Match 3 Game Core Systems Documentation

1. GlobalEnums

Feature overview

- Holds shared enumerations that define global types and states.

Items

- GemType

Purpose: Represents the logical color or type of a gem on the board (Blue, Green, Red, Yellow, Purple, Bomb).
Integration: Used by ScGem, ScBomb, GameBoard, and ScGameLogicManager to compare and reason about gem types.
Notes: Centralizes type values so that code does not rely on magic numbers or strings.

- GameState

Purpose: Represents the current interaction state of the game (Wait or Move).
Integration: Used by ScGameLogicManager and ScGem to gate player input and to avoid moves while the board is resolving.
Notes: Simple state enum that supports a small state machine for user input.

2. ScGameVariables (ScriptableObject)

Feature overview

- Central configuration asset for visual and gameplay tuning.

Fields (no methods)

- bgTilePrefabs

Purpose: Prefab for background tiles under each board cell.
Integration: Used by ScGameLogicManager.Setup to build the background grid.

- bomb

Purpose: Prefab for bomb pieces.
Integration: Used by ScGameLogicManager to spawn bombs when matches are large enough.

- gems

Purpose: Array of regular gem prefabs.
Integration: Used by ScGameLogicManager.GetRandomGemForPosition when spawning or refilling the board.

- bonusAmount, bombChance, dropHeight, gemSpeed, scoreSpeed, cascadeDelay, bombNeighborDestroyDelay, bombSelfDestroyDelay

Purpose: Tunable floats that control scoring bonus, bomb creation chance, spawn height, movement speed, score animation speed, and cascade and bomb timing.
Integration: Read by ScGameLogicManager and ScGem to control timing and animation behavior.
Notes: Keeps gameplay tuning data out of code.

- RowsSize, ColsSize

Purpose: Board dimensions exposed as read only properties.
Integration: Used by ScGem and other systems to know grid limits.

3. GameBoard

Feature overview

- Pure data and logic model of the board grid.
- Owns the 2D array of gems and all match detection logic including bomb aware matching.

Methods

- GameBoard(int width, int height)

Purpose: Constructor that sets board size and allocates the 2D array of ScGem.
Integration: Called by ScGameLogicManager.Init to build the game board model.

- bool MatchesAt(Vector2Int positionToCheck, ScGem gemToCheck)

Purpose: Predicts whether placing a given gem at a specific cell would create a match.

Integration: Used by ScGameLogicManager.GetRandomGemForPosition to avoid spawning immediate matches when filling.
Design notes: Uses horizontal and vertical checks and the AreInSameMatchGroup helper so that bombs and colored bombs participate correctly.
- void SetGem(int x, int y, ScGem gem)
Purpose: Stores a gem reference at a given board coordinate.
Integration: Called by ScGameLogicManager and ScGem whenever pieces move or are spawned or removed.
Notes: Keeps board model in sync with the scene.
- ScGem GetGem(int x, int y)
Purpose: Returns the gem stored at a given board coordinate.
Integration: Used by ScGem for swipe logic and by ScGameLogicManager for destruction and refills.
- void FindAllMatches()
Purpose: Scans the entire board and populates an internal list of currently matched gems.
Integration: Called after moves and cascades by ScGameLogicManager before DestroyMatches.
Design notes: Uses TryAddHorizontalMatch, TryAddVerticalMatch, and CheckForBombs to collect both regular chains and bomb interactions.
- bool AreInSameMatchGroup(ScGem a, ScGem b) (private)
Purpose: Decides if two pieces should be treated as part of the same match group.
Integration: Used by MatchesAt and the directional match checks.
Design notes: Delegates to ScBomb.CanInteractWith to support generic bombs and colored bombs rather than only comparing GemType.
- void TryAddHorizontalMatch(int x, int y, ScGem currentGem) (private)
Purpose: Checks neighbors on the left and right of the current cell and adds them to matches if they form a horizontal line.
Integration: Called inside FindAllMatches for each cell.
Notes: Detects both regular three in a row and special double bomb matches, then calls MarkAsMatch.
- void TryAddVerticalMatch(int x, int y, ScGem currentGem) (private)
Purpose: Same as TryAddHorizontalMatch but for vertical neighbors above and below.
Integration: Called by FindAllMatches for each cell.
- bool CheckDoubleBombMatch(ScGem firstGem, ScGem secondGem) (private)
Purpose: Detects when both gems in a pair are bombs, marking a special double bomb interaction.
Integration: Used by both horizontal and vertical match checks before regular color based matching.
- void MarkAsMatch(List gems) (private)
Purpose: Marks the given gems as matched and stores them in the internal match list.
Integration: Core helper for both simple and bomb based match detection.
- void CheckForBombs() (private)
Purpose: After initial match detection, expands matches around bombs that are adjacent to matched pieces.
Integration: Called at the end of FindAllMatches.
Design notes: Looks at four directional neighbors around each matched gem, checks for ScBomb neighbors that can interact, and then calls MarkBombArea to grow the match.
- void MarkBombArea(Vector2Int bombPos, int blastSize) (private)
Purpose: Marks all gems in a square area around a bomb as matched.
Integration: Used by CheckForBombs when a bomb is triggered by a neighboring match.
Design notes: Uses the gem.blastSize value and board bounds to avoid out of range errors.
4. ScGem (base gem behaviour)
==========
Feature overview
- Component attached to gem objects.
- Handles movement toward target cell positions, swipe input and communication with the game logic manager.
Methods

