You Are What You Keep — Complete Project Document

Version 5 — Final Master Document

> This document is a complete record of everything built, every decision made, all known issues, and the full technical roadmap. It is written to allow any developer or LLM to pick up this project with full context. Read it top to bottom before touching anything.

Table of Contents

PART 1 — GAME MANIFESTO

Working Title

You Are What You Keep

Core Philosophy

The goal is not to win — the goal is to live a meaningful life inside a system. This game rejects traditional progression loops (XP, loot, grinding) in favour of emotional payoff, player-authored meaning, and systemic emergence. The game should create memories, not milestones.

Player Objective

There is no explicit win state. The player is trying to live an interesting life, accumulate meaningful moments, and shape who they become through what they experience. Endings are reflective, not victorious.

Design Pillars

Moments over Progression — The primary reward is a felt experience: awe, regret, nostalgia, calm, pride. If a mechanic exists, it must serve a moment.

Identity as Inventory — The player collects memories, traits, and ways of being. Your build is what you've lived, what you've kept, what you've let go of.

Time as Gentle Pressure — Time is always present but not punishing. Bittersweet, inevitable, motivating. Time creates meaning by making things finite.

Systems That Respect the Player — Avoid over-explaining. Trust player interpretation. Allow ambiguity. Meaning emerges through interaction, not participation.

Self-Authored Reward — The best moments are not scripted. The game enables useless but beautiful actions, personal rituals, quiet discovery.

Core Mechanics

Memories as Inventory — Players collect memories instead of items. Each memory occupies a limited slot, subtly alters gameplay or perception, and shapes identity. You cannot keep everything. Forgetting is a mechanic.

Limited Inner Space — 5–8 memory slots. Creates emotional tradeoffs, personal curation, identity evolution. Letting go should feel significant.

Moments That Transform — Certain experiences permanently change the player. A near-death fall becomes fearlessness or fragility. Deep solitude becomes heightened awareness. Transformation replaces levelling.

The World Remembers — Meaningful actions leave subtle echoes. A place where you linger becomes warmer. The world holds the shape of your presence.

Skills as Lived Experience — Skills emerge from behaviour and fade when abandoned. Climb often → natural agility. Sit quietly → heightened awareness.

Tone and Aesthetic

Warm, reflective, slightly melancholic, hopeful. Not bleak. Not cynical. Stylised realism or soft minimalism. Strong lighting language. Expressive colour shifts. Calm environmental storytelling.

Reference points: Journey, Flower (thatgamecompany), Firewatch, Gris.

One-Line Summary

A game about becoming someone through the moments you choose to keep.

PART 2 — PROJECT SETUP & TOOLS

Item	Value
Engine	Unity (Latest LTS)
Render Pipeline	URP (Universal Render Pipeline)
Input System	New Unity Input System (package installed)
Version Control	Git + GitHub, managed via GitHub Desktop
Text Rendering	TextMeshPro (package installed, essentials imported)

Unity Project Settings

- Asset Serialization Mode: **Force Text**
- Version Control Mode: **Visible Meta Files**

Post Processing — Critical Setup Notes

- Global Volume lives in `_Persistent` scene
- Profile must have Color Adjustments, Bloom, Vignette added as overrides
- Every **individual value checkbox** in the profile must be **ticked blue** — grey/unticked values cannot be changed by scripts at runtime
- `PlayerCamera` must have **Post Processing enabled** in Camera component (or Universal Additional Camera Data component) — this is separate from the Volume setup and is required for effects to actually render

Audio — Critical Setup Notes

- `PlayerCamera` must have an **Audio Listener** component
- There must be **exactly one** Audio Listener in the scene
- Audio clips for ambient layers go in `Assets/_Game/Audio/Ambient/`
- Moment sting clip uses a time offset (`momentStingSource.time = 0.3f`) to skip dead air — adjust value as needed per clip

Commit History

1. Initial project setup, folder structure, scenes, Git
2. Phase 2: Player controller with weighted movement and camera bob
3. Phase 3: Memory system — data layer, moment triggers, first memories

4. Phase 4 complete: memory UI fully working — slots, prompts, reflection screen, replace mode, HUD dots
5. Phase 5 complete: emotional response system — post processing world state, layered audio, moment bloom response
6. Phase 6: Identity and trait system — traits derived from memories, movement and perception modifiers
7. Phase 7: Time system and world echo — vividness decay, echo points, time-driven atmosphere
8. Phase 8: Ending and reflection sequence — narrator, passage system, personalised ending

---

PART 3 — FOLDER STRUCTURE

```
Assets/
├── _Game/
│   ├── Art/
│   │   ├── Characters/
│   │   ├── Environment/
│   │   ├── UI/
│   │   └── VFX/
│   ├── Audio/
│   │   ├── Ambient/          ← AMB_ audio files
│   │   ├── Music/
│   │   └── SFX/
│   ├── Materials/
│   ├── Prefabs/
│   │   ├── Characters/       ← PFB_Player
│   │   ├── Environment/
│   │   ├── Memories/         ← MEM_MomentTrigger_Base
│   │   ├── Systems/          ← --- SYSTEMS --- prefab
│   │   └── UI/               ← UI_Canvas, UI_MemoryCard, UI_SlotDot
│   ├── Scenes/
│   │   ├── Core/             ← _Boot, _Persistent
│   │   └── Locations/        ← Location_Opening
│   ├── Scripts/
│   │   ├── Core/             ← GameManager, AudioManager, EmotionalResponseSystem,
│   │   │                       EndingSystem, EndingNarrator
│   │   ├── Identity/         ← IdentitySystem
│   │   ├── Memories/         ← MemoryData, MemoryInstance, MemorySystem
│   │   ├── Player/           ← PlayerController, CameraController
│   │   ├── Time/             ← TimeSystem
│   │   ├── World/            ← MomentTrigger, WorldEchoSystem, EndingTrigger
│   │   └── UI/               ← UIManager, MomentPromptUI, MemorySlotHUD,
│   │                           MemoryReflectUI, MemoryCardUI, EndingUI
│   └── ScriptableObjects/
│       ├── Memories/         ← MEM_ assets
```

```
│   ├── Traits/
│   └── Moments/
└── ThirdParty/
```

