🎮 LIFE - MASTER PROMPTS (OFFLINE ARCHITECTURE - ULTIMATE DIRECTOR'S CUT)

Ces prompts sont conçus pour générer un code complet, robuste et optimisé pour une exécution locale sans appel API externe. Cette version étendue couvre les cas limites, l'optimisation mémoire avancée, les nuances de gameplay et une architecture logicielle de niveau industriel.

🏗️ ÉTAPE 1 : CORE ENGINE & ECS (Architecture & Performance)

🎯 **Modèle cible :** `Claude 4.6 Opus Thinking` **Objectif :** Créer un moteur capable de gérer 5000 entités sur un GPU intégré (Lenovo) avec zéro Garbage Collection en runtime et une fluidité parfaite (Frame Pacing).

COPIER CE PROMPT :

```
Agis en tant que Lead Engine Architect et Expert WebGL/WebGPU. Tu as 20 ans d'exp

PROJET : "LIFE" - Open World RPG Web (Offline Mode).
STACK : React Three Fiber (R3F), TypeScript, Rapier.js (Physique), Zustand (State

TA MISSION :
Construire l'architecture centrale "Core" du jeu. Je ne veux pas de code spaghett

1. ARCHITECTURE ECS (ENTITY COMPONENT SYSTEM) AVANCEE & TYPE-SAFE
   - Mets en place le World ECS avec une gestion stricte des archétypes (Archetyp
   - Définis les Components stricts en TypeScript (utilisant des `Float32Array` e
     - `TransformComponent` : Position (Vec3), Rotation (Quat), Scale (Vec3), Par
     - `RigidBodyComponent` : Configuration Rapier (Mass, Friction, Restitution,
     - `RenderComponent` : ModelRef (InstancedMesh ID), LODLevel, Visible, CastSh
     - `CharacterStats` : Structure complexe incluant HP, Hunger, Stamina, Stress
     - `InputState` : Buffer circulaire d'entrées pour la prédiction (MoveDir, Ai
     - `AIState` : Machine à état finie (FSM) : CurrentBehavior, TargetEntityID,
     - `TagComponent` : Bitmask pour les flags rapides (IsPlayer, IsEnemy, IsInte
   - Crée le "System Manager" qui boucle sur les entités avec filtrage par archét

2. GESTION MÉMOIRE & OBJECT POOLING (CRITIQUE)
   - Le Garbage Collector est l'ennemi mortel du 60fps. Implémente un système d'`
   - Au lieu de `new Vector3()`, utilise un pool statique de vecteurs réutilisabl
   - Pour les entités dynamiques (PNJ, Balles, Particules, Débris), pré-alloue le
   - Implémente une stratégie de "Soft Delete" : Quand une entité meurt, elle est
   - Surveille la mémoire : Ajoute un utilitaire de debug pour tracker la taille

3. BOUCLE DE JEU (GAME LOOP) & THREADING (PARALLÉLISME)
   - Implémente une boucle de jeu découplée (Decoupled Game Loop) robuste :
     - `Time Accumulator` : Pour garantir que la physique tourne exactement à 60H
     - `FixedUpdate` (Logic/Physics) : Rapier.js step(), AI decision ticks (distr
     - `LateUpdate` : Interpolation linéaire ou sphérique (SLERP) de la position/
     - `Render` : R3F render loop synchronisé avec le V-Sync.
   - OPTIMISATION WORKERS : Prépare l'architecture pour déporter Rapier (Physique
   - Utilise `SharedArrayBuffer` et `Atomics` pour la synchronisation ultra-rapic

4. SYSTÈME DE STREAMING (DYNAMIC GRID & LOD HYSTERESIS)
   - Code une logique de "Dynamic Grid System" avec Hystérésis (pour éviter le ch
```

```
   - Divise NeoCity en "Chunks" de 100x100m indexés spatialement (QuadTree ou Spc
   - Implémente un système de priorité de chargement asynchrone :
     - Zone Joueur (High Priority) : Chargement immédiat, physique active, IA hau
     - Zone Adjacente (Medium Priority) : Modèles chargés (LOD Low), physique sim
     - Zone Lointaine (Low Priority) : Uniquement les "Impostors" (sprites 2D bil
   - Gestionnaire de Ressources : `AssetManager` qui charge/décharge les textures

LIVRABLE :
Génère le code complet pour `GameEngine.tsx`, `ECSSetup.ts`, `LoopManager.ts`, `W
```

🌍 ÉTAPE 2 : WORLD DATA & NARRATIVE (Lore, Relations & Économie Profonde)

🎯 **Modèle cible :** `Gemini 3.0 Pro Thinking` **Objectif :** Générer une base de données relationnelle complexe pour un monde vivant, simulant une société crédible.

