# Исчерпывающий Дизайн-Документ (GDD): The Cursed Warden

## Архитектура, Игровые Системы и Нарративная Логика в Среде Bevy Engine

### 1. Введение и Концептуальный Обзор

#### 1.1. Видение Проекта

Проект **"The Cursed Warden" (Проклятый Страж)** представляет собой гибридную систему, объединяющую жанры Narrative Roguelite, Auto-Battler и Inventory Management. Фундаментальная идея игры строится на дихотомии между *Порядком* (управление инвентарем, строгая сетка, тайм-менеджмент) и *Хаосом* (мутации предметов, непредсказуемость боя, заражение). Игрок берет на себя роль Стража — последнего бастиона порядка в деревне XIX века, охваченной эпидемией элдрического безумия.

Ключевой уникальной механикой (USP) является **Системная Коррозия Инвентаря**. В отличие от традиционных игр, где инвентарь является статичным хранилищем, в "The Cursed Warden" это живая экосистема. Предметы, находящиеся в инвентаре, подвергаются воздействию заражения, физически изменяя свою форму, свойства и занимаемое пространство (Inventory Tetris).1 Это создает постоянное напряжение: эффективное мутировавшее оружие требует жертв в виде свободного места и репутации героя.

#### 1.2. Технологический Фундамент: Bevy ECS

Выбор движка Bevy (Rust) обусловлен необходимостью строгой типизации данных и высокой производительности Entity Component System (ECS) для реализации детерминированной симуляции боя и сложной логики инвентаря.3 Архитектура проекта должна строго следовать принципам Data-Oriented Design (DOD), разделяя данные (Components) и логику (Systems), что критически важно для масштабируемости проекта и реализации 32,000+ сущностей при необходимости.5

Цель данного документа — предоставить исчерпывающую техническую и дизайнерскую спецификацию, достаточную для генерации кода игры, с детальным описанием структур данных, алгоритмов и нарративных триггеров.

### 2. Техническая Архитектура и Организация Проекта в Bevy

Для обеспечения поддерживаемости кода и быстрой компиляции, архитектура проекта должна быть модульной. Монолитный подход в Rust недопустим из-за времени компиляции и сложности отладки.

#### 2.1. Структура Плагинов (Plugin Hierarchy)

Приложение Bevy должно быть организовано как совокупность доменных плагинов. Каждый плагин инкапсулирует свои ресурсы, компоненты и системы, регистрируя их в общем планировщике App.6

| **Название Плагина** | **Описание и Ответственность** |
| --- | --- |
| **CorePlugin** | Инициализация окна, загрузка ассетов (AssetServer), глобальные настройки (RNG, Audio), обработка состояний игры (GameState). |
| **GridInventoryPlugin** | Логика "Тетриса", структуры данных сетки (HashMap<IVec2, Entity>), алгоритмы коллизий, Drag-and-Drop события. |
| **CombatSimulationPlugin** | Детерминированная логика авто-битвы, расчет урона, AI стейт-машины, обработка тиков времени (FixedUpdate). |
| **MetagamePlugin** | Системы тайм-менеджмента (День), экономика (Талеры), глобальные переменные (Репутация, Инфекция). |
| **NarrativePlugin** | Парсинг диалогов, дерево событий, условия достижения 5 концовок. |
| **UiPlugin** | Визуализация интерфейса, CSS Grid Layout, обработка ввода пользователя (bevy\_picking). |

#### 2.2. Управление Состояниями (State Management)

Игровой цикл строго разделен на фазы. Использование паттерна States в Bevy позволяет активировать и деактивировать целые группы систем, предотвращая выполнение логики боя во время фазы инвентаря.8

Рекомендуемая структура перечисления состояний:

Rust

#  
pub enum GameState {  
 #[default]  
 AssetLoading,  
 MainMenu,  
 DayPhase(DaySubState), // Вложенные состояния для детализации  
 EveningPhase, // Менеджмент инвентаря  
 NightPhase, // Авто-битва  
 EventResolution, // Диалоговые окна  
 GameOver,  
}  
  
