

# **Pokémon: La Costera**

# ÍNDICE

|   |           |
|---|-----------|
| <b>Pokémon: La Sierpe de La Costera.....</b>                          | <b>4</b>  |
| <b>La trama: Un lío de playa y villanos de chiringuito.....</b>       | <b>4</b>  |
| <b>Lugares clave de La Costera.....</b>                               | <b>5</b>  |
| <b>Pokémon exclusivos de la región.....</b>                           | <b>5</b>  |
| <b>Clímax de la historia.....</b>                                     | <b>5</b>  |
| <b>Personajes.....</b>  | <b>6</b>  |
| El/La Protagonista – El Guardián del Agua.....                        | 6         |
| • Pokémon Inicial:.....   | 6         |
| <b>Andrés "El Fisco" – Líder del Team Almendrao'.....</b>             | <b>7</b>  |
| • Pokémon principales:.....   | 7         |
| • Frases típicas:.....  | 7         |
| <b>Asets.....</b>   | <b>8</b>  |
| <b>Tile sets.....</b>   | <b>9</b>  |
| Sprites y animaciones.....  | 9         |
| <b>Mecánicas de juego.....</b>  | <b>11</b> |
| 1. Sistema de Combate.....  | 12        |
| 1.1 Elementos Claves del Combate.....                                 | 12        |
| 2. Sistema de Clima Dinámico.....                                     | 12        |
| 2.1 Tipos de Clima y Efectos.....                                     | 13        |
| 3. Exploración del Mundo.....   | 13        |
| 3.1 Movilidad del Jugador.....  | 13        |
| 3.2 Interacciones Ambientales.....                                    | 13        |
| 4. Gimnasios y Liga Pokémon.....                                      | 14        |
| 4.1 Características del Sistema de Gimnasios.....                     | 14        |
| 4.2 Liga Pokémon.....   | 14        |
| 5. Team Almendrao' y Batalla Final.....                               | 14        |
| 5.1 Mecánica del Enfrentamiento Final.....                            | 14        |
| 6. Conclusión.....  | 15        |
| Tabla de tipos.....   | 16        |
| Escena: Encuentro con Andrés #1.....                                  | 18        |
| <b>Creación de mapas.....</b>   | <b>21</b> |
| Resumen ultra rápido.....   | 22        |
| Funcionamiento de los "Tiles" y Colisiones en el Editor de Mapas..... | 23        |
| Casillas y Capas.....   | 23        |
| Definición de Colisiones.....   | 23        |
| Resumen rápido.....   | 24        |
| <b>Base de Datos de Pokémon.....</b>                                  | <b>25</b> |
| Estadísticas base y crecimiento.....                                  | 26        |
| Datos de captura, felicidad y habilidades.....                        | 27        |

|  |           |
|--|-----------|
| Movimientos.....                         | 27        |
| Crianza y biología.....                  | 28        |
| Pokédex y combate.....                   | 28        |
| Apariencia en combate.....               | 28        |
| Evolución.....                           | 29        |
| <b>Notas adicionales.....</b>            | <b>29</b> |
| Resumen rápido.....                      | 30        |
| <b>Base de Datos de Movimientos.....</b> | <b>30</b> |
| <b>Animaciones de Movimientos.....</b>   | <b>33</b> |
| <b>Base de Datos HABILIDADES.....</b>    | <b>36</b> |
| <b>Minijuegos.....</b>                   | <b>38</b> |
| Tragaperras.....                         | 39        |
| Clases Principales.....                  | 40        |
| Lógica del Juego.....                    | 42        |
| Combinaciones Ganadoras y Pagos.....     | 42        |
| Gestión de Recursos y Animaciones.....   | 42        |
| Conclusión.....                          | 43        |
| Voltorb Flip.....                        | 44        |
| <b>Items.....</b>                        | <b>49</b> |
| Repelente.....                           | 50        |
| Pociones (en combate).....               | 53        |
| Pociones (Fuera Combate).....            | 56        |
| Despertar.....                           | 58        |
| Revivir y Revivir Máximo.....            | 59        |
| Ether.....                               | 61        |

## **Pokémon: La Sierpe de La Costera**

La historia transcurre en la región de La Costera, un paraíso inspirado en la costa mediterránea de España, con playas doradas, calas escondidas, marisquerías y chiringuitos de toda la vida. Pero no todo es sol y fiesta: una oscura amenaza se cierne sobre la región.

Un grupo de villanos conocidos como Team Almendrao' anda liándola parda por toda La Costera, buscando despertar a un antiguo Pokémon legendario que, según dicen, guarda un poder ancestral capaz de convertir el mar en un vergel cristalino... o en un hervidero de destrucción.

### **La trama: Un lío de playa y villanos de chiringuito**

El protagonista es un chaval o chavala de un pueblo costero sin nombre (porque en los Pokémon siempre vives en un sitio minúsculo con una casa, una tienda y un vecino en chanclas). Un día, tras recibir su primer Pokémon del Profesor Emilio, se entera de que el Team Almendrao' está tramando algo chungo en la Cueva del Levante, un sistema de grutas marinas legendarias.

Esta banda de maleantes, liderada por Andrés "El Fisco", un ricachón con capa, monóculo y un yate de dudosa legalidad, busca despertar a Sierpentez (Dragón/Agua), un Pokémon mitológico que, según los pescadores del lugar, "no trae nada bueno, chiquillo, hazme caso".

Pero, como buenos villanos incompetentes, meten la pata, y el Pokémon se despierta de malas pulgas, provocando tormentas, marejadas y, lo peor de todo, ¡haciendo que suban los precios de la gamba roja!

## ***Lugares clave de La Costera***

- Bahía de Marezuela: Punto de partida del viaje. Ciudad portuaria famosa por su mercado de pescado, su paseo marítimo y su gimnasio tipo Agua liderado por Ramiro el Patón, un pescador con redes y mucha mala leche.
- Las Islas del Fuego: Archipiélago volcánico donde el Profesor Emilio estudia la relación entre los Pokémon de fuego y el mar.
- Las Dunas del Levante: Un extenso desierto de arena dorada, hogar de Muleto (Normal) y Molintón (Roca/Viento), basado en los molinos de viento costeros.
- Cueva del Levante: El refugio de Sierpentez, donde el Team Almendrao' desata el caos.

## ***Pokémon exclusivos de la región***

- Sierpentez (Dragón/Agua) - El legendario de la región, basado en la serpiente mitológica, pero con algas enroscadas.

## ***Clímax de la historia***

Cuando el Andres despierta a Sierpentez, el Pokémon empieza a hacer de las suyas, desatando tormentas tropicales, oleadas de calor y, lo peor de todo, convirtiendo la bahía en una sopa de algas.

Deciden decidir el destino del Pokémon en un duelo de entrenadores

Finalmente, el jugador puede capturar a Sierpentez y restaurar el equilibrio en La Costera. Como recompensa, le regalan una fideuá +3 que restaura toda la vida de su equipo.

Para cerrar con broche de oro, el protagonista se enfrenta a la **Liga Pokémon de Valencia**, donde debe derrotar a los cuatro miembros del Alto Mando:

- **Doña Maruja** (Agua/Psíquico) – Anciana chismosa con un abanico que usa para lanzar golpes críticos.
- **El Mozo** (Bicho/Planta) – Agricultor que pelea con un rastrillo al hombro.
- **Roberto Alonso** (Normal/Siniestro) – Un vago profesional cuya habilidad "Vivir del Cuento" le permite esquivar ataques mientras se queja.
- **La Abuela Remedios** (Hada/Siniestro) – Dice que te va a curar con su "remedio casero", pero te manda al hospital.

El campeón final no es otro que Antonio Lobato, que no se rinde y quiere demostrar que los ricachones con yate todavía tienen algo de dignidad.

- **Antonio Lobato** (Metal, Eléctrico) – Por alguna razón pone mucho interés en que quieras saber cuanto vale tu coche, pero tú no sabes de qué te está hablando, no tienes uno.

## ***Personajes***

### El/La Protagonista – El Guardián del Agua

Vive con su abuela en un pequeño pueblo pesquero y su gran pasión es el mar. Al principio, ve el agua como su refugio personal, pero pronto descubrirá que sin ella, toda La Costera estará en peligro.

- **Pokémon Inicial:**
  - **Brisálido** (Agua/Fantasma) – Una gota de agua flotante con un aire melancólico.
  - **Nublarón** (Volador/Eléctrico) – Una nubecilla eléctrica que refleja su espíritu libre.
  - **Musgorra** (Planta/Tierra) – Un Pokémon musgoso que se esconde en un caparazón.

### Andrés "El Fisco" – Líder del Team Almendrao'

Un estafador profesional que quiere despertar a Sierpentez para convertir La Costera en un infierno tropical y vender flotadores a precio de oro.

- **Pokémon principales:**
  - **Morrósido** (Normal/Siniestro) – Finge estar débil para luego robar objetos.
  - **Chiringuito** (Agua/Fuego) – Un cangrejo con forma de chiringuito ambulante.
  - **Sierpentez** (Dragón/Agua, legendario) – Su arma definitiva.

- **Frases típicas:**

- "Este veranito va a ser mítico, chavales. Y solo yo tengo los abanicos oficiales de Sierpentez, ¡a precio de oro!"
- "Hacienda somos todos... menos yo, jaja."
- "Oye, sin malos rollos, ¿te vendo un pack de tumbonas?"