COPIER CE PROMPT :

````
Agis en tant que Creative Director, Narrative Designer, Sociologue et Data Scient
PROJET : "LIFE" (RPG Offline). Le monde doit sembler vivant, interconnecté et réc

TA MISSION :
Générer la "Bible" de NeoCity (JSON Data) avec des couches de complexité sociale,

1. BASE DE DONNÉES PNJ & RELATIONS (NPC_DB.json)
   - Génère 50 archétypes détaillés. Pour chaque archétype :
     - `identity` : id, name_pool, age_range, ethnicity_weights, background_story
     - `psychology` (Modèle OCEAN) : { openness: 0-1, conscientiousness: 0-1, ext
     - `moral_compass` : { lawful_chaotic: -1 à 1, good_evil: -1 à 1 }.
     - `routine_engine` : Au lieu d'actions simples, utilise des "Desire Paths" e
     - `relationships` : Définit une matrice de factions et de relations interper
     - `visuals` : Liste d'assets possibles (BodyID, HairID, ClothingID, Accessor

2. SYSTÈME DE DIALOGUES CONTEXTUELS & BRANCHING (DIALOGUES.json)
   - Crée un arbre de dialogue "State-Dependent" complexe.
   - Structure JSON supportant les conditions imbriquées :
     ```json
     {
       "trigger": "DEALER_INTERACT",
       "branches": [
         {
           "id": "intro_suspicious",
           "req": { "reputation": "LOW", "money": ">100", "is_police": false },
           "npc_text": "T'es qui toi ? Tu pues le bleu. Montre l'oseille d'abord.
           "responses": [
             { "text": "Montrer le cash (100$)", "action": "show_money", "next":
             { "text": "Intimider (Force)", "check": { "stat": "STR", "val": 5 },
             { "text": "Charmer (Persuasion)", "check": { "stat": "CHA", "val": 6
           ]
         }
       ]
     }
     ```
   - Couvre une vaste gamme d'interactions : Contrôles de Police (Corruption poss

3. ÉCONOMIE DYNAMIQUE & MARKET SIMULATION (ECONOMY.json)
````

```
   - Liste 200 objets (Consommables, Armes, Loot, Tech, Clés, Vêtements, Pièces A
   - Ajoute des métadonnées économiques avancées pour chaque item :
     - `base_price`, `min_price`, `max_price`.
     - `volatility` : Chance que le prix change chaque jour.
     - `elasticity` : Impact de l'offre/demande (Si le joueur inonde le marché de
     - `production_chain` : (Ex: Prix "Puce Électronique" affecte prix "Drone").
     - `illegal_flag` : Niveau de risque (0 = Légal, 5 = Terrorisme).
   - Définis des "World States" (Macros) qui impactent les prix globalement :
     - "PANDEMIC" (Masques & Médocs x10, Bars fermés, PNJ hostiles).
     - "TECH_BOOM" (Actions Cyberware x2, Immo +50%, Demande Énergie +).
     - "GANG_WAR" (Armes +30%, Police presence +200%, Zones bloquées).
     - "HYPER_INFLATION" (Tous les prix x5, PNJ pauvres agressifs).

LIVRABLE :
Produis 4 fichiers JSON massifs et valides (`npcs.json`, `factions_matrix.json`,
```

⚔️ ÉTAPE 3 : GAMEPLAY MECHANICS (Combat, Infiltration, Conduite & Parkour)

🎯 **Modèle cible :** `Claude 4.6 Opus Thinking` **Objectif :** Coder des mécaniques de jeu "Game Feel" satisfaisantes, réactives et émergentes (Systémique).

