CÓMO HICE

12 de abril de 2026

Francisco Álvarez Martínez

Contenido

1. El origen inesperado: un conejo, una jaula y demasiadas cacas

La idea no nació delante del ordenador, sino limpiando la jaula de Luna. Luna caga muchísimo (quien tenga conejos lo sabe) y mientras recogía bolitas pensé en la gente que los tiene sueltos por casa, como si fueran perrillos.

La conejita Luna, la real.

Nuestra casa, además, es muy simétrica. Tanto, que si pudieras verla en vista cenital parecería un mapa de Pac-Man: pasillos rectos, giros de 90 grados, habitaciones que se conectan como si fueran "pantallas".

Y ahí se encendió la chispa: **¿y si Luna fuese el origen de todas las bolitas que hay que recoger?**

En ese momento apareció el recogedor. No tenía nombre, ni forma, ni personalidad. Solo era "el que limpia". Pero ya había un concepto.

2. Cuando el mundo empezó a tomar forma

Pensando en los enemigos, me vino a la cabeza el ratón que se nos coló en casa el año pasado. Un ratón real, de los de verdad, que se paseó por el salón como si fuera suyo.

Y pensé: "Si Luna lo llega a pillar, igual lo revienta."

Ese pensamiento tan cotidiano, tan doméstico, fue el que abrió la puerta al lore. De repente, el ratón no era solo un enemigo: era parte de la historia. Y el recogedor dejó de ser un objeto para convertirse en Fermín, un robot con personalidad propia.

3. Del chiste al prototipo: cuando la idea demuestra que funciona

Una vez que tuve claro el concepto —Luna como origen de las bolitas, el ratón como enemigo, el recogedor como aliado— necesitaba comprobar si aquello podía sostener un juego real. Así que hice lo que hacemos todos los desarrolladores cuando una idea nos pica: monté un prototipo rápido.

En esa fase me apoyé mucho en herramientas de IA para acelerar la iteración. No buscaba arte final ni narrativa pulida: solo quería ver si la mecánica tenía ritmo, si el movimiento era divertido, si la idea podía crecer.

Y ahí pasó algo importante: el prototipo funcionaba sorprendentemente bien. Era simple, sí, pero tenía chispa. Tenía ese "algo" que te hace pensar: "Vale, aquí hay más de lo que parecía".

4. Construyendo un sistema de niveles que crece contigo

Con el prototipo validado, el siguiente paso fue crear un sistema de niveles que no fuera manual, sino automático y escalable. Quería que el juego pudiera generar:

- más pasillos.
- más rutas.
- más zonas de riesgo.
- más oportunidades de huida.

y, sobre todo, más enemigos.

Así nació el sistema de niveles progresivos: cada nivel no solo aumenta en tamaño, sino también en densidad de ratas, en velocidad, en agresividad y en presión sobre el jugador.

Ese crecimiento no es lineal: está diseñado para que el jugador sienta que domina el juego… **hasta que deja de dominarlo**. Y ahí empieza la magia de un arcade: la tensión, el ritmo, la sensación de "una más y lo dejo".

5. La dificultad como identidad

Desde el principio tuve claro que *Conejito Cagón* debía tener dos caras:

- Una amable, accesible, casi familiar.
- Y otra despiadada, pensada para jugadores que buscan un reto real.

Por eso diseñé tres niveles de entrada suaves, pero progresivos, que permiten entender la mecánica sin estrés:

- **MUY FÁCIL**
- **FÁCIL**
- **NORMAL**

Luego, un pequeño desafío para jugadores más avanzados:

- **DIFÍCIL**

Pero a partir de ahí, la curva sube:

- **MUY DIFÍCIL**
- **EXTREMO**

Estos modos no son un añadido artificial: son parte de la identidad del juego. Son la prueba de que, aunque el universo sea adorable, el gameplay es puro arcade clásico.

6. La música: cuando el juego te pide una banda sonora

Después de probar el prototipo y ver que la idea tenía fuerza, me pasó algo que siempre me ocurre cuando un proyecto empieza a respirar por sí mismo: **la música apareció antes que el propio juego**.

Arranqué **Ableton Live 11 Suite**, encendí mi inseparable **Roland Juno DS61**, y empecé a buscar el tono del universo que estaba naciendo.

No quería una música infantil ni caricaturesca. Tampoco algo oscuro. Buscaba ese punto intermedio entre lo adorable y lo frenético, entre lo doméstico y lo arcade. Entre Luna, Fermín y las ratas.

Y en cuanto encontré el primer motivo musical, lo supe: **Conejito Cagón ya tenía voz propia.**

Ese sonido (cierto toque de jazz pianístico con guitarra eléctrica, todo ello sintetizado por VSTs) marcó el camino del resto del desarrollo. La música se convirtió en una guía emocional para el juego, en una especie de brújula creativa que me decía cuándo algo encajaba y cuándo no.
De hecho, tuve tanto reparo en modificar los temas que necesitaba que no hice excesivas variaciones. Solo algunas modificaciones en la instrumentación o en los tonos, para darle voz a Fermín (en temas como *Yo limpiando, ella cagando*, o en el tema de la fase de bonus, que es claramente una tonadilla que describe a alguien que se siente cómodo y que se va a hinchar a comer a placer, a sabiendas de que nadie se lo va a impedir).

