Capítulo 5

Implementación del plugin

Este capítulo describe el proceso de implementación del plugin desarrollado para gestionar la ambientación emocional en escenarios de videojuegos. Se detallan el entorno técnico, la estructura del sistema, la lógica funcional, las decisiones de implementación, y los métodos de validación utilizados. Este proceso responde a la necesidad de integrar técnicas de computación emocional en el diseño de experiencias inmersivas [4] [5].

5.1. Entorno de desarrollo

El desarrollo del plugin se realizó en el motor gráfico Unity, versión 2022.3.4f1, utilizando el lenguaje de programación C#. Se utilizó Visual Studio como entorno de codificación. Durante el inicio del desarrollo, se presentó un error de compatibilidad entre la versión de Visual Studio 2022 y Unity 2021, el cual fue resuelto al actualizar el proyecto a la versión correcta de Unity.

El desarrollo y pruebas se llevaron a cabo en un equipo con las siguientes especificaciones:

- **Procesador:** AMD Ryzen 5 4600G, 3,70 GHz
- **Memoria:** 16GB
- **SDD:** 1TB
- **Sistema Operativo:** Windows 10
- **Software:** Unity 2022.3.4f1, Visual Studio 2022

5.2. Estructura del sistema

El plugin se estructuró en torno a un único controlador principal denominado LightingController, el cual concentra la lógica del sistema. Este enfoque centralizado simplifica el mantenimiento y escalabilidad del código. El sistema se compone de los siguientes elementos funcionales:

- **Controlador principal (LightingController.cs):** gestiona la entrada del usuario (sliders y clics en la rueda emocional), la lógica de mezcla de colores según las emociones seleccionadas, la persistencia de datos y la interacción con los objetos de iluminación.

- **Interfaz de usuario (Canvas):** implementada como un panel dentro de Unity, contiene los controles de entrada (sliders para cada emoción, botón de acción, campo de texto para nombre del perfil), así como la rueda emocional personalizada (plutchikWheel) y el marcador visual (emotionMarker).
- **Objetos de iluminación (EmotionalLight):** cualquier objeto de la escena que implemente el componente EmotionalLight puede recibir y reflejar el color emocional calculado.
- **Archivos de configuración (settings.json, emotion_profiles.json):** utilizados para guardar y recuperar configuraciones específicas de ambientación.

Este diseño se enfoca en la modularidad y claridad del flujo emocional-visual, lo cual responde a buenas prácticas de usabilidad en videojuegos [10], orientado a componentes y funciones específicas, permite que el plugin sea integrable en diferentes proyectos de Unity sin necesidad de modificaciones significativas. Además, se mantiene la separación entre lógica de negocio (cálculo emocional), vista (interfaz gráfica) y almacenamiento (persistencia JSON), lo cual se alinea con principios de arquitectura limpia.

A continuación, se presenta un resumen de las principales clases y métodos utilizados en la implementación del sistema:

Clase / Método	**Descripción funcional**
`LightingController`	Clase principal del sistema. Controla los sliders emocionales, la lógica de mezcla de colores, la interfaz gráfica, el guardado y carga de perfiles, y la aplicación de los efectos de iluminación en la escena.
`ApplyEmotionalLighting()`	Calcula el color emocional final combinando las emociones activas según su intensidad. Aplica este color a todos los objetos tipo `EmotionalLight` en la escena.
`SaveToJson()`	Guarda la configuración emocional actual (valores de sliders, color de luz, posición del marcador) en un archivo `settings.json`.
`LoadFromJson()`	Carga automáticamente la última configuración guardada desde `settings.json` al iniciar la escena.
`SaveProfile()`	Permite guardar un nuevo perfil emocional personalizado, con nombre asignado por el usuario, en el archivo `emotion_profiles.json`.
`ActivateProfile(nombre)`	Carga un perfil previamente guardado y aplica todos sus valores (emociones, color, marcador).
`SetEmotion(emocion)`	Establece una emoción primaria o secundaria y ajusta automáticamente los sliders correspondientes. También actualiza el marcador en la rueda emocional.
`UpdateEmotionMarker(color)`	Determina la emoción más cercana al color actual y reposiciona el marcador sobre la rueda emocional.

