Value

Image

card_name

The UI element for displaying the card's name.

```
public Text card_name
```

Field Value

Text

passive_skills_container

The container for the card's passive skills in the UI.

```
public RectTransform passive_skills_container
```

Field Value

RectTransform

Methods

BuildCardCoroutine()

Coroutine to build the card UI.

```
private IEnumerator BuildCardCoroutine()
```

Returns

[IEnumerator](#)

BuildCardUI()

Coroutine to build the card UI elements.

```
private IEnumerator BuildCardUI()
```

Returns

[IEnumerator](#)

HideObject()

Makes the card UI invisible.

```
public void HideObject()
```

OnValidate()

Validates the object in the editor.

```
private void OnValidate()
```

SetNewCard(CardObject)

Sets a new card object for this UI.

```
public void SetNewCard(CardObject card)
```

Parameters

card [CardObject](#)

The new card object.

SetNewCardCoroutine()

Coroutine to set a new card object for this UI.

```
private IEnumerator SetNewCardCoroutine()
```

Returns

[IEnumerator](#)

ShowObject()

Makes the card UI visible.

```
public void ShowObject()
```

StartBuilding()

Starts building the card UI.

```
[ContextMenu("Build Card")]
public void StartBuilding()
```

Events

build_complete_event

Event triggered when the card build is complete.

```
[HideInInspector]
public event CardUI.CardBuildEvent build_complete_event
```

Event Type

[CardUI.CardBuildEvent](#)

Delegate CardUI.CardBuildEvent

Namespace: [TaleOfTiles.Card](#)

Assembly: Assembly-CSharp.dll

Delegate for the event when the card build is complete.

```
[HideInInspector]  
public delegate void CardUI.CardBuildEvent()
```

Class CardsHolderManager

Namespace: [TaleOfTiles.Card](#)

Assembly: Assembly-CSharp.dll

Manages the collection of card objects for a player. This class is responsible for storing, activating, deactivating, and retrieving player's cards. It extends the Singleton pattern to ensure only one instance of the manager exists.

```
public class CardsHolderManager : Singleton<CardsHolderManager>
```

Inheritance

[Object](#) ↗ ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ←
[Singleton](#)<[CardsHolderManager](#)> ← CardsHolderManager

Inherited Members

[Singleton](#)<[CardsHolderManager](#)>.m_instance , [Singleton](#)<[CardsHolderManager](#)>.Instance ,
[Singleton](#)<[CardsHolderManager](#)>.Awake() , [SingletonBase](#).AllSingletons

Fields

all_cards_holder

Stores all cards available in the game, indexed by their ID.

```
public Dictionary<string, CardObject> all_cards_holder
```

Field Value

[Dictionary](#)<[string](#) , [CardObject](#)>

all_player_cards

Lists all cards owned by the player.

```
public List<CardObject> all_player_cards
```

Field Value

[List ↗ <CardObject>](#)

selected_player_cards

Lists all currently active (selected) cards for the player.

```
public List<CardObject> selected_player_cards
```

Field Value

[List ↗ <CardObject>](#)

Methods

ActivateCard(string)

Activates a card for the player.

```
public void ActivateCard(string ID)
```

Parameters

ID [string ↗](#)

The ID of the card to activate.

DeactivateCard(string)

Deactivates a player's card.

```
public void DeactivateCard(string ID)
```

Parameters

ID [string ↗](#)

The ID of the card to deactivate.

GetCard(string)

Retrieves a card by its ID.

```
public CardObject GetCard(string ID)
```

Parameters

ID [string](#)

The ID of the card to retrieve.

Returns

[CardObject](#)

The card object if found; otherwise, null.

GivePlayerNewCard(string)

Adds a new card to the player's collection.

```
public void GivePlayerNewCard(string ID)
```

Parameters

ID [string](#)

The ID of the card to add.

Prepare()

Prepares the card manager by initializing card collections and loading data.

```
public override IEnumerator Prepare()
```

Returns

[IEnumerator](#)

Coroutine for asynchronous operation.

Struct GemEffectAttribute

Namespace: [TaleOfTiles.Card](#)

Assembly: Assembly-CSharp.dll

Represents a single gem effect attribute, holding its name and value.

```
[Serializable]
public struct GemEffectAttribute
```

Constructors

GemEffectAttribute(string, string)

```
public GemEffectAttribute(string attribute_name, string attribute_value)
```

Parameters

attribute_name [string](#)

attribute_value [string](#)

Fields

attribute_name

```
public string attribute_name
```

Field Value

[string](#)

attribute_value

```
public string attribute_value
```

Field Value

[string](#) ↗

Class GemEffectAttributeUI

Namespace: [TaleOfTiles.Card](#)

Assembly: Assembly-CSharp.dll

Handles the display of gem effect attributes in the user interface.

```
public class GemEffectAttributeUI : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← GemEffectAttributeUI

Fields

attribute_name

The UI element displaying the name of the attribute.

```
public Text attribute_name
```

Field Value

Text

attribute_value

The UI element displaying the value of the attribute.

```
public Text attribute_value
```

Field Value

Text

Methods

SetAttribute(GemEffectAttribute)

Sets the attribute's name and value in the UI.

```
public IEnumerator SetAttribute(GemEffectAttribute attr)
```

Parameters

attr [GemEffectAttribute](#)

The gem effect attribute to display.

Returns

[IEnumerator](#) ↗

An IEnumerator for coroutine management.

Class GemEffectUI

Namespace: [TaleOfTiles.Card](#)

Assembly: Assembly-CSharp.dll

Manages the UI elements for displaying gem effects in a game.

```
public class GemEffectUI : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← GemEffectUI

Fields

CritDamageChance_icon

```
public Sprite CritDamageChance_icon
```

Field Value

Sprite

EvadeChance_icon

```
public Sprite EvadeChance_icon
```

Field Value

Sprite

IncomeDamageReduction_icon

```
public Sprite IncomeDamageReduction_icon
```

Field Value

Sprite

OutcomeDamageIncrease_icon

```
public Sprite OutcomeDamageIncrease_icon
```

Field Value

Sprite

burn_status_icon

```
public Sprite burn_status_icon
```

Field Value

Sprite

damage_icon

```
public Sprite damage_icon
```

Field Value

Sprite

effect

```
public GemActiveEffect effect
```

Field Value

effect_attribute_ui

```
public GameObject effect_attribute_ui
```

Field Value

GameObject

effect_attributes

```
public List<GemEffectAttribute> effect_attributes
```

Field Value

[List](#)<[GemEffectAttribute](#)>

effect_attributes_container

```
public RectTransform effect_attributes_container
```

Field Value

RectTransform

effect_icon

```
public Image effect_icon
```

Field Value

Image

effect_type_description

```
public Text effect_type_description
```

Field Value

Text

effect_type_label

```
public Text effect_type_label
```

Field Value

Text

frozen_status_icon

```
public Sprite frozen_status_icon
```

Field Value

Sprite

heal_icon

```
public Sprite heal_icon
```

Field Value

Sprite

hexed_bonus_icon

```
public Sprite hexed_bonus_icon
```

Field Value

Sprite

move_bonus_icon

```
public Sprite move_bonus_icon
```

Field Value

Sprite

passive_skill

```
public PassiveSkillObject passive_skill
```

Field Value

[PassiveSkillObject](#)

status_effect

```
public GemStatusEffect status_effect
```

Field Value

[GemStatusEffect](#)

weakened_status_icon

```
public Sprite weakened_status_icon
```

Field Value

Sprite

Methods

BuildEffect(PassiveSkillObject)

Constructs and displays the attributes of a passive skill.

```
public void BuildEffect(PassiveSkillObject passive_skill)
```

Parameters

`passive_skill` [PassiveSkillObject](#)

The passive skill to be built.

BuildEffect(GemActiveEffect)

Constructs and displays the attributes of a gem's active effect.

```
public void BuildEffect(GemActiveEffect effect)
```

Parameters

`effect` [GemActiveEffect](#)

The active effect to be built.

BuildEffect(GemStatusEffect)

Constructs and displays the attributes of a gem's status effect.

```
public void BuildEffect(GemStatusEffect effect)
```

Parameters

effect [GemStatusEffect](#)

The status effect to be built.

CreateAttributesList(PassiveSkillObject)

Generates a list of attributes for a passive skill.

```
private void CreateAttributesList(PassiveSkillObject passive_skill)
```

Parameters

passive_skill [PassiveSkillObject](#)

The passive skill whose attributes are to be created.

CreateAttributesList(GemActiveEffect)

Generates a list of attributes for a gem's active effect.

```
private void CreateAttributesList(GemActiveEffect effect)
```

Parameters

effect [GemActiveEffect](#)

The active effect whose attributes are to be created.

CreateAttributesList(GemStatusEffect)

Generates a list of attributes for a gem's status effect.

```
private void CreateAttributesList(GemStatusEffect effect)
```

Parameters

effect [GemStatusEffect](#)

The status effect whose attributes are to be created.

CreateAttributesUI()

Creates UI elements for displaying effect attributes.

```
private IEnumerator CreateAttributesUI()
```

Returns

[IEnumerator](#)

An enumerator for coroutine execution.

Init(PassiveSkillObject)

Initializes the UI for a PassiveSkillObject.

```
public IEnumerator Init(PassiveSkillObject passive_skill)
```

Parameters

passive_skill [PassiveSkillObject](#)

The passive skill to display.

Returns

[IEnumerator](#)

An IEnumerator for coroutine management.

Init(GemActiveEffect)

Initializes the UI for a GemActiveEffect.

```
public IEnumerator Init(GemActiveEffect effect)
```

Parameters

effect [GemActiveEffect](#)

The active gem effect to display.

Returns

[IEnumerator](#)

An IEnumerator for coroutine management.

Init(GemStatusEffect)

Initializes the UI for a GemStatusEffect.

```
public IEnumerator Init(GemStatusEffect status_effect)
```

Parameters

status_effect [GemStatusEffect](#)

The status gem effect to display.

Returns

[IEnumerator](#)

An IEnumerator for coroutine management.

Class SingleCardPageUI

Namespace: [TaleOfTiles.Card](#)

Assembly: Assembly-CSharp.dll

Manages the UI for displaying individual card pages.

```
public class SingleCardPageUI : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← SingleCardPageUI

Fields

back_button

```
[SerializeField]  
private Button back_button
```

Field Value

Button

current_shown_card_id

```
private string current_shown_card_id
```

Field Value

[string](#)

main_canvas_group

```
public CanvasGroup main_canvas_group
```

Field Value

CanvasGroup

page_title

```
public Text page_title
```

Field Value

Text

rendered_cards

```
public Dictionary<string, GameObject> rendered_cards
```

Field Value

[Dictionary](#)<[string](#), GameObject>

scroll_component

```
public ScrollRect scroll_component
```

Field Value

ScrollRect

scroll_viewport

```
public RectTransform scroll_viewport
```

Field Value

RectTransform

ui_single_card_prefab

```
public GameObject ui_single_card_prefab
```

Field Value

GameObject

Methods

BuildAllSingleCards()

Builds UI elements for each card.

```
public void BuildAllSingleCards()
```

Hide()

Hides the UI of the single card page.

```
public void Hide()
```

SetPageTitle(string)

Sets the title of the page.

```
public void SetPageTitle(string name)
```

Parameters

name [string](#)

The title to be set.

ShowCard(string)

Shows a specific card on the UI based on its ID.

```
public bool ShowCard(string ID)
```

Parameters

ID [string](#)

The ID of the card to show.

Returns

[bool](#)

Boolean indicating success of operation.

Namespace TaleOfTiles.Card.Gems

Classes

[GemActiveEffect](#)

Represents an active effect associated with a gem.

[GemItem](#)

Represents a gem item in the game. This class is a ScriptableObject, allowing it to be used as a data asset in Unity.

[GemSuperCombo](#)

Represents a super combo associated with a gem, encompassing various combo types.

[GemSuperCombo.ActivateGemsOnBoard](#)

Represents an 'Activate Gems On Board' Combo in the Gem Super Combo system. This combo allows for the activation of a portion of the gems on the board without needing to match them.

[GemSuperCombo.BurnCombo](#)

Represents a Burn Combo in the Gem Super Combo system, which applies a burning effect to a part of the board.

[GemSuperCombo.CopyEnemyDeck](#)

Represents a 'Copy Enemy Deck' Combo in the Gem Super Combo system. This combo allows the player to copy the enemy's deck or abilities for a certain duration.

[GemSuperCombo.CriticalDamage](#)

Represents a 'Critical Damage' Combo in the Gem Super Combo system. This combo amplifies damage for a certain number of attacks, increasing their critical hit potential.

[GemSuperCombo.DamageEvasion](#)

Represents a 'Damage Evasion' Combo in the Gem Super Combo system. This combo allows the player to evade a certain number of attacks.

[GemSuperCombo.DestroyEnemyGems](#)

Represents a 'Destroy Enemy Gems' Combo in the Gem Super Combo system. This combo allows for a portion of the enemy's gems on the board to be destroyed.

[GemSuperCombo.ExtraMoves](#)

Represents an 'Extra Moves' Combo in the Gem Super Combo system, which grants additional moves to the player.

[GemSuperCombo.FrostCombo](#)

Represents a Frost Combo as part of the Gem Super Combo system. This combo can affect a part of the game board for a certain duration based on the combo activation count.

[GemSuperCombo.HexCombo](#)

Represents a Hex Combo in the Gem Super Combo system, which applies a hex effect on a part of the board.

[GemSuperCombo.Poison](#)

Represents a 'Poison' Combo in the Gem Super Combo system. This combo inflicts poison damage over a set duration, dealing damage each turn.

[GemSuperCombo.RegenerationCombo](#)

Represents a Regeneration Combo in the Gem Super Combo system, which provides health regeneration over a set number of turns.

[GemSuperCombo.SuddenDeathProtection](#)

Represents a 'Sudden Death Protection' Combo in the Gem Super Combo system. This combo provides protection against sudden defeat for a specified number of turns.

Structs

[GemStatusEffect](#)

Represents a status effect associated with a gem.

Interfaces

[ISuperCombo](#)

Interface for defining super combo behavior.

Enums

[GemActiveEffectType](#)

Defines the types of active effects that a gem can have.

[GemEffectShieldType](#)

Defines the types of shields associated with gem effects.

[GemItemTier](#)

Defines the tiers of gem items.

[GemStatusEffectType](#)

Defines the types of status effects that a gem can have.

[GemSuperComboType](#)

Defines the types of super combos available for gems.

Delegates

[GemItem.OnGemItemChanged](#)

Delegate for handling events when a GemItem changes.

Class GemActiveEffect

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents an active effect associated with a gem.

```
[Serializable]  
public class GemActiveEffect
```

Inheritance

[object](#) ← GemActiveEffect

Fields

ID

Unique identifier for the effect.

```
public string ID
```

Field Value

[string](#)

MatchOf3EffectValue

Effect value for a match of 3 gems. Read-only.

```
public float MatchOf3EffectValue
```

Field Value

[float](#)

MatchOf4EffectValue

Effect value for a match of 4 gems. Read-only.

```
public float MatchOf4EffectValue
```

Field Value

[float](#) ↗

MatchOf5EffectValue

Effect value for a match of 5 gems. Read-only.

```
public float MatchOf5EffectValue
```

Field Value

[float](#) ↗

MatchOf6EffectValue

Effect value for a match of 6 gems. Read-only.

```
public float MatchOf6EffectValue
```

Field Value

[float](#) ↗

combo_multiplier

Multiplier for the effect based on gem combinations.

```
public float combo_multiplier
```

Field Value

[float](#) ↗

effect_type

Type of the active effect.

```
public GemActiveEffectType effect_type
```

Field Value

[GemActiveEffectType](#)

effect_value

The value of the effect.

```
public float effect_value
```

Field Value

[float](#) ↗

shield_type

Type of the shield associated with this effect.

```
public GemEffectShieldType shield_type
```

Field Value

[GemEffectShieldType](#)

Methods

Copy()

Creates a copy of this GemActiveEffect.

```
public GemActiveEffect Copy()
```

Returns

[GemActiveEffect](#)

A new instance of GemActiveEffect with the same values as this instance.

GenerateID()

Generates a new unique ID for the effect.

```
public void GenerateID()
```

Hex()

Sets the effect to a hex effect.

```
public void Hex()
```

SetComboMult(float)

Sets the combo multiplier.

```
public void SetComboMult(float new_value)
```

Parameters

`new_value float`

The new value to set for the combo multiplier.

SetEffectValue(float)

Sets the effect value.

```
public void SetEffectValue(float new_value)
```

Parameters

`new_value` [float](#)

The new value to set for the effect.

Enum GemActiveEffectType

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Defines the types of active effects that a gem can have.

