

Projecte | Core-Debt



Juan Felix Garcia Amills

ÍNDEX

Documentació Tècnica del Projecte: Core Debt	3
1. Definició i Objectius	3
2. Arquitectura Tècnica i Desenvolupament	3
3. Mecàniques de Control i Interacció	3
4. Gestió de Dades i Inventari	4
5. Sistema de Progressió i Bosses	4
6. Hosting i Desplegament (GitHub Pages)	4
7. Control de Qualitat (QA)	5

Documentació Tècnica del Projecte: Core Debt

1. Definició i Objectius

Core Debt és un videojoc de supervivència i gestió de recursos en 2D. El projecte se centra en l'optimització del rendiment en entorns web i la integració de mecàniques de control híbrides (ratolí + escriptura). L'objectiu tècnic és oferir una experiència de joc fluida directament al navegador, on el jugador ha de satisfer les demandes del sistema "Core" fins a arribar al nivell final 67.

2. Arquitectura Tècnica i Desenvolupament

- **Motor Gràfic:** Basat completament en l'API de **HTML5 Canvas**. S'ha implementat una correcció de **DPI** per garantir nitidesa en pantalles d'alta resolució.
- **Sincronització de Frames (Delta Time):** El motor utilitza un sistema de *Delta Time* per calcular el temps real entre frames. Això garanteix que la velocitat del joc sigui idèntica en monitors de **60Hz, 144Hz o 240Hz**, evitant que el joc s'acceleri en pantalles d'alt refresc.
- **Interfície d'Usuari (HUD):** Dissenyada mitjançant capes **HTML/CSS** independents del Canvas. Aquesta separació permet una manipulació del DOM més eficient per a alertes visuals i menús sense carregar el renderitzat gràfic.
- **Entorn de Desenvolupament:** Programat íntegrament amb **NeoVim** en un entorn Linux/Unix, prioritzant un codi net i sense dependències externes (*Vanilla JavaScript*).

3. Mecàniques de Control i Interacció

El joc requereix una alta coordinació motora mitjançant tres eixos d'interacció:

- **Navegació Espacial:** Desplaçament de càmera mitjançant *drag* amb el **botó dret** del ratolí i control de visibilitat amb la roda de **zoom**.
- **Recol·lecció i Combat:** Interacció directa amb objectius mitjançant el **clic esquerre**.
- **Input d'Esctura:** La selecció de millores (*Upgrades*) i certs minijocs de caps requereixen teclejar paraules específiques. Aquesta mecànica s'integra directament en el bucle lògic per augmentar el frenetisme.

4. Gestió de Dades i Inventari

- **Estat del Joc:** Tota la lògica és *Client-Side*. Per evitar la dependència de bases de dades, s'ha implementat un sistema de **Persistència de Dades** mitjançant fitxers **JSON**. El jugador pot exportar i importar el seu progrés localment.
- **Gestió de Càrrega:** El sistema controla el pes de l'inventari. S'ha inclòs una funció de buidat ràpid mitjançant la **paperera** (clic dret) per permetre una gestió estratègica dels recursos.

5. Sistema de Progressió i Bosses

La corba de dificultat es gestiona mitjançant esdeveniments cada 5 nivells:

- **Nivell 5-35:** Alternança entre Minions (clics) i Bosses especials (Punteria, Escriptura i Direccions).
- **Nivell 40+:** Introducció de **Bosses Híbrids**, que combinen múltiples minijocs de forma aleatòria en una sola fase.
- **Nivell 67 (Endgame):** Enfrontament contra la deïtat final per completar el "run".

6. Hosting i Desplegament (GitHub Pages)

Per a la distribució del projecte s'ha utilitzat **GitHub Pages** com a plataforma de *hosting* estàtic.

- **CI/CD:** El desplegament és automàtic; cada actualització al repositori es publica instantàniament.
- **Accessibilitat:** El joc és accessible globalment mitjançant l'URL del repositori, aprofitant la infraestructura de servidors de GitHub per garantir un temps de càrrega mínim i alta disponibilitat.
- **Seguretat:** El lloc s'executa sota protocol **HTTPS** forçat.

7. Control de Qualitat (QA)

S'ha desenvolupat un entorn de **Tests Unitaris** a la carpeta **/Tests**.

- S'avaluen 9 punts crítics: salut, inventari, càlcul de XP, probabilitats de *Luck* i integritat dels *Bosses*.
- **Eines de Desenvolupament:** S'inclou un menú de *Dev Tools* intern per a proves de càrrega i testeig ràpid de fases avançades.