Fue el primer elemento del proyecto que sentí como “definitivo”, y lo desarrollé en la misma forma en la que había hecho el de mi otro juego, *BUJERACO*, antes: un tema de portada, un tema para los niveles primos distintos al 11, uno para los múltiplos de 2, uno para los múltiplos de 3, uno para los múltiplos de 4 (así evitas que suene más el de los múltiplos de 2, y ahora explico el sistema), uno para los múltiplos de 5, de 7, de 11 (tema de pantalla de BONUS) y de 13. Con eso se cubre todo el juego de una forma matemática.

7. Fermín: cuando un recogedor se convierte en personaje

Desde el principio tenía claro que necesitaba un personaje que recogiera las cacas. La mecánica lo pedía. Así que pensé en un recogedor… pero no en uno cualquiera. Me imaginé uno de esos que tienen tapa por encima, como los que se usan en algunos lugares públicos, para que no se escape nada. Era perfecto para el papel.

Pero un recogedor abstracto no funcionaba. Necesitaba algo con presencia, con identidad. Y cuando uno quiere humanizar un objeto, hay un recurso universal: **ponerle ojos**.

Hice un boceto muy básico: el recogedor, su tapa y el palo eran una sola pieza, pero la parte inferior se abría como una bandeja para recoger. Y al verlo en pantalla, me di cuenta de algo que no había planeado: **parecía una boca**.

Y ahí surgió una parte esencial del lore: si Fermín tenía boca, y Luna dejaba bolitas por todas partes… entonces Fermín tenía que estar profundamente enamorado para comerse todas las mierdas que Luna le echaba. Era absurdo, tierno y completamente lógico dentro del universo que estaba naciendo.

Luego vino un detalle práctico: el personaje tenía que entrar bien por los pasillos del juego. El palo del recogedor era demasiado largo. Así que lo "partí" hasta dejarlo como un trozo de recuerdo **de lo que Fermín fue una vez** y lo justifiqué dentro de la historia como la consecuencia de un encontronazo previo con las ratas, antes de los hechos del juego.

Ese palo roto terminó de definirlo. Fermín ya no era un objeto: era un robot humilde, golpeado por la vida, resignado a limpiar lo que Luna deja atrás, pero con un punto de heroicidad involuntaria que lo hace entrañable.

Y entonces llegó su nombre. Mientras probaba las primeras ratas y veía cómo Fermín tenía que esquivarlas, me di cuenta de que la mecánica se parecía muchísimo a la de un recortador taurino: fintas, quiebros, valentía y un punto de locura. Lo primero que me vino a la cabeza fue **San Fermín**, y en ese instante supe que ese tenía que ser su nombre.

8. Las ratas: del intruso real al enemigo perfecto

Si Luna era el origen de las bolitas y Fermín el encargado de recogerlas, faltaba una pieza esencial: el enemigo. Y ahí entraron las ratas.

Su papel era claro desde el principio: tenían que ser la amenaza constante, el elemento que rompiera la rutina doméstica y convirtiera el juego en un arcade frenético. La inspiración vino de un episodio real: el ratón que se nos coló en casa el año pasado. Un intruso pequeño, rápido y sorprendentemente valiente. Y pensé: si Luna lo llega a pillar, igual lo revienta. Ese pensamiento tan cotidiano fue suficiente para definir al enemigo.

Pero no quería una sola rata. Quería una presencia creciente, algo que diera la sensación de que el mundo se volvía más hostil a medida que avanzabas. Así que diseñé un sistema muy simple y muy efectivo: **cada 8 niveles aparece una rata nueva**. No solo aumenta el número: también aumenta la velocidad, de forma progresiva, casi imperceptible al principio, pero totalmente evidente cuando llevas un rato jugando.

La idea era que el jugador sintiera que domina la situación… **hasta que deja de dominarla**.

9. La dificultad: cuando las ratas marcan el tono

La cantidad de ratas iniciales y su velocidad dependen directamente del nivel de dificultad elegido. Quería que cada modo tuviera una identidad propia, no solo un ajuste numérico. Así quedó definido:

- **MUY FÁCIL** → 1 rata, velocidad muy baja
- **FÁCIL** → 1 rata, velocidad baja
- **NORMAL** → 1 rata, velocidad estándar
- **DIFÍCIL** → 3 ratas, velocidad normal
- **MUY DIFÍCIL** → 5 ratas, un poco más rápidas
- **EXTREMO** → 7 ratas iniciales, y van a toda leche

A partir de ahí, **cada 8 niveles se suma una rata más**, independientemente del modo. Y todas van acelerando poco a poco.

El resultado es un sistema que parece sencillo, pero que genera una tensión creciente muy característica: el jugador empieza tranquilo, pero el juego no tarda en recordarle que está en un arcade puro, donde la presión es parte de la diversión.

10. La historia y el mundo: un ciclo de 40 niveles y 8 zonas

Una vez definidos los personajes y la mecánica básica, necesitaba darle forma al mundo. No quería un arcade infinito sin estructura, pero tampoco un juego rígido con niveles aislados. La solución fue crear **ciclos de 40 niveles**, divididos en **8 zonas temáticas**, con **5 niveles cada una**.

Cada zona representa un lugar real del recorrido que hace Fermín siguiendo el rastro de Luna: la casa, el patio, el barrio, el bosque, el río, la granja, el veterinario y, finalmente, el Mundo Neón. Las siete primeras son lugares físicos y reconocibles, pero la última es otra cosa: una alucinación psicodélica provocada por el colapso de Fermín tras ingerir cantidades absurdas de caca. En el lore queda deliberadamente en el aire si es un lugar real o un estado mental.