- void Awake()
Purpose: Caches Camera.main and resets board registration flags.
Integration: Called automatically by Unity when the gem object is created.
- void Update()
Purpose: Per frame update entry point for gem behaviour.
Integration: Calls UpdatePosition and HandleInput each frame.
- void UpdatePosition() (private)
Purpose: Moves the gem towards its logical grid position and registers its location in the GameBoard.
Integration: Uses _gameVariables.gemSpeed for smooth movement and ScGameLogicManager.SetGem to keep the board model up to date.
Design notes: Only updates the board when the gem actually reaches or changes its cell to avoid redundant writes.
- void HandleInput() (private)
Purpose: Handles the end of a click and drag and triggers swipe processing.
Integration: Reads mouse input, checks that the game state is Move, and then calls CalculateAngleAndMove.
- void OnMouseDown() (private)
Purpose: Captures the starting mouse position and flags that a swipe gesture has begun.
Integration: Only active when the game state is Move. Uses Camera to convert screen to world position.
- void CalculateAngleAndMove() (private)
Purpose: Computes the swipe direction from first to final touch position and selects the target neighbor cell.
Integration: Sets the _swipeAngle value and then calls MovePieces if the swipe distance passes the minimum threshold.
- void MovePieces() (private)
Purpose: Computes the neighbor gem to swap with and updates both gems posIndex values and the board mapping.
Integration: Calls ScGameLogicManager.GetGem and ScGameLogicManager.SetGem to update the GameBoard, then starts CheckMoveCo.
Design notes: Validates that swipes stay inside the board dimensions from ScGameVariables.
- IEnumerator CheckMoveCo() (private)
Purpose: Coroutine that verifies if the swap created any matches and decides whether to keep or revert the move.
Integration: Sets the game state to Wait, yields for a short delay, triggers ScGameLogicManager.FindAllMatches, and then either reverts positions or calls DestroyMatches.
Design notes: Encapsulates swap validation and ensures that players cannot move while the board is resolving.
- void SetupGem(ScGameLogicManager gameLogicManager, ScGameVariables gameVariables, Vector2Int position)
Purpose: Injects dependencies and initial position into the gem after it is spawned.
Integration: Called by ScGameLogicManager.SpawnGem for every new gem or bomb.
5. ScBomb (special gem)
==========
Feature overview
- Specialized gem that behaves as a bomb and carries a logical color for interaction rules.
Methods
- void SetBombColor(GemType bombColor)
Purpose: Sets the logical color of the bomb and updates its visual sprite color.
Integration: Called by ScGameLogicManager.CreateBombAt when the bomb is spawned based on the source gem.
Design notes: Encapsulates both logic and visuals so that other systems only choose the color.
- bool CanInteractWith(ScGem other)