Tabla 5.1: Descripción de clases y métodos principales del plugin

5.3. Lógica de transformación emocional

El sistema permite que el usuario ajuste la intensidad de ocho emociones primarias: alegría, confianza, miedo, sorpresa, tristeza, asco, ira y anticipación, siguiendo el modelo propuesto por Plutchik. Cada emoción está asociada a un color específico en formato RGB normalizado. Estas asociaciones se definen en un diccionario (`Dictionary<string, Color>`) que actúa como base para los cálculos de mezcla.

Cuando el usuario modifica los sliders, el sistema calcula el color final como una combinación ponderada de los colores emocionales según la intensidad ingresada. Este cálculo considera solo aquellas emociones con valor superior a cero, lo que permite representar combinaciones múltiples o emociones dominantes.

El algoritmo general es:

Algorithm 1 Aplicación de iluminación emocional

Inicializar `colorFinal` ← Color.black
Inicializar `totalPeso` ← 0
for all emociones en `emotionColors` **do**
  Obtener `intensidad` desde el slider correspondiente
  if `intensidad` > 0 **then**
    `colorFinal` ← `colorFinal` + (`colorEmocion` × `intensidad`)
    `totalPeso` ← `totalPeso` + `intensidad`
  end if
end for
if `totalPeso` > 0 **then**
  `colorFinal` ← `colorFinal` / `totalPeso`
end if
for all luces en escena con componente `EmotionalLight` **do**
  Asignar `colorFinal` a la luz
end for
Actualizar el marcador de emoción con `colorFinal`
Guardar estado en archivo `settings.json`

Este enfoque permite generar una iluminación dinámica coherente con los estados emocionales seleccionados, potenciando la inmersión y la narrativa [7] [24].

5.4. Gestión de perfiles emocionales

Uno de los aspectos más útiles del plugin es la posibilidad de guardar configuraciones personalizadas en perfiles emocionales. Esta función permite al desarrollador registrar combinaciones específicas de emociones, colores e intensidades con un nombre definido por el usuario a través de un campo de texto y presionar un botón para almacenarla. Internamente, los perfiles son objetos serializados a formato JSON, y se guardan en una ruta persistente (`Application.persistentDataPath`).

Cada perfil (EmotionProfile) incluye:

- El nombre del perfil.
- Los valores de los ocho sliders emocionales.

- Los valores RGB del color final calculado.
- La posición del marcador en la rueda emocional.

Un ejemplo de cómo se representa un perfil emocional en formato JSON es el siguiente:

```
{
  "name": "misterio_sutil",
  "alegria": 0.0,
  "miedo": 0.5,
  "sorpresa": 0.4,
  "tristeza": 0.2,
  "ira": 0.1,
  "asco": 0.0,
  "desprecio": 0.0,
  "confianza": 0.3,
  "lightR": 0.65,
  "lightG": 0.42,
  "lightB": 0.78,
  "markerX": 73.2,
  "markerY": -149.5
}
```

Los perfiles emocionales definidos por el usuario se almacenan en un archivo denominado emotion_profiles.json, el cual se guarda automáticamente en la ruta persistente del sistema proporcionada por Unity, accesible mediante la propiedad `(Application.persistentDataPath)`. Esta ubicación garantiza que los datos se conserven entre sesiones y sean accesibles independientemente del sistema operativo, lo que mejora significativamente la portabilidad del plugin entre distintos entornos de desarrollo o ejecución.