El ciclo completo funciona como una especie de odisea doméstica que se desmadra: empieza en casa, en un entorno cotidiano, pero pronto se vuelve más salvaje y absurdo. Luna avanza sin control, Fermín intenta alcanzarla, y las ratas aparecen cada vez en mayor número y velocidad. Es una historia sencilla, casi ridícula, pero con una lógica interna muy clara: cuanto más avanza el jugador, más se descontrola el mundo.

Dentro de ese ciclo también hay momentos especiales: **las pantallas de bonus**, que aparecen en los niveles múltiplos de 11. Representan los episodios de alucinación digestiva de Fermín cuando come demasiado rastro: no hay ratas, no hay peligro, solo caca hasta el infinito. Son pausas temáticas que explicaré en detalle en otro apartado, porque tienen su propia música, su propio diseño y su propio sentido narrativo.

La estructura de 40 niveles no fue casual. Es lo suficientemente larga para sentir progresión, pero lo bastante compacta para que cada ciclo tenga un principio, un desarrollo y un final. Y, sobre todo, permite que el jugador entienda que está recorriendo un mundo que crece, se complica y se vuelve más hostil… que está recorriendo el mismo viaje que Fermín, un viaje que empieza siendo doméstico y termina en un delirio neón que cierra (o reinicia) la odisea.

11. La pantalla de bonus: la alucinación digestiva de Fermín

Dentro del ciclo de 40 niveles hay un tipo de pantalla muy especial: los niveles de bonus, que aparecen en los múltiplos de 11. No son un descanso tradicional, ni un premio clásico de arcade. Son algo mucho más… fisiológico.

En el lore, estas pantallas representan un estado muy concreto de Fermín: cuando come demasiado rastro seguido, su sistema se satura y entra en una alucinación digestiva. Es su CPU diciendo "hasta aquí hemos llegado".

En estas pantallas no hay ratas, no hay peligro y no hay límites. Solo caca. Caca hasta el infinito. Es un espacio donde Fermín puede moverse sin miedo, comer sin parar y dejarse llevar por un trance casi hipnótico. La música también cambia: es más cómoda, más despreocupada, casi como si Fermín estuviera disfrutando de un banquete absurdo que nadie puede interrumpir.

Visualmente, la pantalla también refleja ese estado alterado: los muros cambian de color en un barrido continuo, como si el mundo entero respirara al ritmo de la indigestión de Fermín. Ese efecto se consigue con un pequeño algoritmo que cicla colores en función del tiempo y de la posición de cada bloque:

```
Def dibujar_barrido_bonus(surf, off_x, off_y):
    colores = [
        (180, 40, 40),   # rojo
        (40, 180, 40),   # verde
        (200, 200, 40),  # amarillo
        (40, 40, 180),   # azul
    ]

    t = (pygame.time.get_ticks() // 120) % 4

    for f in range(FILAS):
        for c in range(COLS):

            # Solo muros (en el mapa: 1 = muro)
            if MAPA[f][c] != 1:
                continue

            px = c * TILE_SIZE - off_x
            py = f * TILE_SIZE - off_y

            if px < -TILE_SIZE or px > ANCHO_VISIBLE:
                continue
            if py < -TILE_SIZE or py > ALTO_VISIBLE:
                continue

            idx = (f + c + t) % 4
            color = colores[idx]

            # RELLENO del bloque, dejando el bisel sin tapar
            margen = 3
            pygame.draw.rect(
                surf,
                color,
                (px + margen, py + margen,
                 TILE_SIZE - 2*margen, TILE_SIZE - 2*margen)
            )
```

Antes de seguir, conviene explicar un elemento clave del diseño: **el aliviadero**. En todos los niveles normales, el aliviadero es la zona central del mapa que siempre está limpia. No tiene una forma fija: en los primeros niveles suele ser más bien cuadrado, luego se convierte en una especie de rombo y, en los niveles más avanzados, se aproxima a un círculo. Su tamaño también crece ligeramente con el nivel, pero sin llegar a ocupar todo el escenario. Siempre está en el corazón del mapa y cumple la misma función: es el espacio seguro donde Fermín puede maniobrar sin pisar caca y, sobre todo, sin que las ratas lo acorralen. Es su refugio táctico, su única garantía de orden dentro del caos creciente.

Y por eso, en las pantallas de bonus, su alteración es tan significativa. En nivel de bonus, hasta el aliviadero, que en los niveles normales está despejado para que Fermín pueda zafarse de las ratas, aparece completamente cubierto de caca. Es el único momento del juego en el que no existe ningún espacio seguro: todo está ahí para que Fermín coma hasta hartarse, como bien se indica al comienzo del nivel.

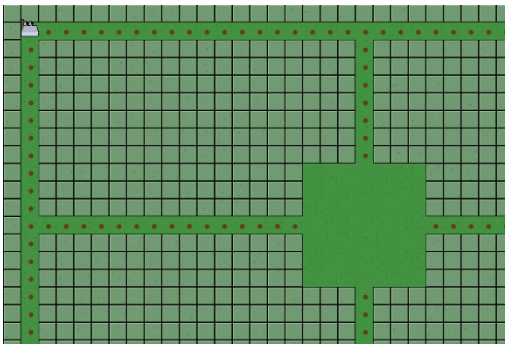
Pero este estado no dura para siempre. Cuando se pasa de rosca, Fermín suelta un regüeldo que hace temblar la pantalla (literalmente) y suena demasiado real (siendo procedural). Ese eructo monumental lo devuelve a la realidad: la alucinación se disipa, el mundo vuelve a encajar y Fermín retoma su misión como si nada hubiera ocurrido. Es un momento cómico, extraño y completamente coherente con el universo del juego.