# Asets



## Tile sets

Hemos optado por utilizar la estética de los juegos de Pokémon de Game Boy Advance porque captura a la perfección la sensación de aventura clásica y la nostalgia de los títulos originales. Además, este estilo gráfico nos permite mantener un equilibrio entre detalle y simplicidad, logrando escenarios visualmente atractivos sin sobrecargar la pantalla.

Para lograr esto, hemos extraído los assets originales utilizando **Advance Map**, una herramienta de ROM hacking que nos permite visualizar, editar y exportar los mapas y sus tilesets desde los juegos de GBA. Posteriormente, escalamos estos assets a **32x32 píxeles** para que se integren correctamente en **RPG Maker**.

Para darle al juego un poco más de vida hemos subido a la saturación de todos los tilesets un poco, dándole un poco más de color al mundo de nuestro juego.

## Sprites y animaciones

Muchos de los **sprites de personajes, Pokémon y animaciones de combate** han sido extraídos directamente de las ROMs originales de Pokémon de GBA (principalmente *Pokémon FireRed* y *Emerald*), utilizando un conjunto de herramientas de ROM hacking especializadas. Esto nos ha permitido mantener la coherencia visual y estilística del universo de Pokémon en todas las interacciones del juego.

Las herramientas principales utilizadas para la extracción y modificación de sprites y animaciones incluyen:

- **UNLZ-GBA**: Un visor y editor gráfico que permite explorar la ROM en busca de imágenes comprimidas y exportarlas como archivos de imagen. Se ha usado principalmente para extraer sprites de entrenadores, objetos y elementos del HUD.
- **Nameless Sprite Editor (NSE)**: Herramienta avanzada para editar y reemplazar sprites de personajes y Pokémon en las ROMs. También permite modificar las hojas de animación y paletas de colores.
- **GBA Graphics Editor**: Útil para previsualizar y editar imágenes de fondo, íconos y menús del juego. Se ha utilizado especialmente para extraer

animaciones relacionadas con efectos de batalla.

- **Advance Map:** Además de editar mapas, también permite acceder y exportar ciertos gráficos asociados a eventos y scripts dentro del juego.

Una vez extraídos, los sprites fueron **escalados, organizados en hojas de sprites compatibles con RPG Maker** y, en algunos casos, editados o recoloreados para adaptarse a nuevas situaciones o variaciones narrativas del juego. También se han creado **nuevas animaciones a partir de bases extraídas**, combinando elementos existentes para simular efectos visuales únicos, como ataques especiales o interacciones personalizadas durante cinemáticas.

Este enfoque ha permitido construir un mundo visualmente familiar para los jugadores veteranos de Pokémon, pero con suficientes elementos nuevos y personalizados como para ofrecer una experiencia fresca y adaptada a la narrativa de *La Costera*.

# Mecánicas de juego

**Pokémon: La Costera** es un RPG de combates por turnos con un sistema de progresión basado en la captura, entrenamiento y gestión de un equipo de Pokémon. Implementa mecánicas clásicas de la franquicia con innovaciones en la manipulación del clima y la resistencia ambiental de los Pokémon.

## 1. Sistema de Combate

Los combates siguen una estructura de turnos en la que cada Pokémon puede realizar una acción por turno. Se mantiene el sistema de tipos con fortalezas y debilidades, además de habilidades y estados alterados que afectan el rendimiento del Pokémon en batalla.

### 1.1 Elementos Claves del Combate

#### ★ Acciones posibles por turno:

- Usar un movimiento (4 por Pokémon, cada uno con PP limitados).
- Usar un objeto.
- Cambiar de Pokémon.
- Huir (solo en encuentros salvajes).

#### ★ Probabilidad de crítico:

Los movimientos pueden infligir daño crítico con una probabilidad base del 6.25%, duplicando el daño infligido.

#### ★ Estados alterados:

Parálisis, Quemadura, Envenenamiento, Sueño y Congelación afectan a los Pokémon hasta que sean curados.

#### ★ Estados alterados temporales:

Reducción de estadísticas, confusión y atracción duran entre 2 y 5 turnos.

## 2. Sistema de Clima Dinámico

**Pokémon: La Costera** introduce un sistema climático que afecta directamente a la exploración. Cada zona de la región tiene un clima base, pero este puede cambiar debido a eventos de la historia o movimientos específicos.

## 2.1 Tipos de Clima y Efectos

| Clima                     | Efecto en combate  | Efecto en exploración  |
|---------------------------|--|--|
| <b>Despejado</b>          | Sin efectos adicionales.   | Normal.  |
| <b>Lluvia</b>             | Movimientos de Agua +50% de daño.<br>Movimientos de Fuego -50%.                  | Rutas más húmedas, algunos Pokémon solo aparecen con lluvia. |
| <b>Soleado</b>            | Movimientos de Fuego +50%.<br>Movimientos de Agua -50%.                          | Fatiga acelerada en Pokémon sin resistencia al calor.        |
| <b>Tormenta de Arena</b>  | Todos los Pokémon que no sean de tipo Tierra/Roca/Acero pierden PS cada turno.   | Visibilidad reducida en ciertas rutas.                       |
| <b>Tormenta Eléctrica</b> | Movimientos Eléctricos nunca fallan.<br>Movimientos Voladores -30% de precisión. | Probabilidad de encontrar Pokémon tipo Eléctrico aumentada.  |

## 3. Exploración del Mundo

La Costera es una región semiabierta con diferentes rutas y zonas urbanas interconectadas.

### 3.1 Movilidad del Jugador

- Desplazamiento terrestre a pie o en bicicleta.
- Desbloqueo progresivo de habilidades como Surf y Vuelo.

### 3.2 Interacciones Ambientales

- El clima afecta la accesibilidad de ciertas rutas.
- Algunas áreas solo son accesibles en condiciones climáticas específicas.
- Eventos dinámicos dependiendo del estado climático (p.ej., incendios en zonas secas, charcos en zonas con lluvia reciente).

## 4. Gimnasios y Liga Pokémon

El sistema de progresión incluye la obtención de **8 medallas de gimnasio** antes de acceder a la Liga Pokémon. Cada gimnasio introduce mecánicas únicas relacionadas con su tipo predominante y el clima de la zona.

### 4.1 Características del Sistema de Gimnasios

- Cada gimnasio tiene un líder con un equipo optimizado según su tipo.
- Existen desafíos previos antes del combate final con el líder (puzles ambientales, resistencia climática, etc.).

### 4.2 Liga Pokémon

La Liga Pokémon sigue una estructura de torneo con **4 entrenadores de élite** y el campeón.

- Cada batalla introduce **cambios climáticos progresivos** a lo largo del combate.
- Enfrentamientos adaptativos donde los Pokémon rivales pueden cambiar de estrategia según el clima activo.

## 5. Team Almendrao' y Batalla Final

El antagonista, **Andrés "El Fisco"**, manipula el clima con el legendario **Sierpentez** para mantener **un verano perpetuo** en La Costera, beneficiando su negocio de productos de playa.

### 5.1 Mecánica del Enfrentamiento Final

- Batalla en **tres fases**, cada una con una intensificación del clima cálido.
- Andrés usa objetos que aumentan la duración del clima soleado.
- La única forma de contrarrestar su estrategia es utilizando la **Lluvia Salvadora** para restaurar el equilibrio climático.




























































## 6. Conclusión

Pokémon: La Costera mantiene la jugabilidad tradicional de Pokémon, pero incorpora **mecánicas ambientales avanzadas**, aumentando la complejidad estratégica del combate y la exploración. Su sistema de clima y fatiga obliga al jugador a **gestionar su equipo en función del entorno**, ofreciendo una experiencia más desafiante y realista dentro del universo Pokémon.

## Tabla de tipos

| <u>TIPO</u>  | <u>EFFECTIVO</u>   | <u>NO EFFECTIVO</u>  | <u>DEBILIDAD</u>  | <u>INMUNE</u>   |
|--|--|--|---|---|
|  Normal     |  |    |    |    |
|  Fuego      |      |       |      |   |
|  Agua       |     |     |     |   |
|  Eléctrico  |    |     |    |   |
|  Planta     |     |    <br>             |             |   |
|  Hielo    |      |       |     |   |
|  Lucha    |    <br> |    <br>  |      |   |
|  Veneno   |    |       |     |   |
|  Tierra   |    <br> |    |      |  |
|  Aéreo    |     |     |      |  |
|  Psíquico |    |    |      |   |
|  Bicho    |     |   <br>    |      |   |



|  |   |   |  |   |
|--|---|---|--|---|
|  <b>Roca</b>      |     |      |    <br> |   |
|  <b>Fantasma</b>  |     |    |    |   |
|  <b>Dragón</b>    |    |    |     |   |
|  <b>Siniestro</b> |     |      |     |    |
|  <b>Acero</b>     |      |     |     |    |
|  <b>Hada</b>      |      |      |    |    |

## Escena: Encuentro con Andrés #1

### Resumen:

El protagonista se esconde en la hierba y presencia los planes secretos de **Andrés**, líder de **Team Almendrao**, que busca cruzar un lago con Surf para despertar al Pokémon legendario **Sierpentez**. Andrés descubre al protagonista y le ordena a su becaria que lo enfrente en combate mientras él huye.