Prefab Naming Convention

- `SYS_` — system objects
- `ENV_` — environment pieces
- `MEM_` — memory-related objects
- `UI_` — interface elements
- `CHR_` — characters
- `PFB_` — player prefab

PART 4 — SCENE ARCHITECTURE

Build Settings Order

1. `_Boot`
2. `_Persistent`
3. `Location_Opening`

_Persistent Scene Hierarchy

```
_Persistent
└── --- SYSTEMS ---          ← DontDestroyOnLoad set HERE and ONLY HERE
    ├── GameManager          ← GameManager.cs
    ├── MemorySystem         ← MemorySystem.cs
    ├── AudioManager         ← AudioManager.cs
    ├── EmotionalResponseSystem  ← EmotionalResponseSystem.cs
    ├── IdentitySystem       ← IdentitySystem.cs
    ├── TimeSystem           ← TimeSystem.cs
    ├── WorldEchoSystem      ← WorldEchoSystem.cs
    └── EndingSystem         ← EndingSystem.cs
```

Critical rules:

- `--- SYSTEMS ---` must be at the **absolute root** of `_Persistent` — not nested under any parent object
- Only the root `--- SYSTEMS ---` calls `DontDestroyOnLoad`
- Child systems do NOT call `DontDestroyOnLoad` — they inherit persistence from the root
- Each system's Awake uses `transform.SetParent(null)` before `DontDestroyOnLoad` as a safety net for additive scene loading

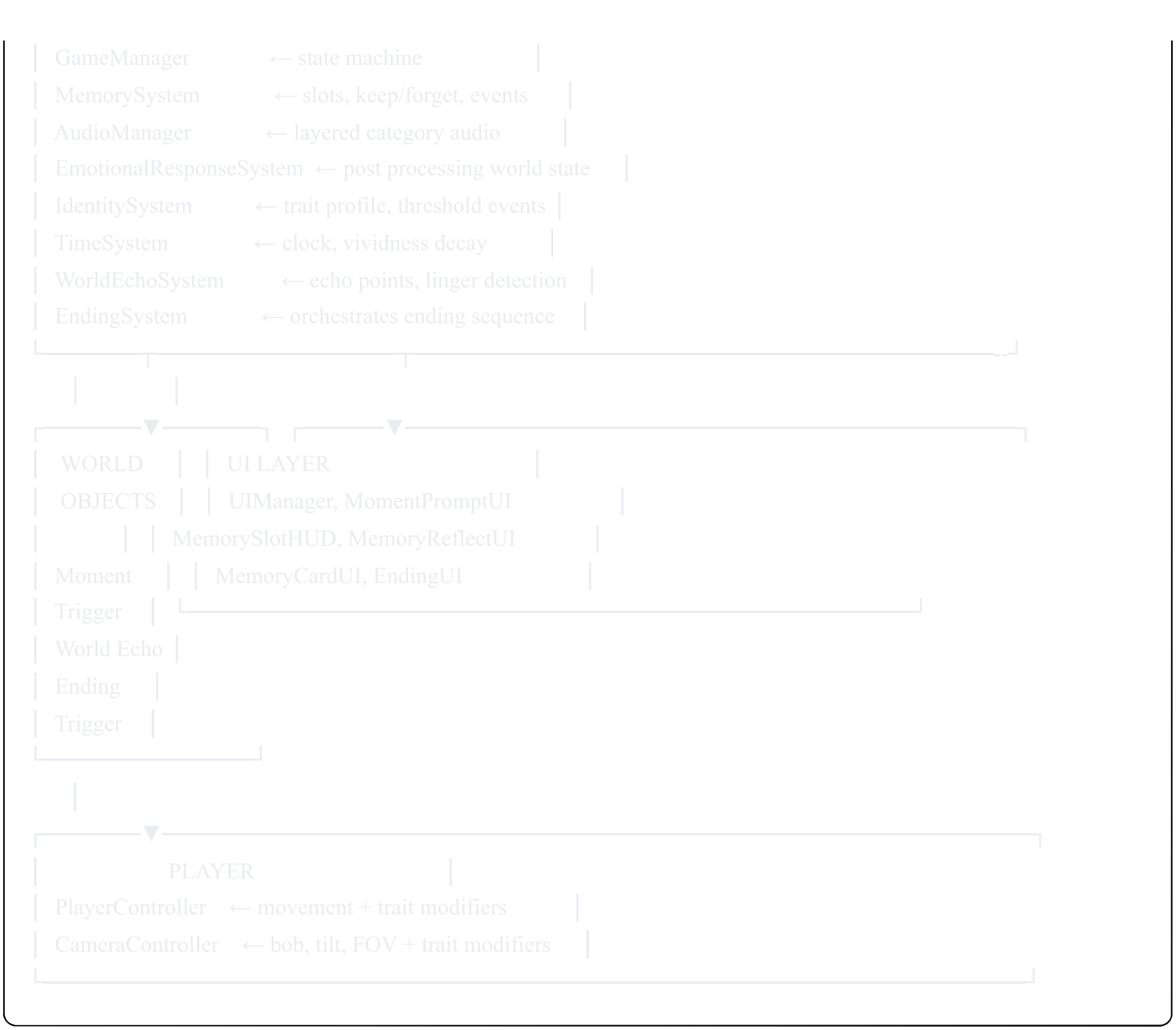
- After adding/changing anything in this hierarchy: select `--- SYSTEMS ---` → **Overrides → Apply All to Prefab**