COPIER CE PROMPT :

```
Agis en tant que Senior Gameplay Programmer (ex-Rockstar, Naughty Dog ou Arkane).
TA MISSION : Coder les systèmes d'interaction, de combat, de parkour et de condui

1. CONTROLLER & STATE MACHINE JOUEUR (KINEMATIC CHARACTER CONTROLLER)
   - Code un `PlayerController.ts` robuste utilisant une State Machine hiérarchic
   - États :
     - `LOCOMOTION` : Walk/Run/Sprint avec accélération/décélération (inertie), i
     - `AIR` : Jump (variable height), Fall, Land (avec roulade si chute haute),
     - `PARKOUR` : Wall Run, Mantle (grimper rebord), Slide (glissade), Vault. Rc
     - `COMBAT` : Aiming (épaule/FPS), Shooting (Recoil procédural), Cover (Snap
     - `VEHICLE` : Driving/Passenger, Drive-by shooting.
   - Interaction Raycast "Smart Context" : Le bouton 'E' change selon la cible et

2. SYSTÈME DE POLICE & STEALTH (IA SENSORIELLE)
   - Ne fais pas juste un rayon de détection. Code un système de perception réali
     - `VisionCone` : Champ de vision (FOV) bloqué par les obstacles (Raycast che
     - `HearingSystem` : Le joueur émet des "NoiseEvents" (Sprint > Marche > Accr
     - `LightLevel` : Le joueur est plus visible sous les lampadaires, invisible
   - Système de Recherche (Wanted Level AI) :
     - Phase 1 : Investigation (Le flic va à la dernière position connue "LKP").
     - Phase 2 : Chasse (Vision directe, communication radio entre unités pour en
     - Phase 3 : Tactique (Utilisation de couvertures, Flanking, Grenades fumigèn
     - Phase 4 : Escalade (SWAT, Hélicoptère, Barrages routiers).

3. MÉCANIQUES VÉHICULES (RAPIER VEHICLE ADVANCED)
   - Implémente un contrôleur véhicule arcade/simu (Raycast Vehicle).
   - Paramètres physiques réglables par modèle : `enginePower`, `brakingForce`, `
   - Gestion des dégâts localisés :
     - Dégâts visuels (Mesh deformation vertex manipulation ou Normal Map blendir
     - Dégâts mécaniques : Roue crevée (friction change), Moteur touché (vitesse

4. SYSTÈME DE MÉTIERS & MISSIONS SYSTÉMIQUES
```

```
   - Code une classe abstraite `Mission` et des enfants `DeliveryMission`, `Assas
   - Hacking Minigame : Code la logique d'un mini-jeu de grille (HexDump, Pipe Mc
   - Générateur de missions procédurales : Assemble des objectifs (Aller à X, Tue

LIVRABLE :
Scripts TypeScript complets pour `PlayerController.ts`, `StealthSensors.ts`, `Veh
```

📱 ÉTAPE 4 : UI/UX LIQUID GLASS (Accessibilité, Design & OS)

🎯 **Modèle cible :** `Claude 4.6 Opus` **Objectif :** Interface réactive, animée, accessible et totalement intégrée à l'univers (Diégétique).