#  
pub enum DaySubState {  
 #[default]  
 Idle,  
 Trading,  
 MapTravel,  
}

Системы должны регистрироваться с условиями выполнения (run\_if), чтобы гарантировать изоляцию логики:

* add\_systems(Update, combat\_tick\_system.run\_if(in\_state(GameState::NightPhase)))
* add\_systems(Update, inventory\_drag\_system.run\_if(in\_state(GameState::EveningPhase)))

#### 2.3. Синхронизация Данных и Визуализации

Критически важным аспектом является разделение *логического* представления сущности и её *визуального* отображения.1

* **Логическая позиция:** Компонент GridPosition(IVec2) хранит координаты предмета в сетке (например, x: 2, y: 3). Это "истина" для симуляции.
* **Визуальная позиция:** Компонент Transform (из bevy\_transform) определяет положение спрайта на экране.
* **Система Синхронизации:** Система sync\_grid\_transform запускается каждый кадр в фазе инвентаря, интерполируя Transform к целевым координатам, рассчитанным из GridPosition. Это позволяет реализовать плавные анимации перемещения без влияния на логику коллизий.

### 3. Механика "Inventory Tetris": Алгоритмы и Реализация

Инвентарь в "The Cursed Warden" — это поле битвы с энтропией. Игрок должен размещать предметы разной формы в ограниченной сетке.

#### 3.1. Структура Данных Сетки

Для эффективного пространственного запроса (Spatial Querying) в Bevy, массив сущностей не подходит. Оптимальным решением является использование HashMap или одномерного вектора, проецируемого на 2D координаты, для O(1) доступа к ячейкам.1

**Компонентная База Предмета:**

* ItemSize: Компонент, хранящий ширину и высоту (width: u8, height: u8).
* GridPosition: Координаты верхнего левого угла предмета.
* ItemShape (опционально): Список смещений для непрямоугольных предметов (L-образные, T-образные), что критично для "мутировавших" предметов неправильной формы.

**Ресурс Сетки:**

Rust

#  
pub struct InventoryGridState {  
 pub cells: HashMap<IVec2, Entity>, // Карта занятых ячеек -> ID сущности  
 pub width: i32,  
 pub height: i32,  
}

#### 3.2. Алгоритм Валидации Размещения

При попытке переместить предмет (Drag-and-Drop), система должна проверить валидность новой позиции.

1. **Проверка Границ:** Для каждой ячейки, занимаемой предметом (с учетом ItemSize и ItemShape), координаты (x, y) должны находиться в пределах 0..width и 0..height.
2. **Проверка Коллизий:** Для каждой вычисляемой ячейки проверяется наличие записи в InventoryGridState. Если запись существует и ID сущности не совпадает с перемещаемым предметом, размещение запрещено.
3. **Обработка "Выталкивания" (Shove Mechanic):** Для улучшения UX (User Experience) можно реализовать алгоритм, который пытается сдвинуть мешающие предметы. Это рекурсивная функция: если в целевых ячейках есть другие предметы, система проверяет, есть ли свободное место для *них* в направлении сдвига. Если цепочка сдвигов возможна, происходит массовое обновление GridPosition.2

#### 3.3. Взаимодействие и UI (Bevy Picking)

Использование крейта bevy\_picking позволяет реализовать сложную логику Drag-and-Drop через систему Наблюдателей (Observers).12

* **События:**
  + On<Pointer<DragStart>>: Сохраняет начальную позицию предмета (для отката при неудачном сбросе) и изменяет ZIndex, чтобы предмет рендерился поверх остальных.
  + On<Pointer<Drag>>: Обновляет визуальный Transform предмета, следуя за курсором мыши.
  + On<Pointer<DragDrop>>: Самый сложный этап. При отпускании кнопки мыши производится Raycast (луч) в мир UI. Если луч попадает в сущность InventorySlot, система вычисляет координаты слота.14
* **Контекст:** Важно учитывать, что координаты мыши (Screen Space) должны быть корректно преобразованы в координаты UI Node (Ui Space), особенно если интерфейс масштабируется или камера перемещается.15

#### 3.4. Механика Мутации (The Mutation)

Каждый вечер, перед переходом к ночной фазе, запускается система InventoryMutationSystem.