Location_Opening Scene Hierarchy

```
Location_Opening
├── Player                    ← PFB_Player (Audio Listener on PlayerCamera)
├── [Environment geometry]    ← floor on Ground layer
├── Moment_[Name]             ← MomentTrigger objects
├── Ending_Leave              ← EndingTrigger object
└── UICanvas                  ← UI_Canvas prefab
    ├── HUD
    │   ├── MemorySlotHUD
    │   └── MomentPrompt
    ├── Screens
    │   ├── MemoryReflect
    │   └── EndingScreen      ← EndingUI.cs attached here
    └── MasterFade            ← Must be named exactly "MasterFade"
```

Cross-scene reference rule: `EndingSystem` lives in `_Persistent` but needs `EndingUI` and `MasterFade` from the location scene. These cannot be wired in the Inspector. Instead, `EndingSystem.TriggerEnding()` finds them at runtime via `FindFirstObjectByType<EndingUI>()` and `GameObject.Find("MasterFade")`. This means the `MasterFade` object must be named exactly `MasterFade` (case sensitive) in every location scene.

PART 5 — SYSTEMS ARCHITECTURE OVERVIEW



Communication Pattern

All systems communicate through **C# events**, not direct references. `MemorySystem` fires events. All other systems subscribe. No system needs to know another exists. This keeps code decoupled and maintainable.

Event Flow (complete)

```
MomentTrigger.ExperienceMoment()
  → MemorySystem.OfferMemory()
    → MemorySystem.OnMemoryKept fired
      → AudioManager subscribes      → fades in category layer, plays sting
      → EmotionalResponseSystem      → recalculates world state, moment bloom
      → IdentitySystem               → shifts trait values
      → WorldEchoSystem              → registers echo at player position
      → MemorySlotHUD                → refreshes dot indicators
      → MemoryReflectUI (if open)    → rebuilds card list

TimeSystem.Update()
  → hourly: TickVividnessDecay()
    → MemorySystem.NotifyMemoriesChanged()
```

```
        → EmotionalResponseSystem      → recalculates colour blend
        → MemoryReflectUI              → cards update vividness opacity

IdentitySystem.ShiftTrait()
  → OnTraitThresholdCrossed fired
      → EmotionalResponseSystem        → world surge + audio boost
      → PlayerController reads traits  → movement modifiers applied each frame
      → CameraController reads traits  → FOV + bob modifiers applied each frame

EndingTrigger.TriggerEnding()
  → EndingSystem.TriggerEnding()
      → GameManager.SetGameState(Ending)  → freezes player input
      → FadeToBlack + FadeAudioOut
      → EndingNarrator.GenerateEnding()   → reads all systems, generates passages
      → EndingUI.ShowPassages()           → displays personalised reflection
```

PART 6 — SCRIPTS COMPLETE REFERENCE

GameManager.cs

Path: `Assets/_Game/Scripts/Core/GameManager.cs` **Attached to:** `GameManager` (child of `--- SYSTEMS ---`)
Pattern: Singleton — `GameManager.Instance`

Game States:

```
csharp

Booting, Playing, Reflecting, Transitioning, Ending
```

Note: `CurrentState` temporarily hardcoded to `GameState.Playing` in Awake for testing.

Awake pattern:

```
csharp

private void Awake()
{
    if (Instance != null && Instance != this) { Destroy(gameObject); return; }
    Instance = this;
    transform.SetParent(null);      // safety for additive scene loading
    DontDestroyOnLoad(gameObject);
}
```

Public API:

```csharp
GameManager.Instance.SetGameState(GameState state)
GameManager.Instance.IsPlaying()              // bool
GameManager.Instance.CurrentState             // GameState
GameManager.Instance.OnGameStateChanged       // event Action<GameState>
```

PlayerController.cs

Path: `Assets/_Game/Scripts/Player/PlayerController.cs` **Attached to:** Root `Player` object **Prefab:** `Assets/_Game/Prefabs/Characters/PFB_Player` **Requires on same object:** Rigidbody, CapsuleCollider **Tag:** `Player`

Rigidbody: Mass 70, Drag 5, Angular Drag 999, Freeze Rotation XYZ **Capsule:** Height 1.8, Radius 0.4, Center Y 0.9 **Floor geometry:** Must be on `Ground` layer for jump ground check

Base Inspector values:

Field	Default
Walk Speed	3.5
Acceleration	8
Deceleration	12
Jump Force	6 (uses VelocityChange)
Gravity Multiplier	2.5
Look Sensitivity	0.15
Vertical Look Clamp	80

Jump uses ForceMode.VelocityChange — ignores Rigidbody mass, value of 6-8 feels natural.

Trait modifier fields (Phase 6):

```csharp
float agileSpeedBonus = 1.5f
float fearlessJumpBonus = 2f
float fragileSpeedPenalty = 0.8f
float fragileGravityBonus = 1.5f
float fragileJumpPenalty = 1f
```

Traits apply as: `effectiveValue = baseValue + (traitStrength * modifier * 2f)` where `traitStrength = GetTraitStrength()` returns 0–0.5 above neutral.

CameraController.cs

Path: `Assets/_Game/Scripts/Player/CameraController.cs` **Attached to:** `PlayerCamera` **Tagged:** MainCamera **Has:** Camera component, Audio Listener

Player Hierarchy:

```
Player ← PlayerController, Rigidbody, CapsuleCollider, Tag:Player
├── CameraRoot (Y pos: 1.7 — eye height)
│   └── PlayerCamera ← Camera, CameraController, Audio Listener, Tag:MainCamera
└── BodyRoot
```

Trait perception modifiers:

- Calm → reduces camera bob amplitude
- Curious → increases FOV (base 60, max +8 at full trait)
- Fragile → slightly narrows FOV (max -3 at full trait)
- FOV clamped 50–80 degrees

Public:

```csharp
TriggerEmotionalMoment(float intensity)   // gentle push-in coroutine
```