A nivel de diseño, las pantallas de bonus funcionan como un respiro rítmico dentro del ciclo: rompen la tensión creciente, permiten al jugador relajarse un instante y prepararse para lo que viene después. Pero también refuerzan la idea central del juego: Fermín está tan enamorado de Luna que es capaz de comerse lo que haga falta… incluso cuando su propio sistema empieza a alucinar.

12. Las 8 zonas (+bonus) una por una

El viaje de Fermín no es solo una excusa narrativa: es un recorrido real por ocho escenarios distintos, cada uno con su estética, su dificultad y sus pequeñas rarezas técnicas. Cada zona tiene su personalidad, su ritmo y su propia forma de poner a prueba la mandíbula de titanio de nuestro protagonista.

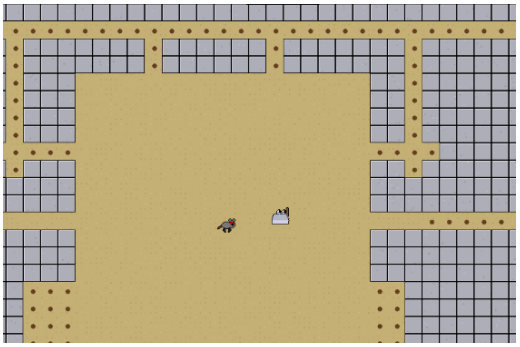
12.1. La Casa



Aquí empieza todo. Fermín aprende a limpiar el rastro de cacas que deja Luna y se encuentra una rata por primera vez, teniendo que aprender a esquivarla.

Los pasillos son estrechos, el aliviadero es pequeño y la zona debería ser sencilla… pero no lo es. Es el tutorial emocional del juego: aquí descubres que Fermín no está preparado para lo que viene, pero lo va a intentar igual.

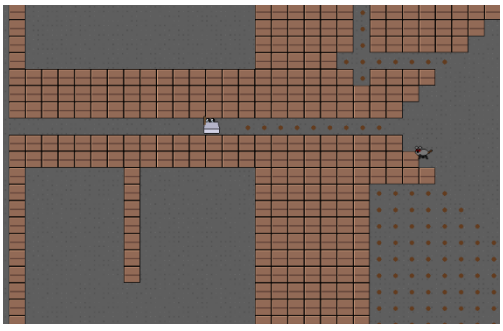
12.2. El Patio



Albero, baldosas y los primeros signos del apocalipsis marrón. El escenario crece y las estrategias para esquivar a las ratas empiezan a aparecer.

El aliviadero aquí es más grande, lo que da un respiro… pero también invita a confiarse. Es la primera zona donde el jugador siente que "ya controla", justo antes de que el juego empiece a apretar.

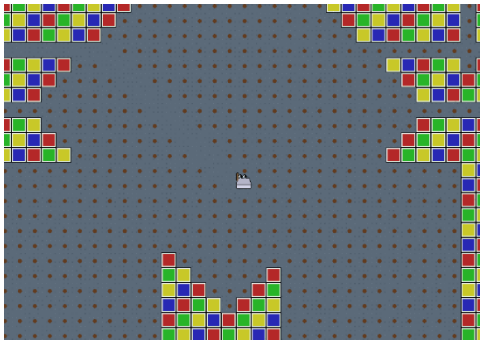
12.3. La Calle



Zurullos por todas partes. Las ratas vienen en grupos y no tienen piedad. Las pantallas empiezan a ser más complejas y el jugador entiende que el viaje no va a ser lineal ni amable.

Es la primera zona donde el caos se vuelve urbano: bordillos, aceras, esquinas traicioneras. Aquí Fermín ya no está en casa: está en el mundo real.

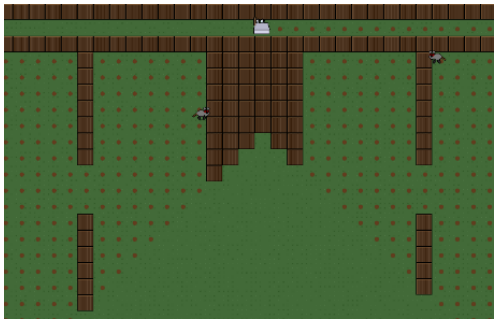
12.4. Bonus (múltiplos de 11)



Cuando Fermín se satura, entra en una alucinación digestiva. No hay ratas. Solo caca infinita. Hasta el aliviadero está repleto.

Es el único momento del juego en el que no existe ningún espacio seguro: todo está ahí para que Fermín coma hasta hartarse. Y cuando se pasa de rosca, suelta un regüeldo que hace temblar la pantalla y suena demasiado real (siendo procedural).

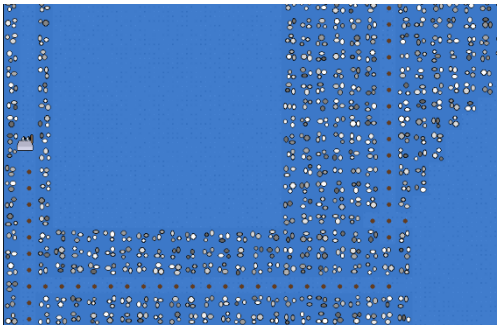
12.5. El Bosque



Raíces, sombras y ratas más rápidas. El aliviadero es tu mejor aliado… si consigues llegar a él.