* **Триггер:** Уровень глобальной переменной GlobalInfection (Заражение).
* **Логика:** Система итерирует по всем предметам в инвентаре. Для каждого предмета рассчитывается вероятность мутации: $P\_{mut} = \text{BaseChance} + (\text{InfectionLevel} \times 0.5)$.
* **Эффект:**
  + **Изменение Формы:** Предмет размером 1x2 может стать 2x2, "обрастая" плотью. Если для роста нет места, предмет может поглотить (уничтожить) соседний предмет (приоритет уничтожения: расходники -> патроны -> дешевое оружие).
  + **Смена Материала:** Компонент MaterialType::Steel заменяется на MaterialType::Flesh. Это меняет его боевые характеристики (см. раздел Боевая Система).
  + **Визуализация:** Смена спрайта на процедурно-сгенерированный или выбранный из пула "искаженных" ассетов.

### 4. Ночная Фаза: Детерминированный Авто-Баттлер

Бой происходит автоматически. Игрок — стратег, а не тактик. Он подготавливает "сцену" днем и вечером, а ночью наблюдает за симуляцией.

#### 4.1. Детерминизм Симуляции

Для того чтобы бой можно было воспроизвести (например, для реплея или отладки), он должен быть строго детерминированным.4

* **Фиксированный Шаг Времени:** Использование FixedTime (например, 64 тика в секунду) для обновления логики боя. Нельзя использовать Time::delta\_seconds(), так как плавающий шаг может привести к расхождениям на разных машинах.
* **Собственный RNG:** Не использовать глобальный генератор случайных чисел. Каждая битва инициализируется с определенным Seed (зерном), и используется локальный экземпляр RNG (ChaCha8 или PCG), передаваемый как Resource или компонент в системы.6

#### 4.2. Система Тиков и Инициативы (Action Points)

Бой не пошаговый в классическом понимании, а тиковый (Real-Time with Pause логика).

* У каждого юнита есть компонент Speed (Скорость) и ActionMeters (Накопитель действия).
* Каждый тик: ActionMeter += Speed.
* Когда ActionMeter >= Threshold (порог, например, 1000), юнит совершает действие и ActionMeter обнуляется.17
* **Влияние Инвентаря:** Вес предметов в инвентаре (сумма компонентов Weight) накладывает штраф на Speed Стража. Перегруженный инвентарь означает, что враги будут атаковать чаще.

#### 4.3. Математика Урона и Материалов (Silver vs. Steel)

Система урона базируется на принципе "Камень-Ножницы-Бумага", реализованном через типы материалов.18

**Типы Материалов:**

1. **Сталь (Steel):** Эффективна против Людей, Культистов. Слаба против Чудовищ. Высокая прочность.
2. **Серебро (Silver):** Эффективно против Чудовищ, Призраков. Слабо против Людей (низкий базовый урон). Низкая прочность.
3. **Плоть (Flesh):** Мутировавшее оружие. Эффективно против всего живого, наносит урон рассудку (Sanity) владельца. Восстанавливает прочность за счет крови врагов.

Формула Расчета Урона:

Чтобы избежать проблемы "нулевого урона" при высокой защите и обеспечить масштабируемость 19, используется гибридная формула:

$$\text{RawDamage} = \text{WeaponDamage} \times \text{MaterialModifier}$$

$$\text{FinalDamage} = \begin{cases} 2 \cdot \text{RawDamage} - \text{Defense} & \text{если } \text{RawDamage} \ge \text{Defense} \\ \frac{\text{RawDamage}^2}{\text{Defense}} & \text{если } \text{RawDamage} < \text{Defense} \end{cases}$$

*Пояснение:* Если атака пробивает защиту, урон линейно зависит от атаки. Если защита слишком высока, урон не становится нулевым, а уменьшается по экспоненте, всегда оставляя шанс нанести минимум 1 единицу урона ("chip damage").