MemoryData.cs

Path: `Assets/_Game/Scripts/Memories/MemoryData.cs` **Type:** ScriptableObject **Create via:** Right-click → Memory → New Memory **Assets in:** `Assets/_Game/ScriptableObjects/Memories/`

Fields:

```csharp
```

```
string memoryTitle
string memoryDescription
MemoryCategory category
float emotionalWeight              // 0-1, influence strength

// Phase 5 — ACTIVE
Color worldTintContribution        // drives post processing colour blend
                                   // set distinct colours per memory

// Phase 7 — HOOK (not yet wired)
AudioClip ambientLayer             // per-memory personal audio layer
                                   // leave empty for now

// Phase 6 — ACTIVE
TraitType[] reinforcedTraits       // traits nudged up when memory kept
TraitType[] erodedTraits           // traits nudged down when memory kept

// Presentation
Sprite memoryIcon
Color memoryColour                 // UI card accent + MomentTrigger glow colour
```

Enums (defined in MemoryData.cs):

```
csharp

MemoryCategory: Nature, Solitude, Connection, Risk, Creation, Loss, Wonder, Stillness
TraitType: Fearless, Fragile, Curious, Calm, Aware, Warm, Agile, Melancholic, Resilient, Open
```

MemoryInstance.cs

Path: `Assets/_Game/Scripts/Memories/MemoryInstance.cs` **Type:** Plain C# class (not MonoBehaviour)

```
csharp

MemoryData data
float timeAcquired
bool hasBeenReflectedOn    // true when player clicks card in UI — used by Phase 8
float vividness            // 0-1, ticked down hourly by TimeSystem
```

Convenience properties: `.Title` `.Description` `.Category` `.EmotionalWeight` `.MemoryColour` `.Icon`

MemorySystem.cs

Path: `Assets/_Game/Scripts/Memories/MemorySystem.cs` **Attached to:** `MemorySystem` (child of `--- SYSTEMS ---`) **Pattern:** Singleton — `MemorySystem.Instance` **Note:** Does NOT call DontDestroyOnLoad (inherits from root)

Config: `maxMemorySlots = 6` (Inspector, range 3–10)

Events:

```csharp
OnMemoryKept(MemoryInstance)
OnMemoryForgotten(MemoryInstance)
OnMemorySlotsFull(MemoryData, List<MemoryInstance>)
OnMemoriesChanged()
```

Public API:

```csharp
OfferMemory(MemoryData)
KeepMemory(MemoryData)
ForgetMemory(MemoryInstance)
ReplaceMemory(MemoryInstance toForget, MemoryData toKeep)
GetAllMemories()                // List<MemoryInstance>
HasFreeSlot()                   // bool
HasMemoryOfCategory(MemoryCategory) // bool
GetTotalEmotionalWeight()       // float — used by Phase 8
GetUsedSlots()                  // int
GetSlotCount()                  // int
AlreadyHolding(MemoryData)      // bool
GetGameTime()                   // float (legacy, replaced by TimeSystem)
NotifyMemoriesChanged()         // fires OnMemoriesChanged — call instead of invoking event directly
```

MomentTrigger.cs

Path: `Assets/_Game/Scripts/World/MomentTrigger.cs` **Base Prefab:** `Assets/_Game/Prefabs/Memories/MEM_MomentTrigger_Base` **Requires:** SphereCollider (auto-added), Player tagged `Player`

Inspector:

```csharp
```

```
MemoryData memoryData
float triggerRadius = 3f
string promptText
float lingerTime = 0f          // 0 = press E, >0 = auto-trigger after N seconds
bool consumeOnUse = true
bool visibleInWorld = true     // enables runtime-created Point Light
float glowRadius = 2.5f
float glowIntensity = 0.5f
float pulseSpeed = 1.2f
float pulseAmount = 0.15f
```

Behaviour:

- Creates Point Light child at runtime using `memoryData.memoryColour`
- Light pulses on sine wave breathing cycle
- Brightens and expands when player enters zone
- Fades out over 1.2s coroutine when consumed
- Scene gizmo draws coloured trigger sphere
- Random phase offset on pulse so multiple triggers don't sync

Suggested trigger placements for test scene:

Object Name	Memory	Linger	Prompt
Moment_WaterEdge	MEM_SatByWater	3s	Sit for a while
Moment_HighPoint	MEM_StoodOnHighGround	0	Look out
Moment_Rain	MEM_WalkedIntoRain	0	Step into it
Moment_Light	MEM_WatchedSunMove	4s	Watch the light
Moment_Hidden	MEM_FoundHiddenPlace	0	You found it

AudioManager.cs

Path: `Assets/_Game/Scripts/Core/AudioManager.cs` **Attached to:** `AudioManager` (child of `--- SYSTEMS ---`)
Pattern: Singleton — `AudioManager.Instance`

Architecture: One AudioSource per layer, all on this GameObject. Layers have target volumes. Update() smoothly lerps toward them using `Mathf.MoveTowards`.

Inspector:

```csharp
AudioClip baseAmbientClip          // always-on quiet background
float baseAmbientVolume = 0.15f
AudioLayerConfig[] categoryLayers  // one per MemoryCategory (category + clip)
float fadeDuration = 3f
float maxLayerVolume = 0.4f
AudioClip momentStingClip
float momentStingVolume = 0.6f
```

Key implementation note: Moment sting uses `momentStingSource.time = 0.3f` before Play() to skip dead air at clip start. Adjust value per clip.

On memory kept: fades in matching category layer + plays sting **On memory forgotten:** fades out category layer if no memories of that category remain

Note: `MemoryData.ambientLayer` is a Phase 7 hook — NOT used by AudioManager. AudioManager operates at category level only.