```
public enum GemActiveEffectType
```

Fields

Damage = 1

Heal = 2

MovesBonus = 3

None = 0

Enum GemEffectShieldType

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Defines the types of shields associated with gem effects.

```
public enum GemEffectShieldType
```

Fields

Fire = 3

Ice = 4

None = 0

Physical = 2

Pure = 1

Class GemItem

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a gem item in the game. This class is a ScriptableObject, allowing it to be used as a data asset in Unity.

```
[CreateAssetMenu(fileName = "MyData", menuName = "Custom/Gem Item", order = 1)]
public class GemItem : ScriptableObject
```

Inheritance

[object](#) ← Object ← ScriptableObject ← GemItem

Fields

ID

Unique identifier for the gem item.

```
public string ID
```

Field Value

[string](#)

gem_color

The color of the gem.

```
public Color gem_color
```

Field Value

Color

gem_description

Description of the gem.

[TextArea]

```
public string gem_description
```

Field Value

[string](#)

gem_effects

List of active effects associated with the gem.

```
public List<GemActiveEffect> gem_effects
```

Field Value

[List](#)<[GemActiveEffect](#)>

gem_frame_sprite

The sprite representing the frame of the gem.

```
public Sprite gem_frame_sprite
```

Field Value

Sprite

gem_name

Name of the gem.

```
public string gem_name
```

Field Value

[string](#)

gem_sprite

The sprite representing the gem.

```
public Sprite gem_sprite
```

Field Value

Sprite

gem_status_effects

List of status effects associated with the gem.

```
public List<GemStatusEffect> gem_status_effects
```

Field Value

[List](#) <[GemStatusEffect](#)>

item_tier

The tier of the gem item.

```
public GemItemTier item_tier
```

Field Value

[GemItemTier](#)

previousSprite

Stores the previous sprite before any changes.

```
private Sprite previousSprite
```

Field Value

Sprite

super_combos

List of super combos associated with the gem. Hidden in the Unity editor.

```
[HideInInspector]  
public List<GemSuperCombo> super_combos
```

Field Value

[List](#) <[GemSuperCombo](#)>

Methods

CalculateMatchEffects()

Calculates and updates the match effects for the gem based on the number of matches.

```
public void CalculateMatchEffects()
```

GenerateNewID()

Generates a new unique ID for the gem item.

```
[ContextMenu("Generate New ID")]  
public void GenerateNewID()
```

OnEnable()

Initialization method called when the script is loaded. Generates a new ID if necessary and stores the current sprite.

```
private void OnEnable()
```

OnValidate()

Method called when the script is loaded or a value is changed in the Inspector. Updates asset properties and triggers change events.

```
private void OnValidate()
```

Events

onGemItemChanged

Event triggered when the GemItem changes.

```
public event GemItem.OnGemItemChanged onGemItemChanged
```

Event Type

[GemItem.OnGemItemChanged](#)

Delegate GemItem.OnGemItemChanged

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Delegate for handling events when a GemItem changes.

```
public delegate void GemItem.OnGemItemChanged(GemItem gemItem)
```

Parameters

gemItem [GemItem](#)

The GemItem that has changed.

Enum GemItemTier

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Defines the tiers of gem items.

```
public enum GemItemTier
```

Fields

Common = 0

Legendary = 3

Mythical = 2

Rare = 1

Struct GemStatusEffect

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a status effect associated with a gem.

```
[Serializable]
public struct GemStatusEffect
```

Fields

basic_duration

The basic duration of the status effect.

```
public int basic_duration
```

Field Value

[int↗](#)

basic_touch_damage

The damage caused upon touch during the effect's duration.

```
public int basic_touch_damage
```

Field Value

[int↗](#)

effect_power

Power of the effect, ranging from 0.0 to 1.0.

```
[Range(0, 1)]  
public float effect_power
```

Field Value

[float](#)

effects_per_matched_gem

Effects amplified per matched gem.

```
public float effects_per_matched_gem
```

Field Value

[float](#)

status_effect

The type of status effect.

```
public GemStatusEffectType status_effect
```

Field Value

[GemStatusEffectType](#)

Properties

ID

Unique identifier for the status effect, combining various properties.

```
public string ID { get; }
```

Property Value

[string](#)

Enum GemStatusEffectType

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Defines the types of status effects that a gem can have.

```
public enum GemStatusEffectType
```

Fields

Burn = 2

Frozen = 3

Hexed = 5

None = 0

Normal = 1

Weakened = 4

Class GemSuperCombo

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a super combo associated with a gem, encompassing various combo types.

```
[Serializable]
public class GemSuperCombo
```

Inheritance

[object](#) ← GemSuperCombo

Fields

activateGemsOnBoardCombo

```
public GemSuperCombo.ActivateGemsOnBoard activateGemsOnBoardCombo
```

Field Value

[GemSuperCombo.ActivateGemsOnBoard](#)

burnCombo

```
public GemSuperCombo.BurnCombo burnCombo
```

Field Value

[GemSuperCombo.BurnCombo](#)

comboType

The type of super combo.

```
public GemSuperComboType comboType
```

Field Value

[GemSuperComboType](#)

copyEnemyDeckCombo

```
public GemSuperCombo.CopyEnemyDeck copyEnemyDeckCombo
```

Field Value

[GemSuperCombo.CopyEnemyDeck](#)

criticalDamageCombo

```
public GemSuperCombo.CriticalDamage criticalDamageCombo
```

Field Value

[GemSuperCombo.CriticalDamage](#)

damageEvasionCombo

```
public GemSuperCombo.DamageEvasion damageEvasionCombo
```

Field Value

[GemSuperCombo.DamageEvasion](#)

destroyEnemyGemsCombo

```
public GemSuperCombo.DestroyEnemyGems destroyEnemyGemsCombo
```

Field Value

[GemSuperCombo.DestroyEnemyGems](#)

extraMovesCombo

```
public GemSuperCombo.ExtraMoves extraMovesCombo
```

Field Value

[GemSuperCombo.ExtraMoves](#)

frostCombo

```
public GemSuperCombo.FrostCombo frostCombo
```

Field Value

[GemSuperCombo.FrostCombo](#)

hexCombo

```
public GemSuperCombo.HexCombo hexCombo
```

Field Value

[GemSuperCombo.HexCombo](#)

poisonCombo

```
public GemSuperCombo.Poison poisonCombo
```

Field Value

[GemSuperCombo.Poison](#)

regenerationCombo

```
public GemSuperCombo.RegenerationCombo regenerationCombo
```

Field Value

[GemSuperCombo.RegenerationCombo](#)

suddenDeathProtectionCombo

```
public GemSuperCombo.SuddenDeathProtection suddenDeathProtectionCombo
```

Field Value

[GemSuperCombo.SuddenDeathProtection](#)

Properties

comboData

Gets or sets the current super combo data based on the combo type. This property allows access to the specific super combo instance currently active or allows setting a new one.

```
public ISuperCombo comboData { get; set; }
```

Property Value

[ISuperCombo](#)

Methods

CreateComboData()

Creates the data for the super combo based on its type.

```
public void CreateComboData()
```

Class GemSuperCombo.ActivateGemsOnBoard

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents an 'Activate Gems On Board' Combo in the Gem Super Combo system. This combo allows for the activation of a portion of the gems on the board without needing to match them.

```
[Serializable]  
public class GemSuperCombo.ActivateGemsOnBoard : ISuperCombo
```

Inheritance

[object](#) ← GemSuperCombo.ActivateGemsOnBoard

Implements

[ISuperCombo](#)

Constructors

ActivateGemsOnBoard()

Default constructor for the Activate Gems On Board combo.

```
public ActivateGemsOnBoard()
```

ActivateGemsOnBoard(float, int)

Constructs an Activate Gems On Board combo with specified parameters.

```
public ActivateGemsOnBoard(float part_of_gems_to_activate, int combo_count_activation)
```

Parameters

part_of_gems_to_activate [float](#)

Proportion of the board's gems to be activated.

combo_count_activation [int](#)

Number of activations required for the combo.

Fields

combo_count_activation

The number of combo activations required to trigger the Activate Gems On Board combo.

```
[Range(3, 12)]  
public int combo_count_activation
```

Field Value

[int](#)

part_of_gems_to_activate

The proportion of gems on the board that will be activated by this combo.

```
[Range(0, 1)]  
public float part_of_gems_to_activate
```

Field Value

[float](#)

Methods

GetCount()

Retrieves the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int](#) ↗

The number of activations needed for the combo.

MakeCopy()

Creates a copy of the current Activate Gems On Board combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of ActivateGemsOnBoard with the same settings.

Class GemSuperCombo.BurnCombo

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a Burn Combo in the Gem Super Combo system, which applies a burning effect to a part of the board.

```
[Serializable]  
public class GemSuperCombo.BurnCombo : ISuperCombo
```

Inheritance

[object](#) ← GemSuperCombo.BurnCombo

Implements

[ISuperCombo](#)

Constructors

BurnCombo()

Default constructor for the Burn Combo.

```
public BurnCombo()
```

BurnCombo(int, float, int, float)

Constructs a Burn Combo with specified parameters.

```
public BurnCombo(int duration, float part_of_board, int combo_count_activation,  
float touch_damage)
```

Parameters

duration [int](#)

Duration of the effect in turns.

part_of_board [float](#)

Proportion of the board affected by the combo.

combo_count_activation [int](#)

Number of activations required for the combo.

touch_damage [float](#)

Damage dealt upon touch during the effect.

Fields

combo_count_activation

The number of combo activations required to trigger the Burn Combo.

```
[Range(3, 12)]  
public int combo_count_activation
```

Field Value

[int](#)

duration

The duration of the Burn Combo effect in turns.

```
[Range(1, 10)]  
public int duration
```

Field Value

[int](#)

part_of_board

The proportion of the board affected by the Burn Combo.

```
[Range(0, 1)]  
public float part_of_board
```

Field Value

[float](#)

touch_damage

The damage dealt upon touch during the effect's duration.

```
public float touch_damage
```

Field Value

[float](#)

Methods

GetCount()

Retrieves the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int](#)

The number of activations needed for the combo.

MakeCopy()

Creates a copy of the current Burn Combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of BurnCombo with the same settings.

Class GemSuperCombo.CopyEnemyDeck

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a 'Copy Enemy Deck' Combo in the Gem Super Combo system. This combo allows the player to copy the enemy's deck or abilities for a certain duration.

```
[Serializable]  
public class GemSuperCombo.CopyEnemyDeck : ISuperCombo
```

Inheritance

[object](#) ← GemSuperCombo.CopyEnemyDeck

Implements

[ISuperCombo](#)

Constructors

CopyEnemyDeck()

Default constructor for the Copy Enemy Deck combo.

```
public CopyEnemyDeck()
```

CopyEnemyDeck(int, int)

Constructs a Copy Enemy Deck combo with specified parameters.

```
public CopyEnemyDeck(int duration_in_turns, int combo_count_activation)
```

Parameters

duration_in_turns [int](#)

The duration in turns for the combo effect.

combo_count_activation [int↗](#)

Number of activations required for the combo.

Fields

combo_count_activation

The number of combo activations required to trigger the Copy Enemy Deck combo.

```
[Range(3, 12)]  
public int combo_count_activation
```

Field Value

[int↗](#)

duration_in_turns

The duration, in turns, for which the enemy's deck or abilities are copied.

```
[Range(1, 10)]  
public int duration_in_turns
```

Field Value

[int↗](#)

Methods

GetCount()

Retrieves the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int](#) ↗

The number of activations needed for the combo.

MakeCopy()

Creates a copy of the current Copy Enemy Deck combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of CopyEnemyDeck with the same settings.

Class GemSuperCombo.CriticalDamage

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a 'Critical Damage' Combo in the Gem Super Combo system. This combo amplifies damage for a certain number of attacks, increasing their critical hit potential.

```
[Serializable]
public class GemSuperCombo.CriticalDamage : ISuperCombo
```

Inheritance

[object](#) ← GemSuperCombo.CriticalDamage

Implements

[ISuperCombo](#)

Constructors

CriticalDamage()

Default constructor for the Critical Damage combo.

```
public CriticalDamage()
```

CriticalDamage(int, float, int)

Constructs a Critical Damage combo with specified parameters.

```
public CriticalDamage(int amplified_attacks_amount, float critical_damage_amplification,
int combo_count_activation)
```

Parameters

`amplified_attacks_amount` [int](#)

The number of attacks to be amplified.

`critical_damage_amplification` [float](#)

The amplification factor for critical damage.

`combo_count_activation` [int](#)

Number of activations required for the combo.

Fields

`amplified_attacks_amount`

The number of attacks that will be amplified in terms of damage.

```
[Range(1, 10)]  
public int amplified_attacks_amount
```

Field Value

[int](#)

`combo_count_activation`

The number of combo activations required to trigger the Critical Damage combo.

```
[Range(3, 12)]  
public int combo_count_activation
```

Field Value

[int](#)

`critical_damage_amplification`

The amplification factor for critical damage.

```
[Range(1, 10)]
```

```
public float critical_damage_amplification
```

Field Value

[float](#)

Methods

GetCount()

Retrieves the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int](#)

The number of activations needed for the combo.

MakeCopy()

Creates a copy of the current Critical Damage combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of CriticalDamage with the same settings.

Class GemSuperCombo.DamageEvasion

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a 'Damage Evasion' Combo in the Gem Super Combo system. This combo allows the player to evade a certain number of attacks.

```
[Serializable]  
public class GemSuperCombo.DamageEvasion : ISuperCombo
```

Inheritance

[object](#) ← GemSuperCombo.DamageEvasion

Implements

[ISuperCombo](#)

Constructors

DamageEvasion()

Default constructor for the Damage Evasion combo.

```
public DamageEvasion()
```

DamageEvasion(int, int)

Constructs a Damage Evasion combo with specified parameters.

```
public DamageEvasion(int count_of_attacks_to_evade, int combo_count_activation)
```

Parameters

count_of_attacks_to_evade [int](#)

The number of attacks that can be evaded.

combo_count_activation [int↗](#)

Number of activations required for the combo.

Fields

combo_count_activation

The number of combo activations required to trigger the Damage Evasion combo.

```
[Range(3, 12)]  
public int combo_count_activation
```

Field Value

[int↗](#)

count_of_attacks_to_evade

The number of attacks that can be evaded with this combo.

```
[Range(1, 10)]  
public int count_of_attacks_to_evade
```

Field Value

[int↗](#)

Methods

GetCount()

Retrieves the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int↗](#)

The number of activations needed for the combo.

MakeCopy()

Creates a copy of the current Damage Evasion combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of DamageEvasion with the same settings.

Class GemSuperCombo.DestroyEnemyGems

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a 'Destroy Enemy Gems' Combo in the Gem Super Combo system. This combo allows for a portion of the enemy's gems on the board to be destroyed.

```
[Serializable]  
public class GemSuperCombo.DestroyEnemyGems : ISuperCombo
```

Inheritance

[Object](#) ← GemSuperCombo.DestroyEnemyGems

Implements

[ISuperCombo](#)

Constructors

DestroyEnemyGems()

Default constructor for the Destroy Enemy Gems combo.

```
public DestroyEnemyGems()
```

DestroyEnemyGems(float, int)

Constructs a Destroy Enemy Gems combo with specified parameters.

```
public DestroyEnemyGems(float part_of_board, int combo_count_activation)
```

Parameters

part_of_board [float](#)

Proportion of the board's enemy gems to be destroyed.

combo_count_activation [int](#)

Number of activations required for the combo.

Fields

combo_count_activation

The number of combo activations required to trigger the Destroy Enemy Gems combo.

```
[Range(3, 12)]  
public int combo_count_activation
```

Field Value

[int](#)

part_of_board

The proportion of the board where enemy gems will be targeted for destruction.

```
[Range(0, 1)]  
public float part_of_board
```

Field Value

[float](#)

Methods

GetCount()

Retrieves the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int](#) ↗

The number of activations needed for the combo.

MakeCopy()

Creates a copy of the current Destroy Enemy Gems combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of DestroyEnemyGems with the same settings.

Class GemSuperCombo.ExtraMoves

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents an 'Extra Moves' Combo in the Gem Super Combo system, which grants additional moves to the player.

```
[Serializable]  
public class GemSuperCombo.ExtraMoves : ISuperCombo
```

Inheritance

[object](#) ← GemSuperCombo.ExtraMoves

Implements

[ISuperCombo](#)

Constructors

ExtraMoves()

Default constructor for the Extra Moves combo.

```
public ExtraMoves()
```

ExtraMoves(int, int)

Constructs an Extra Moves combo with specified parameters.

```
public ExtraMoves(int moves_count, int combo_count_activation)
```

Parameters

`moves_count` [int](#)

The number of extra moves granted by the combo.

combo_count_activation [int↗](#)

Number of activations required for the combo.

Fields

combo_count_activation

The number of combo activations required to trigger the Extra Moves combo.