COPIER CE PROMPT :

```
Agis en tant que Lead UI/UX Designer et Front-End Developer Expert (React, Framer
CONTEXTE : Interface hybride : Hologrammes 3D (World Space) et Smartphone Overlay

TA MISSION :
Créer une UI React qui semble native, fluide (60fps animations, pas de layout trc

1. ARCHITECTURE UI & STATE MANAGEMENT
   - Utilise un Store dédié à l'UI (Zustand) séparé de la boucle physique pour év
   - Gestion des couches (Z-Layers) : `WorldUI` (Canvas), `HUD` (Overlay), `Phone
   - Input Manager : Support complet Clavier/Souris ET Manette (Navigation spatic

2. COMPOSANT SMARTPHONE (NEO-OS SIMULATOR)
   - Crée un composant `PhoneWrapper` simulant un OS complet.
   - Gestion des gestes : Swipe up (Home), Swipe back, Long press (Context menu).
   - Applications natives fonctionnelles :
     - **Bank** : Graphiques D3.js/Recharts temps réel des cryptos. Historique tr
     - **Contacts/Social** : Feed type Twitter/Insta généré procéduralement. Mess
     - **Maps** : Carte vectorielle ou Canvas 2D avec POI, itinéraire GPS, Zones
     - **Missions** : Kanban board (To Do, In Progress, Done). Détails, chronomèt
     - **Camera** : Utilise un `WebGLRenderTarget` pour prendre de vraies photos
     - **Music** : Player audio (Play, Pause, Skip) contrôlant la radio du jeu.

3. HUD IMMERSIF (DIEGETIC & AR)
   - Indicateurs 3D : Au lieu d'une flèche 2D, affiche une ligne holographique pr
   - Gestion des dégâts : Effet vignette rouge pulsante + aberration chromatique
   - Notifications : Système de "Toast" intelligent stackable. Priorisation des m
   - Réticule dynamique : Change de forme selon l'arme, s'écarte quand le joueur

4. ACCESSIBILITÉ & PARAMÈTRES (INCLUSIVITÉ)
   - Modes Daltoniens (Protanopia, Deuteranopia, Tritanopia) appliquant un filtre
   - Taille du texte dynamique (UI Scaling).
   - Option "Reduce Motion" pour désactiver les effets de flou/transparence/shake
   - Remapping complet des touches/boutons.

LIVRABLE :
Code React modulaire (`PhoneOS.tsx`, `Apps/Bank.tsx`, `DiegeticHUD.tsx`, `Setting
```

💾 ÉTAPE 5 : SAVE SYSTEM (Persistence, Sécurité & Cloud-Ready)

🎯 **Modèle cible :** `Gemini 3.0 Pro Low Thinking` **Objectif :** Système de sauvegarde transactionnel, versionné et résilient aux pannes.

COPIER CE PROMPT :

````
Agis en tant que Backend Engineer, Database Architect et Security Expert.
TA MISSION : Créer un système de persistance locale (Offline-first) ultra-robuste

1. SCHÉMA DE DONNÉES AVANCE & VALIDATION (ZOD)
   - Définis un schéma Zod exhaustif et typé strict :
     ```typescript
     type GameSave = {
       meta: {
         version: string, // SemVer "1.2.0"
         build_id: string,
         save_id: string, // UUID
         created_at: number,
         updated_at: number,
         playtime_seconds: number,
         screenshot_thumbnail?: stringBase64, // Low res
         checksum: string
       };
       player: {
         transform: { pos: Vec3, rot: Quat, chunkID: string, safe_location: boole
         status: { hp: number, hunger: number, stamina: number, effects: StatusEf
         inventory: InventorySlot[]; // {id, qty, durability, metadata: {color, m
         skills: Record<string, number>; // XP par compétence
         progression: { level: number, xp: number, karma: number };
       };
       world: {
         time_of_day: number,
         weather_state: string,
         global_flags: Record<string, boolean>; // "BOSS_KILLED", "BRIDGE_OPEN"
         quest_log: Record<string, QuestState>; // {status, steps_completed, time
         containers: Record<string, InventorySlot[]>; // Loot persistants modifié
         parked_vehicles: VehicleData[];
         economy_modifiers: EconomyState;
         npc_relations: Record<string, number>; // ID PNJ -> Valeur relation
       };
     }
     ```

2. SAVE MANAGER & STORAGE STRATEGY (TRANSACTIONNEL)
   - Utilise `idb-keyval` (IndexedDB Wrapper) pour le stockage principal (Large B
   - Implémente une sauvegarde atomique : Écrire dans un fichier temporaire -> Vc
   - Stratégie de compression : Utilise `LZ-String` ou `pako` (zlib) pour compres
   - Système de "Slots" : 3 Slots Manuels + 1 AutoSave (Rolling buffer) + 1 Quick
   - Gestion de Quota : Vérifier l'espace disponible avant sauvegarde.

3. MIGRATION DE VERSION (SCHEMA EVOLUTION)
   - C'est crucial pour la longévité du jeu. Code un pipeline de migration.
   - Si le jeu passe en v1.2 mais que la save est en v1.0 :
     - Détecter version.
     - Appliquer séquentiellement : `migrate_1_0_to_1_1(saveData)` -> `migrate_1_
     - Exemple : Si on ajoute une stat "Soif" en v1.1, le script doit l'injecter

4. SÉCURITÉ, INTEGRITÉ & ANTI-CHEAT
   - Hash & Sign : Calcule un Hash (SHA-256) du contenu JSON + "Salt" secret à lc
   - Au chargement, recalcule le hash. Si différent -> Afficher "Save Corrupted/M
````

```
   - Sanitization : Nettoyer les données chargées pour éviter l'injection de code

LIVRABLE :
Script `SaveSystem.ts`, `MigrationHandlers.ts`, `IntegrityCheck.ts` et `SchemaVal
```