El uso del formato JSON responde a criterios de legibilidad, flexibilidad y compatibilidad. Este formato estructurado y ampliamente adoptado facilita tanto la manipulación automática de los datos por parte del sistema como la edición manual en caso de ser necesario, lo que representa una ventaja importante durante la fase de depuración o ajuste fino de los perfiles emocionales. Además, esta arquitectura de almacenamiento permite implementar funcionalidades adicionales como la carga dinámica de perfiles, su edición desde la interfaz del plugin, la eliminación de perfiles obsoletos o incluso su exportación e importación entre diferentes proyectos. De esta forma, se asegura que el sistema pueda adaptarse de manera eficiente a las necesidades tanto del desarrollador como del motor emocional del juego, fomentando la reutilización y la escalabilidad de las configuraciones emocionales definidas.

5.5. Activación de perfiles emocionales

Una de las funcionalidades más relevantes del plugin es su capacidad de integrarse directamente con la lógica interna del juego mediante la activación programada de perfiles emocionales. Esto permite que los desarrolladores modifiquen la ambientación emocional del entorno en tiempo real y en respuesta a eventos del juego, como la entrada del jugador a una zona específica, la finalización de una misión, o el desencadenamiento de una escena narrativa.

Esta característica convierte al plugin no solo en una herramienta de configuración estática, sino también en un sistema dinámico de ambientación emocional adaptable al gameplay, alineándose con los principios de diseño inmersivo y emocional [24] [5].

Ejemplo de uso en un evento del juego

A continuación se presenta un ejemplo práctico de cómo un desarrollador puede utilizar el plugin para cambiar la ambientación emocional cuando un personaje atraviesa un punto específico del escenario. Este tipo de eventos puede usarse para reforzar el tono emocional de una narrativa o para crear transiciones afectivas entre distintas áreas del juego.

```
using UnityEngine;

public class DoorTrigger : MonoBehaviour
{
    public LightingController lightingController;

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            lightingController.ActivateProfile("tristeza");
        }
    }
}
```

En este ejemplo, el script DoorTrigger se asigna a un objeto con colisionador en la escena (por ejemplo, una puerta o entrada). Cuando el jugador (identificado mediante la etiqueta "Player") entra en contacto con este objeto, se ejecuta el método ActivateProfile(nombrePerfil) del controlador de iluminación (lightingController), pasando como parámetro el nombre del perfil emocional previamente guardado (en este caso, "tristeza").

La funcionalidad ActivateProfile(nombrePerfil) se encarga de buscar en el archivo emotion_profiles.json el perfil con el nombre proporcionado, cargar sus valores y aplicarlos a la interfaz y a la escena. A continuación, se describe el flujo general del algoritmo:

Algorithm 2 Activación de perfil emocional

Require: `nombrePerfil`: cadena con el nombre del perfil a activar
1: Cargar el archivo `emotion_profiles.json`
2: Buscar en la lista el perfil con nombre igual a `nombrePerfil`
3: **if** perfil encontrado **then**
4: **for all** emociones primarias **do**
5: Asignar valor del perfil a cada slider correspondiente
6: **end for**
7: Establecer el color de iluminación desde los valores RGB del perfil
8: Posicionar el marcador emocional en la rueda (X, Y)
9: Aplicar color a objetos con componente `EmotionalLight`
10: **else**
11: Mostrar mensaje de error: "Perfil no encontrado"
12: **end if**

Este algoritmo refleja el funcionamiento interno del método ActivateProfile() implementado en el script LightingController. Asegura que la carga sea segura (verifica existencia del perfil), visualmente coherente (ajusta sliders y marcador), y funcional (aplica la iluminación directamente). Su simplicidad y modularidad permiten al desarrollador invocarlo en cualquier momento del juego sin necesidad de modificar la interfaz.

Integración con la narrativa y la experiencia emocional

Este tipo de integración permite que la ambientación emocional acompañe y refuerce el desarrollo narrativo. Por ejemplo:

- Al ingresar a una zona de ruinas o cementerio, puede activarse un perfil con tonalidades frías y azuladas para inducir tristeza o melancolía.
- Al completar un objetivo importante, puede activarse un perfil de "alegría" para elevar el ánimo del jugador.
- En situaciones de peligro, puede aplicarse automáticamente un perfil de "miedo" con colores oscuros e iluminación tenue.