Public API:

```csharp
TriggerMomentSting()
BoostLayer(MemoryCategory, float boostAmount, float duration)
```

EmotionalResponseSystem.cs

Path: `Assets/_Game/Scripts/Core/EmotionalResponseSystem.cs` **Attached to:** `EmotionalResponseSystem` (child of `--- SYSTEMS ---`) **Pattern:** Singleton — `EmotionalResponseSystem.Instance`

Dependencies:

- `GlobalPostProcessVolume` assigned in Inspector
- URP post processing installed
- PlayerCamera has Post Processing enabled
- All PP profile value checkboxes ticked blue

How it works:

1. `OnMemoryKept` → `RecalculateWorldState()` + `TriggerMomentResponse()`
2. `RecalculateWorldState()` → blends target PP values from all held memories
3. `Update()` → lerps actual PP values toward targets
4. `TriggerMomentResponse()` → bloom spike coroutine + time dilation + recovery

Colour tint system: Blends all `worldTintContribution` colours weighted by `emotionalWeight × vividness`

Time of day response: Subscribes to `TimeSystem.OnTimeOfDayUpdated`

- Golden hour (6–8am, 5–7pm): warm amber tint push, bloom increase
- Night (before 6am, after 8pm): cool blue tint push, saturation reduction

Season response: Subscribes to `TimeSystem.OnSeasonChanged`

- Spring: +8 sat, +0.2 bloom / Summer: +15 sat, +0.3 bloom
- Autumn: -5 sat / Winter: -15 sat

Identity response: Subscribes to `IdentitySystem.OnTraitThresholdCrossed`

- Triggers world surge when player identity shifts significantly

Inspector values (recommended):

```
Base Saturation: -30      Peak Saturation: 60
Base Bloom: 0.2           Peak Bloom: 2.5
Base Vignette: 0.2        Peak Vignette: 0.5
Moment Bloom Spike: 4     Moment Time Dilation: 0.75
Transition Speed: 3
```

Public API:

```csharp
PushEmotionalState(float satBoost, float bloomBoost, float duration)
```

IdentitySystem.cs

Path: `Assets/_Game/Scripts/Identity/IdentitySystem.cs` **Attached to:** `IdentitySystem` (child of `--- SYSTEMS ---`)
Pattern: Singleton — `IdentitySystem.Instance`

Architecture: Maintains `Dictionary<TraitType, float>` (all start at 0.5 neutral). Subscribes to MemorySystem events. Applies memory's reinforced/eroded traits weighted by emotional weight. Traits drift slowly back toward 0.5 when not reinforced (configurable `neutralDriftRate`).

Events:

```csharp
OnTraitChanged(TraitType, float newValue)          // any meaningful shift
OnTraitThresholdCrossed(TraitType, float newValue)  // crosses 0.3, 0.6, or 0.9
```

Public API:

```csharp
GetTraitValue(TraitType)            // float 0-1
HasTrait(TraitType, float threshold = 0.6f)  // bool
GetTraitStrength(TraitType)         // float 0-0.5 (above neutral)
GetDominantTraits(float threshold)  // List<TraitType>
GetFullProfile()                    // Dictionary<TraitType, float> — used by Phase 8
```

Trait effects on player:

Trait	Effect
Fearless	Increased jump height
Fragile	Reduced speed, heavier gravity, narrower FOV
Curious	Wider FOV
Calm	Reduced camera bob
Agile	Increased walk speed

Recommended trait assignments for current memory assets:

Memory	Reinforced	Eroded
MEM_FoundHiddenPlace	Curious, Aware, Open	Warm
MEM_SatByWater	Calm, Aware	Fragile
MEM_StoodOnHighGround	Fearless, Curious, Melancholic	Fragile
MEM_WentToTheEdge	Fearless, Resilient	Fragile, Calm

TimeSystem.cs

Path: `Assets/_Game/Scripts/Time/TimeSystem.cs` **Attached to:** `TimeSystem` (child of `--- SYSTEMS ---`) **Pattern:** Singleton — `TimeSystem.Instance`

Config:

```csharp
```

```
float timeScale = 60f              // game seconds per real second
float startingHour = 8f            // time of day at game start
float vividnessDecayPerHour = 0.02f // how fast memories fade
float minimumVividness = 0.15f     // memories never fully disappear
int daysPerSeason = 7
```

For rapid testing: Set `timeScale = 3600` and `vividnessDecayPerHour = 0.3` to see vividness decay in seconds. Reset after testing.

Ticks vividness on every `MemoryInstance` each game hour, then calls `MemorySystem.NotifyMemoriesChanged()`.

Events:

```csharp
OnHourChanged(float hour)
OnDayChanged(int day)
OnSeasonChanged(Season season)
OnTimeOfDayUpdated(float dayProgress)   // every frame, 0-1
```

Public API:

```csharp
GetFormattedTime()          // "Day 3, 14:30"
IsNight()                   // bool — before 6am or after 8pm
IsGoldenHour()              // bool — 6-8am or 5-7pm
GetNormalisedTimeOfDay()    // float 0-1
CurrentHour                 // float
CurrentDay                  // int
CurrentSeason               // Season enum
```

WorldEchoSystem.cs

Path: `Assets/_Game/Scripts/World/WorldEchoSystem.cs` **Attached to:** `WorldEchoSystem` (child of `--- SYSTEMS ---`) **Pattern:** Singleton — `WorldEchoSystem.Instance`

Architecture: Maintains a list of `WorldEcho` objects. Registers echoes when memories are formed or when player lingers. Checks player proximity to echoes each frame. Pushes `EmotionalResponseSystem` when player is near a strong echo.

Echo types:

```csharp
```

```
MemoryFormed    // registered when OnMemoryKept fires
Lingered        // registered when player stands still for lingerThreshold seconds
Significant     // reserved for Phase 8 / external registration
```

Config:

```
csharp

float echoFeelRadius = 5f
float echoDecayPerHour = 0.1f
float minimumEchoStrength = 0.05f
float lingerThreshold = 8f          // seconds standing still to register linger
float lingerMovementTolerance = 1.5f
float echoAtmosphereStrength = 8f
```

Echoes decay each game hour (via `TimeSystem.OnHourChanged`) and are removed when below minimum strength.