**MaterialModifier (Матрица Эффективности):**

| **Атакующий / Цель** | **Человек (Human)** | **Чудовище (Monster)** | **Призрак (Ethereal)** |
| --- | --- | --- | --- |
| **Сталь** | 1.5x | 0.8x | 0.0x |
| **Серебро** | 0.7x | 2.0x | 3.0x |
| **Плоть** | 1.2x | 1.2x | 0.5x |

### 5. Дневная Фаза: Тайм-Менеджмент и Экономика

Дневная фаза представляет собой цикл управления ресурсами, где время является валютой.

#### 5.1. Ресурс Времени и Граф Локаций

Игровой день длится условные 12 часов (06:00 - 18:00).

* Перемещение между узлами графа деревни (Церковь, Рынок, Кладбище, Поместье) стоит 1-2 часа.22
* Действия в локациях (Торговля, Лечение, Молитва) также расходуют часы.
* **Стратегический выбор:** Игрок не может посетить все локации за один день. Ему приходится выбирать: вылечить раны (4 часа) или купить серебряные пули у кузнеца (2 часа + время на дорогу).

#### 5.2. Экономическая Модель: Талеры и Репутация

В игре присутствуют две параллельные валюты, которые часто находятся в конфликте.23

1. **Талеры (Золото):** Материальный ресурс. Необходим для покупки расходников и починки снаряжения. Источник: Лут с врагов, продажа трофеев.
2. **Репутация:** Социальный ресурс. Необходим для доступа к услугам (например, Кузнец не будет чинить оружие изгою) и влияет на концовку. Источник: Выполнение просьб жителей, отказ от использования "Проклятых" предметов на людях.

Кривая Баланса:

Экономика настроена на дефицит. Починка серебряного оружия стоит дорого. Это заставляет игрока иногда использовать мутировавшее оружие ("Плоть"), которое чинится само, но снижает Репутацию. Это создает петлю положительной обратной связи (Positive Feedback Loop) в сторону "плохой" концовки, из которой игроку нужно активно выбираться.22

### 6. Нарративная Структура: 5 Концовок

Нарратив в игре процедурный, зависящий от двух скрытых глобальных переменных: InfectionValue (0-100) и ReputationValue (0-100).

#### 6.1. Матрица Концовок

Логика определения концовки проверяется в финале 7-го дня (или после финального босса).

| **Название Концовки** | **Условие (Infection)** | **Условие (Reputation)** | **Описание** |
| --- | --- | --- | --- |
| **1. Святой Мученик** | < 20 (Низкая) | > 80 (Высокая) | Страж спасает деревню ценой жизни, оставаясь человеком. Его канонизируют. |
| **2. Изгнанный Герой** | 20 - 60 (Средняя) | 40 - 80 (Средняя) | Деревня спасена, но жители боятся мутаций Стража. Его изгоняют в леса. |
| **3. Новый Кошмар** | > 80 (Высокая) | < 20 (Низкая) | Страж полностью теряет человечность, убивает босса и сам становится новым источником проклятия. |
| **4. Тиран** | 20 - 50 (Средняя) | < 20 (Низкая) | Страж использует силу для захвата власти над деревней, управляя ею железной рукой. |
| **5. Элдрическое Вознесение** | = 100 (Максимум) | Любая | Секретная концовка. Требует сбора специальных артефактов ("Глаза Бездны"). Страж transcends реальность, уничтожая и деревню, и монстров. |

#### 6.2. Контекстное Повествование (Flavor Text)

Вместо кат-сцен используются текстовые описания предметов, которые меняются в зависимости от уровня Инфекции игрока (Sanity Effects).25

* *Низкая Инфекция:* "Старый ржавый меч. Надежный."
* *Средняя Инфекция:* "Меч кажется теплым на ощупь. Он вибрирует."
* *Высокая Инфекция:* "Мой брат. Он хочет пить. Я должен напоить его кровью."