```
[Range(3, 12)]  
public int combo_count_activation
```

Field Value

[int↗](#)

moves_count

The number of extra moves granted by this combo.

```
[Range(1, 5)]  
public int moves_count
```

Field Value

[int↗](#)

Methods

GetCount()

Retrieves the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int](#) ↗

The number of activations needed for the combo.

MakeCopy()

Creates a copy of the current Extra Moves combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of ExtraMoves with the same settings.

Class GemSuperCombo.FrostCombo

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a Frost Combo as part of the Gem Super Combo system. This combo can affect a part of the game board for a certain duration based on the combo activation count.

```
[Serializable]  
public class GemSuperCombo.FrostCombo : ISuperCombo
```

Inheritance

[object](#) ← GemSuperCombo.FrostCombo

Implements

[ISuperCombo](#)

Constructors

FrostCombo()

Default constructor for the Frost Combo.

```
public FrostCombo()
```

FrostCombo(int, float, int)

Constructor for creating a Frost Combo with specific parameters.

```
public FrostCombo(int duration, float part_of_board, int combo_count_activation)
```

Parameters

duration [int](#)

Duration of the combo effect in turns.

part_of_board [float](#)

Proportion of the board affected by the combo.

combo_count_activation [int](#)

Number of activations required to trigger the combo.

Fields

combo_count_activation

The number of combo activations required to trigger the Frost Combo.

[Range(3, 12)]

```
public int combo_count_activation
```

Field Value

[int](#)

duration

Duration of the Frost Combo effect in turns.

[Range(1, 10)]

```
public int duration
```

Field Value

[int](#)

part_of_board

The proportion of the game board affected by the Frost Combo.

[Range(0, 1)]

```
public float part_of_board
```

Field Value

[float](#)

Methods

GetCount()

Gets the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int](#)

The number of combo activations needed.

MakeCopy()

Creates a copy of the current Frost Combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of FrostCombo with the same settings.

Class GemSuperCombo.HexCombo

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a Hex Combo in the Gem Super Combo system, which applies a hex effect on a part of the board.

```
[Serializable]  
public class GemSuperCombo.HexCombo : ISuperCombo
```

Inheritance

[object](#) ← GemSuperCombo.HexCombo

Implements

[ISuperCombo](#)

Constructors

HexCombo()

Default constructor for the Hex Combo.

```
public HexCombo()
```

HexCombo(int, float, int, GemItem)

Constructs a Hex Combo with specified parameters.

```
public HexCombo(int duration, float part_of_board, int combo_count_activation,  
GemItem turn_into_gem)
```

Parameters

duration [int](#)

Duration of the effect in turns.

part_of_board [float](#)

Proportion of the board affected by the combo.

combo_count_activation [int](#)

Number of activations required for the combo.

turn_into_gem [GemItem](#)

The GemItem to turn affected parts into.

Fields

combo_count_activation

The number of combo activations required to trigger the Hex Combo.

```
[Range(3, 12)]  
public int combo_count_activation
```

Field Value

[int](#)

duration

The duration of the Hex Combo effect in turns.

```
[Range(1, 10)]  
public int duration
```

Field Value

[int](#)

part_of_board

The proportion of the board affected by the Hex Combo.

```
[Range(0, 1)]  
public float part_of_board
```

Field Value

[float](#)

turn_into_gem

The specific GemItem to which affected board parts are turned into during the combo effect.

```
public GemItem turn_into_gem
```

Field Value

[GemItem](#)

Methods

GetCount()

Retrieves the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int](#)

The number of activations needed for the combo.

MakeCopy()

Creates a copy of the current Hex Combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of HexCombo with the same settings.

Class GemSuperCombo.Poison

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a 'Poison' Combo in the Gem Super Combo system. This combo inflicts poison damage over a set duration, dealing damage each turn.

```
[Serializable]  
public class GemSuperCombo.Poison : ISuperCombo
```

Inheritance

[object](#) ← GemSuperCombo.Poison

Implements

[ISuperCombo](#)

Constructors

Poison()

Default constructor for the Poison combo.

```
public Poison()
```

Poison(int, float, int)

Constructs a Poison combo with specified parameters.

```
public Poison(int turns_duration, float hp_damage_per_turn, int combo_count_activation)
```

Parameters

`turns_duration` [int](#)

The duration of the poison effect in turns.

hp_damage_per_turn [float](#)

The damage per turn inflicted by the poison.

combo_count_activation [int](#)

Number of activations required for the combo.

Fields

combo_count_activation

The number of combo activations required to trigger the Poison combo.

[Range(3, 12)]

```
public int combo_count_activation
```

Field Value

[int](#)

hp_damage_per_turn

The amount of health points damage per turn inflicted by the poison.

```
public float hp_damage_per_turn
```

Field Value

[float](#)

turns_duration

The duration of the poison effect in turns.

[Range(1, 10)]

```
public int turns_duration
```

Field Value

[int↗](#)

Methods

GetCount()

Retrieves the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int↗](#)

The number of activations needed for the combo.

MakeCopy()

Creates a copy of the current Poison combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of Poison with the same settings.

Class GemSuperCombo.RegenerationCombo

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a Regeneration Combo in the Gem Super Combo system, which provides health regeneration over a set number of turns.

```
[Serializable]  
public class GemSuperCombo.RegenerationCombo : ISuperCombo
```

Inheritance

[object](#) ← GemSuperCombo.RegenerationCombo

Implements

[ISuperCombo](#)

Constructors

RegenerationCombo()

Default constructor for the Regeneration Combo.

```
public RegenerationCombo()
```

RegenerationCombo(int, float, int)

Constructs a Regeneration Combo with specified parameters.

```
public RegenerationCombo(int turns_duration, float hp_regeneration_per_turn,  
int combo_count_activation)
```

Parameters

`turns_duration` [int](#)

Duration of the effect in turns.

`hp_regeneration_per_turn` [float](#)

The amount of health points regenerated per turn.

`combo_count_activation` [int](#)

The number of activations required for the combo.

Fields

combo_count_activation

The number of combo activations required to trigger the Regeneration Combo.

[Range(3, 12)]

`public int combo_count_activation`

Field Value

[int](#)

hp_regeneration_per_turn

The amount of health points regenerated per turn.

`public float hp_regeneration_per_turn`

Field Value

[float](#)

turns_duration

The duration of the regeneration effect in turns.

[Range(1, 10)]

`public int turns_duration`

Field Value

[int↗](#)

Methods

GetCount()

Retrieves the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int↗](#)

The number of activations needed for the combo.

MakeCopy()

Creates a copy of the current Regeneration Combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of RegenerationCombo with the same settings.

Class

GemSuperCombo.SuddenDeathProtection

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Represents a 'Sudden Death Protection' Combo in the Gem Super Combo system. This combo provides protection against sudden defeat for a specified number of turns.

```
[Serializable]  
public class GemSuperCombo.SuddenDeathProtection : ISuperCombo
```

Inheritance

[object](#) ← GemSuperCombo.SuddenDeathProtection

Implements

[ISuperCombo](#)

Constructors

SuddenDeathProtection()

Default constructor for the Sudden Death Protection combo.

```
public SuddenDeathProtection()
```

SuddenDeathProtection(int, int)

Constructs a Sudden Death Protection combo with specified parameters.

```
public SuddenDeathProtection(int turns_duration, int combo_count_activation)
```

Parameters

turns_duration [int](#)

Duration of protection in turns.

combo_count_activation [int↗](#)

Number of activations required for the combo.

Fields

combo_count_activation

The number of combo activations required to trigger the Sudden Death Protection combo.

```
[Range(3, 12)]  
public int combo_count_activation
```

Field Value

[int↗](#)

turns_duration

The duration of protection against sudden death, measured in turns.

```
[Range(1, 20)]  
public int turns_duration
```

Field Value

[int↗](#)

Methods

GetCount()

Retrieves the count of activations required for the combo.

```
public int GetCount()
```

Returns

[int](#) ↗

The number of activations needed for the combo.

MakeCopy()

Creates a copy of the current Sudden Death Protection combo.

```
public ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of SuddenDeathProtection with the same settings.

Enum GemSuperComboType

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Defines the types of super combos available for gems.

```
public enum GemSuperComboType
```

Fields

ActivateGemsOnBoard = 8

Activate Gems On Board: Automatically activates or triggers certain gem effects without the need for matching.

BurnCombo = 2

Burn Combo: Sets a part of the board on fire, causing damage if player touches burned gem.

CopyEnemyDeck = 7

Copy Enemy Deck: Temporarily copies the opponent's deck.

CriticalDamage = 9

Critical Damage: Increases the damage of attacks for a certain number of strikes.

DamageEvasion = 6

Damage Evasion: Allows the player to evade a certain number of attacks or damage instances.

DestroyEnemyGems = 10

Destroy Enemy Gems: Targets and destroys specific gems or elements controlled by the opponent.

ExtraMoves = 4

Extra Moves: Grants additional moves or turns to the player.

FrostCombo = 1

Frost Combo: Freezes a portion of the board or certain elements for a specified duration.

HexCombo = 3

Hex Combo: Transforms certain elements on the board into other elements or has a debilitating effect on the opponent.

None = 0

Represents the absence of a super combo.

Poison = 12

Poison: Inflicts a poison effect on the opponent, causing damage over a number of turns.

Regeneration = 11

Regeneration: Gradually restores health or resources over a number of turns.

SuddenDeathProtection = 5

Sudden Death Protection: Provides a shield or protection against fatal damage for a limited number of turns.

Interface ISuperCombo

Namespace: [TaleOfTiles.Card.Gems](#)

Assembly: Assembly-CSharp.dll

Interface for defining super combo behavior.

```
public interface ISuperCombo
```

Methods

GetCount()

Gets the count of something related to the combo (implementation-specific).

```
int GetCount()
```

Returns

[int](#)

An integer representing the count.

MakeCopy()

Creates a copy of the super combo.

```
ISuperCombo MakeCopy()
```

Returns

[ISuperCombo](#)

A new instance of a class implementing ISuperCombo.

Namespace TaleOfTiles.Data

Classes

[DataManager](#)

Manages game data, including player inventory, experience, gold, and configurations for levels, enemies, and journey maps. Utilizes Singleton pattern to ensure a single instance.

[EnemyObjectManager](#)

Manages operations related to EnemyObject bundles, including loading and retrieval.

[JourneyObjectManager](#)

Manages operations related to JourneyObjects, including loading, saving, and deleting.

[JourneyObjectManager.JourneyObjectSerializable](#)

[LevelObjectManager](#)

Manages operations related to LevelObject bundles, including loading and retrieval.

Structs

[DataManager.ActiveItemJSON](#)

[DataManager.PlayerInventoryJSON](#)

Class DataManager

Namespace: [TaleOfTiles.Data](#)

Assembly: Assembly-CSharp.dll

Manages game data, including player inventory, experience, gold, and configurations for levels, enemies, and journey maps. Utilizes Singleton pattern to ensure a single instance.

```
public class DataManager : Singleton<DataManager>
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton<DataManager>](#) ← DataManager

Inherited Members

[Singleton<DataManager>.m_instance](#) , [Singleton<DataManager>.Instance](#) ,
[Singleton<DataManager>.Awake\(\)](#) , [SingletonBase.AllSingletons](#)

Fields

cashed_enemies

Loads a specific enemy object by its ID.

```
private List<EnemyObject> cashed_enemies
```

Field Value

[List<EnemyObject>](#)

Returns the EnemyObject if found; otherwise, null.

cashed_journey_maps

```
private List<JourneyMap> cashed_journey_maps
```

Field Value

[List ↴ <JourneyMap>](#)

cashed_levels

Loads a specific level object by its ID.

```
public List<LevelObject> cashed_levels
```

Field Value

[List ↴ <LevelObject>](#)

Returns the LevelObject if found; otherwise, null.

configFolderPath

```
private const string configFolderPath = "ScriptableObjects/Maps/Configs/"
```

Field Value

[string ↴](#)

enemy_object_manager

```
public EnemyObjectManager enemy_object_manager
```

Field Value

[EnemyObjectManager](#)

journey_object_manager

```
public JourneyObjectManager journey_object_manager
```

Field Value

[JourneyObjectManager](#)

level_object_manager

```
public LevelObjectManager level_object_manager
```

Field Value

[LevelObjectManager](#)

Methods

AddNewPlayerCard(string)

Adds a new card to the player's card collection if it doesn't already exist.

```
public void AddNewPlayerCard(string id)
```

Parameters

id [string](#) ↗

The unique identifier of the card.

AddNewPlayerItem(string)

Adds a new item to the player's inventory.

```
public void AddNewPlayerItem(string id)
```

Parameters

id [string](#)

The ID of the item to be added.

DeleteConfig(string)

Deletes a configuration file based on the provided config ID.

```
public void DeleteConfig(string configID)
```

Parameters

configID [string](#)

The unique identifier of the config to be deleted.

DownloadAllCardsCollection()

Downloads the entire collection of cards from resources.

```
public List<CardObject> DownloadAllCardsCollection()
```

Returns

[List](#)<[CardObject](#)>

A list of all card objects.

DownloadAllEquipmentInGame()

Loads all equipment items from resources and stores them in a list.

```
public List<EquipmentItem> DownloadAllEquipmentInGame()
```

Returns

[List](#)<[EquipmentItem](#)>

A list of all equipment items available in the game.

GetPlayerDeck()

Retrieves the player's deck.

```
public void GetPlayerDeck()
```

GetPlayerDeckFilePath()

Retrieves the file path for the player's deck. Creates the file if it doesn't exist.

```
public string GetPlayerDeckFilePath()
```

Returns

string ↗

GetPlayerExp()

Retrieves the player's experience points from PlayerPrefs. If not set, initializes to a default value of 100.

```
public int GetPlayerExp()
```

Returns

int ↗

Current experience points of the player.

GetPlayerGold()

Gets the player's current gold amount, initializing it if not set.

```
public int GetPlayerGold()
```

Returns

[int](#)

Current gold amount of the player.

GetPlayerInventoryFilePath()

Gets the file path for the player's inventory file.

```
public string GetPlayerInventoryFilePath()
```

Returns

[string](#)

The file path for the player's inventory.

LoadActivePlayerItems()

Loads the player's active items from a file.

```
public Dictionary<ItemPlace, string> LoadActivePlayerItems()
```

Returns

[Dictionary](#)<[ItemPlace](#), [string](#)>

A dictionary mapping item places to their IDs.

LoadAllEnemyObjects()

```
public EnemyObject[] LoadAllEnemyObjects()
```

Returns

[EnemyObject](#)[]

LoadAllJourneyMaps()

Loads all JourneyMap objects. If not previously loaded, retrieves them from resources.

```
public JourneyMap[] LoadAllJourneyMaps()
```

Returns

[JourneyMap\[\]](#)

An array of all loaded JourneyMap objects.

LoadAllLevelObjects()

Loads all level objects stored in resources.

```
public LevelObject[] LoadAllLevelObjects()
```

Returns

[LevelObject\[\]](#)

An array of all loaded LevelObject instances.

LoadAllPlayerItems()

Loads the player's item inventory from a file.

```
public string[] LoadAllPlayerItems()
```

Returns

[string\[\]](#)

An array of item IDs representing the player's items.

LoadConfig(string)

Loads a JourneyMapConfig from a JSON file based on the provided config ID.

```
public JourneyMapConfig LoadConfig(string configID)
```

Parameters

[configID string](#)

The unique identifier of the config to load.

Returns

[JourneyMapConfig](#)

Returns the loaded JourneyMapConfig if found; otherwise, null.

LoadEnemyObject(string)

```
public EnemyObject LoadEnemyObject(string ID)
```

Parameters

[ID string](#)

Returns

[EnemyObject](#)

LoadJourneyMap(string)

Loads a specific journey map by its ID.

```
public JourneyMap LoadJourneyMap(string map_id)
```

Parameters

`map_id` [string](#)

The unique identifier of the journey map.

Returns

[JourneyMap](#)

Returns the JourneyMap if found; otherwise, null.

LoadLevelObject(string)

`public LevelObject LoadLevelObject(string ID)`

Parameters

`ID` [string](#)

Returns

[LevelObject](#)

LoadPlayerCardsCollection()

Loads the player's card collection from a file.

`public Dictionary<string, bool> LoadPlayerCardsCollection()`

Returns

[Dictionary](#)<[string](#), [bool](#)>

A dictionary of card IDs and their activation statuses.

Prepare()

Initializes the DataManager, loading various game objects and configurations.

```
public override IEnumerator Prepare()
```

Returns

[IEnumerator](#)

Coroutine for asynchronous operation.

RemovePlayerCard(string)

Removes a specific card from the player's collection by its ID.