🔊 ÉTAPE 6 : AUDIO SYSTEM (Immersion Spatiale & Générative)

🎯 **Modèle cible :** `Claude 4.6 Sonnet (Audio)` **Objectif :** Moteur audio 3D complet avec occlusion, propagation, musique procédurale et foley détaillé.

COPIER CE PROMPT :

```
Agis en tant que Lead Audio Programmer et Sound Designer (Expert Web Audio API).
TA MISSION : Créer un paysage sonore riche, dynamique et réaliste. Le joueur doit

1. MOTEUR AUDIO SPATIAL AVANCE (GRAPH NODE SYSTEM)
   - Wrapper robuste autour de Web Audio API (ou Howler.js étendu).
   - Gestion des nœuds audio : `Source` -> `Panner3D` -> `OcclusionGain` -> `Reve
   - Modèle de spatialisation : Utilise HRTF (Head-Related Transfer Function) pou
   - Gestion des entités sonores :
     - `SoundEmitter3D` : Attaché à un Mesh/Bone. Interpolation de la position pc
     - `Listener` : Attaché à la caméra (ou à la tête du personnage pour plus de

2. OCCLUSION, DIFFRACTION & ENVIRONNEMENT (PHYSICAL AUDIO)
   - Système d'occlusion Raycast temps réel (Optimisé : 1 check toutes les 100ms
     - Si obstacle (Mur) détecté -> Appliquer `BiquadFilterNode` (LowPass ~400Hz)
   - Reverb Zones (ConvolverNode) : Définis des volumes (Box/Sphere) dans le monc
     - `Tunnel` : Reverb longue, métallique.
     - `OpenField` : Pas de reverb, léger delay (Echo).
     - `SmallRoom` : Reverb courte, dense.
     - Transition fluide (Crossfade) des Impulse Responses quand on passe d'une z
   - Effet Doppler : Moduler le `detune` en fonction de la vitesse relative Sourc

3. MUSIQUE DYNAMIQUE & PROCÉDURALE (VERTICAL LAYERING)
   - Système de musique interactif basé sur l'intensité ("Intensity Level").
   - Utilise des "Stems" (Pistes séparées) parfaitement synchronisées (Sample-acc
   - États musicaux :
     - `EXPLORE` (Intensité 0-20%) : Pad Synth + Ambient Texture.
     - `TENSION` (Intensité 20-50%, ex: Zone Gang) : Ajout Bassline pulsante + Hi
     - `ACTION` (Intensité 50-80%, ex: Combat) : Ajout Drums complets + Lead Synt
     - `CLIMAX` (Intensité 80-100%) : Distortion + Orchestral hits.
   - Transitions : Filtres passe-bas automatiques et Crossfades sur les mesures (

4. FOLEY & FEEDBACK SONORE (PROCEDURAL SFX)
   - Système de pas (Footsteps) complexe :
     - Détecte le matériau au sol (Raycast vers le bas : Concrete, Grass, Metal,
     - Choisit aléatoirement parmi 5 samples par matériau (Pitch/Volume variation
   - Audio dynamique véhicule :
     - Synthèse granulaire ou Pitch-shifting simple du moteur basé sur `RPM`.
     - Bruit de vent (White Noise filtré) augmentant avec la vitesse.
     - Bruit de pneus (Skid) lors des dérapages.
   - UI Sounds : Sons satisfaisants, courts et non-intrusifs pour chaque interact

LIVRABLE :
```

Classe `AudioEngine.ts`, `MusicController.ts`, `OcclusionSystem.ts` et `FoleyManc