Este modelo de interacción emocional-contextual convierte al plugin en un componente clave del diseño de experiencia (UX) dentro del videojuego, al permitir que el entorno responda emocionalmente al comportamiento del jugador o a la evolución del juego.

El método ActivateProfile(string nombrePerfil) puede ser invocado desde cualquier script del juego que tenga una referencia al LightingController, lo cual ofrece flexibilidad total al diseñador o programador para definir en qué momentos se desea cambiar la ambientación emocional.

En conjunto, esta funcionalidad responde a las necesidades contemporáneas del diseño emocional en videojuegos, donde el entorno debe adaptarse no solo a reglas lógicas, sino también a los estados afectivos deseados, promoviendo una experiencia más envolvente, coherente y significativa para el jugador [2] [7].

5.6. Interfaz gráfica y control emocional

La interfaz gráfica del plugin fue diseñada con un enfoque minimalista, funcional e intuitivo, considerando las necesidades tanto de diseñadores técnicos como de usuarios menos familiarizados con programación. Esta interfaz está integrada directamente dentro del editor de Unity como un Canvas, lo que permite su visualización en modo de ejecución y facilita la interacción en tiempo real con los elementos del escenario.

La estructura y disposición de la interfaz obedecen a principios de usabilidad, carga cognitiva reducida y retroalimentación visual inmediata, elementos esenciales en herramientas de apoyo creativo. La interfaz fue organizada en bloques funcionales que agrupan elementos relacionados por su propósito.

5.6.1. Elementos principales de la interfaz:

- **Sliders para emociones primarias:**
  Se presenta un conjunto de ocho controles deslizantes, uno por cada emoción primaria del modelo de Plutchik (alegría, confianza, miedo, sorpresa, tristeza, asco, ira y anticipación). Cada slider permite al usuario ajustar la intensidad de la emoción correspondiente en un rango de 0 a 1, permitiendo

así una personalización precisa del estado emocional deseado.

Estos sliders están dispuestos verticalmente, agrupados y etiquetados de forma clara. Su modificación en tiempo real activa inmediatamente la recalculación del color emocional, fomentando la experimentación y el diseño iterativo.

- **Campo de texto y botón "Guardar perfil":**
  Esta sección permite al usuario escribir un nombre personalizado para la configuración emocional actual y guardarla como un perfil utilizando el método SaveProfile(). El perfil incluye intensidades, color generado y posición del marcador, lo cual facilita su reutilización posterior. Esta funcionalidad fomenta la iteración creativa y la documentación de decisiones emocionales en el diseño de niveles o escenas.

- **Imagen interactiva de la rueda emocional de Plutchik:**
  Uno de los elementos más distintivos de la interfaz es la inclusión de una representación visual circular del modelo emocional de Plutchik. Esta rueda permite al usuario seleccionar emociones directamente mediante clics, sin necesidad de manipular los sliders manualmente. La acción de clic activa el método SetEmotion(), que ajusta los sliders y reposiciona el marcador según la emoción seleccionada.

  Esta forma de navegación visual convierte el proceso de diseño emocional en una experiencia más accesible e intuitiva, facilitando la comprensión conceptual de la relación entre emociones y su representación visual.

- **Marcador visual (emotionMarker):**
  El marcador actúa como un puntero dinámico que se posiciona automáticamente sobre la rueda emocional, señalando la emoción dominante resultante del cálculo. El sistema también puede actualizar este marcador de forma automática mediante el método UpdateEmotionMarker(), cuando se modifica una combinación emocional compleja, ayudando así a mantener una coherencia entre lo numérico y lo visual.