```
public void RemovePlayerCard(string id)
```

Parameters

[id string](#)

The unique identifier of the card to be removed.

RemovePlayerItem(string)

Removes an item from the player's inventory.

```
public void RemovePlayerItem(string id)
```

Parameters

[id string](#)

The ID of the item to be removed.

SaveConfig(JourneyMapConfig)

Saves a JourneyMapConfig object as a JSON file.

```
public void SaveConfig(JourneyMapConfig config)
```

Parameters

config [JourneyMapConfig](#)

The JourneyMapConfig object to be saved.

SavePlayerCard(Dictionary<string, bool>)

Saves the player's card collection to a file.

```
private void SavePlayerCard(Dictionary<string, bool> data)
```

Parameters

data [Dictionary](#)<[string](#), [bool](#)>

The card data to save.

SetActiveItem(string, ItemPlace)

Sets an item as active in the player's inventory.

```
public void SetActiveItem(string id, ItemPlace itemPlace)
```

Parameters

id [string](#)

The ID of the item to set as active.

itemPlace [ItemPlace](#)

The place to set the item active in.

SetPlayerCardStatus(string, bool)

Sets the status of a specific card in the player's collection.

```
public void SetPlayerCardStatus(string id, bool status)
```

Parameters

id [string](#)

The unique identifier of the card.

status [bool](#)

The new status to be set for the card.

UpdateConfig(JourneyMapConfig)

Updates a JourneyMapConfig by saving the modified version.

```
public void UpdateConfig(JourneyMapConfig config)
```

Parameters

config [JourneyMapConfig](#)

The JourneyMapConfig object to be updated.

UpdatePlayerExp(int)

Updates the player's experience points in PlayerPrefs.

```
public void UpdatePlayerExp(int xp_change)
```

Parameters

xp_change [int](#)

The amount by which to change the experience points.

UpdatePlayerGold(int)

Updates the player's gold amount in PlayerPrefs.

```
public void UpdatePlayerGold(int gold_change)
```

Parameters

gold_change [int](#)

The amount by which to change the gold.

Struct DataManager.ActiveItemJSON

Namespace: [TaleOfTiles.Data](#)

Assembly: Assembly-CSharp.dll

```
private struct DataManager.ActiveItemJSON
```

Fields

ID

```
public string ID
```

Field Value

[string](#)

ItemPlace

```
public ItemPlace ItemPlace
```

Field Value

[ItemPlace](#)

Struct DataManager.PlayerInventoryJSON

Namespace: [TaleOfTiles.Data](#)

Assembly: Assembly-CSharp.dll

```
private struct DataManager.PlayerInventoryJSON
```

Fields

active_items

```
public DataManager.ActiveItemJSON[] active_items
```

Field Value

[ActiveItemJSON\[\]](#)

all_items

```
public string[] all_items
```

Field Value

[string\[\]](#)

Class EnemyObjectManager

Namespace: [TaleOfTiles.Data](#)

Assembly: Assembly-CSharp.dll

Manages operations related to EnemyObject bundles, including loading and retrieval.

```
public class EnemyObjectManager
```

Inheritance

[object](#) ← EnemyObjectManager

Fields

cashed_bundles

```
private List<EnemiesObjectsBundle> cashed_bundles
```

Field Value

[List](#) <[EnemiesObjectsBundle](#)>

resourcePath

```
private const string resourcePath = "ScriptableObjects/Enemies/Bundles/"
```

Field Value

[string](#)

Methods

LoadAllBundleObjects()

```
public EnemiesObjectsBundle[] LoadAllBundleObjects()
```

Returns

[EnemiesObjectsBundle\[\]](#)

LoadEnemiesBundleObject(string)

```
public EnemiesObjectsBundle LoadEnemiesBundleObject(string bundle_id)
```

Parameters

bundle_id [string](#)

Returns

[EnemiesObjectsBundle](#)

Class JourneyObjectManager

Namespace: [TaleOfTiles.Data](#)

Assembly: Assembly-CSharp.dll

Manages operations related to JourneyObjects, including loading, saving, and deleting.

```
public class JourneyObjectManager
```

Inheritance

[object](#) ← JourneyObjectManager

Fields

resourcePath

```
private const string resourcePath = "ScriptableObjects/Journeys/Configs/"
```

Field Value

[string](#)

Methods

DeleteJourneyObject(string)

Deletes a JourneyObject configuration file based on the given ID.

```
public void DeleteJourneyObject(string journeyID)
```

Parameters

journeyID [string](#)

The ID of the JourneyObject to delete.

GetPathFromSprite(Sprite)

```
private string GetPathFromSprite(Sprite sprite)
```

Parameters

sprite Sprite

Returns

[string](#)

LoadAllJourneyObjects()

Loads all JourneyObject configurations from the persistent data path.

```
public JourneyObject[] LoadAllJourneyObjects()
```

Returns

[JourneyObject\[\]](#)

An array of all JourneyObjects loaded.

LoadJourneyObject(string)

Loads a specific JourneyObject based on the given ID.

```
public JourneyObject LoadJourneyObject(string journeyID)
```

Parameters

journeyID [string](#)

The ID of the JourneyObject to load.

Returns

[JourneyObject](#)

The loaded JourneyObject, or null if not found.

LoadSpriteFromPath(string)

```
private Sprite LoadSpriteFromPath(string path)
```

Parameters

path [string](#)

Returns

Sprite

SaveJourneyObject(JourneyObject)

Saves a given JourneyObject as a JSON file in the persistent data path.

```
public void SaveJourneyObject(JourneyObject journeyObject)
```

Parameters

journeyObject [JourneyObject](#)

The JourneyObject to save.

Class

JourneyObjectManager.JourneyObjectSerializable

Namespace: [TaleOfTiles.Data](#)

Assembly: Assembly-CSharp.dll

```
[Serializable]  
public class JourneyObjectManager.JourneyObjectSerializable
```

Inheritance

[Object](#) ← JourneyObjectManager.JourneyObjectSerializable

Fields

ID

```
public string ID
```

Field Value

[string](#)

current_block_id

```
public string current_block_id
```

Field Value

[string](#)

enemies_bundle_id

```
public string enemies_bundle_id
```

Field Value

[string](#) ↗

journey_description

```
public string journey_description
```

Field Value

[string](#) ↗

journey_hardness

```
public JourneyObject.JourneyHardness journey_hardness
```

Field Value

[JourneyObject.JourneyHardness](#)

journey_image_path

```
public string journey_image_path
```

Field Value

[string](#) ↗

journey_name

```
public string journey_name
```

Field Value

[string](#) ↗

levels_bundle_id

```
public string levels_bundle_id
```

Field Value

[string](#) ↗

map_config_id

```
public string map_config_id
```

Field Value

[string](#) ↗

map_id

```
public string map_id
```

Field Value

[string](#) ↗

Class LevelObjectManager

Namespace: [TaleOfTiles.Data](#)

Assembly: Assembly-CSharp.dll

Manages operations related to LevelObject bundles, including loading and retrieval.

```
public class LevelObjectManager
```

Inheritance

[object](#) ← LevelObjectManager

Fields

cashed_level_object_bundles

```
private List<LevelObjectsBundle> cashed_level_object_bundles
```

Field Value

[List](#) <[LevelObjectsBundle](#)>

resourcePath

```
private const string resourcePath = "ScriptableObjects/Levels/Bundles/"
```

Field Value

[string](#)

Methods

LoadAllBundleObjects()

```
public LevelObjectsBundle[] LoadAllBundleObjects()
```

Returns

[LevelObjectsBundle\[\]](#)

LoadLevelBundleObject(string)

```
public LevelObjectsBundle LoadLevelBundleObject(string bundle_id)
```

Parameters

bundle_id [string](#)

Returns

[LevelObjectsBundle](#)

Namespace TaleOfTiles.Inventory

Classes

[EquipSlotUI](#)

Handles the display and functionality of an equipment slot in the UI.

[EquipmentHolderManager](#)

Manages the equipment items in the game, including all available items, player's equipment, and active equipment. Inherits from Singleton to ensure only one instance exists.

[InventoryPageUI](#)

Manages the inventory page UI, displaying player stats, inventory slots, and experience progress.

[ItemSkillInventoryPanelUI](#)

Manages the display of a skill in the item skill inventory panel.

[ItemsPanelUI](#)

Manages the items panel in the game UI, displaying information about items and skills.

Delegates

[EquipSlotUI.ItemChangedEvent](#)

Delegate for the event when the item in the slot changes.

Class EquipSlotUI

Namespace: [TaleOfTiles.Inventory](#)

Assembly: Assembly-CSharp.dll

Handles the display and functionality of an equipment slot in the UI.

```
public class EquipSlotUI : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← EquipSlotUI

Fields

common_frame

The sprite for the common frame.

```
public Sprite common_frame
```

Field Value

Sprite

item

The item to be displayed in the slot.

```
public EquipmentItem item
```

Field Value

[EquipmentItem](#)

item_frame

The UI component for displaying the item's frame.

```
public Image item_frame
```

Field Value

Image

item_image_component

The UI component for displaying the item's image.

```
public Image item_image_component
```

Field Value

Image

item_place

The place of the item.

```
public ItemPlace item_place
```

Field Value

[ItemPlace](#)

item_selection_effect_image

The UI component for displaying the item's selection effect.

```
public Image item_selection_effect_image
```

Field Value

Image

legendary_frame

The sprite for the legendary frame.

```
public Sprite legendary_frame
```

Field Value

Sprite

mythical_frame

The sprite for the mythical frame.

```
public Sprite mythical_frame
```

Field Value

Sprite

rare_frame

The sprite for the rare frame.

```
public Sprite rare_frame
```

Field Value

Sprite

uncommon_frame

The sprite for the uncommon frame.

```
public Sprite uncommon_frame
```

Field Value

Sprite

Methods

ActivateItem()

Activates the item in the slot.

```
[ContextMenu("Activate Item")]
public void ActivateItem()
```

ClearSlot()

Clears the slot.

```
public void ClearSlot()
```

DeSelect()

Deselects the item in the slot.

```
public void DeSelect()
```

LoadItem()

Loads the item into the slot.

```
public void LoadItem()
```

OnValidate()

Validates the object in the editor.

```
public void OnValidate()
```

Select()

Selects the item in the slot.

```
public void Select()
```

SetItem(EquipmentItem)

Sets a new item for the slot.

```
public void SetItem(EquipmentItem new_item)
```

Parameters

new_item EquipmentItem

The new item to be set.

ShowAvailableItems()

Shows the available items for this slot.

```
public void ShowAvailableItems()
```

Events

slot_item_changed_event

Event triggered when the item in the slot changes.

```
[HideInInspector]  
public event EquipSlotUI.ItemChangedEvent slot_item_changed_event
```

Event Type

[EquipSlotUI.ItemChangedEvent](#)

Delegate EquipSlotUI.ItemChangedEvent

Namespace: [TaleOfTiles.Inventory](#)

Assembly: Assembly-CSharp.dll

Delegate for the event when the item in the slot changes.

```
[HideInInspector]  
public delegate void EquipSlotUI.ItemChangedEvent()
```

Class EquipmentHolderManager

Namespace: [TaleOfTiles.Inventory](#)

Assembly: Assembly-CSharp.dll

Manages the equipment items in the game, including all available items, player's equipment, and active equipment. Inherits from Singleton to ensure only one instance exists.

```
public class EquipmentHolderManager : Singleton<EquipmentHolderManager>
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton<EquipmentHolderManager>](#) ← EquipmentHolderManager

Inherited Members

[Singleton<EquipmentHolderManager>.m_instance](#) , [Singleton<EquipmentHolderManager>.Instance](#) , [Singleton<EquipmentHolderManager>.Awake\(\)](#) , [SingletonBase.AllSingletons](#)

Fields

all_game_equipment

```
[SerializeField]  
private List<EquipmentItem> all_game_equipment
```

Field Value

[List<EquipmentItem>](#)

all_items_dict

```
[SerializeField]  
private Dictionary<string, EquipmentItem> all_items_dict
```

Field Value

[Dictionary](#) <[string](#), [EquipmentItem](#)>

all_player_equipment

```
[SerializeField]  
public List<EquipmentItem> all_player_equipment
```

Field Value

[List](#) <[EquipmentItem](#)>

player_active_equipment

```
[SerializeField]  
public List<EquipmentItem> player_active_equipment
```

Field Value

[List](#) <[EquipmentItem](#)>

Methods

AddItem(string)

Adds an item to the player's equipment by its ID.

```
public bool AddItem(string ID)
```

Parameters

ID [string](#)

The unique identifier of the item to add.

Returns

`bool`

True if the item was successfully added, false otherwise.

GetActiveItemByPlace(`ItemPlace`)

Retrieves an active equipment item by its specified place.

```
public EquipmentItem GetActiveItemByPlace(ItemPlace place)
```

Parameters

`place` [ItemPlace](#)

The place of the item in the inventory.

Returns

[EquipmentItem](#)

The active equipment item in the specified place, or null if not found.

GetActiveItems()

Retrieves the list of active equipment items for the player.

```
public List<EquipmentItem> GetActiveItems()
```

Returns

[List](#) <[EquipmentItem](#)>

List of active equipment items.

GetItem(string)

Retrieves an equipment item by its ID.

```
public EquipmentItem GetItem(string ID)
```

Parameters

ID [string](#)

The unique identifier of the item.

Returns

[EquipmentItem](#)

The equipment item with the specified ID, or null if not found.

GetItemsByPlace(ItemPlace)

Retrieves all equipment items by their specified place.

```
public List<EquipmentItem> GetItemsByPlace(ItemPlace place)
```

Parameters

place [ItemPlace](#)

The place of the items in the inventory.

Returns

[List](#)<[EquipmentItem](#)>

A list of equipment items in the specified place.

LoadAllItemsInGame()

Loads all equipment items in the game asynchronously.

```
public IEnumerator LoadAllItemsInGame()
```

Returns

[IEnumerator](#)

An IEnumerator for coroutine compatibility.

LoadPlayerInventory()

Loads the player's inventory asynchronously.

```
public IEnumerator LoadPlayerInventory()
```

Returns

[IEnumerator](#)

An IEnumerator for coroutine compatibility.

Prepare()

Prepares the EquipmentHolderManager by loading all items in the game and the player's inventory.

```
public override IEnumerator Prepare()
```

Returns

[IEnumerator](#)

Coroutine for asynchronous operation.

RemoveActiveItem(string)

Removes an active item from the player's equipment by its ID.

```
public bool RemoveActiveItem(string ID)
```

Parameters

[ID](#) [string](#)

The unique identifier of the item to remove.

Returns

[bool](#)

True if the item was successfully removed, false otherwise.

SetItemActive(string)

Activates an item by its ID. If another item is active in the same slot, it is removed.

```
public bool SetItemActive(string ID)
```

Parameters

[ID](#) [string](#)

The ID of the item to activate.

Returns

[bool](#)

True if the item was successfully activated, false otherwise.

Class InventoryPageUI

Namespace: [TaleOfTiles.Inventory](#)

Assembly: Assembly-CSharp.dll

Manages the inventory page UI, displaying player stats, inventory slots, and experience progress.

```
public class InventoryPageUI : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← InventoryPageUI

Fields

button_close_inventory

```
[SerializeField]  
private Button button_close_inventory
```

Field Value

Button

damage_stat

```
public Text damage_stat
```

Field Value

Text

defense_stat

```
public Text defense_stat
```

Field Value

Text

evade_stat

```
public Text evade_stat
```

Field Value

Text

health_stat

```
public Text health_stat
```

Field Value

Text

name_title

```
public Text name_title
```

Field Value

Text

player

```
public PlayerCharacter player
```

Field Value

[PlayerCharacter](#)

slots

```
public List<EquipSlotUI> slots
```

Field Value

[List](#)<[EquipSlotUI](#)>

slots_container

```
public RectTransform slots_container
```

Field Value

RectTransform

xp_progress_bar

```
public Slider xp_progress_bar
```

Field Value

Slider

Methods

LoadInventory()

Loads the player's inventory into the UI.

```
public void LoadInventory()
```

Start()

Initialization function called before the first frame update. Sets up inventory slots and close button event listener.

```
public void Start()
```

UpdateStats()

Updates the displayed stats based on the player's current equipment and skills.

```
public void UpdateStats()
```

Class ItemSkillInventoryPanelUI

Namespace: [TaleOfTiles.Inventory](#)

Assembly: Assembly-CSharp.dll

Manages the display of a skill in the item skill inventory panel.

```
public class ItemSkillInventoryPanelUI : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ItemSkillInventoryPanelUI

Fields

skill_type

Text field displaying the type of the skill.

```
public Text skill_type
```

Field Value

Text

skill_value

Text field displaying the value of the skill.

```
public Text skill_value
```

Field Value

Text

Class ItemsPanelUI

Namespace: [TaleOfTiles.Inventory](#)

Assembly: Assembly-CSharp.dll

Manages the items panel in the game UI, displaying information about items and skills.