Aquí el mapa deja de ser geométrico: los muros se sustituyen por árboles, lo que da una sensación de laberinto orgánico. Es la primera zona donde el jugador siente que el entorno también es un enemigo.

12.6. El Río



Terreno resbaladizo, agua dudosa y ratas que aparecen desde cualquier dirección. El escenario se vuelve caótico.

Pero lo más especial del Río es algo que parece invisible: **cada baldosa es única**. Las piedras no están dibujadas a mano ni copiadas: se generan proceduralmente, tres por baldosa, sin solaparse, con variaciones fuertes de tono y con un borde negro para mantener la legibilidad.

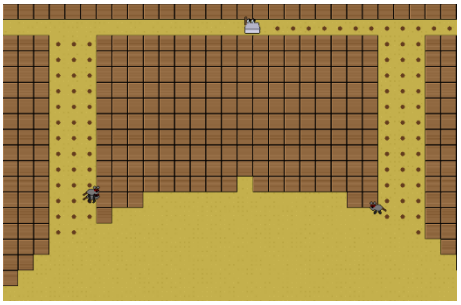
El algoritmo es así de elegante:

```
def generar_tile_rio():
    tile = pygame.Surface((TILE_SIZE, TILE_SIZE), pygame.SRCALPHA)

    # Agua EXACTAMENTE igual que el camino
    base = TEMAS[4]["suelo"]   # o tema["suelo"] si pasas el tema
    tile.fill(base)

    margen = TILE_SIZE // 6

    # Colores claramente distintos
    colores_base = [
        (245, 245, 245),  # clara
        (200, 200, 200),  # media
        (150, 150, 150),  # oscura
```

```
    ]

    piedras = []
    intentos = 0

    # EXACTAMENTE 3 piedras
    while len(piedras) < 3 and intentos < 200:
        intentos += 1

        rx = random.randint(TILE_SIZE // 5, TILE_SIZE // 3)
        ry = random.randint(TILE_SIZE // 5, TILE_SIZE // 3)

        cx = random.randint(margen, TILE_SIZE - rx - margen)
        cy = random.randint(margen, TILE_SIZE - ry - margen)

        rect = pygame.Rect(cx - 1, cy - 1, rx + 2, ry + 2)

        # Evitar solapamientos
        if any(rect.colliderect(r) for r in piedras):
            continue

        piedras.append(rect)

        base_color = random.choice(colores_base)

        # Variación fuerte
        tono = random.randint(-50, 50)
        col = (
            max(0, min(base_color[0] + tono, 255)),
            max(0, min(base_color[1] + tono, 255)),
            max(0, min(base_color[2] + tono, 255)),
        )

        # Borde negro
        pygame.draw.ellipse(tile, (0, 0, 0), rect)

        # Piedra real
        pygame.draw.ellipse(tile, col, (cx, cy, rx, ry))

    return tile
```

El resultado es un río que nunca se repite, que parece sencillo pero está lleno de pequeños detalles para darle un mejor cuerpo visual y que el área no parezca un conjunto de bloques más.

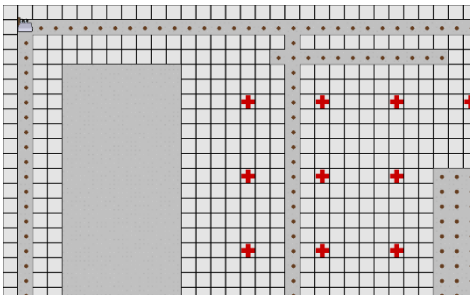
12.7. La Granja



Luna demuestra que puede cagar por encima de sus posibilidades. La granja está infestada de ratas y Fermín tiene una misión titánica que hará sufrir a su mandíbula de titanio.

Es una zona cálida, amarilla, sucia, donde la paja y la madera crean un contraste perfecto con el marrón omnipresente.

12.8. El Veterinario

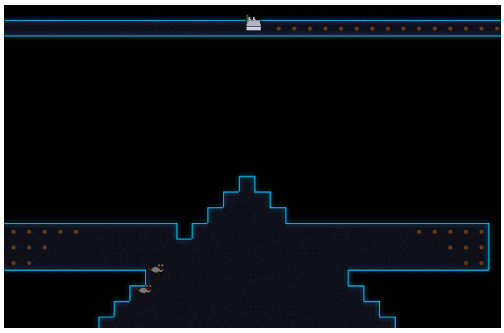


Fermín consigue llevar a Luna al veterinario, porque su problema no es normal. Este debería ser un lugar limpio… pero la pobre Luna no puede contenerse.

Es la zona más clínica, más blanca, más “civilizada”… y por eso mismo la más irónica. El caos marrón en un entorno estéril es casi ofensivo, y por eso funciona tan bien.

12.9. Mundo Neón

La CPU de Fermín colapsa y es llevado al taller de reparaciones. Mientras está siendo reparado, Fermín tiene su alucinación psicodélica final.



El Mundo Neón es un escenario precioso: líneas de luz, contornos brillantes, estética de circuito, un ambiente que parece respirar. Los muros no son muros: son bordes luminosos con glow multicapa.