### Explicación del código.

#### ★ Movimiento inicial de Andrés

@>Set Move Route: [Andrés]

\$>Turn Right

Andrés se gira hacia la derecha, como si mirara hacia la cueva. (Esto no es visible por el jugador)

#### ★ Reacción del Jugador

@>Show Animation: Player, [Exclaim bubble]

@>Wait: 20 frame(s)

El jugador reacciona al escuchar gente y exclama (¡sorpresa!).

#### Movimiento inicial de Andrés:

@>Set Move Route: [Andrés]

\$>Turn Right

#### Reacción del Jugador:

@>Show Animation: Player, [Exclaim bubble]

@>Wait: 20 frame(s)

#### Movimiento del Jugador (automático):

@>Set Move Route: Player

\$>Move Up x6

\$>Move Right x10

\$>Move Left x17

#### Scroll dramático hacia la cueva:

@>Scroll Map: Right, 7, 4

### Monólogo de Andrés:

@>Text: \tg[Andrés] Tch... ahí está. La cueva del Sierpentez.

@>Text: \tg[Andrés] Y yo con el almacén hasta arriba: colchonetas, palas de playa, bañadores, gafas de buceo, frisbees... hasta gorras con ventilador.

@>Text: \tg[Andrés] Je, je... se van a dejar los cuartos...

@>Text: \tg[Andrés] Pero claro... sin Surf no cruzo.

### Reflexión del jugador:

@>Text: ¿Quién era ese? ¿Sierpentez...? ¿Está planeando controlar el clima?

### Confrontación directa:

@>Show Animation: [Andrés], [Question bubble]

@>Text: \tg[Andrés] ¡Eh tú! El de la maleza. ¿Espía, fan o mindundi con curiosidad?

@>Text: \tg[Andrés] ¡Becaria! Atiende al mirón, que yo tengo negocio urgente.

### Huida de Andrés:

@>Set Move Route: [Andrés]

\$>Turn Left

@>Wait: 20 frame(s)

@>Set Move Route: [Andrés]

\$>Move Left x2, Down x2, Left x2

### Entrada de la Becaria:

@>Set Move Route: [Becario]

\$>Move Left x5

@>Wait for Move's Completion

### Diálogo inicial de la becaria:

@>Text: \tg[Becaria] \r Te acabas de ganar una clase exprés de estilo... ¡y de combates!

### Inicio del combate:

@>Conditional Branch: Script: pbTrainerBattle(PBTrainers::BECARIO, "Becc")

@>Control Self Switch: A = ON

### Movimiento y combate de la becaria:

@>Set Move Route: [Becario]

\$>Move Left x5

@>Wait for Move's Completion

@>Text: \tg[Becaria] \r Te acabas de ganar una clase exprés de estilo... ¡y de combates!

@>Conditional Branch: Script: pbTrainerBattle(...)

@>Control Self Switch: A = ON

### Cierre cómico:

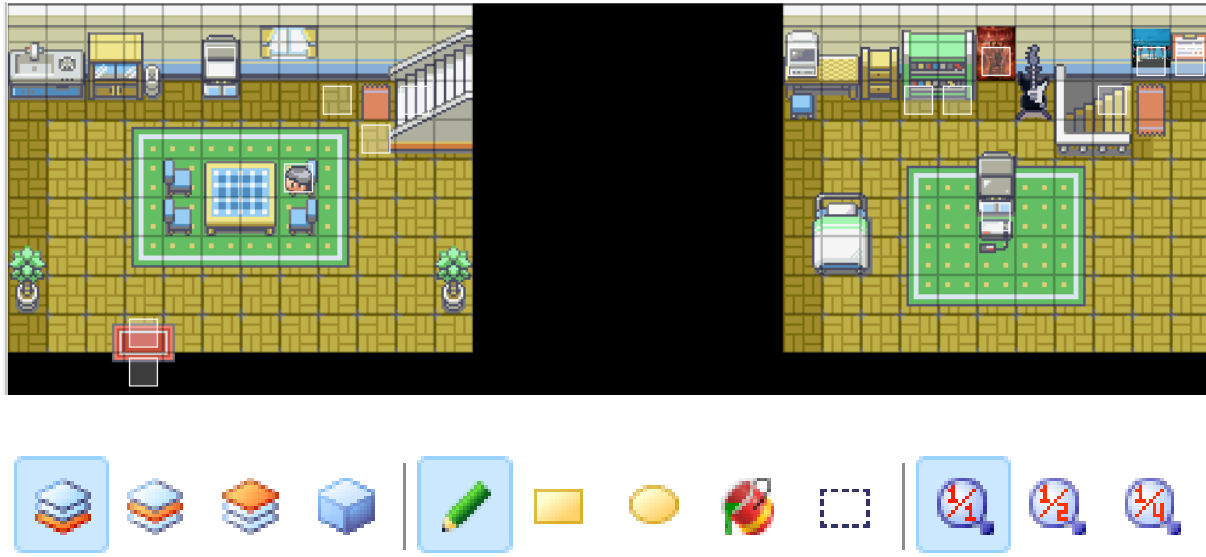
@>Scroll Map: Left, 7, 4

@>Text: \tg[Becaria] Joder... Mejor me voy yendo.

@>Set Move Route: [Becario]

(Después de esto ya recuperas la movilidad de tu personaje y la cámara vuelve a su posición original)

# Creación de mapas



Seleccionar capa de fondo (Lower Layer)

→ Permite dibujar en la capa de suelo/base del mapa (piso, caminos, agua...).

Seleccionar capa de objetos (Middle Layer)

→ Permite dibujar en la capa intermedia (paredes, hierba alta, árboles... cosas que "suben" del suelo).

Seleccionar capa de detalles (Upper Layer)

→ Permite dibujar detalles superiores como techos, sombras, señales, decoraciones pequeñas.

Seleccionar capa automática (Events Layer)

→ Cambia para trabajar en la capa de eventos (NPCs, puertas, zonas de cambio de mapa, etc).

Lápiz

→ Herramienta de dibujo libre para pintar casillas una a una.

Rectángulo

→ Dibuja rectángulos del tile que tengas seleccionado.

Círculo

→ Dibuja círculos/óvalos de tiles.

Rellenar (cubeta)

→ Rellena un área cerrada con el tile que hayas seleccionado (como el bote de pintura de un editor de imágenes).

#### Selección

→ Herramienta para seleccionar una zona de tiles y copiarla/cortarla/pegarla.

#### Zoom In

→ Acercar el mapa (hacer zoom).

#### Zoom Out

→ Alejar el mapa (hacer zoom out).

## Resumen ultra rápido

| Icono     | Función                                 |
|-----------|---|
| Capas     | Cambiar entre suelo, objetos o detalles |
| Evento    | Entrar en modo de edición de eventos    |
| Lápiz     | Dibujar casillas                        |
| Cubo      | Rellenar zonas                          |
| Cuadro    | Dibujar áreas rectangulares             |
| Prohibido | Borrar eventos                          |
| Selección | Seleccionar parte del mapa              |
| Lupa      | Acercar o alejar el mapa                |

# Funcionamiento de los "Tiles" y Colisiones en el Editor de Mapas

## Casillas y Capas

- El mapa está dividido en **casillas cuadradas** (tiles), de **32x32 píxeles** cada una.
- Cada casilla puede tener **una imagen base**, pero esa imagen puede componerse de **varias capas superpuestas**:
  - **Capa de Suelo (Lower Layer)**: El fondo o suelo (tierra, agua, caminos...).
  - **Capa de Objetos (Middle Layer)**: Elementos como árboles, paredes, hierba alta...
  - **Capa de Detalles (Upper Layer)**: Techos, decoraciones, sombras, etc.
  - **Capa de Eventos (Events Layer)**: NPCs, puertas, teleportadores, etc.

Esto permite que una casilla pueda verse compleja (por ejemplo, un suelo de piedra con una farola encima y un personaje al lado) aunque solo haya **una casilla física**.

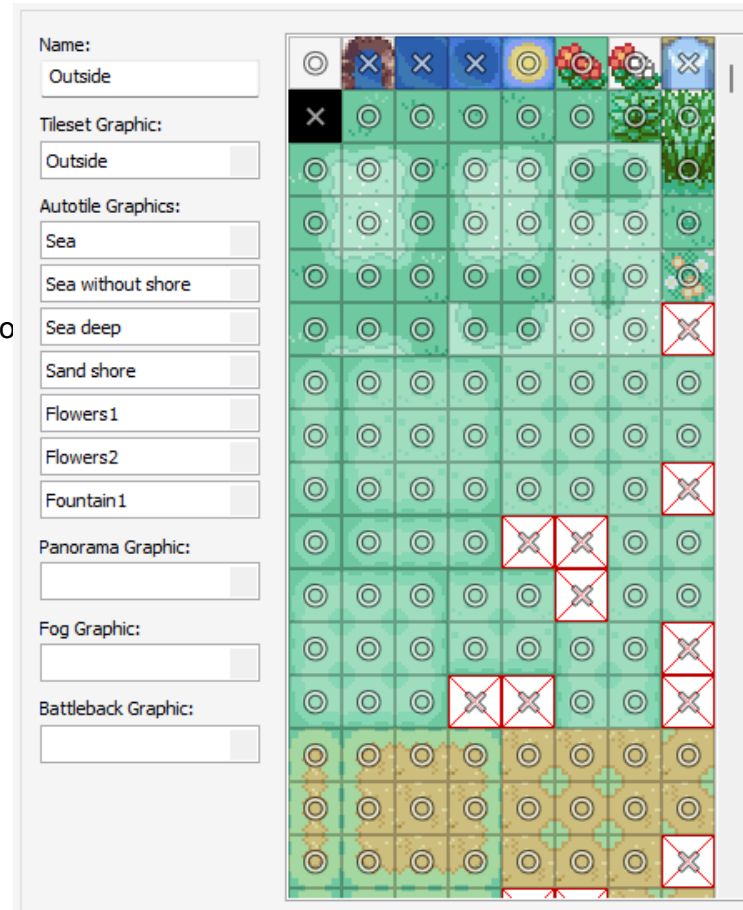
## Definición de Colisiones

- Cada casilla puede tener **propiedades de colisión** que determinan si el jugador puede caminar sobre ella o no.