Scene Gizmos: Draws coloured spheres at all echo positions in Scene view (opacity reflects strength).

Public API:

```
csharp

IsNearEcho()                    // bool
GetCurrentEchoStrength()        // float
GetAllEchoes()                  // List<WorldEcho> — used by Phase 8
RegisterSignificantEcho(Vector3, Color, string)
```

EndingSystem.cs

Path: `Assets/_Game/Scripts/Core/EndingSystem.cs` **Attached to:** `EndingSystem` (child of `--- SYSTEMS ---`)
Pattern: Singleton — `EndingSystem.Instance`

Critical architecture note: `EndingSystem` lives in `_Persistent` but needs `EndingUI` and `MasterFade` from the location scene. These are found at runtime via `FindFirstObjectByType<EndingUI>()` and `GameObject.Find("MasterFade")` — they cannot be wired in the Inspector as cross-scene references are not supported.

`MasterFade` object must be named exactly `MasterFade` (case sensitive) in every location scene.

Sequence:

1. Set `GameState.Ending` (freezes player)
2. Unlock cursor
3. Generate passages via `EndingNarrator.GenerateEnding()`

4. Fade to black (3s) + fade audio out simultaneously
5. Wait 2s in darkness
6. Activate `EndingScreen`, call `EndingUI.ShowPassages()`
7. Final passage holds indefinitely

Public API:

```
csharp

EndingSystem.Instance.TriggerEnding()
```

EndingNarrator.cs

Path: `Assets/_Game/Scripts/Core/EndingNarrator.cs` **Type:** Static class (no MonoBehaviour, no singleton)

Reads from all systems to generate personalised text:

- `MemorySystem.GetAllMemories()` — what was kept, sorted by vividness
- `MemorySystem.GetTotalEmotionalWeight()` — overall life weight
- `IdentitySystem.GetFullProfile()` — complete trait values
- `IdentitySystem.GetDominantTraits(0.65f)` — who the player became
- `WorldEchoSystem.GetAllEchoes()` — linger and memory echo count
- `TimeSystem.GetFormattedTime()` — when the game ended

Generates 5 passages:

1. **Opening** — tone set by memory count
2. **Memories** — list of kept memories with vividness qualifiers
3. **Identity** — dominant traits described in plain language, contradiction checks
4. **World** — echo count, linger behaviour
5. **Closing** — personalised final line based on trait + category combinations

Vividness qualifiers:

- > 0.85: no qualifier (fresh)
- > 0.6: "already fading a little"
- > 0.35: "growing distant now"
- ≤ 0.35: "barely more than a feeling"

Trait descriptions (used in Identity passage):

Trait	Description
Fearless	Someone who went to the edges
Fragile	Someone who felt things deeply
Curious	Someone who kept looking
Calm	Someone who knew how to be still
Aware	Someone who paid attention
Warm	Someone who turned toward others
Agile	Someone who moved through the world easily
Melancholic	Someone who understood that things end
Resilient	Someone who kept going
Open	Someone who stayed open to what came

EndingTrigger.cs

Path: `Assets/_Game/Scripts/World/EndingTrigger.cs` **Attached to:** World object (e.g. `Ending_Leave`)

Inspector:

```csharp
bool requireInteract = true        // E to trigger vs walk-in
string promptText = "Leave this place"
Color triggerColour                // warm golden glow
float glowIntensity = 0.4f
```

Testing shortcut: Right-click `EndingTrigger` component header in Inspector → **Trigger Ending Now** — triggers ending without walking to the object.

UI Scripts

UIManager.cs

Path: `Assets/_Game/Scripts/UI/UIManager.cs` **Attached to:** `UICanvas` root in each location scene **Pattern:** Singleton — `UIManager.Instance` **Manages:** cursor lock state, Tab toggle, memory system event routing

```csharp
ShowMomentPrompt(string text)
HideMomentPrompt()
CloseReflectScreen()
```

MomentPromptUI.cs

Fades prompt text in/out via CanvasGroup alpha. Clears text after fade to prevent ghost text.

MemorySlotHUD.cs

Spawns dots per slot. Subscribes to events inside `InitialBuild()` called via `Invoke(0.15f)` — delay prevents singleton race condition on startup. Dots lerp between filled/empty colour.

MemoryReflectUI.cs

Two modes: Normal (Tab, optional forget) and Replace (auto when slots full, forced forget). CanvasGroup fade in/out.

MemoryCardUI.cs

Individual card. Displays title, category, accent colour. Applies vividness to text and accent opacity:

```csharp
Initialise(MemoryInstance, Action<MemoryInstance>)
SetSelected(bool)
BoundMemory    // public property
```

EndingUI.cs

Path: `Assets/_Game/Scripts/UI/EndingUI.cs` **Attached to:** `EndingScreen` object (child of Screens, child of UICanvas) **Active:** false by default — activated by EndingSystem

Displays passages one by one. Each fades in, holds, fades out. Final passage never fades — player sits with it indefinitely.