El efecto se consigue así:

```
def dibujar_muro_neon(surf, mapa, f, c, x, y):
    if mapa[f][c] != 0:
        return

    rows = len(mapa)
    cols = len(mapa[0])

    color = (0, 200, 255)
    grosor = 2

    x0 = x
    y0 = y
    x1 = x + TILE_SIZE
    y1 = y + TILE_SIZE

    U = (f > 0          and mapa[f-1][c] == 1)
    D = (f < rows-1     and mapa[f+1][c] == 1)
    L = (c > 0          and mapa[f][c-1] == 1)
    R = (c < cols-1     and mapa[f][c+1] == 1)

    if U:
        draw_glow_line(surf, color, (x0, y0), (x1, y0), grosor)

    if D:
        draw_glow_line(surf, color, (x0, y1), (x1, y1), grosor)

    if L:
        draw_glow_line(surf, color, (x0, y0), (x0, y1), grosor)

    if R:
        draw_glow_line(surf, color, (x1, y0), (x1, y1), grosor)
```

Y el glow:

```
def draw_glow_line(surf, color, p1, p2, base_width):
    # Calcular bounding box de la línea
    x1, y1 = p1
    x2, y2 = p2

    min_x = min(x1, x2)
    min_y = min(y1, y2)
    max_x = max(x1, x2)
    max_y = max(y1, y2)

    # Margen para el glow
    margen = base_width + 20

    w = max_x - min_x + margen * 2
    h = max_y - min_y + margen * 2

    # Superficie pequeña
    glow_surf = pygame.Surface((w, h), pygame.SRCALPHA)

    # Coordenadas locales
    lp1 = (x1 - min_x + margen, y1 - min_y + margen)
    lp2 = (x2 - min_x + margen, y2 - min_y + margen)

    # Capas del glow
    glow_colors = [
        (*color, 40),
        (*color, 25),
        (*color, 15)
    ]
    widths = [base_width + 6, base_width + 10, base_width + 14]

    for gc, w2 in zip(glow_colors, widths):
        pygame.draw.line(glow_surf, gc, lp1, lp2, w2)

    # Línea principal
    pygame.draw.line(glow_surf, color, lp1, lp2, base_width)

    # Blit final
    surf.blit(glow_surf, (min_x - margen, min_y - margen))
```

El resultado es un escenario que parece salido de una recreativa de los 80 mezclada con una alucinación digital. Es el cierre perfecto antes de reiniciar el ciclo.

13. La zanahoria: el modo berserker de Luna

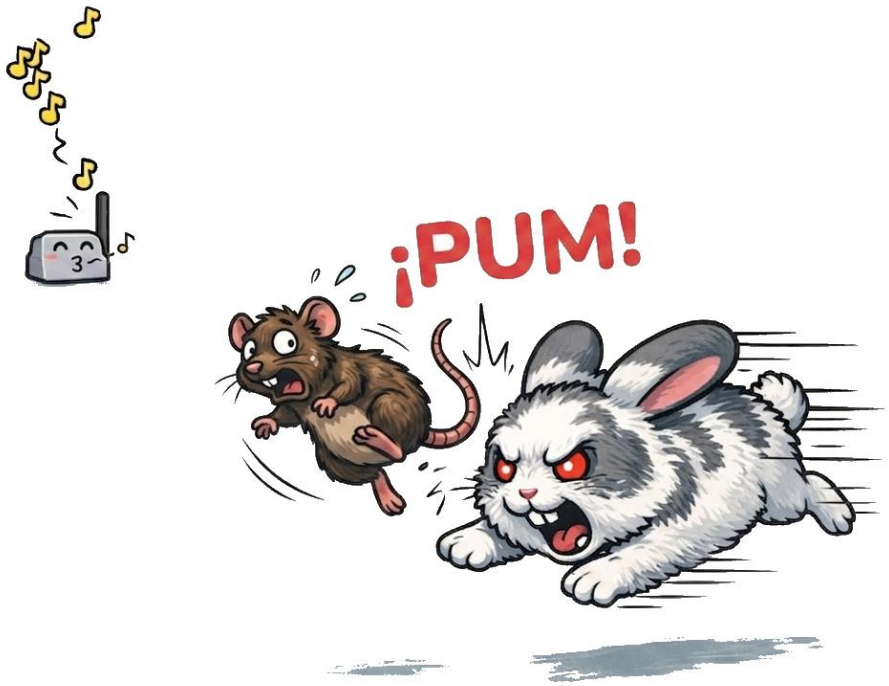
Cada cinco niveles aparece una zanahoria en el centro del aliviadero. A simple vista parece un objeto simpático, un guiño inocente… pero no. Es un arma de destrucción masiva emocional.

Porque Fermín conoce perfectamente a Luna. Sabe cómo es. Sabe lo que significa esa zanahoria. Y, sobre todo, sabe lo que va a pasar si se la come.

Así que, como es un poco sinvergüenza, Fermín se acerca, se la zampa sin dudarlo y acto seguido disimula silbando, mirando hacia otro lado como quien no quiere la cosa. Es uno de los gags más potentes del juego: ese silbidito culpable que dice “yo no he sido” mientras él mismo se limpia la boca.

Y entonces aparece Luna, con los ojos inyectados en sangre. Convencida, como siempre, de que han sido las ratas quienes le han robado su zanahoria.

Lo que ocurre a continuación es una mezcla de venganza, ternura y violencia adorable: Luna entra en modo berserker y se lanza contra todas las ratas del nivel, reventándolas a topetazos mientras Fermín sigue silbando, fingiendo que no tiene nada que ver con el asunto.