- Estas propiedades se definen en el **Editor de Tilesets**:
  - En RPG Maker XP, se accede a través de **Database (F9) → Tilesets → [Seleccionar tileset]**.

Ahí se puede marcar cada tile con:

- **(Círculo)**: El jugador **puede caminar** sobre esa casilla.
- ✕ **(Equis)**: El jugador **no puede caminar** (bloquea el paso).
- ☆ **(Estrella)**: La casilla es "transparente" para el paso pero **se dibuja por encima del personaje** (por ejemplo, ramas o tejados).



## Resumen rápido

| Elemento                | Función  |
|-------------------------|--|
| Casilla (Tile)          | Unidad básica del mapa (32x32 px)              |
| Capas                   | Superponen imágenes para hacer mapas más ricos |
| Propiedades de Colisión | Controlan si el jugador puede moverse o no     |
| Editor de Tileset (F9)  | Lugar para configurar colisiones y pasajes     |



# Base de Datos de Pokémon

[pokemon.txt](#)

Cada Pokémon en el sistema se define mediante una serie de propiedades. A continuación se explica cada campo usando como ejemplo a **Bulbasaur**:

[1]

Name=Bulbasaur

InternalName=BULBASAUR

Type1=GRASS

Type2=POISON

BaseStats=45,49,49,45,65,65

GenderRate=FemaleOneEighth

GrowthRate=Parabolic

BaseEXP=64

EffortPoints=0,0,0,0,1,0

Rareness=45

Happiness=70

Abilities=OVERGROW

HiddenAbility=CHLOROPHYLL

Moves=1,TACKLE,3,GROWL,7,LEECHSEED,9,VINEWHIP,13,POISONPOWDER,13,SLEEPPOWDER,15,TAKEDOWN,19,RAZORLEAF,21,SWEETSCENT,25,GROWTH,27,DOUBLEEDGE,31,WORRYSEED,33,SYNTHESIS,37,SEEDBOMB

EggMoves=AMNESIA,CHARM,CURSE,ENDURE,GIGADRAIN,GRASSWHISTLE,GRASSYTERRAIN,INGRAIN,LEAFSTORM,MAGICALLEAF,NATUREPOWER,PETALDANCE,POWERWHIP,SKULLBASH,SLUDGE

Compatibility=Monster,Grass

StepsToHatch=5355

Claudia Mur  
Jordi Cañas

2n C

20/03/2025

Height=0.7

Weight=6.9

Color=Green

Habitat=Grassland

RegionalNumbers=1,0

Kind=Semilla

Pokedex=A Bulbasaur es fácil verle echándose una siesta al sol. La semilla que tiene en el lomo va creciendo cada vez más a medida que absorbe los rayos del sol.

BattlerPlayerY=24

BattlerEnemyY=32

BattlerAltitude=0

Evolutions=IVYSAUR,Level,16

## [1]

Número interno de la Pokédex o listado. Sirve para identificar al Pokémon dentro del juego.

### **Name**

Nombre mostrado en el juego. Puede llevar tildes o caracteres especiales.

### **InternalName**

Nombre interno usado por el motor del juego (siempre en mayúsculas y sin espacios).

### **Type1**

Primer tipo elemental del Pokémon (por ejemplo, GRASS).

### **Type2**

Segundo tipo elemental (opcional; puede omitirse si solo tiene uno).

## **Estadísticas base y crecimiento**

### **BaseStats**

Valores iniciales de sus estadísticas, en este orden: PS (HP), Ataque, Defensa, Velocidad, Ataque Especial, Defensa Especial.

### **GenderRate**

Ratio de género. Algunos ejemplos:

- AlwaysMale (siempre macho)
- AlwaysFemale (siempre hembra)
- FemaleOneEighth (12,5% hembra y 87,5% macho, como Bulbasaur)

### **GrowthRate**

Curva de crecimiento de experiencia. Ejemplo: Parabolic (crece de forma equilibrada).

### **BaseEXP**

Cantidad de experiencia que otorga al ser derrotado.

### **EffortPoints**

EVs (Effort Values) que otorga al derrotarlo, en el mismo orden de BaseStats.

## **Datos de captura, felicidad y habilidades**

### **Rareness**

Valor de rareza para captura. Cuanto más bajo, más difícil de capturar.

### **Happiness**

Felicidad base con la que nace el Pokémon. Afecta a movimientos y evoluciones que dependen de la felicidad.

### **Abilities**

Lista de habilidades normales (puede tener una o dos, separadas por coma).

### **HiddenAbility**

Habilidad oculta, disponible en situaciones especiales.

## **Movimientos**

### **Moves**

Lista de movimientos que aprende automáticamente al subir de nivel.  
Formato: Nivel, Movimiento, Nivel, Movimiento, etc.

### **EggMoves**

Movimientos que puede heredar al nacer de un huevo.

## Crianza y biología

### Compatibility

Grupos huevo a los que pertenece. Puede tener uno o dos.

### StepsToHatch

Número de pasos necesarios para que un huevo de este Pokémon eclosione.

### Height

Altura en metros.

### Weight

Peso en kilogramos.

### Color

Color principal del Pokémon, usado para organizar la Pokédex o efectos de brillo.

### Habitat

Hábitat natural (como selva, bosque, ciudad, mar, etc.).

## Pokédex y combate

### RegionalNumbers

Número que ocupa en una Pokédex regional (puede tener varios números si aparece en distintas regiones).

### Kind

Descripción breve del tipo de criatura (por ejemplo, "Semilla").

### Pokedex

Texto descriptivo que aparece en la Pokédex.

## Apariencia en combate

### BattlerPlayerY

Posición vertical del sprite del jugador durante un combate.

### BattlerEnemyY

Posición vertical del sprite enemigo en combate.

### BattlerAltitude

Altura a la que se muestra en el combate (por ejemplo, para Pokémon voladores).

# Evolución

## Evolutions

Define cómo evoluciona.

Formato: Pokémon destino, Método, Parámetro.

Por ejemplo: IVYSAUR, Level, 16 (evoluciona al nivel 16).

## Notas adicionales

- Los campos de listas (como Moves o EggMoves) están separados por comas.
- No se deben usar tildes ni caracteres raros en los nombres internos (InternalName).

## Resumen rápido

| Campo           | Descripción   |   |  |
|-----------------|---|---|--|
| Número          | Identificador único del Pokémon                             | <u>Abilities</u>                        | Lista de habilidades normales                    |
|                 |   | <u>HiddenAbility</u>                    | Habilidad oculta especial                        |
| Name            | Nombre visible para el jugador                              | <u>Moves</u>                            | Movimientos que aprende (nivel, movimiento)      |
| InternalName    | Nombre interno (mayúsculas, sin espacios)                   | <u>EggMoves</u>                         | Movimientos heredados por cría                   |
|                 |   | <u>Compatibility</u>                    | Grupos huevo (para criar)                        |
| Type1 / Type2   | Tipos principales   | <u>StepsToHatch</u>                     | Pasos para eclosionar un huevo                   |
| BaseStats       | PS, Ataque, Defensa, Velocidad, At. Especial, Def. Especial | <u>Height / Weight</u>                  | Altura en metros / Peso en kilos                 |
| GenderRate      | Probabilidad de género                                      | <u>BattlerPlayer Y / BattlerEnemy Y</u> | Ajustes de posición de <u>sprites</u> en batalla |
| GrowthRate      | Curva de experiencia  |   |  |
| BaseEXP         | Experiencia otorgada al vencerlo                            | <u>BattlerAltitude</u>                  | Altura flotante en combate                       |
| EffortPoints    | EVs otorgados   |   |  |
| Rareness        | Dificultad de captura                                       | <u>Evolutions</u>                       | Método de evolución (Pokémon, forma, condición)  |
| Happiness       | Nivel de felicidad inicial                                  |   |  |
| Color           | Color principal   |   |  |
| Habitat         | Tipo de hábitat natural                                     |   |  |
| RegionalNumbers | Número/s en la Pokédex regional                             |   |  |
| Kind            | Tipo breve de criatura ("Semilla", "Ratón"... )             |   |  |
| Pokedex         | Texto descriptivo de Pokédex                                |   |  |

# Base de Datos de Movimientos

[moves.txt](#)

Cada movimiento del juego se define en una línea del archivo moves.txt. Los campos están separados por comas, y cada campo tiene un propósito específico que se describe a continuación.