```
public class ItemsPanelUI : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ItemsPanelUI

Fields

caller_slot

```
private EquipSlotUI caller_slot
```

Field Value

[EquipSlotUI](#)

current_displayed_items

```
private List<GameObject> current_displayed_items
```

Field Value

[List](#)<GameObject>

item_list_container

```
public GameObject item_list_container
```

Field Value

GameObject

item_prefab

```
public GameObject item_prefab
```

Field Value

GameObject

item_specs_loader

```
public GameObject item_specs_loader
```

Field Value

GameObject

selected_item_description_text

```
public Text selected_item_description_text
```

Field Value

Text

selected_item_name_text

```
public Text selected_item_name_text
```

Field Value

Text

selected_item_rank_text

```
public Text selected_item_rank_text
```

Field Value

Text

selected_slot

```
[SerializeField]  
private EquipSlotUI selected_slot
```

Field Value

[EquipSlotUI](#)

skill_prefab

```
public GameObject skill_prefab
```

Field Value

GameObject

skills_container

```
public GameObject skills_container
```

Field Value

GameObject

Methods

ApplySelected()

Applies the selected item to the corresponding slot.

```
public void ApplySelected()
```

HideItemSpecsLoader()

Hides the item specifications loader.

```
public void HideItemSpecsLoader()
```

HidePanel()

Hides the panel and makes it non-interactive.

```
public void HidePanel()
```

LoadItems(EquipSlotUI)

Loads and displays items relevant to the selected slot.

```
public void LoadItems(EquipSlotUI slot)
```

Parameters

slot [EquipSlotUI](#)

The slot for which items are to be loaded.

SelectSlot(EquipSlotUI)

Selects an item slot and updates the UI to show its details.

```
public void SelectSlot(EquipSlotUI slot)
```

Parameters

slot [EquipSlotUI](#)

The selected item slot.

ShowItemSpecsLoader()

Shows the item specifications loader.

```
public void ShowItemSpecsLoader()
```

ShowPanel()

Makes the panel visible and interactive.

```
public void ShowPanel()
```

Namespace TaleOfTiles.Journey

Classes

[EnemiesObjectsBundle](#)

Represents a bundle of enemies in the game, categorized by difficulty. This is a ScriptableObject which allows for easier management in the Unity editor.

[JourneyManager](#)

Singleton class for managing game journey.

[JourneyMap](#)

Represents a Journey Map in the game, holding configurations for journey map blocks, potential enemies, levels, and other settings. This class is a ScriptableObject that can be created via the Unity Editor.

[JourneyMapAssetBundle](#)

Represents a Journey Map Asset Bundle in the game, which includes a collection of journey map blocks. This class is a ScriptableObject that can be created via the Unity Editor.

[JourneyMapBlock](#)

Represents a block within a journey map in the game. This class includes configurations for different block types and activities.

[JourneyMapBlockActivity](#)

Represents an activity within a journey map block.

[JourneyMapBuilder](#)

Manages the construction and visualization of a journey map in a game. It controls map generation, block placement, and fog of war adjustments.

[JourneyMapButtonsController](#)

The JourneyMapButtonsController class manages the buttons in the journey map.

[JourneyMapConfig](#)

[JourneyMapContainer](#)

The JourneyMapContainer class manages the journey map blocks in the journey map.

[JourneyMapGenerator](#)

Generates a journey map by creating configurations for various blocks including roads and activities.

[JourneyObject](#)

The JourneyObject class represents a journey in the game.

[JourneysUIContainer](#)

The JourneysUIContainer class manages the UI for the journeys.

[MapActivityUIObject](#)

The MapActivityUIObject class manages the UI for a map activity.

Structs

[JourneyMapBlock.JourneyMapBlockConfig](#)

Struct for configuring the properties of a Journey Map Block.

Enums

[JourneyMapBlock.BlockType](#)

Enum defining the type of block.

[JourneyMapBlock.RoadNumber](#)

Enum defining the number of roads.

[JourneyMapBlockActivity.ActivityStatus](#)

[JourneyMapBlockActivity.ActivityType](#)

[JourneyObject.JourneyHardness](#)

Class EnemiesObjectsBundle

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

Represents a bundle of enemies in the game, categorized by difficulty. This is a ScriptableObject which allows for easier management in the Unity editor.

```
[CreateAssetMenu(fileName = "new Enemies Bundle", menuName = "Enemies/new Enemies Bundle",
order = 0)]
public class EnemiesObjectsBundle : ScriptableObject
```

Inheritance

[object](#) ← Object ← ScriptableObject ← EnemiesObjectsBundle

Fields

ID

The unique identifier of the bundle.

```
public string ID
```

Field Value

[string](#)

bundle_name

The name of the bundle.

```
public string bundle_name
```

Field Value

[string](#)

easy_enemies

List of easy difficulty enemies.

```
public List<EnemyObject> easy_enemies
```

Field Value

[List](#) <[EnemyObject](#)>

hard_enemies

List of hard difficulty enemies.

```
public List<EnemyObject> hard_enemies
```

Field Value

[List](#) <[EnemyObject](#)>

medium_enemies

List of medium difficulty enemies.

```
public List<EnemyObject> medium_enemies
```

Field Value

[List](#) <[EnemyObject](#)>

Methods

OnValidate()

Validates the object in the editor. If the ID is not set, it generates a new unique identifier.

```
public void OnValidate()
```

Class JourneyManager

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

Singleton class for managing game journey.

```
public class JourneyManager : Singleton<JourneyManager>
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton<JourneyManager>](#) ← JourneyManager

Inherited Members

[Singleton<JourneyManager>.m_instance](#) , [Singleton<JourneyManager>.Instance](#) ,
[Singleton<JourneyManager>.Awake\(\)](#) , [SingletonBase.AllSingletons](#)

Fields

current_activity_id

```
public string current_activity_id
```

Field Value

[string](#)

Gets and sets the current activity ID.

current_journey

```
public JourneyObject current_journey
```

Field Value

[JourneyObject](#)

Gets and sets the current journey object.

current_journey_id

```
public string current_journey_id
```

Field Value

[string](#)

Gets and sets the current journey ID.

journey_images

```
public List<Sprite> journey_images
```

Field Value

[List](#) <Sprite>

Gets and sets the list of journey images.

Methods

BuildJourneyMap(JourneyObject)

Builds a journey map for a given journey.

```
public IEnumerator BuildJourneyMap(JourneyObject journey)
```

Parameters

journey [JourneyObject](#)

The journey object the map is built with.

Returns

[IEnumerator](#)

An IEnumerator that can be used to execute the method over time.

BuildJourneySelectionMap()

Builds a journey selection map.

```
public IEnumerator BuildJourneySelectionMap()
```

Returns

[IEnumerator](#)

An IEnumerator that can be used to execute the method over time.

GenerateNewJourney(JourneyHardness)

Generates a new journey with a given difficulty level.

```
public JourneyObject GenerateNewJourney(JourneyObject.JourneyHardness difficulty_level)
```

Parameters

[difficulty_level](#) [JourneyObject.JourneyHardness](#)

The difficulty level for the new journey.

Returns

[JourneyObject](#)

The newly created journey object.

LeaveJourney(string)

Deletes a journey from the journey manager with a given journey ID.

```
public void LeaveJourney(string journey_ID)
```

Parameters

journey_ID [string](#)

The ID of the journey to be deleted.

LoadEasyJourney()

Loads an easy journey from the available journey objects.

```
public JourneyObject LoadEasyJourney()
```

Returns

[JourneyObject](#)

The first easy journey object found or null if no such journey exists.

LoadHardJourney()

Loads a hard journey from the available journey objects.

```
public JourneyObject LoadHardJourney()
```

Returns

[JourneyObject](#)

The first hard journey object found or null if no such journey exists.

LoadMiddleJourney()

Loads a middle difficulty journey from the available journey objects.

```
public JourneyObject LoadMiddleJourney()
```

Returns

[JourneyObject](#)

The first middle difficulty journey object found or null if no such journey exists.

Prepare()

Prepares the journey manager for operation. Override of Singleton's Prepare method.

```
public override IEnumerator Prepare()
```

Returns

[IEnumerator](#)

An IEnumerator that can be used to execute the method over time.

ScoreDefeat()

Handles the scoring of a defeat in the current journey.

```
public void ScoreDefeat()
```

ScoreWin()

Handles the scoring of a win in the current journey.

```
public void ScoreWin()
```

Class JourneyMap

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

Represents a Journey Map in the game, holding configurations for journey map blocks, potential enemies, levels, and other settings. This class is a ScriptableObject that can be created via the Unity Editor.

```
[CreateAssetMenu(fileName = "New Journey Map Block", menuName = "Journey Map/new Map", order = 0)]
public class JourneyMap : ScriptableObject
```

Inheritance

[object](#) ← Object ← ScriptableObject ← JourneyMap

Fields

ID

```
public string ID
```

Field Value

[string](#)

activity_prefab

```
public GameObject activity_prefab
```

Field Value

GameObject

asset_bundle

```
public JourneyMapAssetBundle asset_bundle
```

Field Value

[JourneyMapAssetBundle](#)

config_id

```
public string config_id
```

Field Value

[string](#)

journey_map_blocks_config

```
public List<JourneyMapBlock.JourneyMapBlockConfig> journey_map_blocks_config
```

Field Value

[List](#)<[JourneyMapBlock.JourneyMapBlockConfig](#)>

possible_enemies

```
public List<EnemyObject> possible_enemies
```

Field Value

[List](#)<[EnemyObject](#)>

possible_levels

```
public List<LevelObject> possible_levels
```

Field Value

[List](#) <[LevelObject](#)>

use_for_journey

```
public bool use_for_journey
```

Field Value

[bool](#)

Methods

GenerateMap(int)

```
[ContextMenu("Generate Map")]
public void GenerateMap(int activities = 4)
```

Parameters

activities [int](#)

OnValidate()

Validates the object, generating a new unique ID if necessary.

```
public void OnValidate()
```

SaveConfig()

```
public void SaveConfig()
```

UpdateConfig()

```
public void UpdateConfig()
```

Class JourneyMapAssetBundle

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

Represents a Journey Map Asset Bundle in the game, which includes a collection of journey map blocks. This class is a ScriptableObject that can be created via the Unity Editor.

```
[CreateAssetMenu(fileName = "New Journey Map Bundle", menuName = "Journey Map/new asset  
Bundle", order = 1)]  
public class JourneyMapAssetBundle : ScriptableObject
```

Inheritance

[object](#) ← Object ← ScriptableObject ← JourneyMapAssetBundle

Fields

asset_name

The name of the asset bundle.

```
public string asset_name
```

Field Value

[string](#)

block_list

A list of journey map blocks included in this asset bundle.

```
public List<JourneyMapBlock> block_list
```

Field Value

[List](#)<[JourneyMapBlock](#)>

Class JourneyMapBlock

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

Represents a block within a journey map in the game. This class includes configurations for different block types and activities.

```
[ExecuteAlways]  
public class JourneyMapBlock : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← JourneyMapBlock

Fields

ID

```
public string ID
```

Field Value

[string](#)

activities_ui_position

```
public List<GameObject> activities_ui_position
```

Field Value

[List](#) <GameObject>

activities

```
public List<JourneyMapBlockActivity> activities
```

Field Value

[List](#)<[JourneyMapBlockActivity](#)>

block_image

```
public Image block_image
```

Field Value

Image

roads_in

```
public JourneyMapBlock.RoadNumber roads_in
```

Field Value

[JourneyMapBlock.RoadNumber](#)

roads_out

```
public JourneyMapBlock.RoadNumber roads_out
```

Field Value

[JourneyMapBlock.RoadNumber](#)

sprite_index

```
private int sprite_index
```

Field Value

[int](#)

sprite_variants

```
public List<Sprite> sprite_variants
```

Field Value

[List](#)<Sprite>

type

```
public JourneyMapBlock.BlockType type
```

Field Value

[JourneyMapBlock.BlockType](#)

Methods

ActivateActivity(string)

Activates a specific activity within the map block based on its ID. Handles enemy encounter logic.

```
public void ActivateActivity(string activity_id)
```

Parameters

activity_id [string](#)

ID of the activity to activate.

GetConfig()

Creates and returns the configuration of the current JourneyMapBlock.

```
public JourneyMapBlock.JourneyMapBlockConfig GetConfig()
```

Returns

[JourneyMapBlock.JourneyMapBlockConfig](#)

OnValidate()

Method to validate and set the ID of the block.

```
public void OnValidate()
```

SetSprite(int)

Sets the sprite for the map block based on the specified index.

```
public void SetSprite(int index)
```

Parameters

index [int](#)

Index of the sprite variant to be set.

Start()

Initializes the JourneyMapBlock on startup. Validates if the block image and sprite variants are properly set.

```
public void Start()
```

Enum JourneyMapBlock.BlockType

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

Enum defining the type of block.

```
public enum JourneyMapBlock.BlockType
```

Fields

Activity = 1

Road = 0

Struct JourneyMapBlock.JourneyMapBlockConfig

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

Struct for configuring the properties of a Journey Map Block.

```
[Serializable]  
public struct JourneyMapBlock.JourneyMapBlockConfig
```

Fields

ID

```
public string ID
```

Field Value

[string](#)

activities

```
public List<JourneyMapBlockActivity> activities
```

Field Value

[List](#) <[JourneyMapBlockActivity](#)>

internal_id

```
public string internal_id
```

Field Value

[string](#) ↗

roads_in

```
public JourneyMapBlock.RoadNumber roads_in
```

Field Value

[JourneyMapBlock.RoadNumber](#)

roads_out

```
public JourneyMapBlock.RoadNumber roads_out
```

Field Value

[JourneyMapBlock.RoadNumber](#)

sprite_index

```
public int sprite_index
```

Field Value

[int](#) ↗

sprite_variants_count

```
public int sprite_variants_count
```

Field Value

[int](#)

type

`public JourneyMapBlock.BlockType type`

Field Value

[JourneyMapBlock.BlockType](#)

Enum JourneyMapBlock.RoadNumber

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

Enum defining the number of roads.

```
public enum JourneyMapBlock.RoadNumber
```

Fields

One = 1

Three = 3

Two = 2

Zero = 0

Class JourneyMapBlockActivity

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

Represents an activity within a journey map block.

```
[Serializable]
public class JourneyMapBlockActivity
```

Inheritance

[object](#) ← JourneyMapBlockActivity

Constructors

JourneyMapBlockActivity(ActivityType, string, string, ActivityStatus, string)

Constructor for initializing a new instance of JourneyMapBlockActivity.

```
public JourneyMapBlockActivity(JourneyMapBlockActivity.ActivityType type, string enemy_id,
string level_id, JourneyMapBlockActivity.ActivityStatus activity_status, string id = "")
```

Parameters

type [JourneyMapBlockActivity.ActivityType](#)

enemy_id [string](#)

level_id [string](#)

activity_status [JourneyMapBlockActivity.ActivityStatus](#)

id [string](#)

Fields

activity_id

```
public string activity_id
```

Field Value

[string](#) ↗

activity_status

```
public JourneyMapBlockActivity.ActivityStatus activity_status
```

Field Value

[JourneyMapBlockActivity.ActivityStatus](#)

enemy_id

```
public string enemy_id
```

Field Value

[string](#) ↗

level_id

```
public string level_id
```

Field Value

[string](#) ↗

type

```
public JourneyMapBlockActivity.ActivityType type
```

Field Value

[JourneyMapBlockActivity.ActivityType](#)

Enum JourneyMapBlockActivity.ActivityStatus

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

```
public enum JourneyMapBlockActivity.ActivityStatus
```

Fields

Blocked = 3

Finished = 1

Neutral = 0

Skipped = 2

Enum JourneyMapBlockActivity.ActivityType

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

```
public enum JourneyMapBlockActivity.ActivityType
```

Fields

Enemy = 0

Class JourneyMapBuilder

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

Manages the construction and visualization of a journey map in a game. It controls map generation, block placement, and fog of war adjustments.

```
[ExecuteAlways]  
public class JourneyMapBuilder : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← JourneyMapBuilder

Fields

activities_for_random_map_generation

```
public int activities_for_random_map_generation
```

Field Value

[int](#)

fog_of_war

```
public GameObject fog_of_war
```

Field Value

GameObject

map

```
public JourneyMap map
```

Field Value

[JourneyMap](#)

map_mask

```
public RectTransform map_mask
```

Field Value

RectTransform

map_root

```
public GameObject map_root
```

Field Value

GameObject

placed_blocks

```
private List<GameObject> placed_blocks
```

Field Value

[List](#) <GameObject>

scroll_rect

```
public ScrollRect scroll_rect
```

Field Value

ScrollRect

stop_activity_generating

```
private bool stop_activity_generating
```

Field Value

[bool](#)

Methods

BuildMap()

Builds the map based on the specified journey map configuration.