A nivel de diseño, la zanahoria cumple varias funciones:

- Rompe el ritmo sin convertirse en una pantalla de bonus.
- Da un respiro emocional al jugador, que tendrá que llegar, primero, al centro del aliviadero para comérsela.
- Refuerza la relación entre Fermín y Luna, que es tan cómica como disfuncional.
- Introduce un momento de poder que no depende del jugador, sino del carácter de Luna.

Y añade un gag recurrente que se vuelve parte de la identidad del juego.

Es un detalle pequeño, sí, pero define a los personajes mejor que cualquier cinemática: Fermín es un pillo adorable, Luna es un torbellino emocional, y las ratas… bueno, las ratas pagan el pato.

14. El sonido procedural: Fermín no silba… Fermín genera música

El sonido en *Conejito Cagón* no intenta ser realista. Intenta ser coherente con Fermín, que es un robot con más corazón que sentido común. El ejemplo perfecto es su silbido.

Cuando Fermín se come la zanahoria (a propósito, porque es un sinvergüenza) no silba un "fiu fiu" grabado. No. **Fermín genera un silbido procedural pentatónico**, como si su CPU estuviera componiendo una melodía improvisada mientras finge inocencia.

El silbido se construye nota a nota:

- usando una escala pentatónica mayor, que siempre suena bien
- con vibrato para darle vida
- con movimiento suave entre notas (−1, 0, +1)
- con frecuencias reales (C, D, E, G, A)
- y con una duración exacta de semicorchea a 120 BPM

Es decir: **Fermín no silba. Fermín compone.**

El corazón del sistema es este:

```
# ================================
#  SILBIDO PROCEDURAL PENTATÓNICO
# ================================

# Escala pentatónica mayor (C mayor)
PENTA = [
    261.63,  # C
    293.66,  # D
    329.63,  # E
    392.00,  # G
    440.00   # A
]

# Octava superior para variedad
PENTA_EXT = PENTA + [f * 2 for f in PENTA]

# Duración de semicorchea a 120 BPM
DUR = 0.125   # 1/8 de segundo
TEMPO_DELAY = int(DUR * 1000)
```

Y la generación de cada nota:

```
def generar_seno(freq, dur=DUR, vol=0.35):
    sr = 44100
    n = int(sr * dur)
    buf = bytearray()

    vibrato_freq = 6.0     # Hz
    vibrato_depth = 0.015  # 1.5%

    for i in range(n):
        t = i / sr

        # Vibrato: variación ligera de frecuencia
        freq_mod = freq * (1 + vibrato_depth * math.sin(2 * math.pi * vibrato_freq
* t))

        s = vol * math.sin(2 * math.pi * freq_mod * t)
        v = int(s * 32767)
        buf += v.to_bytes(2, byteorder='little', signed=True)
```

Y la melodía completa:

```
def generar_melodia(n_notas=10):
    melodia = []
    idx = random.randint(0, len(PENTA_EXT) - 1)

    for _ in range(n_notas):
        melodia.append(PENTA_EXT[idx])

        # Movimiento suave: -1, 0 o +1
        idx += random.choice([-1, 0, 1])
        idx = max(0, min(idx, len(PENTA_EXT) - 1))

    return melodia
```

Y la reproducción asíncrona del silbido:

```
def reproducir_silbido_async():
    global silbido_activo, canal_silbido

    if not silbido_activo:
        return

    def loop():
        global silbido_activo, canal_silbido

        while silbido_activo:
            melodia = generar_melodia()
            for freq in melodia:
                if not silbido_activo:
                    break
                snd = generar_seno(freq)
                canal_silbido.play(snd)
```

```
            # NO usar pygame.time.delay
            # Usar un sleep que permita interrupción
            for _ in range(TEMPO_DELAY // 5):
                if not silbido_activo:
                    break
                time.sleep(0.005)

    canal_silbido.stop()

threading.Thread(target=loop).start()
```

El resultado es un silbido que:

- Nunca es igual.
- Siempre suena musical.
- Tiene personalidad.

y encaja perfecto con el gag de la zanahoria.

Mientras Luna entra en modo berserker, Fermín está ahí, componiendo jazz pentatónico como si no fuera con él.

15. El ciclo de 40 niveles: un viaje que se repite… pero el mundo crece

El ciclo de *Conejito Cagón* no consiste en repetir los mismos niveles una y otra vez. Consiste en repetir la misma estructura, pero en un mundo que **crece físicamente** a medida que avanzas.

La progresión no altera la frecuencia de aparición de las ratas ni su lógica interna: lo que cambia es **el tamaño del escenario**.

Y ese cambio lo altera todo.

La función que determina el tamaño del mapa es esta:

```
def get_map_size(nivel):
    if 1 <= nivel <= 5:
        return 40, 24
    elif 6 <= nivel <= 10:
        return 60, 24
    elif 11 <= nivel <= 24:
        return 80, 48
    elif 25 <= nivel <= 49:
        return 100, 64
    elif 50 <= nivel <= 74:
        return 120, 80
    else:
        base = 4 + (nivel - 75) // 24
        return 32 * base, 24 * base
```

Esto significa que:

- La Casa siempre es la Casa… pero en la segunda vuelta es más grande.
- El Patio siempre es el Patio… pero hay más espacio para cagar y más espacio para morir.
- La Calle siempre es la Calle… pero ahora es una avenida.
- El Bosque siempre es el Bosque… pero ahora es un bosque de verdad.
- El Río siempre es el Río… pero ahora parece un delta.
- La Granja siempre es la Granja… pero ahora es una explotación industrial.
- El Veterinario siempre es el Veterinario… pero ahora es un hospital entero.
- El Mundo Neón siempre es el Mundo Neón… pero ahora es un circuito psicodélico gigantesco.