## EJEMPLO:

1,MEGAHORN,Megacuerno,000,120,BUG,Physical,85,10,0,00,0,abef,"Violenta embestida con cuernos imponentes."

## DESCRIPCIÓN DE CAMPOS:

1. ID numérico  
Número interno del movimiento. Debe ser único.
2. InternalName  
Nombre interno usado por el motor. Siempre en mayúsculas, sin espacios ni tildes.
3. Name  
Nombre visible que se muestra en el juego. Puede incluir tildes o caracteres especiales.
4. FunctionCode  
Código que indica el comportamiento especial del movimiento.  
000 significa que no tiene efectos especiales.
5. BasePower  
Poder base del movimiento. Si es 0, no causa daño directo.
6. Type  
Tipo elemental del movimiento (por ejemplo: NORMAL, FIRE, WATER, etc.).
7. Category  
Categoría del movimiento: Physical, Special o Status.
8. Accuracy  
Precisión del movimiento en porcentaje. Si es 0, tiene precisión perfecta.
9. TotalPP  
Puntos de Poder (PP) máximos del movimiento.

10. AdditionalEffectChance

Probabilidad de que ocurra el efecto secundario (en porcentaje).  
Se deja en 0 si el movimiento no tiene efecto adicional.

11. Target

Define a qué objetivo afecta el movimiento.

00 = un enemigo

10 = todos los enemigos

0A = el usuario

Otros valores también están disponibles según la documentación del motor.

12. Priority

Prioridad del movimiento en el turno.

0 = prioridad normal

Positivos = prioridad alta (actúa antes)

Negativos = prioridad baja

13. Flags

Conjunto de letras que indican propiedades especiales del movimiento:

a = afectado por Protect

b = reflejable con Magic Coat

e = contacto físico

f = puede copiarse con Mimic

Se pueden combinar varias letras, sin separadores.

14. Description

Texto mostrado al jugador como descripción del movimiento.

EJEMPLO DETALLADO:

1,MEGAHORN,Megacuerno,000,120,BUG,Physical,85,10,0,00,0,abef,"Violenta embestida con cuernos imponentes."

- ID: 1
- InternalName: MEGAHORN
- Name: Megacuerno
- FunctionCode: 000 (sin efecto especial)
- BasePower: 120
- Type: BUG
- Category: Physical



- Accuracy: 85%
- TotalPP: 10
- AdditionalEffectChance: 0
- Target: 00 (un enemigo)
- Priority: 0
- Flags: abef (afectado por Protect, reflejable, contacto físico, se puede copiar)
- Description: Violenta embestida con cuernos imponentes.

#### CONCLUSIÓN:

- El archivo moves.txt define cada movimiento con 14 campos obligatorios, en orden fijo y separados por comas.
- El InternalName debe ser único y coincidir con otras referencias internas.
- Las Flags controlan las interacciones del movimiento con mecánicas como Protect, contacto, o copia.
- Es obligatorio conservar el orden exacto de campos. No se deben omitir ni reordenar.
- El campo de descripción es el único que va entre comillas, especialmente si contiene comas.

# Animaciones de Movimientos

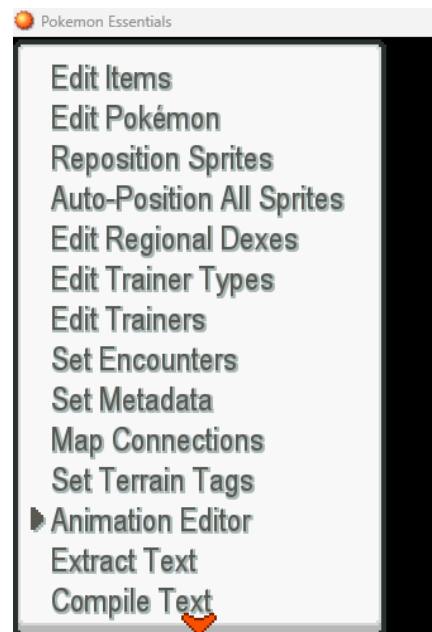
## FORMATO Y UBICACIÓN

- Todas las animaciones están guardadas en el archivo Animations.rxdata.
- Se accede y edita mediante el **Editor de animaciones**, desde el menú de depuración (Debug) → Editor de animaciones.
- Cada animación tiene un **ID**, un **nombre interno** y una **lista de celdas, fotogramas y comandos**.

## ESTRUCTURA GENERAL DE UNA ANIMACIÓN

Una animación se compone de:

1. **ID de la animación**
  - Número único que identifica la animación.
  - Este ID se asocia a los movimientos en moves.txt.
2. **Nombre interno**
  - Nombre descriptivo, por ejemplo: Tackle, Fire Blast.
  - Solo sirve para organización interna en el editor.
3. **Celdas (cells)**
  - Representan gráficos individuales usados en cada fotograma.
  - Vienen de hojas de sprites (Graphics/Animations/).
4. **Fotogramas (frames)**
  - Cada uno indica qué celdas mostrar, dónde y con qué efectos (escala, opacidad, rotación).
  - Se colocan en secuencia para simular movimiento.

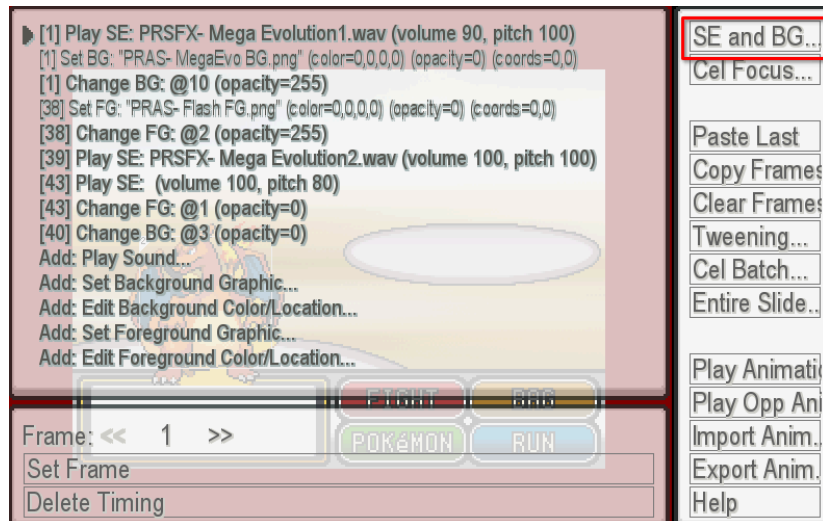


## 5. Comandos (timing)

- Efectos especiales sincronizados con los fotogramas (pantalla parpadeando, vibración, sonido, etc.).
- Controlan sonidos, flashes, vibración, tintes, etc.

### CREACIÓN DE UNA ANIMACIÓN NUEVA (PASOS BÁSICOS)

1. Entra en el **Editor de animaciones** desde el menú Debug.
2. Crea una nueva animación con un nombre identificable.
3. Añade celdas desde los sprites disponibles (Graphics/Animations/).
4. Crea los fotogramas uno por uno, colocando las celdas donde quieras.
5. Añade comandos de sonido o efectos en el timeline.
6. Guarda la animación. Se asignará un ID automáticamente.
7. Usa ese ID en el campo Animation de moves.txt.



### ARCHIVOS RELACIONADOS

- Carpeta Graphics/Animations/: Contiene los sprites utilizados por las animaciones.

### CONCLUSIÓN RÁPIDA

- Las animaciones se editan visualmente con la herramienta interna de Pokémon Essentials.
- Cada movimiento puede tener una animación del usuario y del objetivo.
- Se usan celdas gráficas, fotogramas y comandos sincronizados.
- La vinculación final se hace en moves.txt mediante el ID de la animación.

# Base de Datos

# HABILIDADES

[abilities.txt](#)

Cada habilidad se define en una línea dentro del archivo abilities.txt, utilizando valores separados por comas.

EJEMPLO:

1,STENCH,Hedor,"Debido al mal olor que emana, al atacar al rival puede hacerlo retroceder."

DESCRIPCIÓN DE CAMPOS:

1. ID

Número interno de la habilidad.

- Debe ser un valor único e incremental.
- Se utiliza internamente para referencias cruzadas.

2. InternalName

Nombre interno de la habilidad.

- Debe ir todo en mayúsculas.
- No debe contener espacios, tildes ni caracteres especiales.
- Se usa en código para identificar la habilidad.

3. Name

Nombre mostrado al jugador en la interfaz del juego.

- Puede incluir tildes, espacios y caracteres especiales.
- Este es el nombre que se muestra en combate, menús y Pokédex.

4. Description

Texto descriptivo de la habilidad.

- Aparece en la Pokédex o cuando se consulta la información del Pokémon.
- Debe ir siempre entre comillas si incluye comas.

- No debe ser demasiado larga para que encaje en las interfaces.

#### EJEMPLO DETALLADO:

- ID: 1
- InternalName: STENCH
- Name: Hedor
- Description: Debido al mal olor que emana, al atacar al rival puede hacerlo retroceder.