Inspector references:

```csharp
CanvasGroup canvasGroup
TextMeshProUGUI passageText
TextMeshProUGUI passageTypeLabel    // subtle label above text
Image backgroundImage               // tinted per passage type
```

Background colours per passage type:

- Opening: near black blue-tinted

- Memories: near black green-tinted
- Identity: near black red-tinted
- World: near black deep blue
- Closing: near pure black

PART 7 — UI CANVAS STRUCTURE

Full Hierarchy

```
UICanvas                        ← UIManager.cs, Canvas Scaler 1920x1080 Match 0.5
├── MasterFade                  ← Image (black), CanvasGroup alpha 0, ALWAYS ACTIVE
│                                 Must be named exactly "MasterFade"
├── HUD
│   ├── MemorySlotHUD           ← MemorySlotHUD.cs, anchored bottom-right
│   │   └── SlotContainer       ← Horizontal Layout Group
│   └── MomentPrompt            ← MomentPromptUI.cs, CanvasGroup, bottom-centre
│       └── PromptText          ← TMP, italic, cream
└── Screens
    ├── MemoryReflect           ← MemoryReflectUI.cs, CanvasGroup alpha 0, inactive
    │   ├── Backdrop            ← Image full screen, black 70%, Raycast ON
    │   └── Panel               ← 860x520 centred, Raycast OFF
    │       ├── Header          ← height 56
    │       │   └── CloseButton ← TMP "×"
    │       ├── ReplaceModeNotice ← amber, hidden by default
    │       ├── MemoryCardContainer ← Scroll View, left 52%
    │       │   └── Viewport    ← Rect Mask 2D (NOT Mask+Image)
    │       │       └── Content ← Vertical Layout Group + Content Size Fitter
    │       ├── DetailPanel     ← right 48%, hidden by default
    │       │   ├── DetailCategory
    │       │   ├── DetailTitle
    │       │   ├── DetailDescription
    │       │   └── ForgetButton
    │       └── SlotCountText
    └── EndingScreen            ← EndingUI.cs, CanvasGroup, inactive by default
        ├── Background          ← Image, full stretch, near black
        ├── PassageTypeLabel    ← TMP, top-centre, small caps, faint
        └── PassageText         ← TMP, centre, size 20, italic, cream, wrapping ON
```

Critical UI Notes

Viewport masking: Use **Rect Mask 2D** on Viewport — NOT Mask + Image. The Mask + Image approach caused a cream-coloured overlay blocking card visibility. Rect Mask 2D clips cleanly with no Image required.

Raycast Target rules:

- MasterFade: OFF (visual only)
- Backdrop: ON (blocks world clicks during reflect screen)
- Panel: OFF
- Cards: ON (must receive clicks)

Hierarchy render order: Objects render bottom-to-top in Unity hierarchy. Backdrop must be listed ABOVE Panel so Panel renders on top of Backdrop.

MasterFade position: Must be a direct child of UICanvas and listed first in hierarchy so it can render over everything including the ending screen.

PART 8 — MEMORY ASSETS

All in `Assets/_Game/ScriptableObjects/Memories/`

Asset	Title	Category	Weight	Reinforced	Eroded	Tint
MEM_SatByWater	Sat by the water alone	Stillness	0.6	Calm, Aware	Fragile	Soft cool blue
MEM_StoodOnHighGround	Stood somewhere high	Wonder	0.7	Fearless, Curious, Melancholic	Fragile	Pale gold
MEM_WalkedIntoRain	Walked into the rain	Risk	0.5	Fearless, Resilient	Fragile	Deep slate
MEM_WatchedSunMove	Watched the light change	Stillness	0.4	Calm, Melancholic	—	Warm amber
MEM_FoundHiddenPlace	Found somewhere nobody knew	Solitude	0.8	Curious, Aware, Open	Warm	Forest green
MEM_WentToTheEdge	Went to the edge	Risk	0.5	Fearless, Resilient	Fragile, Calm	Deep red

worldTintContribution: Use obviously distinct saturated colours during development to confirm the blending system is working. Dial to subtle muted tones for final release.

ambientLayer: Leave empty — Phase 7 hook, not yet wired.

PART 9 — INPUT SYSTEM

Asset: `Assets/_Game/Scripts/Core/PlayerInputActions`

Action	Type	Binding	Used By
Move	Value/Vector2	WASD + Arrows	PlayerController, CameraController
Look	Value/Vector2	Mouse Delta	PlayerController
Jump	Button	Space	PlayerController (VelocityChange mode)
Interact	Button	E	MomentTrigger, EndingTrigger
Reflect	Button	Tab	UIManager

PART 10 — BUILD PHASES ROADMAP

Phase	Focus	Status
1	Project setup, folder structure, scenes, GameManager, Git	✅ Complete
2	First-person player, weighted movement, camera bob	✅ Complete
3	Memory system — data layer, moment triggers	✅ Complete
4	Memory UI — slots, prompts, reflection screen, replace mode	✅ Complete
5	Emotional Response System — post processing, layered audio	✅ Complete
6	Identity & Trait System — traits change movement/perception	✅ Complete
7	Time System + World Echo — vividness decay, echo points	✅ Complete
8	Ending & Reflection Sequence	✅ Complete (known bug — see Part 12)
Polish	Art, music, sound design, more memories, build	⬜ Next

PART 11 — PRE-WIRED DEPENDENCIES & FUTURE HOOKS

Hook 1 — ambientLayer on MemoryData

Where: `MemoryData.ambientLayer` **Status:** Field exists, not wired **Future:** Per-memory personal audio layer. When a specific memory is kept, its unique clip fades into the mix. More granular than AudioManager's category-level approach. Wire in polish phase by extending AudioManager to also check individual memory clips on `OnMemoryKept`.

Hook 2 — GetTotalEmotionalWeight()

Where: `MemorySystem.GetTotalEmotionalWeight()` **Status:** Built, used by EndingNarrator **Extends to:** Could drive ending music selection — heavier weight → more intense finale track.

Hook 3 — RegisterSignificantEcho()

Where: `WorldEchoSystem.RegisterSignificantEcho(Vector3, Color, string)` **Status:** Built, never called **Future:** Scripted story moments can register permanent or very strong echoes at specific locations. A place where something important happened always feels different.

Hook 4 — TriggerEmotionalMoment() on CameraController

Where: `CameraController.TriggerEmotionalMoment(float intensity)` **Status:** Built, never called from outside **Future:** EndingSystem or world scripted events can trigger a subtle camera push-in for cinematic emphasis.