El jugador reconoce la estética, pero **no reconoce el espacio**. Y eso genera una sensación muy particular: familiaridad + desorientación.

A nivel de diseño, este crecimiento progresivo aporta:

- Más distancia entre puntos clave (aliviadero, zanahoria, ratas).
- Más tiempo para reaccionar… pero también más tiempo para equivocarte.
- Más espacio para que las ratas te encierren.
- Más recorrido para Fermín.
- Más caos visual.
- Más épica.

Y lo más importante: cada vuelta del ciclo es más grande que la anterior, pero **no más injusta**. La dificultad no sube por trucos baratos: sube porque el mundo se expande.
Es un viaje que se repite, sí… pero **no vuelves al mismo sitio**. Vuelves a una versión más amplia, más abierta y más peligrosa del mismo sitio.

El ciclo no es un bucle. Es una **espiral ascendente**.

¿Hasta dónde puede crecer el mundo?

Con esta progresión matemática, el mapa no tiene un límite teórico, pero sí uno práctico: la potencia del equipo donde se ejecuta. Cada pantalla del juego ocupa 32×24 tiles, un tamaño que tampoco es casual: es un homenaje directo a mis orígenes como programador en MSX, donde muchas aventuras se construían exactamente con esa proporción.

Por eso, cuando el mundo crece, no lo hace tile a tile, sino **pantalla a pantalla**, expandiéndose como lo haría un mapa de 8 bits que se estira más allá de lo que debería.

En un PC normalito de hoy, un portátil con un procesador medio, *Conejito Cagón* puede mover sin despeinarse mapas enormes, del orden de **8 pantallas de ancho** por **8 pantallas de alto** (es decir, unos **256 × 192 tiles**), que corresponden aproximadamente al **nivel 200**.

Eso significa que, aunque la estética se repita, el mundo no deja de expandirse. La Casa acaba siendo casi un palacio, el Patio parece una plaza de toros, la Calle se convierte en una avenida interminable y el Mundo Neón se vuelve un circuito psicodélico gigantesco.

El viaje es el mismo, sí… pero cada vuelta ocurre en un escenario más grande, más abierto y más desorientador.

16. Reflexión final: el juego que no sabía que necesitaba hacer

Conejito Cagón empezó como una broma, pero mientras lo hacía me di cuenta de que estaba construyendo algo que llevaba dentro desde hace mucho tiempo. No era solo un juego: era una forma de volver a mis orígenes, a los mapas cuadriculados del MSX, a las tardes de inventar mundos imposibles, a esa sensación de que programar era abrir una puerta secreta.

Y también nació de algo mucho más cotidiano: de Luna.

Luna no sabe que existe este juego. No sabe que cada nivel, cada persecución absurda, cada gag, nace de observarla, de cuidarla, de reírme con ella. Pero está en todas partes. En su forma de moverse. En su forma de mirar. En su forma de convertir lo doméstico en épico sin proponérselo.
Y luego está Fermín. Fermín, que sin quererlo, acabó siendo un reflejo de mí mismo: un robot que intenta poner orden donde no lo hay, que se rompe, que se recompone, que sigue adelante aunque el mundo crezca más de lo razonable. Un robot que silba para disimular cuando mete la pata. Un robot que persigue algo que quiere, aunque nunca llegue del todo.

Y sí, también un robot que se pasa el juego comiendo mierda. Porque la vida es así: a veces estás haciendo algo precioso, íntimo, significativo… y al mismo tiempo estás recogiendo lo que otros van dejando por el camino. Y sigues. Porque toca. Porque quieres. Porque forma parte del viaje.

Mientras hacía este juego, entendí que no estaba programando un arcade. Estaba programando un recuerdo. Un cariño. Una etapa de mi vida. Una forma de decir “esto soy yo ahora”, usando ratas, cacas, zanahorias y mundos que se expanden como si tuvieran vida propia.

No sé si este juego gustará a mucha gente. No sé si será grande, pequeño, raro o simplemente mío. Pero sé que es honesto. Sé que tiene alma. Sé que está hecho con una mezcla de humor, técnica y afecto que no se puede fingir.

Y si alguien lo juega y siente, aunque sea por un segundo, que detrás de todo esto hay una historia real… entonces habrá valido la pena.

Muchas gracias por su atención.

Descargas

Puedes descargar *Conejito Cagón* desde la web oficial:
https://conejitocagon.es

Disponible para:

- Windows 7 o superior
- Linux (kernel 4.15 LTS o superior, glibc 2.28 o superior)

Nota importante para Linux: *Conejito Cagón* requiere glibc ≥ 2.28. Esto significa que distribuciones como Ubuntu 20.04, Debian 10, Linux Mint 20 o superiores son compatibles. Distribuciones más antiguas (como Ubuntu 18.04) no podrán ejecutar el juego.

La versión gratuita incluye los primeros 5 niveles. El resto se desbloquea mediante licencia.

Propiedad intelectual y registros

Conejito Cagón y todo su contenido narrativo, gráfico, musical y jugable están protegidos por registro de propiedad intelectual.

SAFE Creative

- Videojuego (código, mecánicas, narrativa y manual): **2604105236562**
- Versión PC registrada como prueba tecnológica: **2603275095894**
- Banda sonora original: **2603184980601**

SGAE (RGO) – Obras musicales registradas Incluye todos los temas de la banda sonora original.

© Derechos y Aviso Legal