Purpose: Decides whether this bomb should interact with another gem during matching.
Integration: Used by GameBoard.AreInSameMatchGroup and CheckForBombs to implement colored bomb rules.
Design notes: Generic bombs (GemType.Bomb) interact with any gem; colored bombs interact with gems or bombs of matching color or generic bombs.
- void SetImageColor() (private)
Purpose: Maps the logical GemType to a sprite color for visual feedback.
Integration: Called internally by SetBombColor.
6. ScGemPoolService
==========
Feature overview
- Object pool for gem and bomb instances to avoid frequent allocations and Destroy calls.
Methods
- ScGemPoolService(Transform parent)
Purpose: Constructor that stores the parent Transform for pooled objects.
Integration: Created by ScGameLogicManager.Init.
- void Prewarm(ScGem prefab, int count)
Purpose: Preallocates a number of inactive instances of a given prefab.
Integration: Can be used at startup to avoid runtime instantiation spikes.
Design notes: Stores instances in a queue and tracks their prefab in _instanceToPrefab.
- ScGem Get(ScGem prefab)
Purpose: Retrieves an active instance for the given prefab from the pool or instantiates a new one.
Integration: Used by Spawn and can also be used directly if needed.
Design notes: Ensures that returned objects are parented under the configured pool parent.
- ScGem Spawn(ScGem prefab, Vector3 position)
Purpose: Convenience wrapper that gets a pooled gem and places it at a given world position.
Integration: Used by ScGameLogicManager when spawning board pieces and bombs.
- void Release(ScGem gem)
Purpose: Returns an instance to its pool and deactivates it.
Integration: Called by ScGameLogicManager.ReturnGemToPool when gems are destroyed during matches.
Design notes: If the pool cannot identify the prefab for an instance, it falls back to destroying it to avoid leaks.
7. ScoreView
==========
Feature overview
- Simple UI adapter that displays the current score.
Methods
- void SetScore(float score)
Purpose: Sets the text value of the TMP text field to show the score rounded to an integer.
Integration: Called by ScGameLogicManager.UpdateScoreTask whenever the score display changes.
8. ScGameLogicManager
==========
Feature overview
- Central game controller for the match 3 loop.
- Owns GameBoard, ScGemPoolService, score state, and the current GameState.
- Controls spawning, matching, bomb creation, destruction, cascades, refills, and score animation.
Public API and properties
- GameState CurrentState
Purpose: Exposes the current game interaction state (Wait or Move) to other components, especially ScGem.
Integration: Used by ScGem to know when input is allowed.
- void SetGem(int x, int y, ScGem gem)

Purpose: Forwards board updates to GameBoard.
Integration: Used by ScGem and internal logic to keep the model in sync.
- ScGem GetGem(int x, int y)
Purpose: Forwards lookups to GameBoard.
Integration: Used by ScGem swipe logic and by refill logic.
- void SetState(GameState currentState)
Purpose: Changes the interaction state when the board needs to lock or unlock input.
Integration: Used by ScGem and internal coroutines.
- void DestroyMatches()
Purpose: Entry point for resolving currently detected matches on the board.
Integration: Called by ScGem.CheckMoveCo and by cascade logic after refills.
Design notes: Splits matching pieces into regularMatches and bombMatches, then routes to HandleRegularMatchesOnly or DestroyMatchesWithBombsCo.
- void FindAllMatches()
Purpose: Convenience wrapper that asks GameBoard to rebuild the match list.
Integration: Used by ScGem.CheckMoveCo and by FilledBoardCo.
Lifecycle and setup
- void Awake()
Purpose: Unity lifecycle hook that calls Init at startup.
- void Init() (private)
Purpose: Creates GameBoard and ScGemPoolService and then calls Setup.
Integration: Centralizes creation of internal services.
- void Setup() (private)
Purpose: Builds the initial board: instantiates background tiles, picks random gem prefabs, and spawns them into all board cells.
Integration: Uses gameVariables, GameBoard, and ScGemPoolService.
Spawning and bomb support
- ScGem GetRandomGemForPosition(Vector2Int position) (private)
Purpose: Picks a random gem prefab for a given cell while avoiding immediate matches.
Integration: Used by Setup and RefillBoardCo.
Design notes: Uses GameBoard.MatchesAt in a loop with a maximum iteration count to prevent infinite loops.
- void SpawnGem(Vector2Int position, ScGem gemToSpawn) (private)
Purpose: Spawns a gem from the pool at a given logical position, sets its name, board cell, and calls SetupGem.
Integration: Used by setup, refills, and bomb creation routines.
Notes: Adds the gem to the bomb set if the spawned prefab is the global bomb prefab.
- void CreateBombAt(Vector2Int position, ScGem sourceGem) (private)
Purpose: Replaces whatever is at the given cell with a bomb and configures its logical color based on the source gem.
Integration: Called by HandleRegularMatchesOnly and DestroyMatchesWithBombsCo when large enough matches occur.
Design notes: Reuses existing bomb color if the source is already a bomb; otherwise uses the source gem type.
Match resolution and bombs
- void HandleRegularMatchesOnly(List regularMatches) (private)
Purpose: Handles the case where there are matched pieces but no bombs in the match list.
Integration: Optionally selects a bomb source for matches with 4 or more pieces, spawns the bomb, and destroys the rest.
Notes: Ends by starting DecreaseRowCo to trigger cascades.
- ScGem SelectBombSourceFromRegularMatches(List regularMatches) (private)
Purpose: Chooses which matched gem should be upgraded into a bomb.
Integration: Currently uses a simple strategy (first match), but can be extended to prefer the swapped