Hook 5 — GameState.Transitioning

Where: `GameManager.GameState.Transitioning` **Status:** Enum value exists, never set **Future:** Used when moving between location scenes. PlayerController already checks `IsPlaying()` so it will automatically stop during transitions.

Hook 6 — Season enum

Where: `TimeSystem`, `Season` enum **Status:** Seasons fire events and shift post processing **Future:** Season could also change which memories are available — some moment triggers only appear in certain seasons.

Polish Phase Priorities

Environment art — Replace grey test space with real geometry. The emotional response system will look completely different against actual environmental lighting.

Composed ambient score — A layered music system on top of the existing AudioManager category layers. The score should breathe with the player's emotional state.

More memory assets — The current 6 memories are a skeleton. A full game needs 30–50 distinct memories across all categories, each with their own prompt text, description, tint, and trait assignments.

Sound design — Footsteps, interaction sounds, memory formation sounds beyond the sting.

Boot sequence — Replace hardcoded `GameState.Playing` with a proper boot: title screen → fade in → begin.

Save system — Deliberately consider whether impermanence is a feature. A game about memory that doesn't save might be the right choice. If saving, MemorySystem and IdentitySystem state need serialization.

PART 12 — KNOWN ISSUES LOG

Issue	Severity	Status	Fix
EndingUI not found — ending screen text never appears	**HIGH — OPEN**	Open	See below
DontDestroyOnLoad warning (fixed with SetParent null)	Medium	✅ Resolved	`transform.SetParent(null)` before DontDestroyOnLoad in Awake
Memory cards not visible	High	✅ Resolved	Replaced Mask+Image on Viewport with Rect Mask 2D
Close button not clickable	High	✅ Resolved	Panel Image Raycast Target set to OFF
Panel proportions wrong	Medium	✅ Resolved	Panel anchor set to centre point (0.5,0.5) both min and max
Post processing not rendering	High	✅ Resolved	PlayerCamera needed Post Processing enabled
PP values calculating but not changing	Medium	✅ Resolved	Individual value checkboxes in PP profile must be ticked blue
Tab key not opening reflect screen	Medium	✅ Resolved	Input action binding lost — rebind Tab if recurs
HUD dots not updating	Medium	✅ Resolved	Subscriptions moved inside Invoke-delayed InitialBuild()
Moment prompt persists after memory taken	Low	✅ Resolved	HideMomentPrompt() called in ExperienceMoment()
Audio Listener warning	Low	✅ Resolved	Audio Listener added to PlayerCamera
Moment sting plays from wrong point	Low	✅ Resolved	momentStingSource.time offset in TriggerMomentSting()
Jump not working	Medium	✅ Resolved	Changed ForceMode to VelocityChange, Jump Force to 6

Issue	Severity	Status	Fix
anyChanged compiler warning in IdentitySystem	Low	✅ Resolved	Variable removed from Update()
OnMemoriesChanged invoked from outside MemorySystem	Low	✅ Resolved	Added NotifyMemoriesChanged() public method

---

PRIORITY BUG — EndingUI Not Found in Scene

Symptom:

```
[EndingSystem] EndingUI not found in scene
```

Fade to black works. Audio fades. But no ending text appears. Sequence completes immediately with nothing shown.

Root Cause: `EndingSystem.TriggerEnding()` calls `FindFirstObjectByType<EndingUI>()` to locate the EndingScreen object in the current scene. This call is failing, meaning Unity cannot find an active `EndingUI` component in the scene at the moment `TriggerEnding()` is called.

Most likely causes:

1. **EndingScreen object is inactive** — `FindFirstObjectByType` by default only finds active objects. If `EndingScreen` is set inactive in the Hierarchy (unchecked), it won't be found even if the script is attached.
2. **EndingUI.cs script not attached** — The `EndingScreen` object exists but `EndingUI.cs` is not attached to it as a component.
3. **EndingScreen is a child of an inactive parent** — If `Screens` or `UICanvas` is inactive, all children are effectively inactive.

Suggested fixes (try in order):

Fix A — Change FindFirstObjectByType to include inactive objects:

Open `EndingSystem.cs`, find the line in `TriggerEnding()`:

```
csharp

endingUI = Object.FindFirstObjectByType<EndingUI>();
```

Replace with:

```
csharp

endingUI = Object.FindFirstObjectByType<EndingUI>(FindObjectsInactive.Include);
```

This tells Unity to search inactive objects too. Since `EndingSystem` itself activates `EndingScreen` later in the sequence, it needs to find it while it's still inactive.

Fix B — Find by name instead:

```csharp
GameObject endingScreenObj = GameObject.Find("EndingScreen");
if (endingScreenObj == null)
    endingScreenObj = FindInactiveByName("EndingScreen");
endingUI = endingScreenObj?.GetComponent<EndingUI>();
```

With helper:

```csharp
private GameObject FindInactiveByName(string name)
{
    Transform[] all = Resources.FindObjectsOfTypeAll<Transform>();
    foreach (var t in all)
    {
        if (t.name == name && t.hideFlags == HideFlags.None)
            return t.gameObject;
    }
    return null;
}
```

Fix C (cleanest long term) — Register EndingUI with EndingSystem on Start:

In `EndingUI.cs`, add:

```csharp
private void Start()
{
    // Self-register so EndingSystem can always find us
    // regardless of active state
    if (EndingSystem.Instance != null)
        EndingSystem.Instance.RegisterEndingUI(this);
}
```

In `EndingSystem.cs`, add:

```csharp
```

```
public void RegisterEndingUI(EndingUI ui)
{
    endingUI = ui;
    Debug.Log("[EndingSystem] EndingUI registered");
}
```

And remove the `FindFirstObjectByType` call from `TriggerEnding()`.

Recommended: Fix A is the quickest single-line change. Fix C is the most robust long-term pattern consistent with how other systems in this project communicate.

Document version 5 — Final Master. Replaces all previous versions. All 8 phases built. One known open bug in ending sequence. Polish phase ready to begin once resolved.