#### CONCLUSIÓN:

- El archivo abilities.txt define habilidades con 4 campos estrictamente ordenados y separados por comas.
- El InternalName es clave para referenciar la habilidad en scripts y datos de Pokémon.
- La Description debe ir entre comillas si contiene comas.
- Este archivo es esencial para que los Pokémon puedan tener habilidades funcionales en el juego.

# Minijuegos

# Tragaperras



## Descripción General

Este código implementa un mini-juego de máquina tragaperras en un videojuego, con diferentes niveles de dificultad (fácil, medio y difícil). La estructura del código está basada en clases que representan los distintos elementos del juego, como los carretes de la máquina, las puntuaciones, y la interfaz de la escena.

Este minijuego está por defecto en el feach de Pokémon essentials, pero lo he modificado para aumentar las recompensas y las probabilidades de ganar

## Clases Principales

### 1. SlotMachineReel

- Representa un carrete en la máquina tragaperras.
- **Atributos:**
  - @reel: Un arreglo que contiene los símbolos que pueden aparecer en el carrete.
  - @toppos: Posición vertical del carrete.
  - @spinning: Estado que indica si el carrete está girando.
  - @stopping: Estado que indica si el carrete está parando.
  - @slipping: Un valor que controla el efecto de deslizamiento al parar.
  - SCROLLSPEED: Velocidad del desplazamiento del carrete se puede cambiar, pero ha de ser un divisor de 48.
  - ICONSPOOL: Arreglo que contiene las configuraciones de símbolos disponibles según la dificultad.
  - SLIPPING: Efectos de deslizamiento posibles.
- **Métodos:**
  - initialize(x, y, difficulty=1): Inicializa el carrete con una posición y dificultad específica.
  - startSpinning: Comienza el giro del carrete.
  - stopSpinning(noslipping=false): Detiene el giro del carrete.
  - showing: Devuelve los tres símbolos que se muestran en el carrete (arriba, centro, y abajo).
  - update: Actualiza la animación y el estado del carrete.

### 2. SlotMachineScore

- Representa un contador de puntos o fichas en la pantalla de la máquina tragaperras.
- **Atributos:**



- @score: El puntaje actual.

- **Métodos:**

- initialize(x, y, score=0): Inicializa el contador de puntuaciones.
- score=(value): Establece el valor del puntaje y lo limita al valor máximo de fichas.
- refresh: Actualiza la visualización del contador de puntuaciones.

### 3. SlotMachineScene

- Maneja la lógica de la escena de la máquina tragaperras, incluyendo la gestión de los carretes, puntuaciones y animaciones.

- **Atributos:**

- @gameRunning: Indica si el juego está en ejecución.
- @gameEnd: Indica si el juego ha terminado.
- @wager: El número de fichas apostadas.
- @replay: Indica si se activó una jugada de repetición.

- **Métodos:**

- update: Actualiza los elementos visuales de la escena.
- pbPayout: Calcula y muestra el pago de la máquina basándose en los resultados.
- pbStartScene(difficulty): Inicializa la escena de la máquina tragaperras con una dificultad dada.
- pbMain: El ciclo principal del juego, donde se manejan las interacciones del usuario.
- pbEndScene: Finaliza la escena y limpia los recursos.

### 4. SlotMachine

- Un objeto que representa la máquina tragaperras en sí.
- **Métodos:**

- `initialize(scene)`: Inicializa la máquina con la escena.
- `pbStartScreen`: Inicia la pantalla del juego, ejecutando la escena y el ciclo principal.

## Lógica del Juego

- El juego permite tres niveles de dificultad (fácil, medio y difícil), cada uno con diferentes combinaciones posibles de símbolos en los carretes.
- El jugador puede apostar fichas para jugar, con un máximo de tres fichas por jugada.
- Se determinan combinaciones ganadoras en las filas del carrete, y las combinaciones incluyen premios o la posibilidad de un "replay" (jugada gratuita).
- Los resultados de la tragaperras se calculan de acuerdo con las combinaciones de símbolos alineadas en las filas, con pagos diferentes para cada combinación.
- Si el jugador gana, se muestra una animación de victoria y se realiza el pago.
- Si el jugador pierde, se muestra una animación de pérdida y no se paga.

## Combinaciones Ganadoras y Pagos

Las combinaciones ganadoras posibles incluyen:

- **Tres símbolos iguales**: Se paga una cantidad de fichas dependiendo del símbolo (por ejemplo, 777 paga más que otros símbolos).
- **Combinaciones de "777" en varias posiciones**: Estas combinaciones especiales tienen pagos altos.
- **Combinación de "Cerezas"**: Si hay dos cerezas en una fila, el jugador puede ganar una pequeña cantidad.

## Gestión de Recursos y Animaciones

- **Gráficos y Animaciones**:
  - La animación de los carretes es controlada mediante la actualización del bitmap de cada carrete.

- Se utilizan imágenes animadas para mostrar los carretes y las animaciones de pago o pérdida.
- La animación de las luces y las ventanas del juego se actualizan según el estado del juego (gano o pierdo).
- **Manejo de Fichas:**
  - Las fichas se gestionan mediante el contador `@sprites["credit"].score`, que representa las fichas del jugador.
  - Cuando el jugador gana, las fichas se aumentan según el pago calculado en `pbPayout`.
  - El jugador puede seguir jugando mientras tenga fichas y no haya alcanzado el máximo permitido.

## Conclusión

Este código implementa una máquina tragaperras en un estilo gráfico adecuado para un juego RPG, manejando diferentes niveles de dificultad, combinaciones ganadoras y animaciones. Los recursos gráficos y de sonido se gestionan a través de la interfaz de sprites, mientras que la lógica del juego está estructurada en torno a las clases de carrete, puntuación y escena. La jugabilidad se basa en apuestas y pagos dinámicos, con un sistema de repetición en caso de ganar ciertas combinaciones.

# Voltorb Flip



## Descripción general

El mini-juego **Voltorb Flip** es un juego basado en la mecánica de un tablero donde los jugadores deben descubrir valores ocultos en un conjunto de casillas. Los valores incluyen fichas de valor 1, 2 o 3 y "Voltorbs" que al ser revelados causan la pérdida del juego. Los jugadores deben usar su lógica para evitar los Voltorbs y acumular puntos. Los valores en el tablero están distribuidos aleatoriamente, y se necesita resolver el juego a través de la exploración y la memoria.

Este minijuego está por defecto en el feach de Pokémon essentials, pero lo he modificado para aumentar las recompensas.

## Función Principal: VoltorbFlip

Esta clase encapsula la lógica y las interacciones del mini-juego. Se encargará de gestionar la visualización, las entradas del jugador, las reglas del juego y la progresión de los niveles.

## Métodos Principales

1. update

- **Descripción:** Actualiza el estado visual del juego.
- **Acción:** Llama a `pbUpdateSpriteHash(@sprites)` para actualizar los gráficos en pantalla.

## 2. `pbStart`

- **Descripción:** Inicia una nueva partida configurando los valores iniciales.
- **Acción:** Establece el nivel actual a 1 y configura los rangos máximos y mínimos de puntos para cada nivel. Luego, llama a `pbNewGame` para comenzar una nueva partida.

## 3. `pbNewGame`

- **Descripción:** Inicializa las variables necesarias para una nueva partida y configura el tablero de juego.
- **Acción:**
  - Inicializa los sprites del juego.
  - Crea una distribución aleatoria de valores para las casillas del tablero (`@squares`).
  - Dibuja las casillas con valores aleatorios.
  - Calcula los números de fila y columna de las casillas y muestra la interfaz inicial (nivel, monedas).
  - Si es la primera ronda, ejecuta un efecto de cortina en la pantalla.

## 4. `pbCreateSprites`

- **Descripción:** Crea y configura los sprites gráficos del juego.
- **Acción:**
  - Crea una serie de sprites de fondo, texto, y cursor para representar los elementos del juego en la pantalla.
  - Configura imágenes de fondo, niveles, cortinas, monedas, y más.

## 5. `getInput`

- **Descripción:** Gestiona las entradas del jugador.
- **Acción:**
  - Detecta las teclas de entrada del jugador para mover el cursor, marcar una casilla, o revelar una casilla.
  - Permite el uso de las teclas de dirección (arriba, abajo, izquierda, derecha) para mover el cursor y la tecla de confirmación para seleccionar casillas o marcar.
  - Si se descubre un "Vultorb", el juego termina con una animación de explosión y una penalización de nivel.

#### 6. pbUpdateRowNumbers

- **Descripción:** Actualiza los números en las filas del tablero.
- **Acción:**
  - Calcula y muestra los números de puntos y Vultorbs en las filas del tablero.

#### 7. pbUpdateColumnNumbers

- **Descripción:** Actualiza los números en las columnas del tablero.
- **Acción:**
  - Calcula y muestra los números de puntos y Vultorbs en las columnas del tablero.

#### 8. pbCreateCoins

- **Descripción:** Crea las representaciones visuales de las monedas acumuladas.
- **Acción:**
  - Muestra el número de monedas actuales en el marcador del jugador.

#### 9. pbUpdateCoins

- **Descripción:** Actualiza la visualización de las monedas en pantalla.
- **Acción:**

- Actualiza las imágenes de las monedas ganadas y las monedas actuales.