gems.
- IEnumerator DestroyMatchesWithBombsCo(List regularMatches, List bombMatches) (private)
Purpose: Coroutine that handles complex interactions when bombs participate in a match.
Integration: Decides whether a new bomb should be created, triggers scoring and destruction of regular matches, then processes bomb explosions.
Design notes: Supports double bomb interactions by recognizing when at least two bombs are in the bombMatches list and skipping new bomb creation in that case.
- IEnumerable GetBombNeighborPositions(Vector2Int center) (private)
Purpose: Yields the coordinates of neighbors around a bomb within a 3x3 block excluding the center.
Integration: Used in DestroyMatchesWithBombsCo to know which neighboring cells should be destroyed by bomb explosions.
- void DestroyMatchedGemsAt(Vector2Int pos) (private)
Purpose: Spawns destroy effects, returns the gem to the pool, and clears the board cell.
Integration: Called by both regular and bomb match resolution paths.
Cascades and refills
- IEnumerator DecreaseRowCo() (private)
Purpose: Handles falling of gems after matches are removed by compressing columns and letting gems drop into empty cells.
Integration: Called after match destruction to create cascades and variation in the board.
Design notes: Uses cascadeDelay between movements to create visible falling animations and then starts FilledBoardCo.
- IEnumerator FilledBoardCo() (private)
Purpose: Orchestrates the refill and subsequent auto matches until the board stabilizes.
Integration: Calls RefillBoardCo, waits, then asks GameBoard to find matches and either destroys them or returns control to the player.
- IEnumerator RefillBoardCo() (private)
Purpose: Iterates over all board cells and spawns new gems for any empty positions.
Integration: Uses GetRandomGemForPosition and SpawnGem.
Notes: Ensures that after destruction and falling, the board is always fully populated.
Scoring and pooling helpers
- async Task UpdateScoreTask() (private)
Purpose: Smoothly interpolates the displayed score toward the actual score using a small step each frame.
Integration: Called from ScoreCheck and updates ScoreView.SetScore.
Design notes: Uses an async loop with Task.Yield and a cancellation token to stop on destruction.
- void ScoreCheck(ScGem gemToCheck) (private)
Purpose: Adds the gem scoreValue to the total score and starts the score animation task.
Integration: Called whenever a gem is destroyed, both for regular matches and bomb explosions.
- void ReturnGemToPool(ScGem gem) (private)
Purpose: Removes a gem from the local bomb set if needed and returns it to the pool.
Integration: Called by DestroyMatchedGemsAt and CreateBombAt.
- bool IsBomb(ScGem gem) (private)
Purpose: Checks whether a gem is registered in the internal bomb set.
Integration: Used by DestroyMatches to separate bombMatches from regularMatches.
Maintenance
- void CheckMisplacedGems() (private)
Purpose: Finds any ScGem instances in the scene that are not referenced in the GameBoard and cleans them up.
Integration: Useful as a safety net to prevent orphan objects from staying in the scene.
Summary
=======
The system is structured around clear responsibilities:
- GameBoard acts as a pure data and match detection model.

- ScGem and ScBomb handle piece level behaviour and bomb specific rules.
- ScGemPoolService manages allocation and reuse of gem objects.
- ScGameVariables centralizes configuration and tunable values.
- ScoreView provides a thin adapter from score value to UI.
- ScGameLogicManager coordinates all of the above to implement the match 3 loop, including spawning, input gating, matching, bombs, cascades, refills, and scoring.

This separation follows core SOLID principles: each class has a focused responsibility, dependencies are injected explicitly where needed, and shared rules like gem type and game state are kept in common enums and configuration assets.