### 7. План Реализации (Roadmap) и Инструкции по Кодогенерации

Для генерации кода следуйте этому поэтапному плану. Каждый этап представляет собой изолированный модуль.

#### Этап 1: Базовая Сетка и Инвентарь

1. Создать InventoryPlugin.
2. Определить компоненты InventorySlot, Item, GridPosition.
3. Реализовать ресурс GridState с HashMap.
4. Написать систему spawn\_inventory\_ui используя Node с Display::Grid.27
5. Реализовать Drag-and-Drop через bevy\_picking и Observer-ы для событий DragStart, Drag, DragDrop.28

#### Этап 2: Генерация Предметов и Мутации

1. Создать базу данных предметов (Resource ItemDatabase загружаемый из.ron файлов).
2. Реализовать систему resize\_item\_system, которая меняет компонент ItemSize и обновляет спрайт.
3. Написать логику check\_overlap, предотвращающую наложение предметов при генерации и мутации.

#### Этап 3: Боевая Система

1. Создать CombatPlugin.
2. Определить компоненты Health, Attack, Defense, Speed, ActionMeter, Material.
3. Реализовать систему tick\_timer\_system на FixedUpdate.
4. Имплементировать формулу урона с учетом матрицы материалов.

#### Этап 4: Мета-игра и Сохранения

1. Интегрировать serde для сериализации компонентов.
2. Создать SaveState структуру, хранящую состояние сетки, статы игрока и текущее время.
3. Реализовать переходы между сценами (OnEnter(GameState::DayPhase)).

### 8. Заключение

"The Cursed Warden" — это проект, требующий высокой дисциплины в управлении данными. Использование Bevy ECS позволяет создать гибкую систему, где поведение предметов (мутации) и исход битв (авто-баттлер) эмерджентно возникают из взаимодействия базовых компонентов. Основной вызов при разработке — баланс между стратегическим планированием инвентаря ("Тетрис") и неизбежностью потерь в бою ("Roguelite"), скрепленный мрачным нарративом о цене силы.

*Документ подготовлен для использования в качестве мастер-спецификации при разработке на Rust/Bevy.*