```
[ContextMenu("Build Map")]
public void BuildMap()
```

BuildMapCoroutine()

Coroutine for building the journey map with delays and visual updates.

```
public IEnumerator BuildMapCoroutine()
```

Returns

[IEnumerator](#)

CenterOnItem(RectTransform, ScrollRect)

Centers the view on a specific item within the scrollable map area.

```
public void CenterOnItem(RectTransform target, ScrollRect scrollRect)
```

Parameters

target RectTransform

The RectTransform of the item to center on.

scrollRect ScrollRect

The ScrollRect component of the map area.

ClearMap()

Clears all JourneyMapBlocks from the map.

```
[ContextMenu("Clear Map")]
public void ClearMap()
```

GenerateRandomMap()

Generates a random map configuration and builds the map accordingly.

```
[ContextMenu("Generate Random Map")]
private void GenerateRandomMap()
```

PlaceBlock(JourneyMapBlock, JourneyMapBlockConfig)

Places a JourneyMapBlock in the scene based on the given block configuration.

```
private void PlaceBlock(JourneyMapBlock block, JourneyMapBlock.JourneyMapBlockConfig
block_config)
```

Parameters

block [JourneyMapBlock](#)

The JourneyMapBlock to be placed.

block_config [JourneyMapBlock.JourneyMapBlockConfig](#)

The configuration details for the block.

StartBuildMapCoroutine()

Starts a coroutine to build the journey map.

```
[ContextMenu("Build Map Coroutine")]
public void StartBuildMapCoroutine()
```

UpdateFog()

Updates the fog of war effect based on the last visible map block.

```
[ContextMenu("Update Fog")]
private void UpdateFog()
```

UpdateMapMask()

Updates the map mask based on the position of the last visible block.

```
[ContextMenu("Update Map Mask")]
private void UpdateMapMask()
```

Class JourneyMapButtonsController

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

The JourneyMapButtonsController class manages the buttons in the journey map.

```
public class JourneyMapButtonsController : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← JourneyMapButtonsController

Fields

home

```
public Button home
```

Field Value

Button

Gets or sets the home button.

Methods

Start()

Start is called before the first frame update. It adds a new listener to the home button which triggers the loading of the Journey Selection Scene when clicked.

```
private void Start()
```

Class JourneyMapConfig

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

```
[Serializable]
public class JourneyMapConfig
```

Inheritance

[object](#) ← JourneyMapConfig

Constructors

JourneyMapConfig(string, string, JourneyMapBlockConfig[])

```
public JourneyMapConfig(string config_id, string map_id,
JourneyMapBlock.JourneyMapBlockConfig[] blocks)
```

Parameters

config_id [string](#)

map_id [string](#)

blocks [JourneyMapBlockConfig](#)[]

Fields

blocks

```
public JourneyMapBlock.JourneyMapBlockConfig[] blocks
```

Field Value

[JourneyMapBlockConfig](#)[]

config_id

```
public string config_id
```

Field Value

[string](#) ↗

map_id

```
public string map_id
```

Field Value

[string](#) ↗

Class JourneyMapContainer

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

The JourneyMapContainer class manages the journey map blocks in the journey map.

```
public class JourneyMapContainer : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← JourneyMapContainer

Methods

AddActivityButtonOnClickListeners()

Adds click listeners to all activity buttons in the journey map blocks. When an activity button is clicked, the corresponding activity is activated.

```
public void AddActivityButtonOnClickListeners()
```

Class JourneyMapGenerator

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

Generates a journey map by creating configurations for various blocks including roads and activities.

```
public class JourneyMapGenerator
```

Inheritance

[object](#) ← JourneyMapGenerator

Methods

GenerateMap(JourneyMapAssetBundle, List<EnemyObject>, List<LevelObject>, int)

Generates a list of configurations for a journey map based on the provided asset bundle, possible enemies, and levels.

```
public List<JourneyMapBlock.JourneyMapBlockConfig> GenerateMap(JourneyMapAssetBundle  
asset_bundle, List<EnemyObject> possible_enemies, List<LevelObject> possible_levels, int  
activities_blocks = 1)
```

Parameters

asset_bundle [JourneyMapAssetBundle](#)

The asset bundle containing map blocks.

possible_enemies [List](#) <[EnemyObject](#)>

List of possible enemies that can appear in the journey map.

possible_levels [List](#) <[LevelObject](#)>

List of possible levels that can be included in the journey map.

activities_blocks [int](#)

The number of activity blocks to include in the map. Default is 1.

Returns

[List ↗ <JourneyMapBlock.JourneyMapBlockConfig>](#)

A list of [JourneyMapBlock.JourneyMapBlockConfig](#) representing the journey map configuration.

GetActivityBlock(JourneyMapAssetBundle, RoadNumber)

```
private JourneyMapBlock.JourneyMapBlockConfig GetActivityBlock(JourneyMapAssetBundle asset_bundle, JourneyMapBlock.RoadNumber roads_in)
```

Parameters

[asset_bundle](#) [JourneyMapAssetBundle](#)

[roads_in](#) [JourneyMapBlock.RoadNumber](#)

Returns

[JourneyMapBlock.JourneyMapBlockConfig](#)

GetRoadBlock(JourneyMapAssetBundle, RoadNumber, int)

```
private JourneyMapBlock.JourneyMapBlockConfig GetRoadBlock(JourneyMapAssetBundle asset_bundle, JourneyMapBlock.RoadNumber roads_in, int blocks_count)
```

Parameters

[asset_bundle](#) [JourneyMapAssetBundle](#)

[roads_in](#) [JourneyMapBlock.RoadNumber](#)

[blocks_count](#) [int ↗](#)

Returns

[JourneyMapBlock.JourneyMapBlockConfig](#)

Class JourneyObject

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

The JourneyObject class represents a journey in the game.

```
[CreateAssetMenu(fileName = "New Journey", menuName = "Journey/new Journey", order = 0)]  
public class JourneyObject : ScriptableObject
```

Inheritance

[object](#) ← Object ← ScriptableObject ← JourneyObject

Fields

ID

```
public string ID
```

Field Value

[string](#)

Gets or sets the ID of the journey.

current_block_id

```
public string current_block_id
```

Field Value

[string](#)

Gets or sets the current block ID of the journey.

enemies_bundle

```
public EnemiesObjectsBundle enemies_bundle
```

Field Value

[EnemiesObjectsBundle](#)

Gets or sets the enemies bundle of the journey.

journey_description

```
public string journey_description
```

Field Value

[string](#)

Gets or sets the description of the journey.

journey_hardness

```
public JourneyObject.JourneyHardness journey_hardness
```

Field Value

[JourneyObject.JourneyHardness](#)

Gets or sets the hardness of the journey.

journey_image

```
public Sprite journey_image
```

Field Value

Sprite

Gets or sets the image of the journey.

journey_map

```
public JourneyMap journey_map
```

Field Value

[JourneyMap](#)

Gets or sets the journey map of the journey.

journey_name

```
public string journey_name
```

Field Value

[string](#)

Gets or sets the name of the journey.

levels_bundle

```
public LevelObjectsBundle levels_bundle
```

Field Value

[LevelObjectsBundle](#)

Gets or sets the levels bundle of the journey.

map_config_id

```
public string map_config_id
```

Field Value

[string](#) ↗

Gets or sets the map config ID of the journey.

map_id

```
public string map_id
```

Field Value

[string](#) ↗

Gets or sets the map ID of the journey.

Methods

GenerateMap()

Generates a map for the journey.

```
public void GenerateMap()
```

OnValidate()

OnValidate is called when the script is loaded or a value is changed in the inspector. It generates a new unique ID for the journey if it doesn't have one yet.

```
public void OnValidate()
```

SetupMap()

Sets up the map for the journey.

```
public void SetupMap()
```

Enum JourneyObject.JourneyHardness

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

```
public enum JourneyObject.JourneyHardness
```

Fields

Easy = 0

Hard = 2

Tough = 1

Class JourneysUIContainer

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

The JourneysUIContainer class manages the UI for the journeys.

```
public class JourneysUIContainer : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← JourneysUIContainer

Fields

continue_button

```
public Button continue_button
```

Field Value

Button

Gets or sets the continue button.

default_image

```
public Sprite default_image
```

Field Value

Sprite

Gets or sets the default image.

form_button

```
private Button form_button
```

Field Value

Button

Gets or sets the form button.

journey

```
public JourneyObject journey
```

Field Value

[JourneyObject](#)

Gets or sets the journey object.

journey_hardness

```
public JourneyObject.JourneyHardness journey_hardness
```

Field Value

[JourneyObject.JourneyHardness](#)

Gets or sets the hardness of the journey.

journey_image

```
public Image journey_image
```

Field Value

Image

Gets or sets the journey image.

journey_name_text

```
public Text journey_name_text
```

Field Value

Text

Gets or sets the journey name text.

journey_progress_bar

```
public Slider journey_progress_bar
```

Field Value

Slider

Gets or sets the journey progress bar.

leave_button

```
public Button leave_button
```

Field Value

Button

Gets or sets the leave button.

new_journey_text

```
public Text new_journey_text
```

Field Value

Text

Gets or sets the text for new journey.

Methods

CalculateProgress()

Calculates the progress of the current journey as a percentage.

```
private int CalculateProgress()
```

Returns

[int](#)

The progress of the journey as a percentage.

SetJourney(JourneyObject)

Sets the current journey, updates the UI accordingly, and adds the relevant listeners to the buttons.

```
public void SetJourney(JourneyObject journey)
```

Parameters

[journey](#) [JourneyObject](#)

The JourneyObject to set.

Class MapActivityUIObject

Namespace: [TaleOfTiles.Journey](#)

Assembly: Assembly-CSharp.dll

The MapActivityUIObject class manages the UI for a map activity.

```
public class MapActivityUIObject : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← MapActivityUIObject

Fields

activity_blocked_color

```
public Color activity_blocked_color
```

Field Value

Color

Gets or sets the color for a blocked activity.

activity_finished_color

```
public Color activity_finished_color
```

Field Value

Color

Gets or sets the color for a finished activity.

activity_neutral_color

```
public Color activity_neutral_color
```

Field Value

Color

Gets or sets the color for a neutral activity.

activity_skipped_color

```
public Color activity_skipped_color
```

Field Value

Color

Gets or sets the color for a skipped activity.

enemy_image

```
public Image enemy_image
```

Field Value

Image

Gets or sets the enemy image.

status

```
public JourneyMapBlockActivity.ActivityStatus status
```

Field Value

[JourneyMapBlockActivity.ActivityStatus](#)

Gets or sets the status of the activity. This property is read-only in the inspector.

status_background_image

```
public Image status_background_image
```

Field Value

Image

Gets or sets the status background image.

status_frame_image

```
public Image status_frame_image
```

Field Value

Image

Gets or sets the status frame image.

Methods

SetStatus(ActivityStatus)

Sets the status of the activity and updates the color of the status frame and background images accordingly.

```
public void SetStatus(JourneyMapBlockActivity.ActivityStatus status)
```

Parameters

status [JourneyMapBlockActivity.ActivityStatus](#)

The status to set.

Namespace TaleOfTiles.Level

Classes

[LevelHolderManager](#)

Manages level-related functionalities within the game. This class is attached to a Unity GameObject and is part of the game's level management system.

[LevelObject](#)

The LevelObject class represents a level in the game.

[LevelObjectsBundle](#)

A ScriptableObject representing a bundle of level objects.

[LevelSelectionScreen](#)

Manages the level selection screen, dynamically creating buttons for each available level.

[LoadingTextAnim](#)

Controls the animation for a loading text, creating a simple "..." animation.

Class LevelHolderManager

Namespace: [TaleOfTiles.Level](#)

Assembly: Assembly-CSharp.dll

Manages level-related functionalities within the game. This class is attached to a Unity GameObject and is part of the game's level management system.

```
public class LevelHolderManager : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← LevelHolderManager

Methods

Start()

Called before the first frame update. Use this for initialization.

```
private void Start()
```

Update()

Update is called once per frame. Use this method to update the frame-dependent logic.

```
private void Update()
```

Class LevelObject

Namespace: [TaleOfTiles.Level](#)

Assembly: Assembly-CSharp.dll

The LevelObject class represents a level in the game.

```
[CreateAssetMenu(fileName = "New Level", menuName = "Custom/New Level", order = 2)]
public class LevelObject : ScriptableObject
```

Inheritance

[object](#) ← Object ← ScriptableObject ← LevelObject

Fields

ID

```
public string ID
```

Field Value

[string](#)

Gets or sets the ID of the level. This property is read-only in the inspector.

level_background_image

```
public Sprite level_background_image
```

Field Value

Sprite

Gets or sets the background image of the level.

level_background_music

```
public AudioClip level_background_music
```

Field Value

AudioClip

Gets or sets the background music of the level.

level_description

```
public string level_description
```

Field Value

[string](#)

Gets or sets the description of the level.

level_enemy

```
public EnemyObject level_enemy
```

Field Value

[EnemyObject](#)

Gets or sets the enemy of the level.

level_loading_screen_background_image

```
public Sprite level_loading_screen_background_image
```

Field Value

Sprite

Gets or sets the loading screen background image of the level.

level_miniature

```
public Sprite level_miniature
```

Field Value

Sprite

Gets or sets the miniature of the level.

level_name

```
public string level_name
```

Field Value

[string](#)

Gets or sets the name of the level.

Methods

CopyOtherLevel(LevelObject)

Copies the properties of another level to this one.

```
public void CopyOtherLevel(LevelObject level)
```

Parameters

level [LevelObject](#)

The LevelObject to copy from.

OnEnable()

OnEnable is called when the script instance is being loaded. It generates a new unique ID for the level if it doesn't have one yet.

```
private void OnEnable()
```

Class LevelObjectsBundle

Namespace: [TaleOfTiles.Level](#)

Assembly: Assembly-CSharp.dll

A ScriptableObject representing a bundle of level objects.

```
[CreateAssetMenu(fileName = "new Level Objects Bundle", menuName = "Level/new Levels
Bundle", order = 0)]
public class LevelObjectsBundle : ScriptableObject
```

Inheritance

[object](#) ← Object ← ScriptableObject ← LevelObjectsBundle

Fields

ID

Unique identifier for the bundle. Automatically generated if not set.

```
public string ID
```

Field Value

[string](#)

bundle_name

The name of the bundle.

```
public string bundle_name
```

Field Value

[string](#)

level_objects

A list of LevelObject instances included in this bundle.

```
public List<LevelObject> level_objects
```

Field Value

[List](#) <[LevelObject](#)>

Methods

OnValidate()

Validates the object, ensuring it has a unique ID.

```
public void OnValidate()
```

Class LevelSelectionScreen

Namespace: [TaleOfTiles.Level](#)

Assembly: Assembly-CSharp.dll

Manages the level selection screen, dynamically creating buttons for each available level.

```
public class LevelSelectionScreen : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← LevelSelectionScreen

Fields

buttonLevelPrefab

```
public GameObject buttonLevelPrefab
```

Field Value

GameObject

buttonsContainer

```
public Transform buttonsContainer
```

Field Value

Transform

Methods

ClearButtonsContainer()

Clears all existing level buttons from the container.

```
private void ClearButtonsContainer()
```

LoadLevels()

Loads and displays level buttons for each available level.

```
private IEnumerator LoadLevels()
```

Returns

[IEnumerator](#)

Start()

Initializes the level selection screen by loading level buttons.

```
private void Start()
```

Class LoadingTextAnim

Namespace: [TaleOfTiles.Level](#)

Assembly: Assembly-CSharp.dll

Controls the animation for a loading text, creating a simple "..." animation.

```
public class LoadingTextAnim : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← LoadingTextAnim

Fields

anim_coroutine

```
private Coroutine animCoroutine
```

Field Value

Coroutine

delay

```
public float delay
```

Field Value

[float](#)

text

```
private Text text
```

Field Value

Text

Methods

AnimateText()

Animates the text to simulate a loading indicator.

```
private IEnumerator AnimateText()
```

Returns

[IEnumerator](#)

OnDisable()

Stops the animation when the object becomes disabled.

```
private void OnDisable()
```

OnEnable()

Restarts the animation when the object becomes enabled.

```
private void OnEnable()
```

Start()

Initializes the loading text animation.

```
private void Start()
```

Namespace TaleOfTiles.Misc

Classes

[CameraSingleton](#)

Singleton instance for the Camera. Inherits from the generic Singleton class.

[DontDestroyOnLoad](#)

Ensures that the object this script is attached to is not destroyed when loading a new scene. If the object has a Canvas component, it also sets the Canvas's render camera to the main camera.

[EventBus](#)

Singleton class for event bus, used to manage and dispatch events.