A continuación, se presenta la interfaz gráfica final del plugin desarrollada en Unity. Esta interfaz, basada en un diseño minimalista y funcional, integra controles intuitivos que permiten la gestión emocional mediante sliders, selección visual en la rueda emocional de Plutchik y creación de perfiles personalizados. La imagen muestra cómo se representa visualmente la emoción dominante mediante el marcador dinámico (emotionMarker), facilitando así una interacción directa y comprensible con el espacio emocional. Esta implementación refuerza la conexión entre el usuario y el modelo emocional, consolidando una experiencia coherente con los fundamentos del modelo circumplejo de Russell [25] y la teoría de la afectividad de Plutchik.

Figura 5.1: Interfaz gráfica final del plugin

5.6.2. Justificación conceptual del diseño visual

El uso de la rueda emocional como interfaz no solo aporta funcionalidad, sino que responde a principios psicológicos y teóricos sobre cómo se conceptualizan las emociones humanas. Esta representación está inspirada tanto en el modelo circumplejo de Russell [25] como en la estructura circular de Plutchik, que permiten mapear emociones en un espacio bidimensional según su intensidad y relación afectiva.

Incorporar este tipo de visualización no es únicamente una mejora estética, sino una herramienta cognitiva que reduce la carga mental del usuario al ofrecer un marco visual intuitivo [5] [1], el diseño emocional eficaz en videojuegos no depende exclusivamente del contenido narrativo, sino también de las herramientas que permiten moldear y modular dicho contenido con intención emocional.

5.6.3. Experiencia del usuario (UX)

La interfaz está diseñada para **proporcionar retroalimentación inmediata** cualquier cambio en los sliders o en la rueda genera efectos visibles en la escena, fortaleciendo el ciclo de acción-reacción que caracteriza una buena experiencia de usuario. Además, se prioriza la consistencia visual, el uso de colores coherentes con las emociones, y la legibilidad de todos los elementos interactivos.

Este diseño no solo permite al usuario configurar emociones, sino que también invita a explorar y experimentar, lo cual es esencial para el diseño de ambientaciones complejas en videojuegos que busquen modular la experiencia del jugador mediante cambios ambientales sutiles pero significativos [26] [23].

5.7. Manejo de emociones secundarias

Además de las emociones primarias, el plugin contempla un conjunto de emociones secundarias, las cuales se derivan de la combinación de dos emociones básicas, conforme al modelo propuesto por Plutchik (1980). Estas emociones complejas surgen de la interacción entre emociones primarias adyacentes en la rueda emocional, como lo son "amor" (alegría + confianza) o "remordimiento" (tristeza + asco). La inclusión de estas emociones tiene como propósito ampliar la expresividad del sistema y permitir una representación emocional más rica, coherente con el diseño narrativo contemporáneo en videojuegos [1]

A continuación, se presenta una tabla con las emociones secundarias actualmente implementadas y sus respectivas combinaciones:

Emoción secundaria	**Combinación de emociones primarias**
Amor	Alegría + Confianza
Sumisión	Confianza + Miedo
Susto	Miedo + Sorpresa
Decepción	Sorpresa + Tristeza
Remordimiento	Tristeza + Asco
Desprecio	Asco + Ira
Agresión	Ira + Anticipación
Optimismo	Anticipación + Alegría

Tabla 5.2: Emociones secundarias y sus combinaciones en el modelo de Plutchik

A nivel de implementación, las emociones secundarias se gestionan mediante un diccionario de tipo `(Dictionary<string, string[ ]>)` , donde la clave representa la emoción secundaria y el valor es un arreglo de cadenas que contiene las dos emociones primarias que la conforman. Por ejemplo:

```
{ "amor", new string[] { "alegría", "confianza" } }
```

Este enfoque permite mantener la simplicidad del sistema, evitando la necesidad de agregar más controles o sliders a la interfaz. Cuando el usuario selecciona una emoción secundaria, el sistema realiza automáticamente los siguientes pasos:

1. **Ajuste de sliders:** Se asigna un valor intermedio (generalmente 0.5) a cada una de las emociones primarias que componen la emoción secundaria. Esto simula una mezcla proporcional de ambas emociones y activa visualmente los sliders correspondientes en la interfaz.