#### 10. pbAnimateTile

- **Descripción:** Anima la revelación de una casilla seleccionada.
- **Acción:**
  - Muestra una animación al revelar una casilla, indicando si se ha descubierto un Voltorb o un valor numérico.

#### 11. pbShowAndDispose

- **Descripción:** Muestra los resultados y finaliza el juego.
- **Acción:**
  - Muestra el mensaje de finalización del juego y limpia la pantalla.

## Variables Principales

- @level: Nivel actual del juego (1-8).
- @points: Puntos actuales del jugador.
- @squares: Array de casillas del tablero, cada casilla tiene un valor (0 = Voltorb, 1 = ficha de valor 1, 2 = ficha de valor 2, 3 = ficha de valor 3).
- @marks: Array que guarda las casillas marcadas por el jugador.
- @voltorbNumbers: Números de Voltorbs en las filas y columnas.
- @sprites: Hash de todos los sprites del juego (fondo, cursor, cortina, etc.).
- @coins: Representaciones gráficas de las monedas del jugador.

## Flujo del Juego

1. **Inicio:** El juego comienza con un nivel inicial y se genera un tablero de 5x5 casillas con valores aleatorios.

2. **Interacción:** El jugador mueve un cursor para seleccionar casillas. Si una casilla tiene un valor numérico (2 o 3), el jugador gana puntos.
3. **Marcado:** El jugador puede marcar casillas para recordar las que ha descubierto.
4. **Voltorbs:** Si el jugador selecciona una casilla con un Voltorb (valor 0), el juego termina y el jugador pierde puntos.
5. **Finalización:** Si el jugador revela todas las casillas sin Voltorbs, gana el juego y recibe monedas. Si se pierde, el nivel puede bajar dependiendo de los resultados.

## Sistema de Niveles

- Hay 8 niveles con rangos de puntos diferentes, que se utilizan para determinar los valores máximos y mínimos de las casillas.
- El nivel aumenta cuando se completa un juego sin perder, y puede bajar si el jugador selecciona demasiados Voltorbs.

## Animaciones

El juego utiliza animaciones para mostrar la revelación de casillas, la explosión de Voltorbs y los efectos visuales cuando el jugador gana o pierde.

## Conclusión

El mini-juego **Voltorb Flip** es una experiencia de lógica y memoria donde los jugadores deben evitar los Voltorbs y acumular puntos para ganar monedas. La interfaz visual se actualiza constantemente según las acciones del jugador, proporcionando una experiencia dinámica y envolvente.



# Items

La mayoría de ítems del juego proviene del fork de Pokémon essentials, pero hemos hecho la traducción de todas y agregado los siguientes:

# Repelente

|   |  |
|---|--|
| <p>Internal Name="REPEL"</p> <p>Item Name=Repelente</p> <p>Item Name Plural=Repelentes</p> <p>Pocket=Items</p> <p>Purchase price=350</p> <p>Description=Repele Pokémon salvajes débiles en un recorrido de 100;</p> <p>Use Out of Battle=Use directly</p> <p>Use In Battle=Can't Use</p> <p>Special Items=None of Below</p> <p>Machine=</p> | <p>Repelente</p> <p>Internal name that appears in constructs like PBIItems::XXX.</p> |
|---|--|

## Objetivo:

Implementa el uso de objetos repelentes que evitan encuentros con Pokémon salvajes durante una cierta cantidad de pasos.

## Archivos involucrados:

- Scripts de juego (eventos globales y manejadores de objetos)
- Datos de objetos definidos en la interfaz de Pokemon Essentials (GUI del editor)

## Función principal:

```
def pbRepel(item,steps)
  if $PokemonGlobal.repel > 0
    Kernel.pbMessage(_INTL("Pero todavía tiene efecto el último Repente utilizado."))
    return 0
  else
    Kernel.pbMessage(_INTL("{1} ha usado {2}.", $Trainer.name, PBIItems.getName(item)))
    $PokemonGlobal.repel = steps
    return 3
  end
end
```

## Descripción:

Esta función gestiona el uso del repelente. Comprueba si ya hay un efecto de repelente activo (\$PokemonGlobal.repel > 0).

Si lo hay, se muestra un mensaje y no se activa uno nuevo. Si no, se activa un nuevo efecto de repelente durante la cantidad de pasos especificados.

### Registro del objeto en ItemHandlers:

```
ItemHandlers::UseFromBag.add(:REPEL, proc{|item| pbRepel(item,100) })  
ItemHandlers::UseFromBag.add(:SUPERREPEL, proc{|item| pbRepel(item,200) })  
ItemHandlers::UseFromBag.add(:MAXREPEL, proc{|item| pbRepel(item,250) })
```

#### Descripción:

Cada tipo de repelente se registra en el manejador UseFromBag con su efecto correspondiente en pasos:

- REPEL: 100 pasos
- SUPERREPEL: 200 pasos
- MAXREPEL: 250 pasos

### Evento global - Uso del repelente en movimiento:

```
Events.onStepTaken += proc {  
  if !PBTerrain.isIce?($game_player.terrain_tag)  
    if $PokemonGlobal.repel > 0  
      $PokemonGlobal.repel -= 1  
    if $PokemonGlobal.repel <= 0  
      Kernel.pbMessage(_INTL("El Repelente dejó de tener efecto..."))  
      ret = pbChooseItemFromList(_INTL("¿Quieres utilizar otro Repelente?"), 1,  
        :REPEL, :SUPERREPEL, :MAXREPEL)  
      pbUseItem($PokemonBag, ret) if ret > 0  
    end  
  end  
end  
}
```

#### Descripción:

Este evento se ejecuta en cada paso dado por el jugador. Si hay un repelente activo y el terreno no es hielo (donde no se gasta el efecto), se reduce su contador. Al agotarse, se ofrece al jugador usar otro repelente automáticamente.

### Definición del objeto en la interfaz de Pokemon Essentials (GUI):

Objeto: REPEL

Nombre: Repelente

Nombre plural: Repelentes

Categoría: Pocket - Items

Precio de compra: 350

Descripción: Repele Pokémon salvajes débiles en un recorrido de 100

Uso fuera de batalla: Usar directamente

Uso en batalla: No se puede usar

Especial: Ninguno

Máquina: No

Nombre interno: REPEL (aparece como PBItems::REPEL en los scripts)

## Conclusión

El sistema de repelentes en Pokémon Essentials está diseñado para evitar encuentros aleatorios con Pokémon salvajes durante una cantidad limitada de pasos. Este comportamiento se gestiona mediante una variable global `$PokemonGlobal.repel`, la cual almacena la duración restante del efecto.

El método `pbRepel(item, steps)` se encarga de aplicar el efecto del repelente seleccionado, siempre y cuando no haya uno activo ya. Este sistema integra los tres tipos de repelentes disponibles (Repelente, Súper Repelente y Máx. Repelente) a través del uso de `ItemHandlers::UseFromBag`.

Adicionalmente, un evento enlazado a `Events.onStepTaken` controla la reducción del contador de pasos restantes y notifica al jugador cuando el efecto del repelente finaliza. Al hacerlo, también ofrece la opción de utilizar otro repelente de forma inmediata.

En conjunto, este sistema proporciona una mecánica clara y funcional que respeta las reglas del juego original, a la vez que permite fácil extensión o personalización en proyectos derivados.

# Pociones (en combate)

## Propósito del código:

Este código define cómo se comportan los objetos de curación cuando se usan sobre un Pokémon durante un combate. Utiliza un diccionario (heal\_map) para asignar valores de curación a distintos objetos. Esto permite simplificar el código y evitar múltiples condicionales.

## Explicación del comportamiento:

- `ItemHandlers::BattleUseOnPokemon.add(...)`  
Añade un manejador para el objeto :POTION (y por copia, otros similares) cuando se usa sobre un Pokémon en combate.
- `heal_map`:  
Diccionario que asocia cada objeto con la cantidad de PS que cura. Algunos valores dependen de si USENEWBATTEMECHANICS está activado (por ejemplo, Superpoción y Hiperpoción).

| Objeto       | PS curados                        |
|--------------|-----------------------------------|
| :POTION      | 20                                |
| :SUPERPOTION | 60 si mecánica nueva, 50 si no    |
| :HYPERPOTION | 120 si mecánica nueva, 200 si no  |
| :MAXPOTION   | Cura total (totalhp - hp)         |
| :FRESHWATER  | 30 si mecánica nueva, 50 si no    |
| :SODAPOP     | 50 si mecánica nueva, 60 si no    |
| :LEMONADE    | 70 si mecánica nueva, 80 si no    |
| :MOOMOOMILK  | 100                               |
| :ORANBERRY   | 10                                |
| :SITRUSBERRY | 1/4 de los PS totales del Pokémon |

- En caso de que el objeto no esté en el diccionario, se aplica una curación por defecto de 20 PS.
- La función pbBattleHPItem se encarga de aplicar la curación real en el sistema de batalla.

### **Compatibilidad extendida:**

El manejador del objeto :POTION se copia para funcionar también con los siguientes objetos usando:

ruby

CopiarEditar

ItemHandlers::BattleUseOnPokemon.copy(...)