[ReadOnlyAttribute](#)

[SceneSwitcherManager](#)

The SceneSwitcherManager class is a singleton that manages scene loading.

[SingletonBase](#)

The SingletonBase class is an abstract class that provides a base for all singleton classes.

[Singleton<T>](#)

A generic Singleton class for Unity MonoBehaviour objects. Ensures only one instance of a type T exists throughout the application.

Class CameraSingleton

Namespace: [TaleOfTiles.Misc](#)

Assembly: Assembly-CSharp.dll

Singleton instance for the Camera. Inherits from the generic Singleton class.

```
public class CameraSingleton : Singleton<CameraSingleton>
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton<CameraSingleton>](#) ← CameraSingleton

Inherited Members

[Singleton<CameraSingleton>.m_instance](#) , [Singleton<CameraSingleton>.Instance](#) ,
[Singleton<CameraSingleton>.Awake\(\)](#) , [Singleton<CameraSingleton>.Prepare\(\)](#) ,
[SingletonBase.AllSingletons](#)

Class DontDestroyOnLoad

Namespace: [TaleOfTiles.Misc](#)

Assembly: Assembly-CSharp.dll

Ensures that the object this script is attached to is not destroyed when loading a new scene. If the object has a Canvas component, it also sets the Canvas's render camera to the main camera.

```
public class DontDestroyOnLoad : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← DontDestroyOnLoad

Methods

Awake()

Called when the script instance is being loaded. Prevents the object from being destroyed when loading a new scene. If the object has a Canvas component, it sets the Canvas's render camera to the main camera.

```
private void Awake()
```

Class EventBus

Namespace: [TaleOfTiles.Misc](#)

Assembly: Assembly-CSharp.dll

Singleton class for event bus, used to manage and dispatch events.

```
public class EventBus : Singleton<EventBus>
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton<EventBus>](#) ← EventBus

Inherited Members

[Singleton<EventBus>.m_instance](#), [Singleton<EventBus>.Instance](#), [Singleton<EventBus>.Awake\(\)](#),
[Singleton<EventBus>.Prepare\(\)](#), [SingletonBase.AllSingletons](#)

Fields

eventDictionary

Dictionary storing the event listeners for each event name.

```
private Dictionary<string, Action<object>> eventDictionary
```

Field Value

[Dictionary<string, Action<object>>](#)

Methods

Publish(string, object)

Publishes an event, invoking all subscribed listeners.

```
public void Publish(string eventName, object eventArg = null)
```

Parameters

eventName [string](#)

The name of the event to publish.

eventArg [object](#)

The argument to pass to the event listeners.

Subscribe(string, Action<object>)

Subscribes a listener to a specific event.

```
public void Subscribe(string eventName, Action<object> listener)
```

Parameters

eventName [string](#)

The name of the event to subscribe to.

listener [Action](#)<[object](#)>

The listener to subscribe.

Unsubscribe(string, Action<object>)

Unsubscribes a listener from a specific event.

```
public void Unsubscribe(string eventName, Action<object> listener)
```

Parameters

eventName [string](#)

The name of the event to unsubscribe from.

listener [Action](#)<[object](#)>

The listener to unsubscribe.

Class ReadOnlyAttribute

Namespace: [TaleOfTiles.Misc](#)

Assembly: Assembly-CSharp.dll

```
public class ReadOnlyAttribute : PropertyAttribute
```

Inheritance

[object](#) ← [Attribute](#) ← PropertyAttribute ← ReadOnlyAttribute

Class SceneSwitcherManager

Namespace: [TaleOfTiles.Misc](#)

Assembly: Assembly-CSharp.dll

The SceneSwitcherManager class is a singleton that manages scene loading.

```
public class SceneSwitcherManager : Singleton<SceneSwitcherManager>
```

Inheritance

[Object](#) ↗ ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton<SceneSwitcherManager>](#) ← SceneSwitcherManager

Inherited Members

[Singleton<SceneSwitcherManager>.m_instance](#) , [Singleton<SceneSwitcherManager>.Instance](#) ,
[Singleton<SceneSwitcherManager>.Awake\(\)](#) , [Singleton<SceneSwitcherManager>.Prepare\(\)](#) ,
[SingletonBase.AllSingletons](#)

Methods

LoadScene(string)

Loads a scene by its name.

```
public void LoadScene(string sceneName)
```

Parameters

sceneName [string](#) ↗

The name of the scene to be loaded.

LoadSceneAsync(string, Action)

Loads a scene asynchronously by its name.

```
public void LoadSceneAsync(string sceneName, Action callback = null)
```

Parameters

sceneName [string](#)

The name of the scene to be loaded.

callback [Action](#)

An optional callback to be invoked after the scene is loaded.

LoadSceneCoroutine(string, Action)

A coroutine to load a scene asynchronously by its name.

```
public IEnumerator LoadSceneCoroutine(string sceneName, Action callback = null)
```

Parameters

sceneName [string](#)

The name of the scene to be loaded.

callback [Action](#)

An optional callback to be invoked after the scene is loaded.

Returns

[IEnumerator](#)

An IEnumerator for coroutine sequencing.

Class SingletonBase

Namespace: [TaleOfTiles.Misc](#)

Assembly: Assembly-CSharp.dll

The SingletonBase class is an abstract class that provides a base for all singleton classes.

```
public abstract class SingletonBase : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← SingletonBase

Derived

[Singleton<T>](#)

Fields

AllSingletons

```
public static List<SingletonBase> AllSingletons
```

Field Value

[List<SingletonBase>](#)

Gets a list of all singletons.

Methods

Prepare()

An abstract method that prepares the singleton. All derived classes must implement this method.

```
public abstract IEnumerator Prepare()
```

Returns

[IEnumerator](#)

An IEnumerator for coroutine sequencing.

Class Singleton<T>

Namespace: [TaleOfTiles.Misc](#)

Assembly: Assembly-CSharp.dll

A generic Singleton class for Unity MonoBehaviour objects. Ensures only one instance of a type T exists throughout the application.

```
public class Singleton<T> : SingletonBase where T : MonoBehaviour
```

Type Parameters

T

The type of the MonoBehaviour that is a singleton.

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← Singleton<T>

Derived

[BattleArenaManager](#), [CombatManager](#), [AudioManager](#), [GeneralAudioGroup](#), [CardsHolderManager](#), [DataManager](#), [EquipmentHolderManager](#), [JourneyManager](#), [MainGameManager](#), [CameraSingleton](#), [EventBus](#), [SceneSwitcherManager](#), [AdditionalScreens](#), [LoadingScreenManager](#), [NavigationButtons](#), [TopBarMenu](#), [UIManager](#)

Inherited Members

[SingletonBase.AllSingletons](#)

Fields

m_instance

```
private static T m_instance
```

Field Value

T

Properties

Instance

Gets the instance of the Singleton, creating it if it doesn't already exist.

```
public static T Instance { get; }
```

Property Value

T

Methods

Awake()

Initializes the singleton instance and sets it to persist across scenes. Destroys the gameObject if another instance already exists.

```
public virtual void Awake()
```

Prepare()

Prepares the singleton. Can be overridden in derived classes for custom behavior.

```
public override IEnumerator Prepare()
```

Returns

[IEnumerator](#)

Namespace TaleOfTiles.Player

Classes

[PlayerCharacter](#)

Represents the player character in the game, managing their cards, equipment, skills, and battle statistics.

Class PlayerCharacter

Namespace: [TaleOfTiles.Player](#)

Assembly: Assembly-CSharp.dll

Represents the player character in the game, managing their cards, equipment, skills, and battle statistics.

```
public class PlayerCharacter : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← PlayerCharacter

Fields

_available_moves

```
private int _available_moves
```

Field Value

[int](#)

all_active_skills

List of all active skills available to the player.

```
private List<GemItem> all_active_skills
```

Field Value

[List](#)<[GemItem](#)>

cards

List of card objects held by the player.

```
public List<CardObject> cards
```

Field Value

[List](#) <[CardObject](#)>

copy_deck_timer

Timer for deck copying functionality.

```
public float copy_deck_timer
```

Field Value

[float](#)

critical_damage_queue

Queue of critical damage effects to be applied.

```
public List<GemSuperCombo.CriticalDamage> critical_damage_queue
```

Field Value

[List](#) <[GemSuperCombo.CriticalDamage](#)>

damage_evasion_queue

Queue of damage evasion effects to be applied.

```
public List<GemSuperCombo.DamageEvasion> damage_evasion_queue
```

Field Value

[List](#) <[GemSuperCombo.DamageEvasion](#)>

equip_items

List of equipment items equipped by the player.

```
public List<EquipmentItem> equip_items
```

Field Value

[List](#) <[EquipmentItem](#)>

health

Current health of the player.

```
public float health
```

Field Value

[float](#)

init_available_moves

Initial number of available moves for the player.

```
public int init_available_moves
```

Field Value

[int](#)

is_human

Indicates if the player character is controlled by a human.

```
public bool is_human
```

Field Value

[bool](#)

last_damage_sound_index

Index of the last damage sound played.

`public int last_damage_sound_index`

Field Value

[int](#)

last_hit_sound_index

Index of the last hit sound played.

`public int last_hit_sound_index`

Field Value

[int](#)

max_health

Maximum health of the player.

`public float max_health`

Field Value

[float](#)

own_active_skills

List of active skills in the form of gems owned by the player.

```
public List<GemItem> own_active_skills
```

Field Value

[List](#)<[GemItem](#)>

own_passive_skills

List of passive skills owned by the player.

```
public List<PassiveSkillEffect> own_passive_skills
```

Field Value

[List](#)<[PassiveSkillEffect](#)>

player_battle_card

UI representation of player's battle card.

```
public PlayerBattleCardUI player_battle_card
```

Field Value

[PlayerBattleCardUI](#)

poison_queue

Queue of poison effects to be applied.

```
public List<GemSuperCombo.Poison> poison_queue
```

Field Value

[List](#) <[GemSuperCombo.Poison](#)>

regeneration_queue

Queue of regeneration combos to be applied.

```
public List<GemSuperCombo.RegenerationCombo> regeneration_queue
```

Field Value

[List](#) <[GemSuperCombo.RegenerationCombo](#)>

sudden_death_protection

Sudden death protection status of the player.

```
public GemSuperCombo.SuddenDeathProtection sudden_death_protection
```

Field Value

[GemSuperCombo.SuddenDeathProtection](#)

temporary_active_skill_reservation

```
[HideInInspector]
```

```
public List<GemItem> temporary_active_skill_reservation
```

Field Value

[List](#) <[GemItem](#)>

Properties

avaliable_moves

Number of moves available to the player.

```
public int available_moves { get; set; }
```

Property Value

[int](#)

available_moves_text

Text component displaying the number of available moves.

```
public Text available_moves_text { get; }
```

Property Value

Text

health_bar

Health bar UI component associated with the player.

```
public HealthBar health_bar { get; }
```

Property Value

[HealthBar](#)

Methods

CalculateMaxHealth()

Calculates the maximum health of the player based on various factors.

```
public int CalculateMaxHealth()
```

Returns

[int↗](#)

The calculated maximum health as an integer.

Remarks

The calculation includes static health increase from cards, equipment, and own passive skills, as well as relative health increase percentages.

CalculatePassiveSkillPercentageChange(PassiveSkillType, bool, GemEffectShieldType)

Calculates the percentage change of a passive skill.

```
public int CalculatePassiveSkillPercentageChange(PassiveSkillType skill_type, bool reduction, GemEffectShieldType damage_type = GemEffectShieldType.None)
```

Parameters

skill_type [PassiveSkillType](#)

Type of the passive skill.

reduction [bool↗](#)

Indicates if the calculation is for reduction.

damage_type [GemEffectShieldType](#)

Type of damage for the GemEffectShield, if applicable.

Returns

[int↗](#)

Calculated percentage change as an integer.

ChangeHealth(float, bool)

Changes the health of the player, considering factors like critical damage, evasion, and passive skills.

```
public void ChangeHealth(float change, bool critical_damage = false)
```

Parameters

change [float](#)

critical_damage [bool](#)

GetActiveSkills()

Gets the current list of active skills of the player.

```
public List<GemItem> GetActiveSkills()
```

Returns

[List](#)<[GemItem](#)>

List of active skills.

GetPassiveSkills(PassiveSkillType)

Retrieves passive skills of a specific type.

```
public List<PassiveSkillEffect> GetPassiveSkills(PassiveSkillType desiredType)
```

Parameters

desiredType [PassiveSkillType](#)

The type of passive skills to retrieve.

Returns

[List](#)<[PassiveSkillEffect](#)>

List of passive skills of the specified type.

Prepare()

Prepares the player for battle, initializing health and moves.

```
public void Prepare()
```

ReloadActiveSkills()

Reloads the active skills of the player from their cards and own active skills.

```
public void ReloadActiveSkills()
```

ResetHealthPoints()

Resets the player's health points to their maximum value.

```
public void ResetHealthPoints()
```

ResetMoves()

Resets the player's available moves to the initial value.

```
public void ResetMoves()
```

ResetStatuses()

Resets all status effects on the player.

```
public void ResetStatuses()
```

SetActiveSkills(List<GemItem>)

Sets the list of active skills for the player.

```
public void SetActiveSkills(List<GemItem> new_active_skills)
```

Parameters

`new_active_skills` [List](#)<[GemItem](#)>

List of new active skills to be set.

Namespace TaleOfTiles.UI

Classes

[AdditionalScreens](#)

This class handles the functionality of additional screens in the game. It inherits from the Singleton class to ensure only one instance of this class exists.

[CollapseUIScript](#)

Handles the collapse and expand functionality of UI elements.

[DeckCanvasUI](#)

Handles the display and functionality of the deck in the UI.

[DefeatScreenUI](#)

Handles the display and functionality of the defeat screen in the UI.

[LoadingScreenManager](#)

The LoadingScreenManager class is a singleton that manages the game's loading screen.

[NavigationButtons](#)

[ParticleManager](#)

Manages particle effects for various game events, such as clearing pieces or breaking tiles.

[ParticlePlayer](#)

Manages the playback of particle systems attached to the game object.

[ParticleSystemSizeAdaptation](#)

Adapts the size of a particle system to match the size of its parent RectTransform.

[TopBarMenu](#)

The TopBarMenu class manages the top bar menu.

[UIManager](#)

The UIManager class is a singleton that manages the UI of the game.

[VictoryScreenUI](#)

Manages the UI for the victory screen in the game, including display of rewards and navigation buttons.

Class AdditionalScreens

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

This class handles the functionality of additional screens in the game. It inherits from the Singleton class to ensure only one instance of this class exists.

```
public class AdditionalScreens : Singleton<AdditionalScreens>
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton<AdditionalScreens>](#) ← AdditionalScreens

Inherited Members

[Singleton<AdditionalScreens>.m_instance](#) , [Singleton<AdditionalScreens>.Instance](#) ,
[Singleton<AdditionalScreens>.Awake\(\)](#) , [Singleton<AdditionalScreens>.Prepare\(\)](#) ,
[SingletonBase.AllSingletons](#)

Class CollapseUIScript

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

Handles the collapse and expand functionality of UI elements.

```
public class CollapseUIScript : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← CollapseUIScript

Fields

isCollapsed

Flag to indicate whether the UI elements are collapsed or not.

```
private bool isCollapsed
```

Field Value

[bool](#)

Methods

ToggleCollapse(Transform)

Toggles the collapse state of the target UI elements.

```
public void ToggleCollapse(Transform target)
```

Parameters

target Transform

The UI elements to collapse or expand.

Class DeckCanvasUI

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

Handles the display and functionality of the deck in the UI.

```
public class DeckCanvasUI : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← DeckCanvasUI

Fields

back_button

The button for going back in the UI.

```
[SerializeField]  
private Button back_button
```

Field Value

Button

cards_deck_container

The container for the cards deck in the UI.

```
public RectTransform cards_deck_container
```

Field Value

RectTransform

ui_deck_card_prefab

The prefab for the UI of the deck card.

```
public GameObject ui_deck_card_prefab
```

Field Value

GameObject

Methods

BuildDeck()

Builds the deck in the UI.

```
public void BuildDeck()
```

Class DefeatScreenUI

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

Handles the display and functionality of the defeat screen in the UI.

```
public class DefeatScreenUI : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← DefeatScreenUI

Fields

background

The UI element for displaying the background.

```
public Image background
```

Field Value

Image

background_init_color

The initial color of the background.

```
private Color background_init_color
```

Field Value

Color

button_go_home

The button for going back home.

```
[SerializeField]  
private Button button_go_home
```

Field Value

Button

button_reload_level

The button for reloading the level.

```
[SerializeField]  
private Button button_reload_level
```

Field Value

Button

exp_text

The UI element for displaying the experience points.

```
public TextMeshProUGUI exp_text
```

Field Value

TextMeshProUGUI

gold_text

The UI element for displaying the gold.