2. **Cálculo de color resultante:** A través del método ApplyEmotionalLighting(), se calcula el color final como una combinación ponderada de los colores asociados a las emociones primarias activadas. Este color se aplica de inmediato a los objetos EmotionalLight en la escena, generando una ambientación coherente con la emoción secundaria seleccionada.

3. **Actualización del marcador emocional:** El sistema posiciona el marcador (emotionMarker) en una coordenada específica definida para cada emoción secundaria en el diccionario emotionRanges. Si la emoción secundaria no tiene una coordenada propia, se calcula el promedio de las coordenadas de sus emociones primarias para determinar su ubicación aproximada en la rueda.

4. **Reflejo visual inmediato:** La interfaz se actualiza en tiempo real, permitiendo al usuario observar cómo la combinación emocional afecta la ambientación del escenario. Esto apoya tanto la validación de la configuración como su ajuste fino posterior si se desea guardar como un nuevo perfil.

Este mecanismo no solo incrementa la capacidad expresiva del plugin, sino que también potencia su utilidad como herramienta de diseño emocional, especialmente en contextos donde se desea modular la atmósfera según eventos narrativos, perfiles de jugador o decisiones dentro del juego. La posibilidad de trabajar con emociones complejas responde a recomendaciones del diseño emocional avanzado en juegos digitales[3] [2].

Por último, es importante resaltar que este sistema permite ampliar fácilmente el conjunto de emociones secundarias simplemente añadiendo nuevas entradas al diccionario correspondiente, sin modificar la estructura base del código ni la interfaz. Esta escalabilidad garantiza que el plugin pueda adaptarse a distintas necesidades creativas o contextos de diseño en proyectos futuros.

5.8. Guardado automático del estado

Uno de los componentes clave del plugin es su capacidad para preservar el estado emocional actual del sistema mediante un mecanismo de guardado automático. Esta funcionalidad fue diseñada con el propósito de garantizar continuidad, consistencia y eficiencia durante el desarrollo y las pruebas de ambientaciones emocionales en escenarios interactivos.

Cada vez que el usuario aplica una nueva configuración emocional —ya sea ajustando sliders manualmente, seleccionando emociones en la rueda de Plutchik o cargando un perfil previamente guardado—, el sistema ejecuta el método SaveToJson(), que serializa todos los datos relevantes y los almacena automáticamente en un archivo JSON persistente llamado settings.json. Este archivo se guarda en el directorio proporcionado por Unity a través de `(Application.persistentDataPath)`, lo que asegura compatibilidad entre diferentes sistemas operativos y sesiones de ejecución.

El archivo settings.json incluye la siguiente información estructurada:

- **Intensidad de las emociones primarias:** los valores de cada uno de los sliders (alegría, confianza, miedo, sorpresa, tristeza, asco, ira y anticipación).
- **Color emocional aplicado:** los valores normalizados del color RGB final que se aplicó a los objetos EmotionalLight.
- **Posición del marcador emocional:** coordenadas X e Y del marcador visual dentro de la rueda emocional (emotionMarker), que representan gráficamente la emoción dominante o combinada.

Al iniciar una nueva sesión o volver a cargar una escena, el método LoadFromJson() es llamado automáticamente desde el evento Start(), lo cual permite restaurar de inmediato la ambientación emocional previamente establecida. Esto evita que el usuario deba repetir manualmente la configuración anterior y proporciona una experiencia de trabajo fluida, especialmente útil durante iteraciones de prueba o en proyectos de diseño narrativo emocional prolongado.

Desde el punto de vista del diseño de sistemas, este enfoque responde a principios de persistencia de estado y mejora la ergonomía del flujo de trabajo, características que son especialmente valiosas en herramientas creativas. Al garantizar que los cambios emocionales se mantengan entre sesiones, el plugin no solo respeta las decisiones del diseñador, sino que también favorece la coherencia emocional entre diferentes instancias del juego, incluso en fases tempranas del desarrollo [27].