Esto incluye:

- :SUPERPOTION
- :HYPERPOTION
- :MAXPOTION
- :BERRYJUICE
- :RAGECANDYBAR
- :SWEETHEART
- :FRESHWATER
- :SODAPOP
- :LEMONADE
- :MOOMOOMILK
- :ORANBERRY
- :SITRUSBERRY

### **Resumen rápido**

- Se define un solo manejador para objetos curativos usando un diccionario (heal\_map) que asigna cuánto cura cada ítem.
- Admite mecánicas clásicas y nuevas (con USENEWBATTLEMECHANICS).
- Si el objeto no está listado, cura 20 PS por defecto.
- Se aplica con pbBattleHPItem.
- El manejador de :POTION se copia para muchos otros ítems similares (como :SUPERPOTION, :MOOMOOMILK, etc.).

# Pociones (Fuera Combate)

**Resumen:** Este código maneja el efecto de objetos curativos cuando se usan directamente sobre un Pokémon desde el menú (fuera de combate). Está optimizado para reducir código redundante.

## Descripción General:

Se define un único manejador `ItemHandlers::UseOnPokemon` para `:POTION`, que cubre múltiples objetos curativos a través de un diccionario (`heal_map`) con la cantidad de PS a restaurar según el ítem utilizado.

## Detalles Técnicos:

- **Variable `heal_map`:**  
Mapea símbolos de objetos (`:POTION`, `:SUPERPOTION`, etc.) con sus valores de curación.
- **Compatibilidad con mecánicas nuevas:**  
Si `USENEWBATTLEMECHANICS` es `true`, se usan valores de curación actualizados (por ejemplo, `:SUPERPOTION` cura 60 en lugar de 50).
- **Valor por defecto:**  
Si el objeto no está en `heal_map`, cura 20 PS (comportamiento por defecto equivalente al de una Poción). Esto aplica a objetos como `:BERRYJUICE`, `:SWEETHEART`, etc.
- **Curación dinámica:**
  - `:MAXPOTION`: Cura toda la vida restante (`pokemon.totalhp - pokemon.hp`).
  - `:SITRUSBERRY`: Cura 1/4 de la vida máxima del Pokémon (`pokemon.totalhp / 4`).
- **Llamada final:**  
Se ejecuta `pbHPItem(pokemon, heal, scene)` para aplicar la curación.



## **Copia del manejador:**

El manejador definido para :POTION se copia a otros objetos equivalentes:

ruby

CopiarEditar

```
ItemHandlers::UseOnPokemon.copy(
```

```
  :POTION, :SUPERPOTION, :HYPERPOTION, :MAXPOTION,
```

```
  :BERRYJUICE, :RAGECANDYBAR, :SWEETHEART, :FRESHWATER,
```

```
  :SODAPOP, :LEMONADE, :MOOMOOMILK, :ORANBERRY, :SITRUSBERRY
```

```
)
```

## **Resumen rápido – Uso de pociones fuera de combate:**

El manejador UseOnPokemon permite usar objetos curativos desde el menú. Utiliza un mapa (heal\_map) para asignar cuánto cura cada objeto. Soporta mecánicas clásicas y nuevas (USENEWBATTLEMECHANICS), y si el objeto no está en el mapa, cura 20 PS por defecto. El mismo manejador se reutiliza para pociones, bebidas y bayas curativas. Todo se ejecuta mediante pbHPItem.

# Despertar

## Descripción:

Este manejador se utiliza cuando se emplea un AWAKENING fuera de combate en un Pokémon. El objeto se utiliza para curar el estado alterado de "Sueño" en un Pokémon, permitiendo que despierte si está dormido.

## Funcionamiento:

- Primero, se verifica si el Pokémon está dormido (`status == PBStatuses::SLEEP`) y tiene puntos de salud (HP) mayores a 0.
- Si el Pokémon no está dormido o tiene los PS a 0, se muestra un mensaje que indica que el uso del objeto no tendrá efecto y no hace nada.
- Si el Pokémon está dormido y tiene HP, el objeto lo despierta utilizando el método `healStatus`, que elimina el estado de sueño.
- Se refresca la escena con `scene.pbRefresh` para actualizar la visualización.
- Finalmente, se muestra el mensaje indicando que el Pokémon se ha despertado.

## Código:

ruby

CopiarEditar

```
ItemHandlers::UseOnPokemon.add(:AWAKENING,proc{|item,pokemon,scene|
  if pokemon.hp<=0 || pokemon.status!=PBStatuses::SLEEP
    scene.pbDisplay(_INTL("No tendrá ningún efecto."))
    next false
  else
    pokemon.healStatus
    scene.pbRefresh
    scene.pbDisplay(_INTL("{1} se ha despertado.",pokemon.name))
    next true
  end
})
```

## Resumen rápido:

El manejador de AWAKENING permite despertar a un Pokémon que está dormido, siempre que tenga HP y esté bajo el estado de sueño. Si no se cumplen estas condiciones, el objeto no tendrá ningún efecto.

# Revivir y Revivir Máximo

## Descripción:

Estos manejadores se utilizan para los objetos REVIVE y MAXREVIVE, que permiten revivir a un Pokémon que ha caído en combate. Dependiendo del objeto, los Pokémon revividos reciben una cantidad de HP diferente.

### REVIVE:

- Si el Pokémon tiene HP mayor a 0, el objeto no tendrá efecto y se mostrará un mensaje informando que no se puede usar.
- Si el Pokémon tiene HP igual a 0, el objeto lo revive restaurándole la mitad de su HP máximo (redondeado hacia abajo) y curando cualquier estado alterado.
- Se actualiza la escena con `scene.pbRefresh` para reflejar el cambio y se muestra el mensaje indicando que el Pokémon ha recuperado los PS.

### MAXREVIVE:

- Si el Pokémon tiene HP mayor a 0, el objeto no tendrá efecto y se muestra un mensaje que indica que no se puede usar.
- Si el Pokémon tiene HP igual a 0, el objeto lo revive restaurándole todos sus puntos de salud (curando completamente su HP) y curando cualquier estado alterado.
- Se actualiza la escena con `scene.pbRefresh` y se muestra el mensaje indicando que el Pokémon ha recuperado sus PS.

### **Código:**

ruby

CopiarEditar

```
ItemHandlers::UseOnPokemon.add(:REVIVE,proc{|item,pokemon,scene|
  if pokemon.hp>0
    scene.pbDisplay(_INTL("No tendrá ningún efecto."))
    next false
  else
    pokemon.hp=(pokemon.totalhp/2).floor
    pokemon.healStatus
    scene.pbRefresh
    scene.pbDisplay(_INTL("{1} ha recuperado los PS.",pokemon.name))
    next true
  end
})
```

```
ItemHandlers::UseOnPokemon.add(:MAXREVIVE,proc{|item,pokemon,scene|
  if pokemon.hp>0
    scene.pbDisplay(_INTL("No tendrá ningún efecto."))
    next false
  else
    pokemon.healHP
    pokemon.healStatus
    scene.pbRefresh
    scene.pbDisplay(_INTL("{1} ha recuperado los PS.",pokemon.name))
    next true
  end
})
```

### **Resumen rápido:**

Los manejadores para REVIVE y MAXREVIVE permiten revivir a un Pokémon con HP igual a 0. REVIVE restaura la mitad de los PS del Pokémon, mientras que MAXREVIVE lo revive con todos sus PS restaurados. Ambos objetos curan cualquier estado alterado y actualizan la escena para reflejar el cambio. Si el Pokémon no tiene HP igual a 0, el objeto no tiene efecto.

# Ether

## Descripción:

Este manejador se utiliza para el objeto ETHER, que permite restaurar los puntos de poder (PP) de un movimiento específico de un Pokémon durante el combate.

## Funcionamiento:

- Al usar el ETHER, el jugador es solicitado a elegir un movimiento del Pokémon para restaurar sus PP mediante scene.pbChooseMove.
- Si el jugador elige un movimiento válido (es decir, el índice del movimiento es mayor o igual a 0), el objeto intentará restaurar 10 PP al movimiento seleccionado usando pbRestorePP.
- Si la restauración de PP es exitosa (es decir, se restauran PP), se muestra el mensaje "Los PP han sido restaurados".
- Si no es posible restaurar los PP (por ejemplo, si no hay PP para restaurar), se muestra el mensaje "No tendrá ningún efecto".
- Si el jugador no elige un movimiento (es decir, si la selección es inválida), el objeto no tendrá efecto y se devuelve false.

## Código:

```
ruby
CopiarEditar
ItemHandlers::UseOnPokemon.add(:ETHER,proc{|item,pokemon,scene|
  move=scene.pbChooseMove(pokemon,_INTL("¿Qué movimiento recuperar?"))
  if move>=0
    if pbRestorePP(pokemon,move,10)==0
      scene.pbDisplay(_INTL("No tendrá ningún efecto."))
      next false
    else
      scene.pbDisplay(_INTL("Los PP han sido restaurados."))
      next true
    end
  end
end
next false
})
```

**Resumen rápido:**

El objeto ETHER permite restaurar 10 PP de un movimiento seleccionado por el jugador. Si el movimiento no tiene PP que restaurar o si el jugador no elige un movimiento válido, el objeto no tiene efecto. Si la restauración es exitosa, se muestra un mensaje confirmando la restauración de los PP.