The core game mechanics of your alphabet tracing game rely on a hierarchical structure: the **Letter** is composed of **Parts**, and each **Part** is defined by a sequence of **Waypoints**. Tracing involves following these waypoints in order.

Here is a comprehensive breakdown of the mechanics, initiation, visuals, audio, and scene management based on the provided scripts:

Core Tracing Mechanics and Win Condition

1. Letter Structure and Parts Initiation (Example: Letter 'A' has 3 parts)

The overall tracing progression is managed by the LetterController.cs script.

- **Initiation and Access:** The individual segments of the letter (e.g., the 3 parts of 'A') are represented by a collection of TracePart objects stored in a **list** called m_TraceParts. This list acts as the array of parts.
    - Developers must manually assign these TracePart objects to the list in the Unity Inspector, ensuring they are in the exact sequential order required for tracing (e.g., left stroke, right stroke, crossbar).
    - A private integer, m_CurrentPartIndex, is used to access the list and track which part is currently active, starting at index 0.
- **Sequential Tracing:** During InitializeLetter(), all TracePart scripts are disabled (part.enabled = false) to prevent the player from tracing them out of order. Only the current part (at m_CurrentPartIndex) is enabled and activated via ActivateCurrentPart().
- **Finishing the Letter (Game Win):** When a single part is completed, the OnPartCompleted() method in LetterController increments the m_CurrentPartIndex. If this index is now greater than or equal to the total count of m_TraceParts, the whole letter is complete. This triggers the OnLetterCompleted event, which the GameManager listens to in order to display the 'You Win!' panel (m_WinPanel) and play the completion sound.

2. Joining Waypoints to Finish One Part

The individual tracing of a single part is handled by the TracePart.cs script.

- **Waypoint Setup:** The path for tracing is defined by the **Way points with multiple child points to track the tracing**. These Waypoints are loaded into a list called m_Waypoints by iterating through every child object of the m_WaypointsContainer Transform when the script loads.
- **Tracing Detection:** When the player drags the mouse (OnMouseDrag), the script calculates the mouse's position in the game world.
- **Waypoint Collection:** The script checks if the mouse position is close enough to the next required waypoint (m_Waypoints[m_CurrentWaypointIndex]). The proximity requirement is defined by the WAYPOINT_THRESHOLD (0.5f).
- **Part Completion:** If the mouse reaches a waypoint, m_CurrentWaypointIndex (for the waypoint list) is incremented. The part is considered completed when the waypoint index

reaches or exceeds the total number of waypoints (m_CurrentWaypointIndex >= m_Waypoints.Count). At this point, CompleteTrace() is called.

3. Restart Drawing if Player Misses in the Middle

If the player fails to successfully trace a part, the progress resets, forcing them to **restart drawing that part of the letter**.

- **Reset Triggers:**
    1. **Releasing the Mouse:** If the player releases the mouse button (OnMouseUp) before the part is finished (!IsCompleted) while actively tracing, the ResetTrace() function is called.
    2. **Exiting the Collider:** If the mouse cursor leaves the area of the part's collider (OnMouseExit) while tracing (m_IsTracing), ResetTrace() is called.
- **Reset Action:** The ResetTrace() function sets the tracing progress back to the beginning:
    - The next required waypoint index is reset (m_CurrentWaypointIndex = 0).
    - The fill sprite color reverts to the semi-transparent m_GuideColor.
    - All associated visual feedback (the little green dots) on the waypoints are deactivated (waypoint.Deactivate()).
    - The m_TraceResetSound is played.

4. Visual Waypoint Activation

The visual sprites are spawned by making it SetActive true after completing every way point.

- When the player successfully reaches a waypoint and m_CurrentWaypointIndex is incremented, the ActivateNextWaypointVisual() function is called.
- This function targets the waypoint that was just passed (at m_CurrentWaypointIndex - 1) and calls its Activate() method.
- The Waypoint.cs script handles the visual feedback: its Activate() function sets the assigned visual GameObject (m_WaypointVisual) to true (SetActive(true)).
- Additionally, tracing progress is visually represented by the m_FillSprite's transparency, which updates based on the ratio of collected waypoints to the total count (progress = m_CurrentWaypointIndex / m_Waypoints.Count).

The 3 Sounds

The sources define three distinct sounds used across the game scripts:

Sound Name	Source Variable	Trigger/Purpose	Script
Letter Completed Sound	m_LetterCompletedSound	A "success" or "congratulations" sound that plays when the whole letter is finished.	GameManager.cs

Part Completed Sound	m_PartCompletedSound	Plays when a single part of the letter has been successfully traced.	TracePart.cs
Trace Reset Sound	m_TraceResetSound	Plays when the player makes a mistake (e.g., releases the mouse prematurely or exits the collider) and the current trace is reset.	TracePart.cs

---

Scene Access

Scene management is handled by the SceneLoader.cs script. The scripts provide public methods that allow the **3 scenes with letters 'A', 'B' and 'C' to be accessed from the Home Scene**.

- **Accessing Letter 'A':** The LoadLetterA() method uses SceneManager.LoadScene("Letter A").
- **Accessing Letter 'B':** The LoadLetterB() method uses SceneManager.LoadScene("Letter B").
- **Accessing Letter 'C':** The LoadLetterC() method uses SceneManager.LoadScene("Letter C").
- **Accessing Home Scene:** The LoadHome() method uses SceneManager.LoadScene("Home") to take the player back to the main menu scene.