```
public TextMeshProUGUI gold_text
```

Field Value

TextMeshProUGUI

message_text

The UI element for displaying the message text.

```
public TextMeshProUGUI message_text
```

Field Value

TextMeshProUGUI

popup_form

The form for the popup in the UI.

```
public RectTransform popup_form
```

Field Value

RectTransform

popup_init_pos

The initial position of the popup.

```
private Vector3 popup_init_pos
```

Field Value

Vector3

Methods

Hide()

Hides the defeat screen.

```
public void Hide()
```

HideCoroutine()

Coroutine to hide the defeat screen with an animation.

```
public IEnumerator HideCoroutine()
```

Returns

[IEnumerator](#)

Popup()

Triggers the popup in the UI.

```
public void Popup()
```

Start()

Initializes the defeat screen.

```
public void Start()
```

UpdateScreen(int, int)

Updates the screen with the provided gold and experience points.

```
public void UpdateScreen(int gold, int exp)
```

Parameters

gold [int](#)

The amount of gold to display.

exp [int](#)

The amount of experience points to display.

Class LoadingScreenManager

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

The LoadingScreenManager class is a singleton that manages the game's loading screen.

```
public class LoadingScreenManager : Singleton<LoadingScreenManager>
```

Inheritance

[Object](#) ↗ ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton<LoadingScreenManager>](#) ← LoadingScreenManager

Inherited Members

[Singleton<LoadingScreenManager>.m_instance](#) , [Singleton<LoadingScreenManager>.Instance](#) ,
[Singleton<LoadingScreenManager>.Awake\(\)](#) , [SingletonBase.AllSingletons](#)

Fields

animating

```
private bool animating
```

Field Value

[bool](#) ↗

Indicates whether the loading screen is currently animating.

background

```
public Image background
```

Field Value

Image

Gets and sets the background image of the loading screen.

enable

```
public bool enable
```

Field Value

[bool ↗](#)

Gets and sets the state of the loading screen. If true, the loading screen is enabled.

loader_init_pos

```
private Vector3 loader_init_pos
```

Field Value

Vector3

Holds the initial position of the loading circle.

loading_circle

```
[SerializeField]  
public GameObject loading_circle
```

Field Value

GameObject

Gets and sets the loading circle object displayed on the loading screen.

text

```
public Text text
```

Field Value

Text

Gets and sets the text displayed on the loading screen.

Methods

DisableLoaderCircle()

Disables the loading circle on the loading screen.

```
public void DisableLoaderCircle()
```

EnableLoaderCircle()

Enables the loading circle on the loading screen.

```
public void EnableLoaderCircle()
```

Prepare()

Prepares the loading screen manager for operation. Override of Singleton's Prepare method.

```
public override IEnumerator Prepare()
```

Returns

[IEnumerator](#)

An IEnumerator that can be used to execute the method over time.

SetLoadingScreenText(string)

Sets the text displayed on the loading screen.

```
public void SetLoadingScreenText(string text)
```

Parameters

text [string](#)

The text to be displayed on the loading screen.

Class NavigationButtons

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

```
public class NavigationButtons : Singleton<NavigationButtons>
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ←
[Singleton](#)<[NavigationButtons](#)> ← NavigationButtons

Inherited Members

[Singleton<NavigationButtons>.m_instance](#) , [Singleton<NavigationButtons>.Instance](#) ,
[Singleton<NavigationButtons>.Awake\(\)](#) , [Singleton<NavigationButtons>.Prepare\(\)](#) ,
[SingletonBase.AllSingletons](#)

Fields

cards_button

```
[SerializeField]  
private Button cards_button
```

Field Value

Button

inventory_button

```
[SerializeField]  
private Button inventory_button
```

Field Value

Button

journey_button

```
[SerializeField]  
private Button journey_button
```

Field Value

Button

Methods

SceneChanged(object)

```
private void SceneChanged(object args)
```

Parameters

args [object](#)

Start()

```
private void Start()
```

Class ParticleManager

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

Manages particle effects for various game events, such as clearing pieces or breaking tiles.

```
public class ParticleManager : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ParticleManager

Fields

board

```
public BoardManager board
```

Field Value

[BoardManager](#)

bombFXPrefab

```
public GameObject bombFXPrefab
```

Field Value

GameObject

breakFXPrefab

```
public GameObject breakFXPrefab
```

Field Value

GameObject

clearFXPrefab

```
public GameObject clearFXPrefab
```

Field Value

GameObject

doubleBreakFXPrefab

```
public GameObject doubleBreakFXPrefab
```

Field Value

GameObject

effectsGrid

```
public GameObject effectsGrid
```

Field Value

GameObject

Methods

BombFXAt(int, int, int)

Plays the bomb effect at a specific position on the board.

```
public void BombFXAt(int x, int y, int z = 0)
```

Parameters

x [int](#)

y [int](#)

z [int](#)

BreakTileFXAt(int, int, int, int)

Plays the break tile effect based on breakable value at a specific position.

```
public void BreakTileFXAt(int breakableValue, int x, int y, int z = 0)
```

Parameters

breakableValue [int](#)

x [int](#)

y [int](#)

z [int](#)

ClearPieceFXAt(int, int, int)

Plays the clear piece effect at a specific position on the board.

```
public void ClearPieceFXAt(int x, int y, int z = 0)
```

Parameters

x [int](#)

y [int](#)

z [int](#)

PlaceFXObject(GameObject, int, int)

Places and initializes an FX object at a specified board position.

```
private GameObject PlaceFXObject(GameObject prefab, int x, int y)
```

Parameters

prefab GameObject

x [int](#)

y [int](#)

Returns

GameObject

Start()

Initializes the ParticleManager by assigning the BoardManager component.

```
private void Start()
```

Class ParticlePlayer

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

Manages the playback of particle systems attached to the game object.

```
public class ParticlePlayer : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ParticlePlayer

Fields

allParticles

```
public ParticleSystem[] allParticles
```

Field Value

ParticleSystem[]

lifetime

```
public float lifetime
```

Field Value

[float](#)

Methods

Play()

Plays all attached particle systems.

```
public void Play()
```

Start()

Initializes the particle player, gathering all child particle systems and setting up destruction.

```
private void Start()
```

Class ParticleSystemSizeAdaptation

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

Adapts the size of a particle system to match the size of its parent RectTransform.

```
public class ParticleSystemSizeAdaptation : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ParticleSystemSizeAdaptation

Fields

change_color_coroutine_main

```
private IEnumerator change_color_coroutine_main
```

Field Value

[IEnumerator](#)

change_color_coroutine_sub

```
private IEnumerator change_color_coroutine_sub
```

Field Value

[IEnumerator](#)

originalParentSize

```
private Vector2 originalParentSize
```

Field Value

Vector2

originalPosition

```
private Vector2 originalPosition
```

Field Value

Vector2

originalStartSizeX

```
private float originalStartSizeX
```

Field Value

[float](#)

originalStartSizeY

```
private float originalStartSizeY
```

Field Value

[float](#)

start_color

```
private Color start_color
```

Field Value

Methods

AdaptSize()

Dynamically adapts the particle system size based on the parent RectTransform's size.

```
private void AdaptSize()
```

Awake()

Initializes variables and adapts the particle system size.

```
private void Awake()
```

OnEnable()

Enables the adaptation logic and starts the color flashing coroutine.

```
private void OnEnable()
```

RecursiveColorFlah()

```
private IEnumerator RecursiveColorFlah()
```

Returns

[IEnumerator](#)

RestartParticle(float)

Restarts the particle system with a specified time.

```
public void RestartParticle(float itme)
```

Parameters

itme [float](#)

Start()

Sets initial state and deactivates the game object.

```
private void Start()
```

Update()

```
private void Update()
```

Class TopBarMenu

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

The TopBarMenu class manages the top bar menu.

```
public class TopBarMenu : Singleton<TopBarMenu>
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton](#)<[TopBarMenu](#)> ← TopBarMenu

Inherited Members

[Singleton](#)<[TopBarMenu](#)>.m_instance , [Singleton](#)<[TopBarMenu](#)>.Instance ,
[Singleton](#)<[TopBarMenu](#)>.Awake() , [Singleton](#)<[TopBarMenu](#)>.Prepare() , [SingletonBase](#).AllSingletons

Methods

SceneChanged(object)

Handles the SceneChanged event. It adjusts the UI depending on the current scene.

```
private void SceneChanged(object args)
```

Parameters

args [object](#)

The name of the current scene.

Start()

Start is called before the first frame update. It subscribes to the SceneChanged event.

```
private void Start()
```

Class UIManager

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

The UIManager class is a singleton that manages the UI of the game.

```
public class UIManager : Singleton<UIManager>
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [SingletonBase](#) ← [Singleton<UIManager>](#) ← UIManager

Inherited Members

[Singleton<UIManager>.m_instance](#) , [Singleton<UIManager>.Instance](#) , [Singleton<UIManager>.Awake\(\)](#) , [SingletonBase.AllSingletons](#)

Fields

bubble_text_prefab

```
public GameObject bubble_text_prefab
```

Field Value

GameObject

Gets or sets the GameObject of the bubble text prefab.

bubble_text_spawn_point

```
public RectTransform bubble_text_spawn_point
```

Field Value

RectTransform

Gets or sets the RectTransform of the bubble text spawn point.

common_canvas

```
public RectTransform common_canvas
```

Field Value

RectTransform

Gets or sets the RectTransform of the common canvas.

deck_canvas

```
[Header("Canvases")]
public RectTransform deck_canvas
```

Field Value

RectTransform

Gets or sets the RectTransform of the deck canvas.

defeat_screen_canvas

```
public RectTransform defeat_screen_canvas
```

Field Value

RectTransform

Gets or sets the RectTransform of the defeat screen canvas.

inventory_canvas

```
public RectTransform inventory_canvas
```

Field Value

RectTransform

Gets or sets the RectTransform of the inventory canvas.

level_background_image

```
private Image level_background_image
```

Field Value

Image

Gets or sets the Image of the level background.

loading_screen

```
public RectTransform loading_screen
```

Field Value

RectTransform

Gets or sets the RectTransform of the loading screen.

single_card_canvas

```
public RectTransform single_card_canvas
```

Field Value

RectTransform

Gets or sets the RectTransform of the single card canvas.

ui_deck_card_prefab

```
public GameObject ui_deck_card_prefab
```

Field Value

GameObject

Gets or sets the GameObject of the UI deck card prefab.

ui_single_card_prefab

```
[Header("Prefabs")]
public GameObject ui_single_card_prefab
```

Field Value

GameObject

Gets or sets the GameObject of the UI single card prefab.

victory_screen_canvas

```
public RectTransform victory_screen_canvas
```

Field Value

RectTransform

Gets or sets the RectTransform of the victory screen canvas.

Methods

BuildAllSingleCards()

Builds all single cards.

```
public void BuildAllSingleCards()
```

BuildDeck()

Builds the deck.

```
public void BuildDeck()
```

DisableCanvasGroup(CanvasGroup)

Disables a CanvasGroup.

```
public void DisableCanvasGroup(CanvasGroup canvas_group)
```

Parameters

canvas_group CanvasGroup

The CanvasGroup to be disabled.

DisableLoadingScreen()

Disables the loading screen.

```
[ContextMenu("Hide Loading Screen")]
public void DisableLoadingScreen()
```

DisableLoadingScreenCoroutine()

A coroutine to disable the loading screen.

```
private IEnumerator DisableLoadingScreenCoroutine()
```

Returns

[IEnumerator](#)

An IEnumerator for coroutine sequencing.

EnableCanvasGroup(CanvasGroup)

Enables a CanvasGroup.

```
public void EnableCanvasGroup(CanvasGroup canvas_group)
```

Parameters

canvas_group CanvasGroup

The CanvasGroup to be enabled.

HideCard()

Hides a card.

```
public void HideCard()
```

HideDeckPanel()

Hides the deck panel.

```
public void HideDeckPanel()
```

HideDefeatScreen()

Hides the defeat screen.

```
public void HideDefeatScreen()
```

HideInventory()

Hides the inventory.

```
public void HideInventory()
```

HideVictoryScreen()

Hides the victory screen.

```
public void HideVictoryScreen()
```

Prepare()

Prepares the UIManager.

```
public override IEnumerator Prepare()
```

Returns

[IEnumerator](#)

An IEnumerator for coroutine sequencing.

PrepareForBattleArena()

Prepares the UIManager for the Battle Arena.

```
public IEnumerator PrepareForBattleArena()
```

Returns

[IEnumerator](#)

An IEnumerator for coroutine sequencing.

SetLevelBackgroundImage(Sprite)

Sets the background image of the level.

```
public void SetLevelBackgroundImage(Sprite image)
```

Parameters

image Sprite

The Sprite to set as the background image.

SetLoadingScreenBackgroundImage(Sprite)

Sets the background image of the loading screen.

```
public void SetLoadingScreenBackgroundImage(Sprite image)
```

Parameters

image Sprite

The Sprite to set as the background image.

ShowCard(string)

Shows a card by its ID.

```
public void ShowCard(string ID)
```

Parameters

ID [string](#)

The ID of the card to be shown.

ShowDeckPanel()

Shows the deck panel.

```
public void ShowDeckPanel()
```

ShowDefeatScreen()

Shows the defeat screen.

```
public void ShowDefeatScreen()
```

ShowInventory()

Shows the inventory.

```
public void ShowInventory()
```

ShowLoadingScreen()

Shows the loading screen.

```
[ContextMenu("Show Loading Screen")]
public void ShowLoadingScreen()
```

ShowLoadingScreenCoroutine()

A coroutine to show the loading screen.

```
public IEnumerator ShowLoadingScreenCoroutine()
```

Returns

[IEnumerator](#)

An IEnumerator for coroutine sequencing.

ShowMessage(string, Color, float)

Shows a message with a bubble text effect.

```
public void ShowMessage(string message, Color color, float size = 1.5)
```

Parameters

message [string](#)

The message to be shown.

color Color

The color of the text.

size [float](#)

The size of the text.

ShowVictoryScreen()

Shows the victory screen.

```
public void ShowVictoryScreen()
```

Class VictoryScreenUI

Namespace: [TaleOfTiles.UI](#)

Assembly: Assembly-CSharp.dll

Manages the UI for the victory screen in the game, including display of rewards and navigation buttons.

```
public class VictoryScreenUI : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← VictoryScreenUI

Fields

background

```
public Image background
```

Field Value

Image

background_init_color

```
private Color background_init_color
```

Field Value

Color

button_go_home

```
[SerializeField]  
private Button button_go_home
```

Field Value

Button

button_go_next

```
[SerializeField]  
private Button button_go_next
```

Field Value

Button

button_reload_level

```
[SerializeField]  
private Button button_reload_level
```

Field Value

Button

card_prefab

```
[SerializeField]  
private GameObject card_prefab
```

Field Value

GameObject

cards_container

```
[SerializeField]
```

```
private GameObject cards_container
```

Field Value

GameObject

cards_reward_panel

```
[SerializeField]
```

```
private GameObject cards_reward_panel
```

Field Value

GameObject

exp_text

```
public TextMeshProUGUI exp_text
```

Field Value

TextMeshProUGUI

gold_text

```
public TextMeshProUGUI gold_text
```

Field Value

TextMeshProUGUI

init_size

```
private Vector2 init_size
```

Field Value

Vector2

item_prefab

```
[SerializeField]  
private GameObject item_prefab
```

Field Value

GameObject

items_container

```
[SerializeField]  
private GameObject items_container
```

Field Value

GameObject

items_reward_panel

```
[SerializeField]  
private GameObject items_reward_panel
```

Field Value

GameObject

message_text

```
public TextMeshProUGUI message_text
```

Field Value

TextMeshProUGUI

popup_form

```
public RectTransform popup_form
```

Field Value

RectTransform

popup_init_pos

```
private Vector3 popup_init_pos
```

Field Value

Vector3

Methods

Hide()

Hides the victory screen.

```
public void Hide()
```

HideCoroutine()

Coroutine to manage hiding animation for the victory screen.

```
public IEnumerator HideCoroutine()
```

Returns

[IEnumerator](#)

Popup()

Activates the popup animation for the victory screen.

```
public void Popup()
```

Start()

Initializes the victory screen UI.

```
public void Start()
```

UpdateScreen(int, int, List<CardObject>, List<EquipmentItem>)

Updates the screen with provided rewards and information.

```
public void UpdateScreen(int gold, int exp, List<CardObject> card_reward = null,  
List<EquipmentItem> equipment_reward = null)
```

Parameters

gold [int](#)

Gold earned.

exp [int](#)

Experience points earned.

`card_reward` [List](#) <[CardObject](#)>

List of card rewards.

`equipment_reward` [List](#) <[EquipmentItem](#)>

List of equipment rewards.