#### Источники

1. Grid based game Bevy : r/rust\_gamedev - Reddit, дата последнего обращения: января 3, 2026, <https://www.reddit.com/r/rust_gamedev/comments/1apql1o/grid_based_game_bevy/>
2. SinlessDevil/TetrisInventorySystem: Inventory Tetris + UI MVVM A grid-based inventory system inspired by games like Escape from Tarkov and Diablo. - GitHub, дата последнего обращения: января 3, 2026, <https://github.com/SinlessDevil/TetrisInventorySystem>
3. Bevy 0.16 - Bevy Engine, дата последнего обращения: января 3, 2026, <https://bevy.org/news/bevy-0-16/>
4. Bevy's Gift - Alex Bagnall, дата последнего обращения: января 3, 2026, <https://bagnalla.github.io/posts/newton_bevy.html>
5. ECS - Bevy Engine, дата последнего обращения: января 3, 2026, <https://bevy.org/learn/quick-start/getting-started/ecs/>
6. tbillington/bevy\_best\_practices: An opionated set of Best Practices for the Bevy game engine - GitHub, дата последнего обращения: января 3, 2026, <https://github.com/tbillington/bevy_best_practices>
7. Plugins and managing states : r/bevy - Reddit, дата последнего обращения: января 3, 2026, <https://www.reddit.com/r/bevy/comments/1glt0mp/plugins_and_managing_states/>
8. Bevy Best Practices - trent.kiwi, дата последнего обращения: января 3, 2026, <https://trent.kiwi/bevy-best-practices>
9. Bevy roguelike tutorial / devlog part 3 - Action queue - MyBlog, дата последнего обращения: января 3, 2026, <https://maciejglowka.com/blog/2023/bevy-roguelike-tutorial-devlog-part-3-action-queue/>
10. Bevy Archetypes | Tainted Coders, дата последнего обращения: января 3, 2026, <https://taintedcoders.com/bevy/archetypes>
11. Steam Workshop::Inventory Tetris - Grid Based Inventory Overhaul [B42] - Steam Community, дата последнего обращения: января 3, 2026, <https://steamcommunity.com/sharedfiles/filedetails/?id=3397561666>
12. Bevy Picking | Tainted Coders, дата последнего обращения: января 3, 2026, <https://taintedcoders.com/bevy/picking>
13. bevy\_picking - Rust - Docs.rs, дата последнего обращения: января 3, 2026, <https://docs.rs/bevy_picking>
14. Hovering, Dragging and Picking: Deficiencies with the current framework · bevyengine bevy · Discussion #19402 - GitHub, дата последнего обращения: января 3, 2026, <https://github.com/bevyengine/bevy/discussions/19402>
15. How to position UI in bevy? - Reddit, дата последнего обращения: января 3, 2026, <https://www.reddit.com/r/bevy/comments/1lwhmqx/how_to_position_ui_in_bevy/>
16. Avian 0.4: ECS-Driven Physics for Bevy : r/rust - Reddit, дата последнего обращения: января 3, 2026, <https://www.reddit.com/r/rust/comments/1o5hsbi/avian_04_ecsdriven_physics_for_bevy/>
17. What are some good ways of balancing an action-point based action economy? - Reddit, дата последнего обращения: января 3, 2026, <https://www.reddit.com/r/RPGdesign/comments/1cj666z/what_are_some_good_ways_of_balancing_an/>
18. Elemental Rock-Paper-Scissors | Tropedia - Fandom, дата последнего обращения: января 3, 2026, <https://tropedia.fandom.com/wiki/Elemental_Rock-Paper-Scissors>
19. The Simplest Non-Problematic Damage Formula | Tung's Word Box, дата последнего обращения: января 3, 2026, <https://tung.github.io/posts/simplest-non-problematic-damage-formula/>
20. Why are silver weapons more powerful than steel weapons? : r/fireemblem - Reddit, дата последнего обращения: января 3, 2026, <https://www.reddit.com/r/fireemblem/comments/750tcr/why_are_silver_weapons_more_powerful_than_steel/>
21. Need help with combat formula for game : r/MathHelp - Reddit, дата последнего обращения: января 3, 2026, <https://www.reddit.com/r/MathHelp/comments/1pwzc94/need_help_with_combat_formula_for_game/>
22. Game Economy Design - Playtank, дата последнего обращения: января 3, 2026, <https://playtank.io/2025/08/12/game-economy-design/>
23. Game Economy Design: Understanding the Art of Balance - 300Mind, дата последнего обращения: января 3, 2026, <https://300mind.studio/blog/what-is-game-economy-design/>
24. Money Tropes - The Arcana Wiki, дата последнего обращения: января 3, 2026, <http://arcana.wikidot.com/money-tropes>
25. Grimdark Fantasy: What Is It & How to Write It - Campfire, дата последнего обращения: января 3, 2026, <https://www.campfirewriting.com/learn/grimdark-fantasy>
26. Grimdark JRPGs for fans of Fear & Hunger - Dark RPGs, дата последнего обращения: января 3, 2026, <https://darkrpgs.home.blog/2025/11/23/grimdark-jrpgs-for-fans-of-fear-hunger/>
27. CSS Grid - Bevy Engine, дата последнего обращения: января 3, 2026, <https://bevy.org/examples/ui-user-interface/grid/>
28. DragDrop in bevy::picking::prelude - Rust - Docs.rs, дата последнего обращения: января 3, 2026, <https://docs.rs/bevy/latest/bevy/picking/prelude/struct.DragDrop.html>